ANADOLU ÜNİVERSİTESİ

anadolum
eKampüs
ve
anadolu mobil

dilediğin yerden,
dilediğin zaman,
öğrenme fırsatı!

(ekampus.anadolu.edu.tr)

(mobil.anadolu.edu.tr)

ekampus.anadolu.edu.tr

- Takvim
- Duyurular
- Ders Kitabı (PDF)
- Epub
- Html5
- Mobi Kitap
- Sesli Kitap
- Canlı Ders
- Video
- Ünite Özeti
- Sesli Özet
- Sorularla Öğrenelim
- Alıştırma
- Çözümlü Sorular
- Deneme Sınavı
- Tartışma Forumu
- Çıkmış Sınav Soruları
- Sınav Giriş Bilgisi
- Sınav Sonuçları
- Öğrenci Topluluklar

AOSDESTEK
AÇIKÖĞRETİM DESTEK SİSTEMİ

Açıköğretim Sistemi ile ilgili merak ettiğiniz her şey AOS Destek Sisteminde...

- Kolay Soru Sorma ve Soru-Yanıt Takibi
- Sıkça Sorulan Sorular ve Yanıtları
- Canlı Destek (Hafta İçi Her Gün)
- Telefonla Destek

aosdestek.anadolu.edu.tr

AOS DESTEK Sistemi İletişim ve Çözüm Masası

0850 200 46 10

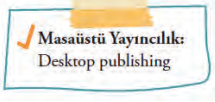
www.anadolu.edu.tr

/AOFAnadolum /Anadolu_Univ instagram.com/anadoluuniv

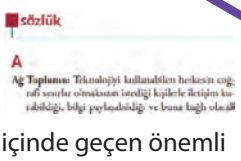
AÖF Kitapları Öğrenci Kullanım Kılavuzu

Öğrenme çıktıları
Bölüm içinde hangi bilgi, beceri ve yeterlikleri kazanacağınızı ifade eder.

Bölüm Özeti
Bölümün kısa özetini gösterir.



Tanım
Bölüm içinde geçen önemli kavramların tanımları verilir.



Sözlük
Bölüm içinde geçen önemli kavramlardan oluşan sözlük ünite sonunda paylaşılır.

Dikkat
Konuya ilişkin önemli uyarıları gösterir.

Neler Öğrendik ve Yanıt Anahtarı
Bölüm içeriğine ilişkin 10 adet çoktan seçmeli soru ve cevapları paylaşılır.

Karekod
Bölüm içinde verilen karekodlar, mobil cihazlarınız aracılığıyla sizi ek kaynaklara, videolara veya web adreslerine ulaştırır.

Öğrenme Çıktısı Tablosu

Araştır/İlişkilendir/Anlat-Paylaş
İlgili konuların altında cevaplayacağınız soruları, okuyabileceğiniz ek kaynakları ve konuyla ilgili yapabileceğiniz ekstra etkinlikleri gösterir.

Yaşamla İlişkilendir
Bölümün içeriğine uygun paylaşılan yaşama dair gerçek kesitler veya örnekleri gösterir.

Araştırmalarla İlişkilendir
Bölüm içeriği ile ilişkili araştırmaların ve bilimsel çalışmaları gösterir.

ANADOLU ÜNİVERSİTESİ

İnternet Tabanlı Programlama

Editör

Doç.Dr. Mehmet FIRAT

Yazarlar

BÖLÜM 1 — Doç.Dr. Işıl KABAKÇI YURDAKUL, Dr.Öğr.Üyesi Fatih YAMAN

BÖLÜM 2 — Doç.Dr. Yusuf Levent ŞAHİN

BÖLÜM 3, 6 — Dr.Öğr.Üyesi Serkan ÇANKAYA

BÖLÜM 4, 5 — Dr.Öğr.Üyesi Onur DÖNMEZ

BÖLÜM 7 — Doç.Dr. Tarık KIŞLA

BÖLÜM 8 — Sıtkı BAĞDAT

T.C. ANADOLU ÜNİVERSİTESİ YAYINI NO: 3607
AÇIKÖĞRETİM FAKÜLTESİ YAYINI NO: 2436

Bu kitabın basım, yayım ve satış hakları Anadolu Üniversitesine aittir.
"Uzaktan Öğretim" tekniğine uygun olarak hazırlanan bu kitabın bütün hakları saklıdır.
İlgili kuruluştan izin almadan kitabın tümü ya da bölümleri mekanik, elektronik, fotokopi, manyetik kayıt veya başka şekillerde çoğaltılamaz, basılamaz ve dağıtılamaz.

Copyright © 2017 by Anadolu University
All rights reserved
No part of this book may be reproduced or stored in a retrieval system, or transmitted in any form or by any means mechanical, electronic, photocopy, magnetic tape or otherwise, without permission in writing from the University.

Öğretim Tasarımcısı
Dr.Öğr.Üyesi İlker Usta

Grafik Tasarım ve Kapak Düzeni
Prof.Dr. Halit Turgay Ünalan

Grafikerler
Gülşah Karabulut
Hilal Özcan

Dizgi ve Yayıma Hazırlama
Selin Çakır
Murat Uzun
Gül Akar
Sinem Yüksel
Gülşah Sokum
Burak Arslan

İNTERNET TABANLI PROGRAMLAMA

E-ISBN
978-975-06-3220-4

Bu kitabın tüm hakları Anadolu Üniversitesi'ne aittir.

ESKİŞEHİR, Ocak 2019

3144-0-0-0-2002-V01

İçindekiler

BÖLÜM 1 İnternet Tabanlı Programlamanın Temelleri

Giriş 3
İnternet Tabanlı Programlama Nedir? 3
İnternet Tabanlı Programlamanın Tarihsel ve Yapısal Gelişimi 5
İnternet Tabanlı Programlama Temel Kavramları 8
- Web Tarayıcı 8
- WWW 9
- Hosting ve Domain 9
- Statik ve Dinamik Web Sitesi 9
- HTML ve HTML Etiketleri 10
- CSS (Cascading Style Sheets) 12
- Veri Tabanı 12
- Protokol 12
- Proxy 12

İnternet Tabanlı Programlama Dilleri 13
- Betik Diller 13
- İstemci-Sunucu Taraflı Programlama 14
- Nesne Tabanlı Programlama 15

Dinamik Web 17
- Model View Controller (MVC) 17
- Model Template View (MTV) 18

BÖLÜM 2 PHP'ye Giriş

Giriş 27
Sunucu Taraflı Programlama Dilleri 27
PHP Destekli Web Sunucu Kurulumu 29
PHP'nin Temel Kodlama Özellikleri 34
- İlk PHP Sayfası: Merhaba Dünya! 34
- PHP'de Değişken Kullanımı 36
- PHP'de Sabit Kullanımı 36
- PHP Kodlarına Yorum Satırlarının Eklenmesi 37

PHP'nin Yapılandırılması 38

BÖLÜM 3 PHP ile Dinamik Web Kodlama

Giriş 47
Değişkenler ve Değişken Türleri 47
Diziler 47
HTML Formları 50
Kontrol Yapıları 51
If 51
Karşılaştırma İfadeleri 53
For 55
While 57
Do while 57
Foreach 58
Break 58
Continue 59
Switch 60
Goto 61
include/include_once 61
require/require_once 62
Fonksiyonlar 63
Kapsam 64
Parametre Gönderimi 65
Varsayılan Parametre Değerleri 65
Sınırsız Sayıda Parametre 66
Sınıflar 67
Veritabanı İşlemleri 68
Çatılar ve Laravel 74

BÖLÜM 4 JavaScript

Giriş 87
JavaScript'in İşlevleri 88
Bir HTML Elementinin İçeriğini Değiştirmek 88
Bir Elementin Niteliklerini Değiştirmek 89
Sayfalara Element Eklemek ve Silmek ... 89
Bir Elementin Görünümünü Değiştirmek 89
Kullanıcıdan Onay Almak 90
Zamanlanmış Görev Kullanmak 90
JavaScript Nereye Yazılır? 91
Yazım Kuralları (Sentaks) 93
Değerler 93
Değişkenler 94
İşleçler 95
Kıyaslama İşleçleri ve Mantıksal İşleçler 97
Dom Nesnelerine Erişmek 99
document.getElementById() 99
document.getElementsByTagName() ... 100
document.getElementsByClassName() ... 101
Çıktı Mekanizmaları 101
innerHTML 101
document.write() 102
appendChild() 102
console.log() 103
window.alert() 103
window.confirm() 103
Veri Tipleri 104
String 105
Number 105
Boolean 105
Array 105
Object 108
Koşul Yapıları 109
if 109
else 110
switch 111
Döngüler 112
while 112
do/while 113
for 114
Fonksiyonlar 115
Değişkenlerin Etki Alanı 117
Olaylar 118
Tarih Nesnesi 119
Zamanlanmış Görevler 120

BÖLÜM 5 JS Kütüphaneleri ile Çalışmak

Giriş 131
Neden JS Kütüphaneleri 131
Tekrar Kullanılabilirlik 131
Önemli JS Kütüphaneleri 132
AngularJS 132
React 133
Underscore 133
D3.js 133
BabylonJS 133
Bootstrap 133
JQuery 134
Sayfalara JQuery Eklemek 135
JQuery Yazım Kuralları 136
JQuery Olayları 136
JQuery Efektleri 138
JQuery DOM Metotları 139
JQuery CSS Metotları 142
JQuery Mobile 145
Sayfalara JQM Eklemek 145
JQM Sayfaları 147
JQM Kutuları 151
JQM Geçiş Efektleri 154
JQM Arayüz Araçları 155

BÖLÜM 6 ASP.NET ile Dinamik Web Kodlama

Giriş 173
ASP.NET 173
Kurulum 175
C# Programlama Dili 177
Değişkenler ve Türler 178
Tür Dönüşümleri 180
Diziler 181
ArrayList 182
Operatörler 183
If 184
For 185
While 187
Do While 187
Foreach 187
Break ve Continue 188
Switch 189
Sınıflar 190
Web Formları 193
Veritabanı İşlemleri 199

BÖLÜM 7 Python ve Django

Giriş 213
Python Programlama Dili 214
Python Kurulumu 214
Python ile Çalışma 214
Python Programlama Dilinde Değişkenler 216
Python Veri Tipleri 216
Python Programlama Dilinde Operatörler 219
Python Programlama Dilinde Hazır Fonksiyonlar 219
Python Program Denetimi 222
Döngü Yapıları: for ifadesi 223
range() Fonksiyonu 223
Koşullu Döngü Yapıları: while ifadesi 224
Python Programlama Dilinde Fonksiyonlar 225
Python Modüller ve Paketler 227
Python ile Nesneye Yönelik Programlama 230
Python'da Sınıf ve Nesne 230
Metotlar 232
Erişim Belirleyicileri 233
Django Web Çatısı 240
Kurulum 240
Proje Oluşturma 241
Şablon Kullanımı 245
Temel Etiketler 246
Filitreler 246
Formlar 248
Django Modeller 249
Model ile Veri Erişimi 252
Veri Listeleme 252
Veri Ekleme 253
Veri Seçme 254
Veri Güncelleme 256
Veri Silme 256

BÖLÜM 8 Ruby ve Ruby On Rails

Giriş 267
Ruby Nedir ve Ne İşe Yarar? 267
Ruby Dilinin Temel Özellikleri 268
Ruby Dili Söz Dizimi ve Bileşenleri 270
Ruby Kodunun Çalıştırılması 270
Ruby Dili Sözdizimi 271
Ruby On Rails Çatısı 281
Ruby On Rails ile Uygulama Geliştirme 283

Önsöz

Sevgili öğrenciler,

Programlama, sıklıkla güncel teknolojilerle anıldığı için yeni bir kavrammış gibi anlaşılsa da kökeni dokuma tezgahlarına kadar uzanır. Geçtiğimiz yüzyılda ise programlama yöntemi elle takılıp çıkarılan fişler, kablolar ve kumanda edilen düğmelerden bilgisayar dili olarak bilinen ikili kodlamaya (1 ve 0) evirilmiştir. Bu dönüşüm de bilgisayarların yapısal işlevlerinin artmasına yol açmıştır. Temel olarak, programlamayı cihazların öngörülen görevleri yerine getirmesi için yazılan, algoritmik komutlar dizisi şeklinde tanımlamak mümkündür. İnternet tabanlı programlama ise günümüzün vazgeçilmez internet tabanlı uygulamalarının geliştirilmesi ve bu uygulamaların Web tarayıcılarında sorunsuz çalışması amacıyla geliştirilen programlama yapısını ifade etmektedir.

Internet tabanlı programlamanın temel mantığını anlamak için öncelikle internet tabanlı programlamanın gelişim sürecinin, alanla ilgili terim, kavram, yapı ve modellerin iyi anlaşılması gerekmektedir. Bu nedenle kitapta öncelikle temel kavramlar, internet tabanlı programlama dillerinin yapısı, betik dillerin rolleri, sunucu-istemci modelleri, istemci ve sunucu taraflı programlamanın mantığı, nesne tabanlı programlama, dinamik Web ile statik Web ayrımı, MVC ve MTV modellerinin yapısı anlatılmıştır. Daha sonra internet tabanlı programlama dilleri, ilgili Web çatıları ve kütüphaneler uygulama örnekleriyle birlikte bölümler halinde ayrıntılı olarak anlatılmıştır.

Statik internet uygulamaları ve tasarımları temel HTML kodları ile üretilen sayfalardan oluşurken dinamik internet tabanlı uygulamalar ve Web siteleri HTML kodlarının yanında CSS, JavaScript, PHP, ASP, .NET gibi kodlar ile SQL ve Oracle gibi veritabanı desteğiyle çalışırlar. Sunucu ve istemci taraflı teknolojilerin birlikte çalışma süreci bildiğimiz Web erişiminin temelini oluşturur. Bu kitapta HTML ve CSS betik dillerin yanında PHP, ASP, .NET, JavaScript, Ruby ve Python internet tabanlı programlama dilleri anlatılmıştır. Kitapta ayrıca JQuery Kütüphanesi, Django Web Çatısı ve Ruby on Rails Web Çatısı üzerinde ayrıntılı olarak durulmuştur. Bu kitabın Web Tasarımı ve Kodlama ile ilgilenen herkese yararlı olmasını temenni eder tüm öğrencilerimize başarılar dilerim.

Editör

Doç.Dr. Mehmet FIRAT

Bölüm 1

İnternet Tabanlı Programlamanın Temelleri

Öğrenme çıktıları

1 İnternet Tabanlı Programlama Nedir?

1 İnternet tabanlı programlamayı tanımlayabilme

2 İnternet Tabanlı Programlamanın Tarihsel ve Yapısal Gelişimi

2 İnternet tabanlı programlamanın tarihsel ve yapısal gelişimini açıklayabilme

3 İnternet Tabanlı Programlama Temel Kavramları

3 İnternet tabanlı programlama temel kavramlarını karşılaştırabilme

4 İnternet Tabanlı Programlama Dilleri

4 İnternet tabanlı programlama dillerini sınıflayabilme

5 Dinamik Web

5 Dinamik Web yapılarını karşılaştırabilme

Anahtar Sözcükler: • İnternet Tabanlı Programlama • Statik Web • Dinamik Web • Hosting • Domain • Proxy • Sunucu • İstemci

GİRİŞ

Programlamanın 1801 yılında Joseph Marie Jacquard'ın dokuma tezgâhındaki işlemleri otomatikleştirmek adına ürettiği delikli kartlar ile başladığı söylenebilir. Programlama, bilgisayarların gelişmesiyle ilerleme kaydetmiştir. İlk bilgisayar ENIAC, altı bayan operatör tarafından programlanıyordu. Programlama yöntemi ise elle takılıp çıkarılan fişler, kablolar ve kumanda edilen düğmelerdi. Bilgisayar dili olarak bilinen ve ikili kodlamaya (1 ve 0) sahip programlamanın yapısı bilgisayarın işlevlerinin gelişmesi ile değişmiş ve dönüşüm yaşamıştır. İlk programlama dili olarak bilinen Assembly dili 1 veya 0 ile tanımlanan kodlama yapısından kelimelerle ifade edilen yapılara dönüşmüştür. Programcının Assembly dilinde yazdığı "Load, Add, Store" gibi komutları bilgisayarlar arka planda Assembler adı verilen programlarla yine 1'lere ve 0'lara dönüştürmeye devam etmektedir. Fakat programcı artık bu kısımla ilgilenmemektedir. Assembly dilinden sonra kodlamayı veya programlamayı kolaylaştıran BASIC, FORTRAN, COBOL, Pascal, C/C++ ve Java gibi programlama dilleri geliştirilmiştir. Programlama dillerinde bu şekilde değişim ve dönüşüm yaşanırken bilgisayarların kendi aralarında haberleşebilmelerini sağlamak amacıyla internet teknolojileri de gelişme kaydetmiştir.

İnternet teknolojisi, savunma sanayiinde kullanılmak üzere ortaya çıkmış olsa da zamanla herkesin kullanımına açılmaya başlanmıştır. 1990'lı yıllarda WWW ve HTML teknolojilerinin gelişmesiyle birlikte internet hızla yayılmıştır. Uluslararası bir gerçek zamanlı istatistik projesi olan *Internet Live Stats* verilerine göre dünya genelinde internet kullanıcı sayısı 3.7 milyara ulaşmıştır. İnternetin kullanımı yaygınlaşırken internet tabanlı programlama dilleri de hızlı bir gelişim göstermiştir.

Programlama dillerinin yapılarına bakıldığında; alt seviye programlama dilleri, orta seviye programlama dilleri ve üst seviye programlama dilleri olarak gelişim gösterdikleri söylenebilir. Günümüze kadar 2500'den fazla programlama dili geliştirildiği bilinmektedir. Seviyelere göre geliştirilen programlama dillerinin yapıları farklılık gösterse de herhangi bir seviyede olan kullanıcılar bu programlama dillerinden yararlanabilmektedir.

İnternet tabanlı programlamanın tarihine bakıldığında 25-30 yıllık bir geçmişinin olduğu görülmektedir. Fakat bu kısa sürede çok çeşitli internet tabanlı programlama dilleri geliştirilmiştir. İnternet tabanlı programlamanın ilk yıllarında statik Web sayfaları geliştirmeye olanak sağlayan HTML bulunmakta iken; takip eden yıllarda dinamik Web sayfaları geliştirebilmek amacıyla PHP ve ASP gibi internet tabanlı programlama dilleri geliştirilmiştir. Uzmanlık bilgisi gerektiren internet tabanlı programlama zamanla uzman olmayan kişilerin de kolaylıkla Web sayfaları geliştirmesine olanak sağlar duruma gelmiştir. Günümüzde internet tabanlı programlama hakkında az bilgisi olan kişilerin dahi Web tasarımı ve kodlama için kolaylıkla kullanabildiği uygulamalar bulunmasına rağmen internet tabanlı programlama önemini kaybetmemektedir. Çünkü bu tür paket uygulamaların da geliştiricileri, internet tabanlı programlama bilgisine sahip insanlardır.

İNTERNET TABANLI PROGRAMLAMA NEDİR?

Program, belirli bir işin belirli bir düzen içerisinde yapılmasıdır. Bu düzen genellikle algoritmik yapıyla sağlanmaktadır. Elektronik devrelere sahip cihazların belirli görevleri yapabilmesi için o görevlerin nasıl yapılacağını bilmeleri gerekir. Burada karşımıza **programlama** kavramı çıkmaktadır.

> **Programlama**
> Üzerinde elektronik devreleri bulunan cihazların belirli görevleri yerine getirmesi için yazılan, algoritmik yapıda olan ve bir programlama dili ile yazılmış olan komutlar dizisidir.

Programlama yaparak cihazlara ne yapması gerektiği bilgisi verilir. Cihazlar da o görevleri programlama içerisinde bulunan komutlara göre gerçekleştirir. Diğer bir ifadeyle elektronik cihazların ne yapması gerektiği programlama ile belirlenmektedir. Programlamanın tarihçesine bakıldığında dokuma tezgahlarında kullanılan **delikli kartlar** (Resim 1.1) karşımıza çıkmaktadır.

> **Delikli kartlar**
> Programlamanın ilk örneğidir. Dokuma tezgâhlarında değişik şekillerde dokuma yapılabilmesi için geliştirilmiş kartlardır.

Resim 1.1 Delikli Kart

Delikli kartlar ile dokumada çeşitli şekiller kolaylıkla yapılabilmektedir. Farklı tasarımları içeren delikli kartlar dokuma tezgâhlarına takılarak dokumalar gerçekleştirilmiştir. 19. yüzyılda dokuma tezgahlarını denetlemekte kullanılan delikli kartlar 20. yüzyılda önce hesap makinalarında ve sonra bilgisayarlarda kullanılmaya başlanmıştır. Zamanla bilgisayar teknolojileri gelişme kaydetmiş ve programlama yapısı elektronik ortama geçerek farklı bir hal almıştır. Bilgisayarların aralarında iletişim kurmaları için geliştirilen internet teknolojilerinin ortaya çıkmasıyla birlikte internet tabanlı programlama dilleri de gelişim göstermiştir.

Web tarayıcısı ile açılan Web sitelerinin her sayfasının arka planında çeşitli kodlar bulunmaktadır. Bu kodlar, bilgisayarda bazı işlemleri yapmak amacıyla geliştirilen programlama dilleri gibi yapılara sahiptir. Web sayfalarının gördüğümüz ekran görüntüsü arka planda işleyen, görmediğimiz bir kodlama altyapısına sahiptir. Bu arka planda işleyen yapının Web tarayıcılarında görüntülenmesini sağlayacak şekilde düzenlenmesi için kodlamalarının yapılması **internet tabanlı programlama dilleri** ile gerçekleştirilmektedir.

İnternet tabanlı programlama dili
Web sitelerinin geliştirilmesi ve Web tarayıcılarda sorunsuz bir şekilde görüntülenebilmesi için geliştirilen programlama yapısıdır.

Öğrenme Çıktısı

1 İnternet Tabanlı Programlama Nedir?

Araştır 1

Programlama dili nedir? Araştırınız.

İlişkilendir

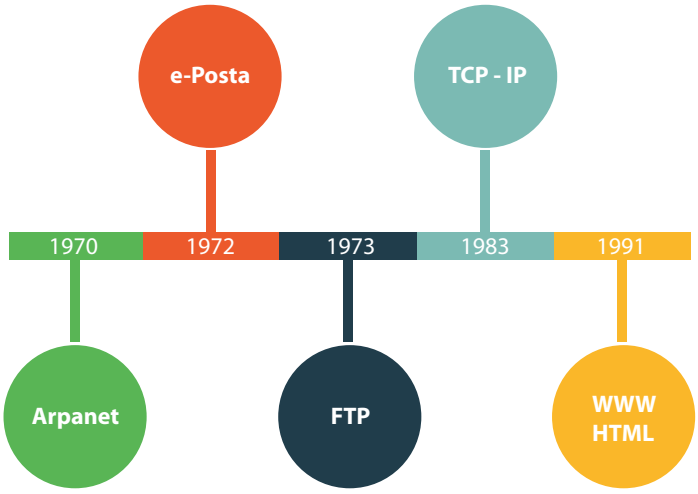
Günümüz programlama dillerinin atası olan delikli kartların çalışma mantığını güncel programlama dillerinin çalışma mantığı ile ilişkilendiriniz.

Anlat/Paylaş

Daha önce bilgi ve deneyim sahibi olduğunuz programlama dillerini paylaşınız.

İNTERNET TABANLI PROGRAMLAMANIN TARİHSEL VE YAPISAL GELİŞİMİ

İnternetin tarihsel gelişim sürecine bakıldığında, bilgisayarların gelişmesinin ardından bilgisayarların kendi aralarında haberleşmelerini sağlamak amacıyla bilgisayar ağlarının kurulması gereksinimine dayandığı görülmektedir. Amerika Birleşik Devletleri'nde (ABD) 1960'ların başında Savunma Bakanlığı tarafından çalışmalara başlanan ve ARPANET olarak ortaya çıkan yapı İnternet Protokolü (IP) kullanan ilk yapıdır. Bilgisayarlar arasında kurulan bu yapıya 1972 yılında e-posta hizmeti eklenmiş ve takip eden 1973 yılında FTP (Dosya Transfer Protokolü) hizmeti eklenmiştir. Savunma alanında kullanılmak üzere geliştirilen internet 1990'lardan itibaren yaygınlaşmasını devam ettirmiştir. World Wide Web (WWW) ve Hyper Text Markup Language (HTML) hizmetlerinin internet hizmeti olarak kullanılmaya başlamasıyla internetin yaygınlaşması hızlanmış ve 1994-1995 yıllarında 60 milyon kullanıcıya ulaşmıştır.



Şekil 1.1 İnternet Teknolojilerinin Tarihsel Gelişimi

Şekil 1.1'de görüldüğü gibi 1991 yılında WWW ve HTML'in gelişmesiyle birlikte internet tabanlı programlama önem kazanmış ve süreçte çok sayıda internet tabanlı programlama dili geliştirilmiştir. İlk internet tabanlı dil olarak karşımıza HTML dili çıkmaktadır. HTML, 1991 yılında Tim Berners-Lee tarafından geliştirilen bir betik dili olup ilk amacı CERN'de internet tabanlı bilgi paylaşımı yapmaktır. HTML ile hazırlanan içerikler .html ya da .htm uzantılı olmaktadır. HTML dili ile hazırlanan içerikleri Web tarayıcıları ile görüntülemek mümkündür. Aşağıda HTML dilinde hazırlanmış basit bir kod yapısı görülmektedir.

> HTML, bir programlama dili değildir. Web sayfalarının görsel boyutta düzenlenmesini sağlayan işaretleme dilidir.

```
<html>
  <head>
    <title>Anadolu Üniversitesi Açıköğretim Fakültesi</title>
  </head>
  <body>
    <p>Merhaba Dünya!</p>
  </body>
</html>
```

Bu kodların tarayıcılarda görüntülenmesi sonucunda elde edilen ekran görüntüleri Resim 1.2'de görüldüğü gibidir.

Resim 1.2 HTML Kodlarının Farklı Web Tarayıcılarda Görüntülenmesi

Tarihsel süreçte bakıldığında HTML dilinin versiyonları Tablo 1.1'de görüldüğü gibidir.

Tablo 1.1 HTML Versiyonları

Versiyon	Yıl
HTML	1991
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2014

Kaynak: http://www.w3schools.com/html/html_intro.asp adresinden edinilmiştir.

1991 yılında geliştirilen HTML ile sayfalarda işaretlemeler yapılmakta ve geliştirilen sayfalar statik çalışmaktadır. 2014 yılında yayınlanan ve geliştirilme süreci devam eden HTML5 ile *media* etiketleri, *canvas* etiketleri, *form* etiketleri ve normal kullanım için yeni etiketler kullanılmaya başlanmıştır. Bu etiketlerin yanı sıra form elemanı olan *input type* için de çok sayıda yeni türler geliştirilmiştir. Bu yeni etiketler ile Flash programında yapılabilen animasyonlar, Web galerileri, hareketli Web sayfaları yapmak artık mümkündür. Apple firmasının Web tarayıcılarında Flash Player desteğini kaldırdığı düşünüldüğünde HTML5'in öneminin arttığını söylemek mümkündür. HTML5 ile hazırlanan Web sayfalarının görüntülenmesi için Flash ile hazırlanan sayfalarda olduğu gibi ek bir uygulamanın bilgisayarlara yüklenmesine gerek yoktur. HTML5 kodları Web tarayıcılarında kolaylıkla görüntülenebilmektedir.

HTML dilinden sonra 1995 yılında Rasmus Lerdorf tarafından *Personal Home Page* (PHP) dilinin geliştirildiği görülmektedir. Ücretsiz ve nesne yönelimli olarak geliştirilen PHP dili, çoğunlukla hareketli ve etkileşimli Web sayfaları oluşturmak için kullanılan, sunucu taraflı, çok geniş kullanımlı, genel amaçlı, HTML içerisine gömülebilen bir betik programlama dilidir. PHP dili açık kaynak kodlu bir yapıya sahip olduğu için Rasmus Lerdorf'tan sonra PHP dilinin geliştirilmesini PHP topluluğu gerçekleştirmektedir.

Dinamik Web sayfaları geliştirmeye olanak sağlayan PHP dilinin kodları PHP işleme modülü bulunan Web sunucuları tarafından yorumlanarak Web tarayıcılarında görüntülenebilir. Aşağıda HTML içerisine PHP kodlarının gömülmesiyle oluşturulan yapı görülmektedir.

```
<html>
<head>
<title>Anadolu Üniversitesi Açıköğretim Fakültesi</title>
</head>
<body>
<?php echo "Merhaba Dünya!"; ?>
</body>
</html>
```

PHP dili ile oluşturulan kodlar <?php…?> arasına yazılmaktadır. Bu kod HTML sayfaların içine gömülebileceği gibi .php uzantılı sayfalarla da kodlar yazılabilir. PHP dili ile yazılmış kodlardan oluşan içeriğin sorunsuz bir şekilde çalışabilmesi için siteyi çalıştıran Web sunucularının Linux işletim sisteminde olması gerekmektedir. Bunun yanı sıra veritabanı olarak MySQL veritabanı önerilmektedir.

En yaygın kullanılan internet tabanlı programlama dillerinden biri de *JavaScript* dilidir. JavaScript, Mayıs 1995 yılında Brendan Eich tarafından geliştirilen ve Web tarayıcılarında kullanılmakta olan dinamik bir programlama dilidir. Aralık 1995 yılında ise Netscape firması tarafından piyasaya sürülerek kullanımına başlanmıştır. JavaScript, C dilinin Web ortamına uyarlanmış halidir. JavaScript prototip-tabanlı, dinamik türlere ve birinci-sınıf fonksiyonlara sahip bir betik dilidir. Hazırlanan Web sayfalarında yapılması gereken kontroller JavaScript ile yapılabilmektedir.

JavaScript, HTML kodlarının içine *<head>…</head>* etiketlerinin arasına *<script> …</script>* etiketleri ile yazılır. Bu şekilde HTML kodlarının içine JavaScript kodları yazılabileceği gibi *.js* uzantılı dosyalar içerisine Javascript kodları yazılarak da sayfalar oluşturulabilir. Bu sayfalar HTML sayfalarına bağlanarak veya dışarıdan çağrılarak çalıştırılır. JavaScript kodlarının düzgün bir şekilde Web tarayıcılarında görüntülenebilmesi için Web tarayıcılarının JavaScript desteğine sahip olması gerekmektedir. JavaScript kodları ile hazırlanmış HTML sayfası aşağıda görülmektedir.

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Javascript Kodu ile İçerik De-
ğişimi";}
</script>
</head>
<body>
<h1>Anadolu Üniversitesi Açıköğretim Fakültesi</h1>
<p id="demo">Web Tasarımı ve Kodlama Önlisans Programı</p>
<button type="button" onclick="myFunction()">Yazıyı Değiştir</button>
</body>
</html>
```

Yukarıda verilen örnek kod yapısında da görüldüğü gibi *head* etiketleri arasına JavaScript kodları yazılmıştır. Bu sayfanın Web tarayıcılarındaki ekran görüntüsü Resim 1.3'te olduğu gibi oluşmaktadır.

Resim 1.3 Javascript Kodlarıyla Hazırlanmış HTML Sayfasının Ekran Görüntüsü

Resim 1.3'te görüldüğü gibi *Yazıyı Değiştir* butonu bulunmaktadır. *Yazıyı Değiştir* butonu ile "Web Tasarımı ve Kodlama Önlisans Programı" yazısı "Javascript Kodu ile İçerik Değişimi" yazısına dönüşecektir. Bu değişimi Resim 1.4'te görmek mümkündür.

Resim 1.4 Javascript ile Yazı İçeriği Değişimi

PHP dilinin geliştirilmesini takip eden 1996 yılında Microsoft firması tarafından dinamik Web sayfaları oluşturmak amacıyla sunucu taraflı çalışabilen *Active Server Pages (ASP)* dili geliştirilmiştir. ASP dilinin kodları <%...%> arasına yazılmaktadır. ASP dili, PHP dilinin aksine açık kaynak kodlu bir yapıda değildir. Dil ile ilgili geliştirmeler Microsoft firması tarafından gerçekleştirilmektedir. ASP dilinde hazırlanan kodlar PHP dilinde olduğu gibi HTML sayfalarının içine gömülebileceği gibi .asp sayfaları şeklinde yeni

sayfalar içerisinde de yer alabilir. ASP dilinin devamı olarak Microsoft firması tarafından ASP.NET geliştirilmiştir. ASP.NET, dinamik Web sayfaları, Web uygulamaları ve XML tabanlı Web hizmetleri geliştirmeye olanak sunmaktadır. ASP.NET ile oluşturulan Web sayfalarının uzantıları .aspx olarak oluşmaktadır. Tablo 1.2'de ASP.NET'in versiyonları görülmektedir.

ASP.NET, interaktif Web siteleri oluşturabilmek amacıyla farklı dillerin ortak dil çalışma zamanı (Common Language Runtime) altyapısına uygun olarak kullanılabilmesini sağlayan bir teknolojidir. Bu dillerin yanı sıra Python, Django, Ruby ve Ruby on Rails gibi programlama dilleri de internet tabanlı programlama dilleri olarak geçmektedir. Kitabın ilerleyen ünitelerinde bu dillere ilişkin detaylı bilgiler verilecektir.

Tablo 1.2 ASP.NET Versiyonları

.NET Versiyonu	Yıl
1.0	-
1.1	2003
2.0	2005
3.0	-
3.5	2010
4.0	2000
4.5	2012
4.5.1	2013
4.5.2	-
.NET 4.6	2015
.NET 4.7	2016

Kaynak: https://msdn.microsoft.com/tr-tr/library/bb822049(v=vs.110).aspx adresinden edinilmiştir.

Öğrenme Çıktısı

2 İnternet Tabanlı Programlamanın Tarihsel ve Yapısal Gelişimi

Araştır 2

Ziyaret ettiğiniz Web sayfalarının internet tabanlı programlama dillerinden hangisiyle yazıldığını nasıl anlarsınız?

İlişkilendir

İnternet tabanlı programlamanın tarihi ile İnternetin tarihsel gelişimi arasında nasıl bir ilişki var? Açıklayınız.

Anlat/Paylaş

Şimdiye kadar HTML veya bir programlama diliyle ilgili eğitim aldınız mı? Deneyimlerinizi anlatınız.

İNTERNET TABANLI PROGRAMLAMA TEMEL KAVRAMLARI

İnternet veya internet tabanlı programlama denildiğinde Web tarayıcısı, Web sayfası, www, http, URL, hosting, domain, IP, statik, dinamik, Proxy, HTML, CSS, PHP, ASP, Java ve Veritabanı gibi kavramlar karşımıza çıkmaktadır. Burada bahsedilen kavramlara ilerleyen sayfalarda ayrı ayrı değinilmiştir.

Web Tarayıcı

Geliştirilen Web sitelerinin sunuculara yüklenmesi sonucunda o Web sitelerini ziyaret etmek için bilgisayarlarda aracı programlar olması gerekmektedir. Bu aracı programlar Web tarayıcılarıdır. Standart Web tarayıcısı; metin veya çoklu ortam dosyalarını açabilir, kaydedebilir, HTML'den HTTP'ye bütün protokolleri ve standartları destekler, açılan sayfada aranan nesneyi bulabilir, sık kullanılanlar ve geçmiş listesi yapabilir, genel ağa dosya yükleme ve genel ağdan dosya indirme yapabilir, e-posta ve metin editörleriyle bütünleşebilir, linkleri (bağlantı) izleyebilir, dosya sistemlerini okuyabilir, bağlayabilir ve kaydedebilir, çoklu ortam dosyalarını oynatabilir veya kaydedebilir, sayfanın çıktısını alabilir ve çevrimdışı da çalışabilir. İnternet Explorer, Google Chrome, Mozilla Firefox, Apple Safari, Opera ve Yandex Browser en çok tercih edilen Web tarayıcılarıdır. Mobil cihazlar için Opera Mini, Opera Mobile, Apple Safari, Google Chrome ve Microsoft Edge Web tarayıcıları kullanılmaktadır.

WWW

WWW, bilgisayarların birbiriyle iletişim kurabildiği, görüntü, ses ve veri paylaşımının yapılabildiği tüm dünyaya yayılmış büyük bir ağdır. Web sayfası ise WWW üzerinde görüntülenen Web sitelerinin her bir dosyasını temsil etmektedir. URL (Uniform Resource Locator), internet üzerinde Web sitelerinin sahip oldukları adreslerin genel adıdır. "www.anadolu.edu.tr" Anadolu Üniversitesi'nin Web sitesinin URL'sidir.

Hosting ve Domain

Sunucularda kullanıcılar için ayrılan bölümlere hosting denilmektedir. Web sitelerinin içerikleri hostingler vasıtası ile sunucularda o Web siteleri için ayrılan bölümlerde saklanır. Web sitesi için ayrılan hosting alanlarına FileZilla, CuteFTP, WinSCP, Easy FTP ve SmartFTP gibi yardımcı programlarla erişilebilir.

Her Web sitesinin belli bir IP'si vardır. IP'ler sayılardan oluşan bir yapıda olduğu için her sitenin IP'sini akılda tutmak zor olacaktır. Bu nedenle Web sitelerinin adreslerinin kolay hatırlanabilmesi için alan adı (domain) geliştirilmiştir. Web sitelerinin kendine özgü alan adları vardır. Domain belirlerken uzantıları da önemlidir. Kullanım amaçlarına göre bazı domain uzantıları:

> Domain (alan adı) uzantıları her ülke için özelleştirilmektedir. Örneğin ".com" uzantılı domaine sahip bir Web sayfası Türkiye'de ".com.tr" adresini alırken; Almanya'da ".com.de" adresini alabilir.

- .aero Havacılık Sektörü
- .arpa İnternet Mimari Tasarım Kurulu tarafından rezerve edilmiştir.
- .asia Asya-Pasifik bölgesindeki kurum ve organizasyonlar
- .biz İş
- .cat Katalan dili ve kültürü
- .com Ticari organizasyonlar için ama herkes alabilir
- .coop Kooperatifler
- .edu Eğitim
- .gov Devlet Kurumları
- .info Bilgi amaçlı ama herkes alabilir
- .int Uluslararası organizasyonlar
- .jobs İş amaçlı kullanımlar
- .mil Amerikan Askerlik Kurumu
- .mobi Mobilite ilişkili cihaz ve firmalar
- .museum Müzeler
- .name Aile ve bireyler
- .net Network altyapıları ama herkes alabilir
- .org Organizasyonlar ama herkes alabilir
- .pro Belli profesyonellikler
- .tel Telefon servisleri
- .travel Seyahat ajansları, uçak firmaları, oteller vs

Statik ve Dinamik Web Sitesi

Statik Web siteleri, daha çok uzman kişiler tarafından hazırlanan ve düzenlemelerinin de yine uzman kişiler tarafından yapılabildiği Web siteleridir. Statik Web sayfaları, daha çok bilgi verme amaçlı hazırlanan ve sık sık düzenleme gerektirmeyen sitelerdir. Üzerinde değişiklik yapılan statik Web sayfalarının sunucuya

tekrar yüklenmesi gerekmektedir. Statik Web sitesini oluştururken HTML, XHTML, DHTML ve CSS dilleri kullanılabilir.

Dinamik Web siteleri ise verilerinin veri tabanlarında tutulduğu ve istenildiğinde bu bilgilerin uzman olmayan kişiler tarafından da kolaylıkla değiştirilebildiği yapılardaki sitelerdir. Dinamik Web siteleri PHP veya ASP gibi internet tabanlı programlama dilleriyle geliştirilmektedir. Veritabanı üzerinde yapılan değişikliklerle sayfalarda otomatik değişiklik yapıldığından dolayı yapılan değişiklik durumunda Web sayfalarının sunucuda manuel olarak güncellenmesine gerek yoktur. Dinamik Web sitesi oluştururken ASP.Net ASP, PHP, JavaScript ve CFM gibi diller kullanılabilir.

HTML ve HTML Etiketleri

HTML: HyperText Markup Language yani Hareketli-Metin İşaretleme Dili. Web sayfalarının programlama mantığına dayalı olmayan şekilde oluşturulmasını sağlayan metin işaretleme dilidir. HTML etiketlerinden bazıları şunlardır:

- *<html> </html>*: Oluşturulan belgenin HTML dokümanı olduğunu belirtir.
- *<head> </head>*: Oluşturulan HTML belgesinin baş kısmını oluşturur. Bu iki etiket arasında belgenin başlığının belirtilmesi zorunludur.
- *<title> </title>*: Oluşturulan Web sayfasının başlığının yazıldığı kısımdır. Head etiketlerinin arasında yer alır.
- *<meta> </meta>*: Meta etiketleri oluşturulan sayfa hakkında bilgiler (description) ve Web sayfası yazarı (author) bilgisi verilmekte ve Web sayfasını tanımlayan anahtar kelimeler (keywords) yazılabilmektedir.
- *<body> </body>*: Oluşturulan belgenin gövde kısmını oluşturur. Görüntülenmesi istenen içerik kodları bu iki etiket arasına yazılır.
- Başlık Etiketleri: Web sayfası içeriklerinde başlık hiyerarşisini belirtmek için kullanılır. *<h1>*'den *<h6>*'ya kadar başlıklandırma yapılabilir.
- * *: Web sayfalarının içeriğindeki paragraf yapısını belirtir.
- *
* : Bir alt satıra geçiren etikettir.
- * *, * * ve * * : Bu etiketler Web sayfalarının içeriklerinde liste şeklinde sıralanmış bilgiler verileceği zaman kullanılır. etiketi, düzensiz yani madde imleriyle belirtilen listeler oluşturulmasında kullanılırken; etiketi düzenli yani "1,2,3,…" veya "a,b,c,…" gibi belli sırada listeler oluşturulmasında kullanılır. etiketleri ise *<ul>* veya *<ol>* etiketleri arasında kullanılır.
- *<hr>*: Web sayfası içeriğinde bölümleri çizgilerle ayırmada kullanılır.
- *<table> </table>* : Web sayfasında tablo eklemek için kullanılır.
- *<tr> </tr>*: Tablo oluştururken satır(ları) oluşturmak için kullanılır.
- *<th> </td>*: Tablo oluştururken sütun(ları) oluşturmak için kullanılır.
- *<img>* : Web sayfasına görsel eklemek için kullanılır.
- *<a href> *: Web sayfası içeriklerinde herhangi bir nesneye bağlantı (link) eklemek için kullanılır.

Aşağıda HTML etiketlerinin kullanıldığı kod örneği görülürken bu kodların Web tarayıcısında görüntülenmesi sonucu oluşan ekran görüntüsü Resim1.5'te görülmektedir.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Anadolu Üniversitesi Açıköğretim Fakültesi</title>
<meta name="description" content="Anadolu Üniversitesi Açıköğretim Sayfasıdır">
<meta name="author" content="AÖF">
</head>
<body>
<h1>Anadolu Üniversitesi</h1>
<h2>Açıköğretim Fakültesi</h2>
<h3>Web Tasarımı ve Kodlama Önlisans Programı</h3>
```

```
<p>Uzaktan öğretim, "öğrencinin iletişim teknolojilerinden yararlanarak <br> öğ-
renmesini sağlayan" çağdaş bir uygulamadır. Gelişmiş ya da gelişmekte olan pek çok
ülkede uygulanmakta olan uzaktan öğretim yoluyla her yaş, her gelir ya da...</p>
<ul>
<li>Düzensiz</li>
<li>Liste</li>
</ul>
<ol>
<li>Düzenli</li>
<li>Liste</li>
</ol>
<table width="200" border="1">
  <tr>
    <td>a</td>
    <td>b</td>
    <td>c</td>
  </tr>
  <tr>
    <td>d</td>
    <td>e</td>
    <td>f</td>
  </tr>
</table><br>
<img  src="anadolu_universitesi.jpg"  width="317"  height="207"  alt="Anadolu
Üniversitesi"><br>
<a href="www.anadolu.edu.tr"> Anadolu Üniversitesi </a>
</body>
</html>
```

Resim 1.5 HTML Etiketleriyle Oluşturulan Örnek Sayfa

Şekil 1.5'te görüldüğü gibi HTML kodlarıyla hazırlanan Web sayfası tarayıcıda aynen görüntülenebilmektedir.

CSS (Cascading Style Sheets)

CSS bir programlama dili olmayıp HTML'e ek olarak metin ve şekilsel düzenlemelere olanak sağlayan bir Web teknolojisidir. Yerel CSS, genel CSS ve harici CSS olmak üzere üç türü bulunmaktadır. Yerel CSS, sayfada tanımlandığı yerde geçerliyken; genel CSS sayfanın bütünü için geçerlidir. Harici CSS ise Web sayfalarının içerisinde bulunmayıp *.css* uzantılı harici dosyalar içerisinde kodların yazılarak oluşturulduğu ve Web sayfalarına bağlantılarının yapılması yoluyla çalıştırıldığı CSS türüdür. Yerel CSS kodları <*body*> <*/body*> arasına yazılırken genel CSS kodları <head> </head> arasına yazılır. Harici CSS ise <*head*> <*/head*> arasına harici CSS'in bağlantısının verilmesi yoluyla kullanılabilir.

Temel CSS3 yapısı için http://www.w3schools.com/css/css3_intro.asp adresini ziyaret edebilirsiniz.

Veri Tabanı

Veritabanı (Database), birbirleriyle ilişkili verilerin depolandığı alanlardır. Veri tabanları verilerin yönetilmesini, güncellenmesini ve taşınmasını kolaylaştırır. Veritabanı üzerinde yapılan değişiklikler ile Web sitesindeki bütün içerikler kolaylıkla değiştirilebilmektedir. SQL, Pl/SQL ve OQL gibi veritabanı dilleri bulunmaktadır. Bu dillerle oluşturulmuş MySQL, MsSQL, PostgreSQL, Oracle, Sybase, Berkeley, Firebird, Interbase, Informix ve MsAccess gibi veritabanı sistemleri mevcuttur.

Protokol

Kelime anlamı itibariyle iki taraf arasında yapılan sözleşme olan protokol internet tabanlı teknolojiler açısından düşünüldüğünde de benzer anlamdadır. Temel olarak istemci ve sunucu taraflı olarak iki tarafın sözleşmeye uygun davranması anlamına gelir. Web teknolojilerinde kullanılan protokollerden bazıları şunlardır:

- *TCP/IP* (Transmission Control Protocol / Internet Protocol): Yapı olarak iki katmanlı bir haberleşme protokolüdür. Üst Katman TCP (Transmission Control Protocol) verinin iletimden önce paketlere ayrılmasını ve karşı tarafta bu paketlerin yeniden düzgün bir şekilde birleştirilmesini sağlar. Alt Katman IP (Internet Protocol) ise, iletilen paketlerin istenilen ağ adresine yönlendirilmesini kontrol eder.
- *HTTP* (Hyper-Text Transfer Protocol, Hiper-Metin Transfer Protokolü): 1990 yılından itibaren www ile birlikte geliştirilen bu protokol, bir kaynaktan dağıtılan ve ortak kullanıma açık olan hiperortam bilgi sistemleri için uygulama seviyesinde bir iletişim protokolüdür. Kısaca, Web sayfalarının HTML formatında Web tarayıcısına aktarılmasını sağlayan protokoldür. İnternet üzerinden *http* protokolü ile istenilen her bilgiye ulaşmak mümkündür.
- *FTP* (File Transfer Protocol, Dosya Transfer Protokolü): İnternete bağlı olan bir bilgisayardan başka bir bilgisayara veri transferi gerçekleştirmek için kullanılan protokoldür. FTP için kullanılan port numarası 21'dir.
- *SMTP* (Simple Mail Transfer Protocol, Basit Posta Aktarım Protokolü): İnternet üzerinden bir kişinin başka bir kişi ile e-posta gönderip alabildiği protokoldür.

Proxy

Vekil sunucu olarak bilinen Proxy, internete erişim sırasında kullanılan ara sunucudur. Proxylerin çalışma yapısı şu şekildedir:

- Tarayıcı vekil sunucuya bağlanır ve hangi sayfayı istediğini söyler.
- Vekil sunucu gerekiyorsa o sayfaya bağlanır ve içeriği alır.
- Vekil sunucu tarayıcıya içeriği gönderir.

Proxy ayarları Web tarayıcıları üzerinde gerçekleştirilmektedir. Bir Web tarayıcısında gerçekleştirilen Proxy ayarı değişikliği diğer Web tarayıcıda çalışmayacağı için her Web tarayıcısına ayrı ayrı Proxy ayarı yapılması gerekmektedir.

Öğrenme Çıktısı

3 İnternet Tabanlı Programlama Temel Kavramları

Araştır 3

Çalıştığınız işyerinde yöneticiniz internet üzerinden satış yapma kararı aldığını ve bu yönde bir Web sitesinin geliştirilmesi için sizi görevlendirdiğini söylediğinde Web sitesi geliştirmek için hangi adımları izlersiniz?

İlişkilendir

http ile Ftp arasındaki benzerlik ve farklılıkları anlatınız.

Anlat/Paylaş

Bildiğiniz domain uzantılarını ve kullanım amaçlarını paylaşınız.

İNTERNET TABANLI PROGRAMLAMA DİLLERİ

İnternet tabanlı programlama dilleri betik diller, istemci sunucu taraflı programlama ve nesne tabanlı programlama olarak üç kategoride ele alınmaktadır.

Betik Diller

Betik dili, Web sayfalarında dinamik içerik oluşturmak ve kullanıcılarla iletişim kurmak amacıyla çalışan dildir. HTML etiketleri arasında tanımlanarak kullanılabilmektedir. JavaScript, Lua, Perl, PHP ve Python programlama dilleri betik dil ailesindedir.

Javascript

1995 yılında Brendan Eich tarafından geliştirilen ve daha sonra Netscape firması tarafından yayınlanan JavaScript dili, istemci tarafı betikler ile tarayıcı ve kullanıcı arasında etkileşim kurulması, tarayıcının kontrol edilmesi, asenkron bir şekilde sunucu ile iletişime geçilmesi ve Web sayfası içeriğinin değiştirilmesi gibi işlevleri vardır. JavaScript ağırlıklı olarak istemci taraflı çalıştığı için Web sunucularını yormaz. JavaScript kodları, HTML kodları arasına *<script> </script>* etiketleri ile tanımlanarak yazılmaktadır. Javascript kütüphaneleri aşağıda sıralanmıştır.

- AngularJS
- Backbone.js
- Bootstrap
- D3.js
- Ember.js
- Express.js
- Foundation
- Ionic
- jQuery
- jQuery UI
- LESS
- Node.js
- ReactJS
- Script.aculo.us
- Underscore.js

Bu kütüphanelerden yaygın olarak kullanılan kütüphane jQuery'dir. jQuery, 2006 yılında John Resig tarafından geliştirilen ve jQuery ekibi tarafından geliştirilmesi devam eden açık kaynak kodlu JavaScript kütüphanesidir. http://jquery.com/ adresinden jQuery dosyası indirilerek Web sitelerine bağlantısı sağlanarak kullanılabilir. "Az kod, çok iş" sloganı ile jQuery Web sitelerine animasyonlar ve efektler eklenmesine olanak tanımaktadır. jQuery UI ise jQuery kütüphanesi kullanılarak geliştirilen ve daha gelişmiş arayüzlerin ve araçların daha kolay kullanabilmesine olanak veren ek kütüphanelere sahip JavaScript kütüphanesidir.

Lua

Lua, JavaScript gibi C dilinden üretilmiş bir script dilidir. Lua dili 1993 yılında Roberto Ierusalimschy, Luiz Henrique de Figueiredo ve Waldemar Celes tarafından tasarlanmış olan bir programlama dilidir. Oyun geliştirilmesinde daha çok tercih edilmektedir.

Perl

Larry Wall tarafından 1987 yılında geliştirilen Practical Extraction and Report Language (Pratik Çıkarım ve Raporlama Dili, Perl), Unix işletim sistemi için yorumlayıcı bir programlama dilidir. Sunucu tarafında çalışır. Dosya uzantısı "pl" dir. Perl programlarının çalıştırılabilmesi için bir Web sunucunun olması gerekir. Unix veya Linux ortamlarında çalışan Apache sunucu veya buna benzeyen programlar kullanılabilir. Açık kaynak kodlu olarak sunulan Perl programlama dili çoğu işletim sistemlerinde çalışmaktadır.

PHP

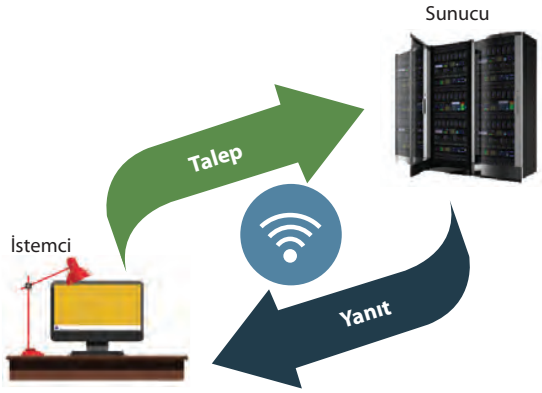
Rasmus Lerdorf tarafından 1995 yılında geliştirilen PHP dili sunucu taraflı çalışan ve HTML kodlarının içine gömülebilen betik ve programlama dilidir. Lerdorf, Perl dilinde geliştirdiği sayfaları ziyaret eden kullanıcı sayılarını tutmak amacıyla PHP dilini geliştirmiştir. PHP, açık kaynak kodlu bir yapıya sahiptir ve geliştirilmesi devam etmektedir. Şu an PHP5 sürümü yayınlanmaktadır. PHP kodları *<?php…?>* arasına yazılarak oluşturulmaktadır. PHP dilinde hazırlanan Web sayfaları için MySQL veritabanı tercih edilmektedir. Web ortamında yayınlanması için Web sunucuların Linux işletim sistemine sahip olması gerekmektedir. Facebook, YouTube, Yahoo, Wikipedia ve OGame gibi bilindik birçok Web sitesi PHP dili ile geliştirilmiştir.

Python

Guido van Rossum tarafından 1990 yılında geliştirilmeye başlanan ve 1994 yılında 1.0 sürümü ile yayınlanan Python, nesne yönelimli, yorumlamalı, birimsel (modüler) ve yüksek seviye etkileşimli bir programlama dilidir. Python, derlenmeye gerek duymadan çalışabilen bir yapıya sahiptir. Unix , Linux, Mac, Windows, Amiga, Symbian gibi platformlarda çalışabilmektedir.

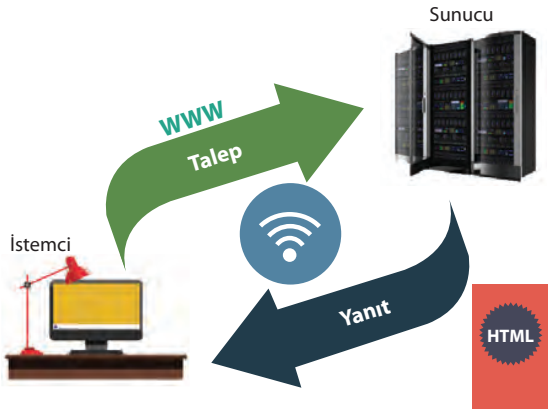
İstemci-Sunucu Taraflı Programlama

İnternete bağlı bir bilgisayardan Web tarayıcısında adres çubuğuna ziyaret edilecek Web sayfasının adresi veya IP adresi yazılarak sunucudan bilgi istenir. Sunucudan bilgi istenmesi işlemini istemci gerçekleştirir. İstemciye cevap veren ve istenilen içeriği istemciye ulaştıran yapı sunucu olarak tanımlanmaktadır (Resim 1.6)



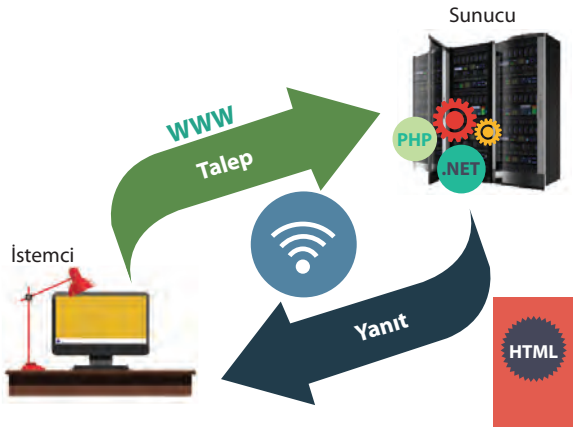
Resim 1.6 İstemci-Sunucu İlişkisi

İstemciden sunucuya gönderilen istek sayfası HTML sayfasıysa sunucu, Web tarayıcılarının HTML dilini yorumlayarak ekrana görüntüsünü yansıtabildiği için istemciden gelen sayfayı aynen geri gönderir (Resim 1.7). HTML belgesinin içinde CSS veya JavaScript kodlarının olması Web tarayıcısının görüntülemesini engellememektedir. Sunucudan istemciye aynen geri gönderilen istemci taraflı çalışabilen kodların güvenlik sınırlılığı vardır. Görüntülenen Web sayfasında fare ile sağ tıklayıp "Sayfa kaynağını görüntüle/göster" seçilirse ya da Web tarayıcıda klavyeden Ctrl+U tuş takımına basılırsa Web sitesini oluşturan bütün kodları görmek mümkün olacaktır.



Resim 1.7 İstemciden HTML Belgesinin İsteği

İstemciden sunucuya, sunucu taraflı çalışan Asp.Net, Php, Ruby ve Jsp gibi bir Web sitesi isteği gönderilmişse sunucu bu siteyi kendi yorumlayarak istemciye HTML formatında gönderecektir (Resim 1.8). Yorumlama işlemini sunucu kendi özelliğine göre yapabilmektedir. Sunucu taraflı hazırlanan Web sayfasında kodlar sunucuda yorumlandıktan sonra istemciye döndüğü için sayfa kaynağı görüntülense dahi Asp.Net, Php, Ruby ve Jsp ile yazılmış kodlar görüntülenmeyecektir.



Resim 1.8 İstemciden Sunucu Taraflı Çalışan Bir İstek Geldiğinde

Sunucuların (Server) özellikleri, oluşturulan Web sayfalarının sağlıklı bir şekilde görüntülenebilmesi için önemlidir. Sunucu, bilgisayar ağlarında, diğer ağ bileşenlerinin (kullanıcıların) erişebileceği, kullanımına ve/veya paylaşımına açık kaynakları barındıran bilgisayar birimidir. Sunucu türleri olarak Web sunucusu (Web Server), dosya sunucusu (File Server), Veritabanı sunucusu (Database Server), e-posta sunucusu (Mail Server) ve vekil sunucusu (Proxy Server) sayılabilir. Web sunucuları, Web sitelerinin barındırılması için kullanılan alanlardır. Web sunucuları, Linux veya Windows işletim sistemlerine sahip Web sunucuları olarak ikiye ayrılmaktadır. Linux Web sunucuları, PHP dilinde hazırlanan Web sayfalarının görüntülenmesini sağlarken; Windows Web sunucuları, ASP dilinde hazırlanan Web sayfalarının görüntülenmesini sağlamaktadır.

✓ PHP ve ASP gibi internet tabanlı programlama dilleriyle hazırlanmış Web siteleri sunucuda yorumlanır ve istemciye HTML sayfası gibi gönderilir.

Web sitesi geliştirirken yapılan her değişikliği sunucuya yüklemeden önce kontrol etmek amacıyla bazı durumlarda yerel sunucular (LocalHost) tercih edilmektedir. Bu şekilde yapılan değişikliklerde hata varsa öncelikle tespiti yapılır ve hata giderildikten sonra sunucuya yüklenerek Web sitesi yayınlanabilir. ASP ile geliştirilen Web sitelerinin yerel sunucuda kontrolü için bilgisayar(lar)ın işletim sisteminin Windows olması ve Internet Information Services (IIS) hizmetlerinin açık olması gerekmektedir. IIS, Windows işletim sisteminin yeni sürümlerinde yüklü olarak gelirken eski sürümlerde yüklü olmadığından dolayı eklenti olarak yüklenmesi gerekmektedir. PHP dilinde geliştirilen Web sitelerinin yerel sunucularda kontrol edilmesi için Xammp, Wamp, Appserv gibi yerel sunucular tercih edilebilir. PHP dilinde hazırlanan Web sayfalarının kontrolü için bu programların bilgisayara kurulması gerekmektedir.

Nesne Tabanlı Programlama

Nesne kelime anlamı itibariyle beş duyu organımızla algılayabildiğimiz her şeydir. Gerçek dünyada nesneler bu şekilde tanımlanırken bilgisayar ortamında da benzer şekilde gözle görebildiğimiz her şey nesnedir. Gerçek dünyada algıladığımız nesnelerin dijital ortamda da kullanılması kodlarla mümkündür (Resim 1.9). Bu bağlamda ilk kez 1960'lı yıllarda ortaya çıkan nesne tabanlı programlama (Object Oriented Programming), gerçek dünyadaki nesnelerin modellenmesi ve bu modele göre kodlanması yoluyla oluşturulan programlama türüdür. Beş duyu organımızla tanımlayamadığımız fakat dijital ortamda nesne olarak tanımlayabileceğimiz nesneler de mevcuttur. Örneğin hastane programı yaparken hastaların randevu alması da nesne olarak tanımlanabilir.

Resim 1.9 Gerçek Dünya Nesnelerinin Modellenmesi

Nesne tabanlı programlamada geliştirilecek olan yazılımda gereksinimler, nesneler ve nesneler arasındaki ilişkilerle karşılanmaya çalışılır. Nesne tabanlı programlamada nesneler, özellikleriyle (Properties), işlevleriyle (Methods) ve olaylarıyla (Events) tanımlanır. Nesne tabanlı programlama dillerine örnek olarak ABAP/4, C#, C++, Eiffel, Java, Object Pascal, Objective-C, PHP, Python, REALbasic, Ruby, Simula, Smalltalk ve Visual Basic .NET'i verilebilir.

> ✓ Nesne tabanlı programlama kapsülleme/paketleme, kalıtım ve çok biçimlilik olmak üzere üç prensibe sahiptir.

Nesne tabanlı programlama üç prensibe sahiptir. Bunlar;

- *Kapsülleme/Paketleme (Encapsulation):* Hazırlanan programda kullanıcıyla ilgisi olmayan kodlama kısmının gizlenmesi kapsülleme olarak geçmektedir. Kullanıcı, etkileşim kurduğu aracın istediği sonucu vermesini bekler, aracın nasıl çalıştığıyla ilgilenmez.
- *Kalıtım (Inheritance):* Kalıtım kelimesi programlamada da gerçek anlamında kullanılmaktadır. Programlama açısından bakıldığında oluşturulan bir sınıftan yeni sınıf üretmektir. Yeni sınıf, eski sınıfın özelliklerine sahiptir.
- *Çok Biçimlilik (Polymorphism):* Belli bir işlemin (metodun), farklı nesnelerde veya farklı ortamlarda farklı sonuçlar vermesidir. Örneğin konuşma işlemi, yüksek sesle, normal ses tonuyla ya da kısık sesle olabilir.

Öğrenme Çıktısı

4 İnternet Tabanlı Programlama Dilleri

Araştır 4

İstemci ve sunucu taraflı çalışan internet tabanlı programlama dillerinin farkları nelerdir?

İlişkilendir

Nesne tabanlı programlama ile gerçek dünyada nesnelerin modellenmesini bir örnekle ilişkilendiriniz.

Anlat/Paylaş

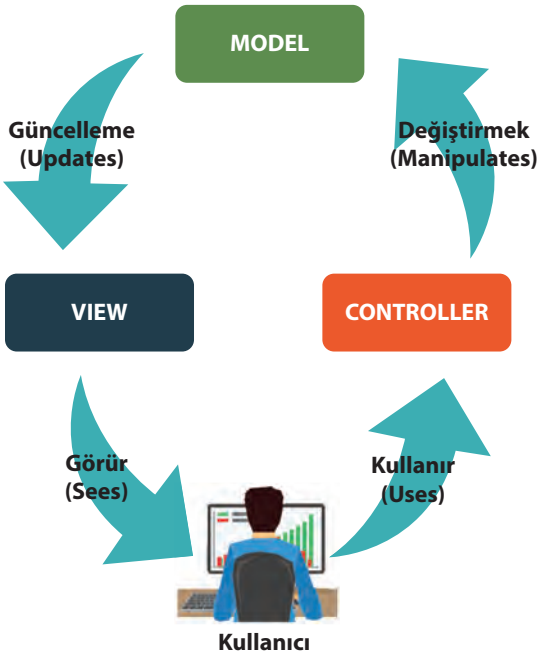
Bulut bilişim teknolojilerinin yaygınlaşmasıyla sunucu taraflı internet tabanlı programlama dillerinin gelişimini ilişkilendirerek açıklayınız.

DİNAMİK WEB

Dinamik Web siteleri, uzmanlık bilgisi gereksinimi olmadan Web sitelerinin içeriklerin güncellenebilmesine dayalı olan yapılardır. Dinamik Web sitelerinin güncellemelerinin kolay bir şekilde yapılabilmesi için kullanıcı panelleri geliştirilmektedir. Bu amaçla geliştirilen panellerde çeşitli mimariler kullanılmaktadır.

Model View Controller (MVC)

Trygve Reenskaug tarafından 1979 yılında yayınlanan Model-View-Controller (MVC) yazılım mühendisliği alanıyla ilgili bir kavramdır. Models (veriler), views (kullanıcı arayüzü) ve controller (denetleyici) terimleri bir araya getirilerek oluşturulan yapıdır (Resim 1.10). MVC, Bilgisayarlarda kullanıcı arayüzü uygulaması için tasarlanan model yazılımıdır.



Resim 1.10 Model View Controller Yapısı

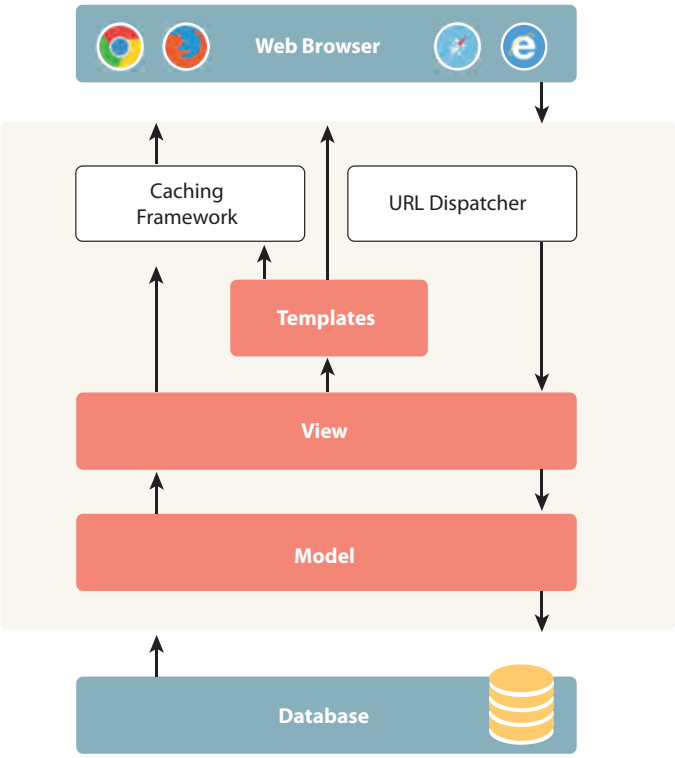
- ***Model (Veriler):*** Model, çalışmada kullanılan nesnelerin oluşturulduğu kısımdır. Günlük hayattaki somut nesnelerin, bilgisayar ortamında modellenmesi anlamına gelir. Modeller arasında birebir haberleşme olmalıdır. Kullanıcıdan controller aracılığıyla gelen verileri işler. Models, verileri, uygulamanın mantığını ve kurallarını yönetir. Models, controller'dan gelen komutlara göre aldığı verileri depolar.
- ***Controller (Denetleyici):*** Kullanıcı ile sistem arasındaki bağı kurar. Girdi görevini üstlenerek ekranda uygun yere ilgili arayüzün ayarlanması için kullanıcılara kolaylık sağlar. Kullanıcıdan gelen girdileri alır ve model veya görünüm için komutlara dönüştürür. Kısacası controller, çalışmalarda veritabanı işlemleri, hesaplamalar, veri aktarımı gibi işlemlerin yapıldığı bölümdür. Controller ayrıca model ve view arasındaki veri akışını da kontrol eder.
- ***View (Kullanıcı Arayüzü):*** Kullanıcıların gördüğü çıktıdır. Bu çıktı bir Web sayfası, masaüstü uygulaması arayüzü veya mobil bir tasarım olabilir. Controller ile gelen ve models'ta işleme tabi tutulan verilerin bir çıktı olarak sunulmasıdır. View'da, models'ta meydana gelen değişimlere bağlı olarak kullanıcılara yeni çıktılar üretilir.

Ruby on Rails, ASP.NET MVC, AngularJS, EmberJS, JavaScriptMVC ve Backbone MVC bileşenlerine izin veren yapılardır. MVC'lerin üstünlükleri arasında proje geliştirme süresini kısaltması, yeniden kullanılabilirlik ve performans sayılabilir.

Model Template View (MTV)

Model Template View (MTV), Model Controller View yapısına benzemektedir. MTV'de View yerine Template tercih edilirken; Controller yerine View tercih edilmiştir (Resim 1.11).

- ***Model:*** Uygulamanın veriye erişim katmanıdır. Veriye nasıl erişileceği, veriler arasındaki ilişkiler gibi veriyle alakalı her şey bu kısımda yer alır.
- ***Template:*** Uygulamanın sunum katmanıdır. Bu katman, Web sayfasında veya diğer tür dokümanlarda verilerin nasıl gösterilmesi gerektiği gibi sunum kısmıyla ilgili kararları içerir. Template kısmında işlenip gelen veriler, kullanılan şablon dili aracılığıyla şekillendirilerek kullanıcılara sunulur.
- ***View:*** Uygulamanın iş mantığı kısmıdır. Bu katman, uygun şablon ve modellere erişimle ilgili mantıksal kısmı içerir. MVC mimari desenin-de, denetleyiciye (controller) denk gelmektedir. Model kısmını uygulamanın iskeleti, View kısmını da beyni olarak düşünülebilir.



Resim 1.11 MTV'nin Çalışma Yapısı

Kaynak: http://allenchien.logdown.com/posts/517242-django-study-notes-1-introduction

MTV mimari desenini kullanan, yüksek seviyeli ve açık kaynaklı yapı olarak Django yapısı karşımıza çıkmaktadır. Instagram, Mozilla, Pinterest ve DisqUS gibi yapılar Django yapısı ile geliştirilmiştir.

yaşamla ilişkilendir

Gerçek hayattan bir örnek ver de MVC nedir tam anlayalım!

Satranç uygulaması yaptığımızı düşünelim… Model'de oyunun "durumu" barındırılacaktır. Oyunun durumunu değiştirecek etkiler (örneğin bir taşın hareketinin doğru olup olmadığı) veya oyunun bitip bitmediği gibi bilgiler model üzerinde yer alacaktır. Kısaca, oyunun tüm bilgileri ve yapılacak işlemlerin validasyonu Model üzerinde barınacak.

View kısmında satranç tahtasının görünümü, yönettiğimiz piyonların şekilleri ve piyonları hareket ettirdiğimizde hangi piyonun nereye gittiğini söyleyen bildirimler yer alacaktır. Piyonların nasıl hareket ettiği, oyunun durumuyla ilgili mantıksal bilgilerin View ile hiçbir işi olmayacaktır. View sadece ve sadece görselliği barındıracaktır.

Controller ise View ve Model arasında haberleşmeyi sağlayacaktır. Örneğin, kullanıcı View'da yer alan "Yeni oyun başlat" butonuna bastığında Controller, Model'e giderek böyle bir isteğin geldiğini söyleyecektir. (Tüm bu işleri yapan Model olacaktır, Controller'ın amacı böyle bir isteğin geldiğini ve alakalı isteğin detaylarını Model'e iletmektir)

Kaynak: http://koddit.com/yazilim/mvc-nedir-gercek-orneklerle-mvc-nedir-anlayalim/

Öğrenme Çıktısı

5 Dinamik Web

Araştır 5

Model View Controller (MVC) ile Model Template View (MTV) arasındaki farklar nelerdir?

İlişkilendir

MVC ve MTV mimarileri ile güncel internet tabanlı programlama dillerini sınıflandırarak ilişkilendiriniz.

Anlat/Paylaş

Pinterest'in hangi dinamik Web mimarisi ile geliştirildiğini açıklayınız.

öğrenme çıktıları ve bölüm özeti

1 İnternet tabanlı programlamayı tanımlayabilme

İnternet Tabanlı Programlama Nedir?

Kullanılan elektronik cihazlara verilen işi yapmaları için komutlar verilmesi gerekmektedir. Bu komutlar kodlar yoluyla gerçekleşmektedir. Elektronik cihazların kodlama yapısına benzer şekilde sanal ortam diye tanımlanan Web ortamında da www'nun gelişmesiyle birlikte internet tabanlı programlama gelişme göstermiştir. İnternet tabanlı programlama ile Web siteleri geliştirilmeye başlanmıştır. Geliştirilen internet tabanlı programlama dilleri kullanılan araçlara ve amaçlara göre farklılaşmakta ve şu an 2500'ün üzerinde internet tabanlı dil olduğu bilinmektedir.

2 İnternet tabanlı programlamanın tarihsel ve yapısal gelişimini açıklayabilme

İnternet Tabanlı Programlamanın Tarihsel ve Yapısal Gelişimi

ABD'de 1960'lı yıllarda savunma sanayiinde kullanılması amacıyla geliştirilen internet teknolojileri 1991 yılına geldiğinde amacının dışında kullanılmaya başlanmış ve www ve HTML dillerinin gelişmesiyle birlikte savunma sanayii dışında da kullanılmaya başlamıştır. Statik yapıda olan internet teknolojileri HTML dilini takiben geliştirilen PHP, ASP gibi internet tabanlı programlama dilleri ile dinamik bir yapıya doğru evrilmiştir. Hem HTML'in hem de dinamik Web siteleri yapmaya olanak sağlayan PHP ve ASP gibi programlama dillerinin geliştirilmesi aşamaları devam etmektedir. Yeni özellikler eklenerek bu dillerin kullanımı yaygınlaştırılmaya çalışılmaktadır. Bu dillerden açık kaynak kodlu olanlarının geliştirilmesi çalışmaları çalışma ekipleri tarafından devam ettirilirken; açık kaynak kodlu olmayan programlama dillerinin geliştirilmesi üretici firmaları tarafından yapılmaktadır.

3 İnternet tabanlı programlama temel kavramlarını karşılaştırabilme

İnternet Tabanlı Programlama Temel Kavramları

İnternet tabanlı programlama denildiğinde programlama dili olmamasına rağmen işaretleme dili olan ve www'nun gelişmesiyle gelişen HTML kavramı bulunmaktadır. HTML içerisinde kullanılan etiketler Web sitesi tasarlarken kolaylık sağlamaktadır. Bunun yanı sıra yine tasarımla ilgili olan ve işaretleme amaçlı kullanılan CSS; Web sitelerinin kullanım amaçlarına göre statik mi yoksa dinamik mi olacağı; Web sitesinde kullanılacak veritabanı; yayınlanacağı yerin hangi hosting olacağı ve Web sitesinin sahip olması gereken domaininin (alan adı) ne olacağı; hazırlanan Web sitesinin Proxy (vekil) sunucuyla mı yayınlanacağı ve Web tarayıcılarında nasıl görüntülendiğinin bilinmesi gerekmektedir. Bu amaçla belirtilen kavramlar önemlidir.

4 İnternet tabanlı programlama dillerini sınıflayabilme

İnternet Tabanlı Programlama Dilleri

İnternet tabanlı programlama diller, betik diller, istemci sunucu taraflı programlama ve nesne tabanlı programlamadır. Betik dili, Web sayfalarında dinamik içerik oluşturmak ve kullanıcılarla iletişim kurmak amacıyla istemci tarafında çalışan bir dildir. İstemci sunucu taraflı programlamada sunucudan internet aracılığıyla talepte bulunur. Sunucu, istemciden gelen talep HTML ise aynen cevap verirken; ASP veya PHP gibi Web tarayıcıda direk görüntülenemeyen programlama diliyle yazılmış kodlar içeriyorsa kendisi yorumlar ve istemciye HTML formatına dönüştürerek cevap verir. Beş duyu organımızla algılayabildiğimiz nesneler dijital ortamda nesne olarak açıklanmakta ve bu nesnelerin modellenmesi ile nesne tabanlı programlama gerçekleştirilmektedir. Nesne tabanlı programlama kapsülleme/paketleme, kalıtım ve çok biçimlilik özelliklerine sahiptir.

5 Dinamik Web yapılarını karşılaştırabilme

Dinamik Web

Dinamik Web sitesi, kullanıcıların Web sitesi geliştirmeyle ilgili bilgileri olmaksızın içerik değiştirmelerine, Web sitesini güncellemelerine olanak sağlayan yapılardır. Web sitelerinin dinamik olması için kullanıcı panelleri geliştirilmekte ve bu panellerde çeşitli mimariler kullanılmaktadır. Model View Controller ve Model Template View bu mimarilerdendir. İkisi benzer yapıda olmakla birlikte sadece isim değişikliği yapılmıştır. Her iki ortamda da geliştirilmesi planlanan Web sitesi veya uygulama için kullanımı kolay arayüz imkânı sunar.

neler öğrendik?

1 Aşağıdakilerden hangisi HTML etiketlerinden biri **değildir?**

A. head B. echo
C. title D. body
E. html

2 Brendan Eich tarafından geliştirilen ve 1995 yılında Netscape firması tarafından piyasaya sürülen programlama dili aşağıdakilerden hangisidir?

A. PHP
B. ASP
C. HTML
D. Javascript
E Ruby on Rails

3 Javascript ile yazılan fonksiyonlar HTML kodlarında hangi etiketler arasına yazılmalıdır?

A. <head> </ head >
B. <body> </body>
C. <a href>
D.
E. <tr> </tr>

4 Kodları "<% %>" arasında yazılan internet tabanlı programlama dili aşağıdakilerden hangisidir?

A. HTML
B. PHP
C. ASP
D. Ruby
E. Django

5 I. Web sayfası yazarı (Author)

II. Web sayfası anahtar kelimeleri (Keywords)

III. Web sayfası hakkında bilgi (Description)

Yukarıdakilerden hangisi ya da hangileri "meta" etiketi içerisinde tanımlanmaktadır?

A. Yalnız I
B. Yalnız II
C. Yalnız III
D. II-III
E. I-II-III

6 Hazırlanan Web sayfalarını ya da dosyaları sunucuda ayrılan hostinge taşımak için hangi protokol kullanılır?

A. FTP B. HTTP
C. SMTP D. TCP
E. IP

7 Aşağıdakilerden hangisi internet tabanlı programlama dillerinden betik dil ailesinde **yer almaz?**

A. Javascript B. Lua
C. PHP D. Python
E. Django

8 Aşağıdakilerden hangisi istemci sunucu taraflı çalışan internet tabanlı programlama dillerinin çalışma yapısını doğru sırada vermektedir?

A. İstemci-Sunucu-Talep-Yanıt
B. İstemci-Talep-Yanıt-Sunucu
C. Sunucu-Yanıt-İstemci-Talep
D. İstemci-Talep-Sunucu-Yanıt
E. Talep-İstemci-Yanıt-Sunucu

9 I. Özellikler (Properties)

II. Olaylar (Events)

III. İşlevler (Methods)

Yukarıdakilerden hangisi ya da hangileri nesne tabanlı internet programlamada kullanılan nesnelerin özelliklerindendir?

A. Yalnız I B. Yalnız II
C. Yalnız III D. I-II
E. I-II-III

10 Aşağıdakilerden hangisi dinamik Web tasarımında kullanılan Model Template View tasarım mimarisinde uygulamanın veriye erişim katmanı kısmıdır?

A. Template
B. View
C. Model
D. Controller
E. Variables

1. B	Yanıtınız yanlış ise "İnternet Tabanlı Programlamanın Tarihsel ve Yapısal Gelişimi" konusunu yeniden gözden geçiriniz.
2. D	Yanıtınız yanlış ise "İnternet Tabanlı Programlamanın Tarihsel ve Yapısal Gelişimi" konusunu yeniden gözden geçiriniz.
3. A	Yanıtınız yanlış ise "İnternet Tabanlı Programlamanın Tarihsel ve Yapısal Gelişimi" konusunu yeniden gözden geçiriniz.
4. C	Yanıtınız yanlış ise "İnternet Tabanlı Programlamanın Tarihsel ve Yapısal Gelişimi" konusunu yeniden gözden geçiriniz.
5. E	Yanıtınız yanlış ise "İnternet Tabanlı Programlama Temel Kavramları" konusunu yeniden gözden geçiriniz.
6. A	Yanıtınız yanlış ise "İnternet Tabanlı Programlama Temel Kavramları" konusunu yeniden gözden geçiriniz.
7. E	Yanıtınız yanlış ise "İnternet Tabanlı Programlama Dilleri" konusunu yeniden gözden geçiriniz.
8. D	Yanıtınız yanlış ise "İnternet Tabanlı Programlama Dilleri" konusunu yeniden gözden geçiriniz.
9. E	Yanıtınız yanlış ise "İnternet Tabanlı Programlama Dilleri" konusunu yeniden gözden geçiriniz.
10. C	Yanıtınız yanlış ise "Dinamik Web" konusunu yeniden gözden geçiriniz.

1 Araştır Yanıt Anahtarı

Araştır 1

Programlama dili programcının belli bir amaç için oluşturulan bir algoritmayı ifade etmek için kullandığı standartlaşmış bir notasyondur. Programlama dili, bilgisayara hangi veri üzerinde hangi koşullarda ne işlem yapılacağını tam olarak anlatmasını sağlar.

Araştır 2

Ziyaret edilen Web sitelerinin internet tabanlı programlama dillerinden hangisiyle yazıldığını anlamak için Web tarayıcılarının adres çubuğuna bakılabilir. Eğer hazırlanan ortam HTML dilinde hazırlanmış ise adres çubuğunda .html veya .htm uzantılı adresler görülür. PHP dilinde hazırlanan bir sayfa ise adres çubuğunda .php uzantılı bir adres görülecektir. ASP ve ASP.NET ile hazırlanan bir sayfayı ziyaret ettiyseniz adres çubuğunda .asp veya .aspx uzantılı bir adres görülecektir. PHP ve ASP kodları HTML sayfa içerisine gömülerek oluşturulan bir sayfa ziyaret edildiği takdirde sayfanın uzantısı yine .html veya .htm olacaktır.

1 Araştır Yanıt Anahtarı

Araştır 3

İşyeri yöneticisinin internet üzerinden satış yapan bir site görevini verdiğinde öncelikle hazırlanacak olan sitenin statik mi yoksa dinamik mi olacağına karar verilir. Devamlı güncelleme gerektiren bir yapıda bir site olacağı için dinamik bir Web sitesi yapımına karar verilmelidir. Dinamik bir Web sitesi yapılacağından dolayı Php ya da asp gibi dinamik Web sitesi yapmaya olanak sağlayan internet tabanlı programlama dillerinden biri seçilebilir. Bu aşamada kişinin hangi programlama dilinin bildiği tercih sebeplerinden biri olabilir. Çalışan kişinin Php programlama diline hakim olduğu düşünülürse Veritabanı olarak MySQL veritabanını seçmesi gerekir. Web sunucusu olarak da Linux işletim sistemine sahip bir hosting alanına sahip olması gerekmektedir. İşyeri çalışanının Asp programlama diline hakim olduğu düşünüldüğünde her şey değişecektir. Öncelikle kullanılacak Veritabanı MsAccess ya da SQL olabilir. veritabanı seçildikten sonra Web sunucunun Windows işletim sistemli bir sunucu olarak seçilmesi ve hosting hizmetinin sağlanması gerekmektedir. Son aşamada ise Web sitesini çağrıştıracak alan adının seçilmesi gerekmektedir.

Araştır 4

İstemci taraflı çalışan internet tabanlı programlama dillerinde HTML, Javascript ve CSS kodları sunucu tarafında yorumlamaya gerek olmadan görüntülenebilmektedir. İstemci HTML bir Web sitesi için sunucudan talepte bulunduğunda sunucu bu Web sitesini hiçbir işleme tabi tutmadan istemciye aynen iletecektir. Fakat sunucu taraflı çalışan ve sunucunun derlemesine, yorumlamasına gerek olan Asp.Net, Php, Ruby ve Jsp gibi dillerle hazırlanan Web sayfaları istemci tarafında direk görüntülenemez. İstemci, bu dillerle hazırlanmış sitelerin isteğini sunucuya iletir ve sunucu bu Web sayfalarını yorumlayarak Html formatında istemciye iletir. Yani istemci taraflı çalışan yapıda sunucu içeriği aynen iletirken; sunucu taraflı çalışan yapıda sunucuda kodların yorumlanması işlemi gerçekleşmektedir.

Araştır 5

Model View Controller ile Model Template View benzer çalışma mantığına sahip dinamik Web geliştirme panelleridir. Kullanıcılar controller ile gerekli değişiklikleri yaparlar. Yapılan değişiklikler bilgilerin tutulduğu model üzerinde değiştirilir ve model gerekli güncellemeleri yaparak görüntüyü view aracılığı ile kullanıcılara aktarır. MVC'nin çalışma mantığı bu şekildeyken aynı çalışma mantığı MTV'de de gerçekleşir. MVC ile MTV'nin tek farkı isimlendirilmeleridir. MVC'de View ile nitelenen yapı MTV'de Template iken; MVC'de Controller diye nitelenen yapı MTV'de View'dur.

Kaynakça

Çamoğlu, K. (2009). Programlama & Veritabanı Mantığı. Kodlab Yayın Dağıtım Yazılım ve Eğitim Hizmetleri, İstanbul.

internet kaynakları

http://hbogm.meb.gov.tr/modulerprogramlar/kursprogramlari/bilisim/moduller/nesnetabanliprogramlama1.pdf

http://www.internetlivestats.com/watch/internet-users/

http://www.w3schools.com/html/default.asp

https://tr.wikipedia.org/wiki/Programlama_dilleri_listesi

Bölüm 2

PHP'ye Giriş

öğrenme çıktıları

1 Sunucu Taraflı Programlama Dilleri

1 PHP'nin çalışma mantığını açıklayabilme
2 PHP alternatiflerini sıralayabilme

2 PHP Destekli Web Sunucu Kurulumu

3 PHP'nin kurulumunu gerçekleştirebilme

3 PHP'nin Temel Kodlama Özellikleri

4 PHP'nin temel kodlama özelliklerini açıklayabilme

4 PHP'nin Yapılandırılması

5 PHP derleyicisini yapılandırabilme

Anahtar Sözcükler: • PHP • Apache • MySQL • WampServer • WampManager • Web Sunucusu • Sunucu Taraflı Kodlama • PHP Yapılandırması

GİRİŞ

Tim Berners Lee tarafından ortaya çıkarılan Web teknolojisi, internetin işlevini önemli ölçüde genişletmiştir. HTML denilen bir işaretleme dili ile oldukça kolay bir şekilde oluşturulup yayınlanabilen sayfalar, biçimlendirilebilen metinlerin yanı sıra çoklu ortam öğelerini de barındırabiliyor olmalarıyla kullanışlı bir yapı sunmaktaydı. Bu yapı sayesinde henüz 1990 yılına gelinmeden, yüz bin bilgisayar tarafından Web sunumu yapılmaya başlandı. Söz konusu dönem için oldukça yüksek sayılabilecek olan bu sayı, Web teknolojisinin geleceğine ilişkin fikir vermeye başlamıştı.

HTML üzerinde yapılan ve günümüzde de sürmekte olan iyileştirmeler, 1992 yılında bir milyon, 1995 yılında da on milyon Web sunucusuna ulaşılmasını sağladığında, Web'in önemi artık gözler önüne serilmişti. Her geçen gün yeni bir işlev kazanan Web sayfalarının günümüzde yüz milyonlarca Web sunucusu üzerinden yayınlanıyor olduğu tahmin edilmektedir.

Web sayfaları, ortaya çıktığı ilk yıllarda sabit bir içerik sunumunun ötesinde bir işleve sahip değildi. Yayıncısı tarafından HTML ile oluşturulmuş olan Web sayfaları, Web sunucusu üzerindeki özel bir klasöre yerleştirilip, bu haliyle istemcilere gönderilmekteydi. Yayıncı tarafından üzerinde değişiklik yapılmadığı sürece değişmeyen bu sayfalar, statik Web sayfaları olarak anılmaktaydılar.

Statik Web sayfaları, uzun bir süre içerik yayıncılığına ilişkin birçok ihtiyacı karşılamış olsa da Web ziyaretçilerinin daha çok etkileşim içinde olabileceği bir Web teknolojisine ilişkin ihtiyaç her geçen gün kendini daha çok belli ediyordu. Günümüzde "sunucu taraflı" olarak sınıflandırılan diller ise, bu ihtiyacı karşılamak için ortaya çıkmıştır. Ziyaretçilere özel sayfalar üretilebilmesine ve ziyaretçilerin içeriğe kolayca katkı sağlayabilmesine olanak sağlayan yapı, sunucu taraflı programlama dilleri sayesinde oluşabilmiştir.

SUNUCU TARAFLI PROGRAMLAMA DİLLERİ

Uzun bir süre statik olarak hizmet veren Web sayfalarının, kullanıcı etkileşimi anlamında iyileştirilmesine yönelik olarak ortaya çıkan sunucu taraflı programlama dilleri, aynı zamanda Web 2.0 olarak anılan teknolojinin temelini oluşturmuştur. Sunucu taraflı programlama dillerinin ortaya çıkması, Web içeriği oluşturan programların yazılmasını sağlayarak dinamik Web sayfalarına kapı açmıştır.

Statik bir Web sayfası ziyaret edildiğinde Web sunucu yazılımı, ziyaret edilen sayfayı içeriğine bakmaksızın karşı tarafa göndermektedir. Sunucu taraflı dil ile kodlanmış olan ve dinamik olarak adlandırılabilecek bir Web sayfası ziyaret edildiğinde ise süreç farklı işlemektedir.

Sunucu taraflı dil ile dinamik olarak oluşturulmuş bir Web sayfası ziyaret edildiğinde, Web sunucu yazılımı söz konusu sayfayı doğrudan istemciye göndermez. İstenen sayfa öncellikle sunucu üzerinde çalışan bir derleyici tarafından derlenir ve çalıştırılır. İstemciye gönderilecek olan içerik ise bu süreçte ortaya çıkar. İstemciye, derlenen sayfanın ürettiği içerik gönderilir. **Derleme** süreci sunucu üzerinde yapıldığı için istemci, sunucu taraflı kodları göremez.

> **Derleme** kavramı, programlama diliyle yazılmış kodların bilgisayar tarafından işlenebilecek hale dönüştürülmesi sürecini ifade eder.

Günümüzde, çalışma mantıkları büyük ölçüde birbirine benzeyen çok sayıda sunucu taraflı dil mevcuttur. Bu dillerin önde gelenlerinden birinin ise, bu kitapta ele alınacak olan PHP olduğu söylenebilir. Ancak PHP'ye geçmeden önce, diğer önemli sunucu taraflı dillerden sayılabilecek olan ASP, JSP, Perl, Pyton, Ruby ve ColdFusion'a değinmek yararlı olacaktır.

- *JSP* : Açılımı JavaScript pages (JavaScript sayfaları) olan bu dil, oldukça güvenilir ve yerleşik bir dil olan Java'ya dayalıdır. Java'ya benzer kod yazım özelliklerini taşıyor olması, Java dili ile kodlamaya alışkın geliştiricileri kendisine çekmiştir. Ayrıca JSP, Java tarafından sağlanan geniş bir kod kütüphanesi ile destekleniyor olduğu için, projelerin hızlıca tamamlanmasını sağlayabilmektedir.
- *ASP* : Active server pages (aktif sunucu sayfaları) ifadesinin içerdiği kelimelerin baş harfleri ile adlandırılmıştır. 1997 yılında Microsoft tarafından ortaya çıkarılmış oldukça eski bir dil olduğu söylenebilir. BA-

SIC dili üzerinde birtakım değişiklikler yapılarak oluşturulmuş olan VBScript diline çok yakın bir yapısı vardır. Microsoft teknolojilerine yakın olan geliştiricilerin büyük bir kısmı, sunucu taraflı programlama gereksinimlerini bu dille karşılamaktadırlar.

- *Perl*: Ortaya çıkan ilk sunucu taraflı dillerden birisi olan Perl'in diğer sunucu taraflı dillere göre daha zor öğrenilebildiği söylenebilir. Bu nedenle yeni geliştiriciler tarafından tercih edilmediği gözlenmektedir. Ortaya çıkışının 1990'lı yıllara dayanması, köklü bir kod kütüphanesine sahip olmasını sağlamıştır.
- *Pyton*: Aslında kökleri 1980'lere uzanan genel amaçlı bir programlama dili olan Pyton, günümüzde genellikle dinamik Web sayfalarının oluşturulması için kullanılmaktadır. Hızlı kod yazımı sağlaması nedeniyle her geçen gün daha da popülerleştiği gözlenen Pyton, önemli modül ve çatılarla desteklendiği için kolay uygulama geliştirme olanağı da sağlamaktadır. Pyton, Google ve YouTube gibi önemli projelerin de arkasında yatan dildir.
- *Ruby*: Genel amaçlı bir dil olması ve günümüzde bir sunucu taraflı uygulamalar için gittikçe popülerlik kazanıyor olması yönüyle Pyton'a benzetilmektedir. Nesne tabanlı yapısı sayesinden modülerlik ve kodların yeniden kullanılabilmesi anlamında kolaylık sağladığı için, projelerin görece hızlı tamamlanmasına olanak vermektedir.
- *ColdFusion*: ColdFusion'un da 1990'lara dayanan oldukça eski bir geçmişinin olduğu söylenebilir. Öğrenilmesi oldukça kolay olmakla birlikte, özellikle veritabanı yönetimi gerektiren projelerde son derece hızlı bir şekilde sonuca ulaşılmasını sağlar ancak, çok esnek bir programlama dili olmadığı söylenebilir.

Çeşitli kaynaklara göre Personel Home Page (Kişisel Ana Sayfa) ifadesinin baş harfleriyle isimlendirilmiş olan PHP'nin geliştirilmesi, Danimarka kökenli bir Kanada'lı olan Rasmus Lerdorf'un 1995 yılında C dili üzerinden gerçekleştirdiği çalışmalarla başlamıştır. 1997 yılına kadar birinci ve ikinci sürümüyle yaygınlaşan PHP, 1997 yılında Zeev Suraski ve Andi Gutmans tarafından büyük ölçüde yeniden yazılmış ve üçüncü sürümüne kavuşmuştur. Bu sürümü itibariyle "PHP:Hypertext Preprocessor" olarak adlandırılan dilin dördüncü sürümü ise 2000 yılında yayınlanmıştır. Halen beşinci sürümüyle var olan PHP, tamamen ücretsiz bir şekilde kullanılmakla beraber hemen her sunucu tarafından destekleniyor olmasıyla, en yaygın sunucu taraflı dillerden biridir. Ayrıca PHP'nin güncel sürümü, kısmen de olsa nesne yönelimli programlamayı desteklemektedir. Sözü edilen özellikleri nedeniyle bazı geliştiriciler tarafından Web sayfası oluşturma görevinin dışına taşınarak, genel amaçlı bir programlama dili olarak da kullanılmaktadır.

Öğrenme Çıktısı

1 PHP'nin çalışma mantığını açıklayabilme
2 PHP alternatiflerini sıralayabilme

Araştır 1

Sunucu taraflı programlama dillerinin çalışma sistemini göz önünde bulundurarak istemci taraflı programlama dillerinin nasıl çalışıyor olabileceğini yorumlayınız.

İlişkilendir

PHP'nin tarihsel gelişiminin genel olarak programlama dillerinin tarihsel gelişimini nasıl etkilediğini araştırınız.

Anlat/Paylaş

Sunucu taraflı diğer internet tabanlı programlama dillerini PHP ile karşılaştırınız.

PHP DESTEKLİ WEB SUNUCU KURULUMU

Günümüzde bilgi işlem kapasiteli hemen her cihazın kolayca bir Web sunucusu haline dönüştürülebilmesi mümkündür. Web sunumu yapmak için yüksek işlem gücüne ve belleğe sahip güçlü bilgisayarların kullanılması zorunlu değildir. Eğer çok yoğun bir ziyaretçi trafiği beklenmiyorsa, ortalama bir masaüstü bilgisayar, dizüstü bilgisayar, tablet hatta akıllı telefon olarak sınıflandırılan bir cep telefonu bile Web sunucusu haline getirilebilir. Bunun için üzerine bir Web sunucu yazılımının yüklenmesi yeterli olacaktır. Web sunucu yazılımını aktif olarak çalıştırmakta olan ve internet üzerinden ulaşılabilen bir cihazın bütün dünyaya Web yayını yapıyor olduğu söylenebilir.

İnternette en çok kullanılan Web sunucu yazılımının tamamen ücretsiz, hızlı çalışan ve esnek bir kullanım sunan Apache adlı yazılım olduğu söylenebilir. Şu an Web sunucularının yarısı, üzerlerinde Apache çalıştırmaktadır. Apache, Unix, Linux, Solaris, Windows, MacOS X ve OS/2 gibi yaygın işletim sistemlerinin hepsini desteklemektedir. www.apache.org adresi üzerinden en güncel sürümü indirilip kolayca kurulabilecek olsa da, Apache'nin tek başına dinamik bir Web sunumu yapması mümkün değildir. Çok gelişmiş bir **HTTP** sunucu olsa da Apache, yalnızca Web istemlerini kabul edip, istenen Web sayfalarını teslim etmekle sınırlı bir göreve sahiptir. Sunucu üzerinde PHP kodlarının derlenip çalıştırılması ise, Apache'den bağımsız olarak geliştirilmiş olan PHP derleyicisinin işidir.

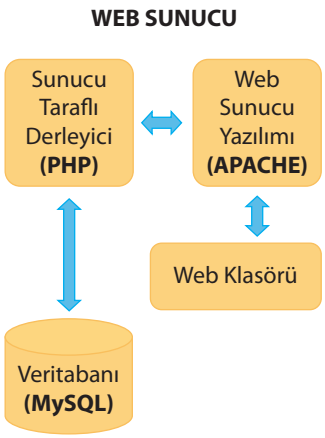
> HTTP, Türkçe karşılığı Hipermetin Transfer Protokolü olan HyperText Transfer Protokol ifadesinin kısaltmasıdır ve Web sayfalarının iletimi için oluşturulmuştur.

Halihazırda Apache kurulu bir bilgisayara PHP derleyicisi kurmak, statik Web sayfası sunumuyla sınırlı becerilere sahip söz konusu bilgisayarın, dinamik Web sayfaları sunabilen bir Web sunucuya dönüşmesini sağlar. www.php.net adresi üzerinden ulaşılabilen PHP organizasyonu Web sayfası, PHP'nin hemen her işletim sistemine yönelik derlenebilecek kaynak kodlarını sunmaktadır. Aynı sitede, Windows işletim sistemi üzerinde çalıştırılabilecek dosyalar da yer almaktadır. Bu dosyaları indirip konumlandırdıktan sonra, Apache'nin konfigürasyon dosyaları, PHP uzantılı dosyaların bu derleyiciye yönlendirilmesi anlamında düzenlenmelidir.

PHP sayfalarını destekleyen bir Web sunucunun üzerinde çalıştırması beklenen diğer bir uygulamanın veritabanı yönetim sistemi olduğu söylenebilir. Hemen her PHP Web projesinde kendini gösteren veri yönetim ihtiyacı, bu şekilde karşılanmış olacaktır. Bu noktada, küçük ve orta ölçekli projelerde yıllardır kullanılmış olmasıyla kendini ispatlayan MySQL adlı veritabanı yönetim sisteminin kullanılması önerebilir. Güncel ihtiyaçları karşılamaya yönelik bir Web sunucuyu oluşturmanın son adımı olan MySQL kurulumu için gereken dosyalar, www.mysql.com adresinden edinilip kurulabilir. MySQL, hemen her işletim sistemi üzerinde çalışabilecek şekilde sürümlendirilmiş bir veritabanı yönetim sistemidir.

Yukarıda da belirtildiği gibi, bir Web sunucunun güncel ihtiyaçları asgari düzeyde de olsa karşılayabilmesi için Web sunucu yazılımına, sunucu taraflı bir derleyiciye ve veritabanı yönetim sistemine ihtiyacı vardır. Sırasıyla Apache, PHP ve MySQL'in bu ihtiyaçları karşılama anlamında ön planda oldukları söylenebilir. Aynı bilgisayara kurulmuş olan söz konusu öğelerin, birbirleriyle Resim 2.1'de görüldüğü gibi etkileşerek, uyum içinde çalışacağı söylenebilir.

Resim 2.1'deki sunucu içinde görülen Web klasörü, internet üzerinden sunumu yapılacak olan dosyaları içermektedir. Birçok formatta olabilecek bu dosyalar, Apache'den Web istemciler aracılığı ile istenilip edinilebilirler. HTTP protokolü üzerinden gerçekleştirilen bu istem, PHP dosyaları için farklı bir sürece sahip olacaktır. PHP ile kodlanmış olan dosyalar istendiğinde Apache, bu dosyaları doğrudan istemciye göndermek yerine, yine sunucu üzerinde çalışan derleyiciye yönlendirir. Derleyicinin derleyip çalıştırdığı PHP sayfası, bu süreçte veritabanı yönetim sistemi



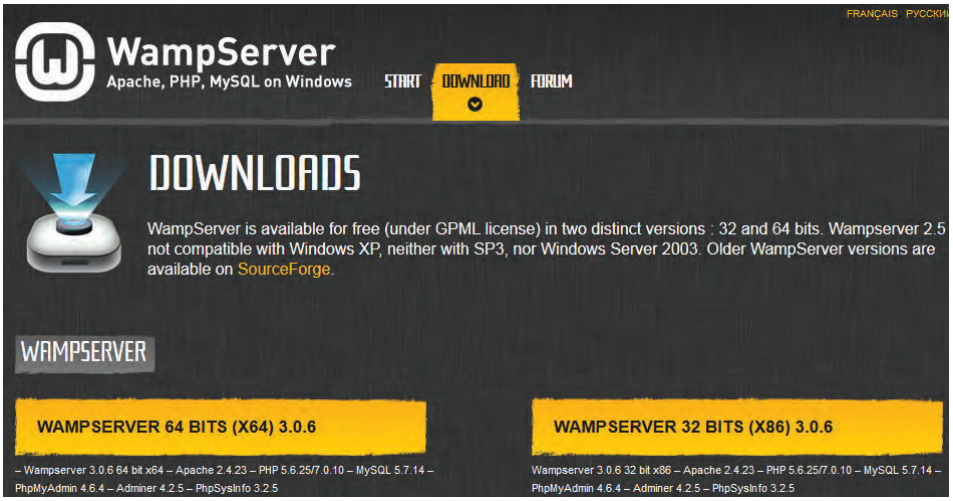
Resim 2.1 Apache, PHP ve MySQL Yazılımları Yüklü Bir Web Sunucusu.

gibi diğer öğelerle etkileşim kurabilme potansiyeline sahiptir. Çalıştırılma süreci tamamlandıktan sonra, PHP dosyası tarafından oluşturulan çıktı, Apache tarafından tekrar teslim alınarak istemciye gönderilir.

Yukarıda anlatılan süreçten de anlaşılacağı üzere PHP ile oluşturulan sayfalar, Web sayfası olmanın ötesinde, Web sayfalarını oluşturmaktadırlar. Bu anlamda PHP sayfası, sayfa üreten sayfa olarak değerlendirilebilir.

Her ne kadar ayrı ayrı edinilip kurulabilseler de Apache, PHP ve MySQL yazılımlarının iyi bir üçlü oluşturması, bu yazılımların üçü birden içeren yazılım paketlerinin ortaya çıkmasını sağlamıştır. Bu paketler sayesinde sunucu yazılımların ayrı ayrı kurulup birbirlerine göre yapılandırılması gibi görece zahmetli bir sürece de gerek kalmamaktadır.

Sözü edilen Web sunucu paketlerinin en çok bilinenlerinden birinin Wamp adlı paket olduğu söylenebilir. Özellikle Windows işletim sistemleri için geliştirilmiş olan WampServer, www.wampserver.com adresinden ücretsiz olarak indirilebilir.

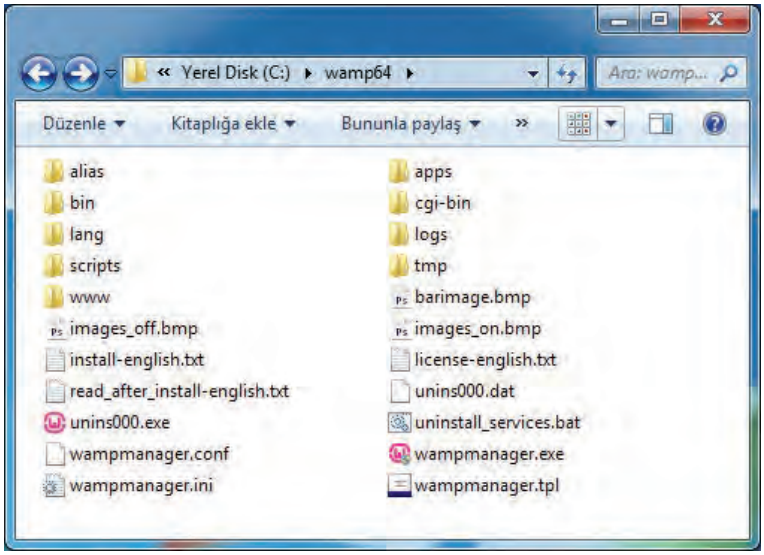


Resim 2.2 WampServer'in Ücretsiz Olarak Edinilebileceği Web Sitesinden Bir Görüntü.

WampServer'in Resim 2.2'de sunulmuş olan Web sitesindeki "Download" etiketli linki, WampServer'in indirme butonlarına ulaştırır. Resmin sağ ve sol alt kısımlarında görüntülenen söz konusu butonlar, Windows'un 32 ve 64 bitlik versiyonlarına uygun WampServer kurulumlarının indirilmesini sağlarlar. Ayrıca her butonun altında, ilgili kurulumun içerdiği yazılımlar, versiyonlarıyla birlikte belirtilmişlerdir.

WampServer'in kurulum dosyası yaklaşık olarak 200 Megabyte büyüklüğe sahiptir. Söz konusu dosyanın indirilip çalıştırılması, kullanıcı dostu bir kurulum arayüzü açacaktır. Bu paragrafın yanındaki karekodun ulaştıracağı videodan da görülebilecek şekilde kurulumu tamamlandıktan sonra, kurulum yapılan klasörün içinde Resim 2.3'te sunulan klasör ve dosyalar oluşturulacaktır.

WampServer klasörünün içindeki "www" isimli klasör ve wampmanager.exe adlı dosyanın önemi, standart bir Web geliştiricisi açısından daha ön plandadır. WWW klasörü, sunumu yapılacak olan Web sayfalarının ve diğer dosyaların yerleştirileceği klasördür. Apache'nin yapılandırmasında herhangi bir değişiklik yapılmadıysa, buraya konumlanan dosyaların internet üzerinden dünyaya yayınlanacağı söylenebilir. O neden kişisel veya özel dosyaların bu klasöre yerleştirilmediğinden emin olunmalıdır. Yayını yapılacak Web sayfaları ve dosyalar için ise doğu yerdir.



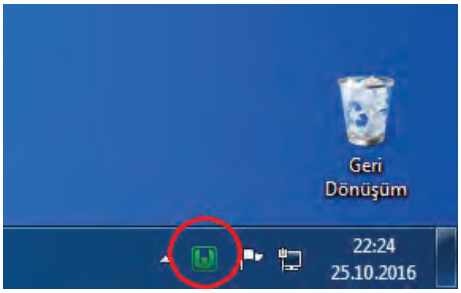
Resim 2.3 WampServer Kurulumu Yapılan Klasörün İçeriği.

WampServer'in www klasöründe çok sayıda dosya bulunabilir. Bir Web sitesi ziyareti sırasında, istenen dosya adres satırı üzerinden belirtildiyse, bu dosya doğrudan karşı tarafa gönderilir. Örneğin "http://www.boteci.com/ogrenciler.html" şeklinde girilmiş bir adres, Web tarayıcı boteci.com alanadı üzerinde bulunan Web sunucusundan "öğrenciler.html" adlı dosyayı ister. Bu noktada Apache, istenen Web sayfasını www klasöründe bulursa, istemi yapan tarayıcıya gönderir. İstenen isme sahip bir dosya yok ise, 404 numaralı "sayfa bulunamadı" hatasını üretir.

Eğer internet kullanıcısı, "http://www.boteci.com" örneğinde olduğu gibi yalnızca alan adı yazıp, herhangi bir sayfa eklemezse, varsayılan olarak "index", "default" veya "main" ismine sahip dosyalar gönderilir. Bu nedenle çoğunlukla, Web sitelerinin ana sayfalarının adı index olarak belirlenmiştir. Bu sayfalar, diğer sayfaları çağırmak ya da diğer sayfalara yönlendirmek gibi görevlere sahiptirler.

> Web istem ve sunumunun yapılması sırasında varsayılan olarak 80 numaralı port kullanılır. Özel durumlar için, yapılandırma ayarları üzerinden söz konusu port numarası değiştirilebilir.

WampServer'in ana klasöründeki "Wampmanager.exe" adlı dosyanın çalıştırılması, Windows işletim sisteminin görev çubuğuna Resim 2.4'te görüntülenen ikonu yerleştirir. Bu ikonun rengi Web hizmetlerinin durumuna ilişkin de fikir vermektedir. İkonun kırmızı renkte olması, söz konusu hizmetlerin kapalı olduğunu, sarı renkte olması başlatılmakta olduğunu, yeşil renkte olması ise servislerin çalışıyor olduğunu gösterir.

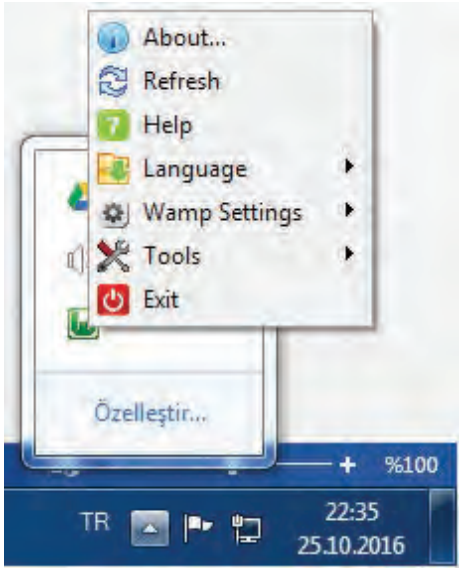


Resim 2.4 Wampmanager.exe Tarafından Oluşturulan Uygulama İkonu.

WampServer ikonunun üzerine *sağ tıklanarak*, Resim 2.5'te sunulan panel görüntülenebilir. WampServer servisleri bu panel üzerinden yönetilebilmektedir. Panelin ilk arayüzünde bulunan komut ve alt menülerin işlev ve içerikleri aşağıdaki gibi maddelendirilebilir.

- *About…*: WampServer'a ilişkin telif ve versiyon bilgilerinin sunulduğu bir pencere görüntüler. Bu pencere, içeriğindeki Apache, PHP ve MySQL gibi yazılımlara ilişkin versiyon bilgilerini de sunar.
- *Refresh*: Apache ve MySQL gibi WampServer servislerinin yeniden başlatılmasını sağlar. Yeniden başlatma süreci WampServer ikonunun renklerinden takip edilebilir.

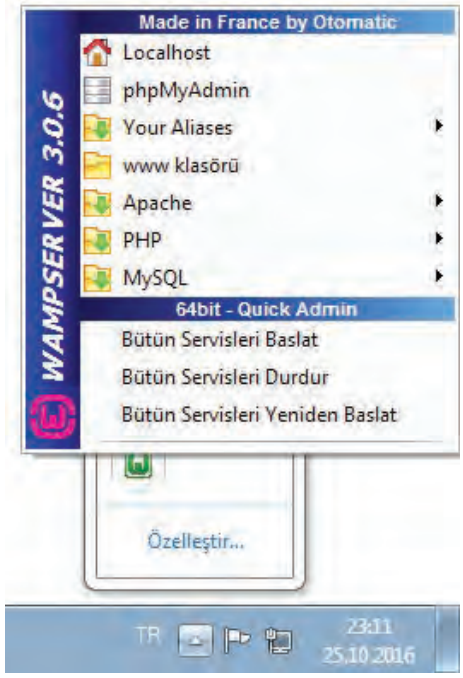
- *Help*: WampSever'a ait forum platformuna yönlendirir.
- *Language*: WampServer arayüzünün sunum dilinin seçilmesini sağlar. Türkçe de seçenekler arasındadır. Ancak varsayılan dil İngilizce olduğu için buradaki anlatım İngilizce arayüz üzerinden yapılmıştır.
- *Settings*: WampServer yazılım paketinin yapılandırmasına ilişkin komutlar barındırır.
- *Tools*: Birtakım servislerin yeniden başlatılmasının ve test edilmesinin yanı sıra, bellek ve kayıtların temizlenmesini sağlayan komutlar içerir.
- *Exit*: WampServer yazılımının çalışmasının sonlandırılmasını ve ikonun yok olmasını sağlar.



Resim 2.5 Wampserver İkonunun Sağ Tıklanması Durumunda Görüntülenen Panel.

Görev çubuğundaki WampServer ikonunun üzerine *sol tıklanması* ise, Resim 2.6'da görüntülenen panelin açılmasını sağlar. Söz konusu panelin açılması için "tek tıklama" eyleminin gerekliliğine dikkat edilmelidir. İkonun çift tıklanması, panelin yerine "hakkında" penceresinin görüntülenmesini sağlayacaktır. Görüntülenen panel, Web sunucunun yönetimi için oldukça önemli komut ve alt menüleri sunar.

Resim 2.6'da görüntülenen panel, sırasıyla aşağıda maddelenen işlevleri sunar.



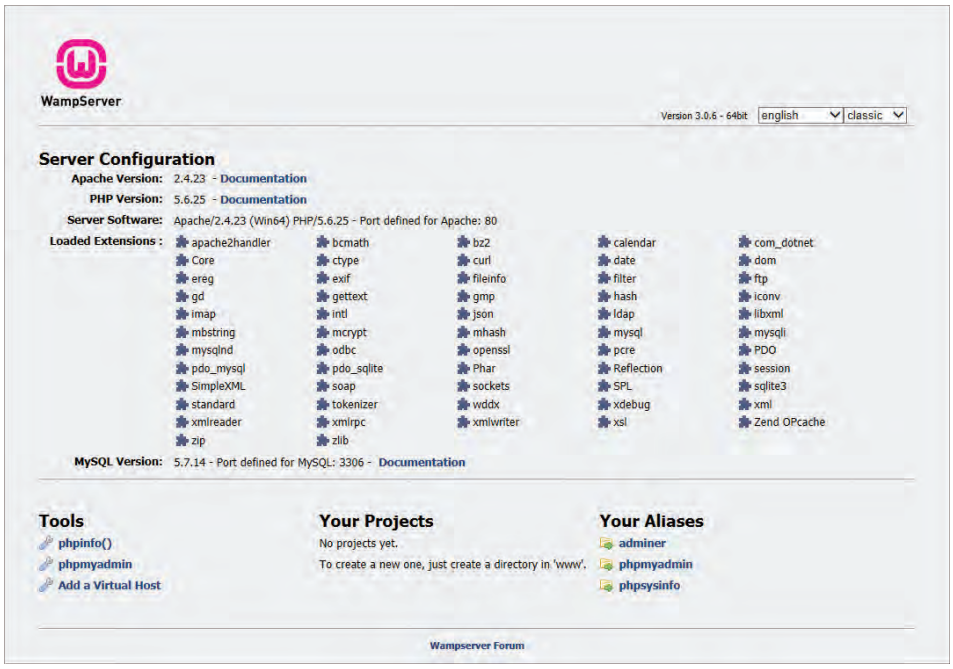
Resim 2.6 Wampserver İkonunun Sol Tıklanması Durumunda Görüntülenen Panel.

- *Localhost*: Web tarayıcısının açılarak WampServer üzerinden sunulan Web sitesinin ana sayfasını görüntülemesini sağlar.
- *phpMyAdmin*: WampServer ile birlikte gelen phpMyAdmin adlı veritabanı yönetim arayüzüne ulaştırır.
- *Your Aliases*: Projeler için ayrı ayrı tanımlanabilecek olan takma adlara (Alias) ulaşılmasını sağlar. Birden fazla Web projesinin aynı anda sunulması gerektiğinde, her bir projeye takma ad atanarak farklı lokasyonlarda barındırılmaları sağlanabilir.
- *www klasörü*: WampServer'in sunulacak Web sayfalarını barındıran klasörünün içeriğini görüntüler.
- *Apache*: Web sunumu yapan Apache adlı yazılımın yapılandırılması için kullanılabilecek komut ve bağlantıları içerir.
- *PHP*: PHP derleyicisinin yapılandırılması için kullanılabilecek komut ve bağlantıları içerir.
- *MySQL*: Veritabanı yönetim sistemi olan MySQL'in yapılandırılması için kullanılabilecek komut ve bağlantıları içerir
- *Bütün Servisleri Başlat*: Apache ve MySQL gibi servislerin tamamının başlatılmasını sağlar.
- *Bütün Servisleri Durdur*: Yukarıda değinilen servislerin çalışmasını sonlandırır.
- *Bütün Servisleri Yeniden Başlat*: Yukarıda değinilen servislerin çalışmasını sonlandırıp, yeniden başlatır.

WampServer kurulduğunda "www" adlı özel klasör, içinde birtakım dosyalarla birlikte gelir. Anlaşılacağı üzere bu dosyalar, silinene ya da değiştirilene kadar, Apache tarafından yayınlanan içeriği oluştururlar. Bu içerik, Web sunucu hakkında hem yazılım hem de donanım anlamında bilgiler sağlayan araçları sunmaktadır.

www klasöründe yer alan ve Apache tarafından sunulan içeriği görüntülemek için Web tarayıcıya sunucunun adresinin yazılması yeterlidir. Buradaki senaryoda, ziyaret edilecek Web sayfasının sunulduğu bilgisayar ile ziyaretin gerçekleştiği bilgisayar aynı olduğu için bir "kendi adresine ulaşma" durumu söz konusudur. "127.0.0.1" şeklindeki ip adresi ve "localhost" şeklindeki alanadı her zaman ve her platformda, kullanıldığı cihazın kendisini gösterir. Bu nedenle Web tarayıcının adres kısmına bu iki ifadeden biri yazılıp Apache tarafından sunulan Web içeriği ziyaret edilebilir.

> Local host ifadesi Türkçe'de "yerel sunucu" anlamına gelmektedir.

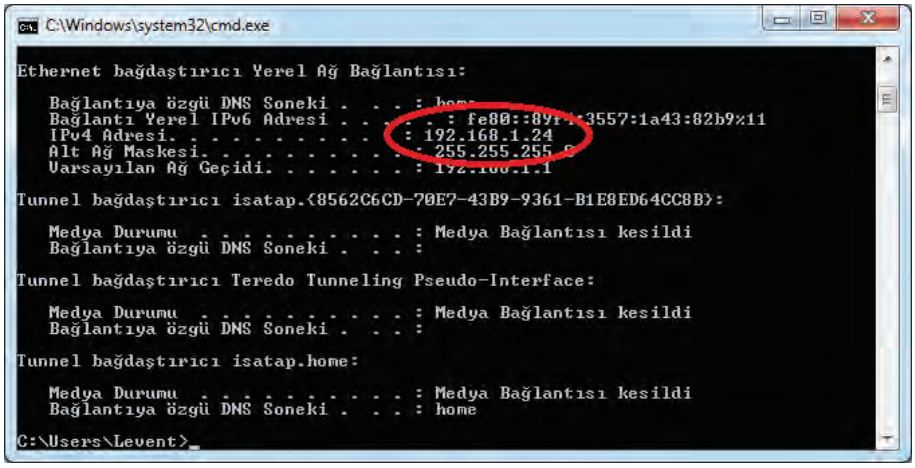


Resim 2.7 WampServer Kurulumu Sırasında Otomatik Olarak Oluşturulan Web İçeriği.

WampServer'ın yeni kurulduğu bir sunucudaki ilk ziyaret, Resim 2.7'de gösterilen sayfayı sunar. Yukarıda da değinildiği gibi bu sayfa, Web sunucusuna ilişkin yazılım ve donanım bilgilerini sunmanın yanı sıra, temel düzeydeki birkaç işlem için araçları barındırır. Resim 7'de de görülebileceği gibi, Yazılımların sürümleri ve içerilen eklentiler bu sayfa aracılığıyla görüntülenebilir. Sayfanın alt kısmındaki bölümde ise, görev çubuğunda yer alan WampServer ikonu ile de ulaşılabilecek olan yönetsel görevlere ulaştıran linkler bulunmaktadır.

Bu noktada, sunulan Web sayfalarına dışarıdan nasıl erişilebileceğine de değinmekte yarar olacaktır. Bir Web sunumuna başka bir bilgisayardan erişim sağlanmak isteniyorsa, Web tarayıcının adres satırına söz konusu sunucunun ip adresi veya alan adı girilmelidir. Windows tabanlı bir bilgisayarda ip adresini öğrenmenin en hızlı yolu, komut satırına ipconfig komutunu girmektir. Bu işlem aşağıdaki adımları takip edilerek gerçekleştirilebilir.

1. Klavyedeki "Windows" tuşuyla birlikte "R" karakterine basılır.
2. Görüntülenen "Çalıştır" penceresine "cmd" yazılıp tamam butonu tıklanır.
3. Açılan komut satırı ekranına "ipconfig" komutu yazılıp *enter* tuşuna basılır.



Resim 2.8 Komut Satırı Arayüzünün "ipconfig" Komutuna Cevaben Ürettiği Çıktı.

Yukarıdaki işlemlerin yapılması, Resim 2.8'dekine benzer bir sonuç üretecektir. Bu görüntü ağ bağlantısına ilişkin temel bilgileri görüntülemektedir. "IPv4 Adresi" etiketinin karşısındaki adres, bilgisayarın IP adresidir. Aynı ağ üzerindeki diğer bilgisayarlar ile oluşturulan Web sunucusuna bu adres üzerinden erişilebilir. Ancak bu noktada, internet üzerinden erişilebiliyor olmanın farklı bir hazırlık gerektirebileceği göz önünde bulundurulmalıdır.

Öğrenme Çıktısı

3 PHP'nin kurulumunu gerçekleştirebilme

Araştır 2

PHP kurulumu için yaygın olarak kullanılan localhost paket programları nelerdir?

İlişkilendir

Sıklıkla birlikte kullanılan Apache, PHP ve MySQL yazılımları arasındaki uyumun nedenlerini değerlendiriniz.

Anlat/Paylaş

Bir localhost paket program kurulum sırasında karşılaştığınız zorlukları ve çözümlerinizi paylaşınız.

PHP'NİN TEMEL KODLAMA ÖZELLİKLERİ

Her ne kadar PHP ile kod yazımı kitabın sonraki bölümünde ele alınacak olsa da ilk PHP kodlarının nasıl yazılıp yayınlanacağına, değişkenlerin kullanımına, sabitlerin oluşturulmasına ve yorum satırlarının eklenmesine burada değinilecek, dolayısıyla PHP ile kod yazımına giriş yapılmış olacaktır.

İlk PHP Sayfası: Merhaba Dünya!

Daha önce de söz edildiği gibi, WampServer'ın kurulduğu klasörün içinde yer alan "www" adlı alt klasör, sunumu yapılacak Web sayfalarını içerir ve başlangıçta bu klasör boş değildir. Yeni bir sayfa oluşturmadan önce, söz konusu dosyanın içini tamamen boşaltmak, başlangıç için daha temiz bir çalışma ortamı sunacaktır. Bu nedenle www klasörünün söz konusu içeriği silinebilir ya da daha sonra kullanılmak üzere başka bir konuma taşınabilir.

Boşaltılmış olan www klasörünün içerisine aşağıda sunulan kodları içeren "index.php" adlı dosyanın yerleştirilmesi, ortalanmış bir şekilde "Merhaba Dünya" metnini içeren bir Web sitesi sunulmasını sağlayacaktır. Hatırlanacağı üzere index isimli sayfa, varsayılan olarak sunulacak sayfadır.

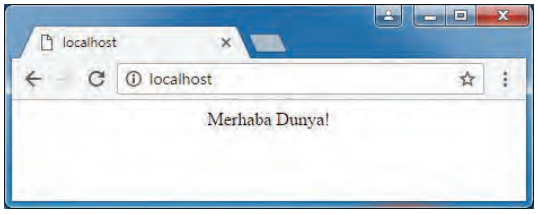
dikkat

Windows işletim sistemi dosya uzantılarını gizleyecek şekilde yapılandırılmış ise, dosya adı yanlışlıkla "index.php" yerine "index.php.txt" şeklinde bırakılabilmektedir. Bu hata, sayfanın görüntülenmemesine neden olacaktır.

```
<?php
    echo "<center>Merhaba Dünya!</center>";
?>
```

Yukarıdaki kodun ilk satırında yer alan "<?php" ifadesi, PHP derleyicisine PHP kodlarının başladığını bildirir. Bu ifadeden başlayarak son satırdaki "?>" ifadesine kadar olan her şey, PHP kodu olarak algılanıp derlenecektir. Dolayısıyla sözü edilen aralığa yazılan kodların yazım hatası gibi hataları kabul etmeyeceği ve yapılan küçük bir hatanın bile sayfanın işletilememesi sonucunu doğurabileceği akılda tutulmalıdır. Yukarıdaki kodun ikinci satırındaki "echo" deyimi ise, istemciye gönderilecek olan Web sayfasına metin yazılmasını sağlar. Örnekteki " Merhaba Dunya! " metni, gönderilecek Web sayfasının içeriğini oluşturur. Oluşturulan içeriği alan Web tarayıcı, " " ve " " ifadelerini HTML etiketi olarak algılayıp, aralarında yazan "Merhaba Dunya!" ifadesini, Resim 2.9'da sunulduğu gibi ortalanmış bir şekilde gösterecektir. Bu noktada, HTML kodlarının da aslında PHP dili ile yazdırılmış olduğuna dikkat edilmesi önemlidir.

Resim 2.9'daki Web sayfasının içi sağ tuşla tıklanıp, kaynağının görüntülenmesi için gereken komut verildiğinde, içeriğin yalnızca * Merhaba Dunya! * şeklinde olduğu görülecektir. Anlaşılacağı üzere, yazılan hiçbir PHP kodu istemciye gönderilmemiştir. Yukarıda da sözü edildiği gibi, bu kodlar sunucuda derlenmiş ve derleme sonucunda üretilen çıktı istemciye gönderilmiştir. Bu sayede yazılan PHP kodlarının da gizli kalması sağlanmıştır.

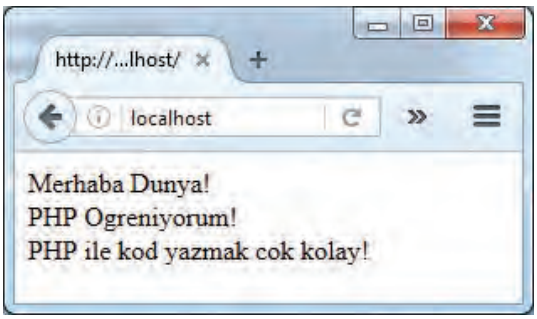


Resim 2.9 Örnek PHP Kodunun Oluşturduğu Web Sayfası.

PHP derleyici, *<?php…?>* ifadelerinin arasında kalmayan hiçbir satırı derlemez. Bu ifadelerin dışındaki her şey, doğrudan çıktı olarak üretilir. Bu özellik sayesinde özellikle Web sayfası oluşturmak üzere yazılacak olan PHP kodları, sayfayı oluşturan içeriğin içine bir anlamda "serpiştirilebilir". Aşağıdaki kodlar, bu durumu örneklendirmektedir.

```
Merhaba Dunya!
<?php
    echo "PHP Öğreniyorum!»;
?>
PHP ile kod yazmak çok kolay!
```

Yukarıdaki satırların ilki, herhangi bir PHP kodu değildir. Dolayısıyla doğrudan istemciye gönderilir. İki ve dördüncü satırdaki ifadeler PHP kodlarının başladığını ve bittiğini belirterek, üçüncü satırın PHP derleyici tarafından derlenmesini sağlarlar. Bu satır *echo* deyimi sayesinde, Web sitesine "PHP Ogreniyorum!" içeriğini ekler. Anlaşılacağı üzere beşinci satır da derlenmeden gönderilecektir. Dolayısıyla bu sayfanın görüntüsü Resim 2.10'daki gibi olacaktır. *<?php ve ?>* ifadeleri kullanılarak istenilen sayıda PHP kod bloğu açılıp kapatılabilir. Böylece bir sayfa, tamamen PHP kodlarından oluşabileceği gibi, yalnızca satır aralarında PHP kodlarını içerebilir.



Resim 2.10 Yalnızca Bir Satırı PHP ile Üretilmiş Olan Örnek Web Sayfası.

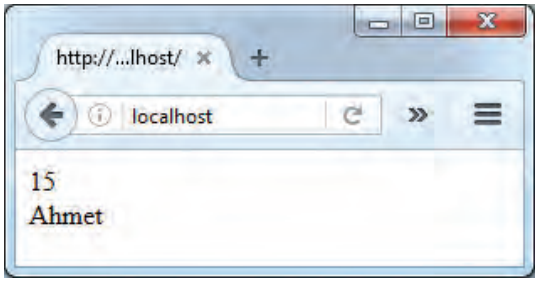
PHP'de Değişken Kullanımı

PHP'nin sağladığı kodlama kolaylıklarından birisi de değişkenlerin kullanımı noktasındadır. Programlamada verilerin yazılacağı bellek hücrelerini temsil eden değişkenlerin oluşturulması, kullanılması veya dönüştürülmesi gibi işlemler, PHP'de oldukça kolaydır. Birçok programlama platformunda bulunan "değişken türü" tanımlaması, PHP'de üzerinde düşünülmesi gereken bir konu değildir. Kodlama sırasında istenen noktada "$" işareti kullanımı ile değişken oluşturulup, içine doğrudan herhangi bir tipteki değer yüklenebilir. Aşağıdaki kod, bu kullanımı örneklendirmektedir.

```
<?php
    $test=10;
    $test=$test+5;
    echo "$test";
    $test="Ahmet";
    echo "$test";
?>
```

Yukarıdaki kodun iki numaralı satırında "test" adlı bir değişken oluşturulmuş ve içine 10 sayı değeri yüklenmiştir. Üç numaralı satırdaki "$test=$test+5;" ifadesi de, bu değişkenin değerini kendisinin 5 fazlası olacak şekilde değiştirmektedir ki bu işlem, içindeki değerin sayı değeri olmaması durumunda mümkün olmazdı. Resim 2.11'deki ekran görüntüsünde de görüleceği üzere dördüncü satır, değişken değerini Web sayfasına 15 olarak yazmıştır.

Beş numaralı satırda, PHP'nin önemli kodlama özelliklerinden biri görülmektedir. Bu satırda, üzerinde tamsayı (integer) tipinde değer taşıyan değişkene metin (string) tipinde değer atanmaya çalışılmıştır. Birçok programlama dilinde buna benzer bir satır, hata mesajıyla karşılaşılmasını sağlamaktadır. Ancak PHP, bu noktadaki tip dönüşümünü otomatik olarak gerçekleştirir. Resim 2.11'teki Web sayfasının ikinci satırında, bu değişkenin metin değerini tanımış olduğu görülmektedir.



Resim 2.11 PHP'de Değişken Kullanımı Örneğinin Oluşturduğu Web Sayfası.

Bir programlama dilinin en önemli öğelerinden biri olan değişkenlerin PHP'deki kullanımı iç içe de geçebilmektedir. PHP'de "değişken değişkeni" şeklinde ifade edilebilecek yapılar da mümkündür. Aşağıdaki PHP kodunu dikkatlice incelemek, bu kullanımın anlaşılmasını sağlayacaktır.

```
<?php
    $deneme="sayi";
    $sayi=15;
    echo $$deneme;
?>
```

Yukarıda, bir değişkenin değerinin başka bir değişkenin ismini tanımlamak için kullanımı örneklendirilmektedir. Resimdeki $deneme adlı değişkenin içindeki "sayi" metni, "$$deneme" şeklindeki tanımın "$sayi" anlamına gelmesini sağlamıştır. Dolayısıyla bu kodun ürettiği sayfada "15" yazacaktır.

PHP'de Sabit Kullanımı

Adından anlaşılacağı üzere sabitler, sahip olduğu değeri değiştirmeyen yapılardır. Genellikle kodların başında tanımlanır ve kod boyunca isimleriyle çağrılarak kullanılırlar. Örneğin iletişim kurulacak bir veritabanının adresi, matematikteki pi sayısı, bir şeyin maksimum ya da minimum değeri sabitler üzerine atanabilir.

```
<?php
    define("MAKSIMUM", 100);
    define("MINIMUM", 0);
    define("PI",3.14159);
?>
```

Anlaşılacağı üzere yukarıdaki kodda, *define()* adlı fonksiyonun kullanımıyla üç adet sabit tanımlanmıştır. Bu tanımlamalardan sonra gelen kodlar için "MAKSIMUM" ifadesinin anlamı 100, MINIMUM ifadesinin anlamı 0, PI ifadesinin anlamı da 3.14159 olacaktır.

dikkat

PHP de yerleşik olarak bulunan "pi()" adlı fonksiyon ve M_PI adlı ön tanımlı sabit, pi sayısını yüksek hassasiyetle sunmaktadır.

PHP Kodlarına Yorum Satırlarının Eklenmesi

Özellikle PHP ile gerçekleştirilen büyük projelerde, kodların arasına yalnızca geliştiriciler tarafından görülebilecek notların alınması gerekmektedir. Yorum satırı olarak da anılan bu satırlar, birçok dilde olduğu gibi PHP tarafından da desteklenmektedir. PHP'de "//" işaretli bir satırda işaretin sağında kalan metin, geliştirici notu olarak değerlendirilir ve derleyici tarafından göz ardı edilir. Aşağıda sunulan kodun üçüncü satırı bunu örneklendirmektedir.

Birden fazla yorum satırı eklemek istendiğinde, her satırın başına "//" işaretini koymak yerine "/*" işareti kullanılabilir. Derleyici, "/*" işareti ile "*/" işareti arasında kalan metnin tamamını göz ardı edecektir. Aşağıda sunulan kodun beş ile onuncu satırları arasında bu kullanım örneklendirilmektedir.

```
<?php
    echo "Merhaba Dunya!";
    // Bu satır PHP derleyici tarafından göz ardı edilir.
    echo "PHP Ogreniyorum!";
    /*
    Bu satırlar
    PHP derleyici
    tarafından
    göz ardı edilir.
    */
?>
```

yaşamla ilişkilendir

Neden PHP?

Çünkü PHP; stabildir, güvenlidir, açık kaynak kodludur, tüm dünyada binlerce özgür programcı tarafından geliştirilir, kodlaması kolay ama buna karşılık çok gelişmiştir. Web Siteniz için PHP tüm ünlü web sitelerinin (Facebook, sahibinden.com vb.) yaptığı seçim gibi en doğru seçim olacaktır. PHP en yüksek web yazılımcı kitlesine sahip bir programlama dilidir. Gerek dökümantasyonu gerek uzman sayısı diğer web programlama dilleri ile kıyaslanamaz düzeydedir.

Eğer PHP'yi öğrenmeyi ve ileride bunu iş veya hobi olarak benimsemeyi istiyorsanız bu da doğru bir seçim olacaktır. Çünkü iş ilanları web sitelerini incelediğinizde göreceksiniz ki web programlama alanında PHP açık ara önde, daha çok eleman talebi bulunmaktadır. Hobi açısından bakıyorsanız da maliyetlerinin düşük olması sebebi ile kolaylıkla web siteleri üretebilir ve bunları yönetebilirsiniz.

(Bir web sitesi yapmak için PHP bildikten sonra ücretini ödeyeceğiniz sadece bir Linux sunucu ve bir alan adıdır.)

Kaynak: http://www.phpprogramlama.com/neden-php.php

Öğrenme Çıktısı

4 PHP'nin temel kodlama özelliklerini açıklayabilme

Araştır 3

"Merhaba Dünya!" metnini ortalanmış, kalın harflerle ve yeşil renkte içeren Web sayfasını oluşturan PHP kodunu yazınız.

İlişkilendir

PHP'nin kodlama yapısının avantaj ve sınırlıklarını diğer internet tabanlı programlama dillerinin kodlama yapılarıyla ilişkilendirerek karşılaştırınız.

Anlat/Paylaş

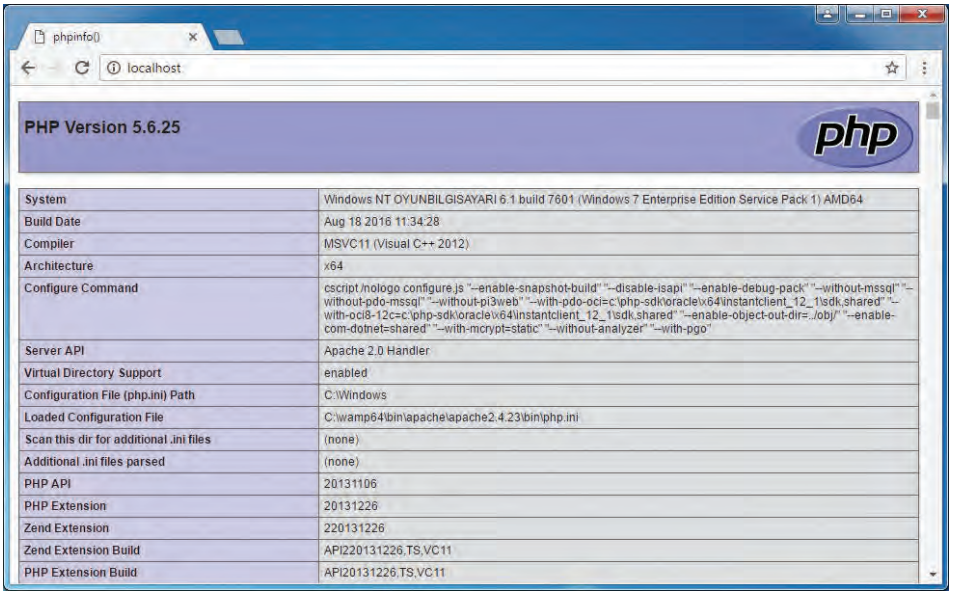
PHP ile ilk "Merhaba Dünya" sayfasını kodlama deneyimlerinizi paylaşınız.

PHP'NİN YAPILANDIRILMASI

PHP'nin birbirinden oldukça farklı sayılabilecek amaçlarla kullanılabiliyor olması, yapılandırma ayarlarının son derece esnek bir şekilde değiştirilebilmesinden kaynaklanmaktadır. PHP, Web ya da Web harici projeleri kapsayan pek çok amaca göre yapılandırılabilmektedir. Ayrıca PHP, varolan yapılandırmanın listelenmesini sağlayan "phpinfo()" adlı bir fonksiyon barındırır. Aşağıda bu fonksiyonun basitçe kullanımı görülmektedir.

```
<?php
    phpinfo();
?>
```

Yukarıdaki kodun oluşturduğu sayfanın ziyaret edilmesi, Resim 2.12'de sunulan görüntüyü oluşturacaktır. Sol taraftaki kaydırma çubuğunun boyutundan da anlaşılacağı üzere bu sayfa, halihazırdaki PHP yapılandırmasına ilişkin hemen her veriyi içeren oldukça dolu bir içeriğe sahiptir. Herhangi bir yapılandırma değişkeninin değeri, bileşenlerin versiyon numaraları veya bir modülün yüklü olup olmadığı gibi bilgilere bu sayfa aracılığıyla ulaşılabilir.

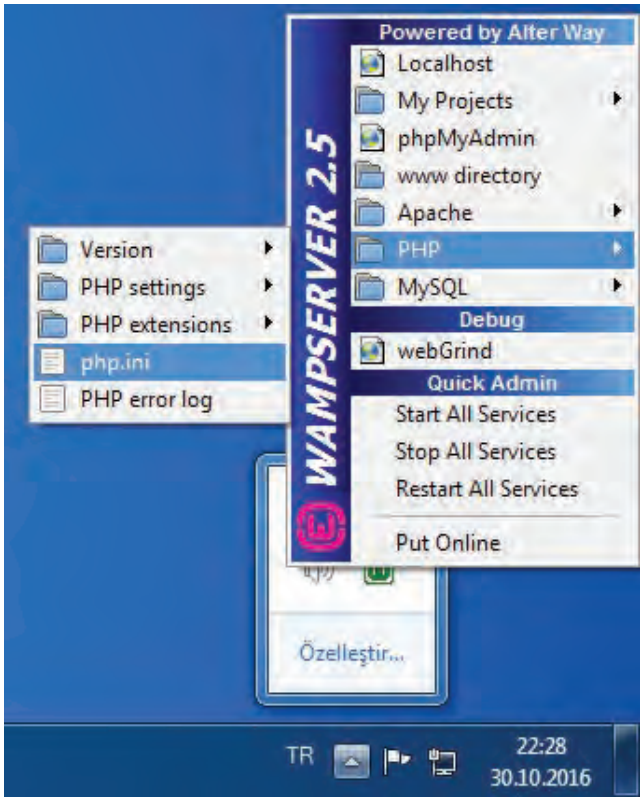


System	Windows NT OYUNBILGISAYARI 6.1 build 7601 (Windows 7 Enterprise Edition Service Pack 1) AMD64
Build Date	Aug 18 2016 11:34:28
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x64
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=c:\php-sdk\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-sdk\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--with-mcrypt=static" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\wamp64\bin\apache\apache2.4.23\bin\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,TS,VC11
PHP Extension Build	API20131226,TS,VC11

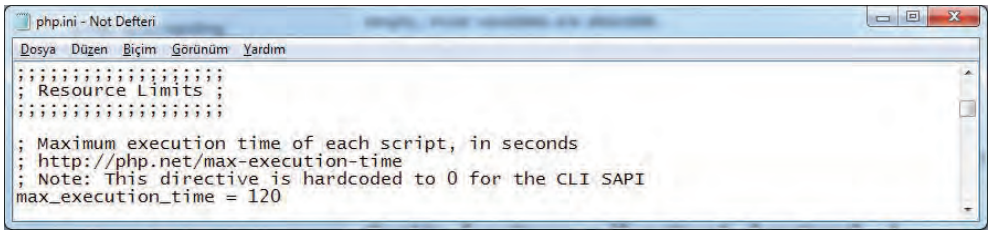
Resim 2.12 PHP'nin Yapılandırma Özelliklerini Sunan Sayfa.

PHP'nin, phpinfo sayfası üzerinde görüntülenen yapılandırma ayarlarının değiştirilmesi için, yapılandırma dosyasına ulaşmak ve gerekli değişiklikleri yapmak yeterlidir. "php.ini" adıyla kayıtlı olan bu dosya, PHP'nin çalışma amacına göre özelleştirilebilmesi açısından oldukça önemlidir. Sunucu üzerinde yapılabilecek basit bir aramayla ulaşılabilecek olan "php.ini" dosyasına, görev çubuğunda yer alan WampServer ikonu üzerinden de ulaşılabilir (Resim 2.13).

Resim 2.14'te örnek bir görünümü verilen php.ini dosyası üzerindeki satırları değiştirerek veya yeni satırlar ekleyerek PHP derleyicisinin çalışma yapılandırılması önemli ölçüde değiştirilebilir. Ancak bu dosya, yapılandırma satırları içerdiği gibi yapılandırma satırlarına ilişkin açıklamalar da içerebilmektedir. ";" karakteri ile başlayan satırlar, PHP yapılandırmasının okunması sırasında dikkate alınmazlar. Ancak Resim 2.14'deki örneğin son satırında da görüldüğü gibi, ; ile başlamayan satırlar yapılandırma değişkenlerini ve bu değişkenlerin değerlerini içerirler.



Resim 2.13 php.ini Adlı Dosyaya WampServer İkonu Üzerinden Ulaşılması.



Resim 2.14 php.ini Adlı Dosyadan Örnek Bir Görünüm.

Yapılandırma dosyasında değişiklikler yapılarak PHP'ye ilişkin yüzlerce özellik değiştirilebilir. Ünitenin devamında, söz konusu değişikliklerin neler olabileceği örneklendirilecektir.

Yukarıda değinildiği gibi, PHP kodlarının başladığını ve bittiğini belirtmek üzere "<?php" ve "?>" etiketleri kullanılmaktadır. *php.ini* adlı dosyaya yer alacak "short_open_tag = On" ifadesi, "<?" etiketinin de PHP kodlarının başlangıcını belirtmesini sağlayacaktır. Bu ifadede yer alan "On" değerinin "Off" olarak değiştirilmesi ise, anlaşılacağı üzere "<?php" ifadesini zorunlu kılacaktır.

PHP derleyicisinin derlediği kodları çalıştırma anlamındaki zaman sınırı da php.ini dosyası üzerinden değiştirilebilir. Bunun için "max_execution_time" ifadesinin değeri değiştirilmelidir. Bu ifadenin karşısında yazan değer, saniye cinsinden maksimum çalışma süresini ifade eder. Örneğin "max_execution_time=60" şeklindeki kullanım, derlenmiş bir PHP kodunun en fazla 60 saniye çalışmasını sağlar. Söz konusu sürenin dolması durumunda, kod kendi çalışmasını henüz sonlandırmamış bile olsa, sistem tarafından sonlandırılır.

PHP tarafından verilen hata mesajlarına ilişkin yapılandırma için en sık kullanılan değişken "error_reporting" adlı olanıdır. Bu değişkenin karşısına yazılan ifade, PHP derleyicisinin derleme sırasında

oluşturduğu hataların, uyarıların ya da bildirimlerin görüntülenme durumunu değiştirecektir. Örneğin "error_reporting = E_ALL" şeklindeki kullanım hata, uyarı ve bildirimlerin görüntülenmesini sağlar. "error_reporting = E_ALL & ~E_NOTICE" şeklindeki kullanım ise, bildirimlerin (notice) dışındaki mesajların, yani yalnızca hata ve uyarıların görüntülenmesini sağlar.

"file_uploads" değişkeninin karşısına yazılacak "On" değeri, PHP ile sunucuya dosya yükleme olanağı sağlanmış olur. Söz konusu değişkenin "Off" değerini alması ise, PHP ile dosya alımının engellenmiş olması anlamına gelir.

PHP ile oluşturan sayfayı ziyaret eden kullanıcıların tarayıcılarındaki durma butonuna basması, farklı bir adrese yönlenmesi veya tarayıcısını kapatması gibi durumlarda, henüz derlenip çalıştırılmakta olan PHP kodlarının çalıştırılmaya devam edip etmemesine yönelik bir yapılandırma da söz konusudur. php.ini adlı dosyaya eklenecek "ignore_user_abort" ifadesinin karşısına yazılacak değer, bunun ifade edilmesini sağlayacaktır. "ignore_user_abort=On" şeklindeki kullanım, kullanıcı ziyaretini sonlandırsa bile PHP kodunun çalışmaya devam etmesini sağlayacaktır. "ignore_user_abort=Off" ifadesi ise, kullanıcının ziyaretini sonlandırması durumunda PHP kodunun derlenmesini de sonlandıracaktır. PHP'nin yapılandırma ayarlarıyla ilgili detaylı bilgi edinmek için bu paragrafın yanındaki karekodun ulaştırdığı sayfa incelenebilir.

Öğrenme Çıktısı

5 PHP derleyicisini yapılandırabilme

Araştır 4

PHP hatalarının sistem günlüğü dosyasına kaydedilmemesini sağlamak için gereken yapılandırma düzenlemesini gerçekleştiriniz.

İlişkilendir

phpinfo(); fonksiyonunu kullanarak kurulumunu yaptığınız localhost paket programındaki yapılandırma bilgilerinin dökümünü görüntüleyiniz.

Anlat/Paylaş

PHP yapılandırma süreçlerinde karşılaştığınız zorlukları ve çözümlerinizi paylaşınız.

1 PHP'nin çalışma mantığını açıklayabilme

2 PHP alternatiflerini sıralayabilme

Sunucu Taraflı Programlama Dilleri

1 PHP, sunucu taraflı çalışan bir dildir. Sunucu taraflı diller ile oluşturulmuş bir Web sayfaları ziyaret edildiğinde, Web sunucu yazılımı söz konusu sayfayı doğrudan istemciye göndermez. Bu yapıda istemi yapılan sayfa, sunucu üzerinde yer alan bir derleyiciye yönlendirilir. Sayfayı oluşturacak olan kodları alan derleyici, söz konusu sayfayı derleyip çalıştırır. Sözü edilen çalışma sürecinde kodlar tarafından üretilen çıktı istemciye gönderilmek üzere web sunucu yazılımına teslim edilir. İstemciye ise, sunucu üzerinde çalışan kod tarafından üretilen söz konusu sayfa gönderilir. Böylece istemci, sunucuda derlenen kodları değil, bu kodların ürettiği içeriği görür. Bu yapı, sayfayı oluşturan kodları istemciye hiç göndermeyeceğinden dol dolayı sayfa ziyaretçileri tarafından görünmemesini sağlayacağı için, kaynak kodu gizliliğini de sağlamaktadır.

2 PHP'nin dışında altı önemli sunucu taraflı dilden bahsedilebilir. Bu dillerin başında Java'ya benzer kod yapısıyla ASP, ilk sunucu taraflı dillerden biri sayılabilecek Perl, hızlı kod yazmaya uygun yapısıyla Python ve oldukça modüler kod sistemi sağlayan Ruby gelmektedir. Ayrıca yine Java'ya dayalı JSP'nin ve kolaylıkla öğrenilebilme özelliğine sahip olan ColdFusion'un da önemli sunucu taraflı dillerden olduğu söylenebilir. Söz konusu dillerin hepsi sunucu taraflı çalışması yönüyle birbirlerine benziyor olsalar da, kod yazım yapısı ve işlev anlamında farklılık göstermektedirler.

3 PHP'nin kurulumunu gerçekleştirebilme

PHP Destekli Web Sunucu Kurulumu

3 PHP derleyiciye sahip bir Web sunucusunun kurulumu öncelikle bir Web sunucu yazılımının kurulumunu gerektirir. Apache adlı Web sunucu yazılımı bu amaç için oldukça uygun görünen ücretsiz bir yazılımdır. Ancak Web sunucusu olacak bilgisayara Apache'nin kurulmuş olması PHP kodlarının çalıştırılıp gönderilmesi için yeterli olmayacaktır. Apache üzerinde birtakım yapılandırma işlemleri gerçekleştirmek gerekecektir. Sözü edilen süreçleri işletmeden kurulum yapılmak isteniyorsa ApacheTriad, WampServer veya XampServer gibi yazılım paketleri tercih edilebilir. Bu paketler; Apache, PHP derleyici ve MySQL gibi önemli sunucu uygulamalarını barındırmakla beraber, kurulum sonrası yapılandırmalarını da otomatik olarak gerçekleştirirler. Web sitelerinden kurulum dosyaları indirilip, standart kurulum süreçleriyle kolayca kurulabilirler. Sözü edilen paketlerden oldukça popüler olanı sayılabilecek WampServer, www.wampserver.com adresi üzerinden ücretsiz olarak edinilebilir. Kullanıcı dostu kurulum arayüzü ile kurulumu tamamlanan WampServer, görev çubuğuna yerleştirdiği ikonu tıklanarak yönetilebilir.

öğrenme çıktıları ve bölüm özeti

4 PHP'nin temel kodlama özelliklerini açıklayabilme

PHP'nin Temel Kodlama Özellikleri

4 Derleyiciye, PHP kodlarının başladığını bildirmek amacıyla "<?php" etiketi kullanılır. Derleyicinin PHP kodlarının bittiğini anlaması için ise "?>" etiketi kullanılmalıdır. Derleyici, bu etiketler arasında bulunmayan herzeyi derlemeden istemciye gönderir. Böylece PHP kodları, diğer kodların arasına serpiştirilebilir. PHP, diğer programlama dillerinin sahip olduğu özelliklerin hemen hemen tamamına sahiptir. Bu kapsama temel programlama özellikleri olan değişkenler, sabitler ve yorum satırları gibi kullanımlar da dahildir. PHP için değişkenler için tip tanımlaması ve tip dönüşümü gibi işlemlerin yapılmasına özel durumların haricinde gerek yoktur. Derleyici, bu işlemleri otomatik olarak gerçekleştirmektedir. Sabitler ise, C ve Java gibi dillerle oldukça benzer bir şekilde oluşturulmaktadır. Kodların arasında yer alması istenen geliştirici notları için diğer birçok dilde olduğu gibi "//" ve "/*" gibi işaretlemelerin kullanımı mümkündür. "//" işareti, bulunduğu satır için sağında kalan her şeyin geliştirici notu olarak görülmesini ve derlenmemesini sağlar. "/*" ve "*/" işaretleri arasında kalan bir ya da birden fazla satır için de aynı şey geçerlidir.

5 PHP derleyicisini yapılandırabilme

PHP'nin Yapılandırılması

5 PHP derleyicisinin yapılandırma özellikleri, phpinfo() adlı PHP fonksiyonu ile görüntülenebilir. Bu fonksiyonun oluşturduğu web sayfası, PHP derleyicisinin sahip olduğu özellikleri ve derleyicinin yapılandırma özelliklerini oldukça detaylı bir şekilde sunacaktır. Söz konusu özellikleri değiştirmek için ise "php.ini" adlı yapılandırma dosyası üzerinde değişiklik yapılmalıdır. Bu dosyada yer alan ve ";" karakteri ile başlayan metinler, yapılandırma değişkenlerine ilişkin bilgi sunmaktadırlar.

1 Aşağıdakilerden hangisi sunucu taraflı bir dil **değildir?**

A. HTML
B. Perl
C. ASP
D. PHP
E. Python

2 Aşağıdakilerden hangisi VBScript diline çok yakın bir yapısı olan sunucu taraflı dildir?

A. JSP
B. ASP
C. PHP
D. Perl
E. Python

3 Aşağıdakilerden hangisi PHP derleyicisi, Apache ve MySQL yazılımlarını içeren bir Web sunucu yazılım paketidir?

A. Office
B. Perl
C. Pyhton
D. WampServer
E. Linux

4 Web tarayıcının üzerinde çalışmakta olduğu bilgisayardaki Web sunucusuna ulaşmasını sağlamak için adres satırına aşağıdakilerden hangisi yazılabilir?

A. Droppoint
B. Python
C. Legalnetwork
D. DistanceWeb
E. Localhost

5 WampServer yüklü bir bilgisayarda, sunumu yapılacak olan Web sayfaları hangi isme sahip klasöre yerleştirilmelidir?

A. www
B. local
C. htmlpages
D. server
E. wampdir

6 Aşağıdakilerden hangisi PHP'de değişkenleri ifade eden özel karakterdir?

A. &
B. #
C. //
D. $
E. ?>

7 PHP değişkenlerinin isimlendirmesine ilişkin aşağıdaki ifadelerin hangisi doğrudur?

A. Sayı ile başlayabilirler.
B. Boşluk içerebilirler.
C. Çift tırnak içerebilirler
D. Alt çizgi karakteri içerebilirler
E. // işareti ile başlamalıdırlar.

8 Aşağıdakilerden hangisi PHP'nin yapılandırma özelliklerinin görüntülenmesini sağlayan fonksiyondur?

A. echo
B. phpinfo
C. config
D. define
E. perl

9 Aşağıdakilerden hangisi PHP sabitlerinin tanımlanması için kullanılan fonksiyondur?

A. phpinfo
B. config
C. define
D. www
E. xamp

10 PHP kodlarının arasına yerleştirilecek olan birden fazla satıra sahip yorumlar hangi işaretle başlatılmalıdır?

A. <?php
B. ?>
C. //
D. /*
E. $

neler öğrendik yanıt anahtarı

1. A	Yanıtınız yanlış ise "Sunucu Taraflı Programlama Dilleri" konusunu yeniden gözden geçiriniz.
2. B	Yanıtınız yanlış ise "Sunucu Taraflı Programlama Dilleri" konusunu yeniden gözden geçiriniz.
3. D	Yanıtınız yanlış ise "PHP Destekli Web Sunucu Kurulumu" konusunu yeniden gözden geçiriniz.
4. E	Yanıtınız yanlış ise "PHP Destekli Web Sunucu Kurulumu" konusunu yeniden gözden geçiriniz.
5. A	Yanıtınız yanlış ise "PHP Destekli Web Sunucu Kurulumu" konusunu yeniden gözden geçiriniz.
6. D	Yanıtınız yanlış ise "PHP'nin Temel Kodlama Özellikleri" konusunu yeniden gözden geçiriniz.
7. D	Yanıtınız yanlış ise "PHP'nin Temel Kodlama Özellikleri" konusunu yeniden gözden geçiriniz.
8. B	Yanıtınız yanlış ise "PHP'nin Yapılandırılması" konusunu yeniden gözden geçiriniz.
9. C	Yanıtınız yanlış ise "PHP'nin Temel Kodlama Özellikleri" konusunu yeniden gözden geçiriniz.
10. D	Yanıtınız yanlış ise "PHP'nin Temel Kodlama Özellikleri" konusunu yeniden gözden geçiriniz.

2 Araştır Yanıt Anahtarı

Araştır 1

Sunucu taraflı dillerde yazılmış kodların sunucu üzerinde derleniyor olması gibi, istemci taraflı dillerle yazılmış kodlar istemci bilgisayarı üzerinde derlenir. JavaScript dilinin Web sayfalarında kullanımı buna örnek teşkil etmektedir. JavaScript ile Web sayfasına yazılmış olan kodlar, sunucu üzerinde herhangi bir işlem görmeksizin istemciye gönderilirler. Web istemini yapan tarayıcı, JavaScript kodlarını yükledikten sonra derler ve çalıştırır. Yani istemciye kodların çıktıları değil, kendisi gönderilir. Bu durum, istemi yapan kişinin kodları görebileceği anlamına da gelmektedir.

Araştır 2

Apache, PHP ve MySQL yazılımları ayrı kurulabileceği gibi üçünü birden içeren yazılım paketleri de bulunmaktadır. Bu paketler sayesinde sunucu yazılımların ayrı ayrı kurulup yapılandırılması gerekmemektedir. Yaygın olarak kullanılan localhost programlarına WampServer, XAMPP, LAMP Server, Zend Server ve Easy PHP örnek verilebilir.

Araştır 3

Web içeriğinin biçimlendirilmesi HTML ile gerçekleştirilebilecek bir işlemdir. Ancak HTML kodlarının PHP ile oluşturulduğu düşünülürse, PHP ile dolaylı bir biçimlendirme yapılabileceği söylenebilir. Bu bağlamda, aşağıdaki "echo" komutu ile sayfaya basılan içerik, Web tarayıcısının ortalanmış, kalın harflerle ve yeşil renkte bir "Merhaba Dünya!" metni görüntülemesini sağlayacaktır.

```
<?php
echo "<center><b><font color='green></font></b></center>'";
?>
```

Araştır 4

Basit bir internet araştırmasıyla görülebileceği gibi, PHP tarafından yakalanmış hataların sistem günlüğüne (log) kaydedilmesiyle ilgili değişken, php.ini adlı dosyada yer alan "log_errors" adlı değişkendir. Varsayılan olarak karşısında "On" yazan bu değişkenin değerini "Off" olacak şekilde değiştirmek, PHP tarafından yakalanan hataların sistem günlüğüne yazılmamasını sağlayacaktır.

Kaynakça

Balaban, E. (2013). *PHP*. İstanbul: Pusula yayıncılık.

Çakır, S. (2015). *Oku, İzle, Dinle, Öğren! PHP* 5.4. İstanbul: Kodlab

Çelik, R. (2015). *A'dan Z'ye PHP*. İstanbul: Seçkin yayıncılık.

Satman, A. G. (2014). *Uygulamalarla PHP* 6.3. İstanbul: Dikeyeksen yayınevi.

Şamlı, M. (2010). *Her Yönüyle PHP 6*. İstanbul: Kodlab.

Bölüm 3

PHP ile Dinamik Web Kodlama

öğrenme çıktıları

1 Değişkenler ve Değişken Türleri
1 Değişken türleri ve dizileri kullanabilme

2 HTML Formları
2 HTML formlarından gelen verileri kullanabilme

3 Kontrol Yapıları
3 Kontrol yapılarını kullanabilme

4 Fonksiyonlar
4 Fonksiyon tanımlaması yapabilme

5 Sınıflar
5 Sınıf tanımlaması yapabilme

6 Veritabanı İşlemleri
6 Veritabanı işlemleri yapabilme

7 Çatılar ve Laravel
7 Çatılar ve Laravel ile ilgili bilgi verebilme

Anahtar Sözcükler: • diziler • if • for • while • do while • foreach • break • continue • switch • goto • include • require • function • class • Laravel

GİRİŞ

Bir önceki ünite de PHP programlama dilinin nasıl çalıştığı, PHP programlama dili ile yazılmış kodları çalıştırmak için gerekli olan araçların neler olduğu konusunda bilgi verilmiştir. Bu ünitede ise PHP programlama dilinin yapısı konusunda bilgi verilecek ve bir PHP çerçevesi olan Laravel tanıtılacaktır. PHP programlama dili birçok özelliği ve uygulama alanı olan son derece gelişmiş bir programlama dili olduğu için tüm özelliklerini detaylı olarak bir veya iki ünitede anlatmak mümkün değildir. Bu bakımdan bu ünite de PHP dilinin temel özellikleri konusunda özet bilgiye yer verilmiştir. Bu ünitede veriler bilgiler ışığında basit etkileşimli web sayfaları yapmak mümkün olacaktır. Ancak PHP'nin daha detaylı özelliklerini kullanabilmek için php.net adresinde bulunan PHP Kılavuzu veya PHP ile ilgili kitaplardan faydalanmak uygun olacaktır.

PHP programlama dilinin sözdizimi, C ve Java gibi programlama dillerindeki sözdizimine oldukça benzemektedir. Dolayısıyla benzer sözdizimine sahip programlama dillerine aşina olan bir programcının PHP programlama dilini öğrenmesi oldukça kolay olacaktır. Genel olarak birçok modern programlama dilinde de benzer bir sözdizimi yapısının kullanıldığı söylenebilir. PHP oldukça popüler bir dil olduğu için youtube üzerinden birçok ders videosuna ulaşmak mümkündür.

DEĞİŞKENLER VE DEĞİŞKEN TÜRLERİ

PHP dilinde değişkenlerin türünü özellikle belirtmeye gerek yoktur. Bir değişken tanımlanıp değer atandığı anda o değişkenin türü atanan değere göre otomatik olarak belirlenmiş olur. Örneğin bir değişkene *true* değeri atandığında bu değişkenin türü *boolean* olur. Aynı değişkene daha sonra bir tam sayı atanabilir. Bu durumda değişkenin türü de değişmiş olacaktır. PHP, değişkenler konusunda oldukça esnek bir yapıya sahiptir. Genel olarak PHP'de 4 temel türden bahsedilebilir: *boolean, integer, float* ve *string*. *Boolean* türündeki değişkenler sadece iki değer alabilmektedirler: *true* ve *false*. Bu tür değişkenler genellikle şartlı ifadelerde kullanılmaktadır. *Integer* ve *float* türleri ise tamsayı ve ondalıklı sayı değerleri ifade etmektedir. *String* türü değişkenler ise içlerinde metinsel ifadeleri barındırırlar. Aşağıda 4 değişken türüne yönelik tanımlamalar yer almaktadır. Karmaşık tür olarak ise *array* ve *object* türleri bulunmaktadır.

```
1   <?php
2       $kosul = false;
3       $notu = 80;
4       $ortalama = 77.5;
5       $sonuc = "Geçti";
```

Diziler

PHP programlama dilinde birçok değeri içinde barındırabilen diziler oluşturmak mümkündür. Diziler aslında anahtarların ve değerlerin ilişkilendirilmesi ile ilgili bilgiyi barındıran yapılardır. PHP dili diziler konusunda son derece esnek bir kullanım imkânı sunmaktadır. Dizi oluşturmak için *array* ya da [] ifadeleri kullanılabilir.

```
1   <?php
2   $ulkeler = array ("Türkiye","KKTC","Azerbaycan","Türkmenistan");
3   $ulkeler2 = ["Türkiye","KKTC","Azerbaycan","Türkmenistan"];
4   $karisik = [1,3,10.2,"Deneme",7,false];
5   echo $ulkeler[2]; //Azerbaycan
```

Yukarıdaki örnekte *$ulkeler* ve *$ulkeler2* olmak üzere içeriği aynı olan iki dizi tanımlaması yapılmıştır. Dizi tanımlamak için bu iki tanımlama yönteminden herhangi biri kullanılabilir. Örnekte verilen dizide 4 string değer bulunmaktadır. Bu değerler için anahtar tanımlaması yapılmadığı için 0'dan başlayacak şekilde anahtarlar otomatik olarak belirlenmiş olur. Dolayısıyla Azerbaycan değerinin anahtarı 2 olur ve dizinin

bu değerine ulaşabilmek için *$ulkeler[2]* ifadesi kullanılır. Dizi elemanlarına ulaşmak için değişken adı yazıldıktan sonra köşeli parantez içinde anahtar yazılmalıdır. Diğer bazı programlama dillerinde olduğu gibi dizi elemanlarının tek bir türde olma zorunluluğu bulunmamaktadır. Bir dizi; *string, integer, float, boolean, array, oject,* vb. türünde değerler barındırılabilir. Örnekteki *$karisik* isimli dizide farklı türde değerlerin dizi elemanlarına atandığı görülebilir. Yukarıda verilen örnekte oluşturulan diziler sadece değerler verilerek oluşturulmuş, anahtar tanımlaması yapılmamıştır. Aşağıda verilen örnekte anahtar tanımlaması yapılan dizi tanımlamalarına yer verilmiştir.

```
1   <?php
2       $araba = array(
3           "model"=>1970,
4           "marka"=>"Anadol",
5           "km"=>200000,
6       );
7       echo $araba["marka"]; //Anadol
```

Dizilerde anahtar tanımlaması yapmak için => işareti kullanılmaktadır. Bu işaretin solundaki ifade anahtar sağındaki ifade ise değerdir. Anahtar tanımlaması yapılan dizilerdeki elemanlara ulaşmak için *$araba["marka"]* örneğinde olduğu gibi değişken ismi yazıldıktan sonra köşeli parantez içinde anahtarın yazılması gerekmektedir. Anahtarlar sadece *string* veya *integer* olabilir. Aynı dizide hem *integer* hem de string türünde anahtar tanımlanabilir. PHP ile çok boyutlu diziler tanımlamakta mümkündür.

```
1   <?php
2       $koordinatlar = [
3          [3,5,7],
4          [2,4,6],
5          [2.5,3.5,4.5]
6       ];
7       echo $ulkeler[1][1]; //4
```

Yukarıdaki örnekte 3x3 boyutunda sayılardan oluşan bir dizi tanımlanmıştır. Aslında 3 elemanlı bir dizi tanımlanmıştır. Bu dizinin her bir elemanı ise yine 3 elemanlı başka dizileri barındırmaktadır. Sonuç olarak elde edilen yapı için 3x3 boyutunda bir dizi tanımlandığı söylenebilir. Anahtar tanımlaması yapılan diziler içinde çok boyutlu bir yapı oluşturmak mümkündür. Aşağıda bununla ilgili bir örneği yer verilmiştir.

```
1   <?php
2       $etiketler = array(
3          "div" => array(
4              "class" => "baslik",
5              "style" => array(
6                  "boder" => "1px solid black",
7                  "background-color" => "yellow",
8                  "color" => "purple"
9              )
10         ),
11         "a" => array(
12             "href" => "http://www.google.com",
13             "target" => "_black"
14         )
15      );
16      echo $etiketler["div"]["style"]["color"]; //purple
```

Örnekte *$etiketler* dizisinin "div" anahtarının değeri bir başka dizidir. Bu dizi ise "style" ve "class" anahtarlarının kullanıldığı iki eleman barındırmaktadır. "style" anahtarının değeri ise yine başka bir dizi olarak tanımlanmıştır. Bu dizi ise "border", "background-color" ve "color" anahtarları olmak üzere 3 elemana sahiptir. Örnekte *purple* değerine ulamak için $etiketler["div"]["style"]["color"] ifadesi kullanılmalıdır.

```
1   <?php
2       $dizi = array(5 => 1, 12 => 2);
3       $dizi[] = 56;
4       $dizi["x"] = 42;
5       unset($dizi[5]);
6       unset($dizi);
```

Yukarıda dizilerle ilgili bir örneğe yer verilmiştir. 2. satırda tanımlanan dizi iki elemanlıdır ve 5 ve 12 olmak üzere iki anahtar tanımlaması yapılmıştır. 3. satırdaki kullanım ile diziye yeni bir eleman eklenmektedir. Anahtar bölümü boş bırakıldığı için anahtar otomatik olarak üretilir. Otomatik anahtar üretilirken dizideki en büyük anahtar tam sayı değerine bir eklenir. Örnekte en büyük anahtar 12 olduğu için yeni anahtar 13 olur. Bir başka deyişle *$dizi[13]* ifadesi 56 değerine sahip olur. 4 satırda ise diziye x anahtarı ve bu anahtar için 42 değeri eklenmiştir. *Unset* fonksiyonu hem dizi elemanları hem de dizinin tamamını silmek için kullanılabilir. PHP, diziler ile ilgili çok sayıda yararlı fonksiyon sunmaktadır. Bu fonksiyonlardan birkaçı aşağıda açıklanmıştır.

- *array_reverse:* Dizideki elemanların sırasını ters çevirir. Bir başka deyişle son elemanı ilk sıraya alır.
- *sort ve rsort:* Dizideki elemanları değerlerine göre küçükten büyüğe veya büyükten küçüğü sıralamak için kullanılan fonksiyonlardır.
- *count:* Dizinin eleman sayısını geri döndürür.
- *shuffle:* Dizinin elemanlarının sırasını rasgele olarak değiştirir.
- *array_splice:* Diziden eleman silmek ve silinen elemanlar yerine yeni elemanlar eklemek için kullanılır.
- *array_merge:* Dizileri birleştirmek için kullanılır.
- *array_push:* Dizinin sonuna yeni eleman eklemek için kullanılır.
- *array_pop:* Dizinin sonundaki elemanı silmek için kullanılır.
- *array_unshift:* Dizinin başına yeni eleman eklemek için kullanılır.
- *array_shift:* Dizinin başındaki elemanı silmek için kullanılır.

dikkat

PHP'de farklı amaçlar için kullanılabilecek birçok fonksiyon bulunmaktadır. Bu fonksiyonlar ile ilgili olarak bir PHP kılavuzundan yararlanılması uygun olacaktır.

```
1   <?php
2       $renkler = ["Sarı","Yeşil","Siyah","Mavi","Beyaz"];
3       array_splice($renkler, 2, 2, array("Mor","Lila"));
4       var_dump($renkler);
```

Yukarıdaki örnekte *array_splice* fonksiyonunun kullanımına örnek verilmiştir. Bu fonksiyonun ilk parametresi üzerinde işlem yapılacak olan dizidir. İkinci parametre olarak ise işlem yapılacak elemanın sıra numarası verilmelidir. Üçüncü parametre olarak ise işlem yapılacak yerden itibaren kaça elemanın silineceği belirtilmelidir. Son olarak ise silinen bu elemanların yerine eklenecek olarak yeni elemanların listesi yine bir dizi olarak verilmelidir. Yukarıdaki örnekte diziden ikinci elemandan itibaren iki eleman silinmiş ("Siyah" ve "Mavi" değerleri) ve bu noktaya "Mor" ve "Lila" değerleri eklenmiştir. Dizinin yeni hali ise ["Sarı","Yeşil","Mor","Lila","Beyaz"] şeklindedir. Silinen eleman sayısı ile eklenecek eleman sayısının aynı olma zorunluluğu bulunmamaktadır.

internet

PHP Kılavuzuna http://docs.php.net/manual/tr/ adresinden erişebilirsiniz.

Öğrenme Çıktısı

1 Değişken türleri ve dizileri kullanabilme

Araştır 1

PHP'de birden fazla dizinin oluşturmak zorunda kaldığınız varsayın. Bu dizileri birleştirmek için kullanılması gereken kod ne olur?

İlişkilendir

PHP'de kullanılan 4 temel değişken türü olan boolean, integer, float ve string değişkenlerini özelliklerine göre karşılaştırınız.

Anlat/Paylaş

PHP'de kullanılan değişken ve dizi kavramlarını gerçek hayatla ilişkilendirerek açıklayınız.

HTML FORMLARI

HTML sayfalarında kullanıcının veri girişi yapabilmesi için HTML formları kullanılmaktadır. Bu formlara girilen değerler kullanıcı formu gönderdiği anda PHP tarafından işlenebilmektedir. Aşağıda bir HTML Formu örneği verilmiştir. HTML ile ilgili kitaplardan Formlar ile ilgili daha detaylı bilgi alınabilir.

```
1   <form action="isle.php" method="post">
2       Metin Kutusu: <input name="deger" /><br />
3       Radyo Düğmesi: <br />
4       <input type="radio" name="radyo" value="1" />Evet<br />
5       <input type="radio" name="radyo" value="2" />Hayır<br />
6       <select name="secim">
7           <option value="26">Eskişehir</option>
8           <option value="10">Balıkesir</option>
9       </select><br />
10      <textarea name="metinalani"></textarea><br />
11      <input name="gonder" type="submit" value="Gönder" />
12  </form>
```

Formlarda PHP için en önemli bölüm Form etiketi altındaki *action* ve *method* özellikleridir. Action özelliği ile forma girilen bilgilerin hangi dosyaya gönderileceği belirlenmektedir. Burada bir PHP dosyasının tam yolu verilmelidir. Örnekte kullanıcı gönder (submit) düğmesine tıkladığında forma girilen bilgiler *isle.php* dosyasına yönlendirilmektedir. Form etiketinin method özelliği ise iki değer almaktadır: *post* ve *get*. Post seçeneği seçilir ise, veriler arka planda gönderilir, *get* seçeneği seçilir ise forma girilen tüm veriler adres çubuğuna eklenerek gönderilir. Formlarda genellikle post değeri tercih edilmektedir. Formlarda kullanıcının değer girebileceği tüm etiketler için name özelliği tanımlamak gerekir. PHP, formdan gelen değerleri $_POST, $_GET ve $_REQUEST dizileri ile almaktadır. Bu diziler anahtarlı dizilerdir ve anahtar olarak etiketlerde tanımlanmış name özellikleri kullanılır. $_POST, post yöntemiyle gönderilmiş değerleri almak için, $_GET, *get* yöntemiyle gönderilmiş değerleri almak için, $_REQUEST ise herhangi bir yöntem ile gönderilmiş değerleri almak için kullanılabilir. $_REQUEST dizisi kullanıldığında gönderme yönteminin önemi yoktur. Yukarıdaki örnekteki form ile gönderilen değerler $_POST veya $_REQUEST dizileri

dikkat

$_POST, $_GET ve $_REQUEST dizileri ile kullanıcıdan gelen değerlerin zararlı içerik bakımından kontrol edilmesi gerekir. Örneğin kontrolsüz şekilde echo ile yazdırılan bir $_POST değeri içeriğinde "<script>...</script>"şeklinde bir JavaScript kodu barındırabilir ve sayfanın istenmeyen şekilde çalışmasına neden olabilir.

ile alınabilir. Ancak $_GET dizisi ile alınamaz. Aşağıda yukarıdaki formdaki değerleri almak için *isle.php* dosyasına yazılabilecek ifadeler gösterilmiştir.

```
1   <?php
2       echo $_POST["deger"];
3       echo $_REQUEST["radyo"];
4       echo $_REQUEST["secim"];
5       echo $_POST["metinalani"];
```

Bir php dosyasına doğrudan adres çubuğuna yazarak GET türü değişkenler göndermek mümkünüdür. Örneğin *http://localhost/isle.php?islem=sil&id=15&kullanici=Anadolu* şeklinde bir kullanım ile isle.php dosyasına 3 adet GET türü değişken gönderilmiştir. GET türü değişken göndermek için PHP dosyasının tam yolu verildikten sonra soru işareti konulur ve gönderilecek değişkenler aralarına & işareti konularak eklenir. Bu değişkenler yukarıda belirtilen yönteme benzer şekilde *isle.php* dosyası içinde kullanılabilir.

```
1   <?php
2       echo $_GET["islem"];
3       echo $_REQUEST["id"];
4       echo $_REQUEST["kullanici"];
```

Görüldüğü üzere GET ile gönderilen değişkenler $_GET ve $_REQUEST dizileri ile alınabilmektedir. Fakat $_POST dizisi ile alınamamaktadır.

Öğrenme Çıktısı

2 HTML formlarından gelen verileri kullanabilme

Araştır 2

PHP tarafından bagla.php dosyası ile işlenen bir HTML formunda arka planda veri göndermek için yazılan kodun ilk satırı nasıl olmalıdır ?

İlişkilendir

Web sitelerinde karşılaştığınız HTML formlarının hangi programlama dili ile işlendiğini belirlemeye çalışınız.

Anlat/Paylaş

Veri güvenliği açısından HTML formlarının get ve post etiket metod özelliklerini karşılaştırınız.

KONTROL YAPILARI

PHP dili, tüm programlama dillerinde bulunan *if, for, while* gibi kontrol yapılarını desteklemektedir. Bu yapıların kullanımları, PHP dili ile benzerlik gösteren diğer programlama dilleri ile oldukça benzerdir.

If

Programlama dillerinin en önemli ifadelerinden biri olan *if* ile bir kod bloğunun belirlenen koşullara bağlı olarak çalıştırılıp çalıştırılmayacağı belirlenir. *if*(karşılaştırma ifadesi) {çalıştırılacak komutlar;} şeklinde bir kullanıma sahiptir. Parantez içinde bulunan karşılaştırma ifadesinin sonucunun doğru (true) olması durumunda küme parantezi içinde bulunan komutlar çalıştırılır. Yanlış olması durumunda ise bu küme parantezi içinde bulunan komutlar çalıştırılmaz ve programın çalışması *if* bloğundan sonraki komut

ile devam eder. Parantez içinde sonucu doğru (true) ya da yanlış (false) olan bir karşılaştırma ifadesi yer almalıdır.

```
1   <?php
2       $vize = 80;
3       $final = 90;
4       $ortalama = 0;
5       if($final > $vize)
6           echo "Gelişim Var";
7       if($final >= 60){
8           $ortalama = $final*0.6+$vize*0.4;
9           echo $ortalama;
10      }
```

Yukarıdaki örnekte iki *if* kullanımına yer verilmiştir. Beşinci satırdaki *if* ifadesinde $final değişkeninin değeri *$vize* değişkeninin değerinden büyük olması durumunda ekrana "Gelişim Var" metni yazdırılmaktadır. Eğer *if* ifadesindeki koşulun doğru olması durumunda çalıştırılacak komut sadece bir adet ise küme parantezleri ile bir blok oluşturmadan bu komut altıncı satırda olduğu gibi doğrudan *if* ifadesinden sonra yazılabilir. *if* satırının sonunda noktalı virgül konmamalıdır. Noktalı virgül konduğunda *if* sonlandırılmış olur ve altındaki komut ile bağlantılı olmamış olur. Ancak *if* ile tek bir komut kullanılsa bile küme parantezlerini kullanmak kodların okunurluğunu arttıracağı için tercih edilmelidir. Yedinci satırda bulunan *if* ifadesi ile *$final* değişkeninin değerinin 60'dan büyük olup olmadığı kontrol edilmektedir. Eğer *$final* değişkeninin değeri 60'dan büyük ise *$ortalama* değişkeni hesaplanarak ekrana yazdırılmıştır. Sonuçta *if* ile bağlantılı iki komut bulunduğu için küme parantezi ile blok oluşturulmuştur.

Eğer *if* ile belirtilen koşul sağlanmadığında, bir başka deyişle karşılaştırma ifadesi sonucunun yanlış (false) olması durumunda çalıştırılmak istenen komutlar var ise *if* bloğundan sonra else bloğu kullanılmalıdır.

```
1   <?php
2       if($final > $vize) {
3          echo "Notlarda Yükselme Var";
4       } else {
5           echo "Notlarda Düşüş Var";
6       }
```

Örnekte de görüldüğü üzere ($final > $vize) sonucunun yanlış olması durumunda, bir başka deyişle *$final* değişkeninin değerinin *$vize* değişkeninin değerinden küçük veya eşit olması durumunda else bloğundaki komutlar çalıştırılır ve ekrana "Notlarda Düşüş Var" yazısı yazdırılır.

Birden fazla ve birbiriyle bağlantılı karşılaştırma ifadelerinin kullanılması durumunda ise *if* bloğundan sonra else *if* blokları yazılabilir. Aşağıda bununla ilgili bir örneğe yer verilmiştir.

```
1   <?php
2       if($ortalama>=90) {
3           echo "Pekiyi";
4       } else if ($ortalama>=80) {
5           echo "İyi";
6       } else if ($ortalama>=70) {
7           echo "Orta";
8       } else if ($ortalama>=60) {
9           echo "Zayıf";
10      } else {
11          echo "Başarısız";
12      }
```

Örnekte de görüldüğü üzere *$ortalama* değişkeninin değerine göre ekrana Pekiyi, İyi, Orta, Zayıf ve Başarısız yazıları yazdırılmaktadır. Yapılan karşılaştırma ifadeleri birbiri ile ilişkilidir. *else if* yapısı kullanıldığında karşılaştırma ifadeleri sırayla kontrol edilir. Doğru sonucunu üreten karşılaştırma ifadesinin bloğu çalıştırılır ve tüm *else if* yapısından çıkılır. Böylece *$ortalama* değişkeninin değeri 95 olduğunda ekrana sadece Pekiyi yazdırılır ve diğer karşılaştırma ifadeleri kontrol dahi edilmez. Eğer her karşılaştırma ifadesi tek tek kontrol edilmiş olsaydı, ekran Pekiyi, İyi, Orta, Zayıf değerlerinin hepsinin yazdırılması gerekirdi, çünkü *$ortalama* değişkeninin 95 değeri için tüm karşılaştırmalar doğru sonucunu üretir. *else if* yapısındaki tüm karşılaştırma ifadelerinin yanlış sonucunu üretmesi durumunda çalıştırılmak istenen komutlar var ise yapının en sonuna else bloğu eklenebilir. Örneğin *$ortalama* değişkeninin 55 değeri için tüm karşılaştırma ifadeleri yanlış sonucunu üretir ve *else if* yapısının en sonunda bulunan else bloğu çalıştırılır.

Karşılaştırma İfadeleri

if, for, while gibi komutların doğru çalışabilmesi için uygun karşılaştırma operatörlerinin kullanıldığı karşılaştırma ifadelerinin yazılması gerekmektedir. Bu karşılaştırma operatörleri ile ilgili olarak açıklama Tablo 3.1'de verilmiştir.

Tablo 3.1 Karşılaştırma Operatörleri

Operatör	Açıklama
==	Eşittir operatörüdür. $a == $b şeklinde kullanılır ve her iki tarafa yazılan ifadelerin değerleri eşit olduğunda doğru, olmadığında ise yanlış sonucunu üretir.
===	Özdeştir operatörüdür. Operatörün her iki tarafında yazılan ifadelerin sadece değerleri değil, aynı zamanda türlerinin de eşit olması sonucunda doğru, diğer durumlarda yanlış sonucunun üretir. Örneğin "1"===1 karşılaştırması yanlış sonucunu üretecektir, çünkü ifadelerden biri string diğeri ise integer türündedir.
!=	Eşit Değildir operatörüdür. $a != $b şeklinde kullanılır ve her iki tarafa yazılan ifadelerin değerlerinin eşit olmaması durumunda doğru, eşit olması durumunda ise yanlış sonucunu üretir.
!==	Özdeş Değildir operatörüdür. Operatörün her iki tarafında yazılan ifadelerin değerlerinin veya türlerinin eşit olmaması durumunda doğru, diğer durumlarda ise yanlış sonucunu üretir.
<	Küçüktür operatörüdür. $a < $b şeklinde kullanılır ve sol tarafa yazılan ifadenin değerinin sağ tarafa yazılan ifadenin değerinden küçük olması durumunda doğru sonucunu, diğer durumlarda ise yanlış sonucunu üretir.
<=	Küçük Eşittir operatörüdür. $a <= $b şeklinde kullanılır ve sol tarafa yazılan ifadenin değerinin sağ tarafa yazılan ifadenin değerinden küçük veya eşit olması durumunda doğru sonucunu, diğer durumlarda ise yanlış sonucunu üretir.
>	Büyüktür operatörüdür. $a > $b şeklinde kullanılır ve sol tarafa yazılan ifadenin değerinin sağ tarafa yazılan ifadenin değerinden büyük olması durumunda doğru sonucunu, diğer durumlarda ise yanlış sonucunu üretir.
>=	Büyüktür operatörüdür. $a >= $b şeklinde kullanılır ve sol tarafa yazılan ifadenin değerinin sağ tarafa yazılan ifadenin değerinden büyük veya eşit olması durumunda doğru sonucunu, diğer durumlarda ise yanlış sonucunu üretir.
!	Değildir operatörüdür. Diğer karşılaştırma ifadeleri ile birlikte kullanılmalıdır. Sağında bulunan karşılaştırma ifadesinin sonucu doğru ise yanlış, yanlış ise doğru sonucuna dönüştür. Örneğin ! ($a < $b) ifadesi eğer $a, $b'den küçük ise parantez içinin değeri doğru, ! işareti ile birlikte tüm ifadenin değeri ise yanlış olacaktır. Bu durumda aslında ! ($a < $b) ile $a >= $b ifadeleri aynı sonucu üreten ifadelerdir.

Birden fazla karşılaştırma ifadesi "ve" (&&) operatörü ve "veya" (||) operatörleri ile birleştirilerek daha karmaşık karşılaştırma ifadeleri yazmak mümkündür. Bu operatörlerin her iki tarafında karşılaştırma ifadeleri yer almalıdır. "ve" operatörü her iki tarafındaki karşılaştırma ifadelerinin sonuçlarının doğru olması durumunda doğru diğer durumlarda ise yanlış sonucunu üretir. "Veya" operatörü ise her iki tarafındaki karşılaştırmalardan herhangi birisinin sonucu doğru olması durumunda doğru, diğer durumda (her iki tarafında yanlış olması) ise yanlış sonucunu üretir. Örneğin $notu < 0 || $notu >100 şeklinde bir karşılaştırma yapılabilir. Örnekte de görüldüğü üzere $notu 0'da küçük ve notu 100'de büyük karşılaştırmaları "veya" operatörü ile birleştirilmiştir. $notu değişkeninin değeri 50 olması durumunda her iki karşılaştırma da yanlış sonucunu üreteceği için ifadenin tamamının sonucu yanlış olacaktır. Eğer $notu değişkeninin değeri 110 olursa, sağ taraftaki ifade doğru sol taraftaki ifade ise yanlış sonucunu üretir. Bu durumda ise ifadenin tamamının değeri doğru olacaktır. Programlama dillerinde 0<$notu<100 şeklinde bir karşılaştırma yapılamaz. Karşılaştırmalar ikili şekilde yapılmalı ve gerekirse && ve || operatörleri ile bağlanmalıdır.

Karşılaştırma ifadeleri ile ilgili aşağıda örnek bir uygulama verilmiştir. Kullanıcı web sayfasında bir üçgenin kenar uzunluklarını girebilmektedir. Göster düğmesine tıklandığında girilen üçgenin çeşitkenar, ikizkenar veya eşkenar üçgen olup olmadığı ile ilgili kullanıcıya bilgi verilmektedir.

```
1   <html>
2     <body>
3       <p>Lütfen Üçgenin Kenar Uzunuluklarını Giriniz</p>
4       <form action="ucgen.php">
5           Kenar 1: <input name="kenar1" /><br />
6           Kenar 2: <input name="kenar2" /><br />
7           Kenar 3: <input name="kenar3" /><br />
8           <input name="gonder" type="submit" value="Göster" />
9       </form>
10      <?php
11        $kenar1 = $_REQUEST['kenar1'];
12        $kenar2 = $_REQUEST['kenar2'];
13        $kenar3 = $_REQUEST['kenar3'];
14        if(!empty($_REQUEST["gonder"])){
15          if(empty($kenar1) || empty($kenar2) || empty($kenar3)){
16             echo "Lütfen kenar ölçülerini boş bırakmayınız.";
17         }else if(!is_numeric($kenar1) || !is_numeric($kenar2)
   || !is_numeric($kenar3)){
18                 echo "Lütfen sayısal değer giriniz.";
19         }else if($kenar1==$kenar2 && $kenar2==$kenar3){
20            echo "Eşkenar Üçgen";

21         }else if($kenar1==$kenar2 || $kenar1==$kenar3 ||
   $kenar2==$kenar3){
22              echo "İkizkenar Üçgen";
23           }else{
24               echo "Çeşitkenar Üçgen";
25           }
26        }
27      ?>
28    </body>
29  </html>
```

Örnekte de görüldüğü üzere kullanıcının girmiş olduğu kenar uzunluklarına göre formun altına üçgenin türü yazdırılmaktadır. İyi bir programcı, kullanıcının yapabileceği hataları göz önünde bulundurarak

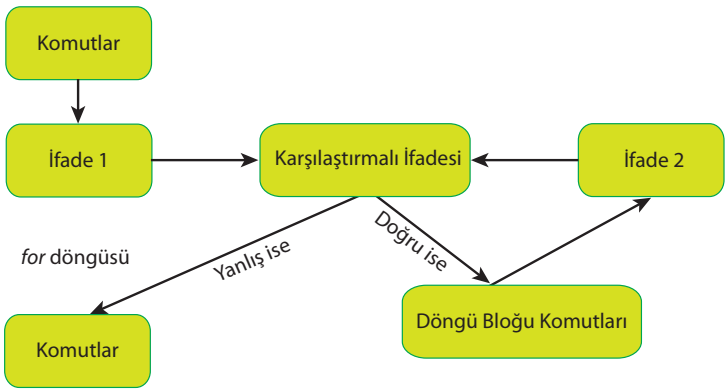
önlemler almalıdır. Bu örnekte de kullanıcın yapabileceği olası hatalar göz önünde bulundurulmuş ve bir uyarı mesajı gösterilmiştir. Öncelikle !empty($_REQUEST["gonder"]) karşılaştırması ile blok içindeki komutların sadece kullanıcının gönder düğmesine tıkladığında çalışması sağlanmıştır. PHP'nin *empty* fonksiyonu parametre olarak verilen değerin boş olması durumunda doğru, dolu olması durumunda ise yanlış sonucunu döndürür. 15. satırdan başlayan *if else* bloğunda öncelikle kullanıcının boş bıraktığı kenar ölçüsü olup olmadığı tespit edilmektedir. Veya operatörü ile bağlandığı için eğer kullanıcı kenar ölçülerinden birini bile boş bırakması durumunda 15. satırdaki karşılaştırma ifadesi doğru sonucunu üretecek ve kullanıcıya "Lütfen kenar ölçülerini boş bırakmayınız" mesajı verilecektir. 17. satırda ise PHP'nin *is_numeric* fonksiyonu ile kullanıcını girdiği değerlerin sayısal olup olmadığı kontrol edilmektedir. Eğer parametre olarak verilen değer sayısal bir değer ise *is_numeric* fonksiyonu doğru, sayısal bir değer değil ise yanlış sonucunu döndürmektedir. Eğer girilen değerlerden en az biri sayısal bir değer değil ise kullanıcıya "Lütfen sayısal değer giriniz" mesajı gösterilmektedir. 19. satırda girilen değerlerin bir eşkenar üçgene ait olup olmadığı ile ilgili kontrol yapılmaktadır. "Ve" operatörü kullanıldığı için hem *$kenar1* ile *$kenar2,* hem de *$kenar2* ile *$kenar3*'ün eşit olması durumunda ekrana "Eşkenar Üçgen" yazdırılmaktadır. *$kenar1* ile *$kenar2* eşit ve *$kenar2* ile *$kenar3* eşit ise *$kenar1* ile *$kenar3*'de eşit olacaktır. Dolayısıyla ayrıca karşılaştırma ifadesine kenar1==$kenar3 karşılaştırmasını eklemeye gerek yoktur. 21. satırda ise girilen değerlerin bir ikizkenar üçgene ait olup olmadığı ile ilgili kontrol yapılmaktadır. Bu kontrolün çalıştırılması için öncelikle üçgenin bir eşkenar üçgen olmaması gerekir, çünkü eşkenar üçgen olmuş olsaydı 19. satırdaki kontrole takılır ve tüm *else if* yapısından çıkılır. Eşkenar üçgen olmayan değerlerin ikiz kenar olup olmadığını test etmek için her ikili kenarın birbiri ile eşit olup olmadığı karşılaştırılmış ve "veya" operatörü ile bağlanmıştır. Böylece herhangi iki kenarın eşit olması durumunda ekrana "İkizkenar Üçgen" yazdırılmaktadır. Eğer üçgen ikizkenar da değil ise ekrana "Çeşitkenar Üçgen" yazdırılmaktadır.

For

for döngüsü, döngü bloğu içine yazılan kodların belirlenen koşullara bağlı olarak istenilen sayıda tekrar tekrar çalıştırılmasını sağlar. *for* döngüsünün söz dizimi aşağıdaki gibidir.

```
1   for(ifade1; karşılaştırma ifadesi; ifade2) {
2       komutlar;
3   }
```

for döngüsünde parantez içinde noktalı virgül ile ayrılmış 3 bölüm bulunmaktadır. Birinci ve üçüncü bölümde herhangi bir php ifadesi yazılabilir. İkinci bölümde ise mutlaka bir karşılaştırma ifadesinin yer alması gerekir. *for* döngüsünün çalışmasındaki en önemli parametre bu karşılaştırma ifadesidir. Her döngü adımında karşılaştırma ifadesi çalıştırılır ve doğru sonucunu üretmesi durumunda döngü içindeki komutlar çalıştırılır, yanlış sonucunu üretmesi durumunda döngüden çıkılır. *for* döngüsündeki ifade ve komutların çalıştırılma sırası Şekil 3.1'de verilmiştir.



Şekil 3.1 For Döngüsü

Şekilde de görüldüğü üzere program akışı *for* döngüsüne girdiğinde ilk olarak ifade ifade1 bölümü çalıştırılır. Daha sonra Karşılaştırma ifadesi kontrol edilir. Eğer karşılaştırma ifadesi doğru ise döngü bloğu komutları çalıştırılır. Eğer karşılaştırma ifadesi yanlış ise döngüden çıkılır. Döngü komutları çalıştırıldıktan sonra *ifade2* bölümü çalıştırılır ve tekrar karşılaştırma ifadesi kontrol edilir. Karşılaştırma ifadesi yanlış sonucunu üretene kadar döngü dönmeye devam edecektir.

Eğer parantez içindeki tüm bölümler boş bırakılır ise, sonsuz bir döngü kurulmuş olur. Bu durum web sayfasının cevap vermemesine neden olacaktır. Bu bakımdan sonsuz döngüye girmeyi engelleyen uygun bir karşılaştırma ifadesinin yazılması önemlidir. Aşağıda *for* döngüsünün kullanımına bir örnek verilmiştir.

```
1   <?php
2   for($i=0; $i<3; $i++){
3       echo "Merhaba Dünya<br />";
4   }
```

Bu örnekte karşılaştırma ifadesi $i<3 şeklindedir. Bu karşılaştırma ifadesi $i'nin 3'ten küçük olduğu durumlarda doğru, $i'nin 3 veya daha büyük olması durumunda yanlış sonucunu üretir. *for* döngüsüne ilk girildiğinde $i'nin değeri 0 olarak belirlenmektedir. 0, 3'ten küçük olduğu için döngü komutları $i'nin 0 değeri için çalıştırılır ve ekrana "Merhaba Dünya" yazısı yazılır. Döngü komutları çalıştıktan sonra $i++ ifadesi çalıştırılır ve $i'nin değeri 1 arttırılır. Böylece $i'nin değeri 1 olur. 1, 3'ten küçük olduğu için döngü komutları tekrar çalıştırılır ve ardından $i++ ifadesi tekrar çalıştırılır. $i'nin değeri 2 olur. 2, 3'ten küçük olduğu için döngü komutları tekrar çalıştırılır ve ardından $i++ ifadesi tekrar çalıştırılır. $i'nin değeri 3 olur. 3, 3'ten küçük olmadığı için döngüden çıkılır. Sonuç olarak, $i'nin 0, 1, ve 2 değerleri için döngü toplam 3 defa çalıştırılmış ve alt alta üç adet "Merhaba Dünya" yazısı yazılmıştır. Aşağıda yukarıdaki döngünün farklı bir yazımı verilmiştir. Parantez içindeki birinci bölüm ile üçüncü bölüm örnekte olduğu gibi taşınabilir.

```
1   <?php
2   $i=0;
3   for( ;$i<3; ){
4       echo "Merhaba Dünya<br />";
5       $i++;
6   }
```

Aşağıda 1'den kullanıcının girdiği sayıya kadar olan sayıların toplamını hesaplayarak sonucu gösteren bir PHP dosyası örneği verilmiştir.

```
1   <html>
2       <body>
3           <form>
4               Sayı: <input name="sayi" /><br />
5               <input type="submit" value="Hesapla" />
6           </form>
7           <?php
8           $sayi = $_REQUEST["sayi"];
9           if(!empty($sayi)){
10              if(is_numeric($sayi)){
11                  $toplam=0;
12                  for($i=1; $i <= $sayi; $i++){
13                      $toplam += $i;
14                  }
15                  echo "Toplam: ".$toplam;
16              }
17          }
```

Hesaplama yapmak için öncelikle *$toplam* değişkeni tanımlanmış ve 0 değeri atanmıştır. Daha sonra $i değişkeninin değeri 1'den kullanıcı tarafından girilen değere kadar dönen bir *for* döngüsü yazılmıştır. *for* döngüsünün içinde ise $i değişkeninin değeri $toplam değişkenine eklenmektedir. Böylece örneğin kullanıcı 3 değerini girmesi durumunda 1+2+3 toplamı olan 6 değeri elde edilmiş ve 15 satırdaki komut ile ekrana yazdırılmış olur. Aşağıdaki örnekte ise iç içe *for* döngüleri kullanılarak rasgele sayılardan oluşan bir tablo oluşturulmuştur.

```
1   <?php
2   echo "<table border=1>";
3   for($i=0;$i<3;$i++){
4       echo "<tr>";
5       for($j=0;$j<5;$j++){
6           echo "<td>".rand(0, 100)."</td>";
7       }
8       echo "</tr>";
9   }
10  echo "</table>";
```

Birinci *for* döngüsü tablonun satırlarını oluşturmak için kullanılmıştır ve 3 defa dönmektedir. Bu durumda ortaya 3 satırdan oluşan bir tablo ortaya çıkmaktadır. Bu *for* döngüsünün içinde ise tablonun hücrelerini oluşturmak için toplam 5 kere dönen bir *for* döngüsü daha kullanılmıştır. Bu *for* döngüsü ise her satır için 5 kez dönmektedir. Bu durumda 6. satırdaki komut toplam 15 defa çalışmış olur. 0 ile 100 arasında rasgele sayı üretmek için PHP'nin *rand* fonksiyonu kullanılmıştır.

While

PHP ile döngü oluşturmak için kullanılabilecek bir başka yapı ise *while* döngüsüdür. *while* döngüsünün sözdizimi aşağıdaki gibidir

```
1   while (karşılaştırma ifadesi){
2        komutlar;
3   }
```

for döngüsü ile aynı şekilde çalışır. *for* döngüsünden tek farkı parantez içindeki birinci ve üçüncü bölümün olmamasıdır. Karşılaştırma ifadesinin sonucu doğru olduğu sürece döngü içindeki komutlar çalıştırılır ve her döngü adımında karşılaştırma ifadesi kontrol edilir. Karşılaştırma ifadesi yanlış sonucunu ürettiği anda döngüden çıkılır.

```
1   <?php
2   $i=0;
3   while($i<3){
4       echo "Merhaba Dünya<br />";
5       $i++;
6   }
```

Örnekte görüldüğü üzere daha önce *for* döngüsü ile yapılan işin aynısı *while* döngüsü ile gerçekleştirilmiştir.

Do while

PHP'de bir başka döngü yöntemi ise *do while* döngüsüdür. *do while* döngüsünün sözdizimi aşağıdaki gibidir.

```
1   do {
2       komutlar;
3   } while (karşılaştırma ifadesi);
```

do while döngüsü *while* döngüsüne oldukça benzemektedir. *while* döngüsünde döngüye girilmeden önce karşılaştırma ifadesi kontrol edilir ve doğru sonucunu üretmesi durumunda döngü komutları çalıştırılır. Hatta döngüye ilk girişte karşılaştırma ifadesi yanlış sonucunu üretir ise döngü bloğu komutları bir kere bile çalıştırılmadan döngüden çıkılmış olur. Ancak *do while* döngüsünde önce döngü bloğu komutları çalıştırılır ve daha sonra karşılaştırma ifadesi kontrol edilir. Dolayısıyla *do while* yapısında döngü bloğu içindeki komutlar en az bir kez mutlaka çalıştırılır.

Foreach

Dizi elemanlarına yönelik döngü oluşturmak için *foreach* döngüsü kullanılır. *foreach* döngüsünün sözdizimleri aşağıdaki gibidir.

```
1   foreach ($dizi as $deger){
2       komutlar;
3   }
4   foreach ($dizi as $anahtar => $deger){
5       komutlar;
6   }
```

Anahtar kullanılmayan diziler için üstteki sözdizimi, anahtar kullanılan diziler için ise alttaki sözdizimi kullanılabilir. Parantez içinde "as" kelimesinin solunda döngünün gerçekleştirileceği dizinin değişken adı verilmelidir. As kelimesinin sağında ise herhangi yeni bir değişken adı yazılabilir. Her bir döngü adımında dizi elemanları bu değişkene atanmaktadır. Anahtarlı yapı kullanılır ise dizi elemanının anahtarı ve değeri ayrı ayrı belirtilen değişkenlere atanmaktadır. Aşağıda *foreach* döngüsünün kullanımı ile ilgili bir örneğe yer verilmiştir.

```
1   <?php
2   $dizi = ["Mandalina","Portakal","Muz","Elma","Armut"];
3   foreach ($dizi as $deger) {
4       echo $deger."<br />";
5   }
```

Örnekte döngü *$dizi* isimli dizi için toplam 5 kez dönmektedir. Her bir döngü adımında sıradaki dizi elemanının değeri *$deger* isimli değişkene atanmaktadır. Döngü bloğu içinde ise bu değer ekrana yazdırılmaktadır.

Break

Break komutu; çalışmakta olan *for, foreach, while, do while* veya *switch* yapılarının çalışmasını sonlandırır ve program akışı bu yapılardan sonraki komut ile çalışmaya devam eder. Aşağıda break komutunun kullanımı ile ilgili bir örnek bulunmaktadır.

```
1   <?php
2   $dizi = ["Mandalina","Portakal","Muz","Elma","Armut"];
3   for($i=0;$i<count($dizi);$i++){
4     if($dizi[$i]=="Muz"){
5       echo "Muz değeri dizinin".($i+1).". elemanında bulunmaktadır";
6        break;
7     }
8   }
```

Örnekte Muz değerinin dizinin kaçıncı elemanında bulunduğu tespit edilmiştir. Muz değeri *if* ile kontrol edildikten sonra ekrana kaçıncı eleman olduğu ile ilgili bilgi yazılmış ve break ile *for* döngüsü sonlandırılmıştır. Aşağıda break kullanımına bir örnek daha verilmiştir.

```
1   <?php
2   $satislar = array(
3       "Pazartesi"=>array(40,70,40,90,80),
4       "Salı"=>array(40,100,90,80),
5       "Çarşamba"=>array(90,70,30,100,80,40),
6       "Perşembe"=>array(50,70,40,200,80,20,10,70),
7       "Cuma"=>array(120,70,40,90,80),
8   );
9   $toplam=0;
10  $say=0;
11  foreach ($satislar as $gun => $ucretler){
12    for($i=0;$i<count($ucretler);$i++){
13      $say++;
14      $toplam += $ucretler[$i];
15      if($toplam>1000){
16        echo "Toplam 1000TL'lik satışa ".$gun." günü
    ulaşılmıştır.<br/>";
17       break 2;
18      }
19    }
20  }
21  $ortalama = $toplam / $say;
22  echo "Toplam 1000TL'lik satışın ortalaması" .number_
    format($ortalama,2)."TL'dir";
```

Bu örnekte öncelikle bir firmanın günlük satış miktarları *$satıslar* ismindeki dizide tutulmaktadır. Bu programdaki amaç toplamda 1000TL'lik satışa hangi gün ulaşılmıştır ve bu satışların ortalaması nedir. *$satislar* dizisi iki boyutlu tasarlanmış bir dizidir. Dolayısıyla iç içe *for* döngüsü kullanılmıştır. Burada dikkat edilmesi gereken nokta *$toplam* değişkeninin değeri 1000TL'yi geçtiği anda break 2; komutu kullanılmıştır. Böylece iç içe iki döngü sonlandırılmıştır ve ortalama hesaplama komutuna geçilmiştir. Eğer break komutunun sonuna 2 sayısı yazılmamış olsaydı sadece içerdeki döngü sonlandırılmış olurdur ve *foreach* döngüsü dönmeye devam ederdi.

Continue

continue komutu sadece döngülerde kullanılır ve kullanıldığı noktadan itibaren döngü bloğu içinde yazılmış kodların çalıştırılmadan döngü başına gidilmesini sağlar. Aşağıda *continue* kullanımına bir örnek verilmiştir.

```
1   <?php
2   $notlar=[40,0,50,80,100,0,70,90,0,100,80];
3   $toplam=0;
4   $say=0;
5   for($i=0;$i<count($notlar);$i++){
6       if($notlar[$i]==0 || $notlar[$i]==100){
7           continue;
8       }
9       $toplam += $notlar[$i];
10      $say++;
11  }
12  echo "Sınıf Ortalaması:".number_format($toplam/$say,2);
```

Örnekte bir sınıfa ait notların bulunduğu bir $notlar dizisi bulunmaktadır. Sınıf ortalaması hesaplanmak istenmektedir. Ancak bu ortalamaya 100 ve 0 notları dahil edilmemiştir. 6. satırda *$notlar* dizinin

ilgili elemanının değerinin 100 veya 0 olması durumunda continue komut çalıştırılmıştır. Bu durumda döngü içindeki diğer komutlar çalıştırılmadan döngü başa dönmüştür. Bir başka değişle *continue* komutu çalıştırıldığında 9. ve 10. satırdaki kodlar çalıştırılmadan döngü başa döner ve dizinin bir sonraki elemanı ile işlem yapılmaya devam edilir. Böylece $toplam değişkenine 0 ve 100 olan notlar eklenmemiş olur. $say değişkeni ile ortalaması alınacak not sayısı belirlenmektedir ve *for* döngüsünün bitiminde sınıf ortalaması ekrana yazdırılmaktadır.

continue komutu varsayılan olarak sadece içinde bulunduğu döngünün başa dönmesini sağlar. Bir başka deyişle break ile benzer şekilde *continue* için varsayılan değer 1'dir (continue 1;). Eğer *continue* 2; şeklinde kullanılır ise içe yazılmış olan döngülerden üstte bulunan döngünün başa dönmesi sağlanmış olur. Benzer şekilde 3 adet iç içe yazılmış bir döngü yapısında en iç döngüde yazılan *continue* 3; komutu en üst düzeydeki döngünün başa dönmesini sağlar.

Switch

Switch bir karşılaştırma yapısıdır ve *if* ve *else if* yapısına benzerlik gösterir. Bir değişkenin farklı değerleri için farklı komutların çalıştırılmasını sağlar. Aşağıda *switch* kullanımına bir örnek verilmiştir.

```
1   <?php
2   if ($i == 0) {
3       echo "i 0'a eşittir";
4   } elseif ($i == 1) {
5       echo "i 1'e eşittir";
6   } elseif ($i == 2) {
7       echo "i 2'ye eşittir";
8   }else{
9       echo "i'nin değeri 0,1 veya 2 değildir";
10  }
11  switch ($i) {
12      case 0:
13          echo "i 0'a eşittir";
14          break;
15      case 1:
16          echo "i 1'e eşittir";
17          break;
18      case 2:
19          echo "i 2'ye eşittir";
20          break;
21      default:
22          echo "i'nin değeri 0,1 veya 2 değildir";
23  }
```

Örnekte verilen *if* ile *switch* yapıları birebir aynı sonucu vermektedir. Program $i değişkeninin değerini kontrol etmekte ve ekrana bir bilgi mesajı yazdırmaktadır. *swith* yapısında parantez içinde verilen değişkenin değeri için *case* bölümleri kontrol edilmektedir. Örnekte $i değişkeninin değeri 1 olduğunda *case* 1: bölümü çalıştırılır. Eğer $i değişkeninin değerine hiçbir *case* ifadesinde rastlanamaz ise *default*: bölümü çalıştırılır. Her *case* bölümünün sonunda break ile *switch* yapısından çıkılmıştır.

```
1   <?php
2   $gun="Çarşamba";
3   switch ($gun) {
4      case "Pazartesi":
5         echo "Pazartesi<br />";
```

```
6       case "Salı":
7          echo "Salı<br />";
8       case "Çarşamba":
9          echo "Çarşamba<br />";
10      case "Perşembe":
11         echo "Perşembe<br />";
12      case "Cuma":
13         echo "Cuma<br />";
14      default:
15         echo "Hafta Sonu";
16   }
```

Goto komutu eski yordamsal programlama dilleri ile yaygın olarak kullanılmaktaydı. Ancak günümüzde bu komut her ne kadar programlama dilleri tarafından desteklese de hemen hemen hiç kullanılmamaktadır.

Yukarıdaki örnekte her *case* bölümü sonunda break kullanılmamıştır. *$gun* değişkeninin değeri "Çarşamba" olduğu zaman ekrana alt alta "Çarşamba", "Perşembe", "Cuma" ve "Hafta Sonu" değerleri yazdırılır. Herhangi bir *case* değerinin doğru olması durumunda o *case* bölümü ve sonraki tüm *case* bölümlerinde bulunan tüm komutlar çalıştırılır. Bu çalışmayı durdurmanın yolu ise bir önceki örnekte olduğu gibi break komutunu kullanmaktır.

Goto

goto komutu ile program akışını değiştirmek mümkündür. Normalde PHP kodları sırayla çalıştırılır. *goto* komutu ile program akışının istenilen bölümden devam etmesi sağlanabilir. goto kullanmadan önce program akışında etiketler koymak gerekir. Etiket koymak için etiketin adı ve ":" işareti kullanılmalıdır. Aşağıdaki örnekte bolum1, bolum2, bolum3 ve son etiketleri kullanılmış ve *goto* komutları ile program akışı bu etiketlerin olduğu yerlerden devam etmesi sağlanmıştır.

```
1   <?php
2   goto bolum1;
3   bolum2:
4       echo "Bölüm 2<br />";
5       goto bolum3;
6   bolum3:
7       echo "Bölüm 3<br />";
8       goto son;
9   bolum1:
10      echo "Bölüm 1<br />";
11      goto bolum2;
12  son:
```

Program başlar başlamaz *goto* komutu ile bolum1 etiketine yönlendirilmiştir. Ekrana "Bölüm 1" yazıldıktan sonra program bolum2 etiketine yönlendirilmiştir. Bu bölümde de ekrana "Bölüm 2" yazdırılmaktadır. Daha sonra program akışı bolum3 etiketine yönlendirilmiştir. Bu bölümde ekrana "Bölüm 3" yazılmış ve program akışı son etiketine yönlendirilmiş ve program akışı sona ermiştir. Görüldüğü üzere yanlış bir yönlendirme ile rahatlıkla program akışı sonsuz döngüye sokulabilir. Bu durumu yaşamamak için *goto* komutlarının mümkün olduğunca kullanılmaması veya çok dikkatli kullanılması gerekmektedir.

include/include_once

Bu komutlar ile çalışan php dosyasına bir başka php dosyası dahil edilir ve içeriği çalıştırılır. *include* komutu ile bir dosya birden fazla dahil edilebilirken, *include_once* komutu kullanıldığında eğer belirtilen dosya daha önce dahil edildiyse tekrar dahil edilmez. Bu komutlarda belirtilen dosya PHP tarafından bulunamaz ise uyarı mesajı yayınlanır, ancak program çalışmasına devam eder. Örnek kullanım aşağıdaki gibidir.

```
1   <?php
2   include_once "kutuphane/fonksiyonlar.php";
3   include "../vt.php";
4   include "ust.php";
5   include "/home/serkan/public_html/deneme/orta.php";
```

include ve include_once komutlarında belirtilen dosya öncelikle belirtilen yolda aranır. Eğer belirtilen yolda bulunamaz ise PHP'nin ayar dosyasında *include_path* ile ayarlanmış klasörlere bakılır. Belirtilen dosyanın yolu, 5. satırda olduğu gibi kök klasörden itibaren tam yol olarak yazılabileceği gibi, çalışan dosyanın konumuna bağlı olarak ta ayarlanabilir. Satır 2'de olduğu gibi eğer dahil edilecek dosya çalışan dosyaya göre alt klasörlerde yer alıyorsa önce alt klasörlerin adı aralarında / işareti olacak şekilde yazılır ve daha sonra dosya adı yazılır. 3. satırda olduğu gibi eğer dahil edilecek dosya üst klasörlerde ise .. (iki nota yan yana) işareti ile üst klasörlere çıkılabilir. Örneğin "../../deneme.php" ifadesi iki üst klasördeki dosyaya işaret etmektedir. Bir başka örnek olarak çalışan dosya ile dahil edilecek dosya aynı düzeyde farklı klasörlerde ise "../diger/deneme.php" şeklinde önce üst klasöre çıkılıp, daha sonra o klasörde bulunan bir başka klasörün içine girilerek dahil edilebilir. 4. satırda olduğu gibi eğer dahil edilecek dosya, çalışan dosya ile aynı klasörde ise sadece dahil edilecek dosyanın adını yazmak yeterli olacaktır.

require/require_once

Bu iki komut da aslında *include ve include_once* ile aynı işi yapmaktadır. Kullanım şekilleri de aynıdır. Tek fark *require* ve *require_once* komutları kullanıldığında belirtilen dosyanın bulunmaması durumunda PHP'de ölümcül hataya neden olur ve program akışı sonlandırılır. Dolayısıyla bu iki komut mutlaka dahil edilmesi gereken dosyalar için kullanılmalıdır. Bu dosyalar dahil edilemediğinde program hata verecek ve çalışmaz hale gelecektir. *İnclude* ve *include_once* komutları ise olsa da olur olmasa da olur denebilecek dosyaları dahil etmek için kullanılmalıdır.

Öğrenme Çıktısı

3 Kontrol yapılarını kullanabilme

Araştır 3

Bir elemanın koordinatlarının, koordinat düzleminde (100,50) noktası ile (500,300) noktası arasında kalan dikdörtgenin içinde yer alıp almadığını kontrol etmek için nasıl bir karşılaştırma ifadesi yazılabilir?

İlişkilendir

PHP karşılaştırma operatörlerini ve görevlerini matematik sembolleri ile ilişkilendiriniz.

Anlat/Paylaş

PHP'de yaygın olarak kullanılan for ve while döngülerini benzerlikleri ve farklılıklarıyla birlikte karşılaştırınız.

FONKSİYONLAR

Diğer programlama dillerinde olduğu gibi PHP ile de fonksiyon tanımlamak mümkündür. Fonksiyonlar daha sonra çalıştırmak için programcı tarafından yazılan kod bloklarıdır. Fonksiyonlar genellikle sıkça yapılan işlemler için tanımlanır. Böylece benzer kodlar tekrar tekrar yazılmamış olur. Aynı zamanda kodların organizasyonunu sağlamak içinde fonksiyon tanımlaması yapılabilir. Böylece kodlar daha okunaklı ve anlaşılır olacaktır. Yazılan bir fonksiyonun çalışması için mutlaka çağrılması gerekir. Fonksiyon tanımlama sözdizimi aşağıda verilmiştir.

```
1   function fonksiyonAdi(parametre listesi){
2       komutlar;
3       return değer;
4   }
```

Fonksiyon tanımı *function* kelimesi ile yapılmaktadır. Daha sonra fonksiyona bir isim verilmelidir. Fonksiyon isimlendirme kuralları, değişken isimlendirme kuralları ile aynıdır. Fonksiyon isminden sonra parantez içinde virgül ile ayrılmış parametre listesi verilmelidir. Parametreler fonksiyona değer göndermek için kullanılmaktadır. Parametre sayısı, fonksiyonun amacına bağlı olarak değişkenlik gösterebilir. Fonksiyon bloğu içinde fonksiyonun yapması istenen komutlar yer almalıdır. Son olarak ise isteğe bağlı olarak fonksiyon geriye bir değer döndürebilir. Geriye değer döndürmek için ise *return* komutu kullanılmalıdır. *return* komutunun kullanıldığı yerde fonksiyon sonlandırılır ve fonksiyondan çıkılır. Bu bakımdan *return* komutu döngüleri sonlandırmak için kullanılan *break* komutu gibi de kullanılabilmektedir. Aşağıda parametre olarak verilen sayıya kadar olan sayıların toplamını hesaplayıp geri döndüren bir fonksiyon tanımlaması yapılmıştır.

```
1   <?php
2   $top = sayilarToplami(10);
3   echo "10'a kadar olan sayıların toplamı: ".$top;
4   function sayilarToplami($sayi){
5       $toplam=0;
6       for($i=1;$i<=$sayi;$i++){
7           $toplam += $i;
8       }
9       return $toplam;
10  }
```

Örnekte fonksiyon tanımı 4. satır ile 10. satır arasında yapılmıştır. *fonksiyon* ismi olarak *sayilarToplami* ifadesi kullanılmıştır. Bu fonksiyon bir parametre almaktadır. C, Java gibi bazı bazı programlama dillerinden farklı olarak PHP dilinde parametrelerin türü ve fonksiyonun geri dönüş türü belirtilmemektedir. Fonksiyon bloğu içinde parametre olarak verilen değere kadar olan sayıların toplamı hesaplanmakta ve 9. satırda *return* komutu ile geri döndürülmektedir. Daha öncede belirtildiği üzere fonksiyonlar çağrılmadan çalışmazlar. Örnekte ise 2. satırda tanımlanan bu fonksiyon çağrılmıştır. Gerekli olması durumunda istenilen yerde bu fonksiyon defalarca çağrılabilir. Fonksiyon tanımında bir adet parametre bulunmaktadır. O yüzden fonksiyon çağrılırken fonksiyona bir adet değer parametre olarak verilmiştir. Ayrıca fonksiyon *return* ile geriye bir değer döndürdüğü için geriye dönen bu değeri saklamak için fonksiyon *$top* değişkenine atanmıştır. Son olarak ise 3. satırda fonksiyonun döndürdüğü değer *echo* komutu ile ekrana yazılmıştır.

Program akışı göz önünde bulundurulduğunda fonksiyonların çağrılmadan önce tanımlanmış olması gerekmez. Bir başka deyişle, örnekte olduğu gibi program akışında önce fonksiyon çağrılıp daha sonra tanımlaması yapılabilir. PHP, kodları sırayla çalıştırmadan önce dosyada tanımlanmış olan fonksiyon ve sınıfları indeksler. Daha sonra program akışı başladığında fonksiyonlar ve sınıflar nerede tanımlanmış olursa olsun çalıştırılabilirler. Ayrıca tüm fonksiyonlar bir blok içinde tanımlanmış bile olsa global olarak değerlendirilir ve her yerden çağrılabilir.

```
1   <?php
2   kutu(0,0,100,100,"Merhaba Dünya");
3   kutu(500,0,100,100,"Merhaba Dünya");
4   kutu(250,400,100,100,"Merhaba Dünya");
5   function kutu($x,$y,$genislik,$yukseklik,$mesaj){
6       echo "<div style='position:absolute;"
7               . "border:1px solid black;"
8               . "left:".$x."px;"
9               . "top:".$y."px;"
10              . "width:".$genislik."px;"
11              . "height:".$yukseklik."px;'>$mesaj</div>";
12  }
```

Yukarıdaki örnekte kutu isminde bir fonksiyon tanımlaması yapılmıştır. Bu fonksiyon web sayfasının istenilen koordinatlarında bir div oluşturmakta ve parametre olarak aldığı mesajı bu div etiketinin içine yazmaktadır. Fonksiyon *return* ile geriye bir değer döndürmemektedir. Dolayısıyla çağrılırken fonksiyonu bir değişkene atamaya gerek yoktur. Fonksiyon içinde aslında sadece bir satır kod bulunmaktadır. Ancak bu kodda bulunan *string* (metin) ifade çok uzun olduğu için nokta ile birleştirilerek satırlara bölünmüştür.

Kapsam

PHP dilinde değişkenler sadece tanımlandıkları blok içinde ve o bloğun alt bloklarında geçerlidir. Bir blok içinde tanımlanmış değişkene yerel değişken, bir blok içinde tanımlanmayan değişkene ise global değişken adı verilir. Global değişkenler tanımlandıktan sonra tüm php dosyasında geçerlidirler. Ancak yerel değişkenler tanımlandıkları blok bittiğinde ölürler ve bir daha kullanılamazlar.

```
1   <?php
2   $kuresel = 10;
3   function deneme(){
4       $yerel = 15;
5       echo "Küresel: ".$kuresel." ve Yerel: ".$yerel."<br />";
6   }
7   deneme();
8   echo "Küresel: ".$kuresel." ve Yerel: ".$yerel."<br />";
```

Yukardaki örneğin çıktısı aşağıdaki gibidir:

Küresel: ve Yerel: 15

Küresel: 10 ve Yerel:

Çıktı incelendiğinde *$kuresel* değişkeninin fonksiyon içinde bu şekilde kullanıldığında tanımsız olduğu, *$yerel* değişkeninin ise fonksiyon dışında tanımsız olduğu ve sadece kendi bloğu içinde kullanılabileceği görülmektedir. Global değişkenler fonksiyonlar da dahil tüm PHP dosyasında kullanılabilirler. Bunun için fonksiyon içinde bu değişkenin bir global değişken olduğu belirtilmelidir.

```
1   <?php
2   $kuresel = 10;
3   deneme();
4   function deneme(){
5       global $kuresel;
6       $yerel = 15;
7       echo "Küresel: ".$kuresel." ve Yerel: ".$yerel."<br />";
8   }
```

Örnekte fonksiyon içinde *$kuresel* değişkeninin bir global değişken olduğu 5. satırda belirtilmiş ve fonksiyon için kullanılmıştır. Global değişkenlere global ifadesi ile belirtmeye gerek kalmadan $GLOBALS['kuresel'] şeklinde de ulaşmak mümkündür.

Parametre Gönderimi

Varsayılan olarak bir fonksiyona parametre gönderildiğinde sadece değer olarak gönderilir. Fonksiyon içinde parametre olarak verilen değişkene ulaşılmaz.

```
1   <?php
2   $kuresel = 10;
3   deneme($kuresel);
4   echo "Kuresel: ".$kuresel;
5   function deneme($parametre){
6       $parametre += 5;
7       echo "Parametre: ".$parametre."<br />";
8   }
```

Örnekte deneme fonksiyonuna parametre olarak *$kuresel* değişkeni gönderilmektedir. Fonksiyonda ise parametre olarak verilen *$kuresel* değişkeninin değeri $parametre değişkenine aktarılmaktadır. Fonksiyon bloğunda $parametre değişkeninin değeri arttırılmakta (değeri 15 olur) ve ekran yazılmaktadır. Fonksiyon 3. satırda çağrıldıktan sonra 4. satırda *$kuresel* değişkeninin değeri ekrana yazdırıldığında 10 sonucunun olduğu ve fonksiyon tarafından değiştirilmediği görülecektir. Eğer parametre olarak verilen değişkenin fonksiyon tarafından değiştirilmesi istenir ise değişkenin referans olarak gönderilmesi gerekmektedir. Bir değişkenin referans olarak gönderilebilmesi için fonksiyon parametre tanımlamasında başına & işareti konulur.

```
1   <?php
2   $kuresel = 10;
3   deneme($kuresel);
4   echo "Kuresel: ".$kuresel;
5   function deneme(&$parametre){
6       $parametre += 5;
7       echo "Parametre: ".$parametre."<br />";
8   }
```

Örnekte fonksiyonda *$parametre* parametresi "&" ile tanımlandığı için 3. satırda fonksiyon çağrıldığında *$parametre* değişkeni doğrudan *$kuresel* değişkenini işaret etmektedir ve fonksiyon içinde *$parametre* değişkeni üzerinde yapılan işlemler, aslında *$kuresel* değişkeni üzerinde yapılmış olur. Sonuç olarak 4. satırda küresel değişkeninin değerinin 15 olarak değiştiği görülecektir.

Varsayılan Parametre Değerleri

Fonksiyonlara varsayılan parametre tanımlaması yapılabilir. Böylece fonksiyon çağrılırken varsayılan parametre tanımlaması yapılan değerler parametre olarak gönderilmeyebilir. Parametre gönderilmediğinde fonksiyon içinde varsayılan değer kullanılır. Varsayılan parametre tanımlaması yapılırken mutlaka parametre listenin en sonundan itibaren yapılmalıdır. Örneğin iki parametre alan bir fonksiyonun ilk parametresine varsayılan değer tanımlayıp, ikinci parametresine varsayılan değer tanımlamamak hataya sebep olacaktır. Bu durumda ya her iki parametreye de varsayılan değer tanımlanabilir ya da sadece ikinci parametreye varsayılan değer tanımlaması yapılabilir. Aşağıda varsayılan parametre tanımı ile ilgili bir örneğe yer verilmiştir.

```
1   <?php
2   echo "Karenin Çevresi: ".  cevreHesapla(5)."<br />";
3   echo "Dikdörtgenin Çevresi: ".  cevreHesapla(3,5)."<br />";
4   echo "Üçgenin Çevresi: ".  cevreHesapla(3,4,5)."<br />";
5   function cevreHesapla($kenar1,$kenar2=0,$kenar3=0){
6       if($kenar2==0){
7           return $kenar1*4;
8       }else if($kenar3==0){
9           return 2*($kenar1+$kenar2);
10      }else{
11         return $kenar1+$kenar2+$kenar3;
12      }
13  }
```

Örnekteki fonksiyon 3 parametre almaktadır ve sondan iki parametrenin varsayılan değeri 0 olarak belirlenmiştir. Böylece bu fonksiyon çağrılırken bir parametre ile, iki parametre ile veya 3 parametre ile çağrılabilir. Eğer fonksiyon tek parametre ile çağrılır ise karenin çevresi, iki parametre ile çağrılır ise dikdörtgenin çevresi, üç parametre ile çağrılırsa üçgenin çevresinin hesabını geri döndürmektedir.

Sınırsız Sayıda Parametre

PHP ile sınırsız sayıda parametre alan fonksiyonlar tanımlanabilir. Sınırsız sayıda parametre alan fonksiyonlarda parametre olarak gönderilen değerler fonksiyona dizi olarak aktarılmaktadır. Sınırsız parametre değişkeninin başına üç nokta yan yana konulmalı ve mutlaka fonksiyon tanımında son parametre olarak verilmelidir. Aşağıdaki sınırsız sayıda parametre alan bir fonksiyon örneği verilmiştir.

```
1   <?php
2   echo "Toplam: ".topla(5,7,2,10)."<br />";
3   echo "Toplam: ".topla(5,7,2,10,100)."<br />";
4   function topla(...$sayilar){
5       $toplam=0;
6       foreach($sayilar as $sayi){
7           $toplam += $sayi;
8       }
9       return $toplam;
10  }
```

Örnekteki fonksiyon, parametre olarak alınan sayıların toplamını hesaplayarak geri döndürmektedir. Fonksiyon çağrılırken yazılan parametreler, $sayılar isimli dizinin elemanları şeklinde fonksiyona aktarılmış ve *foreach* döngüsü ile toplamları alınmıştır.

Öğrenme Çıktısı

4 Fonksiyon tanımlaması yapabilme

Araştır 4

1 - 1 - 2 - 3 - 5 - 8 - 13 - 21 şeklinde ilk iki sayısı 1 olan ve önceki iki sayının toplamı şeklinde devam eden Fibonacci dizisini hesaplayarak ekrana yazdıran for döngüsünü kurunuz.

İlişkilendir

Matematikte kullanılan fonksiyonlar ile programlama dillerinde kullanılan fonksiyonları ilişkilendiriniz.

Anlat/Paylaş

Programa dillerinde neden yaygın olarak fonksiyonlara başvurulduğunu açıklayınız.

SINIFLAR

PHP'de modern programlama dillerinde olduğu gibi sınıf tanımlaması yapılabilmektedir. PHP'nin 5 versiyonu ile sınıf tanımlama konusunda önemlide değişiklikler yapılmış ve tamamen yeni bir Nesne Modeli yazılmıştır. Programlama dillerinde sınıflar, kendi içinde tutarlılığı olan özel işleri modellemek için kullanılır. Bu işlerin yürütülmesinden tamamen tanımlanan bu sınıf sorumlu olacaktır. Böylece programlar gerçek hayatta olduğu gibi sınıflar ile modellenmiş ve daha anlaşılır olur. Sınıf tanımlaması ile kodlar düzenli bir şekilde yazılır ve tekrar tekrar kullanılabilir. Örneğin veritabanı ile işleri yürütmek için bir sınıf tanımlaması yapılabilir ve veritabanı ile ilgili tüm işler bu sınıf üzerinden yürütülebilir. Benzer şekilde web sayfasında kullanıcı işlemleri ile ilgili bir sınıf tanımlaması yapılabilir ve kullanıcı, girişi, çıkışı, yeni kullanıcı tanımlama gibi birçok işlem tamamen bu sınır üzerinden gerçekleştirilebilir.

Sınıflar kendi içinde sabit, değişken ve fonksiyon barındırabilirler. Bunlara sınıfın üyeleri denilmektedir. Sınıf tanımlama class kelimesi ile yapılmaktadır. Sınıfa ait her bir örneğe ise nesne adı verilmektedir. İşlemler genellikle sınıfın bir örneği olan nesneler üzerinden yapılmaktadır. Aşağıda sınıf tanımlama ile ilgili bir örneğe yer verilmiştir.

```
1   <?php
2   $a1 = new Araba();
3   $a1->marka="Anadol";
4   $a1->model=2009;
5   $a1->bilgileriYazdir();
6   $a2 = new Araba("Devrim", 2015);
7   $a2->bilgileriYazdir();
8
9   class Araba{
10      public $marka;
11      public $model;
12      public function Araba($mar,$mod){
13          $this->marka=$mar;
14          $this->model=$mod;
15      }
16      public function bilgileriYazdir(){
17              echo "Markası: ".$this->marka." - Modeli: "
    .$this->model."<br />";
18      }
19  }
```

PHP programlama dili ile aşırı yükleme, miras alma, sınıf soyutlama, arayüz tanımlama gibi nesne yönelimli programlama dillerinde bulunan ileri düzey özellikler kullanılabilmektedir.

Örnekte Araba isminde bir sınıf tanımlanmıştır. Bu sınıfın $marka ve $model olmak üzere iki üye değişkeni bulunmaktadır. Üye fonksiyon olarak $marka ve $model üye değişkenlerini ekrana yazdıran *bilgileriYazdir* fonksiyonu tanımlanmıştır. Ayrıca sınıf ismiyle aynı olan ve kurucu fonksiyon olarak adlandırılan bir fonksiyon daha tanımlanmıştır. Kurucu fonksiyonlar sınıfın bir örneğinin oluşturulması sırasında parametre sayısına bağlı olarak çalıştırılırlar. Farklı parametre sayısına sahip çok sayıda kurucu fonksiyon oluşturmak mümkündür. Tanımlanan kurucu fonksiyon iki parametre almakta ve aldığı bu parametreleri üye değişkenlere atamaktadır. Sınıfın bir örneği olan nesnelerin kendi üye değişken ve fonksiyonlarına ulaşmak içine 13, 14 ve 17. satırlarda olduğu gibi *$this* kelimesi kullanılmaktadır. *$this* nesneyi işaret eden bir referanstır. Ayrıca sınıf ve nesnelerin değişken ve fonksiyonlarına ulaşmak için C++'da olduğu gibi -> işaretinden yararlanılır.

2. ve 6. satırlarda sınıfın bir örneği olan *$a1* ve *$a2* nesneleri tanımlanmaktadır. Sınıfın bir örneği olan nesne tanımlamak için *new* kelimesi kullanılır. 3. ve 4. satırlarda sınıfın üye değişkeni olan *$marka* ve $model değişkenlerine değer ataması yapılmaktadır. 6. satırda sınıfın yeni bir örneği oluşturulurken parametre kullanıldığı için iki parametre alan kurucu fonksiyon çağrılmış olur. Örnekte de görüldüğü üzere Araba ile ilgili işler bir sınıf içinde toplanmıştır. Böylece hem kodların okunaklılığı artmakta hem birbirinden bağımsız ya da çok az bağımlı farklı işler farklı sınıflar tarafından yürütülmektedir.

Öğrenme Çıktısı

5 Sınıf tanımlaması yapabilme

Araştır 5

Sık sık iki değişkenin değerini değiştirmeyi gerektiren bir durumda, bu işi kolaylaştırmak için nasıl bir çözüm üretilebilir.

İlişkilendir

PHP programa dilinin hangi özellikleri PHP'nin ileri düzey bir nesne yönelimli programlama olduğunu gösterir?

Anlat/Paylaş

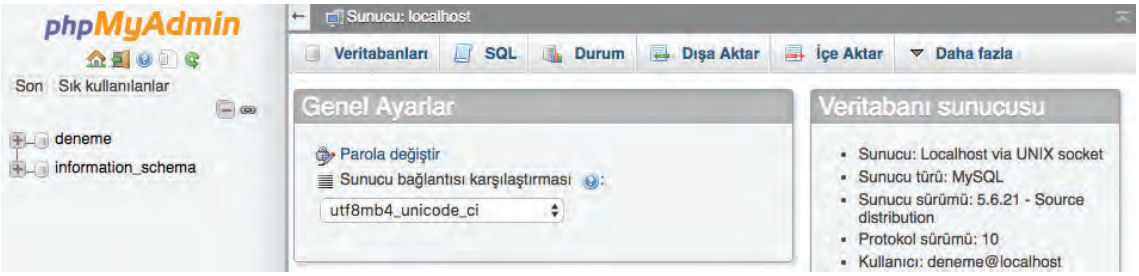
PHP'de sınıf ve nesnelerin ne amaçla kullanıldığını açıklayınız.

VERİTABANI İŞLEMLERİ

Web sayfaları için sunucu tarafında çalışan programlama dillerinin en önemli amaçlarından biri de veritabanı ile ilgili işleri yapabilmektir. Kullanıcı girişinin olduğu bir web sitesinde kullanıcı bilgileri, bir alışveriş web sitesinde siparişler ile ilgili bilgiler, bir kütüphane web sitesinde kitaplar ve ödünç alma/verme ile ilgili bilgilerin saklanması gibi birçok iş için veritabanları kullanılmaktadır. Veritabanları genellikle veritabanı sunucu yazılımları üzerinde oluşturulmaktadır. Sunucu tarafında çalışan programlama dilleri de bu veritabanı sunucu yazılımlarına kullanıcı adı ve şifre ile bağlanmakta ve SQL dili ile veritabanı ile ilgili işler yapabilmektedirler.

PHP programlama dili ile tüm popüler veritabani sunucu yazılımlarına bağlanılabilir. Piyasada Microsoft SQL Server, Oracle, IBM DB2, vb. ücretli ve MySQL, MariaDB, PostgreSQL, MongoDB, SQLite, vb. açık kaynak kodlu ve ücretsiz çok sayıda kendini kanıtlamış veritabanı sunucu yazılımı bulunmaktadır. PHP ile veritabanı sunucu yazılımlarına bağlanabilmek için o yazılımın uzantısının (extension-plugin) kurulu olması ve PHP'nin ayar dosyasından etkinleştirilmesi gerekmektedir. Özellikle PHP kullanıcılarının açık kaynak kodlu ve ücretsiz olan MySQL ve MariaDB veritabanı sunucu yazılımlarını tercih ettikleri söylenebilir.

Bu bölümde de basit bir şekilde MySQL sunucu yazılımı kullanılarak veritabanı işlerinin nasıl yapılabileceğinden bahsedilecektir. Öncelikle MySQL veritabanı yazılımının sisteminizde veya sunucunuzda kurulu ve çalışıyor olması gerekmektedir. Aynı zamanda MySQL'e bağlanmak için bir kullanıcı adı ve şifreye ihtiyaç bulunmaktadır. MySQL veritabanı ile ilgili işleri yapmak için phpMyAdmin yazılımı kullanılabilir. Bu yazılım genellikle MySQL olan sunucularda kuruludur. Ayrıca Wamp gibi yazılımlarda da bulunmaktadır. phpMyAdmin'e ulaşmak için sunucunun ya da kullanılan yazılımın yardımlarına bakmak uygun olacaktır. phpMyAdmin'in giriş yapıldıktan sonraki ekran görüntüsü Resim 3.1'de görülmektedir.



Resim 3.1 phpMyAdmin Ekran Görüntüsü

phpMyAdmin'e giriş yapıldıktan sonra sol tarafta giriş yapan kullanıcının yetkisinin olduğu veritabanları listelenmektedir. Şekilde kullanıcının sadece deneme isimli veritabanı üzerinde yetkili olduğu görülmektedir. Yine kullanıcı yetkilendirmelerine bağlı olarak yeni veritabanları oluşturulabilir. Sol bölümden bir veritabanı adı seçildiğinde sağ bölümde bu veritabanında bulunan tablolar gösterilmektedir. Tablo oluştur seçeneği kullanılarak yeni tablolar oluşturulabilir.

Bu bölümde Php ile MySQL'e bağlanmak için kullanılan mysqli uzantısı (MySQL Improved Extension) ile ilgili örneklere yer verilmiştir. MySQL uzantısı yordamsal veya nesne yönelimli programlama paradigmalarını desteklemektedir. Yordamsal programlama paradigmasında, MySQL'e bağlamak ve veritabanı işleri yapabilmek için geliştirilmiş fonksiyonlar kullanılmaktadır. Nesne yönelimli programlama paradigmasında ise MySQL bağlantılarını yönetmek için geliştirilmiş sınıflardan yararlanılır. İki paradigma arasında geçiş yapılabilmektedir, ancak bu durum kodların daha karmaşık olmasına neden olacağı için önerilmemektedir. Bu bölümde nesne yönelimli paradigma tercih edilmiştir. PHP'ye mysqli uzantısı yüklendiğinde MySQL bağlantılarını yönetmek için *mysqli* isimli bir sınıf kullanılabilir hale gelmektedir. Bu sınıf kullanılarak MySQL bağlantıları yönetilebilir.

```
1   <?php
2   $baglanti = new mysqli("localhost","deneme","deneme123","deneme");
3   if ($baglanti->connect_errno) {
4       echo "Bağlantı Başarısız: (" . $baglanti->connect_errno . ") "
            .$baglanti->connect_error;
5   }else{
6       echo "Bağlantı Başarılı";
7   }
8   $sorgu = $baglanti->query("SELECT * FROM siparis");
9   while($kayit = $sorgu->fetch_assoc()){
10      echo "Kayit ID'si: ".$kayit["id"]."<br />";
11  }
```

Yukarıdaki örnekte *mysqli* sınıfı türünde *$baglanti* isimli bir nesne oluşturulmuştur. Nesne oluşturulurken sınıfın 4 parametre alan kurucu fonksiyonu çağrılmıştır. Bu kurucu fonksiyonun aldığı parametreler ise sırayla şöyledir: sunucu adı veya ip adresi, kullanıcı adı, kullanıcı şifresi ve veritabanı adı. Eğer parametredeki bilgiler doğru ise kurucu fonksiyon çağrıldığı anda yerel bilgisayarda (localhost) çalışan MySQL sunucu yazılımında bulunan deneme (son parametrede belirtilen) veritabanına deneme kullanıcı adı ve

deneme123 şifresi ile bağlantı yapılmış olur. 3. satırda bulunan if ile bağlantının başarılı olup olmadı kontrol edilmektedir. Eğer MySQL sunucu yazılımına bağlantı sağlanamaz ise $baglanti->connect_errno üye değişkeninin değeri ortaya çıkan bağlantı problemi ile ilgili hata koduna (bir pozitif tamsayı) sahip olmakta ve $baglanti->connect_error üye değişkenin değerinde ise bağlantı problemi ile ilgili bir açıklama yer almaktadır.

Veritabanı üzerinde sorgu çalıştırmak için *mysqli* sınıfının *query* üye fonksiyonu çağrılmalıdır. Bu fonksiyona SQL sorgusu parametre olarak verilmelidir. Fonksiyon geriye *mysqli_result* sınıfına ait bir nesne döndürmektedir. Bu nesne ise sorgunun sonucu ile ilgili bilgiler tutmaktadır. Örnekte $sorgu değişkeni *mysqli_result* sınıfına ait bir nesnedir ve "SELECT * FROM siparis" sorgusunun sonucu ile ilgili bilgiler tutmaktadır. Eğer SQL *select* sorgusu yapılmış ise *mysql_result* sınıfının fetch_assoc fonksiyonu kullanılarak sorgu sonucu gelen kayıtlara tek tek ulaşılabilir. Örneğin *select* sorgusu sonucunda 3 kayıta ulaşıldığı varsayılacak olursa, *fetch_assoc* üye fonksiyonu ilk çağrıldığında sadece birinci kaydı döndürür, ikinci kez çağrıldığında ikinci kaydı, üçüncü kez çağrıldığında ise üçüncü kaydı döndürür. Dördüncü kez çağrıldığında ise dördüncü kayıt olmadığı için *false* değerini döndürür. Böylece while döngüsü sonlandırılmış olur. *fetch_assoc* üye fonksiyonu sonucunda dönen kayıt, anahtarlı bir dizidir. Anahtarlar ise tablondaki alan adlarıdır. Örnekte döngü içinde ilgili kaydın *id* alanının değeri yazdırılmaktadır.

Bu noktaya kadar öğrenilen bilgiler ışığında Tablo 3.2'de yapısı verilen veritabanı ile ilgili örnek bir uygulamaya yer verilmiştir. Tablo adı ve alan adları verilen veritabanında müşterilere ait sipariş bilgileri tutulmaktadır. Sadece *siparis* isimli tablo ile ilgili olarak kayıt ekleme, düzenleme ve silme işlemleri yapan örnek bir uygulama yapılmıştır. *musteri* ve *urun* tablolarına ise örnek olması açısında phpMyAdmin üzerinden veri girişi yapılmıştır.

Tablo 3.2 Örnek Tablo Yapısı

siparis	musteri	urun
id	id	id
musteri_id	ad	ad
urun_id	soyad	fiyat
tarih	tel	aciklama
adet	adres	
fiyat		

Sipariş tablosundaki verileri gösteren siparis.php dosyası aşağıdaki gibidir.

```
1   <h1>Siparişler</h1>
2   <?php
3   $baglanti = new mysqli("localhost","deneme","deneme123","deneme");
4   if ($baglanti->connect_errno) {
5       echo "Bağlantı Başarısız: (" . $baglanti->connect_errno . ") "
          .$baglanti->connect_error;
6   }
7   $baglanti->set_charset("utf8");
8   $sorgu = $baglanti->query("SELECT siparis.id as id, urun.ad as urun_ad, "
9           . "musteri.ad as musteri_ad, musteri.soyad as musteri_soyad, "
10          . "siparis.adet as adet, siparis.fiyat as toplam_fiyat, siparis.tarih as
    tarih "
11          . "FROM siparis "
12          . "INNER JOIN musteri ON siparis.musteri_id = musteri.id "
13          . "INNER JOIN urun ON siparis.urun_id = urun.id ");
14  echo "<a href='duzenle.php'>Yeni Sipariş</a>";
15  echo "<table border=1>";
```

```
16    echo "<tr><th>Ürün</th><th>Müşteri</th><th>Adet</th><th>Toplam
Fiyat</th><th>Tarih</th><th>İşlem</th></tr>";
17
18  while($kayit = $sorgu->fetch_assoc()){
19    echo "<tr>";
20    echo "<td>".$kayit["urun_ad"]."</td>";
21    echo "<td>".$kayit["musteri_ad"]." ".$kayit["musteri_soyad"].  "</td>";
22    echo "<td>".$kayit["adet"]."</td>";
23    echo "<td>".$kayit["toplam_fiyat"]."</td>";
24    echo "<td>".$kayit["tarih"]."</td>";
25    echo "<td><a href='sil.php?id=".$kayit["id"]."'>Sil</a> ";
26    echo "<a href='duzenle.php?id=".$kayit["id"]."'>Düzenle</a></td>";
27    echo "</tr>";
28  }
29    echo "</table>";
```

Yukarıdaki kodlarda görüldüğü gibi bu sayfada bir *select* sorgusu bulunmakta ve bu sorgu sonucunda elde edilen kayıtlar *while* döngüsü kullanılarak bir tablo olarak ekrana yazdırılmaktadır. Ayrıca her bir kayıt için tablonun son sütununda sil ve düzenle bağlantıları konulmuştur. Bu bağlantılara ilgili kaydın *id* alanının değeri *get* değişkeni olarak gönderilmektedir. Böylece silme veya düzenleme yapılan sayfada hangi kaydın silineceği veya düzenleneceği *id* alanı kullanılarak tespit edilebilir. Aşağıdaki *sil.php* dosyasının kodları yer almaktadır.

```
1   <?php
2   $baglanti = new mysqli("localhost","deneme","deneme123","deneme");
3   if ($baglanti->connect_errno) {
4       echo "Bağlantı Başarısız: (" . $baglanti->connect_errno . " ) "
            . $baglanti->connect_error;
5   }
6   if(!empty($_GET['id'])){
7        $sorgu = $baglanti->query("DELETE FROM siparis WHERE
    id='".$_GET['id']."'");
8       header('Location: siparis.php');
9   }
```

Kayıt silme işi yapan dosyada *get* değişkeni olarak gelen *id* değeri kullanılarak bir *delete* sorgusu ile kayıt silinmekte ve hemen ardından sayfa *header* komutu ile *siparis.php* sayfasına yönlendirilmektedir. Yeni kayıt ekleme ve var olan kaydı düzenleme sayfasının (duzenle.php) kodları ise aşağıda verilmiştir.

```
1   <?php
2   $baglanti = new mysqli("localhost","deneme","deneme123","deneme");
3   if ($baglanti->connect_errno) {
4     echo "Bağlantı Başarısız: (" . $baglanti->connect_errno . ") "
5       . $baglanti->connect_error;
6   }
7   $baglanti->set_charset("utf8");
8   $id=empty($_REQUEST['id'])?0:$_REQUEST['id'];
9   $musteri_id=0;
10  $urun_id=0;
11  $tarih='';
12  $adet='';
13  $fiyat='';
14  if(!empty($_POST["kaydet"])){
15    if($id==0){
```

```
16      $baglanti->query("INSERT INTO siparis (musteri_id,urun_id,"
17        . "tarih,adet,fiyat) "
18        . "VALUES('".$_POST["musteri_id"]."','".$_POST["urun_id"]."',"
19        .                    "'".$_POST["tarih"]."','".$_POST["adet"]."','"
          . $_POST["fiyat"]."')");
20  }else{
21    $baglanti->query("UPDATE siparis SET "
        . "musteri_id='".$_POST["musteri_id"]."', "
22      . "urun_id='".$_POST["urun_id"]
        . "', tarih='".$_POST["tarih"]."', "
23      . "adet='".$_POST["adet"]."', fiyat='".$_POST["fiyat"]."' "
24      . "WHERE id='$id'");
25  }
26  header("Location: siparis.php");
27 }
28 if($id>0){
29  $sorgu = $baglanti->query("SELECT * FROM siparis WHERE
       id='".$_GET['id']."'");
30  $kayit = $sorgu->fetch_assoc();
31  $id=$kayit["id"];
32  $musteri_id=$kayit["musteri_id"];
33  $urun_id=$kayit["urun_id"];
34  $tarih=$kayit["tarih"];
35  $adet=$kayit["adet"];
36  $fiyat=$kayit["fiyat"];
37 }
38 ?>
39 <h1>Siparis <?php echo $id==0?"Ekle":"Düzenle"; ?></h1>
40 <form method='post' action='duzenle.php'>
41 <input type='hidden' name='id' value='<?php echo $id; ?>' />
42 Müşteri: <select name="musteri_id">
43    <?php
44      $sorgu_musteri = $baglanti->query("SELECT * FROM musteri");
45      while($kayit = $sorgu_musteri->fetch_assoc()){
46      echo      "<option      value='".$kayit["id"]."'>".$kayit["ad"]."      "
          .$kayit["soyad"]."</option>";
47      }
48    ?>
49 </select><br />
50 Ürün: <select name="urun_id">
51  <?php
52    $sorgu_urun = $baglanti->query("SELECT * FROM urun");
53    while($kayit = $sorgu_urun->fetch_assoc()){
54      echo "<option value='".$kayit["id"]."'>".$kayit["ad"]
         ." (".$kayit["fiyat"]."TL)</option>";
55    }
56    ?>
57 </select><br />
58 Tarih:<input name='tarih' value='<?php echo $tarih; ?>' /> <br />
59 Adet:<input name='adet' value='<?php echo $adet; ?>' /> <br />
60 Fiyat:<input name='fiyat' value='<?php echo $fiyat; ?>' /> TL<br />
61 <input type='submit' name='kaydet' value='<?php echo
   $id==0?"Kaydet":"Güncelle"; ?>' />
62 </form>
```

duzenle.php dosyası hem yeni kayıt eklemek hem de var olan bir kaydı düzenlemek için kullanılabilmektedir. Yedinci satırda $_REQUEST['id']) değişkeninin değeri okunmaktadır. *get* veya *post* yöntemiyle gönderilen *id* değişkeninin değeri boş veya 0 ise yeni bir kayıt yapılacak şekilde planlanma yapılmıştır. Böylece eğer *duzenle.php* doğrudan açıldığında yeni kayıt eklenecek şekilde ayarlanmaktadır. 38. satırda form etiketinde *action* özelliği olarak *duzenle.php* değerinin girildiği görülecektir. Bu durumda form gönderildiğinde (submit) post değerleri yine aynı dosya olan *duzenle.php* dosyasına yönlendirilmektedir. Böylece tek bir dosya ile kayıt işlemi gerçekleştirilebilir. Bu bağlamda 13. satırda post yöntemiyle kaydet değişkeninin gönderilip gönderilmediği kontrol edilmektedir. Bir başka deyişle formda kaydet düğmesine basılarak *submit* işleminin yapılıp yapılmadığı tespit edilmektedir. Eğer *submit* işlemi yapılmış ise ve *id* değişkeninin değeri 0 ise bir insert sorgusu, *id* değişkeninin değeri 0'da farklı ise *update* sorgusu oluşturulmuş ve sayfa *siparis.php* dosyasına yönlendirilmiştir. 26. satırdaki kontrolde eğer *id* değişkeninin değeri 0'da büyük ise bu *id* ile ilgili kayıt *siparis* tablosundan çekilmektedir. Böylece eğer bir düzenleme işlemi yapılıyor ise form içindeki alanlar düzenlenecek kaydın var olan değerleri ile açılacaktır ve kullanıcı sadece düzenlemek istediği alanı seçerek kaydedebilecektir. Bunun içinde 58. satırda görülebileceği gibi *siparis* tablosundan çekilen değer *input* etiketinin *value* özelliğine yazdırılmaktadır. Formda 40. ve 48. satırlarda ürün ve müşteri bilgileri *select* etiketi ile açılan liste şeklinde verilmiştir. Select etiketlerinin içinde bulunan *option* etiketleri *musteri* ve urun tablolarından çekilen değerler ile oluşturulmuştur.

Bu bölümde php ve MySQL kullanılarak çok basit bir veritabanı uygulaması yapılmıştır. Bu uygulamada kullanılan SQL sorgularında hiçbir kontrolden geçmeden doğrudan kullanıcı girişleri kullanılmıştır. Kötü niyetli kullanıcılar bu girişlerde SQL sorguları gönderebilir ve kullanıcıların yazdığı bu SQL sorguları veritabanı üzerinde çalıştırılır. SQL Injection denilen bu yöntem kötü niyetli kullanıcılarının veritabanı üzerinde istenmeyen işlemleri yapabilmesine olanak sağlamaktadır. Bu bakımdan kullanıcı tarafından girilen tüm bilgiler bir kontrol mekanizması ile kontrol edilmesi çok önemlidir.

Ayrıca genellikle bir tür uygulamalarda bir kullanıcı girişi bulunmaktadır. PHP ile sistemde kayıtlı kullanıcıların sisteme giriş ve çıkışlarını kontrol etmek için genellikle $_SESSION değişkeni kullanılmaktadır. $_SESSION değişkeni sunucuya yapılan her bağlantı için sunucu tarafında oluşturulmaktadır. Örneğin bir bilgisayardaki Chrome tarayıcısı ile sunucuya bağlantı yapıldığında bir session değişkeni oluşturulur ve aynı bilgisayardaki Chrome tarayıcısı ile yapılan bağlantılarda aynı *session* değişkeni kullanılmaya devam edilir. Eğer başka bir tarayıcı ile aynı sunucuya bağlanılmak istenirse yeni bir *session* değişkeni oluşturulur. Aynı zamanda Chrome tarayıcısı kapatılıp tekrar açılarak tekrar aynı sunucuya bağlanılmak istenirse yine yeni bir session değişkeni oluşturulur. Session değişkeni sunucu tarafında tutulmaktadır. Bazı web sitelerinde kullanıcı girişinde Beni Hatırla seçeneği bulunmaktadır. Bu seçenek seçildiğinde tarayıcı veya bilgisayar kapatılıp açıldığında dahi kullanıcı bilgileri hatırlanmaktadır. Bunun için ise her bir sunucu için tarayıcı üzerinde saklanan *cookie* değişkenleri kullanılmaktadır. Php ile *cookie* kullanarak kullanıcının kullandığı tarayıcıda istenile süre boyunca bilgi saklamak mümkündür. Ancak bu bilgi kullanıcı tarafında olduğu için çok güvenilir değildir. Kullanıcı tarayıcının özelliklerinden bu *cookie* bilgilerini temizleyebilir. $_SESSION ve *cookie*'lerin detaylı kullanımı için PHP'nin referansından yararlanılabilir.

Öğrenme Çıktısı

6 Veritabanı işlemleri yapabilme

Araştır 6

PHP'de MySQL bağlantısı kurmak için nesne1 adlı bir nesne ve server adlı bir sunucuda veritabani1 adlı bir veritabanına admin kullanıcı adı ve admin123 şifresiyle bağlantı kurmak için yazılacak mysqli isimli sınıfın PHP kodu nasıl olur?

İlişkilendir

Web sitelerinde paylaşılan içeriklerin depolanması, sınıflandırılması ve geri çağırılması süreçlerini göz önünde bulundurarak veritabanlarını internetin yaygınlaşmasıyla ilişkilendiriniz.

Anlat/Paylaş

Bilgi sahibi olduğunuz veritabanı sunucu yazılımlarıyla ilgili bilgilerinizi paylaşınız.

ÇATILAR VE LARAVEL

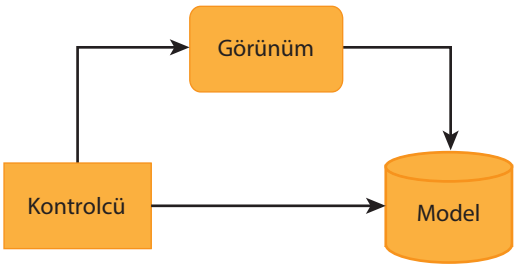
Son yıllarda web uygulama geliştirme alanında çatı (framework) kullanımının önemi artmıştır. Web uygulama geliştirme alanında çalışan ve web uygulama geliştirme çatılarının kullanımı hakkında bilgisi olmayan yok denecek kadar azdır. Web uygulama geliştirme çatıları ile yazılım geliştirme süreci önemli derecede hızlanmaktadır. Çatılar, uygulamalar için gerekli olabilecek temel işlevleri ve araçları bünyelerinde bulundurmaktadırlar. Ayrıca çatılarda bulunan güvenlik araçları sayesinde web sitelerinin daha güvenli olmasını sağlanmakta ve dış saldırılardan etkilenme olasılığı düşmektedir.

Çatılar, birçok uygulamada bulunan benzer öğelerin kullanılmalarına yönelik ortak bir tasarım ortaya koymaktadır. Genel olarak tüm uygulamalarda ortak olarak değerlendirilebilecek temel öğeler bulunmaktadır. Çatılar, bu ortak temel öğeler için bir alt yapı ve bir tasarım deseni ortaya koymaktadırlar. Böylece uygulama geliştirici veri tabanı ve kullanıcı yönetimi gibi temel konularda daha az zaman harcayarak asıl zamanını uygulama üzerine yoğunlaştırması mümkün olabilmektedir. Ayrıca çatılar, programcıları belli kurallar içinde kod yazma zorlamaktadır. Böylece geliştirilen kodlar daha basit, anlaşılır ve hataları daha kolay ayıklanabilir olmaktadır. Eğer web uygulama geliştirme çatısı kullanmayan bir programcı biraz düzensiz çalışırsa, işin içinden çıkılamayacak karmaşık kodlar üretebilmektedir. Çatılar bu durumun üstesinden gelebilmek için bir tasarım desenini kullanmayı zorunlu hale getirmektedirler.

PHP çatılarının çoğu MVC (Model-View-Controler, Model-Görünüm-Kontrolcü) tasarım desenini kullanmayı zorunlu kılmaktadırlar. MVC tasarım deseni, kullanıcı etkileşimi gerektiren uygulamalar için geliştirilen tasarım desenlerinden en yaygın olarak kullanılanıdır (Abeysinghe, 2009). Tasarım desenlerinin ortaya çıkmasındaki sebep, daha önce birçok geliştiricinin karşılaşmış oldukların problemlere ortak bir çözüm yolu sunuyor olmasıdır. Örneğin MVC tasarım deseni, uygulamanın görünüm (sunum) katmanı, kontrolcü (business logic) katmanı ve model (veri yönetimi) katmanının nasıl birbirinden ayrılabileceği konusunda geliştiricilere yardımcı olur.

Model katmanı; veri tabanı ile ilgili işlerde kullanılan katmandır. Görünüm katmanı, kullanıcı arayüzünün geliştirilmesi için kullanır, HTML, CSS ve JavaScript gibi kodlar bu bölümde bulunur. Kontrolcü katmanı ise organizasyonun yapıldığı bölümdür. Bu katman ile geliştirici sistemdeki Modelleri ve Görünümleri yöneterek uygulamanın kullanıcıları ile olan etkileşimleri yönetir. Kontrolcü genel olarak hesaplamaların ve yönetim işlerinin yapıldığı bölümdür. Model ve Görünüm katmanlarının Kontrolcü katmanına ulaşması mümkün değildir. Görünüm katmanı Model katmanına ulaşılabilmektedir.

Şekil 3.2'de de görüldüğü üzere sistem Kontrolcü katmanı tarafından yönetilmektedir. Görünüm katmanı doğrudan bir Modelin kullanıcı arayüzüne dönüştürülmesi şeklinde olabilmektedir. Görünüm katmanının hangi Model katmanına ulaşabileceği de çoğunlukla Kontrolcü katman tarafından belirlenmektedir.



Resim 3.2 MVC Yapısı

MVC tasarım desenin kullanılmadığı bir web uygulamasında veri tabanına erişim, yönetim ve kullanıcı arayüzü kodlarının tamamı tek bir PHP dosyasına yazılabilir. Bu yazılacak kod miktarını azaltacak ve kodların çalışma performansını arttıracaktır. Ancak bu yazım şekli bazı önemli problemleri beraberinde getirmektedir. Web uygulamalarında kullanıcı arayüzlerinde sık sık güncellemeler ve değişimler olabilmektedir. Uygulamanın yönetim ve veri yönetimi bölümlerinde ise daha seyrek değişimler olur. Eğer tüm sistem tek bir PHP dosyası içine yazılırsa, uygulamanın yönetim ve veri yönetimine dokunmadan sadece kullanıcı arayüzünü değiştirmede sıkıntılar olabilmektedir. Uygulama büyüdükçe, kod miktarı arttıkça ve karmaşıklaştıkça, uygulamanın yönetimi zorlaşacak, anlaşılırlığı azalacak ve buna bağlı problemler çoğalacaktır. Bu nokta MVC deseni ile karmaşık uygulamalarda Model, Görünüm ve Kontrolcü katmanlarının birbirlerinden izole edilerek sistemin karmaşıklığı daha anlaşılır ve yönetilebilir bir boyuta getirilmektedir. Ayrıca bu sistem ile diğer katmanlara dokunmadan Görünüm katmanında rahatlıkla değişiklikler yapılabilmektedir.

Laravel çatısı da MVC desenini destekleyen bir PHP web uygulama geliştirme çatısıdır. Laravel çatısı MVC desenini kullanmayı zorunlu kılmaktadır. Açık kaynak kodlu olan Laravel'in geliştirilmesine tüm dünyadan çok sayıda geliştirici destek vermektedir ve dünyada yaygın kullanılan PHP çatılarından biridir (Winesett, 2010).

Laravel çatısı MVC tasarım desenini kullanmayı mecbur bırakması yanında Elequent teknolojisi ile veri tabanı yönetimi, uluslararasılaştırma teknolojisi ile web uygulamasını çoklu dil desteği, ön-bellekleme, kullanıcı yönetimi ve rol tabanlı erişim kontrolü ve test araçları vb. gibi özellikler ve araçlar içermektedir. Bu özellikler ve araçlar sayesinde Laravel çatısı uygulama geliştirme zamanını önemli ölçüde azaltabilmekte ve Laravel'in kurallarına uygun davranıldığında oldukça güvenli sistemler ortaya konabilmektedir. Laravel çatısı birçok özelliği içinde barındıran oldukça kapsamlı bir çatıdır. Bu bakımdan bu bölümde Laravel çatısı detaylı incelenmemiş sadece kontrolcü, model ve görünüm katmanları ile ilgili birkaç örnek koda yer verilmiştir.

```
1   <?php
2   Route::get('/register', 'UserController@actionRegister');
3   Route::post('/user/register', 'UserController@actionRegister');
4   Route::get('/login', 'UserController@actionLogin');
5   Route::post('/login', 'UserController@handleLogin');
6   Route::get('/logout', 'UserController@logout');
7   Route::get('/change/password', 'UserController@changePassword');
8   Route::post('/change/password/store',
    'UserController@changePasswordStore');
```

Laravel'de öncelikle *routes.php* dosyası kullanılarak yollar tanımlanmalıdır. Yukarıda *routes.php* dosyasından örnek kodlar yer almaktadır. Bu dosyadaki bilgilere göre sunucunun *localhost* üzerinde kurulu olduğun varsayarsak, *localhost/login* adresi yazıldığında *UserController* kontrolcüsündeki *actionLogin* fonksiyonu çalışır. Yollar belirtilirken aynı zamanda bu sayfaya hangi yöntemle ulaşılacağı ile bilgide belirtilmelidir. Örneğin /login yolu için get ve post yöntemlerine yönelik farklı tanımlar yapılmıştır. Kullanıcı *actionLogin* fonksiyonundaki belirtilen formu doldurarak (kullanıcı adı ve şifre bilgilerini girerek) formu gönderdiğinde post yöntemiyle göndermiş olur. Böylece post yöntemi kullanıldığı için *handleLogin* fonksiyonu çalışacak ve formdan gelen veriler işlenerek kullanıcının sisteme giriş yapabilmesi sağlanacaktır. Aşağıda *UserController* isimli kontrolcü dosyasından iki fonksiyon örnek olarak verilmiştir. Kontrolcü dosyaları tanımlanırken *BaseController* sınıfı miras alınarak tanımlanmalıdır. Örnekte verilen kontrolcüde iki üye fonksiyon bulunmaktadır ve bu fonksiyonlar için routes.php dosyasında yollar tanımlanmıştır.

```
1   <?php
2   class UserController extends BaseController {
3     public function changePassword(){
4       return View::make('user.change_password');
5     }
6     public function changePasswordStore(){
7       if(!Auth::check()) {
8         return Redirect::route('login');
9       }
10      $user = User::find(Auth::id());
11      if (!Hash::check(Input::get('password_old'), $user->password)) {
12        return Redirect::route('/change/password',array(
13          'alertMessage'=>'wrong_password_alert',
14          'alertClass'=>'danger',
15          'alertDelay'=>'5000'))->withInput();
16      }
17      if(Input::get('password') !==
    Input::get('password_confirmation')){
18        return Redirect::route('/change/password',array(
19          'alertMessage'=>'confirm_does_not_match',
20          'alertClass'=>'danger',
21          'alertDelay'=>'5000'))->withInput();
22      }
23      $user->password = Hash::make(Input::get('password'));
24      $user->save();
25      return Redirect::route('/profile',array(
26        'alertMessage'=>'password_changed',
27        'alertClass'=>'success',
28        'alertDelay'=>'5000'))->withInput();
29    }
30  }
```

/change/password yolu çağrıldığında *UserController* sınıfının *changePassword* fonksiyonu çağrılır. Bu fonksiyon ise *View::make* ile *user.change_password* görünüm dosyası çağrılmakta ekranda görünmesi sağlanmaktadır. *changePasswordStore* fonksiyonu ise *change_password* görünüm dosyasından post yöntemiyle gönderilerin yeni şifre bilgilerini kontrol ederek veritabanına yazmaktadır. Laravel'de sisteme giriş yapan kullanıcılar ile ilgili bilgiler *Auth* sınıfı ile yönetilmektedir. 7. satırda bulunan kontrol ile daha önce bir kullanıcı girişi yapılıp yapılmadığı kontrol edilmekte ve eğer kullanıcı girişi yapılmamış ise fonksiyonun geri kalanı çalıştırılmadan doğrudan */login* yoluna yönlendirilmektedir. Böylece bu fonksiyonun sadece sisteme giriş yapmış kullanıcılar tarafından çalıştırılacak olması garanti altına alınmıştır. 10. satırda User

modeli kullanılarak sisteme giriş yapmış kullanıcıların bilgileri veritabanından çekilmiştir. Burada aslında bir SQL *select* sorgusu çalıştırılmıştır. Ancak programcı bu noktada SQL sorguları ile uğraşmamakta, bu iş *Elequent* sınıfını miras alan User sınıfı tarafından yapılmaktadır. 11. satırda kullanıcının giriş yaptığı eski şifre bilgisinin doğru olup olmadığı *Hash* sınıfı ile kontrol edilmiştir. Ayrıca kullanıcının form üzerine girdiği bilgiler *Input* sınıfı ile çekilmiştir. Kullanıcıların formlar üzerinde yapmış oldukları girişler *Input* sınıfı tarafından kontrol edilmekte ve SQL Injection gibi güvenlik açıkları verilmemektedir. Eğer kullanıcının girmiş olduğu şifre bilgisi veritabanındaki şifre ile örtüşmüyor ise bir hata mesajı ile *change_password* görünüm dosyası tekrar çağrılmakta ve kullanıcının şifre değiştirme formunu tekrar doldurması beklenmektedir. Formda kullanıcının yeni şifresini iki defa girmesi gerekmektedir. 17. satırda ise bu iki şifrenin aynı olup olmadığı kontrol edilmiş ve eğer aynı değil ise bir hata mesajı ile *change_password* görünüm dosyası tekrar çağrılmıştır. 11 ve 17. satırlardaki karşılaştırma ifadelerinin yanlış olması durumunda ise 23. satırdaki komut ile User modelinin password özelliği, bir başka deyişle *users* tablosunun *password* alanına yeni şifrenin *Hash* sınıfı ile kriptolanmış değeri aktarılmaktadır. 24. satırda ise güncellenen *User* modelinin veritabanına yazılması sağlanmış olur. Şifre değiştirme işlemi başarılı bir şekilde yapıldığında kullanıcının profil sayfasına yönlendirilmektedir. Aşağıda örnekte kullanılan User modelinin kodlarına yer verilmiştir. Modeller *Elequent* sınıfını miras almalıdırlar. Örnekteki kullanılan model aynı zamanda kullanıcı giriş bilgilerini barındırdığı için *UserInterface, RemindableInterface* nesne arayüzlerini kullanmaktadır. Modeller oluşturulurken *$table* isminde bir üye değişken tanımlanmalı ve bu modelin veritabanında hangi tabloyu temsil ettiği belirtilmelidir. User modeli veritabanında users tablosu ile ilişkilendirilmiştir.

```
1   <?php
2   use Illuminate\Auth\UserTrait;
3   use Illuminate\Auth\UserInterface;
4   use Illuminate\Auth\Reminders\RemindableTrait;
5   use Illuminate\Auth\Reminders\RemindableInterface;
6   class User extends Eloquent implements
    UserInterface,RemindableInterface {
7       use UserTrait, RemindableTrait;
8       protected $table = 'users';
9   }
```

Son olarak Laravelde kullanılabilecek bir görünüm örneği aşağıda verilmiştir. Bu örnek yukarıda kontrolcü tarafından çağrılan *change_password* görünümüdür. Laravel'de görünümler çoğunlukla uzantısı *blade.php* olarak kaydedilmektedir. Böylece güçlü bir şablon aracı olan *blade*'ın kullanılması sağlanmış olur.

```
1   @extends('layouts.master')
2   @section('content')
3   <h2>{{trans('g.change_password')}}</h2>
4   <form method="post" action="{{URL::to('user/changepassword')}}">
5     <label for="password">{{trans("g.password")}}</label>
6   <input type="password" class="form-control" name="password"
    id="password" /><br />
7   <label for="password2">{{trans("g.password_repeat")}}</label>
8   <input type="password" class="form-control" name="password2"
    id="password2" />
9   <input type="hidden" name="_token" value="{{ csrf_token() }}">
10  <br />
11  <input type="submit" value="{{trans('g.change_password')}}"
    class='btn btn-primary' />
12  </form>
13  @endsection
```

blade sayesinde @ işareti ile çeşitli komutlar kullanılabilmektedir. Örnekte *extends* komutu ile var olan görünümün *layouts.master* görünümünün içinde görüntülenmesi sağlanmış olur. Böylece *layouts.master* görünüm dosyası içinde temel HTML komutları ve web sitesinin temel yapısı ve ana menü seçenekleri yer alabilir. *change_password.blade.php* dosyasının içeriği ise *masters.blade.php* dosyasının *content* bölümünde görüntülenir. Laravel'de farklı dillere ait içerikler *resources/lang* klasöründe yer alır. Bu klasör içine istenilen dil için bir klasör oluşturularak içine dil dosyaları konulmalıdır. *trans* fonksiyonu ise sistemin diline göre uygun dil klasöründeki içeriği bulup görüntülenmesini sağlar. Böylece kolaylıkla web sitesinin birçok farklı dilde görüntülenmesi sağlanabilmektedir.

Öğrenme Çıktısı

7 Çatılar ve Laravel ile ilgili bilgi verebilme

Araştır 7

Programlama dilinde miras alma nedir? Açıklayınız.

İlişkilendir

Laravel ile Model View Controller (MVC) mimarisini ilişkilendiriniz.

Anlat/Paylaş

Birden fazla dilde görüntüleme olanağı sağlayan Web sitelerinin bunu nasıl başardığını açıklayabilir misiniz?

1 Değişken türleri ve dizileri kullanabilme

Değişkenler ve Değişken Türleri

PHP değişken tanımlama konusunda çok esnek bir yapıya sahiptir. Değişkenler tanımlamaya gerek kalmadan doğrudan değer ataması yapılarak kullanılabilirler. Değişkenin türü atanan değere göre otomatik olarak belirlenir. PHP'de dört temel tür bulunmaktadır: boolean, integer, float ve string. Temel türler dışında bir de karmaşık türler bulunmaktadır. Dizi oluşturmak için karmaşık türlerden biri olan Array türünü kullanmak gerekmektedir. PHP, dizi kullanımı ile ilgili de oldukça esnek bir yapı sunmaktadır. Anahtarlı diziler tanımlanabileceği gibi, çok boyutlu dizeler de tanımlanabilir. Ayrıca bir dizi farklı türde elemanlara sahip olabilmektedir.

2 HTML formlarından gelen verileri kullanabilme

HTML Formları

Kullanıcı etkileşimli web sayfaları geliştirmenin en önemli araçlarından biri de HTML formlarıdır. HTML formları ile kullanıcılar web sayfasına bilgi girişi yapabilmektedirler. Kullanıcılar tarafından girilen bu bilgiler PHP dosyaları tarafından işlenebilmektedir. Formlara girilen veriler get ve post olmak üzere iki yöntem ile gönderilebilmektedir. PHP ile farklı yöntemler ile gönderilen bu veriler $_POST ve $_GET dizileri kullanılarak alınabilmektedir. Ayrıca $_REQUEST dizisi ile hem get hem de post yöntemiyle gönderileri verileri almak mümkündür.

3 Kontrol yapılarını kullanabilme

Kontrol Yapıları

PHP programlama dili, diğer programlama dillerinde olduğu gibi kontrol yapılarına sahiptir. Bu kontrol yapılarının söz dizimi, PHP programlama dili ile benzerlik gösteren C++, Java, C#, JavaScript gibi dillerinin kontrol yapılarındaki söz dizimi ile hemen hemen aynıdır. If komutunda bir kod bloğunun çalışıp çalışmayacağı bir karşılaştırma ifadesinin sonucuna bağlı olarak gerçekleşmektedir. for, while, ve do while komutları ile bir kod bloğunun istenilen sayıda tekrar tekrar çalışabilmesini sağlayan döngüler oluşturulabilmektedir. switch komutu ise if komutuna benzer bir kontrol mekanizmasıdır. Goto komutu ile program akışı değiştirilebilmektedir. include ve require komutları ise bir PHP dosyası başka bir php dosyası içine dahil edilebilmektedir.

öğrenme çıktıları ve bölüm özeti

4 Fonksiyon tanımlaması yapabilme

Fonksiyonlar

Fonksiyonlar daha sonra çalıştırmak için programcı tarafından yazılan kod bloklarıdır. Bu kod blokları defalarca çağrılarak çalıştırılabilir. Böylece benzer kodlar tekrar tekrar yazılmamış olur. Fonksiyonlar programların daha anlaşılabilir olmasını sağlamak amacıyla kullanılabilmektedir. Fonksiyonlara parametreler yardımıyla veri gönderilmektedir. Fonksiyonlardan geriye ise return komutu ile değer döndürülebilmektedir. Fonksiyonlar içinde tanımlanan değişkenler yerel değişkendir ve sadece o fonksiyon içinde geçerlidirler, fonksiyon dışında kullanılamazlar. Doğrudan PHP dosyasın içinde tanımlanan değişkenler ise küresel değişkenlerdir. Küresel değişkenlere her yeden ulaşılabilir. Fonksiyonların içinden küresel değişkenlere ulaşabilmek için global komutu ile tanımlama yapılması gerekmektedir. PHP fonksiyon parametrelerine varsayılan değer tanımlaması yapılabilmektedir. Ayrıca sınırsız sayıda parametre alan fonksiyon tanımlaması yapmakta mümkündür.

5 Sınıf tanımlaması yapabilme

Sınıflar

Programlama dillerinde sınıflar, kendi içinde tutarlılığı olan özel işleri modellemek için kullanılır. PHP programlama dili ile de sınıf tanımlaması yapmak mümkündür. Sınıflar sayesinde kodlar daha düzenli bir şekilde yazılır ve tekrar tekrar kullanılabilir. Sınıflar yazılan kodları daha anlamlı olmasını sağlar. Sınıf tanımlaması class komutu ile yapılmaktadır. Sınıf tanımlaması yapıldıktan sonra, sınıfın kullanılabilmesi için sınıfın bir örneği olan nesne tanımlaması yapılması gerekmektedir. Nesne tanımlama işi new komutu ile yapılmaktadır. Sınıfların üye değişkenleri, sabitleri ve fonksiyonları vardır. Ayrıca sınıf ismi ile aynı isimde kurucu fonksiyon tanımlaması yapılabilmektedir. Kurucu fonksiyonlar new komutu ile yeni bir nesne oluşturulurken çağrılan fonksiyonlardır.

6 Veritabanı işlemleri yapabilme

Veritabanı İşlemleri

PHP ile veritabanı sunucu yazılımlarına bağlanmak veritabanı üzerinde işlem yapmak mümkündür. Piyasada çok sayıda ücretli ve ücretsiz veritabanı sunucu yazılımı bulunmaktadır. PHP kullanıcıları genellikle MySQL veya MariaDB yazılımlarını tercih ettikleri söylenebilir. Veritabanları üzerinde işlem yapmak için SQL dili kullanılmaktadır. PHP ile veritabanına bağlantı yapıldıktan sonra SQL dili ile yazılmış sorgu ifadesi veritabanı üzerinde çalıştırılır ve bu sorgu sonucu elde edilen veriler PHP ile işlenebilir. MySQL veritabanına PHP ile bağlantı yapabilmek için mysqli gibi bir uzantının yüklenmiş olması gerekmektedir. mysqli uzantısı, hem yordamsal programlama hem de nesne yönelimli programlama paradigmalarını desteklemektedir.

7 Çatılar ve Laravel ile ilgili bilgi verebilme

Çatılar ve Laravel

Son yıllarda önemi giderek artan web uygulama geliştirme çatıları ile yazılım geliştirme süreci önemli derecede hızlanmaktadır. Çatılar, birçok uygulamada bulunan benzer öğelerin kullanılmalarına yönelik ortak bir tasarım ortaya koymaktadır. Genel olarak tüm uygulamalarda ortak olarak değerlendirilebilecek temel öğeler bulunmaktadır. PHP çatılarının çoğu MVC (Model-View-Controler, Model-Görünüm-Kontrolcü) tasarım desenini kullanmayı zorunlu kılmaktadırlar. MVC tasarım deseni, uygulamanın görünüm (sunum) katmanı, kontrolcü (business logic) katmanı ve model (veri yönetimi) katmanının nasıl birbirinden ayrılabileceği konusunda geliştiricilere yardımcı olur. Laravel çatısı da MVC desenini destekleyen bir PHP web uygulama geliştirme çatısıdır. Açık kaynak kodlu olan Laravel'in geliştirilmesine tüm dünyadan çok sayıda geliştirici destek vermektedir. Laravel dünyada yaygın kullanılan PHP çatılarından biridir.

neler öğrendik?

1 Aşağıdakilerden hangisi dizi oluşturmak için kullanılır?

A. Array
B. String
C. Count
D. { }
E. ->

2 Aşağıdakilerden hangisi temel türlerden biri **değildir?**

A. int
B. array
C. string
D. boolean
E. float

3 HTML formlarından hem get hem de post yöntemiyle gönderilen veriyi PHP›de almak için hangi dizi kullanılmalıdır?

A. $_GET
B. $_POST
C. $_REQUEST
D. $_GLOBAL
E. $_SESSION

4 if komutu ile yazılan karşılaştırma ifadesinin yanlış sonucunu üretmesi durumunda çalıştırılması istenen komutlar hangi blok içine yazılmalıdır?

A. if
B. for
C. while
D. do
E. else

5 $notu değişkeninin değerinin 0 ile 100 (bu değerler dahil) değerleri arasında olup olmadığını belirmek amacıyla aşağıdaki karşılaştırma ifadelerinden hangisi kullanılabilir?

A. $notu>=0 || $notu<=100
B. $notu<=0 || $notu>=100
C. 0<=$notu<=100
D. $notu<=100 && $notu>=0
E. 0<$notu<100

6 for($i=0; $i<=3; $i++){ echo «Merhaba Dünya
»; } döngüsü çalıştırıldığında «Merhaba Dünya» yazısı sayfaya kaç defa yazılır?

A. 3
B. 4
C. 2
D. 5
E. 0

7 Çalışan bir döngüyü sonlandırmak için hangi komut kullanılmalıdır?

A. Foreach
B. While
C. Break
D. Continue
E. Switch

8 Özellikle dizi elemanlarına yönelik olarak kullanılan döngü aşağıdakilerden hangisidir?

A. Foreach
B. For
C. Do while
D. While
E. Switch

9 Fonksiyondan geriye değer döndürmek için aşağıdaki komutlardan hangisi kullanılmalıdır?

A. Break
B. Continue
C. Void
D. Object
E. Return

10 Laravel çatısı hangi tasarım desenini kullanmayı zorunlu kılmaktadır?

A. Framework
B. View
C. Model
D. MVC
E. Controller

1. A	Yanıtınız yanlış ise "Değişkenler ve Değişken Türleri" konusunu yeniden gözden geçiriniz.
2. B	Yanıtınız yanlış ise "Değişkenler ve Değişken Türleri" konusunu yeniden gözden geçiriniz.
3. C	Yanıtınız yanlış ise "Html Formları" konusunu yeniden gözden geçiriniz.
4. E	Yanıtınız yanlış ise "Kontrol Yapıları" konusunu yeniden gözden geçiriniz.
5. D	Yanıtınız yanlış ise "Kontrol Yapıları" konusunu yeniden gözden geçiriniz.
6. B	Yanıtınız yanlış ise "Kontrol Yapıları" konusunu yeniden gözden geçiriniz.
7. C	Yanıtınız yanlış ise "Kontrol Yapıları" konusunu yeniden gözden geçiriniz.
8. A	Yanıtınız yanlış ise "Kontrol Yapıları" konusunu yeniden gözden geçiriniz.
9. E	Yanıtınız yanlış ise "Fonksiyonlar" konusunu yeniden gözden geçiriniz.
10. D	Yanıtınız yanlış ise "Çatılar ve Laravel" konusunu yeniden gözden geçiriniz.

3 Araştır Yanıt Anahtarı

Araştır 1

PHP'de birden fazla dizini birleştirmek amacıyla array_merge fonksiyonu kullanılır.

Araştır 2

HTML formunun PHP tarafından bağla.php dosyası ile işlenmesi için forma girilen verilerin action ile bağla.php dosyasına gönderilmesi gerekir. Bu verilerin arka planda gönderilmesi için de form etiketinin metod özelliği post olmalıdır. Böylece gerekli kod şu şekilde olmalıdır: `<formaction="bagla.php" method="post">)`

Araştır 3

$x>100 && $x<500 && $y>50 && $y<300 karşılaştırma ifadesi ile kontrol edilebilir.

Araştır 4

Bu problemi çözmenin çok sayıda yolu olabilir. Aşağıda bulunan kodlar bu çözüm yollarından biridir.

```
1   <?php
2   $onceki=1;$sonraki=1;
3   echo $onceki." - ".$sonraki;
4   for($i=1;$i<10;$i++){
5       $yeni = $onceki + $sonraki;
6       echo " - ".$yeni;
7       $onceki=$sonraki;
8       $sonraki=$yeni;
9   }
```

Araştır 5

Sık sık gerçekleştirilen bir işlem için fonksiyon yazılarak, aynı kodların defalarca tekrar yazılmasının önüne geçilmiş olur. Bu problemde iki değişkenin değerlerinin değişmesi gerekmektedir. Ancak fonksiyonlar sadece bir değer döndürebilirler. Dolayısıyla bu işlem geriye değer döndürme yöntemiyle çözülemez. Bu durumda referans olarak tanımlanmış iki parametre alan bir fonksiyon ile bu problemi çözmek mümkündür. Böylece parametre olan alınan değişkenleri değerleri değiştirildiğinde, fonksiyona parametre olarak gönderilen orijinal değişkenlerin değerleri de değişmiş olur. Aşağıda örnek kodlar yer almaktadır.

```
1   <?php
2   function degistir (&$a,&$b){
3     $gecici=$a;
4     $a=$b;
5     $b=$gecici;
6   }
```

3 Araştır Yanıt Anahtarı

Araştır 6

Belirtilen özelliklerdeki bir nesne ve veritabanına bağlantı kurmak için gerekli PHP kodu şöyledir: `$nesne1 = new mysqli("server","admin","admin123","veritabani1");)`

Araştır 7

Miras alma (Inheritance) daha önce tanımlı bir nesnenin tüm özelliklerinin farklı nesneler tarafından da kullanılabilmesini sağlayan bir nesne yönelimli programlama özelliğidir. Bu işlem yapıldığında miras alınan sınıfın tüm özellikleri yeni sınıfa aktarılır.

Kaynakça

Abeysinghe, S. (2009). *PHP team development : easy and effective team work using MVC, agile development, source control, testing, bug tracking, and more.* Olton Birmingham, GBR: Packt Publishing Ltd.

Balaban, E. (2013). PHP ve MySQL (3. Baskı). Ankara: Pusula Yayıncılık ve İletişim.

Çakır, S. (2015). PHP 5.4 / (*oku, izle, dinle, öğren*). İstanbul: Kodlab Yayın.

Çelik, R. (2015). *A' dan Z›ye PHP (6. Baskı).* Ankara: Seçkin Yayıncılık.

Gümüştepe, Y., Çelikbilek, İ., Baltalı, S., Çakır, S. ve Azak, H.İ. (2013). *Php eğitim seti (5 kitap).* İstanbul: Kodlab Yayın.

Koçak, M. (2014). *Hiç bilmeyenler için projelerle PHP sıfırdan web programlama.* Ankara: Seçkin Yayıncılık.

Koçak, M. (2016). *Projelerle PHP 7 sıfırdan web programlama PHP, OOP, MYSQL, PSD, HTML5, CSS3, JQUERY.* Ankara: Seçkin Yayıncılık.

Onur, M.Y. (2016). *Yeni başlayanlar için Laravel oku, izle, dinle, öğren.* İstanbul: Kodlab Yayın.

Uysal, M. ve Uysal, O. (2013). *PHP ile web tasarımı (2. Baskı).* Ankara: Nirvana Yayınları.

Winesett, J. (2010). *Agile web application development with Yii1.1 and PHP5 : Fast-track your web application development by harnessing the power of the Yii PHP framework.* Olton Birmingham, GBR: Packt Publishing Ltd.

Bölüm 4

JavaScript

öğrenme çıktıları

1 JavaScript'in İşlevleri

1 JavaScript programlama dilinin önemini açıklayabilme
2 JavaScript programlama dilinin işlevlerini sıralayabilme

2 Yazım Kuralları

3 JavaScript programlama dilinin temel yazım kurallarını açıklayabilme

3 Dom Nesnelerine Erişmek

4 JavaScript programlama dili ile HTML DOM'da element bulabilme

4 Çıktı Mekanizmaları

5 JavaScript programlama dilinin çıktı mekanizmalarını kullanabilme

5 Veri Tipleri

6 JavaScript programlama dilinde kullanılan veri tiplerini ayırt edebilme

6 Koşul Yapıları

7 JavaScript programlama dilindeki koşul yapılarını kullanabilme

7 Döngüler

8 JavaScript programlama dilindeki döngü yapılarını kurgulayabilme

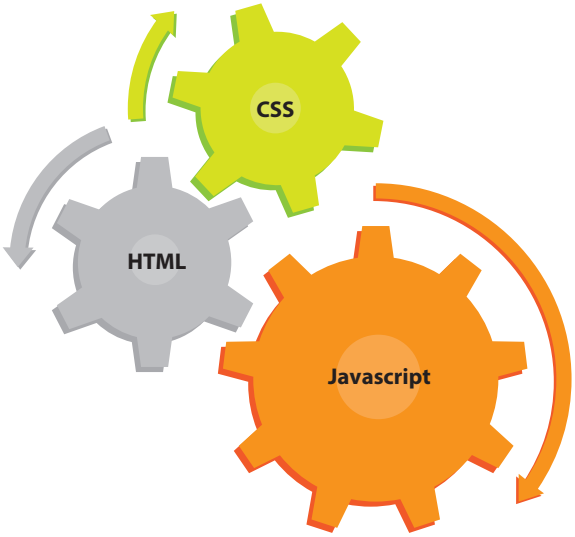
8 Fonksiyonlar

9 JavaScript fonksiyonu oluşturabilme
10 JavaScript ile zamanlanmış görev oluşturabilme

Anahtar Sözcükler: • JavaScript • HTML • Sentaks • Değişken • Koşul • Döngü • Fonksiyon • İşleç • DOM

GİRİŞ

JavaScript (JS) günümüz internet teknolojisinin temel taşlarından biridir (Resim 4.1). HTML ve CSS ile birlikte içerik üretim ve yönetim işlevlerini yerine getirmektedir. Günümüz modern internet tarayıcılarının (Ör: Google Chrome, Microsoft Edge) çoğu herhangi bir eklentiye gerek duymadan JS desteği sunmaktadır. Çoğunlukla internet sayfalarında kullanılmakla birlikte, JS dili masaüstü eklentileri (Ör: hesap makinesi, takvim araçları), PDF dokümanları ve sunucu taraflı uygulamalarda da çalıştırılabilir. Bu ünitede sunulacak konular tüm çalışma alanları için geçerli olmakla birlikte, internet sayfalarına yazılan örnekler tarayıcılar içinde denenecektir.



Resim 4.1 Günümüz internet teknolojilerinin temel bileşenleri

Modern internet teknolojileri düşünüldüğünde JS, HTML ve CSS ile sıkı bir etkileşim içindedir. Bir internet sayfasının içeriği HTML tarafından yapılandırılır. Sayfadaki yönlendirme elementleri (Ör: menüler ve bağlantılar), metinler, başlıklar ve özel alanlar HTML etiketleri kullanılarak işaretlenir. CSS bu yapının görünümünü düzenlemekle görevli bileşendir. Sayfanın artalanı, metinlerin gösterileceği yazı tipi, paragraf aralıkları, kutu kenarlıkları gibi görsel düzenlemeler CSS kullanılarak yapılır. JS ise sayfanın programlanmasını sağlayan mantıksal bileşendir. JS kullanılarak kullanıcıdan alınan veriler işlenerek sayfanın tepki vermesi sağlanabilir. Örneğin, kullanıcıya onay kutuları gösterilebilir, bir elementin içeriği ve nitelikleri değiştirilebilir ya da sayfanın bir bölümünün içeriği değiştirilebilir. Temel olarak JS'nin tarayıcıyı yönettiği/programladığı söylenebilir.

JS bir betik (script) dilidir. Betikler kendilerini yorumlayan ikincil araçlar üzerinde çalıştırılır. Örneğin sunucu tabanlı çalışan PHP dilinin betikleri PHP yorumlayıcı ile çalıştırılır. Benzer şekilde Flash teknolojisinin programlama dili olan Actionscript, Flash Player üzerinde çalıştırılır. C, Visual Basic gibi programlama dillerinin ürünleri ise kendi başlarına çalışabilecek şekilde derlenir. Bu derleme işlemleri sonucunda oluşturulan programlar (Ör: Windows ortamında EXE uzantılı dosyalar) işletim sistemi tarafından doğrudan çalıştırılırlar.

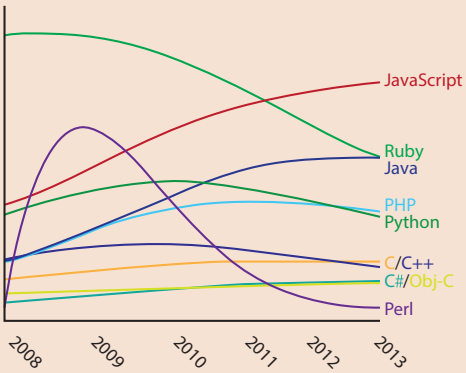
JS, Netscape Navigator isimli internet tarayıcısının üreticisi Netscape firması tarafından üretilmiştir. Başlarda Mocha ve Livescript adlarıyla anılan proje, 1995 yılının sonunda Netscape Navigator 2.0 beta sürümü ile birlikte JavaScript adı altında piyasaya sürülmüştür. Java diline çok yakın olan bu isimlendirme programlama dünyasında bir karmaşaya neden olmuştur. Bu dönemlerde gözlenen tarayıcı savaşları sürecinde Microsoft firması Internet Explorer tarayıcısında VBScript ve JScript gibi diller geliştirse de süreç sonunda JS bu alanda endüstri standardı haline gelmiştir. Netscape firmasının girişimleriyle ECMA International firması tarafından dilin standartları belirlenmiştir. JS 1999 yılında ilan edilen Ecmascript 3 şartnamesiyle uyumludur.

dikkat

İsimleri ve programlama yapıları benzerlik göstermesine rağmen, JS ve Java farklı dillerdir. Java uygulamaları Java Sanal Makineleri (Java Virtual Machine) üzerinde çeşitli platformlarda çalışabilir. JS kodlarıysa genellikle tarayıcılar (ör: Chrome, Safari, Edge) tarafından işletilir.

araştırmalarla ilişkilendir

GitHub vrilerine göre internet tabanlı programlama dillerinin kullanım eğilimlerinin yıllara göre değişimi aşağıdaki gibidir.



Kaynak: https://www.quora.com/Which-language-is-going-to-dominate-the-future-of-web-development-JavaScript-PHP-Ruby-Java-Scala-or-Python

JAVASCRİPT'İN İŞLEVLERİ

JS dilinin iki temel işlevi bulunmaktadır: internet sayfalarını düzenlemek ve tarayıcıları yönetmek. JS, HTML DOM (Document Object Model – Doküman Nesne Modeli) modelini kullanarak internet sayfalarının içeriklerine erişebilir. HTML DOM yapısı W3C (World Wide Web Consortium – Dünya İnternet Konsorsiyumu) tanımlanmış standart bir modeldir. Sayfa yüklendiğinde tarayıcı otomatik olarak sayfanın DOM modelini oluşturur. JS bu modele ulaşarak sayfadaki: elementleri, elementlerin niteliklerini ve CSS stillerini değiştirebilir, sayfaya yeni elementler ekleyebilir, sayfadan element silebilir ve sayfadaki elementlerin tetiklediği tüm olaylara tepki verebilir. Şimdi bu işlevleri örnekleyelim.

Bir HTML Elementinin İçeriğini Değiştirmek

JS sayfadaki bir elementin içeriğini düzenleyebilir. Web sayfamızda bir satın alma formu oluğunu düşünelim. Bu formun sonundaki bir paragraf elementinin içeriğini kullanıcının satın alma formundaki seçimlerine göre değiştirebiliriz. Örneğin 200TL fiyatlı bir ürün için, kullanıcının gireceği adet bilgisine göre toplam bakiyeyi hesaplayarak sayfaya yazabiliriz. Bu örnekte JS metin kutusunun değişim olayına tepki vererek, sayfadaki bir elementin içeriğini düzenlemektedir. Resim 4.2'de kullanıcının Ürün adedi kutusuna farklı değerler girmesi sonucunda sayfaya yazılan bakiye değerinin değiştiğine dikkat ediniz.

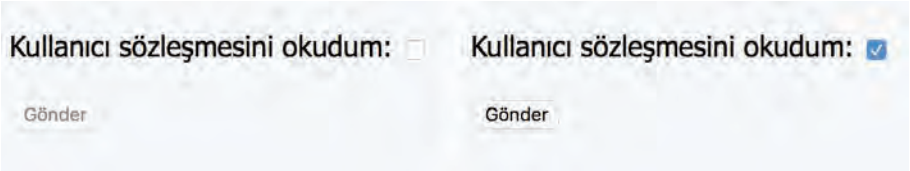
dikkat

Bu bölümdeki tüm örneklerin çalışan kodlarını "https://jsfiddle.net/user/odonmez/fiddles/" adresinde bulabilirsiniz.

Resim 4.2 Satın Alma Formunda Bakiye Bilgisinin Otomatik Olarak Hesaplanması

Bir Elementin Niteliklerini Değiştirmek

HTML elementleri kendilerine tanımlı nitelikler ile birlikte gelir. Bu nitelikler elementlerin açılış etiketlerine yazılarak elementlerin içeriklerini ve durumlarını belirler. Örneğin, tasarladığımız bir formun yalnızca kullanıcının sunulan sözleşmeyi kabul etmesi durumunda gönderilmesini sağlayabiliriz. Bu amaçla formu gönderen düğmenin aktif olup olmama durumunu niteliklerini değiştirerek düzenleyebiliriz. Düğmeleri oluşturabilen *input* elementinin aktifliği *disabled* niteliği ile ayarlanır. *disabled* niteliği *true* olan elementlerin etkileşimi kapatılır. JS ile kullanıcının sözleşmeyi okuduğunu beyan ettiği onay kutusunu dinleyerek, bu kutunun değişimine göre Gönder düğmesinin aktifliğini düzenleyebiliriz. Resim 4.3'te Gönder düğmesinin, kullanıcının, "Kullanıcı sözleşmesini okudum" onay kutusunu işaretlemesi durumunda aktifleştiğine dikkat ediniz.



Resim 4.3 Kullanıcının Sözleşmeyi Kabul Etmesiyle Gönder Düğmesinin Aktifleşmesi

Sayfalara Element Eklemek ve Silmek

JS kullanarak sayfalara yeni elementler ekleyebilir ya da silebiliriz. Örneğin, sayfanıdaki bir listeyi kullanıcının dolduracağı bir formdan alacağımız verilerle doldurabiliriz. Aynı şekilde bu listedeki bir elementin tıklanması halinde listeden silinmesini sağlayabiliriz. Resim 4.4'te Meyveler listesine formdaki giriş kutusundan alınan değerler eklenmektedir.

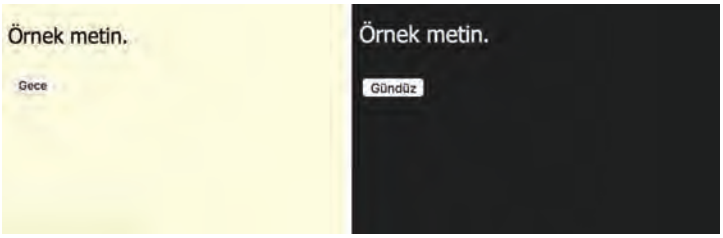
Resim 4.4 Bir Formdan Alınan Verileri HTML Listesine Eklemek

Bir Elementin Görünümünü Değiştirmek

JS sayfadaki bir elementin CSS stillerini değiştirebilir. Örneğin, sitemizin okunmasını kolaylaştırmak için gece ve gündüz saatlerinde iki renk temasıyla sunmak istediğinizi düşünelim. Gece saatlerinde çok parlak ekranlar gözleri yoracağından koyu bir artalan üzerinde parlak yazılar; gündüz saatlerinde de tam tersi bir tema kullanalım. Bu amaçla sitemize gece/gündüz temaları arasında geçiş yapmak için bir düğme ekleyebiliriz. Hatta, JS düğmeler yerine, bilgisayarın sistem saatini kontrol ederek bu geçişleri otomatik olarak da gerçekleştirebilir. Resim 4.5'te sayfa üzerindeki düğme ile koyu ve açık renk temaları arasındaki geçiş sağlanmaktadır. Bu işlemi bilgisayarın sistem saatine göre yapmak için Tarih nesnesi başlığındaki fonksiyonlar kullanılabilir.

JS DOM üzerinden tarayıcıdaki sayfaya erişebildiği gibi, BOM (Browser Object Model – Tarayıcı Nesne Modeli) üzerinden tarayıcı kaynaklarına da erişebilir. JS, BOM kullanarak: (1) yeni bir tarayıcı penceresi açabilir, (2) tarayıcı penceresini düzenleyebilir (ekrandaki yeri ya da boyutlarını değiştirebilir), (3) tarayıcı penceresini kapatabilir, (4) kullanıcının ekranı hakkında bilgi alabilir (boyutlar, renk derinliği vb.), (5) gösterilen

sayfa hakkında bilgi alabilir, (6) gösterilen sayfayı değiştirebilir, (7) tarayıcının tarih nesnesine erişerek önceki/sonraki sayfalara gidebilir, (8) kullanıcı bilgisayarına çerez bırakabilir, (9) zamanlanmış görevler oluşturabilir, (11) tarayıcı uygulaması hakkında bilgi alabilir, (12) tarayıcının uyarı ve alarm kutularını kullanabilir.



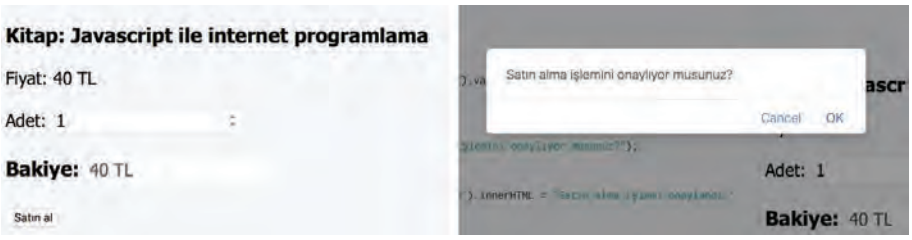
Resim 4.5 Sayfanın Metin Rengi ve Artalan Renginin Değiştirilmesi

Kullanıcıdan Onay Almak

JS diyalog kutularını kullanarak kullanıcıdan işlem onayı alabilir. Örneğin sayfamızdaki bir formdaki "Satın al" düğmesine tıklandığında kullanıcıya onaylama ya da vazgeçme şansı sunabiliriz. Kullanıcının onay vermesi halinde satın alma işlemleri için gerekli rutinleri işletebilir, aksi halde formu sıfırlayabiliriz. Resim 4.6'da sol tarafta örnek bir form, sağ tarafta ise örnek bir diyalog kutusu görülmektedir.

dikkat

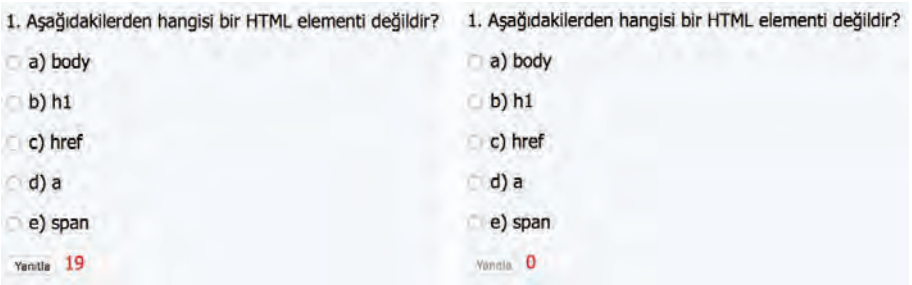
Resim 4.6'daki diyalog kutusu görüntüsü MacOS işletim sisteminde, Safari internet tarayıcısında alınmıştır. Diyalog kutularının görünümleri kullanılan işletim sistemi ve tarayıcıya göre farklılaşmaktadır.



Resim 4.6 Tarayıcı Diyalog Kutularını Kullanmak

Zamanlanmış Görev Kullanmak

JS zamanlayıcı (Timer) nesnesini kullanarak zamanlanmış görev oluşturabilir ya da bir olayın aralıklarla tekrarlanmasını sağlayabilir. Örneğin bir sınav uygulaması oluşturduğumuzu düşünelim. Bu uygulamada her sorunun yanıtlanması için kullanıcıya 30 saniye verilsin. 30 saniye sonunda kullanıcının soruyu yanıtlamasını engellemek için zamanlanmış görevleri kullanabiliriz. Saniyede bir kere çalışacak zamanlanmış görev ile sayfadaki bir sayaç güncellenerek, zamanın bitişinde formu gönderecek olan düğmenin etkileşimi kaldırılabilir. Resim 4.7'de solda zamanlanmış görevin çalışması ve sayacı güncellemesi görülmektedir. Sağ tarafta ise sürenin bitmesi durumunda formu gönderecek düğmenin aktifliği kaldırılmaktadır.



Resim 4.7 Zamanlanmış Görev Kullanmak

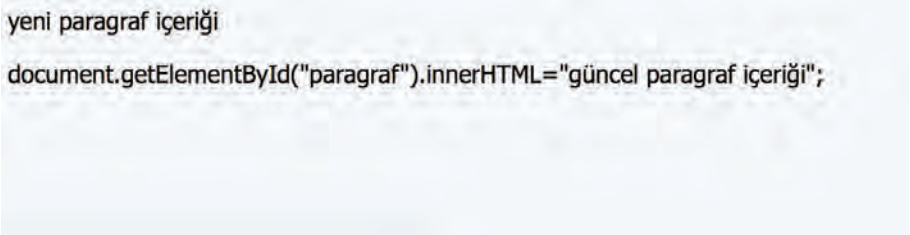
Bu bölümde JS dilinin yetenekleri örnekler ile açıklanmıştır. Yer kısıtları nedeniyle tüm yeteneklerine ilişkin örnekler verilememiştir. Özellikle tarayıcı programlama konusundaki örneklere internette kolaylıkla ulaşılabilirsiniz

JavaScript Nereye Yazılır?

JS kodları HTML dokümanlarında head ya da body bölümlerine yazılabilir. HTML bir işaretleme dili olduğundan, metin halindeki JS kodlarının da işaretlenmesi gerekmektedir. Aksi halde yazılan kodlar programlama kodu olarak yorumlanmak yerine sayfada gösterilecektir. Bu amaçla JS kodları script etiketleri arasına yerleştirilir.

```
1   <p id="paragraf">eski paragraf içeriği</p>
2   <script>
3   document.getElementById("paragraf").innerHTML="yeni paragraf içeriği";
4   <script>
5   document.getElementById("paragraf").innerHTML="güncel paragraf içeriği";
```

Yukarıda verilen kodların Resim 4.8'de etkisi görülmektedir. Bu örnekte sayfadaki bir paragrafın içeriği değiştirilmektedir. Başlangıçta bu paragrafın içinde "eski paragraf içeriği" ifadesi yer almaktadır (1. satır). Bunun ardından *script* etiketleri ile çevrelenen JS kodu ile paragrafın içeriği "yeni paragraf içeriği" olarak değiştirilmektedir (2-3-4. satırlar). Beşinci satırdaki JS kodu paragraf içeriğini "güncel paragraf içeriği" olarak düzenlemeye çalışmaktadır. Çıktı ekranında görüldüğü üzere, paragrafın metni "yeni paragraf içeriği" olarak kalmıştır. Bunun nedeni *script* etiketleri arasındaki JS kodunun (Satır 3) yaptığı düzenlemedir. Fakat 5. satırdaki JS kodları *script* etiketleri içinde yer almadığından sayfa metni olarak yorumlanarak sayfada gösterilmiştir.



Resim 4.8 Script Etiketlerinin Kullanımı

Bir HTML sayfasında istenildiği kadar *script* etiketi kullanılabilir. Sayfanın herhangi bir yerinde *script* etiketleri arasında JS kodu yazılabilir. Bununla birlikte, sayfadaki JS kodlarını tek bir yerde tutmak önerilen bir programlama yaklaşımıdır. Sayfadaki JS kodlarının farklı bölümlerde olması düzenleme ve yönetimi zorlaştırabilir.

> **dikkat**
> HTML dilinin eski versiyonlarında script etiketlerinin type niteliği kullanılmaktaydı. Bu versiyonlarda script etiketleri <script type="text/js"> şeklinde açılmaktaydı. Günümüzde HTML'nin varsayılan programlama dili JS olduğundan type niteliğini kullanmaya gerek yoktur.

HTML sayfaları ilk satırdan itibaren doğrusal olarak işletilir. Bu çerçevede, tarayıcı karşılaştığı JS kodlarını anında yorumlayacaktır. Bu durum, DOM'a erişmeye çalışan JS kodları için önemli bir sorun oluşturabilir. Düzenlenmek istenen elementin üstündeki satırlarda yazılan JS kodları, bu elementler henüz oluşturulmadığından hata verecektir. Bu nedenle özellikle DOM işlemleri içeren kod satırlarının body bölümünün sonunda yazılması önerilmektedir. Bunun yanında, JS kodlarının body bölümünün sonuna eklenmesi sayfa performansını da olumlu etkiler. Satır sayısı yüksek JS kodlarının tarayıcı tarafından yorumlanması ve işletilmesi zaman alacağından, sayfanın gösterilmesini geciktirme ihtimali bulunmaktadır.

JS kodlarının fonksiyonlar (function) içine alınması bu duruma istisna oluşturmaktadır. Fonksiyonlar çağırıldığında çalışan JS kodu bloklarıdır. Fonksiyonlar dışındaki JS kodları anında yürütülürken, fonksiyonların yürütülmesi için kod içinden çağırılmaları ya da bağlandıkları bir olayın

> ✓ Fonksiyonlar çağırıldığında çalışan JS kodu bloklarıdır. Fonksiyonlar sayfadaki olaylara bağlanarak çalıştırılabilirler.

gerçekleşmesi gerekmektedir. Örneğin, *head* ya da *body* bölümünün başında yazılan JS kodları fonksiyon içinde ise bunlar otomatik olarak çalıştırılmazlar. Bu fonksiyonlar DOM'daki bir elementin tetiklediği bir olay ile çalıştırılabilir. Örneğin, bir fonksiyon sayfa üzerindeki bir düğmenin tıklanması sonucu çalışacak şekilde ayarlanabilir. Bu amaçla fonksiyonun adı, elementin onclick niteliğine tırnaklar arasında yazılır. Aşağıda verilen koddaki *tiklandi* fonksiyonu sayfadaki bir düğmenin tıklama (onclick) olayına bağlanmıştır. Kullanıcı bu düğmeye her bastığında tiklandi fonksiyonu bir kez çalıştırılır. Fonksiyon adının arkasındaki parantezlere ve bağlama kodlarının tırnaklar içinde yazıldığına dikkat ediniz.

```
<input type="button" onclick="tiklandi()" value="Tıklayınız"
```

JS kodları sayfa dışında .js uzantılı bir harici dosyaya yazılabilir. Harici dosyalar, sayfaya script etiketleri ile bağlanır. Aşağıda HTML dosyası ile aynı dizindeki "harici.js" isimli dosya sayfaya bağlanmaktadır.

```
<script src="harici.js"></script>
```

Yukarıdaki kodda JS dosyası sayfa ile aynı dizinde olduğundan dolayı doğrudan dosya adı kullanılarak bağlanmıştır. Bunun yanında, farklı dizinler ya da sunuculardaki JS dosyaları sayfalara eklenebilir. Farklı bir sunucudaki JS dosyasını sayfaya eklemek için URL adresi girilmelidir. Örneğin, en çok kullanılan JS kütüphanelerinden biri olan JQuery'i sayfamıza eklemek için aşağıdaki kod satırını kullanabiliriz. Bu işlemden sonra sayfamızda script etiketleri arasında JQuery fonksiyonlarını kullanabiliriz.

dikkat

Harici dosyaların içinde script etiketleri kullanılmaz. JS kodları doğrudan yazılır.

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
```

Sunucumuzda HTML sayfamızdan farklı bir dizindeki bir JS dosyasını sayfaya bağlamak için göreli (relative path) dosya yolu kullanılır. Aşağıda, HTML sayfasının bulunduğu dizindeki js dizininde yer alan "harici.js" sayfaya bağlanmaktadır.

```
<script src="js/harici.js"></script>
```

JS kodlarını sayfa dışında tutmanın pek çok avantajı vardır. Öncelikle bu yaklaşım yazılan JS kodlarının birden fazla sayfada kullanılabilmesine olanak verir. Örneğin sayfalardaki form verilerini kontrol eden bir fonksiyon, form içeren tüm sayfalarda gereklidir. Bu fonksiyon bir JS dosyasına yazılarak form içeren tüm sayfalarda kullanılabilir. Bu yaklaşım sayfanın yüklenme performansını da arttırabilir. Tarayıcılar JS dosyalarını tampon belleklerinde tutabilmektedir. Bu sayede, aynı harici JS dosyasını çağıran ikinci sayfa için JS dosyanız sunucudan indirilmek yerine tarayıcının tampon belleğinden kullanılabilir. Bu durum sayfanızın yüklenme performansını arttıracağı gibi, sunucunuzun veri transferini de azaltacaktır. Son olarak harici JS dosyaları kullanarak HTML ve JS kodlarını farklı sayfalarda tutmak kodlarınızın yönetimini kolaylaştıracaktır. Tüm JS kodlarınızın bir noktada yer alması okuma ve düzenlemeyi kolaylaştıracaktır. Sayfa içine birden fazla script dosyası yazabileceğiniz gibi, birden fazla harici JS dosyasını da sayfanıza bağlayabilirsiniz. Tabii ki, harici JS dosyaları ve sayfa içi kodlamaları birlikte kullanabilirsiniz.

Öğrenme Çıktısı

1 JavaScript programlama dilinin önemini açıklayabilme
2 JavaScript programlama dilinin işlevlerini sıralayabilme

Araştır 1

Yeni bir sayfa üzerinde id niteliği p1 olan boş bir paragraf oluşturunuz. Bu paragrafa JS ile erişerek paragrafın içeriğini "Bu paragraf JS ile düzenlendi" şeklinde değiştiriniz.

İlişkilendir

JavaScript ile yapılabilecek dinamik işlevlerle PHP ile yapılabilecek dinamik işlevleri ilişkilendiriniz.

Anlat/Paylaş

JavaScript kod yapısının öğrendiğiniz yetenekleri ile Web sitelerinde yapılabilecek uygulamaları paylaşınız.

YAZIM KURALLARI (SENTAKS)

JavaScript internet sayfalarını programlamak için kullanılır. Programlar, art arda işletilmesi gereken yönergeler (statement) halinde yazılır. Günlük yazım işlerimizde cümlelerimizi ayırdığımız gibi, programlama dilleri yönergelerini de birbirinden ayırmamız gerekir. JS dilinde yönergeler noktalı virgül kullanılarak ayrılır. Aşağıdaki kodda ikinci ve üçüncü satırlarda üç adet yönerge kullanılmıştır.

```
1   <script>
2   var isim = "Onur";
3   var sayi = 6; var sonuc = 2 + 2;
4   <script>
```

Bu satırların sonunda yönergeyi sonlandırmak için noktalı virgül kullanıldığına dikkat ediniz. Noktalı virgüllerle ayrıldıkları sürece yönergelerin aynı satırda ya da ayrı satırlarda olmasının bir önemi yoktur. Fakat her yönergeyi bir satırda yazmak okumayı kolaylaştıracak bir yazım pratiğidir. JS yönergeleri değerler (value), değişkenler (variable), işleçler (operator), ifadeler (expression), anahtar sözcükler (keyword) ve yorumlardan (comment) oluşur. Şimdi bu parçaları tek tek ele alalım.

Değerler

JS değerleri sayısal ya da metin şeklinde olabilir. Sayısal değerler doğrudan sayılar halinde yazılabilir. JS dilinde ondalık işareti "." noktadır. Aşağıdaki kodda ikinci satırda bir tam sayı, üçüncü satırda ondalıklı bir sayı tanımlanmaktadır. Metinler ise dördüncü satırda görüldüğü gibi tırnaklar içinde yazılır. Metinleri belirtmek için tek tırnak ('metin') ya da çift ("metin") tırnak çiftleri kullanılır.

```
1   <script>
2   var sayi1 = 10;
3   var sayi2 = 11.2;
4   var isim="Onur";
5   <script>
```

dikkat

JS tırnaklar içine yazılmış tüm değerlere metin olarak yaklaşmaktadır. Bu nedenle "11.1" ifadesi JS için bir metindir. Bu ifade üzerinde toplama ya da çarpma gibi bir sayısal işlem gerçekleştirilemez. Tırnaklar içine almadığınız metinler yazım hatalarına (Syntax Error) yol açar. JS bu değerlerin değişken ismi ya da tanımlı bir anahtar sözcük olmadığını belirten hata mesajları gösterecektir.

Tırnaklar içine alınmayan metinler yazım hatalarına yol açabilir. Bu metinler bir anahtar sözcük ya da değişken ismi değilse tarayıcı penceresinin konsolunda yazım hatası mesajı görülecektir (Resim 4.9). Bu durum, bu yönergenin ardında gelen kodların işletilmesini engeller. Aşağıdaki kodda dördüncü satırdaki Onur Dönmez ifadesi JS dilinde tanımlı bir ifade değildir. Bu ifadenin metin olarak ele alınabilmesi için tırnaklar içinde yazılması zorunludur.

```
1   <script>
2   function tiklandi(){
3   document.getElementById("paragraf").innerHTML = "düğmeye tıklandı";
4   var adSoyad = Onur Dönmez;
5   }
6   </script>
```

Resim 4.9 Tarayıcı Konsolunda Yazım Hatası Mesajı Örneği

Değişkenler

Programlama dillerinin en temel yapılarından biri değişkenlerdir. Değişkenler, programın çalıştırılması sırasında farklı değerleri bilgisayar hafızasında tutabilmek için kullanılır. JS değişkenleri var anahtar sözcüğü ile tanımlanır. Değişkenlerin içine değer göndermek için eşittir "=" işleci kullanılır. Aşağıdaki kodda ikinci satırda isim adlı bir değişken tanımlanmaktadır.

```
1   <script>
2     var isim;
3     isim = "onur";
4     alert(isim);
5     var sayi = 56;
6     var sayi2;
7     alert(sayi2);
8     sayi2 = sayi;
9     alert(sayi2);
10  </script>
```

Üçüncü satırda metin bir değer olan "onur", isim değişkenine gönderilmektedir. Bundan sonra "onur" değerine ulaşmak için program kodunda değişkenin adı olan isim yazılması yeterli olacaktır. Dördüncü satırda *alert* metodu kullanılarak isim değişkeninin içeriği uyarı penceresinde gösterilmektedir. Beşinci satırda *sayi* isimli yeni bir değişken tanımlanarak, bu değişkene 56 değeri atanmaktadır. Değişkenlere tanımlama satırlarında değer atanabilmektedir. Değer ataması yapılmayan değişkenlerle işlem yapılmaya çalışıldığında JS varsayılan olarak *undefined* (tanımlanmamış) ifadesini kullanır. Örneğin altıncı satırda tanımlanan *sayi2* değişkenine herhangi bir değer ataması yapılmamıştır. Bu nedenle yedinci satırdan çağırılan ikinci uyarı penceresinde *undefined* değeri gösterilecektir. Sekizinci satırda *sayi2* değişkenine, *sayi* değişkeninin değeri atanmaktadır. Bu nedenle dokuzuncu satırdan çağırılan uyarı penceresi 56 değerini gösterecektir. JS kullanırken oluşturacağınız değişkenler ve fonksiyonların isimlendirilmesi konusunda uymanız gereken kurallar bulunmaktadır. Bu kurallara uymamanız durumunda JS hata verecektir. Bu kurallar aşağıda sıralanmıştır.

- Değişken isimlerinde boşluk kullanılamaz.
- Değişken isimlerinde Türkçe karakterler (ü,ğ,i,ş,ç,ö) kullanılamaz.
- Değişken isimlerinde büyük ve küçük harfler kullanılabilir [A-Z, a-z].
- Değişken isimlerinde, ilk karakter olmamak şartıyla sayılar kullanılabilir [0-9].
- Değişken isimlerinde alt tire "_" ve "$" dolar işareti kullanılabilir.
- _ ve $ işaretlerinin ilk ve son karakter olarak kullanılması önerilmez.
- Değişken isimleri JS rezerve sözcüklerinden (for, var, function vb.) biri olamaz.
- İsimlendirmeler büyük küçük harf duyarlıdır. (sayi ve Sayi ayrı değişkenlerdir).

JS dilinde değişken tanımlamalarında veri tipi kullanılmaz. Buna karşın C# gibi üst düzey programlama dilleri değişken tanımlamasında veri tipi kullanımını zorunlu tutar. JS dilinde veri tipi kullanılmamasının nedeni değişkenlerin veri tiplerinin dinamik olmasıdır. Bu sayede tanımlı bir değişkene faklı veri tiplerinde değerler gönderilebilir. Aşağıda verilen kodun ikinci satırında x değişkeni tanımlanarak 5 değeri atanmaktadır.

```
1   <script>
2     var x=5; // x değişkeninde sayısal değer tutuluyor
3     x="onur"; // x değişkeninde metin değer tutuluyor
4   <script>
```

Üçüncü satırda, sayısal değer tutmakta olan x değişkenine metin olan "onur" değeri atanmaktadır. Değişken veri tiplerinin dinamik olması sayesinde bu işlemler hata vermeden gerçekleştirilmektedir. Bununla birlikte, bu durum JS dilinde hata ayıklama işlemlerini zorlaştırmaktadır.

İşleçler

JS dilinde değerler ve değişkenler arasındaki işlemler işleçler yoluyla gerçekleştirilir. JS işleçleri aritmetik işleçler, atama işleçleri, metin işleci ve kıyaslama ve mantıksal işleçler şeklinde sınıflanabilir.

Aritmetik İşleçler

Değerler arasında işlem yapmak için aritmetik işleçler kullanılabilir. Bu işleçler, uygulama biçimleri ve açıklamaları Tablo 4.1'de verilmiştir.

Tablo 4.1 Değerler Arası Aritmetik İşleçlerin Etkileri

İşleç	Açıklama	Örnek	Sonuç
+	Verilen değerleri toplar.	10+5	15
-	İlk değerden ikincisi çıkarır	10-5	5
*	Verilen değerleri çarpar	10*5	50
/	İlk değeri ikinciye böler	10/5	2
%	İkinci değere göre ilk değerin modunu alır	10%3	1

Aritmetik işleçlerin birbirlerine göre üstünlükleri olduğunu unutmayınız. Yazılan bir ifadede çarpma, bölme ve mod alma grubu işleçleri, toplama ve çıkarma grubu işleçlerinden önce yorumlanır. Bu işleme yöntemi parantez işleçleri kullanılarak düzenlenebilir. Parantez işleçlerinin içerikleri, aritmetik işleçlere göre daha önce yorumlanacaktır. Tablo 4.2'deki örnekleri inceleyiniz.

JS yönergelerdeki boşlukları görmezden gelir. Kodlarınızı daha okunur kılmak için boşluk karakterleri kullanabilirsiniz. İşleçlerin önlerine ve arkasına boşluk bırakmanız önerilen bir programlama yaklaşımıdır.

Tablo 4.2 Aritmetik İşleçlerde İşlem Önceliği

İfade	Sonuç
10*5+2	52
10*5+2*10	70
10+10/2	15
1+10%3	2
10*(10+2)	120
(12-2)*5+5	55

Atama İşleçleri

Atama işleçleri değişkenlere veri göndermek için kullanılır (Tablo 4.3). Bu işleçlerin kullanılabilmesi için ifadenin sol tarafında bir değişken bulunması gerekir. Tablo 4.3'teki yönergelerin işletilmesi sonucunda sayi değişkeninin değeri Sonuç sütunundaki değer olacaktır. Aşağıdaki örnekleri değerlendirirken sayi değişkeninin giriş değerini 10 kabul ediniz.

Tablo 4.3 Değişkenlere Uygulanabilecek Atama İşleçleri (sayi=10)

İşleç	Açıklama	Örnek	Açılımı	Sonuç
=	Değişkene değeri atar	sayi=10	sayi=10	10
++	Değişken değerine 1 ekler	sayi++	sayi=sayi+1;	11
--	Değişken değerinden 1 çıkarır	sayi--	sayi=sayi-1;	9
+=	Değişken değerine verilen sayıyı ekler	sayi+=5	sayi=sayi+5;	15
-=	Değişken değerinden verilen sayıyı çıkarır	sayi-=5	sayi=sayi-5;	5
*=	Değişken değerini verilen sayıyla çarparak sonucu değişkene gönderir	sayi*=5	sayi=sayi*5;	50
/=	Değişken değerini verilen sayıya bölerek sonucu değişkene gönderir	sayi/=5	sayi=sayi/5;	2
%=	İkinci değere göre değişken değerinin modunu alarak sonucu değişkene gönderir	sayi%=3	sayi=sayi%3;	1

dikkat

Tablo 4.3'teki ifadeler size anlamsız geliyor olabilir. sayi=sayi+1 ifadesi matematiksel olarak yanlıştır. Bununla birlikte programlama dillerinde kullanılabilecek bir ifadedir. Atama işleci bu ifadeyi ikiye bölmektedir. Önce işlecin sağ tarafı yorumlanarak işlem gerçekleştirilir. Buradan elde edilen sonuç sayi değişkenine atanır.

Metin İşleci

+ işleci metinlerin birleştirilmesi için de kullanılır. Arasına + işleci konan metinler birleştirilir. Aşağıda verilen kod örneğinde ikinci satırda adsoyad değişkeninin içeriği "OnurDönmez" olarak belirlenmiştir. İki metin ifade arasında boşluk karakteri bırakılmadığına dikkat ediniz. Metinleri birleştirirken değerler arasında boşluk bırakmak için araya " " değeri yazılabilir.

```
1   <script>
2     var adsoyad = "Onur" + "Dönmez";
3     var adsoyad = "Onur" + " " + "Dönmez";
4   </script>
```

Üçüncü satırdaki *adsoyad* değişkeninin içeriği "Onur Dönmez" olarak düzenlenmiştir. Bu işlem Onur ifadesinin arkasında ("Onur ") ya da Dönmez ifadesinin önünde (" Dönmez") bir boşluk bırakılarak da gerçekleştirilebilirdi. Bir metin ve sayının birleştirilmesi de mümkündür. Bu durumda sayısal değer otomatik olarak metne çevrilerek metin toplama yapılır. Böyle bir işlemin sonucu her zaman metin olacaktır. Aşağıda verilen kodda ikinci satırda ali metni ile 10 sayısı toplanmaktadır.

```
1   <script>
2     var metin = "ali" + 10;
3     var metin = "10" + 2;
4   </script>
```

Bu işlemin sonucu "ali10" şeklinde olur. Üçüncü satırda ise "10" metni (tırnaklar içinde yazıldığına dikkat ediniz) ile 2 sayısı toplanmaktadır. Bu işlemin sonucu metin halinde "102" olacaktır.

dikkat

JS otomatik tip dönüşümü yapabilmektedir. Bir metin ve sayının toplanması işleminde sayısal değer otomatik olarak metne dönüştürülür. Aynı şekilde sayılardan oluşan "102" gibi bir metin çarpma işlemine tabi tutulursa ("102"*2), değer sayıya dönüştürülerek 204 sonucu çıkarılır. Bununla birlikte, "1a2" gibi hem sayı hem de karakter içeren bir değerde otomatik tip dönüşümü uygulandığında çıktı NaN olacaktır (Not a Number – Sayısal Olmayan Değer). "1a2"*2 gibi bir işlemin sonucu NaN'dır.

dikkat

Atama işleci (=) kıyaslama işlemleri için kullanılamaz. Bu ifadelerin solunda değişken bulunması durumunda sonuç her zaman true dönecektir. İfadenin solunda değişken olmaması durumunda ise JS hata verecektir.

Kıyaslama İşleçleri ve Mantıksal İşleçler

Mantıksal işleçler verilen ifadenin doğruluğunu sınar (Tablo 4.4). Bu ifadelerin sonuçları *true* (doğru) ya da *false* (yanlış) olur. Kıyaslama ve mantıksal işleçler genellikle karar yapıları ile birlikte kullanılır. Bu yapıların içinde kıyaslama ifadelerinin kullanımına ilgili konularda tekrar değineceğiz.

Tablo 4.4 JS Dilinde Kullanılan Kıyaslama İşleçleri

İşleç	Açıklama	Örnek	Sonuç
==	Değer Eşittir	"2"==2	true
===	Değer ve tür eşittir	"2"===2	false
!=	Eşit değildir	2!=3	true
!==	Değer veya tür eşit değildir	"2"!==2	true
>	Büyüktür	2>3	false
<	Küçüktür	3<3	false
>=	Büyük ya da eşittir	3>=3	true
<=	Küçük ya da eşittir	2<=3	true

Mantıksal işleçler kıyaslama ifadelerinin sonuçları üzerinde işlem yapmak için kullanılır. JS dili üç mantıksal işleç sağlamaktadır. Bunlar 4.5'te verilmiştir.

Tablo 4.5 JS dilinde Sağlanan Mantıksal İşleçler

İşleç	Açıklama	Örnek	Sonuç
!	TERS (NOT) işleci Verilen ifadeyi tersine çevirir	!true !(2==2) !(2>3)	false false true
&&	VE (AND) işleci verilen ifadelerin ikisinin de doğru olması halinde true değeri çıkarır. Aksi halde false çevirir.	true && true 5==5 && 3>2 5==5 && 2>3	true true false
||	VEYA (OR) işleci erilen ifadelerden birinin doğru olması halinde true değeri çıkarır. Aksi halde false çevirir.	true || false 5==5 || 2>3 5>6 || 2==3	true true false

Mantıksal işleçler kıyaslama ifadelerinin bağlanması için kullanılabilir. Örneğin sitemizde kullanıcı kaydı yaparken iki ön şartımız olduğunu düşünelim. İlk olarak, kullanıcının 13 yaş ve üzerinde olması gerekmektedir. Ayrıca kayıt yapabilmek için kullanıcının kullanım şartlarını kabul etmesi gerekmektedir. Bu iki şartın aynı anda sağlanıp sağlanmadığını kontrol etmek için VE işleci kullanılabilir. Mantıksal işleçler genellikle program kodlarının akışını düzenleyen koşul yapıları ile birlikte kullanılır. Aşağıda verilen kodun ikinci satırda ziyaretçinin yaşı 13 olarak belirlenmiştir.

```
1   <script>
2     yas=13;
3     kabul=false;
4     if(yas>12 && kabul)
5     {
6       // Kayıt rutinlerini çalıştır.
7     }
8   </script>
```

Üçüncü satırda ise kullanıcının katılım sözleşmesini kabul etmediği (false) görülmektedir. Örneğimizde bu değişken, kullanıcının formdaki onay kutusunu işaretlemesi durumunda başka bir fonksiyon tarafından düzenlendiği varsayılmaktadır. 4. satırdaki *if* koşul yapısında, yaş ve kabul etme durumları VE işleci ile bağlanmıştır. Bunun anlamı her iki koşulun da sağlanması gerektiğidir. Bu durumda *yas* 12'den büyük olduğu için ifadenin sol tarafı *true* değerini çevirmektedir. Bununla birlikte, sağ tarafta kabul değişkeninin değeri *false* olduğundan, şartlardan sadece biri sağlanmaktadır. Bu nedenle, bu kullanıcının kaydı henüz kullanım şartlarını kabul etmediğinden gerçekleştirilmeyecektir.

İfadeler

İfadeler değerler, değişkenler ve işleçlerin birlikte kullanılmasıyla oluşturulur. Bu birleşimin sonucu her zaman bir değer olur. Bu işleme ifadenin yorumlanması denir. Aşağıdaki örnekte, ikinci satır iki değer ve bir işleçten oluşan bir ifadedir.

```
1   <script>
2     5*10;
3     var sayi=5;
4     sayi-3;
5   </script>
```

Bu ifadenin sonucu 50 değeridir. Benzer şekilde dördüncü satır da bir değişken, bir işleç ve bir değerden oluşan bir ifadedir. Üçüncü satırda sayi değişkeninin değeri 5 olarak belirlendiğinden bu ifadenin sonucu 2 olacaktır.

Anahtar Sözcükler

JS dilinde anahtar sözcükler gerçekleştirilecek eylemleri belirtir. Örneğin var anahtar sözcüğü tarayıcıya yeni bir değişken yaratmasını söyler. Aşağıdaki kodda ikinci satırda var anahtar sözcüğü kullanılarak isim adında bir değişken tanımlanmaktadır.

```
1   <script>
2     var isim="Onur";
3     alert(isim);
4   </script>
```

JS dili bunların dışında pek çok anahtar sözcük içermektedir. Ünitede ilerledikçe benzer anahtar sözcüklerle karşılaşacaksınız. Bu anahtar sözcükleri değişken ismi olarak kullanmamaya dikkat ediniz.

JS dilindeki anahtar sözcüklerin tam listesi için http://www.w3schools.com/js/js_reserved.asp adresini ziyaret ediniz.

Yorumlar

Yazılan bir programla ilgili not almak programcıların sıklıkla kullandığı bir yöntemdir. Bu amaçla program kodu içinde yorum satırları kullanılır. Yorum satırlarının kullanılmasını gerektiren üç senaryodan bahsedelim. Yazdığımız bir programı iki yıl sonra tekrar ele almak zorunda kalabiliriz. Bu durumda iki yıl önce yazmış olduğunuz kodları anlamamız gerekecektir. Programı yazarken ekleyeceğimiz yorum satırları uzun bir süre sonra koda tekrar döndüğümüzde işimizi kolaylaştıracaktır. Bunun yanında, çoğunlukla bir program tek bir kişi tarafından oluşturulmaz. Çalışma arkadaşlarımızın ya da diğer programcıların yazdığımız kodları anlamalarına yardımcı olmak için, onlarla haberleşmek için kodlarınızın arasına yorum satırları ekleyebiliriz. Yorum satırlarının diğer bir kullanım amacı da işletilmesini istemediğimiz program kodlarını tarayıcıdan gizlemektir. Bu kod satırlarını yorum satırına çevirerek silmekten kurtulabiliriz.

```
1   <script>
2     var alan=Math.PI*2*2;
3     //başlangıçta dairenin yarıçapını 2 kabul ediyorum
4   /* Daha önce Pi'yi 3.14 kabul etmiştim. Math.PI sabiti daha güvenilir
5     sonuçlar verdiğinden bu satırı üstteki satırlar değiştirdim
6     var alan=3.14*2 */
7   </script>
```

JS iki tür yorum satırı oluşturmaya izin verir. Tek satırlık yorumlar // işaretleri ile başlatılır. Yukarıda verilen kodun üçüncü satırında tek satırlık bir yorum ifadesi görülmektedir. Bu işaretin ardında kalan karakterler yorum olarak kabul edilir. Birden fazla satıra yayılan yorumların başlangıçları /* ile, bitişleri ise */ ile işaretlenir. Dördüncü ve altıncı satırlar arasındaki yorumlar bu şekilde işaretlenmiştir.

Öğrenme Çıktısı

3 JavaScript programlama dilinin temel yazım kurallarını açıklayabilme

Araştır 2

Bir öğrencinin notunun sonuc isimli değişkende tutulduğunu varsayarak; sonucun değerinin 60'a eşit ya da 60'tan büyük ve 90'dan küçük olma durumunu kontrol eden mantıksal sınama ifadesini yazınız.

İlişkilendir

JavaScript sentax yapısını dildeki gramer ile ilişkilendiriniz.

Anlat/Paylaş

Anlamakta zorlandığınız JavaScript yazım kurallarını paylaşınız.

DOM NESNELERİNE ERİŞMEK

JS dilinin en önemli işlevlerinden biri sayfadaki elementler üzerinde işlem yapmaktır. Bu işlemi yapabilmek için JS kodunun sayfadaki DOM elementine erişmesi gerekmektedir. Sayfamızda 10 adet paragraf oluğunu ve bunlardan birinin içeriğini değiştirmek istediğinizi düşünelim. Bunu gerçekleştirmek için hedeflediğiniz paragraf elementini bulmanız gerekmektedir. Elementler, sayfadaki DOM'u belirten *document* nesnesinin sağladığı metotlarla bulunabilir. *document* nesnesi elementleri bulmak için üç adet metot sağlar. Bunlar; *document.getElementById, document.getElementsByClassName ve document.getElementsByTagName* metotlardır.

document.getElementById()

id, HTML etiketlerinin en önemli niteliklerinden biridir. Her HTML elementi kendine atanmış bir *id* niteliğine sahip olabilir. Sayfalar oluşturulurken id niteliklerinin çakışmaması sağlanırsa elementler bu nitelikleri yoluyla kolaylıkla bulunabilir. Aşağıdaki kodda, ilk satırda tanımlanan *ul* elementinin *id* niteliği meyveler olarak belirlenmiştir. Üçüncü satırda bu elementi bulmak için *getElementById* fonksiyonu kullanılmıştır.

```
1   <ul id="meyveler"></ul>
2   <script>
3     var liste=document.getElementById("meyveler");
4   </script>
```

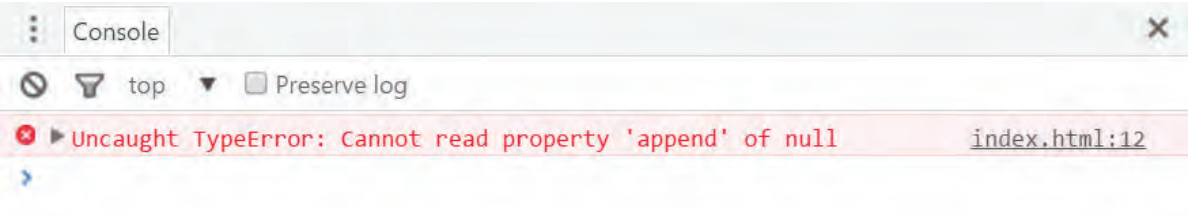
Erişmek istedikleri elementin üzerine yazılan DOM metotları, bu elementler henüz oluşturularak DOM'a eklenmediğinden düzgün çalışamaz. Örneğin, *head* bölümünde yazılan kodlar, *body* bölümü henüz oluşturulmadan işletileceğinden DOM elementlerine ulaşamazlar. Çalışabilmesi için bu kodların *body* bölümünün sonuna yazılması ya da fonksiyonlara yerleştirilerek DOM oluşturulduktan sonra çağırılmaları gerekir. Aşağıda verilen kodda *getElementById* metodu, erişmek istediği meyveler listesinin üzerinde kullanılmıştır.

> **dikkat**
> DOM metotlarının elementlere ulaşabilmesi için bu elementlerin oluşturulmuş olması gerekmektedir.

```
1   <script>
2     var liste=document.getElementById("meyveler");
3     liste.append("<li>Elma</li>");
4   </script>
5   <ul id="meyveler"></ul>
```

İkinci satırdaki *getElementById* metodu çağırıldığında bir hata vermeden çalışmaktadır. Bununla birlikte, meyveler elementi henüz oluşturulmadığından bu elementi bulamayarak *null* değerini döndürmektedir. Bu satırda liste değişkeninin içeriği *null* olarak belirlenmektedir. Üçüncü satırda liste değişkeninde tanımlı elemente bir element eklenmeye çalışılmaktadır. liste değişkenindeki null değerinin *append* gibi bir metodu bulunmadığından bu satır Resim 4.10'da görülen hatayı vererek kodların çalışmasını durduracaktır.

> **dikkat**
> JS büyük küçük harf duyarlı bir dildir. Fonksiyon ve metot isimleri burada gösterildiği gibi yazılmak durumundadır. Örneğin getElementById metodu, getelementbyid şeklinde yazıldığında çalışmayacaktır. Tarayıcı bu metodu tanımayarak konsolda hata mesajı gösterecektir.



Resim 4.10 getElementById metodunun Yanlış Kullanımı Nedeniyle Karşılaşılan Hata

document.getElementsByTagName()

Bu fonksiyon verilen etiket adıyla işaretlenmiş bir elementler kümesi çevirir. Aşağıda verilen kodda, ikinci satırda belgedeki tüm p (paragraf) elementleri bulunarak paragraflar değişkenine gönderilmektedir.

```
1   <script>
2     var paragraflar = document.getElementsByTagName("p");
3     paragraflar[0].innerHTML = "ilk paragraf";
4   </script>
```

Bu metot kullanılırmen dikkat edilmesi gereken en önemli nokta bir element değil, elementler kümesi çevirmesidir. Bulunan paragraflar, değişken içinde bir dizi halinde tutulmaktadır. Sayfanızda 10 adet paragraf olduğunu düşünürsek, paragraflar değişkeni içinde 10 farklı paragrafın referansı yer almaktadır. Örneğin sayfanızdaki ilk paragraf üzerinde işlem yapmak için paragraflar[0] ifadesini kullanmanız gerekecektir. Dizi işlemleri daha sonra detaylı bir şekilde açıklanacaktır.

> **dikkat**
> Pek çok programlama dilinde olduğu gibi, JS dizi indisleri de sıfır ile başlamaktadır.

document.getElementsByClassName()

Class nitelikleri işaretlenerek HTML elementlerinden kümeler oluşturulabilir. Örneğin, sayfada artalanı mavi görünmesini istediğiniz tüm elementlere mavi sınıfını atayabilirsiniz. Sayfa üzerinde aynı sınıflı tüm elementleri bulmak için *getElementsByClassName* metodu kullanılır. Bu metot da bir elementler dizisi çevirmektedir. Aşağıdaki kodun beşinci satırında, sayfada mavi sınıfı ile işaretlenmiş tüm elementler bulunarak maviler değişkenine atanmaktadır. Altıncı satırda maviler değişkenindeki dizinin ikinci elementinin içeriği "ikinci paragraf" olarak düzenlenmektedir.

```
1   <p class="mavi">paragraf 1</p>
2   <p class="mavi">paragraf 2</p>
3   <button class="mavi">düğme 1</button>
4   <script>
5     var maviler = document.getElementsByClassName("mavi");
6     maviler[1].innerHTML = "İkinci paragraf";
7   </script>
```

Öğrenme Çıktısı

4 JavaScript programlama dili ile HTML DOM'da element bulabilme

Araştır 3

Bir sayfadaki paragrafların (p etiketlerinin) sayısını bulan JS kodunu yazınız.

İlişkilendir

Bir Web sayfasındaki DOM elementine ulaşmak için kullanılan document.getElementById, document.getElementsByClassName ve document.getElementsByTagName metotlarını kullanışlılıkları açısından karşılaştırınız.

Anlat/Paylaş

Web sayfalarında DOM elementlerine erişim konusunda yaşadığınız zorlukları ve önerilerinizi paylaşınız.

ÇIKTI MEKANİZMALARI

JavaScript kodlarının sonuçları kullanıcıya beş farklı çıktı mekanizmasıyla yansıtılabilir. Bunların üçü HTML DOM üzerinde çalışırken, ikisi tarayıcının (BOM) iletişim özelliklerini kullanır.

innerHTML

innerHTML özelliği bir HTML elementinin içeriğini düzenlemek için kullanılır. innerHTML bir özellik olduğundan, bir fonksiyon gibi parantezler ile çağırılmaz. Bunun yerine bir atama işlemi gibi çalıştırılır. Aşağıda verilen kodda, üçüncü satırda sayfada id özelliği paragraf olarak belirlenmiş *p* elementi bulunarak *p1* değişkenine atanmaktadır. Dördüncü satırda bu elementin *innerHTML* özelliği kullanılarak içeriği düzenlenmektedir.

```
1   <p id="paragraf"></p>
2   <script>
3     var p1 = document.getElementById("paragraf");
4     p1.innerHTML = "İlk paragrafım.";
5   </script>
```

innerHTML özelliğinin elementin içeriğini silerek düzenlediğine dikkat ediniz. Element içeriğine ekleme yapmak istiyorsanız *innerHTML* niteliği eşittir (=) işleci ile değil; artı eşittir (+=) işleci ile kullanılmalıdır. Bu işlem, elementin içeriğinin arkasına yeni içeriği ekleyecektir.

dikkat

innerHTML özelliği sadece elementin metin içeriğini değil, içinde bulunabilecek diğer elementleri de değiştirir. Bu özellik table, ul, ol gibi içerik tutucu elementlerle kullanılırken dikkatli olunmalıdır.

document.write()

HTML sayfasına içerik eklemek için *document.write()* metodu kullanılabilir. Kullanışlı bir araç gibi görülmesine rağmen, sadece geliştirme ve sınama amaçlarıyla kullanılması önerilmektedir. Bu metot, HTML kodu da dahil olmak üzere sayfaya herhangi bir içeriği eklemeye izin verir. Aşağıdaki kodda *document.write* metodu kullanılarak sayfaya bir paragraf eklenmektedir. Bu metot verilen içeriği sayfaya çalıştığı noktada eklemektedir. Bu nedenle metodun, element ya da içeriğin ekleneceği yerde çalıştırılması gerekmektedir.

```
1   <script>
2     document.write('<p id="paragraf">İlk paragrafım.</p>');
3   </script>
```

Kullanıcıya sunulan sayfalarda appendChild ve innerHTML metotlarının kullanılması daha doğru bir yaklaşımdır.

dikkat

DOM yüklendikten sonra document.write metodunun kullanımı innerHTML kullanımına benzer bir sonuç verir. Örneğin bir düğmenin tıklanmasıyla tetiklenen fonksiyonlarda kullanılması halinde sayfa içeriği sıfırlanarak verilen içerik sayfaya yazılacaktır.

appendChild()

appendChild metodu bir HTML elementine yeni bir çocuk element ekler. DOM mimarisinde bir elementin içine eklenen yeni elementler çocuk element olarak ele alınır. Örneğin sırasız listeler oluşturan *ul* elementinin çocukları içindeki li elementleridir. Benzer şekilde bu *ul* elementi *body* elementinin çocuk elementidir. Kapsayan (dışarıdaki) elementler ebeveyn (parent), kapsanan elementlerse çocuk (child) element olarak ele alınır.

```
1   <script>
2     var yeni = document.createElement("li");
3     yeni.innerHTML = "Ayva";
4     var liste = document.getElementById("meyveler");
5     liste.appendChild(yeni);
6   </script>
```

Yukarıda verilen kodun ikinci satırda *document* nesnesinin *createElement* metodu kullanılarak yeni bir li elementi oluşturulmaktadır. Üçüncü satırda *li* elementinin içeriği *Ayva* olarak düzenlenmektedir. Dördüncü satırda sayfada *id* niteliği *meyveler* olarak belirlenen *ul* elementi bulunarak liste değişkenine atanmaktadır. Son olarak beşinci satırda oluşturulan element *appendChild* metodu kullanılarak *ul* elementine eklenmektedir. Oluşturulan elementin sayfadaki listeye eklenmesi Resim 4.11'de gösterilmiştir.

Resim 4.11 appendChild metodu ile Listeye Element Eklemek

console.log()

Tarayıcılar sayfalardaki hataları ve uyarıları bildirmek üzere konsol (console) adlı bir araç içermektedir. Bu araç pek çok tarayıcının Görünüm menüsü kullanılarak açılabilir. Örneğini Goolge Chrome tarayıcısında konsolu açmak için Görünüm/Geliştirici/JavaScript Konsolu (View/Developer/JavaScript Console) yolu ya da Ctrl+Alt+J kısayolu kullanılabilir. Sayfa üzerine sağ tıklandığında gösterilen menüdeki "İncele" seçeneği de konsol görüntüsüne ulaşmanıza olanak verecektir.

```
1   <script>
2     document.write('<p id="paragraf">paragraf metni</p>');
3     console.log("paragraf eklendi");
4   </script>
```

JS kodlarında hata bulunması halinde bu hataya ilişkin bilgiler konsolda listelenir. Bunun yanında, programcılar *console.log()* metodunu kullanarak konsol çıktısı üretebilir. Yukarıdaki kodun üçüncü satırında, geliştirici sayfadaki bir işlemin gerçekleştiğini göstermek üzere konsola mesaj göndermektedir.

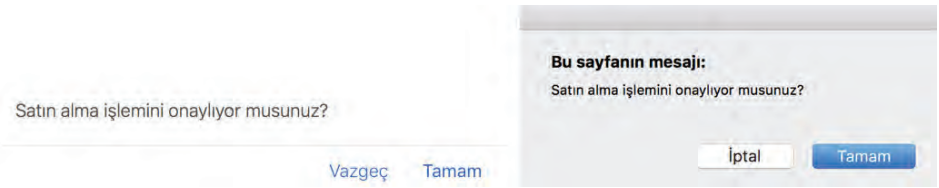
window.alert()

window.alert() metodu kullanıcıya uyarı kutusu göstermek için kullanılır. Bu metot kullanıcıdan veri almayıp, sadece bilgilendirme amacı ile kullanılmaktadır. Aşağıda verilen kodun üçüncü satırında kullanıcıya bir uyarı mesajı gösterilmektedir.

```
1   <script>
2     document.write('<p id="paragraf">Yeni paragraf metni.</p>');
3     window.alert("paragraf eklendi");
4   </script>
```

window.confirm()

window.confirm() metodu kullanıcıya diyalog kutuları göstermek için kullanılır. Bu metot kullanıldığında kullanıcının *Tamam* ya da *İptal* diyebileceği bir onay kutusu gösterilmektedir. Bu kutu üzerinde gösterilen mesajların dili, işletim sisteminden otomatik olarak çekilir. Bu nedenle düğmeler üzerinde yazan metinler değiştirilemez. Bunun yanında diyalog kutularının görünümlerinin tarayıcıdan tarayıcıya değiştiğini unutmayınız. Resim 4.12'te aynı kodla oluşturulmuş diyalog kutusunun Safari (solda) ve Chrome (sağda) tarayıcılarındaki görünümleri görülmektedir.



Resim 4.12 Safari (solda) ve Chrome (sağda) Tarayıcılarından Diyalog Kutusu Örnekleri

Diyalog kutularından *true* ya da *false* değer dönmektedir. Kullanıcının onaylayan düğmeye (Tamam ya da OK) basması halinde *true*, aksi halde (İptal ya da Cancel) *false* değer dönmektedir. Programcı gelen yanıta göre gerekli rutinleri çalıştıracaktır. Aşağıdaki kod örneğinde, "Satın alma işlemini onaylıyor musunuz?" sorusu gösterilerek, kullanıcının yanıtı onay değişkenine aktarılmaktadır. Programcı ilerleyen satırlarda koşul yapısı içinde gerekli kodları yazacaktır.

```
1   <script>
2   var onay = window.confirm("Satın alma işlemini onaylıyor musunuz?");
3     if (onay)
4     {
5       // Kullanıcı Tamam'a bastıysa işletilecek kodlar
6     }
7     else
8     {
9       // Kullanıcı İptal'e bastıysa işletilecek kodlar
10    }
11  </script>
```

Öğrenme Çıktısı

5 JavaScript programlama dilinin çıktı mekanizmalarını kullanabilme

Araştır 4

Bir sayfadaki paragrafların içeriğini, paragrafın sayfadaki sırasını belirtecek şekilde düzenleyen JS kodunu yazınız.

İlişkilendir

JavaScript dilinde kullanılan çıktı elementlerini özellikleri açısından karşılaştırınız.

Anlat/Paylaş

Web tarayıcılarının iletişim özelliklerini kullanan çıktı mekanizmalarını açıklayınız.

VERİ TİPLERİ

Programlama dilleri değişkenler içinde farklı tiplerde veri saklayabilir. C# gibi üst düzey programlama dillerinde uygulama performansını arttırmak ve hata ayıklamayı desteklemek gibi amaçlarla pek çok veri tipi kullanılır. Bu tür dillerde sadece sayısal değerleri tutmak için bile birkaç tane veri tipi kullanılır. Örneğin, C# dilinde 0-255 arasındaki sayısal değerler için *byte*, ±32768 aralığındaki tam sayılar için *short* veri tipi kullanılır. Ondalıklı sayıları ve çok büyük sayıları tutmak içinse *float*, *double*, *decimal* gibi veri tipleri kullanılır.

Bir betik dili olan JS, üst düzey programlama dillerine oranla çok daha küçük bir veri tipi kümesi kullanır. JS programlarında beş farklı veri tipi kullanılır (Tablo 4.6). C# gibi dillerden farklı olarak JS değişkenlerinin ön tanımlı bir veri tipi yoktur. JS değişkenleri dinamik olduğundan, kendilerine atanan değere göre farklı veri tiplerinde olabilirler. Bununla birlikte, değişkende tutulan veri tipi, değişken üzerinde kullanılabilecek metotları belirler. Örneğin Boolean tipinde veri tutan bir değişkende, parça kesme gibi String veri tipine özgü metotları kullanmanız durumunda programınız hata verecektir.

Tablo 4.6 JS Dilinde Kullanılan Veri Tipleri

Tür	Açıklama	Örnek
String	Metinleri tutmak için kullanılır	var isim="onur"
Number	Sayıları tutmak için kullanılır	var sayi=10.5;
Boolean	Mantıksal (doğru/yanlış) verileri tutmak için kullanılır	var sonuc=true;
Array	Dizileri tutmak için kullanılır. Birleşik veri tipidir.	var dizi=[1,2,3,4]
Object	Nesneleri tutmak için kullanılır. Birleşik veri tipidir.	var nesne={r:15, x:3}

String

String veri tipi karakter dizilerini tutabilir. Metin türündeki bu veriler "çift tırnak" ya da 'tek tırnak' çiftleri arasında ifade edilmelidir. Tırnaklar arasına alınmış "100" değerinin sayısal değil, metin olarak saklandığına dikkat ediniz.

dikkat

Bu ünitede yer kısıtları nedeniyle veri tiplerine özgü metotlara değinilmemiştir. Okuyucuların özellikle String veri tipine özgü (Ör: length, indexOf, substr …) metotları incelemesi önerilmektedir.

Number

Number veri tipi sayısal değerleri tutmak için kullanılır. *Number*, tam sayılar, ondalıklı sayılar ve negatif sayıları tutmak için kullanılabilir. Bunun yanında çok büyük sayılar üstel ifadeler olarak yazılabilir. Örneğin 1e6 ifadesi $1x10^6$ =1.000.000 değerine eşittir. Benzer şekilde 2e-3 ifadesi ise 0,002 ($2x10^{-3}$) değerine eşittir.

Boolean

Boolean veri tipi mantıksal doğru ve yanlış ifadelerini tutmak için kullanılır. Mantıksal karşılaştırma (4>3) ve değişken içeriği sorgulama (isArray) gibi işlemlerin sonuçları *Boolean* değişkenlerde tutulur.

✓ isArray fonksiyonu bir değişkenin dizi tipinde veri tutup tutmadığını kontrol eder. Verilen değişken dizi ise true, değilse false değerini çevirir.

Array

Array veri tipi benzer nitelikteki ya da birbirleri ile ilişkili değerlerin tutulması için kullanılır. *Array* birden fazla değeri bir arada tutabildiğinden birleşik bir veri tipidir. Programlama alanında bu veri tipine dizi adı verilmektedir.

Örneğin, bir meyve kümesini ifade etmeye çalıştığımızı düşünelim. Bu meyvelerin her birini farklı bir değişkene atayabilirsiniz.

```
1   <script>
2     var meyve1="Ayva";
3     var meyve2="Elma";
4     var meyve3="Armut";
5   </script>
```

Fakat bu yapı programlamak için oldukça zordur. Burada sadece üç meyve kullanılmıştır, fakat 30 adet meyvemiz de olabilirdi. Bu durumda 30 farklı değişken tanımlamamız ve programlayabilmek için bunların isimlerini hatırlamamız gerekecekti. Bunu yapabilseydik bile, kaç tane meyvemiz olduğunu bilemez, ya da bu meyveler içinde "Nar" değerinin bulunup bulunmadığını otomatik olarak arayamazdık. Bu tür işlemler için dizi (Array) yapılarının kullanılması gerekir. Diziler köşeli parantezler içinde ([…]) tanımlanır. Dizi elementleri virgüller ile birbirlerinden ayrılır. Aşağıdaki kodda iki adet dizi tanımlanmaktadır.

```
1   <script>
2     var meyveler=["Ayva", "Elma", "Armut"];
3     var sayilar=[10, 20, 30, 40];
4   </script>
```

İkinci satırda yukarıda kullandığımız üç meyve, bir diziye atılarak meyveler isimli değişkene atanmaktadır. Üçüncü satırda ise, dört adet sayısal değer içeren bir dizi tanımlanarak sayilar değişkenine atanmaktadır. Diziler herhangi bir veri tipindeki değerleri tutabilir. Bir dizinin içinde başka bir dizi, sayı, metin ya da nesne tipinde bir değişken tutulabilir. Dizi elementlerine atandıkları değişken ismi ve bu dizideki sıra numaralarını ifade eden indislerle ulaşılır.

```
1   <script>
2     Meyvem = meyveler[1];
3   </script>
```

Pek çok programlama dilinde olduğu gibi, JS dilinde ilk elementin indisi sıfırdır. Bir dizi elementine ulaşmak için dizinin atandığı değişken adı ve köşeli parantezler içinde indis değeri kullanılır. Örneğin meyveler dizisindeki "Elma" elementine ulaşmak için yukarıdaki kodda verilen ifade kullanılır.

JS dilinde, dizi kullanmanın en önemli avantajlarından biri dizi boyutlarının sınırlı olmayışıdır. Tanımlanmış bir diziye yeni eleman eklemek için *push*() metodu kullanılabilir. Örneğin meyveler dizisine "Kiraz" meyvesini eklemek için aşağıda verilen koddaki kod satırı kullanılabilir.

```
1   <script>
2     meyveler.push("Kiraz");
3   </script>
```

Diziye değer eklemenin başka bir yolu da kullanılmayan bir indise atama yapmaktır. Örneğin, meyveler dizisinde dört meyve olduğunu düşünelim. Bu nedenle meyveler dizisindeki en yüksek tanımlı indis üçtür. Meyveler dizisine yeni bir element eklemek için, kullanılmayan ilk indis olan dördü aşağıdaki şekilde kullanabiliriz.

```
1   <script>
2     meyveler[4] = "Şeftali";
3   </script>
```

Geçerli bir yöntem olmakla birlikte, kullanılmayan ilk indisten daha büyük indisler kullanılması durumunda (ör: meyveler[9]="Karpuz") dizide delikler oluşabilir. Bu nedenle *push*() metodunun kullanılması daha uygun bir yaklaşımdır. Bunun yanında, kullanılan bir indise atama yapılması halinde yeni değer eskisinin üzerine yazılacaktır. Örneğin aşağıda bir numaralı indisteki "Elma" değeri "Karpuz" ile değiştirilmektedir.

```
1   <script>
2     meyveler[1] = "Karpuz";
3   </script>
```

Bir dizide kaç eleman bulunduğunu öğrenmek için length özelliği kullanılır. Aşağıda verilen kodda meyveler dizisindeki eleman sayısı meyve_sayisi değişkenine atanmaktadır.

> Bir dizideki tüm elemanları sırayla incelemek için, dizinin içindeki eleman sayısına kadar çalışacak bir döngü kullanılır. Bu döngüde bitiş değerini bulmak için length özelliği kullanılır.

```
1   <script>
2     meyve_sayisi = meyveler.length;
3   </script>
```

Dizilerle çalışırken kullanılabilecek bazı önemli metotlar Tablo 4.7'te listelenmiştir.

Tablo 4.7 Dizilerle Çalışırken Kullanılabilecek Önemli Metotlar

Metot	Açıklama	Örnek
toString	Dizi içeriğini metin olarak verir	console.log(meyveler.toString());
push	Dizinin sonuna yeni eleman ekler	meyveler.push("Kiraz");
pop	Dizideki son elemanı çıkarır	son_meyve=meyveler.pop();
shift	Dizideki ilk elemanı çıkarır	ilk_meyve=meyveler.shift();
delete	Diziden verilen indisli elemanı siler	delete meyveler[3];
splice	Diziden verilen indisten itibaren verilen sayı kadar eleman siler, verilen elemanları araya ekler	meyveler.splice(1,2,"Mango");
concat	İki diziyi birleştirir	birleşim=meyveler.concat(sebzeler);
slice	Verilen indisten itibaren verilen sayı kadar elemanı kopyalar	meyveler2=meyveler.slice(1,2);
sort	Diziyi sıralar	meyveler=meyveler.sort();

Bu metotların çalışma prensipleriyle ilgili örnekler ve açıklamalar Tablo 4.8'da verilmiştir. Kodların çalıştırılmasından sonra değişkenlerin durumları kalın yazılar ile belirtilmiştir.

Tablo 4.8 Dizi Metotlarının Örneklenmesi

Kodlar	Açıklama
var meyveler=["elma","armut","portakal"];	meyveler dizisi tanımlanıyor.
var meyveler2=["muz","kavun","mandalina"];	meyveler 2 dizisi tanılanıyor.
meyveler.push("karpuz");	meyveler dizisinin sonuna karpuz elemanı ekleniyor
meyveler: elma, armut, portakal, karpuz	
meyvem=meyveler.pop();	meyveler dizisinden son element çıkarılıyor ve meyvem değişkenine gönderiliyor
meyvem: karpuz	
meyveler: elma, armut, portakal	
meyvem=meyveler.shift();	meyveler dizisinden ilk eleman çıkarılıyor ve meyvem değişkenine gönderiliyor
meyvem: elma	
meyveler: armut, portakal	
delete meyveler[0];	meyveler dizisindeki 0 indisli elementin içi boşaltılıyor, fakat bu elaman silinmiyor
meyveler: ,portakal	
meyveler[0]="armut";	meyveler dizisindeki 0 indisli elemana armut değeri atanıyor
meyveler: armut, portakal	
meyveler.splice(0,1);	meyveler dizisinden sıfırıncı indisten itibaren bir eleman siliniyor, eleman eklenmiyor

meyveler: portakal	
meyveler=meyveler.concat(meyveler2);	meyveler ve meyveler2 dizileri birleştirilerek meyveler değişkenine atanıyor.
meyveler: portakal, muz, kavun, mandalina	
meyveler.splice(1,0, "elma");	meyveler dizisinde 1 numaralı indisten itibaren element silinmeden araya elma elemanı ekleniyor
meyveler: portakal, elma, muz, kavun, mandalina	
meyveler.sort();	meyveler dizisinin içeriği sıralanıyor
meyveler: elma, kavun, mandalina, muz, portakal	
var meyveler3=meyveler.slice(1,3);	meyveler dizisinden 1 indisli elementten itibaren 3 indisli elemente kadar tüm elemanlar kopyalanarak meyveler3 değişkenine gönderiliyor
meyveler: elma, kavun, mandalina, muz, portakal meyveler3: kavun, mandalina, muz	

Bu bölümde dizilerle ilgili pek çok metot kullanılmışsa da yer kısıtlamaları nedeniyle JS'in sunduğu tüm özellikler kullanılmamıştır. Okuyucu "JavaScript dizi metotları" gibi internet aramaları ile diğer metotlara ulaşabilir.

Object

JS dilindeki birleştirilmiş veri yapılarından bir diğeri nesnelerdir (Object). Dizilere benzemekle birlikte nesnelerdeki içerikler indislerle değil, isimlerle tanımlanır. Nesne tanımlamaları küme parantezleri arasında ({ … }) yapılır. Nesnenin özellikleri virgüller ile birbirlerinden ayrılır. Nesne özelliği ve değeri ise iki nokta (özellik : değer) ile birbirinden ayrılır. Bir kişinin kimlik bilgilerini bir nesne içinde tutmak için aşağıdaki gibi bir tanımlama yapabiliriz.

```
1   <script>
2     var kisi={isim:"onur", soyisim:"dönmez", yas:34};
3   </script>
```

Bu örnekte *kisi* nesnesi tanımlanmıştır. Birbirleri ile ilişkili bir veri grubunu nesneler içinde tutabilirsiniz. Örneğin bir kişinin özellikleri, bir arabanın özellikleri, bir bilgisayarın özelliklerinden bahsedebiliriz. Bu çerçevede, bu değişkenlerin nesne olarak tanımlanmaları uygun olacaktır. Bir sınıftaki öğrencileri ise dizi yapısında tutmak daha uygundur. Bu öğrencilerin her birinin kendine özgü özellikleri bulunmaktadır. Her bir öğrenci bir nesne, tüm öğrencilerin kümesi olan sınıf ise bir dizi ile ifade edilmelidir.

dikkat

Diziler nesneleri, nesneler de dizileri eleman olarak tutabilirler.

Tablo 4.9 Kisi Nesnesinin Özelliklerine Erişmek

Özellik	Ulaşım Yöntemi	Değer
isim	kisi.isim	onur
soyisim	kisi.soyisim	dönmez
Yas	kisi.yas	34

Nesnelerin özelliklerine erişmek için değişken adının ardından nokta koyarak özellik adı yazılır (Tablo 4.9). Aşağıda verilen kodda yas değişkenine, *kisi* nesnesinin *yas* özelliğinde tutulan 34 değeri atanmaktadır. Benzer şekilde bir nesnenin özelliğine de atama yapılabilir. Üçüncü satırda kisi nesnesinin yas özelliğine 19 değeri atanmaktadır.

```
1   <script>
2     var yas = kisi.yas;
3     kisi.yas = 19;
4   </script>
```

Öğrenme Çıktısı

6 JavaScript programlama dilinde kullanılan veri tiplerini ayırt edebilme

Araştır 5

Beril, Soyadı: Karasu, Yaşı: 14, Sıralaması: 3 özelliklerde bir ogrenci nesnesi oluşturarak, sınıf dizisine ekleyiniz.

İlişkilendir

JavaScript betik dilindeki veri tipi çeşitliliğini C# programlama dillerindeki veri tipi çeşitliliği ile karşılaştırınız.

Anlat/Paylaş

JS dilindeki nesnelerin kullanım amaçlarını açıklayınız.

KOŞUL YAPILARI

JS programları art arda verilen yönergelerin yorumlanması yoluyla işletilir. Fakat gerçek yaşamda pek az program bu doğrusal işleyişle yönetilebilir. Kullanıcınızın seçtiği bir radyo düğmesi, kullanıcı sözleşmesini kabul edip etmemesi, sitenizin ziyaret edilme saati gibi pek çok duruma bağlı olarak program akışının yönlendirilmesi gerekir. Program akışının yönlendirilmesi için koşul yapıları kullanılır. Koşul yapıları, yazılan şart ifadelerine göre işletilecek kod bloklarının yazılması için alan sağlar. JS diğer programlama dillerinde sıklıkla kullanılan *if*, *else* ve *switch* yapılarını kullanmaktadır.

if

if, kullanılabilecek en temel koşul yapısıdır. Bu yapı, bir şartın sağlanması durumunda işletilecek kod blokları yazılmasına izin verir. Şartlar mantıksal ifadeler (ör: a==b) şeklinde yazılır. Mantıksal ifadeler *if* sözcüğünden hemen sonra parantezler içine yazılır. Mantıksal ifadeden *true* (doğru) sonucunun çıkması durumunda çalıştırılacak kodlar *if* ifadesinin hemen arkasında küme parantezleri ({ … }) içine yazılır. Program akışı küme parantezlerinden sonra doğrusal olarak devam edecektir.

```
1   <script>
2     if(koşul ifadesi)
3     {
4       // koşul ifadesi sağlanırsa çalıştırılacak kodlar
5     }
6   </script>
```

Örneğin, internet sitenize üyelik için en az 13 yaşında olmak ölçütünün arandığını düşünelim. Bu kontrolü gerçekleştirmek için if yapısı kullanılabilir. 13 yaş ve üzerindeki kullanıcılar için kayıt rutinleri çalıştırılacaktır. Aşağıda verilen kodda ikinci satırda yas değişkenine 12 değeri atanmıştır.

```
1   <script>
2     var yas=12;
3     if(yas>12)
4     {
5       // Kayıt rutinlerini çalıştır
6     }
7   </script>
```

Üçüncü satırdaki if yapısında yas değişkeninin 12'den büyük olup olmadığı kontrol edilmektedir. Buradaki kıyaslama işlemi false değerini döndüreceğinden (yanlıştır çünkü 12, 12'den büyük değildir) kayıt rutinleri çalıştırılmayacaktır.

> yas değişkeninin değeri 13 olsaydı if yapısı true değeri döndürecek (13, 12'den büyüktür) küme parantezleri içindeki kayıt rutinleri çalıştırılacaktı.

else

if yapısına ikinci bir blok bağlamak için else anahtar sözcüğü kullanılabilir. *if* yapısında verilen mantıksal ifadenin yanlış olması durumunda else bloğu çalıştırılır. *else* yazılırken mantıksal ifade verilmez. Bu nedenle else blokları her zaman bir *if* yapısının arkasında kullanılır. else blokları, yalnızca *if* yapısındaki şartın yanlış olması durumunda çalıştırılır. Bu şarta bağlı olmayan kodların yazılması için else bloğuna ihtiyaç yoktur. Aşağıda bir *ifelse* bloğunun genel ifadesi yer almaktadır.

```
1   <script>
2     if(koşul ifadesi)
3     {
4       // Koşul ifadesi sağlanırsa çalıştırılacak kodlar
5     }
6     else
7     {
8       // Koşul ifadesi sağlanmazsa çalıştırılacak kodlar
9     }
10  </script>
```

Bir önceki örneğimizden devam edecek olursak; kayıt yapmak isteyen kullanıcımızın yaşının 13'ün üstünde olması halinde kayıt rutinlerini çalıştırabilir, aksi halde *else* bloğunda kendisine uyarı mesajı verebiliriz.

```
1   <script>
2     var yas=12;
3     if(yas>12)
4     {
5       // Kayıt rutinlerini çalıştır
6     }
7     else
8     {
9   alert("Bu siteye üye olmak için en az 13 yaşında olmalısınız.");
10    }
11  </script>
```

Yukarıdaki kodda ikinci satırdaki *if* ifadesi *yas* değişkeninin değerinin 12'den büyük olup olmadığını kontrol etmektedir. Bu örnekte *yas* değişkeni 12'den büyük olmadığından *else* bloğu çalışacak, kullanıcıya bir uyarı penceresi gösterilecektir.

dikkat

else anahtarının arkasında yeni bir if bloğu yazılabilir. Bu sayede ilk şartın sağlanmaması durumunda yeni bir şart kontrol edilerek program dallanması sağlanabilir.

switch

if/else yapısı bir şart ifadesinin doğru ya da yanlış olma durumuna göre çalışır. Bir değişken ile ilgili birden fazla duruma yanıt vermemiz gerektiğinde art arda *if* yapıları kullanmamız gerekebilir. Bu yapıların art arda kullanımını yönetmek zordur. Ayrıca, art arda aynı yapıları kullanmak kodlarınızda karmaşaya yol açabilir. Bunun yerine *switch* yapısını kullanarak bir değişkenin çeşitli durumlarında çalıştırılacak kod blokları yazabilirsiniz. *switch* yapısının genel kullanım ifadesi aşağıda verilmiştir.

```
1   <script>
2     switch (degisken)
3     {
4       case durum1:
5         //durum1 halinde çalıştırılacak kodlar
6       break;
7       case durum2:
8         //durum2 halinde çalıştırılacak kodlar
9       break;
10      default:
11  //üstteki durumdaki şartlar sağlanmadığında çalıştırılacak kodlar
12      break;
13    }
14  </script>
```

switch yapısı bir değişkene bağlı çalışır. Bu değişken *switch* sözcüğünden sonra parantezler içinde yazılır. *switch* bloğu küme parantezleri ({ … }) ile işaretlenir. Küme parantezleri içinde değişkenin alabileceği değerler *case* sözcüğünün arkasına yazılır. Örneğin değişkenin değerinin 1 olduğu durumda çalıştırılacak kodlar "case 1:" ifadesi ile, 7 olması durumunda çalıştırılacak kodlar "case 7:" ifadesi ile belirtilir. Bu ifadeden sonra durum için çalıştırılacak kodlar yazılır. Kodların bitişi break ile belirtilir. Ardından bir sonraki durumun case ifadesi yazılabilir. İstenen durumların belirtilmesinden sonra, bu durumların karşılanmaması halinde çalışacak kodlar "default:" sözcüğü ile başlayan kısma yazılır.

Şimdi bir *switch* bloğu yazarak kullanımını örnekleyelim. Kullanıcılarımıza haftanın hangi gününde olduğumuzu hatırlatmak istediğimizi varsayalım. Bu amaçla bir *switch* bloğu yazabiliriz. JS tarih (date) nesnesinin *getDay* metodu verilen tarihin haftanın hangi gününe denk geldiğini belirtir. Bu metodun çıktısı (sıfır Pazar olmak üzere) 0-6 arasında sayısal bir değerdir. Bu sayısal değeri kullanarak haftanın hangi gününde olduğumuzu kullanıcıya bildirebiliriz. Aşağıda haftanın hangi gününde olduğumuzu sayfada *id*'si paragraf olan elementin içine yazan bir kod bloğu görülmektedir.

```
1   <p id="paragraf"></p>
2   <script>
3     var bugun=new Date().getDay();
4     var mesaj;
5     switch(bugun)
6     {
7       case 0: mesaj="Bugün Pazar"; break;
8       case 1: mesaj="Bugün Pazartesi"; break;
9       case 2: mesaj="Bugün Salı"; break;
```

```
10        case 3: mesaj="Bugün Çarşamba"; break;
11        case 4: mesaj="Bugün Perşembe"; break;
12        case 5: mesaj="Bugün Cuma"; break;
13        case 6: mesaj="Bugün Cumartesi"; break;
14        default: mesaj="Böyle bir gün yok.";
15      }
16      document.getElementById("paragraf").innerHTML=mesaj;
17    </script>
```

Öncelikle hangi günde olduğumuzu sayısal olarak elde etmemiz gerekmektedir. Üçüncü satırda yeni bir tarih (date) nesnesi oluşturularak, bu nesnenin *getDay* metodunun çevirdiği değer *bugun* değişkenine atanmaktadır. Tarih nesnesi oluşturulurken bir tarih verilmezse güncel tarihi belirten bir nesne oluşturulur. Dördüncü satırda kullanıcıya gösterilecek mesaj değişkeni tanımlanmaktadır. 5. Satırda mesajın hazırlanması için *bugun* değişkeninin durumlarını kontrol edecek bir *switch* yapısı başlatılmaktadır. Yedinci ve 13. satırlar arasında bugun değişkeninin farklı değerleri için gerekli mesaj hazırlama işlemleri gerçekleştirilmektedir. *getDay* metodunun sıfır Pazar gününü ifade etmek üzere, sıfır ile altı arasında değerler ürettiğini hatırlayınız. Bu nedenle *case 0*: ile işaretli bölümde "Bugün Pazar" mesajı hazırlanmaktadır. Arta kalan günler için ayrı ayrı case ifadeleri yazılmıştır. *getDay* metodu altıdan büyük bir değer göndermez. Yine de 14. satırda kontrol amaçlı olarak bir mesaj satırı yazılmıştır. 16. satırda belgedeki paragraf bulunarak mesaj değişkeni içine yazılmaktadır.

Öğrenme Çıktısı

7 JavaScript programlama dilindeki koşul yapılarını kullanabilme

Araştır 6

Siz de yılın hangi ayında olduğumuzu kullanıcıya bildiren bir switch yapısı oluşturunuz.

İlişkilendir

JavaScript dilinde yaygın olarak kullanılan if ve else yapılarını ilişkilendirin.

Anlat/Paylaş

JavaScript'te kullanılan koşul yapılarının kodlanmasıyla ilgili yaşadığınız zorlukları ve çözümlerinizi paylaşınız.

DÖNGÜLER

Döngünler verilen bir kod bloğunu bir şart sağlandığı sürece tekrar tekrar işletilmesini sağlayan programlama yapılarıdır. JS üç döngü yapısı sağlamaktadır. Temelde tüm döngü yapılarının görevi belirlenen şart sağlandığı sürece verilen kod bloklarını işletmektir. Sağlanan döngü yapılarının tek farkı şartın ifade ediliş biçimidir.

while

while döngüsü verilen şart ifadesi sağlandığı (true sonuç verdiği) sürece bir kod bloğunu işletir. while döngüsünün temel yazım ifadesi aşağıda gösterildiği şekildedir.

```
1   <script>
2     while(koşul ifadesi)
3     {
4       // Koşul ifadesi sağlandığı sürece çalıştırılacak kodlar
5     }
6   </script>
```

Döngü, *while* anahtar sözcüğü ile başlatılır. Hemen ardından parantezler içinde şart ifadesi verilir. Şart ifadesinden sonra işletilecek kod bloğu küme parantezleri ({ … }) içinde yazılır. Aşağıda verilen kodda ikinci satırda i değişkeninin değeri 10'dan küçük olduğu sürece işletilecek bir döngü tanımlanmıştır. Beşinci satırda tarayıcı konsolunda i değişkeninin değeri yazdırılmaktadır. Altıncı satırda ise i değişkeninin değerine 1 eklenmektedir.

> while döngülerinde kontrol değişkenine döngü içinde müdahale edilmelidir. Aksi halde döngünün işletilmesi sonlanmayabilir. Bu tür döngülere sonsuz döngü denir. Sonsuz döngüler tarayıcıların tepkisiz kalmasına neden olabilir.

```
1   <script>
2     var i = 0;
3     while (i<10)
4     {
5       console.log("i değişkeninin değeri:"+i);
6       i++;
7     }
8   </script>
```

while döngüleri her zaman arttırılarak işletilmez. Bir *while* döngüsü verilen değişkenin değeri bir değerden büyük olduğu sürece de işletilebilir. Aşağıda bir dizi içindeki değerler while döngüsü ile listelenmektedir.

```
1   <script>
2     var meyveler = ["elma", "armut", "portakal", "muz", "nar"];
3     while (meyveler.length>0)
4     {
5       console.log("Sıradaki meyve:"+meyveler.pop());
6     }
7   </script>
```

Beşinci satırda pop metodu kullanılarak dizinin en sonundaki element çıkarılmaktadır. Bu yöntem her dönüşte diziden bir element çıkardığından, döngü dizide element bulunduğu sürece dönecek şekilde düzenlenmiştir. Üçüncü satırda dizide element bulunup bulunmadığı *length* özelliği ile kontrol edilmektedir. Dizide element kalmadığında *length* özelliği 0 değerini aldığından döngü sonlandırılacaktır.

> Programlama alanında istenmeyen bir durum olmakla birlikte, ağ portların dinlenmesi gibi bazı durumlarda sonsuz döngü üretilmek istenebilir. Bu amaçla while(true) şeklinde başlayan bir ifade yazılması yeterlidir.

do/while

do/while döngüsü, *while* döngüsünün bir türevidir. *do/while* döngüleri *do* anahtar sözcüğüyle başlar. Bunun ardından küme parantezleri ({ … }) içinde işletilecek kod bloğu yazılır. Kod bloğunun arkasında döngünün işletilmesini kontrol eden while ifadesi yazılır. *do/while* döngüsünün, *while* döngüsünden tek farkı, verilen şart sağlanmıyor olsa bile döngünün en az bir kez dönmesidir.

```
1   <script>
2   var sayilar = new Array();
3   var sayi;
4   do {
5     sayi = prompt("Bir sayı giriniz (sonlandırmak için 0):");
6     sayilar.push(sayi);
7   } while(sayi!=0)
8   console.log(sayilar);
9   </script>
```

do/while döngüleri genellikle dışarıdan veri okuma işlemleri için kullanılır. Bu veri bir porttan ya da kullanıcıya gösterilen bir diyalog penceresinden gelebilir. Yukarıdaki örneğimizde kullanıcıdan gelen verilerle *sayilar* dizisi doldurulmaktadır. *prompt* metodu veri giriş kutusunu gösterip, kullanıcının girdiği değeri çevirmektedir. Bu değer *sayi* değişkenine atılarak daha sonra *sayilar* dizisine eklenmektedir. Döngünün kontrol bloğunda sayı değişkeninin değeri kontrol edilerek sıfır girilmesi durumunda döngü sonlandırılmaktadır. Bu işlemi *while* döngüsü ile yapmak isteseydik *prompt* metodunu hem döngü içinde hem de döngünün önünde birer kez çağırmamız gerekirdi. *do/while* döngüleri en az bir kez çalıştırıldığı için bu tür bir işleme gerek bırakmazlar.

for

İşlev bakımından *for* ve *while* döngüleri arasında fark yoktur. *for* döngülerinin farkı, döngünün işletilmesi ile ilgili tüm işlemlerin kontrol bloğu içerisinde yapılmasıdır. *while* döngülerinde kontrol bloğunda sadece şart ifadesi yer alırken; *for* döngülerinin kontrol bloğunda, tanımlama ifadesi, şart ifadesi ve işlem ifadesi yer alabilir. *for* döngülerinin genel yazımı aşağıdaki şekildedir.

```
1   <script>
2     for(tanımlama ifadesi; koşul ifadesi; işlem ifadesi;)
3     {
4       //koşul ifadesi sağlandığı sürece çalıştırılacak kodlar
5     }
6   </script>
```

Döngüler *for* anahtar sözcüğüyle ile başlar. Ardından parantezler içinde ifadeleri içeren kontrol bloğu yazılır. Kontrol bloğundaki ilk ifade, döngünün kontrol değişkenini tanımlama ifadesidir. Bu ifade döngü çalıştırılmaya başlamadan önce bir kez çalıştırılır. Aşağıda verilen kodda, ikinci satırdaki kontrol bloğunda döngünün kontrol değişkeni *i* olarak belirlenmiştir. Tanımlama ifadesinde (var i=0;) bu değişken tanımlanmakta ve sıfır değeri atanmaktadır. Bununla birlikte değişkene istenen herhangi bir değer atanabilir. Şart ifadesinin *true* değerini çevirmesi halinde döngü bir kez çalıştırılır. Aşağıdaki kod bloğunda şart ifadesi (i<10;), i değişkeninin 10'dan küçük olması şeklinde belirlenmiştir.

```
1   <script>
2     for(var i = 0; i<10; i++)
3     {
4       console.log("i değişkeninin değeri şu an:"+i);
5     }
6   </script>
```

Döngünün bir kez çalıştırılmasından sonra, öncelikle işlem ifadesi çalıştırılır. Burada kontrol değişkeni üzerinde işlem gerçekleştirilebilir. Örneğimizde işlem ifadesinde i değişkenine +1 eklenmektedir. Bunun ardından tekrar şart ifadesi kontrolü yapılır. Şart ifadesi *true* değerini çevirdiği sürece bu işlemler tekrarlanır. Döngü içinde işletilecek kodlar kontrol bloğunun ardında küme parantezleri ({ … }) içine yazılır. Örnekte, her dönüşte i değişkeninin değeri tarayıcı konsoluna yazdırılmaktadır.

for döngüleri genellikle bir dizinin taranması için kullanılır. Aşağıda verilen kodda, *for* döngüsü ile meyveler dizisi taranmaktadır. İkinci satırda meyveler dizisi tanımlanmaktadır. Üçüncü satırda meyveler dizisindeki elementleri tarayacak bir *for* döngüsü başlatılmaktadır. Tanımlama ifadesinde kontrol değişkeni (i) tanımlanarak 0 değeri atanmaktadır. Bu sayede dizinin

dikkat

Diziler sıfır indisle başlatıldığından, dizinin son elementine ulaşmak için dizi uzunluğu – 1 formülü kullanılmalıdır. Bu değeri aşan ifadeler hata mesajları çıkaracaktır.

ilk elementinden itibaren taramaya başlanacaktır. İşlem ifadesinde kontrol değişkeninin değeri bir attırılmaktadır. Bu yolla baştan sonra doğru tüm elemanlar taranacaktır. Şart ifadesinde, kontrol değişkeninin değerinin dizi uzunluğundan küçük olması kontrol edilmektedir.

```
1   <script>
2     var meyveler=["elma", "armut", "portakal", "muz"];
3     for(var i=0; i<meyveler.length; i++)
4     {
5       console.log(meyveler[i]);
6     }
7   </script>
```

Öğrenme Çıktısı

8 JavaScript programlama dilindeki döngü yapılarını kurgulayabilme

Araştır 7

Yukarıdaki diziyi son elementten ilk elemente doğru tarayan bir for bloğu oluşturunuz.

İlişkilendir

JavaScript'te kullanılan if/else koşul yapıları ile do/while döngülerini ilişkilendiriniz.

Anlat/Paylaş

JavaScript'te öğrendiğiniz döngü yapılarını Web tasarımlarınızda nasıl kullanabileceğinize ilişkin fikirlerinizi paylaşınız.

FONKSİYONLAR

Fonksiyonlar belirli bir görevi yerine getirmek için hazırlanmış kod bloklarıdır. Fonksiyonlar yalnızca çağırıldıklarında işletilir. JS fonksiyonları iki yolla çağırılabilir. İlk yol programcının fonksiyonu kod ile çağırmasıdır. İkinci yol ise fonksiyonun bir DOM elementine bağlanmasıdır. Bu yollardan ilki bu başlıkta, ikincisi Olaylar başlığında detaylı olarak anlatılacaktır.

Fonksiyonlar adları yazılarak çağırılır. Fonksiyon adının ardından parantez açılır ve kapatılır. Fonksiyona parametre gönderilmek isteniyorsa, parantezler içinden gönderilir. Fonksiyon çağırma işleminin genel yazımı aşağıdaki gibidir.

```
1   fonksiyon_adi(parametre1, parametre2, …);
```

Öncelikle, JS dilinin sağladığı iki hazır fonksiyon ile bu işlemi inceleyelim. Bu hazır fonksiyonlar JS dilinin nesnelerine ait metotlar olduğundan bunlara fonksiyon yerine metot dememiz daha doğru olacaktır. Bununla birlikte, işlevsel ve işletilme biçimleri bakımından metotların fonksiyonlardan bir farkı yoktur. *alert* metodu tarayıcı penceresi yoluyla kullanıcıya uyarı göndermek için kullanılır. Kullanıcıya gösterilmek istenen mesaj, *alert* metoduna parametre olarak gönderilir. Aşağıdaki kodda kullanıcının giriş işleminin başarısız olduğunu bildirmek için alert fonksiyonunun kullanımı örneklenmiştir.

Programlama yapısı olarak metot ve fonksiyon aynı şeyleri ifade eder. Fonksiyonlar ve metotlar aynı yollarla oluşturulur ve çağırılırlar. Fonksiyonlar ve metotlar parametre alabilir ve geriye değer döndürebilirler. Fonksiyon ve metot arasındaki tek fark metotların bir nesneye bağlı olmasıdır. Fonksiyonlar yazılan kodlar içinde tek başlarına çalışabilirler. Burada kullandığımız alert metodu, diğer BOM metotları gibi JS dilinin window nesnesine bağlı olarak çalışmaktadır. Bu nedenle isimlendirme farklılaşsa bile kullanım yöntemlerinde bir fark yoktur.

```
1   <script>
2     alert("Giriş işlemi başarısız oldu!");
3   </script>
```

dikkat

Bir fonksiyon parametre gerektirmiyorsa bile çağırılırken parantezler kullanılır.

DOM içindeki bir elementi id niteliği ile bulmaya yarayan *getElementByID* metodu, *document* nesnesine bağlı çalışmaktadır. Bu metot bulduğu nesnenin referansını geri çevirmektedir. Bu referans bir değişkene atanarak bu değişken üzerinden DOM elementi üzerinde işlem gerçekleştirilebilir. Aşağıdaki kodda, ikinci satırda *id* niteliği *p1* olan element aranmaktadır. Bu element paragraf isimli değişkene atanmaktadır. Üçüncü satırda bu elementin içine "Merhaba" yazılmaktadır.

```
1   <script>
2     var paragraf = document.getElementById("p1");
3     p1.innerHTML = "Merhaba";
4   </script>
```

Hazır JS fonksiyonlarının yanında, programcıların tanımladığı fonksiyonlar da kullanılabilir. Fonksiyon tanımlama genel şeması aşağıda gösterilmiştir.

```
1   <script>
2     function fonksiyon_adı(parametre1, parametre2, parametre3 …)
3     {
4       // işletilecek kodlar
5     }
6   </script>
```

Yeni bir fonksiyon tanımlamak için *function* anahtar sözcüğü kullanılır. Ardından fonksiyonun çağırılması için kullanılacak fonksiyon adı belirtilir. Fonksiyonun çalışması için parametre gerekiyorsa bunlar parantezler içinde virgülle ayrılarak yazılır. Fonksiyon parametre gerektirmiyorsa parantezlerin içi boş bırakılır. Fonksiyon içinde bu parametrelere parantezler içinde verilen isimlerle erişilir. Bunlar fonksiyon içinde tanımlı birer değişken gibi düşünülebilir.

Şimdi, bir fonksiyon tanımlayarak süreci örneklendirelim. Parametre olarak gönderilen değerin karesini alarak tarayıcı konsoluna yazan bir fonksiyon hazırlayalım. Aşağıdaki kodda ikinci satırda *karesini_al* tanımlanmaktadır.

```
1   <script>
2     function karesini_al(girdi)
3     {
4       var karesi;
5       karesi = girdi * girdi;
6       console.log(girdi+" sayısının karesi:"+karesi)
7     }
8     karesini_al(4);
9   </script>
```

Bu fonksiyon girdi isimli bir parametre ile çalışacak şekilde düzenlenmiştir. Fonksiyon kodları parametreye girdi adıyla erişecektir. girdi, fonksiyon içinde çalışan özel bir değişken olarak düşünülebilir. Fonksiyonlar çağırılmadıkları sürece çalıştırılmadıklarından bu kod bloğunda çalıştırılacak ilk kod sekizinci satırdaki *karesini_al*(4); kodudur. Bu kod *karesini_al* fonksiyonunu 4 parametresi ile çağırmaktadır. Bu nedenle kodlarımızda girdi değişkeninin değeri 4 olarak belirlenmiştir. Dördüncü satırda fonksiyonun içindeki karesi değişkeni tanımlanmaktadır. Beşinci satırda bu değişkene girdi değişkeninin karesi alınarak atanmaktadır. Altıncı satırda tarayıcı konsoluna "4 değişkeninin karesi: 16" ifadesi yazdırılmaktadır.

Yedinci satırda küme parantezi kapanarak fonksiyon sonlandırılmıştır. Bu noktadan sonra program akışı, fonksiyonun çağırıldığı ifadenin arkasına, yani dokuzuncu satıra dönecektir.

karesini_al fonksiyonu parametre ile gelen değerin karesini alarak tarayıcı konsoluna yazdırmaktadır. Şimdi bu fonksiyonu değer döndürecek şekilde düzenleyelim. *return* anahtar sözcüğü fonksiyonların değer döndürmesi için kullanılır. Döndürülecek değer *return* anahtar sözcüğünün ardından parantezler içinde yazılır. Aşağıda *karesini_al* fonksiyonu değer döndüren bir fonksiyona çevrilmiştir.

```
1   <script>
2     function karesini_al(girdi)
3     {
4       var sonuc;
5       sonuc = girdi * girdi;
6       return(sonuc);
7     }
8     var karesi=karesini_al(4);
9     console.log("Sonuç:"+karesi);
10  </script>
```

Fonksiyon beşinci satırda girdi ile gelen değerin karesini alarak *sonuc* değişkenine atamaktadır. Altıncı satırda *return* anahtar sözcüğü ile *sonuc* değişkeninin değeri döndürülmektedir. Kod 55'teki kod bloğu sekizinci satırdan itibaren çalışır. 2-7. satırlar arasında fonksiyon tanımlaması olduğundan bu satırlar çağırılmadıkça çalışmaz. Sekizinci satırda karesi değişkeni tanımlanmaktadır. *karesini_al* fonksiyonu 4 parametresi ile çağırılarak döndürülen değer karesi değişkenine atanmaktadır. Bu sayede karesi değişkeninin değeri 16 olarak belirlenmiştir. Dokuzuncu satırda tarayıcı konsoluna "Sonuç: 16" ifadesi yazılmaktadır.

Değişkenlerin Etki Alanı

JS dilinde değişkenlerin erişilebilirliği etki alanları kavramı ile belirlenir. Bir değişken yerel ya da genel (global) etki alanında yer alabilir. Fonksiyonlar içinde tanımlanan değişkenler yerel, fonksiyonlar dışında tanımlanan değişkenler genel değişkenlerdir. Genel bir kural olarak bir değişkenin etki alanı tanımlandığı küme parantezlerinin arasıdır denebilir.

```
1   <script>
2     function karesini_al(deger)
3     {
4       var sonuc;
5       sonuc = deger*deger;
6       console.log(deger+" sayısının karesi:"+sonuc)
7     }
8     karesini_al(4);
9     console.log("Sonuç:"+sonuc);
10  </script>
```

Fonksiyonlarda tanımlı yerel değişkenlere yalnızca fonksiyon içinden erişilebilir. Yukarıdaki kodda, *karesini_al* fonksiyonu içinde tanımlanan *sonuc* değişkeni, bu fonksiyon içinde erişilebilen yerel bir değişkendir. Bu değişkenin amacı parametre olarak gelen *deger* değişkeninin karesini tutmaktır. *sonuc* değişkeni yerel bir değişken olduğundan, etki alanı fonksiyonun çalıştığı üç ve yedinci satırlar arasıdır (bu fonksiyonun küme parantezlerinin arasıdır). *karesini_al* fonksiyonu sekizinci satırda 4 parametresi ile çağırılmaktadır. Bu çağrı ile yerel *sonuc* değişkeni dördüncü satırda tanımlanmaktadır. Beşinci satırda parametrenin karesi hesaplanarak 16 değeri *sonuc* değişkenine atanmaktadır. Altıncı satırda parametre ve yerel değişken kullanılarak tarayıcı

> ✓ Fonksiyonlar içinde tanımlanan yerel değişkenlere fonksiyon dışında erişilemez.

konsoluna “4 sayısının karesi: 16” mesajı yazılmaktadır. Bu mesajla fonksiyonun işletilmesi sonlanmaktadır. Fonksiyonun sonlanmasıyla birlikte tanımlı yerel değişkenlerin tümü silinmektedir. Kodların işletilmesine çağrının hemen ardındaki dokuzuncu satırdan devam edilmektedir. Dokuzuncu satırda *sonuc* değişkeni kullanılarak tarayıcı konsoluna mesaj verilmeye çalışılmaktadır. *karesini_al* fonksiyonu içinde tanımlı yerel bir değişken olan *sonuc* değişkeni fonksiyonun çalışması bitince silinmiş olduğundan, artık erişilebilir bir değişken değildir. Bu nedenle dokuzuncu satır bir referans hatasına yol açacaktır.

Fonksiyonlar dışında tanımlanan değişkenler genel değişkenlerdir. Bu değişkenlere fonksiyonlar da dahil olmak üzere programın herhangi bir yerinden erişilebilir. Aşağıdaki kodda, ikinci satırda sonuc isimli genel değişken tanımlanmaktadır. Yedinci satırda *karesini_al* fonksiyonu 4 parametresi ile çağırılmaktadır. Fonksiyonun içinde beşinci satırda *sonuc* değişkenine erişilerek *deger* parametresinin karesi atanmaktadır. Fonksiyonun bitişi ile birlikte kodların çalıştırılmasına sekizinci satırdan devam edilecektir. Bu satır tarayıcı konsoluna “Sonuç: 16” mesajını yazmaktadır. *sonuc* değişkeni genel tanımlı bir değişken olduğundan, bu değişkene hem fonksiyon içinde hem de fonksiyon dışında erişilebilmektedir.

```
1   <script>
2     var sonuc;
3     function karesini_al(deger)
4     {
5       sonuc = deger * deger;
6     }
7     karesini_al(4);
8     console.log("Sonuç:" + sonuc);
9   </script>
```

Genel etki alanında tanımlanan değişkenlere program kodunun tüm bölümlerinden erişilebilir. Bu değişkenler tarayıcı penceresi açık kaldığı sürece kullanılabilir.

Olaylar

HTML elementleri üzerlerinde bir işlem gerçekleştiğinde bunu bildirmek için olaylar tetiklerler. Örneğin bir düğmeye tıklanması, klavyeden bir tuşa basılması, sayfanın içeriğinin yüklenmesi gibi durumlar elementlerin tetiklediği olaylar tarafından JS'ye bildirilir. JS bu olaylara tepki verecek şekilde programlanabilir. Bu amaçla elementlerin sağladığı olaylara JS fonksiyonları bağlanır. Bu işlemin genel şeması aşağıda görüldüğü şekildedir.

```
1   <etiket olay="fonksiyon_adı()">
```

Örneğin sayfanızdaki bir düğmeye tıklandığında *tiklandi* fonksiyonunu çalıştıralım. Aşağıdaki kodun birinci satırında bir paragraf oluşturulmaktadır. İkinci satırdaki *script* etiketleri ile JS yazım alanı başlatılmaktadır. Üçüncü satırda *tiklandi* fonksiyonu tanımlanmaktadır. tiklandi sayfadaki paragrafı bularak içine “düğmeye tıklandı” yazan basit bir fonksiyondur. Yedinci satırda düğmenin tıklanma (onclick) olayına *tiklandi* fonksiyonu bağlanmaktadır.

dikkat

Elementlerin olaylarına doğrudan JS kodu da yazılabilir. Fakat yazılacak kodlar genellikle birkaç satır süreceğinden fonksiyonların kullanılması daha uygun olacaktır.

```
1   <p id="paragraf"></p>
2   <script>
3     function tiklandi(){
4   document.getElementById("paragraf").innerHTML="düğmeye tıklandı";
5     }
6   </script>
7   <button id="dugme" onclick="tiklandi()">Tıkla</button>
```

Fonksiyonlar HTML nesnelerinin olaylarına JS kullanılarak da bağlanabilir. Bu durumu örneklemek için yukarıdaki örneğimizden devam edelim. JS kullanarak bir elementin olayına fonksiyon bağlamak

için öncelikle bu elemente bir referans oluşturmamız gerekir. Bunun için *getElementById* metodunu kullanabiliriz. Aşağıda verilen kodda, ikinci satırda sayfadaki düğme bulunarak referansı *dugme* değişkenine atanmaktadır. Üçüncü satırda düğmenin *onclick* olayına tiklandi fonksiyonu atanarak bağlama işlemi gerçekleştirilmektedir.

```
1   <p id="paragraf"></p>
2   <button id="dugme1">Tıklayınız</button>
3   <script>
4     function tiklandi (){
5   document.getelementById("paragraf").innerHTML="düğmeye tıklandı";
6   }
7   var dugme=document.getelementById("dugme1");
8   dugme.onclick = tiklandi
9   </script>
```

Her etiket kendine has olaylar tetikleyebilir. Bu olaylardan önemli görülenler Tablo 4.10'da listelenmiştir.

Tablo 4.10 HTML DOM'un Sağladığı Önemli Olaylar

Olay	Açıklama	Kullanım
onchange	HTML elementinin durumunun değiştirilmesi anında tetiklenir.	<select onchange="guncelle()">
onclick	Kullanıcının bir HTML elementine tıklaması sonucu tetiklenir.	<button onlick="gönder()">
onmouseover	Kullanıcının fareyi elementin üzerine getirmesi anında tetiklenir.	<a onmouseover="renk_degistir()">
onmouseout	Kullanıcının fareyi elementin üzerinden çekmesi anında tetiklenir.	<a onmouseout="gizle()">
onkeydown	Kullanıcının klavyeden bir tuşa basması anında tetiklenir.	<input onkeydown="güncelle()">
onload	Tarayıcının sayfa içeriğini yüklemeyi bitirmesi anında tetiklenir.	<body onload="başlat()">

Tarih Nesnesi

JS dilinde tarih işlemlerini gerçekleştirmek için *date* nesnesi kullanılır. *date* nesnesi oluşturmak için new anahtar sözcüğü kullanılır. Bunun ardından nesnenin metotları kullanılarak, tarih nesnesinden istenen bilgiler çekilebilir. Aşağıda verilen kodda, ikinci satırda bir *date* nesnesi oluşturularak tarih değişkenine atanmaktadır. Üçüncü satırda ise *date* nesnesinin *getHours* ve *getMinutes* metotları ile saat ve dakika bilgileri çekilmektedir. Bu bilgiler tarayıcı konsoluna mesaj olarak yazdırılmaktadır.

```
1   <script>
2     var tarih = new Date();
3     console.log(tarih.getHours() + ":" + tarih.getMinutes());
4   </script>
```

date nesnesi parametre verilmeden oluşturulduğunda, kullanıcı bilgisayarının sistem saatinden yararlanılarak güncel tarih bilgisi oluşturur. *date* nesnesi belirli bir zaman noktası verilerek de oluşturulabilir. Bu amaçla aşağıdaki kod kullanılır. Bu formatta İngilizce olarak ay bilgisi (January), gün bilgisi, yıl bilgisi ve saat bilgisi verilir. Aşağıdaki örnekte 12 Ocak 2014 tarihinde, 11:13 saatinde bir nesne oluşturulmaktadır.

```
1   <script>
2     var tarih = new Date("January 12, 2014 11:13:00");
3   </script>
```

date nesnesi oluşturulduktan sonra metotları kullanılarak nesneden bilgi alınabilir. Tablo 4.11'de bu metotlar açıklanarak yukarıda oluşturduğumuz tarih noktası ile (12 Ocak 2014, 11:13:00) çevirecekleri sonuçlar örneklenmiştir.

Tablo 4.11 Tarih Metotları ve Kullanımları

Metot	Açıklama	Kullanım	Sonuç
getDate()	Ayın kaçıncı günü (1-31)	tarih.getDate()	12
getDay()	Haftanın kaçıncı günü (0 Pazar günü olmak üzere 0-6)	tarih.getDay()	0
getFullYear()	Hangi yıl	tarih.getFullYear()	2014
getHours()	Saat kaç	tarih.getHours()	11
getMilliseconds()	Kaçıncı milisaniye	tarih.getMilliseconds()	0
getMinutes()	Kaçıncı dakika	tarih.getMinutes()	13
getMonth()	Hangi ay (0 Ocak olmak üzere 0-11)	tarih.getMonth()	0
getSeconds()	Kaçıncı saniye	tarih.getSeconds()	0
getTime()	1 Ocak 1970, 00:00'dan beri geçen milisaniye sayısı	tarih.getTime()	1389517980000

Zamanlanmış Görevler

JS kodları belirli bir zamanda çalışmak üzere ayarlanabilir. Bu işleme zamanlanmış görev denmektedir. JS iki tür zamanlanmış görev desteklemektedir. İlk zamanlanmış görev verilen kodları bir zaman aralığından sonra (ör: 10sn) işletir. Bu amaçla *setTimeout* fonksiyonu kullanılır. Bu fonksiyona parametre olarak işletilecek kodları içeren bir fonksiyon ve milisaniye cinsinden bekleme süresi verilir. Zamanlanmış görevin kontrol edilebilmesi için bu görev oluşturulurken bir değişkene aktarılabilir. Zamanlanmış görevleri bir senaryo içinde kullanalım. Sitenizde bir kullanıcının giriş yaptıktan sonra, uzun bir dönem hiçbir işlem yapmadığını düşünelim. Bu durumda kullanıcıya bir mesaj göstererek 1 dakika içinde işlem yapmazsa çıkış yapılacağını bildirebilirsiniz. Kullanıcınızın 1 dakika içinde gösterdiğiniz düğmeye tıklaması halinde çıkış işlemini iptal edebilirsiniz.

```
1   <p>Uzun süredir işlem yapmadınız. 60 saniye içinde herhangi bir işlem yapmaz-
sanız hesabınızı korumak için otomatik olarak çıkış yapılacak.</p>
2   <p id="sonuc"></p>
3   <script>
4     var kontrol=setTimeout(cikis, 60000);
5     function cikis(){
6       // çıkış rutinlerini çalıştır
7   document.getElementById("sonuc").innerHTML="Çıkış işlemi gerçekleşti.";
8     }
9   </script>
```

Yukarıda verilen kodda, birinci satırda bir paragrafta kullanıcıya mesaj gösterilmektedir. İkinci satırda işlemlerin sonucunu bildirmek için boş bir paragraf etiketi oluşturulmuştur. Dördüncü satırda *setTimeout* fonksiyonu ile cikis

> Zamanlanmış görevler için milisaniye birimi kullanılmaktadır. 1 saniye 1000 milisaniyeye eşittir.

isimli fonksiyonun 60 saniye sonra çalışması ayarlanmıştır. Daha sonra kontrol edebilmek için zamanlanmış görev kontrol değişkenine atanmıştır. *cikis* fonksiyonu çalışması halinde *id*'si sonuc olan paragrafa "Çıkış işlemi gerçekleşti." mesajını yazmaktadır.

Zamanlanmış görevin iptali için elearTimeout fonksiyonu kullanılır. Önceki örneğimizden devam ederek kullanıcının çıkış işlemini iptal edebilmesi için bir düğme sağlayalım. Aşağıdaki kodda, üçüncü satırda bu işlem için kullanıcıya bir İptal Et düğmesi sağlanmıştır.

dikkat

setTimeout fonksiyonunda parametre olarak verilen fonksiyonun parantezler olmadan çağırıldığına dikkat ediniz. Parantezler verilerek yazılan fonksiyon beklenmeden çağırılacaktır.

```
1   <p>Uzun süredir işlem yapmadınız. 60 saniye içinde herhangi bir işlem yapmazsanız hesabınızı korumak için otomatik olarak çıkış yapılacak. Bu işlemi iptal etmek için aşağıdaki düğmeye tıklayınız.</p>
2   <p id="sonuc"></p>
3   <input type="button" onclick="iptal()" value="İptal Et">
4   <script>
5     var kontrol=setTimeout(cikis, 60000);
6     function iptal(){
7       clearTimeout(kontrol);
8   document.getElementById("sonuc").innerHTML="Çıkma işlemi iptal edildi.";
9     }
10    function cikis(){
11      // çıkış rutinlerini çalıştır
12  document.getElementById("sonuc").innerHTML="Çıkış işlemi gerçekleşti.";
13    }
14  </script>
```

İptal Et düğmesine tıklandığında iptal isimli fonksiyon çağırılmaktadır. Zamanlanmış görev beşinci satırda başlatılarak kontrol değişkenine atanmıştır. Kullanıcı *İptal Et* düğmesine 60 saniye içinde tıklamazsa *cikis* fonksiyonu çağırılarak çıkış işlemleri gerçekleştirilecektir. *İptal Et* düğmesinin *onclick* olayı *iptal* fonksiyonunu çalıştıracak şekilde ayarlanmıştır. *iptal* fonksiyonu içinde yedinci satırda kontrol değişkenine atanmış zamanlanmış görev sonlandırılmaktadır. *clearTimeout* fonksiyonuna parametre olarak zamanlanmış görev kurulurken atama işlemi yapılan değişken gönderilir. Bunun ardından *sonuc id*'li paragrafa "Çıkma işlemi iptal edildi" mesajı yazılmaktadır.

İkinci zamanlanmış görev türü, verilen kodları belirli zaman aralıklarıyla (ör: her saniye) sürekli işletir. Bu amaçla *setInterval* fonksiyonu kullanılır. Bu fonksiyona parametre olarak işletilecek kodları içeren bir fonksiyon ve milisaniye cinsinden zaman aralığı verilir. Yineleme görevinin kontrol edilebilmesi için, görev oluşturulurken bir değişkene aktarılabilir. Önceki örneğimizde kullanıcıya bir düğmeye tıklaması için 60 saniye süre tanımıştık. Bu örneğimizi *setInterval* fonksiyonu ile kullanıcıya düğmeye tıklamak için kaç saniyesi kaldığını bildirecek şekilde geliştirelim. Bu işlem için paragraftaki mesajımızın içine kalan süreyi bildirmek için bir *span* elementi oluşturmamız gerekiyor. Bu elementin içini her saniye yinelenen bir fonksiyon ile değiştireceğiz. Kalan süreyi zaman isimli bir değişken ile tutabiliriz. Aşağıda verilen kodda, üçüncü satırda bu değişken oluşturularak 60 saniye değeri atanmıştır.

```
1   <p>Uzun süredir işlem yapmadınız. <span id="zaman">60</span> saniye içinde herhangi bir işlem yapmazsanız hesabınızı korumak için otomatik olarak çıkış yapılacak. Bu işlemi iptal etmek için aşağıdaki düğmeye tıklayınız.</p>
2   <script>
3     var zaman=60;
4     var kontrol = setTimeout(cikis, zaman*1000);
5     var kontrol2 = setInterval(yenile, 1000);
6     function iptal(){
```

```
7        clearTimeout(kontrol);
8        clearInterval(kontrol2);
9   document.getElementById("sonuc").innerHTML="Çıkma işlemi iptal edildi.";
10    }
11    function cikis(){
12      // çıkış rutinlerini çalıştır
13      clearInterval(kontrol2);
14  document.getElementById("sonuc").innerHTML="Çıkış işlemi gerçekleşti.";
15    }
16    function yenile(){
17      zaman--;
18      document.getElementById("zaman").innerHTML=zaman;
19    }
20  </script>
```

Dördüncü satırda çıkış işlemlerini çağıracak zamanlanmış görev ayarlanmıştır. Beşinci satırda saniyede bir kez yenile fonksiyonunu çağıran ikinci zamanlanmış görev ayarlanmıştır. 16. satırda başlayan yenile fonksiyonu zaman değişkeninden 1 çıkartarak mesaj içindeki span alanının içine yeni değeri yazmaktadır. yenile fonksiyonunu çağıran zamanlanmış görevin senaryomuzdaki iki durumda (iptal ve çıkış) da sonlandırılması gerekmektedir. Kullanıcının 60 saniye içinde İptal Et düğmesine basması halinde, sekizinci satırda kontrol2 değişkenine atanmış zamanlanmış görev *clearInterval* fonksiyonu ile iptal edilmektedir. Aksi halde, çıkış yapılması durumunda cikis fonksiyonu içinde 13. satırda zamanlanmış görevimiz yine iptal edilmektedir.

dikkat

setInterval ile ayarlanmış zamanlanmış görevler sürekli çalıştırıldıklarından uygulama performansına olumsuz etkileri olabilmektedir. Bu nedenle gereksiz durumlarda çalıştırılmaları durdurulmalıdır.

Bu ünitede JS dilinin özellikleri olabildiğince açıklamaya çalıştık. Bununla birlikte JS dilinin daha pek çok özelliği bulunmaktadır. JS günümüz internet teknolojilerinin temel programlama dili konumundadır. Bu çerçevede, okuyucuların yer kısıtlamaları nedeniyle ele alınamayan bu özellikleri incelemeleri önerilmektedir. Okuyucunun kitaplar ve internet kaynaklarını kullanarak bu ünitede oluşturulan temeller üzerine bilgisini geliştirebileceği düşünülmektedir.

internet

Bu örneğin çalışan halini görmek için https://jsfiddle.net/odonmez/muts5eLq/ adresini inceleyebilirsiniz.

Öğrenme Çıktısı

9 JavaScript fonksiyonu oluşturabilme
10 JavaScript ile zamanlanmış görev oluşturabilme

Araştır 8

Sayfaya ekleyeceğiniz bir düğmenin kullanıcı tarafından kaç kez tıklandığını alert uyarı kutusu ile gösteren JS kodunu yazınız.

İlişkilendir

JavaScript'te koşul yapıları, döngüler ve fonksiyonları işlevleri açısından ilişkilendiriniz.

Anlat/Paylaş

Web tasarımlarında JavaScript fonksiyonlarından yararlanılarak çözülebilecek sorunları ve bunların nasıl çözüleceğini anlatınız.

1 JavaScript programlama dilinin önemini açıklayabilme

2 JavaScript programlama dilinin işlevlerini sıralayabilme

JavaScript'in İşlevleri

1 Modern internet teknolojileri düşünüldüğünde JS, HTML ve CSS ile sıkı bir etkileşim içindedir. Bir internet sayfasının içeriği (Ör: metinler, bağlantılar, görseller, videolar) HTML tarafından yapılandırılır. CSS bu yapının görünümünü düzenlemekle görevli katmandır. Sayfanın artalanı, metinlerin gösterileceği yazı tipi, paragraflar aralıkları, kutu kenarlıkları gibi görsel düzenlemeler CSS kullanılarak yapılır. JS ise sayfanın programlanmasını sağlayan mantıksal bileşendir. JS, DOM üzerinden sayfayı, BOM üzerinden tarayıcıyı programlayabilir.

2 JS programlama dilinin iki önemli işlevi bulunmaktadır. JS'nin ilk önemli işlevi sayfanın programlanmasıdır. JS tarayıcılar tarafından oluşturulan DOM'a müdahale ederek şu görevleri yerine getirebilir: elementleri, elementlerin niteliklerini ve CSS stillerini değiştirebilir, sayfaya yeni elementler ekleyebilir, sayfadan element silebilir ve sayfadaki elementlerin tetiklediği tüm olaylara tepki verebilir. JS'nin ikinci önemli işlevi tarayıcının programlanmasıdır. JS tarayıcı kaynaklarına BOM üzerinden ulaşarak yeni bir tarayıcı penceresi açmaktan tarayıcının uyarıcı ve alarm kutularını kullanmaya kadar çeşitli görevleri yerine getirebilir.

3 JavaScript programlama dilinin temel yazım kurallarını açıklayabilme

Yazım Kuralları

3 JS kodları sayfalara script etiketleri arasına ya da .js uzantılı harici dosyalara yazılabilir. Harici dosya kullanılması halinde bu dosyalar script etiketinin src niteliği kullanılarak sayfaya bağlanır. Harici dosyalar içinde script etiketleri kullanılmaz. Sayfa üzerinde ya da harici dosyalar içinde olsun JS kodları noktalı virgül ile sonlandırılır. Noktalı virgüller arasındaki her yönerge kendi içinde bir JS kodu olarak ele alınır, değerlendirilir ve işletilir. Bir yönergede hata bulunması halinde bu hata ile ilgili mesaj tarayıcıların konsol bölmelerinde iletilir. JS yönergeleri değerler (value), değişkenler (variable), işleçler (operator), ifadeler (expression), anahtar sözcükler (keyword) ve yorumlardan (comment) oluşur. Bir yönerge içinde bunların haricinde bir işaret ya da ifade bulunması halinde bu yönerge yazım hatası (syntax error) vererek ardındaki yönergelerin işletilmesini engeller. Yukarıda belirtilen program yapılarının kullanımıyla ilgili detaylar için ilgili başlıklara başvurunuz.

öğrenme çıktıları ve bölüm özeti

4 JavaScript programlama dili ile HTML DOM'da element bulabilme

Dom Nesnelerine Erişmek

4 DOM yönetiminin ilk şartı DOM'da erişilmek istenen elementin bulunmasıdır. JS DOM elementlerine document nesnesinde tanımlı üç farklı fonksiyon ile ulaşabilir. getElementById fonksiyonu elementlere id nitelikleri üzerinden erişim sağlar. getElementsByTagName fonksiyonu elementlere etiket (ör: p, div, span)isimleri üzerinden erişim sağlar. getElementsByClassName fonksiyonu ise elementlere class nitelikleri üzerinden erişim sağlar. getElementById fonksiyonu tek bir element çevirirken, diğer iki fonksiyonun bir element kümesi döndürdüğünü unutmayınız. Erişilmek istenen element dizi tipindeki bu küme içinden bulunmalıdır. Bu fonksiyonların kullanılabilmesi için erişilmek istenen elementin DOM'da tanımlanmış olması gerektiğini unutmayınız. Bu amaçla kodlarınızı elementlerin altındaki satırlara ya da sayfanın yüklenmesi bittikten sonra işletilecek fonksiyonlara yazabilirsiniz.

5 JavaScript programlama dilinin çıktı mekanizmalarını kullanabilme

Çıktı mekanizmaları

5 JS programcılara pek çok çıktı mekanizması sağlar. Bu ünitede bunlar arasında en önemli görülenlerden bahsedilmiştir. Çıktı mekanizmalarının üçü DOM, üçü de BOM üzerinde işlem gerçekleştirir. DOM üzerinde işlem gerçekleştiren mekanizmalar innerHTML, document.write ve appendChild metotlarıdır. innerHTML metodu seçilen DOM elementinin içeriğini verilen parametre ile değiştirir. document.write metodu ise çağırıldığı noktada sayfaya içerik yazmak için kullanılır. Sayfada veri kaybını önlemek için bu iki metodun kullanımında dikkatli olunmalıdır. appendChild metodu ise seçilen DOM elementinin altına yeni bir element eklemek için kullanılır. Bu amaçla eklenecek elementin öncelikle createElement metodu ile oluşturulması gerekmektedir. JS, BOM üzerinde üç çıktı mekanizması sağlamaktadır. console.log daha çok programcıların kullandığı sessiz bir metottur. window.alert ve window.confirm metotları ise doğrudan kullanıcıya yönelik metotlardır. console.log metodu her tarayıcı ile birlikte gelen konsol mekanizmasında mesaj göstermek için kullanılır. Tarayıcı konsolu her tarayıcıda farklı menüler ve tuş kombinasyonları ile açılır. window.alert metodu kullanıcıya uyarı mesajları göstermek için kullanılır. Bu metot ile açılan kutularda sadece Tamam düğmeleri bulunur. window.confirm metodu ise kullanıcıdan bildirim almak için kullanılan bir diyalog kutusudur. Bu kutu üzerlerinde Tamam ve İptal yazan iki düğme içermektedir. Kullanıcının bastığı düğmeye göre bu metot true ya da false değerlerini döndürür.

6 JavaScript programlama dilinde kullanılan veri tiplerini ayırt edebilme

Veri Tipleri

6 JS dili beş farklı veri tipi sağlamaktadır. Bu veri tiplerinden üçü basit, ikisi bileşiktir. String, Number ve Boolean basit veri tipleridir. Bu veri tipleri yalnızca tek bir değer alabilirler. String veri tipi bir karakter dizesidir. Harfler, sayılar ve noktalama işaretleri yan yana birleştirilerek metinler oluşturulabilir. Number veri tipi sayıları tutmak için özelleşmiştir. Pozitif, negatif ve ondalıklı sayıları tutabilir. Boolean veri tipi ise doğru (true) ve yanlış (false) olmak üzere iki değer alabilir. Bu veri tipi daha çok kıyaslama işlemlerinin sonuçlarını ve diyalog kutularının dönütlerini tutmak için kullanılır. Döngülerin kontrol ifadelerinin her zaman Boolean veri tipinde bir çıktı verdiğini unutmayınız. Array ve Object veri tipleri bileşik veri tipleridir. Bu veri tiplerinin içinde birden fazla değer tutulabilir. Array veri tipi daha çok bir değişkenler kümesi tanımlamak için kullanılır. Object veri tipi ise birbirleri ile ilişkilendirilmiş verilerin tutulduğu bir veri tipidir. Örneğin bir kişiyle ilgili tüm bilgiler (ad, soyad, yaş, cinisyet vb.) bir nesne üzerinde tutulabilir. Array veri tipinde kişileri temsil eden nesneler bir küme halinde tutulabilir.

7 JavaScript programlama dilindeki koşul yapılarını kullanabilme

Koşul Yapıları

7 Programlar art arda verilen yönergeler şeklinde yazılır. Programların normal çalışma sürecinde bu yönergeler baştan sonra art arda işletilirler. Bununla birlikte gerçek yaşamdaki pek az sorun bu çalışma mantığı ile çözülebilir. Alınan veriler ve parametrelere göre program yönergelerinin bazılarının çalışması bazılarınınsa çalışmaması gerekebilir. Bunun için koşul yapıları kullanılır. JS dili if, else ve switch olmak üzere üç temel koşul yapısı sağlar. if yapısı verilen bir şart ifadesinin sağlanması halinde çalıştırılacak kod blokları oluşturulmasını sağlar. else yapısı her zaman if yapısı ile birlikte kullanılır. else blokları, if yapısındaki şart ifadesinin sağlanmaması durumunda çalışacak şekilde ayarlanmıştır. switch yapısı ise bir değişkenin alabileceği farklı değerlere tepki vermek için kullanılır.

öğrenme çıktıları ve bölüm özeti

8 JavaScript programlama dilindeki döngü yapılarını kurgulayabilme

Döngüler

8 Döngü yapıları programlarda art arda tekrarlanması gereken rutin kodların otomatikleştirilmesi için kullanılır. JS dili, diğer dillerde çokça kullanılan while ve for döngü yapılarını destekler. Yazılabilecek en basit döngü yapısı while döngüleridir. Bu döngüler verilen şart ifadesi doğru (true) sonuç ürettiği sürece döngüyü çalıştırmaya devam eder. do/while döngüleri temelde while döngüleri olmalarına rağmen, kesinlikle en az bir kez çalışmaları yönüyle while döngülerinden ayrışırlar. Bu iki döngü yapısında, döngü içinde kontrol değişkenine müdahale edilmesi gerekmektedir. Aksi halde şart sürekli doğru değer çevireceğinden sonsuz döngü üretilmiş olur. for döngüleri, while döngülerine göre daha karmaşık bir kontrol bloğu gerektirir. for döngülerinin kontrol bloklarında tanımlama, şart ve işlem ifadeleri yer almaktadır. for döngülerinin farkı kontrol değişkeni ile ilgili tüm işlemlerin döngü kontrol bloğunda gerçekleştirilmesidir.

9 JavaScript fonksiyonu oluşturabilme

10 JavaScript ile zamanlanmış görev oluşturabilme

Fonksiyonlar

9 Program kodunda tekrar tekrar gerçekleştirilmesi gereken rutin işler bir fonksiyon haline getirilerek kodun farklı yerlerinde çalıştırılabilir. Fonksiyonlar function anahtar sözcüğü ile oluşturulur. Bunun ardından fonksiyon adı ve gerekliyse parantezler içinde parametre isimleri verilir. Bu parametreler fonksiyon içinde tanımlı değişkenler gibi görülebilir. Bunun ardından küme parantezleri içinde fonksiyonun kodları yazılır. Hazırlanan fonksiyonlar JS kodları içinde fonksiyon adı kullanılarak çağırılır. Fonksiyon tanımlamasında parametreler kullanılmışsa, fonksiyon çağırılırken parantezler içinde bu parametreler fonksiyona gönderilir. Parametre kullanılmamışsa dahi fonksiyon çağırılırken parantezlerin kullanılması gerektiğini unutmayınız. Fonksiyonlar belirli işlevleri tekrar tekrar gerçekleştirmenin yanında hesaplama sonuçlarını geri çevirebilir. Bu amaçla fonksiyonun kod bloğunda return anahtar sözcüğü kullanılır. Fonksiyonları çağırmanın bir diğer yolu da fonksiyonları HTML elementlerinin olaylarına bağlamaktır. Bu yol kullanıldığında element üzerinde olay (ör: tıklama) her gerçekleştiğinde fonksiyonunuz otomatik olarak çağırılıp çalıştırılır.

10 JS dilinde iki tür zamanlanmış görev oluşturulabilir. Bir zaman aralığından sonra yalnızca bir kez çalışması istenen zamanlanmış görevler setTimeout metodu ile oluşturulur. Belirli bir zaman aralığı ile sürekli tekrarlanması istenen zamanlanmış görevlerse setInterval metodu ile oluşturulur. Zamanlanmış görevleri iptal edebilmek için bir referans gerekmektedir. Zamanlanmış görev oluşturan metodlar, görevleri oluşturduktan sonra kontrol referansları çevirir. Bu referanslar bir değişkenlere atanarak zamanlanmış görevleri iptal etmek için kullanılabilir. Zamanlanmış görevleri iptal etmek için clearTimeout ve clearInterval metodmarı kullanılır. clearTimeout fonksiyonu setTimeout ile oluşturulan zamanlanmış görevleri iptal ederken, clearInterval metodu setInterval ile oluşturulmuş görevleri iptal eder. Görevler oluşturulurken kullanılan değişkenler bu metotlara parametre olarak gönderilerek görevlerin iptali sağlanabilir.

1 Aşağıdakilerden hangisi JavaScript dilinin görevlerinden biri **değildir**?

A. Elementlerin niteliklerini değiştirmek
B. Elementlerin CSS stillerini değiştirmek
C. Elementlerin semantik anlamlarını değiştirmek
D. Yeni bir tarayıcı penceresi açmak
E. Zamanlanmış görev oluşturmak

2 Aşağıdakilerden hangisi JavaScript dilinin tarayıcı penceresini programlama yeteneklerine örnek olarak gösterilebilir?

A. CSS kullanarak bir elementin biçimsel özelliklerini değiştirmek
B. Diyalog kutusu çıkarmak
C. Bir elementin niteliklerini değiştirmek
D. Sayfaya yeni bir liste eklemek
E. Sayfadaki bir tablodan satır silmek

3 JavaScript dilinde bir DOM elementini id niteliğine göre bulmak için aşağıdaki fonksiyonlardan hangisi kullanılır?

A. getElementById
B. setElementById
C. getId
D. setId
E. getElementsByClassName

4 JavaScript dilinde yeni bir değişken tanımlamak için aşağıdaki anahtar sözcüklerden hangisi kullanılır?

A. var
B. new
C. switch
D. if
E. return

5 Aşağıdakilerden hangisi bir döngü yapısıdır?

A. when
B. if
C. switch
D. while
E. array

6 JavaScript dilinde bir HTML elementin içeriğini değiştirmek için kullanılan fonksiyon aşağıdakilerden hangisidir?

A. appendChild B. setText
C. setInterval D. innerHTML
E. console

7 JavaScript dilinde kullanıcının seçimini öğrenmek için diyalog kutusu çıkarmaya yarayan fonksiyon aşağıdakilerden hangisidir?

A. window.alert
B. console.log
C. setInterval
D. setTimeout
E. window.confirm

8 Koşul yapıları ile ilgili aşağıdaki ifadelerden hangisi doğrudur?

A. if yapısı her zaman else yapısı ile birlikte kullanılır
B. else yapısı her zaman if yapısı ile birlikte kullanılır
C. else bloğu if ifadesindeki şartın sağlanması durumunda çalışır
D. if yapısı her zaman switch yapısı içinde kullanılır
E. switch yapısı birden fazla değişkenin durumlarına göre dallanma sağlayabilir

9 x değişkeninin değeri 10 olmak üzere x%=3 ifadesinin sonucu aşağıdakilerden hangisidir?

A. 1 B. 2
C. 0 D. 13
E. 30

10 Yazılan bir fonksiyonun sayfadaki bir elemente tıklanması halinde çalışması isteniyorsa, bu fonksiyon elementin hangi olayına yazılmalıdır?

A. onchange
B. onmouseover
C. onload
D. click
E. onclick

neler öğrendik yanıt anahtarı

1. C	Yanıtınız yanlış ise "JS'in İşlevleri" konusunu yeniden gözden geçiriniz.
2. B	Yanıtınız yanlış ise "JS'in İşlevleri" konusunu yeniden gözden geçiriniz.
3. A	Yanıtınız yanlış ise "DOM Nesnelerine Erişmek" konusunu yeniden gözden geçiriniz.
4. A	Yanıtınız yanlış ise "Değişkenler" konusunu yeniden gözden geçiriniz.
5. D	Yanıtınız yanlış ise "Döngü Yapıları" konusunu yeniden gözden geçiriniz.
6. D	Yanıtınız yanlış ise "Çıktı Mekanizmaları" konusunu yeniden gözden geçiriniz.
7. E	Yanıtınız yanlış ise "Çıktı Mekanizmaları" konusunu yeniden gözden geçiriniz.
8. B	Yanıtınız yanlış ise "Koşul Yapıları" konusunu yeniden gözden geçiriniz.
9. A	Yanıtınız yanlış ise "İşleçler" konusunu yeniden gözden geçiriniz.
10. E	Yanıtınız yanlış ise "Olaylar" konusunu yeniden gözden geçiriniz.

4 Araştır Yanıt Anahtarı

Araştır 1

```
1   <p id="p1"></p>
2   <script>
3   </script>
4   document.getElementById("p1").innerHTML="Bu paragraf JS ile düzenlendi.";
5   <script>
```

Araştır 2

```
sonuc>=62 && sonuc<90
```

Araştır 3

```
1   sayi=document.getElementsByTagName("p");
2   alert(sayi..length);
```

Araştır 4

```
1   paragraflar=document.getElementsByTagName("p");
2   for(var i=0; i<paragraflar.length; i++)
3   {
4   paragraflar[i].innerHTML=(i+1)+". paragraf";
5   }
```

Araştır 5

```
1   var sinif=Array();
2   var ogrenci={ad:"Beril", soyad: "Karasu", yas: 14, siralama: 3};
3   sinif.push(ogrenci);
4   console.log(sinif);
```

4 Araştır Yanıt Anahtarı

Araştır 6

```
1   <script>
2     var bugun=new Date();
3     switch(bugun.getMonth())
4     {
5       case 0: alert("Ocak"); break;
6       case 1: alert("Şubat"); break;
7       case 2: alert("Mart"); break;
8       case 3: alert("Nisan"); break;
9       case 4: alert("Mayıs"); break;
10      case 5: alert("Haziran"); break;
11      case 6: alert("Temmuz"); break;
12      case 7: alert("Ağustos"); break;
13      case 8: alert("Eylül"); break;
14      case 9: alert("Ekim"); break;
15      case 10: alert("Kasım"); break;
16      case 11: alert("Aralık"); break;
17    }
18  </script>
```

Araştır 7

```
1   <script>
2   varmeyveler=["elma","armut","muz","karpuz","kiraz"];
3     for(var i=meyveler.length-1; i>=0; i--)
4     {
5       console.log(meyveler[i]);
6     }
7   </script>
```

Araştır 8

```
1   <button onclick="kontrol()">Tıklayınız</button>
2   <script>
3   sayi=0;
4   function kontrol(){
5   sayi++;
6   alert("Düğmeye "+sayi+". kez tıklıyorsunuz");
7   }
8   </script>
```

Kaynakça

w3schools.com (t.y.). JavaScript tutorial. http://www.w3schools.com/js/default.asp adresinden 31.01.2017 tarihinde erişilmiştir.

Vikipedi (t.y.). JavaScript. https://tr.wikipedia.org/wiki/JavaScript adresinden 29.12.2016 tarihinde erişilmiştir.

Bölüm 5

JS Kütüphaneleri ile Çalışmak

Öğrenme çıktıları

1 Neden JS Kütüphaneleri

1 JS kütüphanesi kullanmanın faydalarını açıklayabilme

2 Önemli JS Kütüphaneleri

2 JS kütüphanelerinin kullanım alanlarını açıklayabilme

3 JQuery

3 JQuery kütüphanesinin kullanım amaçlarını sıralayabilme

4 Sayfalara JQuery ve JQuery mobile kütüphanelerini ekleyebilme

5 Kurallara uygun JQuery kodları yazabilme

4 JQuery Mobile

6 JQuery mobile kullanarak bir kullanıcı arayüzü geliştirebilme

Anahtar Sözcükler: • JS • Kütüphane • JS Kütüphaneleri • DOM • JQuery • JQuery Mobile • Tek Sayfalık Uygulama

GİRİŞ

Javascript (JS) günümüz internet teknolojisinin bel kemiği teknolojilerinden biridir. HTML sayfaların içeriklerini ve semantik dizilimlerini işaretlerken, CSS bu içeriklerin sunumunu düzenleyen katman konumundadır. JS ise bu tablodaki programlama bileşenidir. ASP ve PHP gibi sunucu tabanlı bir teknoloji olmayan JS, çoğu tarayıcı tarafından desteklenen standart bir dildir. JS işlevselliği, kolay öğrenilebilmesi ve esnek yapısı sayesinde son zamanlarda önemli derecede popüler hale gelmiştir. Bu popülaritenin bir nedeni de programcıların ve tasarımcıların işlerini önemli derecede kolaylaştıran, ücretsiz dağıtılan ve sürekli geliştirilen JS kütüphaneleridir. JS kütüphaneleri programcıların ihtiyacı olan temel işlevleri, elementleri ve özellikleri sağlayan, JS temelli uygulama geliştirme sürecini kolaylaştıran ve kısaltan hazır kodlarıdır. JS kütüphanelerinin temel amacı tasarımcıların ve programcıların yazmak zorunda kalacağı rutin kodları en aza indirgeyerek iş akışlarını ve proje geliştirme süreçlerini hızlandırmaktır.

> **JS kütüphaneleri** programcıların ihtiyacı olan temel işlevleri, elementleri ve özellikleri sağlayan, JS temelli uygulama geliştirme sürecini kolaylaştıran ve kısaltan hazır kodlarıdır.

NEDEN JS KÜTÜPHANELERİ

Projelerde JS kütüphanelerini kullanmanın pek çok yararı bulunmaktadır. Bunları; tekrar kullanılabilirlik, ücretsiz yardım kanalları, yapısallık ve genişletilebilirlik olmak üzere dört başlık altında ele almak mümkündür.

Tekrar Kullanılabilirlik

Kütüphane kullanmanın en önemli avantajı, kendini ispat etmiş kodların projeye entegre edilmesidir. Kütüphaneler deneyimli geliştiriciler ve geliştirici takımları tarafından tekrar kullanılabilirlik ilkesi ile üretilir. Üretilen kodlar pek çok projede çalıştırılır ve son kullanıcının hizmetine sunulur. Bu projelerin üretimleri ve proje çıktılarının kullanımı sırasında kütüphane kodları defalarca test edilerek sorunları ortaya çıkarılır. Ortaya çıkan sorunlar geliştiriciler tarafından giderilerek kütüphanenin güvenirliği ve bütünlüğü ileri seviyelere taşınır. Bu süreç sonunda, projeleriniz için mükemmele yakın araçlar oluşturulmuş olur. Geliştiriciler bu araçları kullanarak proje güvenirliğini arttırırken, yazılım geliştirme süresini kısaltır. Örneğin, Nivo Slider neredeyse her internet sitesinde kullanılan manşet (kayan resimler) alanı için özelleşmiş bir kütüphanedir. Bu tür bir aracı oluşturmak için bu kütüphane indirilerek siteye eklenebilir. Yazılım geliştirme projelerinde en önemli giderin zaman olduğu unutulmamalıdır.

Bununla birlikte, tekrar kullanılabilirlik ilkesinin küçük çaplı projeler için sorun yaratması olasıdır. Kütüphanelerin geniş bir özellik yelpazesi sunduğu göz önüne alındığında binlerce satırdan oluşan kodların kullanıcı bilgisayarına gönderilmesi ve bu bilgisayarda çalıştırılması gerekmektedir. Özellikle eski model bilgisayarlar ve mobil cihazlarda kütüphaneler nedeniyle projelerin yavaş yüklenmesi ve çalıştırılması olasıdır. Bu durum, kütüphane yeteneklerinin belki de yüzde beşini kullanan küçük çaplı projeler için ekstra bir yük olarak düşünülebilir. Bunun gibi durumlarda programcıların daha hafif kütüphanelere yönelmesi önerilen bir yaklaşımdır.

Ücretsiz Yardım Kanalları

Kütüphaneler dünya genelinde pek çok yazılımcı tarafından kullanılmakta ve geliştirilmektedir. Kütüphaneler aynı zamanda bu kütüphaneyi kullanan tüm geliştiricilerin ve yazılımcıların desteği ile birlikte gelir. Bu sayede, güvenlik, tasarım, yerelleştirme, tarayıcılar arası farklılıklarla başa çıkma ya da çoklu dil desteği gibi konular için zaman ayırmanıza gerek kalmayacaktır. Bu tür sorunlar hali hazırda diğer geliştiriciler tarafından çözülmüş olarak karşınıza geleceğinden zaman ve para harcamanıza gerek kalmayacaktır. Bunun yanında geliştirici forumlarında sıklıkla karşılaşılan sorunlar ya da özel durumlar ile ilgili pek çok yardım dokümanına ulaşılabilmektedir.

Yapısallık

Kütüphaneler genellikle **MVC** gibi yazılım geliştirme yaklaşımlarını kullanmaktadır. Bu durum projenizin temel kabul görmüş geliştirme yaklaşımları çerçevesinde ilerlemesini sağlayacaktır. Bu sayede,

proje takımına yeni katılan bir geliştiricinin uyum sağlaması daha kısa zaman alacaktır. Bunun yanında, kütüphanelerin benzer geliştirme yaklaşımlarını paylaşmaları öğrenilmelerini kolaylaştırmaktadır.

Genişletilebilirlik

Kütüphaneler sürekli geliştirilmektedir. Her yeni versiyonla birlikte güvenlik güncelleştirmelerinin yanında uygulamalarınıza güç katacak modüller, işlevler ve elementler gelebilmektedir. Örneğin bir e-ticaret uygulaması için kullandığınız kütüphanenin yeni versiyonuyla farklı bir ödeme yöntemine destek vermeye başlayabilirsiniz.

MVC (model-view-controller) bir yazılım geliştirme yaklaşımıdır. Bu yaklaşım veri soyutlanması esasına dayanır. Buna göre, kontrolcü (controller) adı verilen bir ara sistem, veri (model) ve arayüzler (view) arasında köprülük görevi görmektedir. Sistemin avantajı, yazılan kodların tekrar tekrar kullanılabilmesi ve çeşitli uygulamalar için arayüzler (views) sağlayabilmesidir. Örneğin internet sitesi kullanıcıları için görsel bir arayüz oluşturulabilirken, uzaktaki uygulamalarla veri alışverişi için farklı bir arayüz (web servisleri) üretilebilmektedir.

Öğrenme Çıktısı

1 JS kütüphanesi kullanmanın faydalarını açıklayabilme

Araştır 1

Kullandığınız bir JS Kütüphanesinin güncelleme bilgisini alıp gerekli güncellemeleri yapmanız sizlere hangi avantajları sağlar?

İlişkilendir

HTML Web sitesi temaları ile JS kütüphanelerini farklılık ve benzerlikleriyle birlikte karşılaştırınız.

Anlat/Paylaş

JS kütüphanelerinin yarar ve sınırlılıklarını tartışınız.

ÖNEMLİ JS KÜTÜPHANELERİ

JS dilinin işlevselliği ve esnekliği sayesinde programcılar herhangi bir amaca yönelik kütüphaneler üretebilmektedir. Bu kütüphanelerin amaçları tarayıcı özelliklerini keşfetmekten, araç fonksiyonları sağlamaya; HTML DOM yönetiminden, bütünüyle bir web uygulaması üretmeye kadar uzanabilmektedir. Bu ünitede yer kısıtları nedeniyle en çok kullanılan JS kütüphanelerinden JQuery ve JQuery Mobile tanıtılacaktır. Bununla birlikte onlarca faydalı kütüphaneden söz edilebilir. Bu bölümde okuyucunun ilgisini çekebilecek bazı kütüphanelerden kısaca bahsedilerek erişim adresleri sunulmuştur.

AngularJS

Google tarafından desteklenen açık kaynak kodlu bir uygulama geliştirme kütüphanesidir. AngularJS veriye dayalı tek sayfalık karmaşık uygulamaların üretilmesi için gerekli tüm araçları sağlamaktadır. AngularJS günümüzde pek çok büyük firmanın internet araçlarında kullanılmaktadır. 2014 yılında ilan edilen ve günümüzde kullanılan AngularJS 2 versiyonuyla JS'in geliştirilmiş bir versiyonu olan TypeScript ile programlanmaya başlanmıştır. AngularJS MVC yazılım geliştirme yaklaşımını temel almaktadır. Modeller ve arayüzler arasında iki yönlü veri bağlantısı

Programlama literatüründe kullanıcının gördüğü arayüzlere ön yüz **(front-end)** denmektedir. Bunun yanında, uygulamaların arkasında çalışan veri tabanı erişimi, hesaplama rutinleri gibi işlevlere ise arka yüz (back-end) adı verilmektedir. Büyük ölçekli uygulamaların ön yüz ve arka yüz geliştirici takımları işbirliği içinde çalışmaktadır.

oluşturabilmektedir. Bu sayede verideki değişimler iki yönde anında güncellenebilmektedir. AngularJS başka hiçbir kütüphaneye gerek duymadan büyük çaplı uygulamaların önyüzlerini **(front-end)** geliştirmenize izin verebilir. Daha fazla bilgi için kütüphanenin resmi internet sitesi https://angularjs.org ziyaret edilebilir. Çok popüler olan bu kütüphane ile ilgili Türkçe pek çok kaynağa ulaşılabilmektedir.

React

Büyük çoğunluğu Facebook tarafından geliştirilen React, açık kaynak kodlu bir ön yüz geliştirme aracıdır. React daha çok MVC modelindeki V parçasına yani kullanıcı arayüzü elementlerine odaklanan bir kütüphanedir. Büyük çaplı tek sayfalık uygulamalar üretmek için ideal bir kütüphanedir. En önemli avantajı HTML DOM mimarisine oranla çok daha hızlı çalışan bir sanal DOM üretmesidir (HTML DOM kavramı JQuery başlığı altında detaylı bir şekilde incelenecektir). Sanal DOM bir JS nesnesi olduğundan, bu nesne üzerinde işlem yapmak HTML DOM nesnelerine erişerek işlem yapmaya oranla oldukça hızlıdır. Verideki herhangi bir değişimde sanal DOM bu değişiklikleri otomatik olarak arayüze aktarmaktadır. React, JS içinde kullanılabilecek XML'e benzer özelliklere sahip JSX adında bir sentaks kullanmaktadır. Bu sayede React kodlarının anlaşılması HTML ya da XML bilen bir tasarımcı için oldukça kolaydır. Daha fazla bilgi için kütüphanenin resmi internet sitesi https://facebook.github.io/react/ ziyaret edilebilir. Bu kütüphane ile ilgili pek çok Türkçe kaynağa internetten ulaşılabilmektedir.

> AngularJS ve React'e benzer işlevler sağlayan ve yaygın kullanımdaki diğer kütüphaneler arasında Ember.js (http://emberjs.com/), Backbone.js (http://backbonejs.org/) ve Knockout (http://knockoutjs.com/) sayılabilir.

Underscore

UnderscoreJS programcıların çoğu projede tekrar tekrar yazmak zorunda kaldıkları genel amaçlı araç fonksiyonlarını sağlayan bir kütüphanedir. Bu kütüphanede nesneler, diziler, fonksiyonlar ve yığınlar üzerinde işlem yapabilen pek çok metot sağlanmaktadır. Örneğin, kütüphanenin sağladığı first metodu, bir dizideki ilk n elementi çevirmek için özelleşmiştir. Bu metot _.first(dizi, [n]) sentaksıyla çağırılmaktadır. Buradaki n değeri, diziden ilk kaç elementin çekileceğini belirler. Örneğin _.first(meyveler, 3) ifadesi, meyveler dizisindeki ilk üç elementi çevirmektedir. Kütüphane bunun gibi pek çok metot sağlamaktadır. Daha fazla bilgi ve metotların referansı için kütüphanenin resmi internet sitesi http://underscorejs.org adresi ziyaret edilebilir.

D3.js

Data-Driven-Documents (veri yönetimli belgeler), belgeleri bir veri setine göre düzenlemek için kullanılan bir kütüphanedir. D3.js veriden elde edilen bilgileri kullanarak HTML, SVG ve CSS teknolojileri üzerinde işlem gerçekleştirebilmektedir. Bu kütüphane kullanılarak veriyi temsil eden etkileşimli grafikler ve görseller hazırlanabilmektedir. Bu grafikleri incelemek ve kütüphane hakkında daha fazla bilgi edinmek için resmi internet sitesi https://d3js.org adresini ziyaret edebilirsiniz.

BabylonJS

BabylonJS modern web standartlarını kullanarak, güncel tarayıcılarda çalıştırılabilen üç boyutlu oyunlar üretmek için araçlar sağlayan bir kütüphanedir. Kütüphane WebGL teknolojisini kullanabilmektedir. Fizik motoru, çarpışma denetimi, animasyon motoru, ses ve parçacık sistemleri gibi oyun oluşturmak için gerekli tüm teknolojiler bu kütüphane tarafından sağlanabilmektedir. Kütüphane ile gerçekleştirilmiş projeleri ve kütüphanenin yeteneklerini incelemek için resmi web sitesi http://www.babylonjs.com adresini ziyaret edebilirsiniz.

Bootstrap

Bootstrap, internet siteleri ve internet tabanlı uygulamaların kullanıcı arayüzlerini oluşturmak için kullanılabilecek bir kütüphanedir. Bootstrap öncelikle akıllı telefon, tablet bilgisayar gibi mobil cihazları hedef alan bir felsefe ile geliştirilmiştir. Bootstrap ile üretilen sayfalar ve uygulamalar, mobil cihazların internet tarayıcılarında sorunsuz bir şekilde çalışmakta ve görüntülenebilmektedir. Oldukça çeşitli ekran boyutları ve piksel yoğunlukları olan mobil cihazlarda içeriklerin okunabilir ve düzgün gösterilebilmesi için metin (tipografi), yerleşim ve ölçekleme özellikleri sunulmaktadır.

Bunun yanında kütüphane günümüz internet teknolojilerinde sıklıkla kullanılan pek çok elementi (menü çubukları, manşetler, sekmeler, listeler, tablolar vb.) hazır olarak sunmaktadır. Bootstrap hakkında daha fazla bilgi edinmek için http://trbootstrap.com adresini ziyaret edebilirsiniz. Bu sayfanın orijinali İngilizce dilinde http://getbootstrap.com adresinde yayınlanmaktadır.

Öğrenme Çıktısı

2 JS kütüphanelerinin kullanım alanlarını açıklayabilme

Araştır 2

Bölümde verilen JS kütüphanelerinin dışında yaygın kullanılan diğer JS kütüphanelerini araştırınız.

İlişkilendir

Kullandığınız alışveriş sitelerinin hangi JS kütüphanesi ile oluşturulmuş olabileceğini düşününüz.

Anlat/Paylaş

Daha önce deneyim veya bilgi sahibi olduğunuz JS kütüphaneleriyle ilgili görüşlerinizi paylaşınız.

JQUERY

JQuery, internet programcılığını kolaylaştıran bir JS kütüphanesidir. JQuery'nin temel amacı, JS ile gerçekleştirilen genel programlama görevlerinin, daha az kod ile gerçekleştirilmesidir. Platformun sloganı "daha az kod ile daha fazla üretimdir" [write less, do more] (Resim 5.1).

Bu durumu bir örnekle açıklayalım. Sayfamızdaki bir liste elementine yeni satır eklemek için uzun JS kodları yazmamız gerekir. Aşağıda verilen kodun ikinci satırında bu amaçla bir li elementi üretilmektedir. Üçüncü satırda üretilen elementin içeriği innerHTML metodu ile belirlenmektedir. Dördüncü satırda ise oluşturulan elementin ekleneceği liste sayfada bulunarak bir referans değişkenine (liste) aktarılmaktadır. Beşinci satırda bu değişken kullanılarak oluşturulan li elementi listeye eklenmektedir. Görüldüğü gibi bunun gibi küçük bir işlem için bile pek çok JS metodu art arda kullanılmak zorundadır.

Resim 5.1 JQuery Sisteminin Logosu

```
1   <script>
2     var yeni=document.createElement("li");
3     yeni.innerHTML="Ayva";
4     var liste=document.getElementById("meyveler");
5     liste.appendChild(yeni);
6   </script>
```

Aşağıda verilen kodda ise bu işlem JQuery ile tek satırda gerçekleştirilmektedir. İkinci satırdaki JQuery'nin append metodu liste elementini otomatik olarak üretip, sayfada id değeri meyveler olan liste elementine eklemektedir.

```
1   <script>
2     $("#meyveler").append("<li>Muz</li>");
3   </script>
```

JQuery'nin temel amacı internet programcılığını kolaylaştırmak ve hızlandırmak olduğundan, öğrenilebilmesi için temel düzeyde HTML, CSS ve JS bilgisi gerektirir. JQuery'nin en temel işlevleri HTML DOM üzerinde gerçekleştirilmektedir. Bunun yanında JQuery, elementlerin CSS özelliklerini düzenleyebilir. JQuery'nin temel işlevleri aşağıdaki şekilde listelenebilir:

1. HTML/DOM düzenleme: Sayfadaki HTML DOM üzerinde ekleme, silme ve düzenleme işlemleri gerçekleştirilebilir.
2. CSS düzenleme: Sayfadaki elementlerin CSS stilleri üzerinde düzenleme yapılabilir.
3. HTML olaylarına tepki verme: HTML'nin sağladığı tüm olaylara tepki verecek fonksiyonlar hazırlanabilir.
4. Görsel efekt ve animasyonlar: Bir elementi soluklaştırarak silme, bulunduğu ortamda sağa sola hareket ettirme gibi pek çok hazır görsel efekt ve animasyon sağlanmaktadır.
5. AJAX: Sayfayı yenilemeden arka planda bir sunucudan veri çekme, sunucuya veri gönderme ve alınan sonuçlara göre sayfa içeriğini düzenleme işlemleri yapılabilir.

İnternet programcılığındaki en önemli sorunlardan biri yazılan kodların tarayıcılar tarafından standart bir şekilde yorumlanmamasıdır. CSS kullanarak Chrome tarayıcısında sağlanan animasyonlar ve görsel düzenleme, Internet Explorer tarayıcısında elde edilemeyebilir. JQuery'nin en güçlü yanlarından biri tarayıcılar arasındaki bu farklılaşmaları aşabilmesidir.

dikkat

Programlama dillerinde fonksiyon ve metot kavramları birbirlerinden farklıdır. Fonksiyonlar bir program bağlamında kendi başlarına çalışan yapılardır. Parametre alıp değer döndürebilirler. Metotlar ise bir nesneye bağlı olarak çalışan fonksiyonlar gibi düşünülebilir. Örneğin burada ele aldığımız createElement metodu document nesnesi içinde tanımlı bir metottur. Buradan sonra konu edeceğimiz çoğu yapı JQuery nesnesinde tanımlı olduğundan fonksiyon yerine metot terimi kullanılacaktır. Bunun yanında function anahtar kelimesini kullanarak olaylara tepki vermek için yazacağımız kodları fonksiyon terimi ile ifade etmek daha yerindedir.

✓ JQuery ile oluşturulan animasyon ve efektler tüm tarayıcılarda aynı şekilde çalışabilir.

JQuery hakkında daha fazla bilgi edinmek için https://jquery.com adresini ziyaret ediniz.

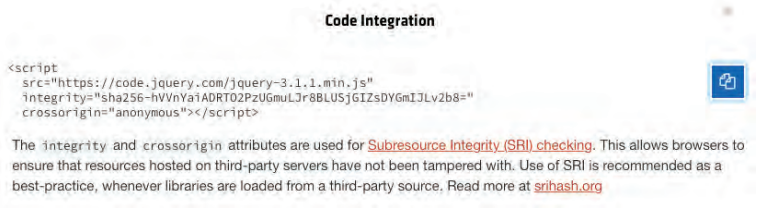
Sayfalara JQuery Eklemek

Bir JS Kütüphanesi olan JQuery, sayfaya herhangi bir JS dosyası gibi eklenir. JS belgeleri projelere yerel ya da uzak yöntemlerle eklenebilir. JQuery'i projede yerel kullanmak için http://jquery.com/download/ adresinden indirebilirsiniz. Bu adreste JQuery'nin en güncel sürümlerini bulabilirsiniz. Bu ünitenin yazıldığı dönemde JQuery'nin en güncel sürümü 3.1.1'dir. Bu sayfada pek çok indirme seçeneği sunulmuştur. Sayfada JQuery'nin tam ve özellikleri azaltılmış (slim) sürümlerini indirmek için bağlantılar sağlanmaktadır. JQuery kodlarını okumak istiyorsanız geliştirme (development), istemiyorsanız geliştirme (production) sürümünü indirebilirsiniz. JQuery'i sayfaya eklemenin ikinci yolu uzak yöntem olarak nitelendirilebilecek **İçerik Dağıtım Ağları**'ndan (İDA) yararlanmaktır. JQuery popüler bir kütüphane olduğundan pek çok İDA üzerinden sunulmaktadır. Bu amaçla https://code.jquery.com adre-

✓ **İçerik Dağıtım Ağları** (Content Delivery Network [CDN]), JS kütüphaneleri gibi internet teknolojilerinde genel kullanımdaki dosyaların paylaşılması için kullanılan sunucu ağladır. Bu ağların sunucuları internet üzerindeki birçok veri merkezinde dağınık bir şekilde dağıtılmış durumdadır. Bu sayede kullanıcılara hızlı ve güvenilir bir servis sağlanabilmektedir.

sindeki bağlantıları kullanabilirsiniz. JQuery aynı zamanda Google ve Microsoft firmaları tarafından da sağlanmaktadır.

code.jquery.com adresinde bir sürüm bağlantısına tıkladığınızda karşınıza Resim 5.2'deki diyalog kutusu çıkacaktır. Bu diyalogdaki mavi düğmeye tıklayarak sayfanıza yapıştırmanız gereken kodları kopyalayabilirsiniz. Sayfanızda JQuery kullanabilmek için kopyaladığınız kodları HTML belgenize yapıştırmanız yeterlidir. Sayfaların yüklenme performanslarını düşürmemek için JQuery gibi kütüphaneler genellikle *body* etiketinin kapatılmasından sonra sayfaya eklenir.



Resim 5.2 JQuery Bağlantısını Kopyalayabileceğiniz Diyalog Kutusu

JQuery Yazım Kuralları

JQuery yazım kuralları HTML elementlerini seçmek ve düzenlemek için özelleşmiştir. Temel yazım biçimi aşağıda gösterildiği şekildedir.

```
1   <script>
2     $(seçici).eylem();
3   </script>
```

Bu yazımda $ simgesi JQuery kütüphanesine erişmek için kullanılır. Ardından parantezler içinde bir seçici ifadesi yer alır. Seçici ifadeleri CSS seçiciler ile aynıdır. Parantezlerin ardından nokta konarak eylemi gerçekleştirecek metot çağırılır. Aşağıdaki kodun ikinci satırında $("p") ifadesi ile sayfadaki tüm paragraf elementleri seçilerek üzerlerine *hide* metodu uygulanmaktadır. hide metodu elementleri gizlemek için kullanılmaktadır. Üçüncü satırda sınıfı (class) sakla olan tüm paragraf elementleri seçilerek saklanmaktadır. Dördüncü satırda ise id özelliği sakla olan p elementi seçilerek saklanmaktadır.

```
1   <script>
2     $("p").hide();
3     $("p.sakla").hide();
4     $("p#sakla").hide();
5   </script>
```

JQuery Olayları

Kullanıcıların sayfa üzerindeki hareketlerine tepki vermek için HTML DOM tarafından yayınlanan olaylar için fonksiyonlar yazılır. HTML DOM kullanıcının bir düğmeye tıklaması, fareyi hareket ettirmesi, bir bağlantının üzerine gelmesi gibi eylemleri için olaylar yayınlar. Olaylara tepki vermek için kullanılması gereken sentaks aşağıdaki kodda gösterilmiştir.

```
1   <script>
2     $(seçici).olay( … tepki veren kodlar …);
3   </script>
```

Öncelikle $ ile JQuery çağırılır. Bunun ardından parantezler içinde olaylarına tepki verilecek element/elementler seçilir. Ardından nokta ile tepki verilecek olay adı yazılır. Ardından parantezler içinde bu olaya

tepki verecek kodlar yazılır. Parantez kapandıktan sonra noktalı virgül ile işlem sonlandırılır. Aşağıda verilen kodda bir düğmeye tıklanması halinde (click olayında) tarayıcının uyarı penceresini çağıran JQurey kodları görülmektedir.

```
1   <script>
2     $("button").click(function(){
3         alert("Bir düğmeye tıklandı.");
4       }
5     );
6   </script>
```

dikkat

Bu bölümdeki tüm örneklerin çalışan kodlarına https://jsfiddle.net/user/odonmez/fiddle/ adresinden ulaşabilirsiniz.

Seçici button olarak belirlendiğinden yazılan kodlar sayfadaki tüm düğme elementleri için geçerlidir. click ifadesi ile sayfadaki tüm button elementlerinin tıklanma olaylarına tepki verecek kodlar yazıldığı belirtilmiştir. JQuery olaylarına tepki verecek kodlar parantezler içinde bir fonksiyon olarak yazılır. Bu amaçla function anahtarı kullanılarak bir fonksiyon üretilmiştir. Üçüncü satırda bu fonksiyonun içine yerleştirilen alert metodu çağırılmaktadır. Bu sayede sayfadaki tüm button elementlerine tıklandığında çağırılacak genel bir fonksiyon üretilmiştir. Yazılan kod düğmenin tıklandığını belirtmek için tarayıcının uyarı penceresini çıkarmaktadır. Projelerde sıklıkla kullanılan HTML DOM olayları Tablo 5.1'de listelenmiştir.

Tablo 5.1 JQuery Projelerinde Sıklıkla Kullanılan Olaylar

Olay	Açıklama	Örnek
Fare Olayları		
click	Fare ile tıklama	$("dugme").click(…);
dblclick	Fare ile çift tıklama	$("p").dblclick(…);
mouseenter	Farenin elementin üzerine gelmesi	$("h1").mouseenter(…);
mouseleave	Farenin elementin üzerinden ayrılması	$("h3").mouseleave(…);
Klavye Olayları		
keypress	Kullanıcının klavyeden bir tuşa basarak elini çekmesi	$("input").keypress (…);
keydown	Kullanıcının klavye tuşuna basması	$("textarea").keydown(…);
keyup	Kullanıcının klavye tuşundan elini çekmesi	$("input").keyup(…);
Form Olayları		
submit	Formun submit düğmesi ile gönderilmesi	$("#form1").submit(…);
change	Bir elementin değerinin değişmesi	$("input#adsoyad").change(…);
focus	Bir elementin klavye odağına alınması	$("#form1 input").focus(…);
blur	Klavye odağının elementten ayrılması	$("#adres").blur(…);
Pencere Olayları		
load	Bir elementin tüm alt elementlerinin yüklenmesinin tamamlanması	$(window).load(…);
resize	Tarayıcı penceresinin boyutlandırılması	$(window).resize(…);
scroll	Pencere içeriğinin kaydırılması	$(window).scroll(…);
unload	Pencerenin farklı bir internet adresine gitmesi	$(window).unload(…);

JQuery kodları, script etiketleri içinde olmak kaydıyla, sayfanın herhangi bir yerinde yazılabilir. Bununla birlikte henüz yüklenmemiş ya da HTML DOM'a eklenmemiş elementler için yazılan kodlar hata verecektir. Bu tür hataları engellemek için sayfayı ifade eden document seçicisinin ready olayı kullanılabilir. Bu

olay HTML DOM üretildiğinde yayınlandığından, yazılan kodlar bu tür hatalar vermeyecektir. Aşağıdaki kodda bu olayın kullanımı örneklenmiştir. İkinci satırda document nesnesinin ready olayı seçilerek içine kod yazmak üzere bir fonksiyon başlatılmıştır. Bu fonksiyon içine yazılan kodlar hata vermeyecektir.

```
1   <script>
2     $(document).ready(function(){
3         $("button").click(function(){
4             alert("Düğme tıklandı.");
5           });
6       }
7     );
8   </script>
```

JQuery tüm HTML DOM olaylarına tepki verebilecek şekilde tasarlanmıştır. Tablo 5.1'de bu olayların küçük bir alt kümesi sunulmuştur. Tüm olayları inceleyebilmek için JQuery referans sayfasını kullanabilirsiniz: https://api.jquery.com/category/events/

JQuery kodları yazılırken iç içe pek çok parantez ve küme parantezi kullanılabilmektedir. Bu konuda hata yapmamak için parantezleri açtığınız gibi kapatınız. Gerekli kodları daha sonra parantezler arasına tıklayarak yazabilirsiniz.

JQuery Efektleri

JQuery kullanarak HTML elementlerine özel efektler uygulanabilir. Bu amaçla seçicinin arkasına efekt metodu bağlanır. Bu efektler JQuery tarafından sağlandığından tüm tarayıcılarda desteklenir ve aynı görsel etkileri yaratır. Efektlerin kullanımı için sentaks aşağıdaki kodda verilmiştir.

```
1   <script>
2     $(seçici).efekt(uygulanma süresi, uygulama_fonksiyonu);
3   </script>
```

Bir efekti kullanmak için öncelikle uygulanacak element JQuery seçici yöntemiyle (ör: $('#birinci')) seçilir. Ardından efekt metodu seçilir (ör: hide). Metot için açılacak parantezler içinde verilen ilk parametre milisaniye cinsinden efektin uygulanma süresidir. İkinci parametre efektin uygulanmasının ardından çağırılacak uygulama fonksiyonudur. Burada function anahtar kelimesi kullanılarak yeni bir fonksiyon yazılabilir. Bu fonksiyonun yazılması zorunlu değildir.

```
1   <script>
2     $("#birinci").hide(1000, function(){
3       alert("paragraf gizlendi.");
4       }
5     );
6   </script>
```

Yukarıda verilen kodda *id* değeri birinci olan *p* elementi 1 saniyede gizlenmekte, ardından çalıştırılan uygulama fonksiyonu ile kullanıcıya bilgi notu gösterilmektedir. Uygulama fonksiyonları JQuery'nin asenkron çalışma yöntemidir. JQuery, uygulama fonksiyonu içindeki kodları efekt metodunn işletilmesi bitene kadar çalıştırmaz. JQuery'nin sağladığı efektler Tablo 5.2'de listelenmiştir.

✓ Milisaniye, saniyenin binde biridir (1sn=1000ms)

Tablo 5.2 JQuery'nin Desteklediği Efektler

Efekt	Açıklama	Örnek
hide	Elementin genişliği, yüksekliği ve saydamlığını değiştirerek görünmez yapar.	$("p.ilk").hide(1000);
show	Elementin genişliği, yüksekliği ve saydamlığını değiştirerek görünür yapar.	$("p.son").show(1000);
toggle	Nesnenin görünürlüğünü tersine çevirir. Görünmeyen nesneleri gösterir, görünen nesneleri saklar.	$("p.ikinci").toggle(500);
fadeIn	Elementin saydamlık özelliğini değiştirerek görünür yapar.	$("a#gonder").fadeIn(200);
fadeOut	Elementin saydamlık özelliğini değiştirerek görünmez yapar.	$("a#ilk").fadeOut(1000);
fadeToggle	Saydamlık efektleri (fadeIn/fadeOut) ile düzenlenmiş element görünürlüğünü tersine çevirir.	$("button").fadeToggle(500);
fadeTo	Elementin saydamlığını 0-1 aralığındaki bir değere getirir. Saydamlığı 0.2 olan bir element %20 görünürdür.	$("p").fadeTo(500, .3);
slideUp	Elementin yüksekliğini değiştirerek yukarı kayma efektiyle görünmez yapar.	$("div").slideUp(300);
slideDown	Elementin yüksekliğini değiştirerek aşağı açılma efektiyle görünür yapar.	$("article").slideDown(200);
slideToggle	Kaydırma efektleri (slideUp/slideDown) ile düzenlenmiş element görünürlüğünü tersine çevirir.	$("p").slideToggle(500);
animate	Elementin CSS özelliklerini verilen süre içinde değiştirir. Anime edilecek özellikler küme parantezleri içinde virgüllerle ayrılarak yazılır. Özelliğin yeni değeri üstüste iki noktadan sonra tırnaklar içinde yazılır.	$("div").animate({left:'300px', opacity:'.3'}, 500);

JQuery efektleri zincirleme çalıştırılabilir. Aşağıda verilen kodda slideUp ve slideDown metotlarının artarda çalıştırılması gösterilmektedir. Bu amaçla, ilk efekt metodunun bitirildiği noktada ikinci efekt metodu nokta ile bağlanarak çağırılır.

```
1   <script>
2     $("#p1").click(function(){
3         $("#p1").slideUp(200, function(){
4         $(this).html("tıklandı");
5       }).slideDown(200);
6       }
7     );
8   </script>
```

Kodda, üçüncü satırda *slideUp* metodu çağırılmaktadır. *slideUp* metodunun çağırıldığı satırda paragrafın içeriğini değiştiren bir uygulama fonksiyonu çağırılmaktadır. Bu nedenle *slideUp* metodunun kapandığı parantez beşinci satırda yer almaktadır. Bu parantezin hemen ardından *slideDown* metodu çağırılarak efektlerin zincirleme çalıştırılması sağlanmıştır.

JQuery DOM Metotları

JQuery DOM elementlerinin içerikleri ve değerleri ile ilgili pek çok metot sağlamaktadır. Bu özellikler, DOM düzenleme, elementlerdeki verileri ve özellikleri çekme, elementlerdeki verileri ve özellikleri düzenleme olmak üzere üç başlık altında ele alınabilir. JQuery, DOM'a element eklemek için dört metot sağlamaktadır. Bu metotlar, açıklamaları ve örnek kullanımları Tablo 5.3'te sunulmuştur.

> JQuery'nin en önemli DOM metotları; DOM düzenleme, elementlerdeki verileri ve özellikleri çekme, elementlerdeki verileri ve özellikleri düzenlemedir.

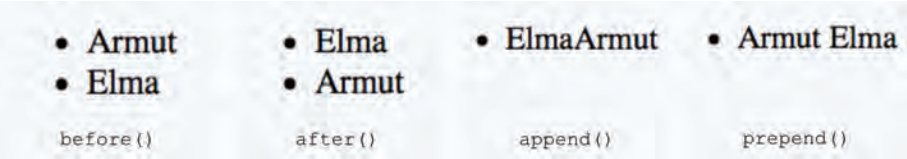
Tablo 5.3 DOM'a Veri Ekleme Metotları

Metot	Açıklama	Örnek
append()	İçeriği seçilen elementin sonuna ekler.	$("#ilk").append(" Armut");
prepend()	İçeriği seçilen elementin başına ekler.	$("#ilk").prepend("Armut ");
after()	İçeriği seçilen elementten sonra ekler.	$("#ilk").after(" Armut ");
before()	İçeriği seçilen elementten önce ekler.	$("#ilk").before(" Armut ");

Aşağıda verilen kodda sırasız bir liste elementi için tek bir eleman (Elma) bulunmaktadır.

```
1   <ul>
2     <li id="ilk">Elma</li>
3   </ul>
```

Tablo 5.3'te verilen örnek kodlarda bu kod bloğundaki Elma elementine veri ekleme metotları uygulandığında gerçekleşecek sonuçlar Resim 5.3'te sunulmuştur. before ve after metotları ile elementin önüne ve arkasına yeni elementler eklenirken, append ve prepend ile elementin içine metin eklenmiştir.



Resim 5.3 Ekleme Metotlarının Sonuçları

JQuery DOM elementlerinin silinmesi için empty ve remove metotlarını sağlamaktadır. İlgili örnekler Tablo 5.4'te verilmiştir.

Tablo 5.4 DOM'dan Silme Metotları

Metot	Açıklama	Örnek
remove()	Seçilen elementi DOM'dan siler	$("#elma").remove();
empty()	Seçilen elementin içindeki elementleri DOM'dan siler.	$("#meyveler").empty();

Aşağıda verilen kodda sırasız bir liste elementi içinde dört adet eleman listelenmiştir.

```
1   <ul id="meyveler">
2     <li id="elma">Elma</li>
3     <li>Armut</li>
4     <li>Portakal</li>
5     <li>Muz</li>
6   </ul>
```

Bu kod bloğunda, *$("#elma").remove()* kodunun çalıştırılması sonucunda listeden sadece *Elma* meyvesi silinir. *$("#meyveler").empty()* kodu ise listedeki tüm meyveleri silecektir.

JQuery, elementler üzerindeki verileri çekmek ve düzenlemek için dört metot sağlamaktadır. Bu metotlar, açıklamaları ve örnek kullanımları Tablo 5.5'te verilmiştir.

Tablo 5.5 JQuery İçerik Düzenleme ve Erişim Metotları

Metot	Açıklama	Örnek
text()	Seçilen elementin içindeki metni çeker ya da düzenler.	$("#p1").text("Birinci paragraf"); $a=$("#p1").text();
html()	Seçilen elementin içeriğini HTML etiketleri ile birlikte çeker ya da düzenler.	$("#p1").html("<a>tıkla"); $a=$("#p1").html();
val()	Seçilen girdi alanının değerini çeker ya da düzenler.	$("#isim").val("Onur"); $isim=$("#isim").val();
attr()	Seçilen elementin özelliklerini çeker ya da düzenler.	$("#link").attr("href","index2.html"); $baglanti=$("#link").attr();

text metodu bir elementin metin içeriğini düzenlemek ya da çekmek için kullanılır. Tablo 5.5'teki ilk örnekte *id* değeri *p1* olan bir elementin metni "Birinci paragraf" şeklinde düzenlenmektedir. Bu kod çalıştırıldığında element içindeki metin, verilen ifade ile değiştirilecektir. Bir elementin içindeki metni çekmek için text metodu parametre verilmeden çağırılır. İkinci örnekteki kod, $a değişkenine, *id* değeri *p1* olan elementin metnini aktarmaktadır.

> text metodu, elementin metnini sıfırlayarak değiştirir. Element metnine ekleme yapmak için append metodu kullanılmalıdır.

Bir elementin içeriğine html kodları eklenmek istenirse text metodu yerine html kullanılır. text metodunun amacı metin eklemek olduğundan, parametre olarak gönderilen HTML kodları yorumlanmadan karakter olarak eklenecektir. html metodu ile gönderilen kodlarsa tarayıcı tarafından yorumlanır. Tablo 5.5'te üçüncü örnekte *id* değeri *p1* olan elementin içine bir bağlantı eklenmektedir. Dördüncü örnekte ise, *id* değeri *p1* olan elementin içeriği çekilerek $a değişkenine aktarılmaktadır.

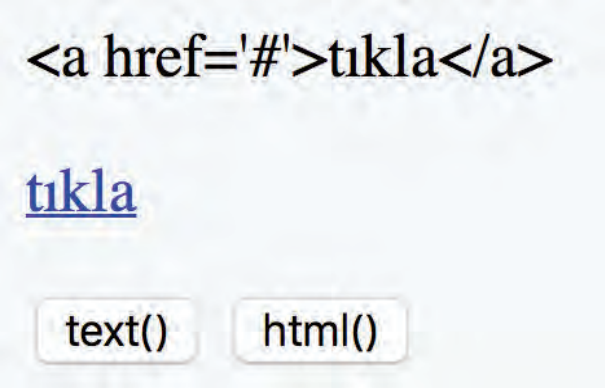
Şimdi *text* ve *html* metotları arasındaki farkı örnekleyelim. Aşağıda verilen kodda, *id* değerleri *p1* ve *p2* olan iki paragraf ve *id* değerleri *b1* ve *b2* olan iki düğme bulunmaktadır. *b1* düğmesi, *p1* paragrafının içeriğini text metoduyla; *b2* düğmesi *p2* paragrafının içeriğini html metoduyla değiştirmektedir. Her iki metotta da elementlerin içine aynı bağlantı kodu (tıkla) gönderilmektedir.

```
1   <p id="p1">Birinci paragraf</p>
2   <p id="p2">İkinci paragraf</p>
3   <button id="b1">text()</button>
4   <button id="b2">html()</button>
5   <script>
6     $(document).ready(function(){
7       $("#b1").on("click", function(){
8         $("#p1").text("<a href='#'>tıkla</a>");
9       });
10
11      $("#b2").on("click", function(){
12        $("#p2").html("<a href='#'>tıkla</a>");
13      });
14    });
15  </script>
```

Bu kodlar çalıştırılarak düğmelere tıklandığında, ilk paragrafta gönderilen kodlar metin olarak görülürken, ikinci paragrafta kodlar yorumlanarak bir bağlantı oluşturulmaktadır (Resim 5.4).

text ve *html* metotları formlarda kullanılan giriş elementlerinin içeriklerini düzenlemek için kullanılamaz. Bu elementlerin içeriklerini düzenlemek ya da içeriklerine erişmek için val metodu kullanılır. Tablo 5.5'teki beşinci örnekte id değeri isim olan bir elementin değeri "Onur" olarak düzenlenmektedir. Altıncı örnekte ise *$isim* değişkenine, *id* değeri isim olan elementin değeri atanmaktadır.

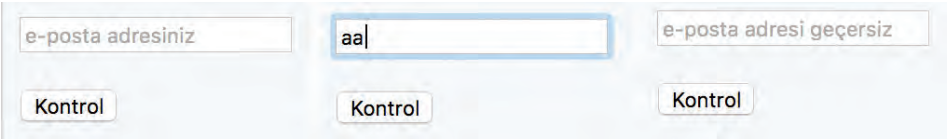
text, html ve *val* metotları elementlerin sayfada görünen değerlerini düzenleyebilir ya da çekebilir. *attr* metodu ise elementlerin niteliklerini (attribute) düzenlemek ya da çekmek için kullanılır. Aşağıda verilen kodda, formdaki bir e-posta alanına girilen değer genel tanımlama ifadesi (regular expression) kullanılarak kontrol edilmektedir.



Resim 5.4 text ve html Metotlarının Farkı

```
1   <input id="eposta" type="email" placeholder="e-osta adresiniz">
2   <button id="kontrol">Kontrol</button>
3   <script>
4   $(document).ready(
5     $("#kontrol").on("click", function(){
6   var pattern = new RegExp(/^[+a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/i);
7       $posta=$("#eposta").val();
8       if(!pattern.test($posta))
9       {
10        $("#eposta").val("");
11  $("#eposta").attr("placeholder", "e-posta adresi geçersiz");
12      }
13    })
14  );
15  </script>
```

Kullanıcı tarafından girilen e-posta adresinin yanlış olması durumunda elementin içeriği boşaltılarak placeholder niteliği değiştirilmektedir. Bu sayede kullanıcının tekrar denemesine izin verilmektedir. Bu kodun çıktısı Resim 5.5'te verilmiştir.



Resim 5.5 js fiddle İşi

JQuery CSS Metotları

JQuery'nin önemli işlevlerinden biri de elementlerin CSS özelliklerini düzenleyebilmesidir. CSS metotları elementlerin görünüşlerini değiştirmek için kullanılır. Elementin CSS özellikleri üzerinde doğrudan düzenleme yapmak ya da özellikleri çekmek için `css` metodu kullanılır. `css` metodunun kullanımını bir önceki örneğimizle devam ederek örnekleyelim. Kullanıcının geçersiz bir e-posta adresi girmesi halinde `id` özelliği `eposta` olan elementin kenarlıklarını kırmızıya boyamak için `css` metodu kullanılabilir. Aşağıda verilen kodda `css` metodu iki parametre ile çağırılmaktadır.

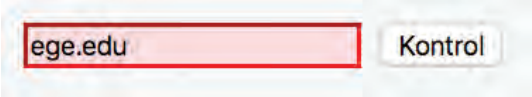
```
$("#eposta").css("border-color","red");
```

İlk parametre düzenlenecek CSS özelliğini belirtmek için, ikinci parametre ise bu özelliğe atanacak değeri vermek için kullanılır. Bu parametrelerin tırnaklar içinde gönderildiğine dikkat ediniz. Bu kodun işletilmesi sonucunda giriş kutusunun görüntüsü Resim 5.6'daki gibi olacaktır.

Resim 5.6 Ekran Çıktısı

css metodu ile birden fazla özelliği aynı anda değiştirmek için nesne yazım tekniği kullanılır. Bu amaçla CSS özelliği-değer çiftleri halinde oluşturulan bir nesne css metodunda parametre olarak eklenir. Nesne içinde CSS özellikleri ile değerler iki nokta ile birleştirilir. Kurallar virgüller ile ayrılır.

```
$("#eposta").css({"border-color": "red", "background-color": "#fdd"});
```



Resim 5.7 Ekran Çıktısı

Yukarıda verilen kodda, elementin hem kenarlık (border-color) hem de artalan rengi (background-color) değiştirilmektedir. Bu kodların çalıştırılması sonucunda elementin görüntüsü Resim 5.7'deki gibi olacaktır.

CSS özelliklerinin elementlere uygulama biçimlerinden biri de elementlerin *class* özelliğine hazırlanmış CSS sınıflarının atanmasıdır. CSS sınıfları elementlere atanarak sınıflardan gelen özelliklerin elemente uygulanması sağlanabilir. Bu işlem genellikle tasarımcılar tarafından HTML kodları yazılırken gerçekleştirilir. Aşağıdaki örnekte *button* elementine *dugme* adlı CSS sınıfı uygulanmaktadır.

```
<button id="giris" class="dugme">Giriş</button>
```

JQuery, sayfanın çalıştırılması sırasında elementlerin class özelliklerine müdahale ederek görünüşlerini değiştirebilir. Sayfada tanımlı bir CSS sınıfını elemente eklemek için addClass, elemente eklenmiş bir sınıfı çıkarmak için removeClass metodu kullanılır. Bu metotların etkisini incelemek için öncelikle sayfamızda bir CSS sınıfı tanımlayalım.

```
1   <style>
2   .hatali {
3     background-color: #fdd;
4     border-color: red;
5   }
6   </style>
```

hatali sınıfı, uygulandığı elementin artalan rengini açık kırmızı, kenarlık rengini ise kırmızı olacak şekilde düzenlemektedir. Örneğimizden devam edecek olursak; Aşağıda verilen kodun dördüncü satırında, kullanıcının yanlış bir e-posta adresi girmesi durumunda elemente hatali sınıfı uygulanmaktadır. Aksi halde, sekizinci satırda hatali sınıfı elementten kaldırılmaktadır.

```
1   <script>
2     if(pattern.test($posta))
3     {
4       $("#eposta").removeClass("hatali");
5     }
6     else
7     {
8       $("#eposta").addClass("hatali");
9     }
10  </script>
```

CSS'in çalışma prensibi gereği elemente eklenen bir özellik daha önce düzenlendiyse bile değiştirilir. Örneğin elemente eklenen iki sınıftan biri kenarlık rengini kırmızı, diğeri yeşil yapmaya çalışıyor olabilir. Bu durumda en son yazılan sınıfın belirlediği renk görünecektir. Bu nedenle, elemente uygulanan iki sınıfın artalan rengi gibi ortak bir özelliğe müdahale etmesi durumunda istenmeyen rengi tutan sınıfın removeClass kullanılarak elementten atılması gerekir. Örneğin hatalı girişler için kırmızı, doğru girişler için yeşil temalı sınıflar kullanmak istediğimizi varsayalım. Bu amaçla örneğimize yeni bir sınıf eklenmektedir.

```
1   <style>
2   .gecerli {
3     background-color: #dfd;
4     border-color: #0f0;
5   }
6   </style>
```

hatali ve *gecerli* sınıflarını bir elemente aynı anda uyguladığımızda hangi rengin gösterileceğinden emin olamayız. Bu nedenle düzgün sonuç vereceğimizden emin olabilmek için kodlarımızı düzenlemeliyiz. Aşağıdaki kodda, kullanıcı girişinin doğru olması durumunda yeşil temanın görüntüleneceğinden emin olmak için dördüncü satırda hatali sınıfı elementten kaldırılmaktadır. Bunun ardından beşinci satırda gecerli sınıfı elemente uygulanmaktadır.

```
1   <script>
2     if(pattern.test($posta))
3     {
4    $("#eposta").removeClass("hatali");
5       $("#eposta").addClass("gecerli");
6     }
7   </script>
```

toggleClass metodu, element üzerindeki sınıfın durumunu değiştirmek için kullanılır. *toggleClass* elementi kontrol ederek, verilen sınıf elementte bulunuyorsa çıkarır, bulunmuyorsa ekler. Bu sayede, *toggleClass* kullanılarak bir elementin iki durum arasında geçiş yapması sağlanabilir. Aşağıda verilen kodda, bir paragrafa tıkladığında artalan rengini değiştiren bir sınıf *toggleClass* ile değiştirilmektedir. Bu sayede paragraf her tıklandığında artalan rengi sarı ve beyaz arasında geçiş yapmaktadır.

```
1   <style>
2     .boya{
3     background-color: #ff0;
4   }
5   </style>
6   <p id="p1">Paragraf metni.</p>
7   <script>
8     $("#p1").on("click", function(){
9       $("#p1").toggleClass("boya");
10    });
11  </script>
```

toggle ifadesi içeren JQuery metotları (toggleClass, slideToggle vb.) ilgili özelliği tersine çevirir.

Öğrenme Çıktısı

3 JQuery kütüphanesinin kullanım amaçlarını sıralayabilme
4 Sayfalara JQuery ve JQuery mobile kütüphanelerini ekleyebilme
5 Kurallara uygun JQuery kodları yazabilme

Araştır 3

Hatalı ve geçerli sınıfları ile ilgili kodu kullanıcı girişinin yanlış olması durumunda düzgün tepki verecek şekilde geliştiriniz.

İlişkilendir

JQuery DOM metotlarını ilişkilendirerek açıklayınız.

Anlat/Paylaş

jQuery ile hazırlanmış Web sitesi ve uygulama örneklerini özgün yapılarına vurgu yaparak paylaşınız.

JQUERY MOBİLE

JQuery Mobile (JQM) HTML5 temelli, öncelikle mobil cihazları hedef alan bir kullanıcı arayüzü geliştirme sistemidir. JQM ile geliştirilen arayüzler farklı ekran boyutlarına tepki verebilir (responsive). Bu sayede oluşturulan arayüzler cep telefonu, tablet bilgisayar ve masaüstü bilgisayar ekranlarında farklı şekillerde sunulabilir. Bunun amacı kullanıcıya en iyi izleme ve okuma deneyimini yaşatmaktır. Bunun yanında JQM, dokunmatik ekranlardaki dokunma, taşıma, kaydırma gibi olaylara tepki verebilir. Bu sayede yalnızca HTML5 teknolojileri kullanılarak mobil cihazlar üzerinde çalışan arayüzler üretilebilir (Resim 5.8).

JQM, kendini kanıtlamış JQuery ve JQuery UI sistemleri üzerine temellenmiştir. Pek çok özelliğini bu sistemlerden miras almaktadır. JQM'in sunduğu başlıca özellikler arasında sayfalar arası geçişler, AJAX kullanarak veri çekme, dokunma olayları ve öncelikle mobil cihazlar için üretilmiş elementler sayılabilir.

JQM, JQuery kuruluşu çatısı altında sürdürülmekte olan açık kaynak kodlu ve ücretsiz bir sistemdir. Sürekli geliştirilmekte olan sistemin, bu ünitenin yazıldığı 2017 yılı Ocak ayındaki kararlı sürümünün numarası 1.4.5'tir.

✓ Tepkisel tasarım (responsive design), internet siteleri ve uygulama arayüzlerinin farklı ekran boyutları ve tarayıcı genişliklerine göre kendilerini yeniden düzenleyebildiği bir tasarım yaklaşımıdır. Bu yaklaşımda, arayüzler tarayıcı boyutuna göre sunumlarını ve ölçeklerini değiştirebilmektedir. Yaklaşımın amacı okunurluğu desteklemek ve kullanıcı etkileşimini iyileştirmektir. Yaklaşımın temelinde CSS 3 tanımlamaları ile birlikte gelen medya sorguları (media querries) yer almaktadır.

Resim 5.8 JQuery Mobile Sisteminin Logosu

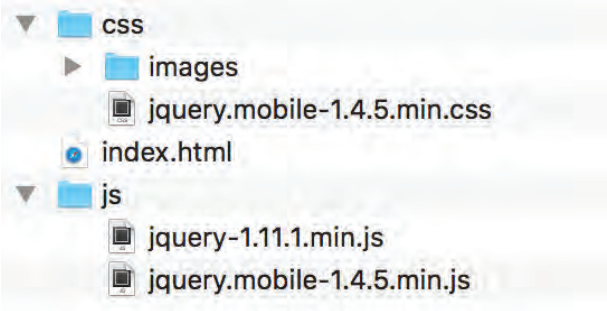
✓ JQuery UI, internet siteleri ve uygulama arayüzleri geliştirmek için elementler, efektler ve temalar sunan farklı bir kütüphanedir.

Sayfalara JQM Eklemek

Bir projeye JQM eklemek için gerekli tüm kaynaklara http://jquerymobile.com/download/ adresinden ulaşılabilir. Bu adresteki "Latest stable version" başlığının altındaki Zip File bağlantısı kullanılarak JQM dosyaları yerel projelerde kullanılmak üzere indirilebilir. Aynı sayfadaki İDA linkleri üzerinden belgelerin uzak sistemlerdeki adreslerine ulaşılabilir.

Yerel kullanım için indirilen .zip dosyası içinde pek çok dosya olduğu görülecektir. Bu paket içindeki demos klasörü JQM'in yeteneklerini inceleyebilmeniz için sağlanan sunum sayfalarını içermektedir. Projenizde bu klasörü tutmanıza gerek yoktur. Bununla birlikte, images klasörü altında projeniz için gerekli olacak simgeler bulunduğundan bu klasörü silmeniz ya da adını değiştirmeniz önerilmez.

JQM kodlamaya başlamak için bu dosyalar arasından jquery.mobile-1.4.5.js (ya da min versiyonu) ve jquery.mobile-1.4.5.css (ya da min versiyonu) dosyalarının projeye bağlanması gerekmektedir. Resim 5.9'te klasör yapısı gösterilen örnek projemizde bu dosyalar uzantıları ile aynı isimli klasörlere yerleştirilmiştir. Bu klasör yapısına göre JQM sisteminin CSS dosyasını belgeye bağlamak için aşağıdaki kod satırının HTML belgesinin head etiketlerinin arasına eklenmesi gerekmektedir.



Resim 5.9 Yerel Bir JQM Projesinin Klasör Yapısı

```
<link rel="stylesheet" href="css/jquery.mobile-1.4.5.min.css">
```

Yerel çalışmak yerine bir İDA üzerinden bu belgenin dosyaya eklenmesi için head etiketleri arasına aşağıdaki kod satırının eklenmesi gerekmektedir.

```
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css" />
```

CSS dosyalarının yanında iki adet JS dosyasının HTML belgesine bağlanması gerekmektedir. Yerel projemizde bu dosyalar js klasörünün altında yer almaktadır. JQM, JQuery altyapısını kullandığından öncelikle JQuery dosyasının belgeye bağlanması gerekmektedir. Aşağıda verilen kodda yerel yöntemde bu dosyanın bağlanması için gerekli kod satırı görülmektedir.

dikkat

Projedeki görsellerin doğru gösterilebilmesi için .zip dosyası içinde gelen images klasörünün css dosyası ile aynı klasörde yer alması gerekmektedir.

```
<script src="js/jquery-1.11.1.min.js"></script>
```

Bir zorunluluk olmamakla birlikte, sayfaların açılış sürelerinin kısaltılması için bu bağlantıların body etiketinin kapanmasının hemen ardından eklenmesi önerilmektedir. Yerel yöntem yerine, İDA üzerinden bu dosyanın belgeye bağlanması için aşağıdaki kod satırı kullanılabilir.

```
<script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
```

JQuery'nin ardından JQM sisteminin JS dosyasının belgeye bağlanması gerekmektedir. Yerel projemizde JS dosyaları js klasörü altında yer aldığından, yerel yöntemde bu dosya aşağıda görüldüğü şekilde projeye bağlanmalıdır.

```
<script src="js/jquery.mobile-1.4.5.min.js"></script>
```

JQM sisteminin JS dosyasının belgeye İDA üzerinden bağlanması için aşağıdaki kod satırı kullanılabilir.

```
<script src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></
script>
```

İDA kaynakları kullanılarak oluşturulmuş temel bir JQM sayfasının kodları aşağıda görülmektedir.

```
1   <!DOCTYPE html>
2   <html lang="tr">
3   <head>
4     <meta charset="utf-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1">
7     <title>JQM Uzak Demo</title>
8     <link rel="stylesheet"
    href="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css" />
9   </head>
10  <body>
11    <h1>Merhaba</h1>
12  </body>
13  <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
14  <script    src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.
    js"></script>
15  </html>
```

Buradaki kodların çoğunluğu tarayıcı ekranında görünmemektedir. Sadece body etiketi içindeki birinci düzey başlık (h1) etiketi içeriği olan "Merhaba" yazısı görülmektedir. Bu içeriğe JQM temel CSS kuralları uygulanmış olduğundan sayfanın artalan rengi açık gri (#F9F9F9), yazı rengi koyu gri (#333333), yazı gölgesi açık gri (#f3f3f3) ve yazı tipi ise kullanılan bilgisayar üzerindeki temel sans-serif font olarak ayarlanmıştır. Bilgisayarınızda bu görüntüyü görebiliyorsanız temel JQM şablonunu düzgün oluşturduğunuzu anlayabilirsiniz.

dikkat

Bu kodlar, Google firması tarafından üretilen Chrome internet tarayıcısının güvenlik ayarları nedeniyle çalışmayabilir. Bu durumda, sayfanın alt bölümünde JS kodlarını bağladığınız bölümü aşağıda verilen kodda görüldüğü şekilde düzenlemeniz gerekecektir.

```
1   <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
2   <script>
3     $(document).bind('mobileinit',function(){
4       $.mobile.changePage.defaults.changeHash = false;
5       $.mobile.hashListeningEnabled = false;
6       $.mobile.pushStateEnabled = false;
7     });
8   </script>
9   <script    src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.
js"></script>
```

JQM Sayfaları

JQM tek sayfalık uygulamalar (Single page application) üretmek için kullanılır. Tek sayfalık uygulamalar tek bir HTML sayfasında oluşturulur. Tüm içerikler bir sayfa içine yazıldığından, bu uygulamalar sayfaların yeniden yüklenmesine gerek bırakmaz. Kullanıcı uygulamanın içinde önceden oluşturulmuş sanal sayfalar arasında gezinebilir ya da bir internet servisinden çekilen sanal sayfa içeriklerini inceleyebilir. Sayfa tamamen tekrar yüklenmediğinden ya da başka bir sayfaya yönlendirilmediğinden kullanıcılar masaüstü uygulamalardakine benzer deneyimler yaşar. Bu ünitede yer kısıtlamaları nedeniyle sadece önceden hazırlanmış sanal sayfalar arasında gezinilebilen bir tek sayfalık uygulamanın geliştirilmesi anlatılacaktır. Okuyucu JQM'in AJAX yeterliklerini kullanarak dinamik sayfa içerikleri yükleyebilen uygulamaları kolayca üretebilecektir.

JQM arayüzlerinin temel bileşeni sayfalardır. Sayfalar internet sitelerindeki HMTL dosyalarına benzetilebilir. Örneğin JQM uygulamanızda bir sayfada giriş formu, bir sayfada iletişim formu, bir sayfada uygulama içerikleri yer alabilir. Bununla birlikte, JQM tek sayfalık uygulamalar ürettiğinden, tüm bu sanal sayfalar tek bir HTML dosyası içinde oluşturulur. JQM sayfası üretmek için bir div elementlerinin data-role özelliği page olarak düzenlenir. HTML dosyasında istenildiği kadar JQM sayfası oluşturularak bunlar birbirlerine bağlanabilir. Aşağıda verilen kodda üç sayfa oluşturulmuştur. JQM'in çalışma mantığına göre uygulama çalışırken bir anda bu sayfalardan yalnızca biri gösterilir. Uygulamanın açılışında body etiketi içinde bulunan ilk sayfa otomatik olarak gösterilir. Buradan sonra, kullanıcı etkileşimi ile bu sayfalar arasında gezinme yapabilir.

dikkat

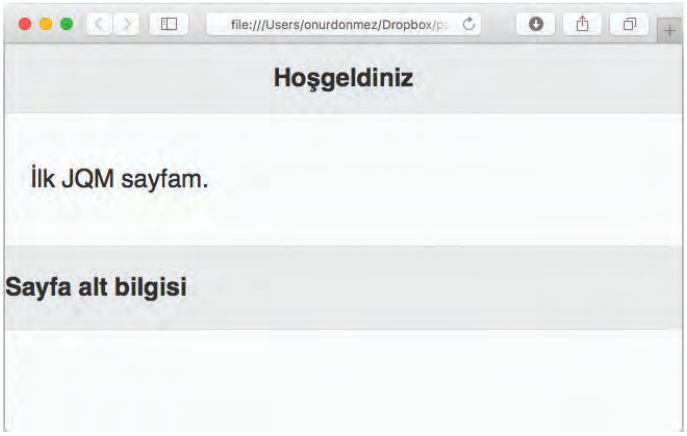
JQM sistemi tek sayfalık uygulamalar üretir. Bu bölümde, buradan sonra sayfa olarak ifade edilen elementler uygulamaların içindeki sanal sayfalardır.

```
<body>
  <div data-role="page">
    <p>1. Sayfa içeriği</p>
  </div>
  <div data-role="page">
    <p>2. Sayfa içeriği</p>
  </div>
  <div data-role="page">
    <p>3. Sayfa içeriği</p>
  </div>
</body>
```

Sayfaların içinde üç ana alan tanımlanabilir. Header (başlık) alanı sayfaların üst kısmında görünen barı ve içeriğini tanımlar. Bu alanda genellikle sayfa başlığı ve araç kutuları yer alır. Header alanını tanımlamak için, sayfanın içindeki bir div elementinin data-role özelliği header olarak belirlenir. Sayfanın alt kısmında görülecek bar ise footer (alt bilgi) alanı olarak tanımlanır. Bu amaçla sayfanın içindeki bir div elementinin data-role özelliği footer olarak belirlenir. Sayfa içeriğinin gösterileceği alan ise data-role özelliği main olarak belirlenmiş bir div elementi ile tanımlanır. Bu alanda sayfada görünecek ana içeriği oluşturan paragraflar, görseller, videolar, tablolar gibi elementler yer alır. Ek olarak kenar boşluklarının ayarlanması ve içeriklerin düzgün yerleşimi için main div elementine ui-content sınıfı eklenmelidir. Bu alanların herhangi birinde paragraflar, görseller, başlıklar kullanılabilir. Aşağıdaki kodda bir sayfalı temel bir JQM uygulaması oluşturmaktadır.

```
1   <body>
2     <div data-role="page">
3       <div data-role="header">
4         <h1>Hoşgeldiniz</h1>
5       </div>
6       <div data-role="main" class="ui-content">
7         <p>İlk JQM sayfam.</p>
8       </div>
9       <div data-role="footer">
10        <p>Sayfa alt bilgisi</p>
11      </div>
12    </div>
13  </body>
```

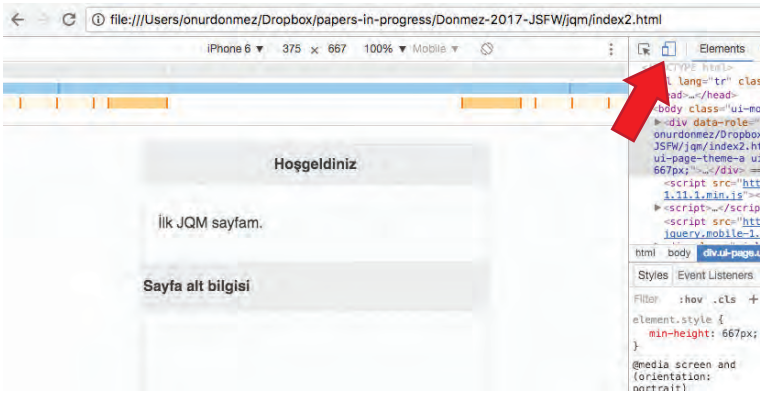
Bu sayfada header, footer ve main alanları ve içerikleri görülebilmektedir. Oluşturulan sayfanın ekran görüntüsü Resim 5.10'de görünmektedir.



Resim 5.10 Temel Bir JQM Sayfasının Bölümleri

Resim 5.10 oluşturulan uygulamanın masaüstü internet tarayıcılarında nasıl göründüğünü örneklemektedir. Google Chrome internet tarayıcısının geliştirici araçlarını kullanarak bu uygulamanın bir mobil cihaz üzerinde nasıl görüneceğini inceleyebilirsiniz. Bu amaçla sayfanıza sağ tıklayarak çıkan menüden "İncele" (İngilizce işletim sistemlerinde Inspect) seçeneğini tıklamanız gerekmektedir. Açılan panelin sol üstündeki mobil cihaz ikonunu tıklayarak Chrome tarayıcısının mobil cihaz arayüzüne geçebilirsiniz. Bu araç ile oluşturduğunuz arayüzün farklı ekran genişliklerindeki cihazlarda (iPad, iPhone, Galaxy S5 vb.) nasıl görüneceğini inceleyebilirsiniz. Resim 5.11'de oluşturulan uygulamanın iPhone 6 akıllı telefonunun ekranında nasıl göründüğü incelenmektedir.

Mobil cihazların ekranlarının en boy oranları, çözünürlükleri, nokta aralıkları ve yönelimleri (yatay ve dikey) gibi özellikleri çok çeşitli olabilmektedir. Bu çerçevede farklı ekran özelliklerinde yazıların ve içeriğin doğru gösterilebilmesi önemli bir sorun haline gelebilmektedir. JQM öncelikle mobil araçları hedef alan bir platform olduğundan bu sorunu kendi içinde çözebilmektedir. Resim 5.12'deki ekran görüntülerinde JQM ile hazırlanmış bir sayfanın tablet ve akıllı telefon ekranlarındaki görüntüleri incelenebilir. Özellikle kenar boşlukları, paragraflar arası boşluklar, satır aralıkları gibi tipografik özelliklerin okunurluğu arttırmak için düzenlenmiş olduğuna dikkat ediniz.



Resim 5.11 Temel Bir JQM Sayfasının Bölümleri

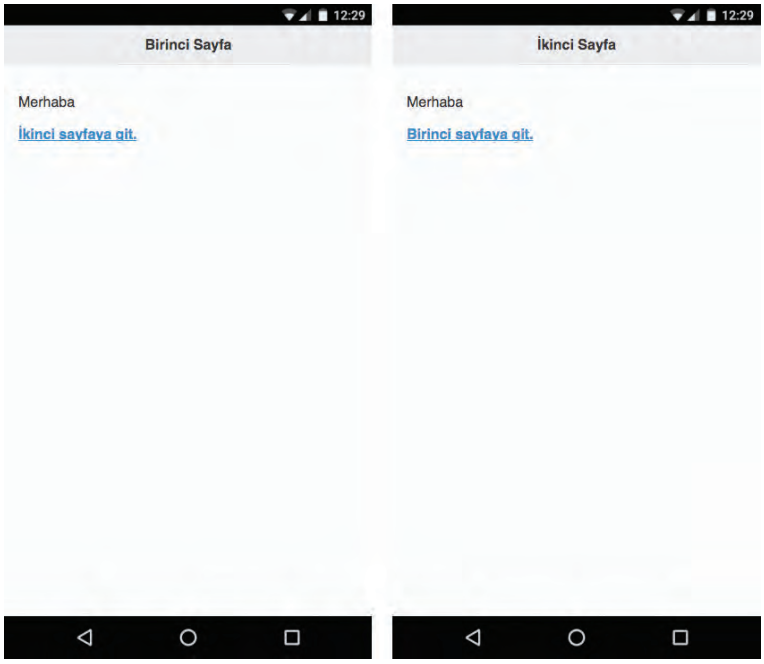
Resim 5.12 Bir JQM Sayfasının Farklı Mobil Cihazlardaki Görüntüsü (Soldan Sağa: İpad Dikey, İpad Yatay, İphone 6)

JQM uygulamalarında genellikle birden fazla sayfa bulunur. Bu sayfalar arasındaki geçişler için HTML bağlantıları kullanılır. Aşağıda verilen kodda iki sayfalı bir yapı oluşturularak bu sayfalar birbirlerine bağlanmaktadır.

```
1   <body>
2     <div id="p1" data-role="page">
3       <div data-role="header">
4         <h1>Birinci Sayfa</h1>
5       </div>
6       <div data-role="main" class="ui-content">
7         <p>Merhaba</p>
8        <p><a href="#p2">İkinci sayfaya git.</a></p>
9       </div>
10    </div>
11    <div id="p2" data-role="page">
12      <div data-role="header">
13        <h1>İkinci Sayfa</h1>
14      </div>
15      <div data-role="main" class="ui-content">
16        <p>Merhaba</p>
17        <p><a href="#p1">Birinci sayfaya git.</a></p>
18      </div>
19    </div>
20  </body>
```

JQM masaüstü tarayıcılarda çalışabilen bir sistemdir. Google Chrome tarayıcısındaki geliştirme araçları ile arayüzlerin mobil cihazlarda nasıl görüneceği de incelenebilir. Bunun yanında, oluşturduğunuz arayüzlerin gerçek mobil telefonlarda nasıl görüleceğini incelemek için bilgisayarınıza yerel bir HTTP sunucu kurabilirsiniz. Bu amaçla XAMP (https://www.apachefriends.org/tr/index.html) ya da WAMP (http://www.wampserver.com) gibi platformları kullanabilirsiniz. Oluşturduğunuz arayüzleri bilgisayarınızdaki yerel sunucuya bağlanarak kendi telefonlarınızdan inceleyebilirsiniz.

Oluşturulan arayüzün Chrome tarayıcısının geliştirme araçlarında Nexus 5X telefonlar için oluşturulan ekran görüntüleri Resim 5.13'te sunulmuştur. JQM sayfalarının referansları id değerleridir. Yukarıda verilen kodda, ikinci satırda ilk sayfanın id değeri p1, 11. satırda ikinci sayfanın id değeri p2 olarak belirlenmiştir. Bir bağlantıyı bir sayfaya yönlendirmek için, sayfa referansının önüne # eklenir. Bu durumda ilk sayfaya yönlendirilecek bir bağlantının href değerinin #p1 olarak düzenlenmesi gerekmektedir. 17. satırda, ikinci sayfadan birinci sayfaya yönlendiren bağlantının href özelliğinin #p1 olduğuna dikkat ediniz. Benzer şekilde, sekizinci satırda, ikinci sayfaya yönlendiren bağlantının href özelliği #p2 olarak düzenlenmiştir. Bu yolla pek çok sayfadan oluşan uygulamalar üretilerek sayfalar birbirlerine bağlanabilmektedir.



Resim 5.13 JQM ile Oluşturulmuş İki Sayfalı Bir Arayüz

JQM Kutuları

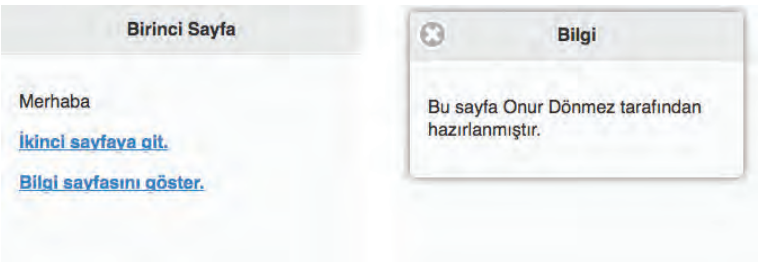
JQM sisteminde diyalog kutuları, açılır kutular ve paneller olmak üzere üç farklı kutu elementi kullanılabilir.

Diyalog Kutuları

Diyalog kutularının amacı kullanıcıya mesaj göstermek ya da kullanıcıdan bildirim almaktır. Hazırlanan herhangi bir JQM sayfası diyalog kutusu olarak gösterilebilir. Bunun için sayfa oluşturulan div etiketine data-dialog="true" ifadesi eklenir. Aşağıdaki kodda oluşturulan sayfa diyalog olarak gösterilmek üzere hazırlanmıştır. Bu sayfanın çağırılması için diğer sayfalarda olduğu gibi bağlantının href özelliğine # ile birlikte sayfa referansının (#p3) yazılması yeterlidir. Burada yalnızca bir mesaj gösterilmek üzere kullanılmışsa da, diyalog kutuları herhangi bir elementi kullanabilir.

```
1   <div id="p3" data-role="page" data-dialog="true">
2     <div data-role="header">
3       <h1>Bilgi</h1>
4     </div>
5     <div data-role="main" class="ui-content">
6       <p>Bu sayfa Onur Dönmez tarafından hazırlanmıştır.</p>
7     </div>
8   </div>
```

Aşağıda Resim 5.14'te sağda oluşturulan diyalog sayfası görülmektedir. Diyalog sayfasını kapatmak için gerekli kontrol düğmesi JQM tarafından otomatik olarak eklenmektedir.

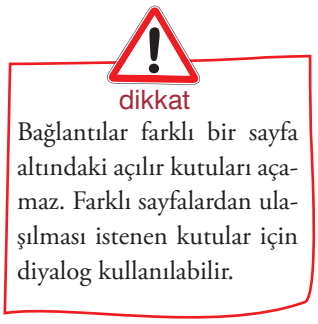


Resim 5.14 JQM Sisteminde Bir Sayfanın Diyalog Olarak Kullanılması

Açılır Kutular

Kullanıcılara mesaj vermek için açılır kutular kullanılabilir. Açılır kutu oluşturmak için sayfanın içindeki bir div elementinin data-role özelliği popup olarak belirlenmelidir. Aşağıda verilen kodlarda bir açılır kutu oluşturulmaktadır.

```
1   <div id="kutu1" data-role="popup" class="ui-content">
2     <h1>Açılır kutu</h1>
3     <p>Açılır kutu mesajı</p>
4   </div>
```



Bu kutunun *id* özelliği *kutu1* olarak belirlendiğinden bu kutuyu açmak için hazırlanacak bağlantının *href* özelliğinin *#kutu1* olarak düzenlenmesi gerekmektedir. Bunun yanında, bu bağlantının bir açılır kutuyu açtığını belirtmek için data-rel özelliği popup olarak düzenlenmelidir. Diğer önemli bir nokta da, bağlantı ve açılır kutunun aynı sayfanın içinde olması gerektiğidir.

Resim 5.15'te bir sayfa üzerinde açılır kutu kullanımı gösterilmektedir. En soldaki görselde yer alan Açılır kutuyu göster bağlantısına tıklandığında, ortadaki resimdeki açılır kutu gösterilmektedir. Açılır kutuların kapatılması için kutunun dışına tıklanması/dokunulması yeterlidir. Bunun yanında, kutunun kapatılması için bir kontrol sağlanabilir. Aşağıdaki kodlar açılır kutunun oluşturulduğu div elementinin içine eklendiğinde, Resim 5.15'te en sağdaki resimdeki açılır kutu gösterilecektir. Bu elementte kullanılan sınıfların açıklamaları düğmeler başlığı altında incelenebilir.

```
<a href="#" data-rel="back" class="ui-btn ui-corner-all ui-shadow ui-btn ui-icon-delete ui-btn-icon-notext ui-btn-right">Kapat</a>
```

Resim 5.15 Açılır Kutu (Popup) Elementi (Dikey Akıllı Telefon)

Açılır kutuların önemli bir kullanım alanı sayfada resim gösterilmesidir. Sayfadaki küçük resimlerin büyük hallerini göstermek için açılır kutular sağlanabilir. Aşağıdaki kodun bağlantı elementinde, sayfadaki bir açılır kutu elementine referans sağlanmıştır. Bu bağlantının içinde genişliği 150 piksel olan bir resim yerleştirildiğine dikkat ediniz. Resmin büyük hali ise sayfadaki bir açılır kutu elementinin içine yerleştirilmiştir. Bu sayede bağlantı içindeki küçük resme tıklandığında büyük resim bir açılır kutu içinde gösterilebilmektedir.

```
1   <a href="#myPopup" data-rel="popup" data-position-to="window">
2   <img src="jquery-logo.png" alt="jqm logo" style="width:150px;"></a>
3
4   <div data-role="popup" id="myPopup">
5    <a href="#" data-rel="back" class="ui-btn ui-corner-all ui-shadow ui-btn ui-
     icon-delete ui-btn-icon-notext ui-btn-right">Kapat</a>
6    <img src="jquery-logo.png" style="width:100%;" alt="JQM logo">
7   </div>
```

Paneller

Oldukça esnek bir tasarıma sahip olan paneller, menüler, formlar ve yardımcı araçlar oluşturmak gibi pek çok amaçla kullanılabilir. Diyaloglar ve açılır kutular gibi normalde görünmeyen paneller, ekranların bir tarafını kaplayacak şekilde yerleştirilir ve kullanıcı etkileşimine bağlı olarak gösterilir.

Aşağıda verilen kodda bir JQM sayfasında panel kullanmak için gerekli genel yerleşim şeması gösterilmiştir. Panel kodları JQM sayfaları içine yazılır. Bu sayede her sayfada farklı bir panel içeriği gösterilebilir.

```
1   <div data-role="page">
2     <div data-role="panel" id="panel1">
3       <!-- panel içeriği buraya yazılır -->
4     </div>
5     <div data-role="header"></div>
6     <div data-role="main"></div>
7     <div data-role="footer"></div>
8   </div>
```

Yukarıdaki kodun ikinci satırında görüldüğü gibi paneller data-role özellikleri panel olarak belirlenmiş div elementleri ile oluşturulur. Panelleri çağırabilmek için elementlere bir id verilmelidir. Bu paneli çağırmak için sayfadaki bir düğme ya da bağlantının href özelliğine #panel1 yazılması yeterlidir. Sayfalar içinde paneller header elementlerinin önüne ya da footer elementlerinin arkasına yazılmalıdır. Bu iki elementin arasına panel yerleştirilmesi önerilmez. Yukarıdaki kodda panel kodlarının *header* elementinin önünde yazıldığına dikkat ediniz. Alternatif olarak panel kodları yedinci satırdaki *footer* elementinin arkasına yerleştirilebilir.

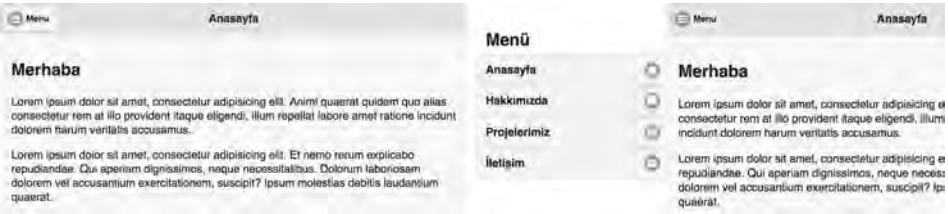
Aşağıda verilen kodda panel içinde bir menü hazırlanmıştır. Birinci satırda paneli çağıran bir düğme oluşturulmuştur. Bu düğme sayfanın herhangi bir yerine yerleştirilebilir. Ekran görüntüsünde düğme header elementinin içine yerleştirildiğinden sayfanın üstünde yer almaktadır.

```
1   <a href="#panel1" class="ui-btn ui-btn-inline ui-btn-corner-all ui-btn-icon-
left ui-icon-bars">Menu</a>
2   <div id="panel1" data-role="panel">
3     <h2>Menü</h2>
4     <ul data-role="listview">
5       <li><a href="#ana" class="ui-icon-home">Anasayfa</a></li>
6       <li><a href="#hak" class="ui-icon-comment">Hakkımızda</a></li>
7       <li><a href="#pro" class="ui-icon-grid">Projelerimiz</a></li>
8       <li><a href="#ile" class="ui-icon-mail">İletişim</a></li>
9     </ul>
10  </div>
```

Paneli oluşturan kodlar ikinci satırda başlamaktadır. Burada panelin id'sinin panel1 olarak ayarlandığına ve düğmedeki href değerinin #panel1 olarak belirlendiğine dikkat ediniz. Üçüncü satırda panelin üst kısmında görünecek bir başlık eklenmiştir. Dördüncü satırda başlayan bir liste elementi ile Ansayfa, Hakkımızda, Projelerimiz ve İletişim bağlantıları oluşturulmuştur. Bu bağlantılar href özellikleri ile (ör: #hak) ilgili sayfalara yönlendirilmiştir. Bağlantılar üzerindeki görseller simge sınıfları kullanılarak (ör: ui-icon-home) değiştirilmiştir.

Varsayılan olarak paneller sayfada sol kenara yerleştirilir. Bu davranışı değiştirmek için panellerin data-position özelliği kullanılabilir. Bu özellik kullanılarak panelin sayfanın solunda (data-position="left") ya da sağında (data-position="right") gösterilmesi sağlanabilir. Sayfanın üstünde ve altında panel gösterimi desteklenmez.

Panellerin gösterilmesi için üç tür etki kullanılabilir. JQM sisteminde varsayılan etkide, panel sabit kalırken sayfa kaydırılarak panelin üzeri açılır (data-display="reveal"). Bunun yanında, overlay ve push etkileri kullanılabilir. overlay etkisinde sayfa sabit kalırken, panel sayfanın üzerine gelecek şekilde kenardan kaydırılır. push etkisinde ise hem sayfa hem de panel kenardan kaydırılır.



Resim 5.16 JQM Sisteminde Panel Örneği (Yatay Akıllı Telefon)

Panellerin kapatılması için özel bir kod yazılması gerekmez. Açılan panelden bir bağlantıya tıklanması ya da dokunulması, sayfaya tıklanması ya da dokunulması, akıllı telefonlarda kaydırma (swipe) hareketlerinin kullanılması ya da bilgisayarlarda klavyeden ESC tuşuna basılması durumlarında paneller otomatik olarak kapatılır. Bunların yanında panelin içinde özel bir kapatma düğmesi sağlanabilir. Bu amaçla panelin içine aşağıdaki düğme kodu yerleştirilebilir.

```
<a href="#panel1" data-rel="back">Kapat</a>
```

Yukarıdaki örnekte menüyü sayfanın sağına yerleştirerek öteleme (push) etkisi ile açılacak şekilde düzenleyelim. Ek olarak bu panelin içine bir kapatma düğmesi yerleştirelim.

```
1   <div id="panel1" data-position="right" data-display="push" data-role="panel">
2   <h2>Menü</h2>
3   <ul data-role="listview">
4   <li><a href="#" class="ui-icon-home">Anasayfa</a></li>
5   <li><a href="#" class="ui-icon-comment">Hakkımızda</a></li>
6   <li><a href="#" class="ui-icon-grid">Projelerimiz</a></li>
7   <li><a href="#" class="ui-icon-mail">İletişim</a></li>
8   </ul>
9   <p>
10  <a href="#panel1" class="ui-btn ui-btn-icon-left ui-icon-delete" data-rel=
    "back">Kapat</a>
11  </p>
12  </div>
```

JQM Geçiş Efektleri

JQM, sayfa geçişleri ve kutuların gösterilmesi için bir dizi hazır geçiş efekti sağlar. Bu efektler sayfa, açılır kutu ya da diyalog kutusunu açacak olan bağlantının data-transition özelliğinde belirlenir. data-transition özelliği için kullanılabilecek değerler ve etkileri verilmiştir.

Tablo 5.6 JQM Sisteminde Sağlanan Geçiş Efektleri

Efekt	Açıklama	Örnek
fade	Kaynak sayfa soluklaştırılarak kaybolur. Varsayılan değerdir.	data-transition="fade"
flip	Hedef sayfa arkadan öne döndürülür	data-transition="flip"
flow	Sayfa uzaklaşıp kayar, yeni sayfa gelir	data-transition="flow"
pop	Sayfa kaybolur ikinci sayfa belirir	data-transition="pop"
slide	Sola doğru kayarak	data-transition="slide"
slidefade	Sayfalar sola doğru kaydırılır, hedef soluktan görünerek gelir	data-transition="slidefade"
slideup	Hedef sayfa aşağıdan yukarı doğru kayarak gelir	data-transition="slideup"
slidedown	Hedef sayfa yukarıdan aşağı doğru kayarak gelir	data-transition="slidedown"
turn	Sayfanın sol kenarı çevresinde dönerek	data-transition="turn"
none	Efekt kullanılmadan geçiş sağlanır.	data-transition="none"

Aşağıda verilen kodun birinci satırında başka bir sayfayı çağıran bağlantı slide efekti ile oluşturulmuştur. İkinci satırda ise bir açılır kutu slideup geçiş efekti ile çağırılmaktadır.

```
1   <a href="#p2" data-transition="slide">İkinci sayfaya git.</a>
2   <a href="#kutu1" data-transition="slideup" data-rel="popup">Açılır kutuyu göster.</a>
3   <a href="#p1" data-transition="slide" data-direction="reverse">Birinci sayfaya git.</a>
```

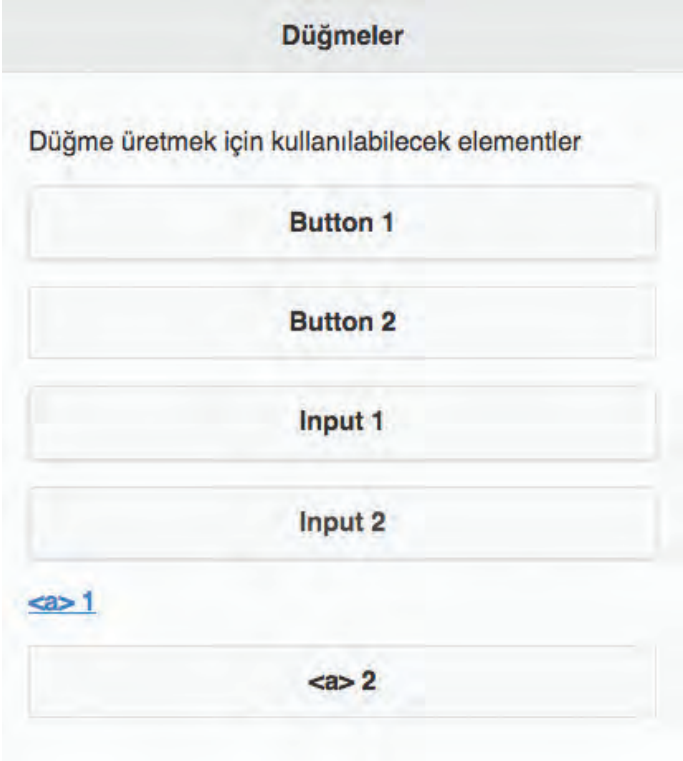
Geçiş efektlerinin tersten oynatılması olanaklıdır. Örneğin ilk sayfadan ikinci sayfaya geçerken sağdan sola; ikinci sayfadan birinci sayfaya gönderken de soldan sağa geçiş efekti uygulanabilir. Bunun için geçiş efektinin tersten oynatılması istenen bağlantıya *data-direction="reverse"* ifadesi eklenir. Yukarıdaki kodun üçüncü satırındaki bağlantı *slide* efektini tersten oynatacak şekilde düzenlenmiştir.

JQM Arayüz Araçları

Düğmeler

Arayüzlerin en çok kullanılan elementleri düğmelerdir. JQM sisteminde *input, button* ve *a* (bağlantı) elementleri düğme üretmek için kullanılabilir. Bu elementleri düğmeye çevirmek için ui-btn sınıfı eklenir.

Resim 5.17'de JQM sistemi ile oluşturulmuş düğmeler görülmektedir. Bu düğmeleri oluşturmak için yazılan kodlar Tablo 5.7'de sunulmuştur.



Resim 5.17 JQM Sisteminde Oluşturulmuş Düğmeler

Tablo 5.7 Farklı Elementler ile Düğme Oluşturmak İçin Yazılabilecek Kodlar Ve Sonuçları

Düğme	Kod
Button 1	<button>Button 1</button>
Button 2	<button class="ui-btn">Button 2</button>
Input 1	<input type="button" value="Input 1">
Input 2	<input type="button" class="ui-btn" value="Input 2">
<a> 1	<a> 1
<a> 2	<a> 2
< ve > HTML için özel karakterler olduğundan bağlantı elementleri içinde < ve > özel ifadeleri ile yazılmıştır.	

Tablo 5.7'de görüldüğü gibi button ve input elementleri JQM sisteminde otomatik olarak düğme şeklinde gösterilir. Bağlantı (a) elementi ise yalnızca *ui-btn* sınıfı ile işaretlenmesi halinde düğme olarak gösterilir. Varsayılan olarak *input* elementleri ile oluşturulan düğmelerin kenarları yumuşatılır ve gölge eklenir. *button* ve *a* elementleri ile oluşturulan düğmelerde bu etkiler yoktur. Düğmeler ve bağlantılar benzer görüntüyle sunulabilseler de, *button* ve *input* elementlerinin formlar içinde, *a* elementlerinin ise sayfalar arası bağlantılar için kullanılması önerilmektedir. Örneğin *id* değeri *p2* olan sayfaya yönlendiren bir düğme üretmek için aşağıdaki kod satır yazılabilir.

```
<a href="#p2" class="ui-btn">2. Sayfa</a>
```

Aksi belirtilmediği sürece JQM sisteminde düğmeler blok elementler olarak yorumlanır. Blok elementler bulundukları genişliği kaplayacak şekilde sunulur. Bu davranışı değiştirmek için düğmelere *ui-btn-inline* sınıfı eklenir. Resim 5.18'deki ilk düğme yalnızca ui-btn sınıfı ile, ikinci düğme ise ui-btn ve ui-btn-inline sınıfları ile oluşturulmuştur. Bu düğmeleri oluşturmak için yazılması gereken kodlar şu şekildedir:

```
1   <a href="#p2" class="ui-btn">2. Sayfa</a>
2   <a href="#p2" class="ui-btn ui-btn-inline">2. Sayfa</a>
```

JQM düğmelerde kullanılabilecek bir çok sınıf sunmaktadır. Bu sınıflar elementlere uygulandığında oldukça farklı düğmeler oluşturulabilir.

Resim 5.18 ui-btn-inline Sınıfının Etkisi

Simgeler

JQM arayüz elementlerini daha anlaşılır kılan bir simge kümesi sunmaktadır. Bu simgeler elementlere data-icon özellikleri kullanılarak ya da ui-icon-SİMGEADI şeklindeki sınıfları uygulayarak eklenir. JQM simgelerinin SVG ve PNG versiyonları bulunmaktadır. SVG vektörel bir format olduğundan ölçeklenme işlemlerinden etkilenmez. Bu nedenle JQM öncelikle simgelerin SVG versiyonlarını kullanmayı dener. SVG dosya formatını desteklemeyen sistemlerde otomatik olarak simgelerin piksel tabanlı (PNG) versiyonları kullanılır. Yerel bir proje geliştirmek için JQM sistemini indirdiğinizde bu simgeleri css klasörü içinde görebilirsiniz. css klasöründeki images klasörü içinde yer alan icons-png ve icons-svg klasörleri ilgili dosya uzantısı ile oluşturulmuş simgeleri tutar.

Simgelerin elementlere uygulanmasını bir fotoğraf makinesine benzeyen camera simgesi ile örnekleyelim. Resim 5.19'da solda JQM sistemi ile gelen camera simgesi görülmektedir. Sağ bölümde ise bu simgenin uygulanmasıyla oluşturulmuş üç düğme görülmektedir.

Resim 5.19 JQM Sisteminde Gelen Camera Simgesi ve Bu Simge Kullanılarak Oluşturulan Düğmeler

Düğmeleri oluşturan kodlar aşağıda verilmiştir. Birinci satırda görüldüğü gibi bu düğmelerden ilki, bir a elementine ui-btn sınıfı ve simge ile ilgili sınıfların eklenmesiyle oluşturulmuştur. Bu sınıflardan ui-icon-camera sınıfı düğmeye kamera simgesini eklerken, ui-btn-icon-left sınıfı simgenin düğmedeki pozisyonunu belirlemek için kullanılmıştır. İkinci düğme yine a elementi ile oluşturulmuş fakat bu sefer data-role ve data-icon özellikleri kullanılmıştır. data-role özelliği elementin sayfadaki rolünü belirlemek için kullanılmaktadır.

```
1   <a href="#" class="ui-btn ui-btn-icon-left ui-icon-camera">Düğme</a>
2   <a href="#" data-role="button" data-icon="camera">Düğme</a>
3   <button data-icon="camera">Düğme</button>
```

Bu örneğimizde a elementinin rolü düğme (button) olarak belirlenmiştir. Bunun yanında bu düğmeye simge eklemek için *data-icon* özelliği kullanılmıştır. Elementlere *data-icon* özelliği ile simge eklemek için doğrudan simge adı (ör: camera, alert, audio) kullanılırken, sınıflar yoluyla simge eklemek için bu simge adlarının önüne ui-icon- ön eki getirilir (ör: ui-icon-camera, ui-icon-alert, ui-icon-audio). Örneğimizde son olarak button elementine simge eklenmiştir. Bu kullanımda yalnızca button elementinin *data-icon* özelliğine simge adının verilmesi yeterlidir. Bu kullanımların sonuçları arasında işlev ve görüntü bakımından fark bulunmazken, daha esnek seçenekler sunması nedeniyle simgelere özgü sınıflar kullanılarak düğmelerin düzenlenmesi önerilen bir yaklaşımdır.

Bir düğmeye sınıflar yoluyla simge eklemek için, simgeyi ifade eden sınıf ve simgenin düğmedeki pozisyonunu belirten sınıfın elemente eklenmesi zorunludur. Örneğimizde kullandığımız fotoğraf makinesi simgesinin sınıfı *ui-icon-camera*'dır. Bunun yanında simgenin düğme üzerinde hangi pozisyonda görüneceğini belirten pozisyonlama sınıflarından biri elemente eklenmelidir. Bu sınıflar *ui-btn-icon-POZİSYON* şeklinde isimlendirilmektedir. Bu isimlendirmede pozisyon için yönlerin İngilizce karşılıkları olan *left* (sol), *right* (sağ), *top* (üst) ve *bottom* (alt) kullanılır. Örneğin *ui-btn-icon-top* sınıfı, simgenin düğmenin üstüne yerleştirilmesini sağlar. Resim 5.20'te *ui-btn* sınıfı ile a etiketleri kullanılarak oluşturulmuş olan düğmelere fotoğraf makinesi ikonunun dört yönde yerleştirildiğini görebilirsiniz.

Resim 5.20 Düğmelerde Simge ve Pozisyonlama Sınıflarının Kullanımı

Resimdeki düğmeleri oluşturmak için yazılan kodlar aşağıda verilmiştir.

```
1   <a href="#" class="ui-btn ui-btn-icon-left ui-icon-camera">Düğme</a>
2   <a href="#" class="ui-btn ui-btn-icon-right ui-icon-camera">Düğme</a>
3   <a href="#" class="ui-btn ui-btn-icon-top ui-icon-camera">Düğme</a>
4   <a href="#" class="ui-btn ui-btn-icon-bottom ui-icon-camera">Düğme</a>
```

Simge sınıfları düğmelerde kullanılabilecek diğer sınıflar birleştirilerek oldukça farklı düğmeler elde edilebilir (Resim 5.21).

Resim 5.21 Düğmelerde Kullanılan Sınıflar Eklenerek Çeşitli Şekillerde Düğmeler Oluşturulabilir

Aşağıda verilen kodda tüm düğmeler ui-btn-inline sınıfı kullanılarak satır içi küçük düğmeye çevrilmiştir. Bunun yanında, her bir düğmede farklı sınıflar eklenerek düğme görünümleri değiştirilmiştir.

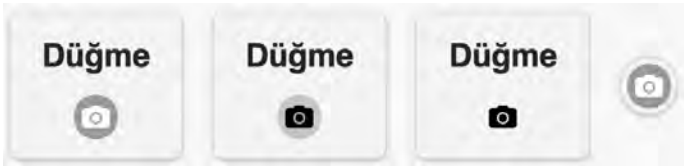
```
1 <a href="#" class="ui-btn ui-btn-inline ui-btn-icon-left ui-icon-camera ui-
  btn-b">Düğme</a>
2 <a href="#" class="ui-btn ui-btn-inline ui-btn-icon-right ui-icon-camera ui-
  btn-b ui-btn-corner-all">Düğme</a>
3 <a href="#" class="ui-btn ui-btn-inline ui-btn-icon-top ui-icon-camera ui-btn-
  corner-all">Düğme</a>
4 <a href="#" class="ui-btn ui-btn-inline ui-btn-icon-bottom ui-icon-camera ui-
  btn-corner-all ui-shadow">Düğme</a>
```

Kodun ilk satırındaki düğmede *ui-btn-b* sınıfı kullanılarak düğmenin koyu temada görüntülenmesi sağlanmıştır. İkinci satırdaki düğmede *ui-btn-corner*-all sınıfı da eklenerek koyu temalı ve kenarları yumuşatılmış bir düğme oluşturulmuştur. Üçüncü ve dördüncü satırlarda *ui-bnt-b* sınıfı kullanılmadığından bu düğmeler açık temada oluşturulmuştur. Üçüncü satırdaki düğmede simge *ui-btn-icon-top* sınıfı ile üste yerleştirilmiş, bunun yanında *ui-btn-corner-all* sınıfı eklenerek düğmenin kenarları yumuşatılmıştır. Son satırdaki düğmede ise *ui-shadow* sınıfı eklenerek düğmeye gölge eklenmiştir.

Düğmelere özgü sınıfların yanında, JQM'in simgelere özgü sınıfları da kullanılarak özelleşmiş düğmeler üretilebilir. Aşağıda verilen kodda simgelere özgü sınıfların eklendiği dört düğme oluşturulmuştur.

```
1 <a href="#" class="ui-btn ui-btn-inline ui-btn-icon-bottom ui-icon-camera ui-
  btn-corner-all ui-shadow">Düğme</a>
2 <a href="#" class="ui-btn ui-btn-inline ui-btn-icon-bottom ui-icon-camera ui-
  btn-corner-all ui-shadow ui-alt-icon">Düğme</a>
3 <a href="#" class="ui-btn ui-btn-inline ui-btn-icon-bottom ui-icon-camera ui-
  btn-corner-all ui-shadow ui-alt-icon ui-nodisc-icon">Düğme</a>
4 <a href="#" class="ui-btn ui-btn-inline ui-icon-camera ui-btn-corner-all ui-
  shadow ui-btn-icon-notext">Düğme</a>
```

İlk düğmede *ui-corner-all* ve *ui-shadow* sınıfları kullanılmıştır. Bunun yanında *ui-btn-icon-bottom* kullanılarak simge düğmenin altına pozisyonlanmıştır. İkinci düğmede simgelere özgü sınıflardan, koyu temalı simge kullanılmasını sağlayan *ui-alt-icon* sınıfının etkisi görülmektedir. Varsayılan olarak simgelerin beyaz renkli versiyonları kullanılırken, *ui-alt-icon* sınıfı kullanılarak koyu renkli simge kullanımı sağlanabilir. Bunun yanında varsayılan olarak simgeler daire içinde gösterilmektedir. Bu dairenin kaldırılması için *ui-nodisc-icon* sınıfı kullanılabilir. Üçüncü düğmede *ui-nodisc-icon* ve *ui-alt-icon* sınıfları birlikte kullanılarak, düğmeye koyu temalı ve çevresinde daire olmayan bir ikon eklenmiştir. Bu kodların sonuçları Resim 5.22'de görülmektedir.



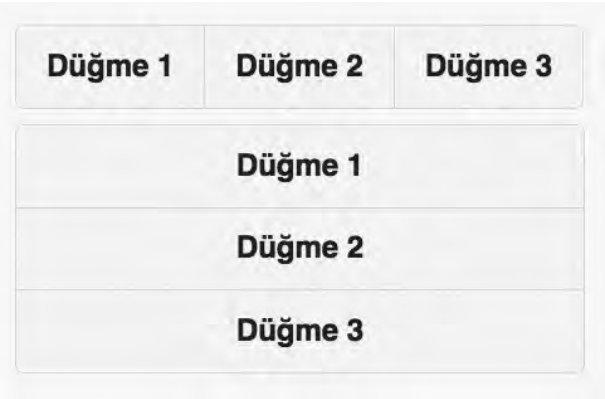
Resim 5.22 JQM'in Simgelere Özgü Sınıflarının Uygulanması

Son olarak arayüzler için oldukça kullanışlı olabilecek, sadece simgelerden oluşan düğmeler üretilebilir. Bu amaçla düğmeye ui-btn-icon-notext sınıfı uygulanır. Bunun yanında düğmedeki pozisyon sınıfı kaldırılmalıdır. Bu sayede yazı alanı görülmeyen sadece simgeden oluşan bir düğme oluşturulabilir.

Düğme Grupları

Belirli bir amaca yönelik düğmeler bir grup halinde sunularak daha kompakt arayüz elementleri oluşturulabilir. Bu amaçla düğmeler data-role özelliği controlgroup olarak belirlenmiş bir gruplayıcı element içine alınır. Düğme grupları yatay ya da dikey şekillerde oluşturulabilir. Bu amaçla gruplayıcı elementin data-type özelliği horizontal (yatay) ya da vertical (dikey) olarak belirlenir.

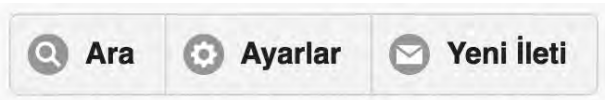
Resim 5.23'te, aşağıda verilen kodlarla oluşturulan iki düğme grubu görülmektedir.



Resim 5.23 Yatay ve Dikey Düğme Gruplarına Örnekler

```
1   <div data-role="controlgroup" data-type="horizontal">
2     <button>Düğme 1</button>
3     <button>Düğme 2</button>
4     <button>Düğme 3</button>
5   </div>
6   <div data-role="controlgroup" data-type="vertical">
7     <button>Düğme 1</button>
8     <button>Düğme 2</button>
9     <button>Düğme 3</button>
10  </div>
```

Birinci satırdaki div elementinin data-role özelliğinin controlgroup, data-type özelliğininse horizontal olduğuna dikkat ediniz. Bu kod, Resim 5.23'te üstte görünen yatay düğme grubunu oluşturmaktadır. Altıncı satırda yer alan div elementinin data-type özelliği vertical olarak belirlendiğinden bu kod alttaki dikey düğme grubunu oluşturmaktadır.



JQM simge referansını kullanarak aşağıdaki araç kutusunu hazırlayalım.

```
1   <div data-role="controlgroup" data-type= "horizontal">
2   <a href="#" class="ui-btn ui-btn-icon-left ui-icon-search">Ara</a>
3   <a href="#" class="ui-btn ui-btn-icon-left ui-icon-gear">Ayarlar</a>
4   <a href="#" class="ui-btn ui-btn-icon-left ui-icon-mail">Yeni İleti</a>
5   </div>
```

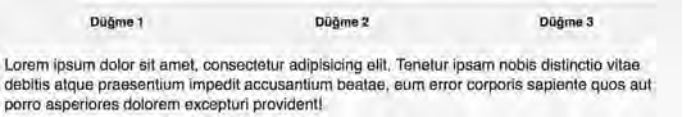
Menü Çubuğu

JQM temel bir menü çubuğu oluşturma aracı sağlar. Bu araç üzerinde yan yana beş adede kadar düğme gösterilebilir. Düğme sayısının beşi geçmesi durumunda araç otomatik olarak düğmeleri birden fazla satıra yaymaya başlar.

Menü çubukları *data-role* özelliği *navbar* olarak belirlenen *div* elementleri içinde oluşturulur. Bu elementlerin içine sırasız bir liste (ul) eklenerek her elementinin (li) içine düğmeleri oluşturacak bağlantı (a) kodları eklenir.

```
1   <div data-role="navbar">
2     <ul>
3       <li><a href="#">Düğme 1</a></li>
4       <li><a href="#">Düğme 2</a></li>
5       <li><a href="#">Düğme 3</a></li>
6     </ul>
7   </div>
```

Yukarıda üç düğmeli bir menü çubuğu oluşturmak için kullanılması gereken genel kodlama şablonu sunulmuştur. Bu kodun çıktısı Resim 5.24'te verilmiştir.



Resim 5.24 JQM Menü Çubuğu Örneği

Menü çubuklarındaki bağlantılar otomatik olarak düğmeye çevrilir. Bu nedenle *ui-btn* gibi sınıfların eklenmesine gerek yoktur. Bununla birlikte çubuktaki düğmelere simge eklemek için simge sınıfları ve data özelliklerinin kullanılması gerekir. Data özelliklerini kullanarak simge eklemek için *data-icon* ve *data-iconpos* özellikleri kullanılır. *data-icon* simgeyi, *data-iconpos* ise simgenin düğmedeki yerini belirlemek için kullanılır. Düğmelere simge eklemek için kullanabileceğiniz sınıfları Simgeler başlığı altında inceleyebilirsiniz. Aşağıdaki kodda her iki yöntemin kullanımı gösterilmektedir.

```
1   <li><a href="#" data-iconpos="top" data-icon="home">Anasayfa</a></li>
2   <li><a href="#" class="ui-btn-icon-top ui-icon-home">Anasayfa</a></li>
```

Her iki yöntemde de *Anasayfa* düğmelerinin üzerinde ev simgesi yerleştirilmiştir. Menü çubuklarındaki düğmelerin gösteriminde kullanılabilecek faydalı bir sınıf *ui-btn-active* sınıfıdır. Bu sınıf bir düğmenin aktif olarak gösterilmesini sağlamaktadır.

Menü çubukları sayfadaki herhangi bir alana eklenebilir. Sayfada istendiği kadar çok menü çubuğu oluşturulabilir. Bununla birlikte, mobil cihazlarda menü çubukları genellikle başlık ya da alt bilgi barları üzerinde sağlanır. Bu amaçla menü çubuklarının *header* ya da *footer* elementlerinin içine yerleştirilmeleri yeterlidir. Aşağıda, sayfadaki alt bilgi barına yerleştirilmiş bir menü çubuğunun kodları görülmektedir.

```
1   <div data-role="footer" data-position="fixed">
2     <div data-role="navbar" data-iconpos="bottom">
3       <ul>
4   <li><a href="#" class="ui-btn-active" data-icon="home"> Anasayfa</a></li>
5         <li><a href="#" data-icon="comment">Hakkımızda</a></li>
6         <li><a href="#" data-icon="grid">Projelerimiz</a></li>
7        <li><a href="#" data-icon="mail">İletişim</a></li>
8        <li><a href="#" data-icon="gear">Ayarlar</a></li>
9       </ul>
10    </div>
11  </div>
```

Birinci satırdaki div elementinin *data-role* özelliği *footer* olarak ayarlandığından çubuk sayfanın altında görülecektir. Bunun yanında *data-position* özelliği *fixed* olarak belirlendiğinden çubuk sayfa uzunluğuna bakmadan sürekli ekranın altında gösterilecektir. İkinci satırdaki *div* elementinin *data-role* özelliği navbar olarak belirtilerek element içeriklerinin menü çubuğuna çevrilmesi sağlanmıştır. Ek olarak bu elementte tüm düğmelerdeki simgelerin alta hizalanmasını sağlamak için *data-iconpos* özelliği *bottom* olarak belirlenmiştir. Bu kullanım bir kısa yoldur. Bunun yerine her bir düğmede *data-iconpos* özelliği kullanılarak simgelerin farklı noktalara yerleştirilmesi sağlanabilir. Bunun ardından ul elementlerinin içine yerleştirilen

a elementleri ile düğmeler oluşturulmaktadır. Burada düğmelere *data-icon* özelliği kullanılarak simgeler eklenmiştir. Ek olarak dördüncü satırdaki düğmeye *ui-btn-active* sınıfının eklendiğine dikkat ediniz. Bu sayede bu düğme aktif olarak gösterilerek diğerlerinden ayrılmıştır. Kodun çıktısı Resim 5.25'te verilmiştir.

Resim 5.25 Alt Bilgi Alanında JQM Menü Çubuğu Örneği

Listeler

Sayfa içeriklerinin gösteriminde en önemli araçlardan biri listelerdir. JQM listeler ile ilgili pek çok özellik sunmaktadır. Resim 5.26'da görüldüğü gibi, HTML dilinin sunduğu sırasız ve sıralı listeler müdahale edilmediği sürece sayfalarda düzenlenmeden gösterilir. Listeler, *data-role* özellikleri *listview* olarak belirlendiğinde JQM listelerine çevrilir.

Aşağıda verilen kodun ikinci satırında sırasız bir liste tanımlanmıştır. Bu listeye müdahale edilmediğinden düz bir HTML listesi olarak görünecektir. Sekizinci satırda tanımlanan listenin data-role özelliği listview olarak belirlendiğinden bu liste bir JQM listesidir.

Resim 5.26 HTML Listeleri ve JQM Listesi

```
1   <h2>HTML sırasız listesi</h2>
2   <ul>
3       <li>Liste 1</li>
4       <li>Liste 2</li>
5       <li>Liste 3</li>
6   </lu>
7   <h2>JQM sırasız listesi </h2>
8   <ul data-role= "listview">
9       <li>Liste 1</li>
10      <li>Liste 2</li>
11      <li>Liste 3</li>
12  </lu>
13  <h2>JQM sırasız listesi </h2>
14  <ul data-role= "listview" data-inset= "true">
15      <li>Liste 1</li>
16      <li>Liste 2</li>
17      <li>Liste 3</li>
18  </lu>
```

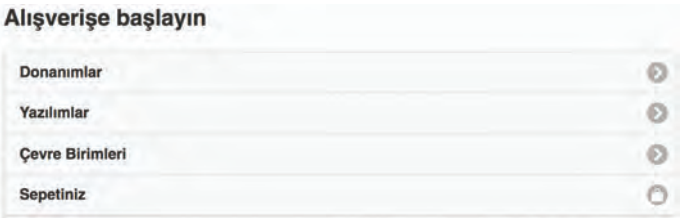
Kod 52. *JQM ile sırasız liste oluşturmak*

JQM listeleri okunurluğu desteklemek için koyu yazı, alt çizgiler gibi pek çok tipografik özellikle sunulur. Varsayılan olarak liste elementleri sayfayı kaplayacak şekilde düzenlenmiştir. Listelerin sayfada daha içeriden gösterilmesi için data-inset özelliğinin true olarak belirlenmesi yeterlidir. Yukarıdaki kodun 14. satırındaki JQM listesinin *data-inset* özelliği *true* olarak düzenlendiğinden bu liste bir önceki listeye göre daha farklı bir sunuma sahiptir.

Şimdiye kadar salt okunur listeler ürettik. Liste elementlerinin içine bağlantılar yerleştirilerek etkileşimli listeler üretilebilir. Bu listeler sayfalar arası dolaşım için kullanılabilir. Aşağıdaki kodun ikinci satırında bir JQM listesi tanımlanmıştır. *data-inset* özelliği true olarak belirlendiğinden bu liste sayfada içeriden gösterilecektir.

```
1   <h2>Alışverişe başlayın</h2>
2   <ul data-role="listview" data-inset="true">
3     <li><a href="#">Donanımlar</a></li>
4     <li><a href="#">Yazılımlar</a></li>
5     <li><a href="#">Çevre Birimleri</a></li>
6     <li data-icon="shop"><a href="#">Sepetiniz</a></li>
7   </ul>
```

Bu listedeki dört elementin içine bağlantılar yerleştirildiğinden, bunlar etkileşimli elementler olarak sunulacaktır. Bu elementler uygulama içinde başka bir sayfaya yönelim, bir açılır kutu açma ya da diyalog penceresi açma gibi herhangi bir görev için kullanılabilir. JQM listelerinde etkileşimli elementler varsayılan olarak sağ taraflarında bir ok ile temsil edilir. Bunun yanında *li* elementinin *data-icon* özelliğine bir simge ismi verilerek bu simgeler düzenlenebilir. Örneğimizde altıncı satırdaki "Sepetiniz" bağlantısının simgesi alışveriş sepeti olarak değiştirilmiştir. Kodun çıktısı Resim 5.27'de verilmiştir.



Resim 5.27 Etkileşimli Bir JQM Listesi

JQM listelerindeki önemli bir etkileşim aracı da filtrelerdir (Resim 5.28). JQM listelerinin üzerine bir arama kutusu eklenerek, listenin bu kutuya girilen değere göre filtrelenmesi sağlanabilir. Bu araç özellikle uzun listelerde kullanıcının aradığını bulmasına yardımcı olabilir.

Resim 5.28 Filtrelenmiş JQM Listesi Örneği

```
1   <ul data-role= "listview" data-inset= "true" data-filter= "true" data-filter-
    placeholder= "Ders ara..."
2   <li><a href="#">Matematik</a></li>
3   <li><a href="#">Fen Bilgisi</a></li>
4   <li><a href="#">Türkçe</a></li>
5   <li><a href="#">Sosyal Bilgiler</a></li>
6   <li><a href="#">Coğrafya</a></li>
7   </ul>
```

JQM listelerine filtre eklemek için liste elementinin *data-filter* özelliğinin true olarak belirlenmesi yeterlidir. Varsayılan olarak filtre kutularında "Filter items …" *placeholder* metni yer almaktadır. Bu metni değiştirmek için *data-filter-placeholder* özelliğine istenen değer atanabilir. Yukarıdaki kod örneğinde bir ders listesi yer aldığından, *placeholder* metni "Ders ara …" şeklinde düzenlenmiştir.

Çok uzun listelerin sayfalarda gösterilmesi uygun olmayabilir. Bu tür listeler için yalnızca arama kutusunun gösterildiği ve arama kutusuna bir değer girildiğinde eşleşen satırların gösterildiği listeler de üretilebilir. Bu amaçla listeye *data-filter-reveal="true"* ifadesinin eklenmesi yeterlidir.

Ders listesi örneğini normalde satır gösterilmeyen, yalnızca kullanıcı giriş yaptığında eşleşen satırları gösteren forma çevirelim.

```
1  <ul data-role="listview" data-inset="true" data-filter-reveal="true" data-
   filter="true" data-filter-placeholder="Ders ara ...">
2  <li><a href="#">Matematik</a></li>
3  <li><a href="#">Fen Bilgisi</a></li>
4  <li><a href="#">Türkçe</a></li>
5  <li><a href="#">Sosyal Bilgiler</a></li>
6  <li><a href="#">Coğrafya</a></li>
7  </ul>
```

Günümüz web teknolojisinin bel kemiği araçlarından biri olan JS kullanımı kolay bir betik yazma dilidir. Bunun yanında, dünya çapında programcıların desteği ve katkıları ile geliştirilmiş pek çok JS kütüphanesi bulunmaktadır. Bu kütüphaneler sayesinde JS'in yetenekleri önemli derecede büyümekte, programcıların iş yükleri önemli derecede azalmaktadır. Bu ünitede önemli JS kütüphanelerinden JQuery ve JQM tanıtılmaya çalışılmıştır. Bununla birlikte bu kütüphanelerin tüm yeteneklerinden bahsedilemediğini belirtmekte fayda vardır. İnternet, ilgili bir programcının herhangi bir kütüphaneyi öğrenmesi için pek çok kaynak ve kütüphane sağlamaktadır. Yakın geçmişte Türkçe içeriklerin sayısının da hızla arttığı görülmektedir. Amaçları ve kapsamları farklı onlarca JS kütüphanesi bulunmakta ve sayıları günden güne artmaktadır. Size düşen bu konuda meraklı olmak ve güncel gelişmeleri takip etmektir.

Öğrenme Çıktısı

6 JQuery mobile kullanarak bir kullanıcı arayüzü geliştirebilme

Araştır 4

İki sayfalık bir arayüz üreterek sayfalar arasında kayarak geçiş gerçekleştiriniz. İkinci sayfadan birinci sayfaya dönerken slide efektinin tersten oynatılmasını sağlayınız.

İlişkilendir

JQM tek sayfalık uygulamalarını yenileme yapmadan arka planda bir sunucudan veri çekme açısından AJAX ile ilişkilendiriniz.

Anlat/Paylaş

Mobile uygulamalarını özellikleri ile birlikte paylaşınız.

öğrenme çıktıları ve bölüm özeti

1 JS kütüphanesi kullanmanın faydalarını açıklayabilme

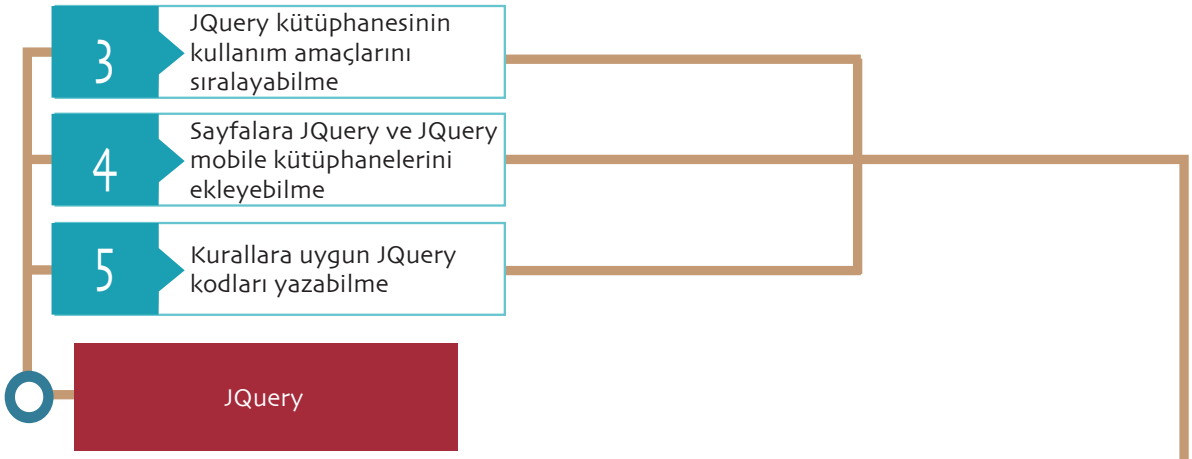
Neden JS Kütüphaneleri

1 JS kütüphanesi kullanmanın faydalarını dört başlık altında toplayabiliriz. Bunlar; tekrar kullanılabilirlik, ücretsiz yardım kanalları, yapısallık ve genişletilebilirliktir. Tekrar kullanılabilirlik ile kendini ispatlamış kodları projeye entegre ederek zaman kazanılır. Bazılarının kullanıcı sayısı milyonlara ulaşan kütüphaneler için hazırlanmış öğreticiler, sorunlara çözümler, geliştirici ağları gibi ücretsiz yardım kanallarından yararlanılır. Kütüphane kullandığınızda MVC gibi güncel yazılım geliştirme yaklaşımlarını kullanarak piyasadaki güncel yapısal eğilimlere ayak uydurulur. Son olarak kütüphaneler sürekli geliştirilen ve güncellenen yapılar olduğundan, yeni versiyonlarla birlikte hiçbir efor sarf etmeden yeni özellikler eklenmesi ve genişletilmesi mümkündür.

2 JS kütüphanelerinin kullanım alanlarını açıklayabilme

Önemli JS Kütüphaneleri

2 JS kütüphaneleri internet teknolojisi ile ilgili herhangi bir alanda kullanılabilmektedir. Bunlar arasında etkileşimli bir web uygulaması için temel sağlamak, internet tarayıcısının yeteneklerini keşfetmek, HTML DOM düzenlemek, oyun geliştirme araçları sağlamak, araç sınıfları sağlamak, hazır arayüz elementleri sağlamak gibi amaçlar sıralanabilir.



3 JQuery kütüphanesinin temel işlevleri; HTML/DOM düzenleme (Sayfadaki HTML DOM üzerinde ekleme, silme ve düzenleme işlemleri gerçekleştirilebilir), CSS düzenleme (Sayfadaki elementlerin CSS stilleri üzerinde düzenleme yapılabilir), HTML olaylarına tepki verme (HTML'nin sağladığı tüm olaylara tepki verecek fonksiyonlar hazırlanabilir) Görsel efekt ve animasyonlar (Bir elementi silikleştirerek silme, bulunduğu ortamda sağa sola hareket ettirme gibi pek çok hazır görsel efekt ve animasyon sağlanmaktadır) ve AJAX (Sayfayı yenilemeden arka planda bir sunucudan veri çekme, sunucuya veri gönderme ve alınan sonuçlara göre sayfa içeriğini düzenleme işlemleri yapılabilir) şeklinde sırlanabilir.

4 Sayfalarınıza JQuery ve JQuery Mobile eklemek için uzak ve yerel olmak üzere iki yöntem kullanabilirsiniz. Uzak yöntemde dosyalar bir İDA üzerinde tutulur ve sayfa çalıştırıldığında kullanıcı bilgisayarına otomatik olarak indirilir. Yerel yöntemde ise dosyalar kendi sunucunuzda tutulmaktadır. Her iki kütüphaneyi uzak yöntemle kullanmak için gerekli dosyaları https://code.jquery.com/ adresindeki bağlantıları kullanarak edinebilirsiniz. JQuery'i yerel yöntemle kullanmak için https://jquery.com/download/ adresinden indirebilirsiniz. Benzer şekilde, JQM dosyaları da yerel projelerde kullanılmak üzere https://jquerymobile.com/download/ adresinden edinilebilir.

5 JQuery'nin temelinde JS dili bulunduğundan yazım kurallarının büyük bir kısmı bu dilden miras alınmıştır. JQuery kodları bir seçici ve bu seçiciye uygulanacak metot şeklinde yazılır. JQuery seçici ifadelerini CSS teknolojisinden miras almıştır. Bir seçici oluşturmak için öncelikle $ ile JQuery aktifleştirilir. Ardından parantezler içindeki tırnaklar arasında CSS seçici ifadesi yazılır. Örneğin sayfadaki tüm bağlantıları seçmek için $("a"), hata sınıfı eklenmiş tüm elementleri seçmek için $(".hata"), id'si google olan bir bağlantıyı seçmek için $("a#google") seçicileri yazılabilir. Bir seçici oluşturduktan sonra bu seçicinin seçtiği elementlere uygulanacak metotlar yazılır. Örneğin elementi gizlemek için kullanılan hide metodu $("a#google").hide() ifadesi ile çağırılabilir. Bu ifadede parametre olarak metodun uygulanma süresi ve efektin uygulanmasının bitiminde çalıştırılacak bir uygulama fonksiyonu yazılabilir. Elementlerin olaylarına fonksiyon bağlamak için on metodu kullanılır. Örneğin bir elementin tıklanması halinde çalışması istenen kodlar $("p").on("click", function(){ … }) ifadesindeki noktalı alana yazılır. on metoduna gönderilen ilk parametre elementin yayınladığı olayı (bu durumda tıklama - click) belirtmektedir. İkinci parametre yine uygulama fonksiyonumuzdur.

öğrenme çıktıları ve bölüm özeti

6 JQuery mobile kullanarak bir kullanıcı arayüzü geliştirebilme

JQuery Mobile

6 JQM, öncelikle mobil ekranları hedef alan uygulama arayüzleri üretir. JQM'in çalışma mantığı tek sayfalık uygulamalar üretmek üzerine kurulmuştur. Bu nedenle sayfalar içinde bir etiket sistemi ile sanal sayfalar üretilir. JQM çalışma mantığına göre uygulama bir anda yalnızca bir sayfayı gösterir. Kullanıcı etkileşimi ile sayfalar arasında gezinim sağlanabilir. Sayfalara verilen id değerleri, başlarına # eklenerek sayfalara yönlenen bağlantılar için referans olarak kullanılır. JQM sistemi sayfaların dışında kullanıcı etkileşimi için üç adet kutu mekanizması sağlamaktadır. Diyaloglar kullanıcıya mesaj vermek ve kullanıcıdan geribildirim almak için kullanılabilir. Açılır kutular genellikle kullanıcıya bildirimde bulunmak için kullanılır. Oldukça esnek bir araç olan paneller ise sayfaların kenarlarına yerleştirilir. Paneller, menüler, listeler, inceleme araçları gibi çeşitli etkileşim araçları üretmek için kullanılabilir. JQM sayfalar ve kutuların gösterilmesi için pek çok geçiş efekti sağlar. Bu efektler sayfayı ya da kutuyu çağıran bağlantı üzerinde seçilir. JQM kullanıcı arayüzü oluşturmak için pek çok element ve sistemle birlikte gelmektedir. Bu ünitede düğmeler, simgeler, düğme grupları, menü çubuğu ve listelerden bahsedilmiştir. Bununla birlikte referansları inceleyerek diğer arayüz elementlerini nasıl kullanabileceğiniz öğrenebilirsiniz.

1 Aşağıdakilerden hangisi projelerde Javascript kütüphanesi kullanmanın faydalarından biri **değildir?**

A. Kendini kanıtlamış kodların tekrar kullanımı
B. Geliştiriciler ve kullanıcıların paylaştığı ücretsiz yardım dokümanları
C. Yeni versiyonlarla birlikte gelen genişletmelerden yararlanabilme olanağı
D. Güncel yazılım geliştirme yaklaşımlarına uyum sağlayabilme
E. Kütüphanelerin küçük çaplı projelerin ihtiyaçlarının ötesinde çok fazla özellik sunması

2 Projelerde sıklıkla kullanılan genel amaçlı Javascript fonksiyonlarını sağlayan kütüphane aşağıdakilerden hangisidir?

A. UnderscoreJS
B. AngularJS
C. D3.js
D. React
E. Bootstrap

3 Aşağıdakilerden hangisi JQuery'nin yetenekleri arasında **yer almaz?**

A. Görsel efekt ve animasyonlar
B. HTML olaylarına tepki vermek
C. CSS düzenlemek
D. Hazır arayüz elementleri sunmak
E. HTML DOM düzenlemek

4 Bir sayfadaki paragraf elementlerine saklama metodunu 200 milisaniye içinde uygulamak için yazılması gereken JQuery kodu aşağıdakilerden hangisidir?

A. $(seçici).metot();
B. $("p").hide(200);
C. $(p).hide(2000);
D. $(200).p.hide();
E. $(p).hide();

5 Bir sayfada alt sınıfı eklenmiş tüm elementlerin tıklanma olaylarına tepki vermek için yazılması gereken kod aşağıdakilerden hangisidir?

A. $(p.alt).on("click", function(){…});
B. $(alt).on("press", function(){…});
C. $("p.alt").on("click", function(){…});
D. $(".alt").on("press", function(){…});
E. $(".alt").on("click", function(){…});

6 JQuery ile HTML DOM düzenleme metotlarının doğru çalışması için kodların nereye yazılması gerekir?

A. $(document).click() olayına
B. HTML dokümanının HEAD bölümüne
C. $(document).ready() olayına
D. $(form).submit() olayına
E. $(window).unload() olayına

7 Bir HTML elementinin özelliklerine JQuery ile erişmek için hangi metot kullanılır?

A. attr
B. val
C. text
D. html
E. hide

8 JQuery Mobile sisteminde elementlerin uygulamadaki işlevleri hangi özellikleri ile belirlenir?

A. data-position
B. data-icon
C. rel
D. data-role
E. listview

9 JQuery mobile sisteminde id özelliği kutu1 olan bir açılır kutuyu hedef alan bir bağlantının kodu aşağıdakilerden hangisidir?

A. Örnek
B. Örnek
C. Örnek
D. Örnek
E. Örnek

10 Aşağıdakilerden hangisi JQuery mobile sisteminde kullanılabilecek bir geçiş efekti **değildir?**

A. Flip
B. Slide
C. Click
D. Pop
E. Turn

neler öğrendik yanıt anahtarı

1. E	Yanıtınız yanlış ise "Neden Javascript Kütüphaneleri" konusunu yeniden gözden geçiriniz.
2. A	Yanıtınız yanlış ise "Önemli JS Kütüphaneleri" konusunu yeniden gözden geçiriniz.
3. D	Yanıtınız yanlış ise "JQuery" konusunu yeniden gözden geçiriniz.
4. B	Yanıtınız yanlış ise "JQuery Yazım Kuralları" konusunu yeniden gözden geçiriniz.
5. E	Yanıtınız yanlış ise "JQuery Olayları" konusunu yeniden gözden geçiriniz.
6. C	Yanıtınız yanlış ise "JQuery Olayları" konusunu yeniden gözden geçiriniz.
7. A	Yanıtınız yanlış ise "JQuery DOM Metotları" konusunu yeniden gözden geçiriniz.
8. D	Yanıtınız yanlış ise "JQM Sayfaları" konusunu yeniden gözden geçiriniz.
9. B	Yanıtınız yanlış ise "Açılır Kutular" konusunu yeniden gözden geçiriniz.
10. C	Yanıtınız yanlış ise "JQM Geçiş Efektleri" konusunu yeniden gözden geçiriniz.

5 Araştır Yanıt Anahtarı

Araştır 1

JS Kütüphanesinin güncellenmesi öncelikle kütüphanedeki güvenlik açıklarının giderilmesini ve bu kütüphane üzerine inşa ettiğiniz uygulamanın da saldırılara karşı güncel koruma altında olmasını sağlar. Ayrıca kütüphane güncellemeleriyle önceki sürümlerdeki hatalar giderilir ve yeni bileşenler eklenir. Böylelikle uygulamanız veya Web siteniz profesyonel bir ekip tarafından sürekli geliştirilmiş, iyileştirilmiş olur.

Araştır 2

Bu ünitede açıklanan JS kütüphanelerinin dışında Backbone.js, Ember.js, Can.js, Agility.js, Stapes.js ve Snack.js gibi çok sayıda farklı JS kütüphanesi bulunmaktadır.

Araştır 3

```
1  <style>
2    input, button{
3      font-size:1em;
4    }
5    .hatali {
6      background-color: #fdd;
7      border-color: red;
8    }
9    .gecerli {
10     background-color: #dfd;
11     border-color: #0f0;
12   }
13 </style>
14 <input id="eposta" type="email" placeholder="e-osta adresiniz">
15 <button id="kontrol">Kontrol</button>
16 <script>
17 $(document).ready(
18   $("#kontrol").on("click", function(){
19         var pattern = new RegExp(/^[+a-zA-Z0-9._-]+@
   [a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/i);
20     $posta=$("#eposta").val();
21     if(pattern.test($posta))
22     {
23 $("#eposta").attr("placeholder", "e-posta adresi geçerli");
24       $("#eposta").removeClass("hatali");
25       $("#eposta").addClass("gecerli");
26     }
27     else
28     {
29 $("#eposta").attr("placeholder", "e-posta adresi geçersiz");
30 $("#eposta").removeClass("gecerli");
31 $("#eposta").addClass("hatali");
32     }
33   })
34 );
35 </script>
```

5 Araştır Yanıt Anahtarı

Araştır 4

```
1  <div id="p1" data-role="page">
2    <div data-role="header">
3      <h1>Birinci Sayfa</h1>
4    </div>
5    <div data-role="main" class="ui-content">
6      <p>Merhaba</p>
7  <p><a href="#p2" data-transition= "slide">İkinci sayfaya
   git.</a></p>
8    </div>
9  </div>
10 <div id="p2" data-role="page">
11   <div data-role="header">
12     <h1>İkinci Sayfa</h1>
13   </div>
14 <div data-role="main" class="ui-content">
15     <p>Merhaba</p>
16 <p><a href="#p1" data-transition= "slide" data-direction=
   "reverse"> Birinci sayfaya git.</a></p>
17   </div>
18 </div>
```

Kaynakça

Sayar, R. (2015). *Top JS Frameworks, Libraries and Tools and When to Use Them.* https://www.sitepoint.com/top-JS-frameworks-libraries-tools-use/ adresinden 16.02.2017 tarihinde erişilmiştir.

jquerymobile.com (t.y.). *Demos.* http://demos.jquerymobile.com/1.4.5/ adresinden 19.02.2017 tarihinde erişilmiştir.

w3schools.com (t.y.). *JQuery tutorial.* https://www.w3schools.com/jquery/default.asp adresinden 30.12.2016 tarihinde erişilmiştir.

w3schools.com (t.y.). *JQuery mobile tutorial.* https://www.w3schools.com/jquerymobile/default.asp adresinden 10.01.2017 tarihinde erişilmiştir.

Bölüm 6

ASP.NET ile Dinamik Web Kodlama

öğrenme çıktıları

1 ASP.NET

1 ASP.NET'in ve .NET Framework'ün çalışma prensiplerini açıklayabilme

2 Kurulum

2 ASP.NET kurulum gereksinimlerini sıralayabilme

3 C# Programlama Dili

3 Değişken türleri ve tür dönüşümlerini kullanabilme

4 Dizileri ve ArrayList'leri kullanabilme

5 Kontrol yapılarını kullanabilme

6 Sınıf tanımlaması yapabilme

4 WEB Formları

7 Web Form kullanarak uygulama geliştirebilme

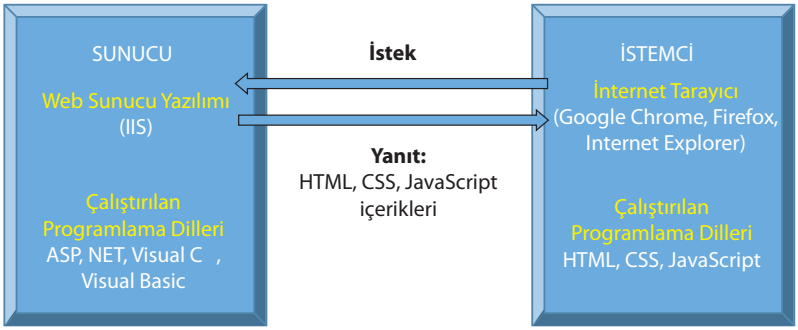
8 Veritabanı işlemleri yapabilme

Anahtar Sözcükler: • ASP.NET • .NET Framework • Değişken Türleri • Tür Dönüşümleri • Operatörler • Kontrol Yapıları • Diziler • ArrayList • If • For • While • Do While • Foreach • Switch • Class • Web Form • SQL Server

GİRİŞ

.NET Framework, içinde bir çok teknolojiyi barındıran Microsoft firmasına ait bir uygulama geliştirme platformudur. ASP.NET ise bu platformun bir parçasıdır ve web uygulamaları geliştirmek için kullanılmaktadır. Microsoft sunucu teknolojilerini kullanan geliştiricilerin birinci tercihinin ASP.NET platformu olduğu söylenebilir. ASP.NET ile PHP benzer bir çalışma mantığına sahiptir. ASP.NET ile geliştirilen uygulamalar çoğunlukla bir web sunucu yazılımı olan Internet Information Server (IIS) üzerinde çalıştırılmaktadır. ASP.NET uygulamalarını Linux, Unix ve macOS gibi işletim sistemleri üzerinden farklı web sunucu yazılımları ile sunmak da mümkündür. Ancak ASP.NET bir Microsoft ürünü olduğu için ve en sorunsuz olarak Microsoft işletim sistemleri ile çalıştığı için çoğunlukla Microsoft Windows Server işletim sistemleri üzerinde IIS web sunucu yazılımının tercih edildiği söylenebilir.

Aynı PHP'de olduğu gibi ASP.NET uygulamaları da bir sunucu üzerinde çalıştırılmakta ve bu uygulamanın ürettiği web sayfaları (HTML, CSS ve JavaScript) istemcilere (internet tarayıcılarına) sunulmaktadır. Internet tarayıcılar ASP.NET'in kodlarını yorumlayamazlar. Dolayısıyla ASP.NET uygulamalarının bir web sunucu (IIS) üzerinde çalıştırılması gerekmektedir. ASP.NET, .NET platformunun bir parçası olduğu için .NET programlama dillerinden herhangi biri ile ASP.NET uygulamaları geliştirmek mümkündür. Bu dillerden en çok kullanılanı ise C# programlama dilidir. Bu ünitede kullanılan örnek kodlarda C# programlama dili kullanılmıştır. Şekil 6.1'de ASP.NET uygulamasının çalıştırılması ile ilgili sunucu-istemci iletişimi gösterilmiştir.



Şekil 6.1 Sunucu - İstemci İletişimi

Şekilde de görüldüğü üzere, öncelikle İnternet tarayıcı web sunucuya bir istekte bulunmaktadır. Web sunucu ise bu istek doğrultusunda eğer istenen dosya bir HTML, CSS veya JavaScript dosyası ise, bu dosyaları hiçbir işlem yapmadan doğrudan İnternet tarayıcıya geri gönderir. Ancak istenen dosya bir ASP.NET dosyası ise bu dosyanın içeriği doğrudan İnternet tarayıcıya göndermez. Gönderilse de İnternet tarayıcı bu içeriği çalıştıramaz. Bunun yerine ASP.NET dosyası sunucu üzerinde çalıştırılır ve bu programın üretmiş olduğu içerikler (çoğunlukla bir HTML içeriğidir) İnternet tarayıcıya gönderilir. ASP.NET dosyalarının sunucu tarafında çalışması sayesinde veritabanı bağlantısı, e-posta sunucusu ile iletişim, banka kredi kartı işlemleri gibi çok sayıda işlem yapılarak kullanıcılara dinamik içerikler sunulabilmektedir. ASP.NET platformu içinde birçok özelliği, binlerce sınıftı barındıran çok büyük bir platformdur. Bu nedenle, bu platformun tüm özelliklerini tek bir ünitede işlemek mümkün değildir. Bu ünitede C# programlama dili ve ASP.NET platformu ile nasıl uygulama geliştirilebileceği ile ilgili bilgi verilmiştir.

ASP.NET

ASP.NET öncesinde ASP programlama dili bulunmaktaydı. ASP programlama dili de Microsoft tarafından geliştirilmiş ve ASP.NET gibi IIS üzerinde çalışan sunucu temelli bir programlama diliydi. ASP, sadece vbscript adı verilen bir programlama dili ile yazılmaktaydı. Geçmişte uzun seneler kullanılmış olmasına rağmen, günümüz ko-

> **ASP.NET**
> .NET Framework yapısı altında bulunan ve Microsoft firmasının sunucu temelli uygulama geliştirme çözümüdür.

şullarında ASP'nin oldukça ilkel bir dil olduğu söylenebilir. Hali hazırda ASP'nin kullanımı tamamen terkedilmiştir. ASP ile ASP.NET dilleri isim benzerliği olmasına rağmen aslında tamamen farklı programlama dilleridir.

Microsoft .NET Framework, Microsoft tarafından geliştirilmiş bir uygulama geliştirme platformudur. .NET Framework'ün geliştirilmesinde en önemli nokta, geliştirilen yazılımların her türlü işletim sistemi ve donanım üzerinde çalışabilir olmasıdır. Bir başka deyişle .NET Framework; işletim sistemi ve donanım ile çalışan yazılım arasında bir ara katman olarak devreye girmektedir. Aslında bu ara katman olarak çalışan yapıya .NET Framework'in bir parçası olan Common Language Runtime (CLR) adı verilmektedir. Bu bakımdan .NET Framework için geliştirilen yazılımlar doğrudan işletim sistemi ve donanım ile iletişim kuramamakta, bunu CLR araçlarını kullanarak yapabilmektedirler. Dolayısıyla .NET ile geliştirilmiş bir yazılımın çalışabilmesi için o sistemde mutlaka .NET Framework'ün kurulu olması gerekir. Bu bakımdan Java Runtime Environment (JRE) ile benzer bir çalışma mantığına sahip olduğu söylenebilir. Java programlama dili ile geliştirilen uygulamaların bir işletim sisteminde çalışabilmesi için mutlaka o işletim sisteminde JRE'nin kurulu olması gerekir. Benzer şekilde Java ile geliştirilen yazılımlar da işletim sistemi ve donanımdan bağımsız olabilmektedir.

.NET dilleri ile geliştirilen yazılımlar derlendiğinde Intermediate Language – Ara Dil (IL) isminde bir dile dönüştürülmektedir. Tüm .NET dilleri sonuçta aynı yapıya dönüştüğü için ortak bir yapıya sahiptirler. Bu yapıya *managed code* adı da verilmektedir. Visual C++ gibi diller ile *managed code* harici ve .NET CLR ile çalışmayan yazılımlar da geliştirilebilmektedir. Ancak bu çokta tercih edilen bir durum değildir. Derleme işleminden sonra IL olarak oluşturulan kodlar paketlenerek bir DLL veya EXE dosyasına dönüştürülmektedir. Derlenmiş EXE veya DLL dosyalarının çalıştırılmasında ise CLR devreye girmektedir. Dosya ilk kez çalıştırıldığında kodlar paketten çıkartılarak CLR'nin Just In Time (JIT) derleyicisi tarafından tekrar derlenir ve bulunduğu sistemin makine diline dönüştürülür. Dolayısıyla bir .NET yazılımı ilk kez yeni bir sistemde çalıştırıldığında daha çok zaman alacaktır, çünkü IL kodları makine diline derlenmektedir. İkinci ve daha sonraki çalışmalarda ise CLR tekrar derleme yapmaz, derlenmiş olan yazılımın çalıştırılmasını sağlar. C#, Visual Basic .NET, Managed Visual C++, Delphi .NET, J# gibi diller ile .NET Framework için yazılım geliştirilebilmektedir.

.NET Framework'ün önceki sürümleri sadece Microsoft Windows işletim sistemleri üzerinde çalışmaktaydı. Bu durumda .NET ile geliştirilmiş yazılımlar da sadece Microsoft Windows işletim sistemleri üzerinde çalışabilmekteydi. Bunun dışında açık kaynak kodlu *mono* projesi ile .NET uygulamalarının Linux ve macOS işletim sistemleri ile de çalışabilmesi sağlanmıştır. Mono aynı zamanda .NET dilleri ile yazılım geliştirmeye de yarayan araçlar sunmaktadır. Daha fazla bilgi için mono projesinin resmi web sitesi ziyaret edilebilir.

Günümüzde ise Microsoft, .NET Core isminde .NET Framework'ün çekirdeği olarak değerlendirilebilecek bir sürümü açık kaynak kodlu olarak yayınlamaktadır. Aynı zamanda .NET Core resmi olarak Linux, macOS ve Windows işletim sistemlerini desteklemektedir. Bu bakımdan.NET Core ile geliştirilen bir yazılımın çok çeşitli işletim sistemleri ve donanımlar üzerinde çalışabileceği söylenebilir. .NET Core'un çalışması da .NET Framework gibidir. JIT derleyicisi .NET Framework'te olduğu gibi .NET Core'da da bulunmaktadır. Microsoft resmi web sitesinde Visual Studio'nun 2017 sürümünde .NET Core için yazılım geliştirmeye yönelik araçlar sunacağını belirtmiştir. Ayrıca Microsoft tarafından yayınlanan bir yazıda ASP.NET'in artık ASP.NET Core olarak adlandırılacağı belirtilmiştir. Bu gelişmeler sonucunda ASP.NET Core uygulamalarını Linux/Unix sunucular üzerinde çalıştırabilmek mümkün hale gelmektedir. Bu durum web uygulama geliştiricileri için oldukça önemlidir. Açık kaynak kodlu birçok web uygulaması PHP ile geliştirilmekte ve genellikle Linux/Unix sunucular üzerinden hizmet vermektedir. ASP.NET Core'un farklı işletim sistemlerinde çalışabiliyor olmasıyla ASP.NET uygulama geliştiricileri geliştirdikleri yazılımları farklı işletim sistemlerinde yayınlayabilirler. Bu durum ASP.NET'in daha da yaygınlaşmasını sağlayabilir.

ASP.NET Core ile şimdilik sadece web uygulaması geliştirilebilmektedir. Web servis (WPF) uygulamaları geliştirmek mümkün değildir. Ayrıca ASP.NET Core, CoreFX sınıf kütüphanesini kullanmakta ve CoreCLR ismindeki .NET Core'un çalışma ortamı üzerinde çalışmaktadır. Ayrıca mobil işletim sistemleri olan iOS ve Android işletim sistemlerine yönelik olarak Xamarin isminde NET Framework'ün bir sürümü daha bulunmaktadır. Bu sürüm ile .NET dilleri ile mobil işletim sistemlerine yönelik uygulama geliştirmek de mümkündür.

Öğrenme Çıktısı

1 ASP.NET'in ve .NET Framework'ün çalışma prensiplerini açıklayabilme

Araştır 1

ASP.NET'in ASP.NET Core olarak adlandırılan yeni sürümünün özelliklerini araştırınız.

İlişkilendir

Ücretli uygulamaları ile bilinen Microsoft'un ASP.NET'i ASP.Net Core'a dönüştürerek ücretsiz hale getirmesini açık kaynak trendi ile ilişkilendirerek açıklayınız.

Anlat/Paylaş

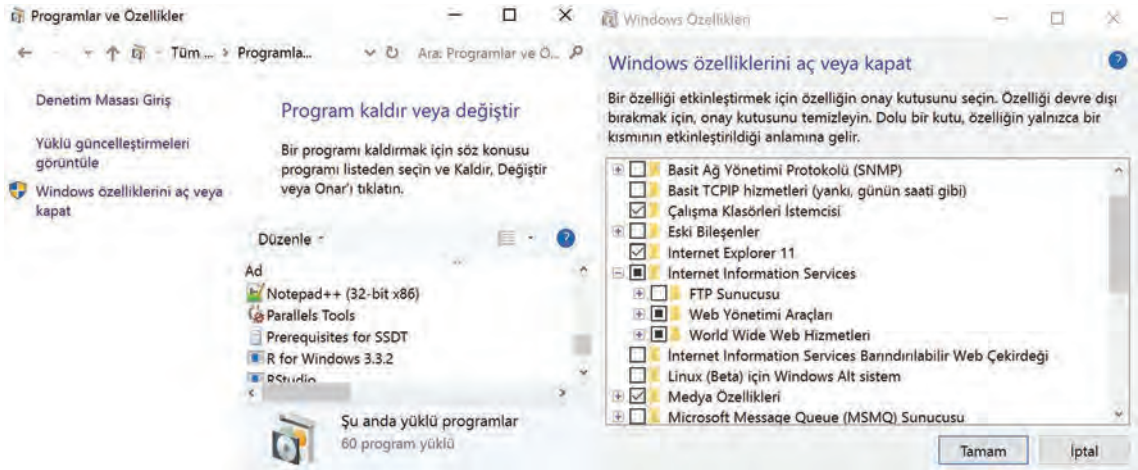
NET Framework üzerinden ne tür yazılımların geliştirilebileceğini tartışınız.

KURULUM

ASP.NET uygulamaları genellikle bir sunucu işletim sistemi olan Microsoft Windows Server kurulu bir sunucu bilgisayar üzerinden yayınlanmaktadır. Internet üzerinden web sayfalarını yayınlamak için kullanılan bu sunucu bilgisayarın İnternet üzerinde bir IP adresi ve deneme.com gibi bir alan adına sahip olması beklenir. Böylece kullanıcılar internet tarayıcının adres çubuğuna sunucunun alan adını yazarak ASP.NET uygulamalarına erişim sağlayabilecektir.

ASP.NET uygulamalarını çalıştırabilmek, test edebilmek veya yayınlayabilmek için öncellikle işletim sistemi üzerinde IIS yazılımının çalışıyor olması gerekmektedir. IIS yazılımı hem Microsoft Windows Server hem de Microsoft Windows işletim sistemlerinde bütünleşik olarak bulunmaktadır. Uygulama geliştirme aşamasında Microsoft Windows Server yerine Microsoft Windows işletişim sistemini tercih etmek daha mantıklı olacaktır. Uygulama tamamen geliştirildikten sonra Internet üzerinden yayınlamak için Microsoft Windows Server kurulu sunucu bilgisayarına aktarılabilir.

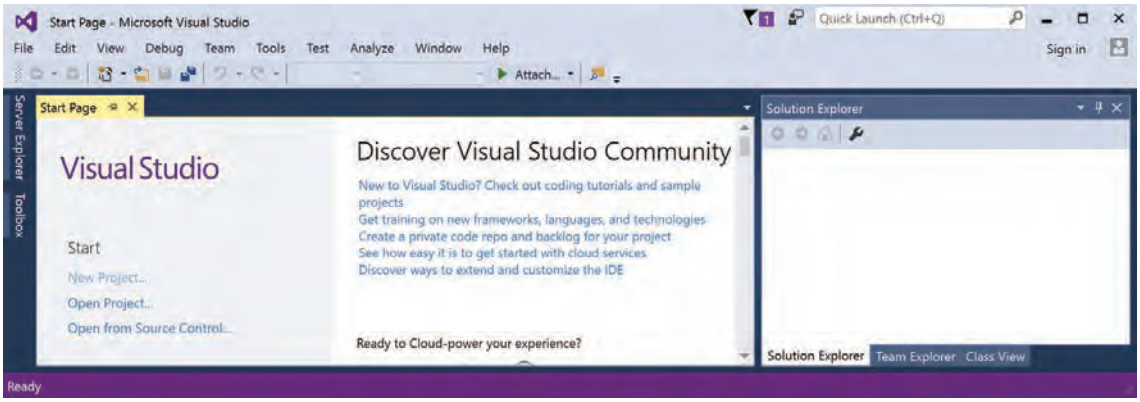
Microsoft Windows 10 işletim sisteminde IIS'i kurmak için Denetim Masasındaki "Program kaldır veya değiştir" penceresi açılmalıdır. Bu pencerenin solunda bulunan "Windows özelliklerini aç veya kapat" bağlantısı tıklanarak Windows Özellikleri penceresi açılmalıdır (Resim 6.1). Bu pencereden *Internet Information Services* onay kutusu işaretlenerek *tamam* düğmesine tıklanmalıdır. İşlem tamamlandığında IIS kurulmuş ve çalışır duruma gelmiş olacaktır. *Denetim Masası/Yönetimsel Araçlar/Internet Information Services (IIS) Yöneticisi* açılarak IIS ile ilgili ayarlamalar yapılabilir.



Resim 6.1 IIS'i Kurma

PHP ile ilgili ünitelerde bahsedildiği üzere sunucu tabanlı web uygulamalarında veri tabanı bağlantısı oldukça önem arz etmektedir. ASP.NET ile birçok veri tabanı sunucu yazılımına bağlantı kurmak mümkündür. Ancak ASP.NET ile veri tabanı sunucusu olarak çoğunlukla Microsoft SQL Server sunucu yazılımı tercih edildiği görülür. SQL Server veri tabanı sunucu yazılımı normalde ücretli bir yazılım olmasına karşın geliştiricilerin üretim dışı ortamlarda test amaçlı kullanmasına yönelik ücretsiz bir Developer sürümü mevcuttur. Bu sürüm Internet'ten indirilerek kurulabilir. Ayrıca veri tabanını yönetmek için (yeni veri tabanları ve tablolar oluşturmak, kullanıcı yetkilerini düzenlemek gibi) SQL Server Management Studio yazılımı kullanılabilir. Bu yazılımda Internet üzerinden ücretsiz olarak indirilip kurulabilmektedir.

Son olarak ASP.NET uygulamalarını geliştirmede kullanmak üzere Microsoft Visual Studio bütünleşik geliştirme ortamına ihtiyaç vardır. Aslında hiçbir geliştirme ortamı kullanmadan ASP.NET kodları basit bir editör ile yazılıp, komut ekranından .NET komutları ile derlenerek çalışır hale getirilebilir. Ancak bu işlem çok zahmetli ve zaman alıcı bir süreç olacaktır. Visual Studio; geliştiricilere kod tamamlama, web sayfalarını kod yazmadan görsel araçlar kullanarak oluşturma gibi çok sayıda araç sunmakta, yazılım geliştirme sürecini kolaylaştırmakta ve hızlandırmaktadır. Bu bakımdan ASP.NET uygulamaları geliştirmek için Visual Studio kullanmak uygun olacaktır. Visual Studio'nun Community, Professional ve Enterprise olmak üzere 3 farklı sürümü bulunmaktadır. Professional ve Enterprise sürümleri ücretli, Community sürümü ise ücretsizdir. Sürümler arasındaki farklar için visual studio'nın resmi web sayfası ziyaret edilebilir. Ayrıca bu web sayfası üzerinden Community sürümü indirilip bilgisayara kurulabilir. Bu ünitedeki örnekler Microsoft Visual Studio Community 2015 sürümü kullanılarak geliştirilmiştir. Visual Studio'nun açılış ekran görüntüsü Resim 6.2'de verilmiştir.



Resim 6.2 Visual Studio Açılış Sayfası

Öğrenme Çıktısı

2 ASP.NET kurulum gereksinimlerini sıralayabilme

Araştır 2

ASP.NET uygulamaları geliştirmek için yararlanılan Visual Studio sürümlerini araştırınız.

İlişkilendir

ASP.NET ile PHP'nin kurulum süreçlerini sistem gereksinimleri açısından ilişkilendiriniz.

Anlat/Paylaş

Daha önce karşılaştığınız ASP.NET (ASP) veya Visual Studio (VisualBasic) uygulamalarını ve deneyimlerinizi paylaşınız.

C# PROGRAMLAMA DİLİ

ASP.NET ile kullanılan en popüler dil C# programlama dilidir. C# programlama dilinin söz dizimi C, C++, Java, PHP, JavaScript gibi diğer popüler programlama dilleri ile oldukça benzerlik göstermektedir. Bu bakımdan benzer bir programlama dilini bilenlerin C# dilini öğrenmeleri oldukça kolay olacaktır. C#'ın C ve C++ programlama dillerine göre daha kolay kullanılabilen bir dil olduğu söylenebilir. Bunula birlikte C# oldukça güçlü bir programlama dilidir ve C++ ile yapılabilecek hemen hemen her şey C# ile de yapılabilir. C# programlama dili .NET platformu için sıfırdan geliştirilmiştir ve bu platformun tüm özelliklerini kullanabilir. C# ile masaüstü uygulamaları, web uygulamaları, mobil uygulamalar, web servis (WCF) uygulamaları gibi çok çeşitli türden uygulama geliştirmek mümkündür. C# ile yazılan kodların çalıştırılabilmesi için öncelikle derleyici tarafından derlenmesi gerekmektedir.

dikkat

Bu bölümde C# programlama dili ile ilgili özet bilgi verilmektedir. C# programlama dili gelişmiş özelliklere sahip kapsamlı bir dildir. Tüm bu özellikler için C# programlama dilinin çevrimiçi kılavuzundan yararlanılabilir.

internet

C# Programlama Dili Kılavuzu https://msdn.microsoft.com/en-us/library/kx37x362.aspx

Bu bölümde C# programlama dilinin yapısı tanıtılacaktır. Yazılacak örnek uygulamaları denemek için konsol uygulaması oluşturulabilir. Konsol uygulaması oluşturmak için *File/New/Project* seçilmelidir. Açılan *New Project* diyalog kutusunda Visual C# altındaki *Console Application* seçeneği seçilip proje ismi ve konumu belirlendikten sonra *Ok* düğmesine tıklanarak yeni bir konsol uygulaması oluşturulabilir. Konsol uygulaması oluşturulduğunda *Program.cs* isminde bir dosya oluşturulur ve içeriği aşağıdaki şekilde olur.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   namespace KonsolUygulamasi
7   {
8       class Program
9       {
10          static void Main(string[] args)
11          {
12              System.Console.WriteLine("Merhaba Dünya");
13              Console.ReadKey();
14          }
15      }
16  }
```

Using komutları ile uygulamada kullanılacak isim uzayları (namespace) belirlenmektedir. Bir isim uzayı *using* komutu ile eklendikten sonra o isim uzayında bulunan bir sınıf ya da fonksiyonu kullanmak için doğrudan fonksiyonun ya da sınıfın ismini yazmak yeterli olacaktır. Yukarıdaki kodda *System* isim uzayındaki *Console* sınıfı doğrudan kullanılmıştır. *Using* konutu ile eklenmemiş bir isim uzayındaki sınıf ve fonksiyonları da kullanmak mümkündür Bunun için yukarıdaki örnekte olduğu gibi isim uzayı ile birlikte sınıfın ya da fonksiyonun tam yolunu belirtmek gerekir. .NET platformunda kullanılabilecek tüm sınıflar .NET'in isim uzayları altında yer almaktadır.

Kodun 6. satırındaki *namespace* komutu ile proje için bir isim uzayı oluşturulmaktadır. Genellikle bir projede yazılan tüm sınıflar aynı isim uzayının içinde yer almaktadır. Dolayısıyla *KonsolUygulamasi* isim uzayı içindeki bir sınıftan yine *KonsolUygulamasi* isim uzayındaki başka bir sınıfa doğrudan ulaşılabilir. Ancak farklı bir projeden *KonsolUygulamasi* isim uzayına ulaşılmak isteniyorsa kullanılabilecek iki yöntem bulunmaktadır. Bunun için; *using KonsolUygulamasi* komutu en başa eklenebilir veya *KonsolUygulamasi.Program* gibi isim uzayı ile birlikte ulaşılmak istenen sınıfın tam yolu belirtilebilir.

C# aynı Java programlama dilinde olduğu gibi geliştiricileri nesne yönelimli programlama yapmaya yönlendirmektedir. Bu bakımdan basit bir konsol uygulamasının bile bir sınıf olarak yazılması gerekmektedir. Örnekte görüldüğü üzere *Program* isminde bir sınıf oluşturulmuş ve bu sınıfa ait *Main* isminde bir üye fonksiyon oluşturulmuştur. Konsol uygulamaları için *Main* fonksiyonu özel bir fonksiyondur. Program çalıştırıldığında *Main* fonksiyonu otomatik olarak çağrılır. *Main* fonksiyonunun *static* olması gerekmektedir. Program akışı; *Main* fonksiyonu bittiğinde ya da *Main* fonksiyonu içinde çalıştırılacak bir *return* komutu ile sonlanır ve programdan çıkılmış olur. Programın komut satırından çalıştırılarak bilgi gönderilmesi durumunda bu bilgileri kullanabilmek için *Main* fonksiyonuna sınırsız sayıda parametre alabilmesini sağlamak için *string* dizisi tanımlanmıştır. Kodun 12. satırında konsola "Merhaba Dünya" yazısı yazdırılmaktadır. Eğer 13. satırdaki komut yazılmaz ise program Visual Studio üzerinde F5 (Start Debugging) tuşuna basılarak çalıştırıldığında konsol penceresi açılacak, "Merhaba Dünya" yazısı yazılacak ve program sonlandığı için hemen kapanacaktır. Bu işlem çok hızlı gerçekleşeceği için konsol penceresindeki yazıları okumak mümkün olmayacaktır. Bu yüzden program akışı kullanıcının tek bir karakter girmesini bekleyen *Console.ReadKey()* fonksiyonu ile durdurulmuştur. Böylece konsol penceresi kullanıcı bir karaktere basana kadar açık kalmaya devam eder. Kullanıcı bir karaktere bastığında ise program sonlanır ve konsol penceresi kapanır.

Yazılan kodların anlamı ve yapılan, yapılacak işler konusunda kod içine açıklamalar eklemek yaygın bir davranıştır. Açıklamalar kodların anlaşılırlığını arttırır. Açıklama eklemek için diğer benzer programlama dillerinde olduğu gibi // veya /* */ karakterlerinden yararlanılır. // karakterlerinin sağında bulunan tüm yazılar açıklama olarak değerlendirilir. /* ile */ arasında kalan tüm yazılar açıklama olarak değerlendirilir.

dikkat

C# dili büyük küçük harfe duyarlıdır. Örneğin yukarıdaki örnekte yanlışlıkla Console.ReadKEY() yazılması durumunda derleme hatası ile karşılaşılır.

Değişkenler ve Türler

C# programlama dilinde değişkenlerin mutlaka kullanılmadan önce tanımlanmış olması gerekir. Tanımlanmamış bir değişken kullanılmak istendiğinde derleyici hata verecek ve kodlar derlenmeyecektir. Ayrıca değişkenler tanımlanırken türlerinin belirtilmesi gerekir. Tamsayı olarak tanımlanan bir değişken sadece tamsayı değerler ile kullanılabilir, farklı bir tür gibi kullanmak mümkün değildir. C# değişkenler konusunda katı kurallara sahip bir dildir. C# ile basit türler ve karmaşık türler olmak üzere iki farklı tür yapısından söz edilebilir. Basit türler sayısal değerler, karakter ve alfa numerik (sayı ve harflerden oluşan karakter dizisi, metin) değerleri tutan değişken türleridir. Basit türler System isim uzayının içinde yer almaktadır. Karmaşık türler ise *struct, class, interface* veya *enum* yapıları ile oluşturulmuş özel türlerdir. .NET platformu içinde çok sayıda sınıfı, bir başka değişle karmaşık türü barındırmaktadır. Basit türler ve açıklamaları Tablo 6.1'de verilmiştir.

Tablo 6.1 Basit Türler ve Açıklamaları

Tür	.NET Yolu	Açıklama
bool	System.Boolean	true ve false değerlerini barındırabilir.
byte	System.Byte	0 ile 255 arasında tamsayı değerlerini barındırabilir.
sbyte	System.SByte	-128 ile 127 arasında tamsayı değerlerini barındırabilir.
char	System.Char	Tek bir karakter barındırabilir. Bu karakterin tam sayı karşılığı 0 ile 65535 arasında olabilmektedir.
decimal	System.Decimal	-7.9 x 1028 ile 7.9 x 1028 arasında ondalık sayı değerleri barındırabilir. Negatif sayılarda virgülden sonra 28, pozitif sayılarda 29 karakter olabilir.
double	System.Double	-1.7 x 10308 ile 1.7 x 10308 arasında ondalık sayı değerleri barındırabilir. Negatif sayılarda virgülden sonra 15, pozitif sayılarda 16 karakter olabilir.
float	System.Single	-3.4 x 1038 ile 3.4 x 1038 arasında ondalık sayı değerleri barındırabilir. Virgülden sonra 7 karakter olabilir.
int	System.Int32	-2,147,483,648 ile 2,147,483,647 arasında tamsayı değerlerini barındırabilir.
uint	System.UInt32	0 ile 4,294,967,295 arasında tamsayı değerlerini barındırabilir.
long	System.Int64	-9 x 1018 ile 9 x 1018 arasında tamsayı değerlerini barındırabilir
ulong	System.UInt64	0 ile 18 x 1018 arasında tamsayı değerlerini barındırabilir
short	System.Int16	-32,768 ile 32,767 arasında tamsayı değerlerini barındırabilir
ushort	System.UInt16	0 ile 65,535 arasında tamsayı değerlerini barındırabilir
string	System.String	Alfa nümerik değerleri barındırabilir.

Değişken tanımlarken önce türü, daha sonra değişken ismi belirtilmelidir. Değişken isim belirleme kuralları PHP'deki ile aynıdır. Değişken isimlerinde harf, sayı ve _ işareti yer alabilir. Değişken isimleri sayı ile başlamamalıdır. Değişken isimlerinde boşluk ve özel karakterler(?,!,],&,...) yer alamaz. Ayrıca C# dilinde kullanılan *if, for, while, switch* gibi anahtar kelimeler de değişken ismi olarak kullanılamaz. Değişkene tanımlandığı yerde ilk değer ataması yapılabilir. Değişken tanımlarken kısa adı veya .NET tam yolu kullanılabilir. Aşağıdaki örnekte 3 farklı şekilde *int* türü değişken tanımlaması yapılmıştır. 2. satırda *int* türünün .NET tam yolu belirtilmiştir. Eğer *using System;* satırı kullanılmış ise 3. satırda olduğu gibi doğrudan *System* isim uzayı içindeki elemanlara ulaşmak mümkün olur.

```
1   int i = 123;
2   System.Int32 j = 123;
3   Int32 k = 123;
```

U (unsigned-işaretsiz) ile başlayan türler sadece pozitif tam sayıları barındırabilmektedirler. Değişkenlerin kullanılmadan önce mutlaka tanımlanması gerekmektedir. Genellikle değişken tanımlama işlemi kodlama bloğunun en üstünde yapılmaktadır. Değerlere de tür tanımlaması yapılabilmektedir. Yukarıdaki örnekte olduğu gibi bir tamsayı değerinin yanında herhangi bir harf yer almaz ise o değer varsayılan olarak *int* türünde değerlendirilir. Eğer bir tamsayının işaretsiz olması isteniyorsa değerin sonuna *U* harfi getirilmelidir. Örneğin *100U* değeri işaretsiz bir tamsayıyı temsil eder ve *ulong* ve *uint* türündeki değişkenlere atanabilir. Bir tam sayı değerinin *long* (uzun) olması istenir ise değerin sonuna *L* harfi getirilmelidir. Örneğin *100L* değeri uzun tamsayıyı temsil eder. Bu sayı *long* ve *ulong* türündeki değişkenlere atanabilir. *100UL* değeri ise hem uzun hem de işaretsiz değerleri temsil eder ve sadece *ulong* türündeki değişkenlere atanabilir. Ondalık sayısal değerler varsayılan olarak *double*'dır. 10.3F değerinde olduğu gibi ondalık değerin yanında *F* harfi getirilmesi durumunda bu değerin *float* türünde olduğu belirtilmiş olur. *10.3M* değerinde olduğu gibi değerin yanına *M* harfi getirilmesi durumunda değerin türünün *decimal* olduğu belirtilmiş olur.

Char türü değerler tek tırnak için de belirtilmelidir. *String* türü değerler ise çift tırnak içinde belirtilmelidir. Özel anlamı olan bazı karakterleri yazdırmak için Tablo 6.2'deki kaçış karakterleri veya Unicode karakterleri kullanılmalıdır. Unicode karakterleri kullanmak için \u ifadesinin yanına 4 karakterlik Unicode hexadecimal (onaltılık taban) değer yazılmalıdır.

Tablo 6.2 String Kaçış Karakterleri

Kaçış Karakteri	Üretilen Değer	Unicode Karşılığı
\'	Tek tırnak	0x0027
\"	Çift tırnak	0x0022
\\	Sola eğik çizgi	0x005C
\b	Geri al	0x0008
\n	Yeni satır	0x000A
\r	Satır başı	0x000D
\t	Sekme	0x0009

```
1   char c = 'B';
2   string str1;
3   str1 =  "Merhaba\nDünya";
4   string str2 = "\"C#\" Programlama Dili";
5   string str3 = "Türkiye\u0027nin Ovaları";
6   Console.WriteLine(str1);
7   Console.WriteLine(str2);
8   Console.WriteLine(str3);
```

Yukarıdaki örnekte 1. satırda *c* isminde *char* türünde bir değişken tanımlanmış ve tek karakterlik değer tek tırnak içine alınarak bu değişkene atanmıştır. 2. satırda *string* türü bir değişken tanımlanmış ancak ilk değer ataması yapılmamıştır. Bu değişkene değer ataması 3. satırda yapılmıştır. Değer çift tırnak içinde belirtilmiştir. Ayrıca değer için de yeni satır (\n) karakteri kullanılmıştır. *str1* değişkeni ekrana yazdırıldığında "Merhaba" değeri yazdırılır, alt satıra geçilir ve "Dünya" değeri yazdırılır. 4. satırdaki örnekte ise C# değeri çift tırnak içinde verilmek istenmiştir. *String* bir değer içinde çift tırnak kullanmak için \" kaçış karakterini kullanmak gerekmektedir. 5. satırda \u0027 unicode değeri ile tek tırnak yazdırılmıştır. Yukarıdaki örneğin ekran çıktısı aşağıdaki gibi olacaktır.

Merhaba
Dünya
"C#" Programlama Dili
Türkiye'nin Ovaları

Tür Dönüşümleri

Önceki bölümde de belirtildiği üzere değişken türlerinin kendilerine has özellikleri ve kapasiteleri mevcuttur. Türlerin hafızada işgal ettikleri alan farklılık gösterebilmektedir. Örneğin *int* türü hafızada 32 bit yer işgal ederken *long* türü 64 bit işgal etmektedir. Bu bakımdan *int* türü bir değişken, *long* türü bir değişkene atanabilir. Bu işlem güvenli bir işlemdir, çünkü *long* türü *int* türünden daha büyüktür. Ancak *long* türü bir değişkenin değerini *int* türü bir değişkene atamak veri kaybına sebep olabilir ve güvenli değildir. Güvenli olan tür dönüşümleri doğrudan kullanılabilir. Derleyici hatası oluşmaz. Güvenli tür dönüşümleri Tablo 6.3'de verilmiştir.

Tablo 6.3 Güvenli Tür Dönüşümleri

Tür	Güvenli Şekilde Atanabileceği Türler
byte	short, ushort, int, uint, long, ulong, float, double, decimal
sbyte	short, int, long, float, double, decimal
short	int, long, float, double, decimal
ushort	int, uint, long, ulong, float, double, decimal
int	long, float, double, decimal
uint	long, ulong, float, double, decimal
long	float, double, decimal
ulong	float, double, decimal
float	double
char	ushort, int, uint, long, ulong, float, double, decimal

Güvenli olmayan tür dönüşümleri doğrudan yapılamaz, yapılmak istenirse derleyici hatası oluşacaktır. Bu tür dönüşümlerde cast işlemi uygulanmalıdır. Aşağıda tür dönüşümleri ile ilgili bir örneğe yer verilmiştir.

```
1   short sayi1 = 5;
2   int sayi2 = sayi1;
3   byte sayi3 = (byte)sayi1;
4   sayi1 = 257;
5   sayi3 = checked((byte)sayi1);
```

Yukarıdaki örnek uygulamada *short* türünde bir değişken tanımlanmıştır. Bu değişkenin değeri *cast* işlemine gerek kalmadan 2. satırda doğrudan *int* türü bir değişkene atanmıştır, çünkü int daha büyük bir türdür ve bu dönüşüm güvenlidir. Ancak *short* türü *sayi1* değişkeninin değeri *byte* türü sayi3 değişkenine atanmak istenir ise cast işlemini yapmak gerekecektir. *Short* türündeki değişkenler en fazla 32767 değerine sahip olabilirler. *Byte* türü değişkenler ise en fazla 255 değerine sahip olabilirler. Eğer programcı olarak *short* türündeki değişkenin değerinin 255 veya daha küçük olduğuna emin olunursa, *cast* yöntemiyle tür dönüşümü yapılabilir. *Cast* yönteminde dönüştürülmek istenen değişken ya da değerin önüne dönüştürülecek türün adı parantez içinde yazılmalıdır. Yukarıda verilen kodun 3. satırında *sayi1* değişkeninin değeri *byte*'a dönüştürülerek *sayi3* değişkenine atanmıştır. *Sayi1* değişkeninin değeri 0 ile 255 arasında olduğu için bu dönüşüm kayıpsız gerçekleştirilir. *Sayi1* değişkeninin değeri 255'den büyük olması durumunda ise bu değer 8 bitlik bir alana sığmayacağı için kayıp ortaya çıkacaktır. Örneğin 257 değeri *byte* türüne dönüştürüldüğünde elde edilen sayı 1'dir.

Değişkenlerin değerleri konusunda emin olunamadığı durumlarda, tür dönüşümü sırasında veri kaybı oluşmuş ise programın çalışmayı sonlandırarak hata üretmesi istenebilir. Bu durumda *checked* fonksiyonundan yararlanmak gerekir. Yukarıdaki örnekte *sayi1*'in değeri 257 yapılmıştır. 5. satırda ise *sayi1 byte* türüne *cast* edilmektedir. Ancak bu durum veri kaybına sebep olur ve *checked* fonksiyon kullanıldığı için program çalışma anında hata vererek sonlandırılır.

Güvensiz tür dönüşümü yapmak için *System. Convert* sınıfından da yararlanılabilir. Bu sınıfı kullanarak tüm temel türlere dönüşüm gerçekleştirmek mümkündür. Aşağıda *Convert* sınıfı ile ilgili bir örneğe yer verilmiştir.

```
1   short sayi1 = 5;
2   byte sayi3 = Convert.ToByte(sayi1);
```

Yukarıdaki örnekte *short* türünde olan *sayi1* değişkeninin değeri *byte* türüne dönüştürülerek *byte* türünde bir değişken olan *sayi3*'e atanmıştır. Tür dönüşümlerinde kullanılan *cast* yöntemi karmaşık türlerde de kullanılabilmektedir. Burada dikkat edilmesi gereken nokta dönüştürülecek değerin hedef tür ile uyumlu olmasıdır.

Diziler

C# ile birçok değeri içinde barındırabilen diziler tanımlanabilir. C# ile tanımlanan diziler PHP'deki esnek yapıya oranla daha katı kurallara sahiptir. C# ile tanımlanan dizilerin boyutları sabittir. Anahtar (indis) olarak sadece tamsayı kullanılabilmektedir. Tüm dizi elemanları tek bir türde olmak zorundadır. Dizi anahtarları 0'dan başlamaktadır. Bir başka deyişle dizinin ilk elemanının anahtarı 0'dır. Aşağıda dizi tanımlama ile ilgili bir örneğe yer verilmiştir.

```
1   int[] dizi1 = { 1, 3, 5 };
2   string[] dizi2 = new string[5];
3   double[] dizi3;
4   dizi3 = new double[10];
5   dizi2[0] = "Armut";
```

Dizi tanımlanırken hangi türde bir dizi olacağı belirlenmelidir. *int[]* ifadesi ile bir tamsayı dizisi tanımlanmaktadır. Dizi kullanılmadan önce mutlaka new kelimesi ile veya {} kullanılarak kaç elemanlı olacağı belirlenmelidir. Küme parantezleri ile dizinin hem kaç elemanlı olduğu belirlenmekte hem de ilk değer ataması yapılmaktadır. 1. satırda *dizi1* isminde bir tamsayı dizisi tanımlanmıştır. Bu dizi üç elemanlı bir dizidir ve elemanları sırasıyla 1, 2 ve 5 değerlerine sahiptir. Yukarıdaki kodun 2. satırında *dizi2* isminde 5 elemanlı bir *string* dizisi tanımlanmıştır. Dizi elemanlarına ilk değer ataması yapılmamıştır. 3. satırda ise bir *double* dizisi tanımlanmıştır. 4. satırda *double* dizisinin 10 elemanlı bir dizi olduğu belirlenmiştir. 5. satırda *string* dizisinin ilk elemanının değeri "Armut" olarak değiştirilmektedir. Tanımlanan dizilerin boyutları daha sonradan değiştirilemez. Ancak farklı boyutlu yeni bir dizi oluşturulabilir. Dizilerin boyutlarını öğrenmek için *Lenght* özelliğinden yararlanılabilir. Örneğin *dizi1.Length* ifadesi 3 değerini verecektir. C# ile çok boyutlu diziler de oluşturmak mümkündür. Aşağıdaki örnekte çok boyutlu dizi tanımlama ile ilgili bir örneğe yer verilmiştir.

```
1   int[,] ikiBoyutluDizi = new int[5, 10];
2   ikiBoyutluDizi[0, 0] = 5;
3   ikiBoyutluDizi[4, 9] = 7;
4   double[,] İkiBoyutluDizi2 = { { 1, 2, 3, 4 }, { 2, 3, 4, 5 },
    { 3, 4, 5, 6 } };
5   int[,,] ucBoyutluDizi = new int[5, 10, 20];
6   ucBoyutluDizi[0, 0, 0] = 10;
7   ucBoyutluDizi[4, 9, 19] = 100;
```

Çok boyutlu dizi tanımlarken *int[,]* yapısı ile dizinin kaç boyutlu olacağı belirtilmelidir. Bir virgül olursa iki boyutlu, iki virgül olursa üç boyutlu bir dizi olacaktır. Bir başka deyişle boyut sayısı, virgül sayısının bir fazlasıdır. Dizi tanımlandıktan sonra dizinin büyüklüğü *new* veya {} ile belirlenmelidir. Yukarıdaki kodun 1. satırında [5,10] büyüklüğünde toplam 50 elemandan oluşan bir dizi tanımlanmıştır. 2. satırda bu dizinin ilk elemanına 3. satırda ise son elemanına ulaşılmıştır. 4. satırda [3,4] elemanlı bir *double* dizisi oluşturulmuştur. 5. satırda ise üç boyutlu olan [5,10,20] elemanlı bir int dizisi oluşturulmuştur.

ArrayList

C# programlama dilinde kullanılan diziler C# programlama dilinin doğası gereği esnek bir yapıda değildir. Daha esnek bir yapıya ihtiyaç duyulması durumunda dizilerden daha esnek bir yapı sunan *System.Collections* isim uzayı içindeki sınıflardan yararlanılmaktadır. Örneğin dizilerde dizi büyüklüğünün sabit olması gerekmektedir. Ancak birçok durumda dizi boyutunun (örneğin veritabanından alınacak bilgiler doğrultusunda) büyüyüp küçülebilen bir yapıda olması istenebilir. Bu durumda *System.Collections* isim uzayı içindeki *ArrayList* sınıfından yararlanılabilir. *ArrayList* sınıfı ile oluşturulacak dizi benzeri yapıya farklı türde değerlerler eklenebilir. *ArrayList*'e yeni değer eklemek ve eski değerleri silmek mümkündür. Ayrıca *ArrayList* sınıfı diziler ile ilgili yapılabilecek sıralama, karıştırma gibi birçok işlevsellikte sunmaktadır. Aşağıda *ArrayList* kullanımına yönelik bir örneğe yer verilmiştir.

```
1   ArrayList dizi = new ArrayList();
2   dizi.Add(5);
3   dizi.Add("Merhaba Dünya");
4   dizi.Add(new int[5]);
5   dizi[0] = 10;
6   int yeni = 10 * (int)dizi[0];
7   ((int[])dizi[2])[1] = 100;
8   Console.WriteLine(dizi[1]);
9   Console.WriteLine(((int[])dizi[2])[1]);
10  dizi.RemoveAt(1);
11  Console.WriteLine(((int[])dizi[1])[1]);
```

Yukarıdaki örneğin 1. satırında *dizi* isminde bir *ArrayList* dizisi tanımlanmıştır. *ArrayList*'e yeni eleman eklemek için *Add* fonksiyonundan yararlanılır. Herhangi bir türde yeni eleman eklemek mümkündür. 4. satırda yeni eleman olarak 5 elemanlı bir *int* dizisi eklenmiştir. 7. satırda dizinin birinci elemanına (anahtar: 0) yeni bir değer atanmıştır. Basit türler için bu şekilde *ArrayList* elemanları değiştirilebilir. Ancak *ArrayList* dizisi içindeki karmaşık türe sahip elemanlar kullanılmak istenir ise öncelikle *cast* (tür dönüştürme) işlemi yapmak gerekir, çünkü *ArrayList* dizisi içindeki tüm elemanlar aslında *Object* türüne dönüştürülmektedirler. *Object* türü ise tüm türlerin atasıdır. 6. satırda parantez içinde *int* ifadesi kullanılarak *dizi[0]*'ın *int* türünde bir değer olduğu belirtilmiştir. Eğer belirtilmemiş olsaydı, derleyici *int* ile *Object* türü arasında çarpma işlemi yapılamıyor şeklinde bir derleme hatası üretecekti. Benzer şekilde *(int[])dizi[2]* ifadesi ile de *ArrayList* dizisinin ikinci elemanının bir *int* dizisi olduğu belirtilmektedir. 7. satırda *ArrayList* içindeki *int* dizisinin ikinci elemanına (anahtar: 1) 100 değeri atanmıştır ve bu değer 9. satırda ekrana yazdırılmıştır. 10. Satırda *ArrayList* dizisinin 2. (anahtar: 1) elemanı silinmektedir. *ArayList* dizisinden eleman silmek için kullanılabilecek fonksiyonlardan biri *RemoveAt* fonksiyonudur. *RemoveAt* fonksiyonu ile anahtar numarası verilen dizi elemanı silinir. Aradaki bir eleman silindiği için *ArrayList* dizisinin 3. elemanı artık 2. elemanı olacak ve anahtarlar değişecektir.

Operatörler

C# programlama dilinde matematiksel, atama ve karşılaştırma işlemlerini yapmak için operatörler kullanılmaktadır. C# programlama dilinde kullanılabilecek operatörlerin bir kısmı Tablo 6.4'te verilmiştir. Tabloda operatör ile ilgili örnek kullanım ve açıklamaya da yer verilmiştir.

Tablo 6.4 Operatörler

Operatör	Örnek	Açıklama
+	y = x + z; y = +x;	Toplama veya birleştirme operatörüdür. Sayısal değerlerde toplama, string değerlerde is birleştirme operatörü olarak çalıştır. Birinci örnekte x ve z'nin değerleri toplanarak y'ye atanmıştır. Eğer y, x, z değerleri string türünde ise x ve y'nin değerleri birleştirilerek else edilen değer y'ye atanır. İkinci örnekte ise x'nin değeri +1 ile çarpılarak sonuç y'ye atanmıştır.
-	y = x – z; y = -x;	Çıkartma operatörüdür. Birinci örnekte x'in değerinden z'nin değeri çıkartılarak y'ye atanmıştır. İkinci örnekte ise x'nin değeri -1 ile çarpılarak sonuç y'ye atanmıştır.
*	y = x * z;	Çarpma operatörüdür. x ve z'nin değerleri çarpılarak sonuç y'ye atanmıştır.
/	y = x / z;	Bölme operatörüdür. x'in değeri, z'nin değerine bölünerek sonuç y'ye atanmıştır.
%	y = x % z;	Kalan operatörüdür. x'in değerinin z'nin değerine bölümünden kalan değer y'ye atanmaktadır.
++	y = ++x; y = x++;	Arttırma operatörüdür. Önüne veya sonuna geldiği değişkenin değerini 1 arttırır. Birinci örnekte olduğu gibi değişkenin önüne gelmesi durumunda önce arttırma işlemi yapılır daha sonra atama işlemi yapılır. İkinci örnekte ise önce atama işlemi sonra arttırma işlemi yapılır. Dolayısıyla ikinci satır çalıştıktan sonra x'in değeri 1 arttırılır. Ancak atama işlemi önce gerçekleştiği için y'nin değeri x'in önceki değerine sahip olacaktır.

--	y = --x; y = x--;	Eksiltme operatörüdür. Önüne veya sonuna geldiği değişkenin değerini 1 azaltır. Birinci örnekte olduğu gibi değişkenin önüne gelmesi durumunda önce azaltma işlemi yapılır daha sonra atama işlemi yapılır. İkinci örnekte ise önce atama işlemi sonra azaltma işlemi yapılır.
=	y = x; y = x = z;	Atama operatörüdür. Birinci örnekte x'in değeri y'ye atanmaktadır. İkinci örnekte ise z'nin değeri x ve y'ye atanmaktadır.
+=	y += x;	y+x işleminin değeri y'ye atanır.
-=	y -= x;	y-x işleminin değeri y'ye atanır.
*=	y *= x;	y*x işleminin değeri y'ye atanır.
/=	y /= x;	y/x işleminin değeri y'ye atanır.
%=	y %= x;	y%x işleminin değeri y'ye atanır.
==	y = x == z;	Eşittir operatörüdür. x ve z'in değerlerinin eşit olması durumunda true, olmaması durumunda false değeri üretilir. Sonuç boolean türü bir değişken olan y'ye atanmıştır.
!=	y = x != z;	Eşittir değildir operatörüdür. x ve z'in değerlerinin eşit olması durumunda false, olmaması durumunda true değeri üretilir. Sonuç boolean türü bir değişken olan y'ye atanmıştır.
<	y = x < z;	Küçüktür operatörüdür. x'in değerinin z'nin değerinden küçük olması durumunda true, olmaması durumunda false değeri üretilir. Sonuç boolean türü bir değişken olan y'ye atanmıştır.
>	y = x > z;	Büyüktür operatörüdür. x'in değerinin z'nin değerinden büyük olması durumunda true, olmaması durumunda false değeri üretilir. Sonuç boolean türü bir değişken olan y'ye atanmıştır.
<=	y = x <= z;	Küçük eşittir operatörüdür x'in değerinin z'nin değerinden küçük veya eşit olması durumunda true, olmaması durumunda false değeri üretilir. Sonuç boolean türü bir değişken olan y'ye atanmıştır.
>=	y = x >= z;	Büyük eşittir operatörüdür. x'in değerinin z'nin değerinden büyük veya eşit olması durumunda true, olmaması durumunda false değeri üretilir. Sonuç boolean türü bir değişken olan y'ye atanmıştır.
&&	y = x && z;	Ve operatörüdür. x, y ve z değişkenleri boolean türündedir. x ve z'nin her ikisinin birden değeri true ise true sonucu, diğer durumlarda false sonucu üretilip ve y'ye atanmıştır.
||	y = x || z;	Veya operatörüdür. x, y ve z değişkenleri boolean türündedir. x ve z'nin her hangi birinin değeri true ise true sonucu, diğer durumlarda false sonucu üretilip ve y'ye atanmıştır.

Operatörlerin aynı ifade içinde kullanılması durumunda bir öncelik sırasına göre değerlendirilmektedirler. Örneğin y = 2 + 3 * 5 ifadesinde öncelik çarpma işleminde, daha sonra toplama işleminde, en son olarak ise atama işlemindedir. Öncelik sırası ilk olarak parantez işaretindedir. Parantez içindeki işlem ilk olarak yapılır. Parantezden sonra öncelik ön ek olarak kullanılan artırma ve eksiltme (++, --) operatörlerindedir. Daha sonra sırasıyla *, /, %, +, -, <, >, <=, >=, ==, !=, &&, ||, =, *=, /=, %=, +=,-= operatörleri çalıştırılır. En son olarak ise son ek olarak kullanılan artırma ve eksiltme operatörleri çalıştırılır. ==, !=, <, >, <=, >= gibi karşılaştırma operatörleri sadece ikili karşılaştırmalar yapmak için kullanılabilir. y'nin x'den büyük ve z'den küçük olup olmadığını test etmek için x<y<z şeklinde bir karşılaştırma ifadesi yazılamaz. Bu durumu test edebilmek için && veya || ile bağlanacak iki karşılaştırma ifadesinin yazılması gerekir. Örneğin x<y && y<z şeklinde bir karşılaştırma ifadesi yazılarak y'nin x'den büyük ve z'den küçük olup olmadığı test edilebilir.

If

Birçok programlama dilinde benzer şekilde çalışan *if* ifadesi ile bir kod bloğunun belirlenen koşullara bağlı olarak çalıştırılıp çalıştırılmayacağı belirlenir. *if* ile belirtilen koşulun sağlanamaması durumunda çalıştırılmak istenen komutlar var ise *if* bloğundan sonra *else* bloğu kullanılmalıdır. Birbiriyle bağlantılı çok sayıda karşılaştırma ifadelerinin kullanılması durumunda ise *else if* yapısı kullanılmaktadır. Aşağıda *if* ifadesinin kullanımıyla ilgili bir örneğe yer verilmiştir.

```
1   int vize, final;
2   double ortalama;
3   Console.Write("Vize: ");
4   vize = Convert.ToInt32(Console.ReadLine());
5   Console.Write("Final: ");
6   final = Convert.ToInt32(Console.ReadLine());
7   ortalama = 0.4*vize+0.6*final;
8   if (ortalama >= 90) {
9       Console.Write("Pekiyi");
10  } else if (ortalama >= 80) {
11      Console.Write("İyi");
12  } else if (ortalama >= 70) {
13      Console.Write("Orta");
14  } else if (ortalama >= 60) {
15      Console.Write("Zayıf");
16  } else {
17      Console.Write("Başarısız");
18  }
```

Örnek uygulamanın amacı, kullanıcı tarafından girilen vize ve final notlarına göre ortalama hesaplayarak öğrencinin durumu hakkında bilgi vermektir. Öncelikle değişkenler tanımlanmıştır. Bu örnek için kullanıcıdan alınacak vize ve final notları için iki adet *int* türü değişken ve hesaplanan ortalamanın saklanabilmesi için bir adet *double* türünde değişken tanımlanmıştır. 3. satırdaki kod ile ekrana "Vize:" yazdırılmıştır. 4. satırda *ReadLine* fonksiyonu ile kullanıcı tarafından girilen değer okunmakta ve *ToInt32* fonksiyon ile bu değer *int* türüne dönüştürülerek vize değişkenine atanmıştır. 5. ve 6. satırda da benzer kodlar final değişkeni için tekrarlanmıştır. 7. satırda ortalama hesaplanarak ortalama değişkenine atanmıştır. 8. satırdan başlayan *if* ve *else if* bloğuyla ortalamanın 90 ve üzeri değerleri için ekrana "Pekiyi", 80 ve üzeri için "İyi", 70 ve üzeri için "Orta", 60 ve üzeri için "Zayıf", tüm bunlar değilse de "Başarısız" yazdırılmıştır. *if else if* bloğunda kullanılan karşılaştırma ifadeleri birbiri ile ilişkilidir. *if else if* yapısı kullanıldığında karşılaştırma ifadeleri sırayla kontrol edilir. Doğru sonucunu üreten karşılaştırma ifadesinin bloğu çalıştırılır ve tüm *else if* yapısından çıkılır. Örneğin ortalamanın 75 olması durumunda öncelikle 8. satırdaki karşılaştırma kontrol edilir. 8. satırdaki karşılaştırma sonucunda 75 sayısı 90'dan büyük olmadığı için yanlış sonucu üretilecek ve bir sonraki karşılaştırma ifadesi olan 10. satırdaki karşılaştırma ifadesi kontrol edilecektir. 10. satırdaki karşılaştırma sonucunda 75, 80'dan büyük olmadığı için yanlış sonucu üretilecek ve bir sonraki karşılaştırma ifadesi olan 12. satırdaki karşılaştırma ifadesi kontrol edilecektir. 12. satırdaki karşılaştırma sonucunda 75, 70'dan büyük olduğu için doğru sonucu üretilecek ve *else if* bloğu içindeki kodlar (13. satırdaki kodlar) çalıştırılacaktır. Daha sonra diğer karşılaştırma ifadeleri kontrol edilmeden *if else if* yapısından çıkılır. Örneğin ortalama değişkeninin değerinin 45 olması durumunda tüm karşılaştırma ifadeleri yanlış sonucunu üretecek ve *if else if* yapısının en sonunda bulunan else bloğu çalıştırılarak ekrana "Başarısız" yazdırılır.

For

For döngüsü, diğer programlama dillerindeki ile benzer bir çalışma mantığında sahiptir. Bloğun içine yazılan kodlar koşul ile belirtilen sayıda tekrar tekrar çalıştırılır. For yapısının söz dizimi aşağıdaki gibidir.

```
1   for(ifade1; karşılaştırma ifadesi; ifade2) {
2   komutlar;
3   }
```

For döngüsünün çalışma mantığı PHP ünitelerinde detaylı şekilde anlatılmıştır. C# programlama dilinde de aynı mantık geçerlidir. Aşağıdaki *for* döngüsü ile alt alta 5 defa "Merhaba Dünya" ifadesi yazdırılmaktadır.

```
1   for(int i = 0; i < 5; i++)
2   {
3       Console.WriteLine("Merhaba Dünya");
4   }
```

Yukarıdaki *for* döngüsünün çalışma mantığı özetle şu şekildedir. *For* döngüsüne ilk girildiğinde *i* değişkeni tanımlanır ve 0 değeri atanır. Daha sonra i < 5 kontrolü gerçekleştirilir. Bu kontrol sonucu doğru olduğu için *for* döngüsü içindeki kodlar çalıştırılır. İkinci döngü adımında i++ ifadesi çalıştırılarak i'nin değeri 1 arttırılır. Tekrar i < 5 kontrolü yapılır ve bir beşten küçük olduğu için döngü içindeki kodlar tekrar çalıştır. İkinci döngü adımındaki bu işlem i'nin değeri 5 oluncaya kadar tekrar edecektir. İ'nin değeri 5 olunca i < 5 kontrolü yanlış sonucunu üreteceği için döngüden çıkılır. Bu durumda döngü içindeki kodlar i'nin 0, 1, 2, 3 ve 4 değerleri için toplam 5 defa çalıştırılmış olacaktır. Aşağıda *for* döngüsü ile ilgili bir örneğe daha yer verilmiştir. Bu örnekte kullanıcıdan bir tam sayı değeri girmesi istenmektedir. Daha sonra 1'den kullanıcı tarafından girilen sayı değerine kadar olan tüm sayıların toplamı hesaplanmış ve ekrana yazdırılmıştır.

```
1   int i = 1, toplam = 0, sayi;
2   Console.Write("Bir sayı giriniz: ");
3   sayi = Convert.ToInt32(Console.ReadLine());
4   for( ; i <= sayi; i++)
5   {
6       toplam += i;
7   }
8   Console.WriteLine("1'den "+sayi+"\'e kadar olan olan sayıların toplamı:
    "+toplam);
```

Programın çalışma mantığı özetle şu şekildedir. Öncelikle değişkenler tanımlanmış ve *i* ile *toplam* değişkenlerine ilk değerleri atanmıştır. 2. ve 3. satırda kullanıcının bir değer girmesi sağlanmış ve bu değer tamsayıya dönüştürülmüştür. *i* değişkeni daha önce tanılandığı için *for* döngüsünün ilk ifadesi boş bırakılmıştır. i'nin tanımlama işlemi bu bölümde de yapılabilirdi. Daha sonra koşul ifadesi olan i <= sayi yazılmıştır. Bu ifadeye göre *i* değişkeninin değerinin *sayi* değişkeninin değerinden küçük veya eşit olması durumunda döngü içindeki kodlar tekrar tekrar çalıştırılacaktır. *For* döngüsünün 3. ifadesinde bulunan i++ komutu döngünün ikinci adımından itibaren çalıştırılır ve i'nin değerini her adımda 1 arttırır. Böylece i'nin değeri 1'den başlayarak kullanıcı tarafından girilen sayı değerine kadar devam edecektir. Döngü içindeki toplam += i; komutu ile ise her adımda i'nin değeri toplam değişkenin değerine eklenmektedir. Son olarak ise elde edilen toplam değişkeninin değeri ekrana yazdırılmıştır. Sonuç olarak kullanıcı 5 değerini girdiğinde 1+2+3+4+5 hesaplaması yapılacak ve ekrana "1'den 5'e kadar olan sayıların toplamı: 15" ifadesi yazdırılacaktır.

Karşılaştırma ifadesinin her zaman doğru sonucunu üretmesi durumunda sonsuz döngüye girilmiş olur. Bu durum programın tüm işlemci gücünü kullanmasına ve yanıt vermemesine neden olabilir, çünkü program akışı döngü içinde dönmeye ve döngü bloğu komutlarını çalıştırmaya devam etmektedir. Bu gibi sonsuz döngüye neden olabilecek karşılaştırma ifadelerinden kaçınmak gerekir.

$\sum_{x=1}^{10} x^2$ ifadesini hesaplayan for döngüsünü, bir başka ifade ile 1'den 10'a kadar olan sayıların kareleri toplamını hesaplayan for döngünü yazalım.

int toplam = 0; for(int i = 1; i <= 10; i++) { toplam += i * i; }

While

Diğer programlama dillerinde olduğu gibi C# ile döngü oluşturmak için kullanılabilecek bir başka yapı *while* döngüsüdür. *While* döngüsünün sözdizimi aşağıdaki gibidir.

```
1   while (karşılaştırma ifadesi){
2      komutlar;
3   }
```

For döngüsü ile benzer bir çalışma mantığı vardır. Karşılaştırma ifadesinin sonucu doğru ise döngü içindeki komutlar çalıştırılır, yanlış ise döngüden çıkılır. Aşağıda *while* döngüsü kullanılarak bir faktöriyel hesabı yapılmaktadır.

```
1   int i = 2, carpim = 1, sayi;
2   Console.Write("Bir sayı giriniz: ");
3   sayi = Convert.ToInt32(Console.
    ReadLine());
4   while(i <= sayi)
5   {
6       carpim *= i;
7       i++;
8   }
9   Console.WriteLine(sayi+"!= "+carpim);
```

Programın çalışma mantığı özetle şu şekildedir. Öncelikle gerekli değişkenler tanımlanmıştır. Daha sonra kullanıcının bir tamsayı değeri girmesi sağlanmıştır. Hesaplamanın yapılabilmesi için *while* döngüsü kurulmuştur. *While* döngüsü i <= sayi koşulu sağlandığı sürece dönecektir. Her döngü adımında i'nin değeri 1 arttırıldığı için i'nin değeri kullanıcı tarafından girilen sayı değişkeninin değeri oluncaya kadar döngü çalışmaya devam edecektir. Her döngü adımında *carpim* *= i komutu ile faktöriyel hesabı yapılmaktadır. Kullanıcının 4 değerini girdiği varsayılır ise, üç döngü adımı olacak ve *carpim* *= 2, carpim *= 3 ve carpim *= 4 komutları çalıştırılacaktır. İlk komut sonucunda *carpim* değişkeninin değeri 2 (1*2) olur. İkinci komut sonunda *carpim* değişkeninin değeri 6 (2*3) olur. Üçüncü komut sonucunda ise carpim değişkeninin değeri 24 (6*4) olur. Son olarak "4! = 24" ifadesi ekrana yazdırılır.

Do While

C# programlama dilinde kullanılabilecek bir başka döngü yöntemi ise do while döngüsüdür. Do while döngüsünün sözdizimi aşağıdaki gibidir.

```
1   do {
2       komutlar;
3   } while (karşılaştırma ifadesi);
```

Do *while* döngüsü *while* döngüsüne benzer bir mantık ile çalışmaktadır. *While* döngüsünde olduğu gibi karşılaştırma ifadesinin doğru olması durumunda döngü bloğu içindeki komutlar çalıştırılır. *While* döngüsünden farkı ise döngüye ilk girildiğinde önce döngü bloğu içindeki kodlar çalıştırılır, daha sonra karşılaştırma ifadesi ile kontrol gerçekleştirilir. Bu durumda döngü içindeki kodlar en az bir defa mutlaka çalıştırılmış olur.

Foreach

Foreach döngüsü ile dizi elemanları veya *System. Collections*'da tanımlanmış dizi amaçlı kullanılan *ArrayList* gibi sınıfların elemanları içinde gezinmek için kullanılır. *Foreach* döngüsünün sözdizimleri aşağıdaki gibidir.

```
1   foreach (tür değişkenAdı in diziAdı){
2      komutlar;
3   }
```

Aşağıda verilen kodda *foreach* kullanımına yönelik bir örneğe yer verilmiştir.

```
1   int[] sayiDizisi = { 1, 3, 5 };
2   ArrayList esnekDizi = new ArrayList();
3   esnekDizi.Add("Armut");
4   esnekDizi.Add("Muz");
5   foreach(int eleman in sayiDizisi)
6   {
7       Console.WriteLine(eleman);
8   }
9   foreach(string eleman in esnekDizi)
10  {
11      Console.WriteLine(eleman);
12  }
```

Yukarıdaki örnekte iki tür dizi tanımlanmıştır: *int[]* ve *ArrayList.* 5. satırdaki *foreach* döngüsü ile *int[]* dizisi elemanları arasında bir döngü kurulmuş ve elemanlar ekrana yazdırılmıştır. 9. satırdaki *foreach* döngüsü ile *ArrayList* dizisi elemanları arasında bir döngü kurulmuştur. Örnekte de görüldüğü üzere klasik diziler ile *ArrayList* dizilerinin *foreach* ile kullanımı aynıdır.

For, while, do while ve foreach döngü oluşturmak için kullanılan benzer yapılardır. Aynı döngü farklı döngü yapıları kullanılarak yazılabilir.

Break ve Continue

Program akışında belli koşullar oluştuğunda döngülerin sonlandırılması gerekebilir. Bu gibi durumlarda *break* komutu kullanılır. Break komutu aynı zamanda *switch* yapısından da çıkılmasını sağlar. Aşağıda *break* komutunun kullanımı ile ilgili bir örnek bulunmaktadır.

```
1   string[] dizi = {"Mandalina", "Portakal", "Muz", "Elma", "Muz"};
2   for (int i = 0; i < dizi.Length; i++) {
3    if (dizi[i] == "Muz") {
4      Console.Write("Muz değeri dizinin " + (i + 1) + ". elemanında bulunmaktadır");
5      break;
6    }
7   }
```

Yukarıdaki örnekte öncelikle 5 elemanlı bir string dizisi tanımlanmıştır. For döngüsü ile dizinin elemanları arasında bir döngü kurulmuştur. 3. satırdaki if komutu ile dizinin i'ninci elemanın değerinin Muz olması durumunda ekrana bir uyarı yazdırılmakta ve break komutu ile for döngüsünden çıkılmaktadır. Muz dizinin 3 elemanı (anahtar: 2) olduğu için i'nin 2 olması durumunda if komutundaki karşılaştırma ifadesi doğru sonucunu üretecek ve "Muz değeri dizinin 3. elemanında bulunmaktadır" metnini ekrana yazdıracak ve break komutu sayesinde for döngüsünden çıkılacaktır. For döngüsünden çıkıldığı için dizinin geri kalan diğer elemanları için kontrol gerçekleştirilmemiş olur. Dolayısıyla dizinin 5. elemanı için benzer bir mesaj ekrana yazılmaz.

Continue komutu döngülerde kullanılır ve program akışının döngü bloğunun en başına gelmesini sağlar. Aşağıda continue kullanımına bir örnek verilmiştir.

```
1   int[] notlar = { 40, 0, 50, 80, 100, 0, 70, 90, 0, 100, 80 };
2   int toplam = 0, say = 0;
3   foreach (int notu in notlar){
4       if (notu == 0 || notu == 100){
5           continue;
6       }
7       toplam += notu;
8       say++;
9   }
10  Console.Write("Sınıf Ortalaması: "+(toplam/say));
```

Örnekte bir sınıfa ait notların bulunduğu notlar dizisi tanımlanmış ve değerleri atanmıştır. Uç değerler olan 0 ve 100 değerleri dahil edilmeden sınıf ortalaması hesaplanacaktır. Bunun için de notların toplamını saklamak için toplam değişkeni ve kaç adet not olduğunu tutmak için say değişkeni tanımlanmıştır. 3. satırda dizinin tüm elemanları içinde gezinmek için bir foreach döngüsü kurulmuştur. 4. satırdaki if komutu kullanılarak notu değişkeni (dizinin o döngü adımındaki elemanı) 100 veya 0 ise continue komutu ile döngü bloğunun başına dönülür ve bir sonraki döngü adımı başlamış olur. Böylece notu değişkeninin değerinin 0 veya 100 olması durumunda 7. ve 8. satırlardaki kodlar çalıştırılmamış olacaktır. Dolayısıyla 0

ve 100 değerleri toplam değişkenine eklenmez ve bu değerler için say değişkeninin değeri artmaz. Sonuçta toplam değişkeninin değeri 0 ve 100 haricindeki sayıların toplamı ve say değişkeninin değeri ise 0 ve 100 haricindeki toplam not sayısı şeklinde olacaktır. Son satırda ise toplam/say değeri ekrana sınıf ortalaması olarak yazdırılmıştır.

Switch

Switch komutu if - else if yapısına benzer bir karşılaştırma yapısıdır. Switch komutu parametre olarak verilen değer doğrultusunda case ifadeleri ile belirlenen kod bloklarını çalıştırmaya yarar. Aşağıda switch komutunun kullanımına yönelik bir örneğe yer verilmiştir.

```
1   int haftaninGunu = 5;
2   switch (haftaninGunu)
3   {
4       case 1:
5           Console.WriteLine("Pazartesi");
6           break;
7       case 2:
8           Console.WriteLine("Salı");
9           break;
10      case 3:
11          Console.WriteLine("Çarşamba");
12          break;
13      case 4:
14          Console.WriteLine("Perşembe");
15          break;
16      case 5:
17          Console.WriteLine("Cuma");
18          break;
19      case 6:
20         Console.WriteLine("Cumartesi");
21         break;
22      case 7:
23          Console.WriteLine("Pazar");
24          break;
25      default:
26          Console.WriteLine("Yanlış değer");
27          break;
28  }
```

Yukarıdaki örnekte haftanın gününü temsilen *int* türü bir değişken kullanılmaktadır. Bu değişkenin değerine göre 1 Pazartesi gününü, 7 ise Pazar gününü temsil etmektedir. Bu durumda değişkenin değerine göre ekrana hangi gün olduğu *switch* yapısı kullanılarak yazdırılmıştır. Aynı işlem elbette *if else if* yapısı ile de yapılabilir. Öncelikle *switch* komutuna parametre olarak *haftaninGunu* değişkeni gönderilmiştir. Bu değişkenin değerine göre *case* ifadeleri oluşturulmuştur. Örneğin *haftaninGunu* değişkeninin değeri 5 olduğunda *case* 5 bölümündeki kodlar çalıştırılır. Her *case* bölümü sonunda mutlaka *break* komutu yer almalıdır. *Break* komutu ile *switch* yapısından çıkılmış olur. *Default* bölümü ise parametre olarak verilen değer case ifadelerinde yer almıyorsa çalıştırılacak olan bölümdür. Örneğin *haftaninGunu* değişkeninin değeri 8 olursa, hiçbir *case* ifadesinde 8 tanımlanmadığı için *default* bölümü çalıştırılır ve ekrana "Yanlış değer" yazdırılır.

Sınıflar

C# programlama dili, programcıları nesne yönelimli programlama yaklaşımı ile çalışmaları konusunda yönlendirmektedir. Visual Studio ile yeni bir form, web form, web servis gibi bir dosya oluşturulduğunda şablondan bir sınıf yapısı ile oluşturulduğu görülecektir. Sınıflar nesne yönelimli programlama yaklaşımının en önemli araçlarıdır.

Sınıf tanımlamak için *class* kelimesi kullanılır. Genellikle sınıf tanımları aynı adı taşıyan ayrı bir dosyada yapılmaktadır. Aynı dosya içinde birçok sınıf tanımlaması yapmak da mümkündür. Visual Studio ile yeni bir sınıf tanımlamak için *Project/Add Class* menüsünden yararlanılabilir. Bu menü seçeneği seçildiğinde *Add New Item* diyalog kutusu açılacak ve *Class* seçeneği seçili olacaktır. *Diyalog* penceresinin altında bulunan *Name* bölümüne dosya adı yazılarak yeni bir sınıf oluşturulabilir. Örneğin içinde çok sayıda topun hareket ettiği bir oyun geliştirirken, topların ekrandaki konumunu ve hareketlerini kontrol eden bir sınıf tanımlaması yapılabilir. Aşağıda yeni oluşturulmuş *Top* isimli sınıfın tanımı görünmektedir.

> C# ile sınıf tanımlarken bir ünitede anlatılamayacak kadar çok sayıda özellik mevcuttur. Bu bölümde sadece basit bir sınıf tanımlamasının nasıl yapılabileceğinden bahsedilmiştir.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace KonsolUygulamasi
8   {
9       class Top
10      {
11      }
12  }
```

Yeni bir sınıf oluşturulduğunda Visual Studio gerekli olabilecek isim uzaylarını otomatik olarak eklemektedir. Ayrıca yeni sınıf tanımını var olan projenin isim uzayı içinde tanımlamaktadır. Böylece projenin diğer dosyalarından bu sınıfa *using* kullanmadan doğrudan ulaşmak mümkün olacaktır. Eğer projede çok sayıda sınıf olacak ise ve bu sınıflar hiyerarşik bir şekilde gruplandırılmak isteniyor ise, aynı proje içinde farklı isim uzayları tanımlanabilir. Sınıf yapıldıktan sonra sınıfın üyeleri tanımlanmalıdır. Sınıf üyesi olarak ise değişkenler, fonksiyonlar ve özellikler tanımlanabilmektedir. Aşağıda *Top* sınıfı için yazılabilecek sınıf üyeleri ile ilgili örneğe yer verilmiştir.

```
1   class Top
2   {
3       private double yariCap;
4       public double xKonum;
5       public double yKonum;
6       public Top(double yCap)
7       {
8           yariCap = yCap;
9           xKonum = 0;
10          yKonum = 0;
11      }
12      public Top()
```

```
13      {
14          yariCap = 10;
15          xKonum = 0;
16          yKonum = 0;
17      }
18      public double YariCap
19      {
20          get { return yariCap; }
21          set { yariCap = value; }
22      }
23      public double Cap
24      {
25          get { return yariCap * 2; }
26          set { yariCap = value / 2; }
27      }
28      public void TopaVur(double xGuc, double yGuc)
29      {
30  Console.WriteLine("Topa x ve y düzlemlerinde belirlenen güç ile vurulduğunda
yapılacak işlemler");
31      }
32     public double Cevresi()
33      {
34          return 2 * Math.PI * yariCap;
35      }
36  }
```

Yukarıdaki örnekte de görüldüğü gibi sınıf üyeleri tanımlanırken erişilebilirlik durumunun belirlenmesi gerekir. Erişilebilirlik durumu için *public, private, protected* gibi erişilebilirlik seçenekleri mevcuttur. *Private* olarak tanımlanmış olan bir üyeye sadece sınıf içindeki diğer üyelerin içine yazılacak kodlar ile erişilebilir. Sınıf dışından *private* üyelere erişmek mümkün değildir. *Public* olan üyeler sınıf içi ve dışından erişilebilirler. *Protected* özelliği ile tanımlanan bir üyeye sınıf içinden ve o sınıfı miras alan alt sınıfların içinden erişilebilir, sınıf dışından erişilemez. Genellikle sınıf tanımlarında en üst bölümde üye değişkenler tanımlanmaktadır. Genellikle bu üye değişkenler *private* olarak tanımlanır. Üye değişkenler sınıfın veri depoları olduğu için bunlara dışarıdan doğrudan ve kontrolsüz şekilde erişilmesi istenmez. Ancak örnek olması açısında yukarıdaki kodda *yariCap* değişkeni *private, xKonum* ve *yKonum* değişkenleri ise *public* olarak tanımlanmıştır. Bu durumda sınıf dışından *xKonum* ve *yKonum* değişkenlerine erişilebilirken, *yariCap* değişkenine erişilemez.

Sınıfın *private* değişkenlerine kontrollü bir şekilde değer atamak ve bu değerleri okumak için sınıf özelliklerinden (properties) yararlanılmaktadır. Örnekte *public double YariCap* şeklinde bir özellik tanımı yapılmıştır. Bu özelliğin amacı sınıfın *private* olan *yariCap* değişkenine değer atması ve okuması yapabilmektir. Görüldüğü üzere özelliklerin içinde *get* ve *set* olmak üzere iki blok yazılabilmektedir. *Get* bloğu ile geriye istenilen değer döndürülmekte, *set* bloğu ile ise özelliğe atanacak değer (value) *private* değişkenine aktarılmaktadır. Bunun yanında *Cap* isminde bir özellik daha tanımlanmıştır. Bu özelliğin amacı da yine *private* olan *yariCap* değişkenine değer atmak ve okumaktadır. Ancak bir çemberin çapı, yarı çapının iki katı olduğu için *get* ve *set* blokları buna uygun olarak düzenlenmiştir. Ayrıca C# programlama dilinde fonksiyon tanımlamalarında mutlaka bir geri dönüş türü belirtilmelidir. Fonksiyon içinde de geri dönüş türüne uygun bir *return* ifadesi mutlaka yer almalıdır. Eğer geri dönüş türü olarak *void* yazılırsa, geriye bir değer dönüşü yapılmayacağı anlamına gelir ve *return* ifadesi kullanılmayabilir. Örnekteki *YariCap* ve *Cap* özelliklerinin geri dönüş türü *yariCap* değişkeninin türü ile aynı olacak şekilde belirlenmiştir.

Örnekte 6. ve 12. satırda iki adet kurucu fonksiyon tanımlaması yapılmıştır. Kurucu fonksiyonlar bu sınıfa ait bir nesne oluşturulduğunda çalışan fonksiyonlardır. Kurucu fonksiyonlarda geri dönüş türü belirtilmez. C# programlama dilinde aynı isimli birden fazla fonksiyon tanımlaması yapılabilmektedir. Bu işleme fonksiyonların aşırı yüklenmesi adı verilmektedir. Fonksiyonları aşırı yükleyebilmek için tanımlanan bu fonksiyonların parametrelerinin farklılık göstermesi gerekmektedir. Tanımlanan kurucu fonksiyonların biri bir parametre alırken, diğeri hiç parametre almamaktadır. Fonksiyon çağrılırken kullanılan parametre sayısına göre derleyici hangi fonksiyonu çağıracağını otomatik olarak belirlemektedir. 8. satırda yariCap = yCap; ifadesi ile kurucu fonksiyona parametre olarak verilen değer üye değişken olan *yariCap* değişkenine atanmaktadır. *xKonum* ve *yKonum* değişkenlerinin değeri ise 0 olarak belirlenmiştir.

Örnekte 28. ve 32. satırlarda iki fonksiyon tanımlaması yapılmıştır. *TopaVur* fonksiyonu iki adet *double* parametre almakta ve geriye bir değer döndürmemektedir (void). Bu fonksiyon oyunda topa vurulması olayı gerçekleştiğinde çalışacak olan fonksiyon olarak düşünülebilir. İçeriğine ise parametre olarak verilen değerler doğrultusunda topun ekranda hareket etmesini sağlayacak işlemleri başlatacak kodların yazılması uygun olacaktır. Örnekte ise sadece ekrana bir mesaj yazdırılmıştır. İkinci fonksiyon ise parametre almamakta ve geriye bir *double* değer döndürmektedir. Bu fonksiyonun amacı topun çevre uzunluğunu hesaplamaktır. *2 * Math. PI * yariCap* işlemi ile üye *yariCap* değişkenini 2π değeriyle çarparak sonucu *return* ile geri döndürmektedir. Fonksiyon, değişken ve özellikler yazılarak sınıf tanımlaması yapılır. Tanımlanan bu sınıfların kullanılması gerekir. Aşağıda Tanımlanan Top sınıfına ait nesnelerin nasıl oluşturulabileceği ile ilgili örneğe yer verilmiştir.

```
1   Top top1 = new Top();
2   Top top2;
3   top2 = new Top(15);
4   top1.YariCap = 40;
5   top1.xKonum = 100;
6   top1.yKonum = 50;
7   top2.Cap = 60;
8   double cevre = top2.Cevresi();
9   top1.TopaVur(10,5);
10  Console.WriteLine("Top2\'nin Yarı Çapı :" + top2.YariCap);
11  Console.WriteLine("Top2\'nin çevresi :" + cevre);
```

Tanımlanan sınıf türünde bir nesne oluşturmak için *new* kelimesi kullanılmalıdır. Yukarıdaki örnekte *top1* ve *top2* isminde iki adet nesne oluşturulmuştur. 1. satırda *new Top()* ifadesi kullanıldığında sınıfın parametre almayan kurucu fonksiyonu çağrılır. 3. satırda *new Top(15)* ifadesi kullanıldığında ise parametre alan kurucu fonksiyon çağrılmaktadır. 4. satırda *top1.YariCap* ifadesi sınıfın *YariCap* özelliğini ifade etmektedir. *YariCap* özelliğine değer ataması yapıldığı için özellik tanımındaki *set* bloğu çalıştırılır ve *top1* nesnesinin *private* değişkeni olan *yariCap*'a değer ataması yapılır. İsimler benzer gibi görünebilir ancak *YariCap* özellik iken *yariCap* üye değişkendir. Ayrıca *top1.yariCap* şeklinde bir kullanım olamaz, çünkü *yariCap* üye değişkeni *private* olarak tanımlanmıştır. *Private* üyelere sınıf dışından ulaşılamaz. *Top1* nesnesinin *xKonum* ve *yKonum* üye değişkenleri *public* olarak tanımlandıkları için doğrudan ulaşılarak değer ataması yapılabilmiştir. 7. satırda *top2* değişkeninin *Cap* özelliğine 60 değeri atanmıştır. *Cap* özelliğinin *set* bloğunda bu değer ikiye bölünerek *top2* nesnesinin *yariCap* değişkenine 30 olarak aktarılır. 8. satırda *top2* nesnesinin *Cevresi* fonksiyonu çağrılmıştır. Bu fonksiyon geriye bir değer döndürdüğü için bu değer *double* türünde *cevre* ismindeki değişkene atanmıştır. 10. satırda ise *top2*'nin *YariCap* özelliği ekrana yazdırılmıştır. Ekrana "Top2'nin Yarı Çapı: 30" yazacaktır. Burada *YariCap* özelliğinin *get* bloğu çalıştırılır ve *yariCap* üye değişkenin değeri *return* ile geri döndürülür. 11. satıda ise cevre değişkeninin değeri ekrana yazdırılmıştır. Bu komut sonucunda ekrana "Top2'nin çevresi :188,495559215388" metni yazdırılmaktadır.

Bir sınıfa ait üye değişkenlerin değerlerinin, bu sınıfa ait nesne oluşturulurken belirlenmesini zorunlu kılmak istenirse nasıl bir yol izlenebileceğini açıklayalım. Bunun için öncelikle üye değişken sayısı kadar parametre alan bir kurucu fonksiyon tanımlanmalıdır. Eğer parametre almayan bir kurucu fonksiyon daha tanımlanmaz ise yeni bir nesne oluşturulurken parametreli kurucu fonksiyonun çağrılması mecburi olmuş olur. O nesne parametre olmadan oluşturulamaz. Dolayısıyla nesne oluşturulurken üye değişkenlerin ilk değerleri de atanmış olur.

Öğrenme Çıktısı

3 Değişken türleri ve tür dönüşümlerini kullanabilme
4 Dizileri ve ArrayList'leri kullanabilme
5 Kontrol yapılarını kullanabilme
6 Sınıf tanımlaması yapabilme

Araştır 3

Dersi geçmek için final (yarıyıl sonu) sınavından en az 50 almış olmak ve ortalamanın en az 45 olması gerekmektedir. Ortalama ve final değişkenleri verildiğinde bu öğrencinin dersi geçip geçmediğini tespit eden karşılaştırma ifadesini yazınız?

İlişkilendir

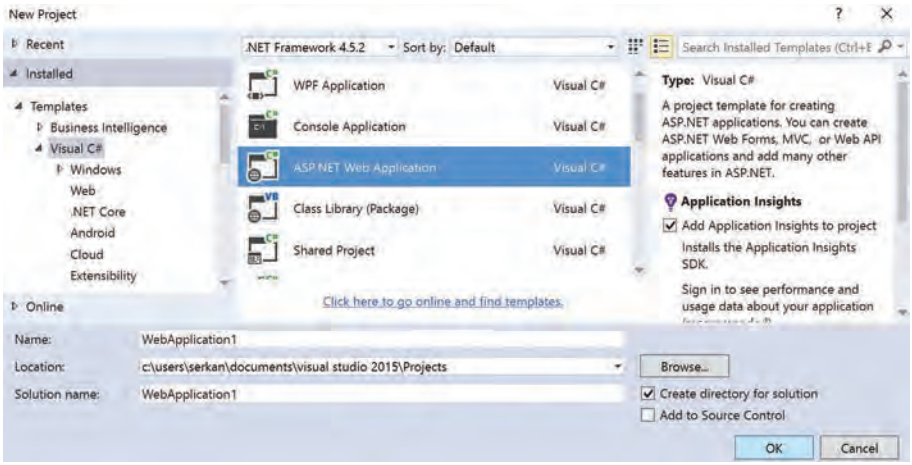
C# ile Java programlama dillerini kabiliyetleri açısından ilişkilendiriniz.

Anlat/Paylaş

C# programlama dilinin neden ileri düzeyde gelişmiş bir programlama dili olarak kabul edildiğini açıklayınız.

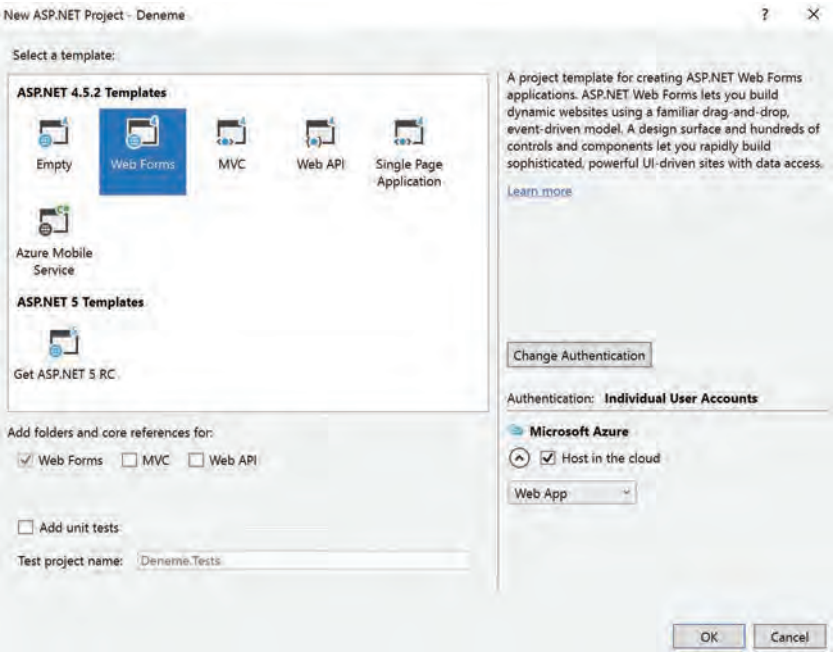
WEB FORMLARI

.NET platformu tamamen nesne yönelimli programlama yaklaşımı ile geliştirilmiş bir platformdur. ASP.NET, platformun sunduğu nesne yönelimli programlamanın tüm avantajlarından yararlanmaktadır. Hatta sistem geliştiriciyi nesne yönelimli programlama yapmaya yönlendirmektedir. ASP.NET ile bir web sayfası geliştirildiğinde aslında üye fonksiyonları, üye değişkenleri ve olayları olan bir sınıf oluşturulmaktadır. Bu web sayfası çağrıldığında ise bu sınıfa ait bir nesne yaratılır ve bu nesne tarafından hizmet sunulur. Nesne yönelimli programlama oldukça kapsamlı bir konudur ve bu konuda yardımcı kaynaklardan yararlanmak uygun olacaktır. Visual Studio ile ASP.NET uygulaması geliştirmek için yeni bir ASP.NET Web Application projesi oluşturmak gerekir. *File/New/Project* seçeneği seçildiğinde Resim 6.3'teki *New Project* diyalog kutusu açılır.



Resim 6.3 New Project Diyalog Kutusu

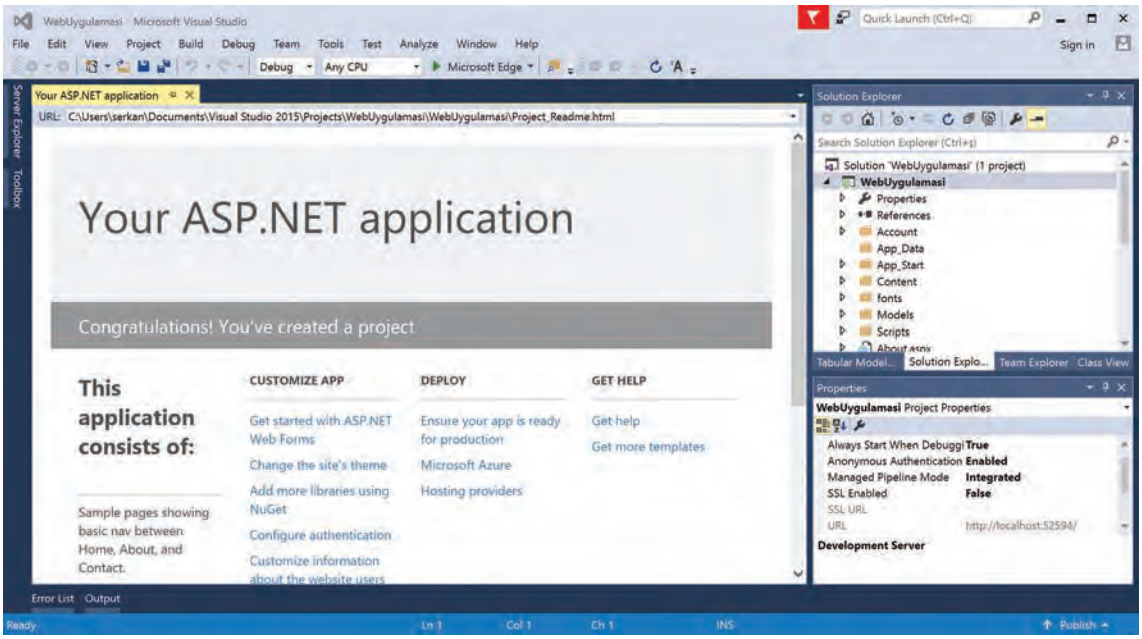
ASP.NET uygulamalarını farklı diller ile geliştirmek mümkündür. Örnekte Visual C# kullanılarak ASP.NET web uygulaması oluşturulmaktadır. Visual Studio'da *Solution* ve *Project* olmak üzere iki kavram vardır. *Solution* hiyerarşinin en üstündedir ve bir *solution* içinde çok sayıda proje yer alabilir. Bir *proje* içinde de çok sayıda web sayfası yer alabilir. Yeni *proje* oluşturulduğunda aynı zamanda yeni bir *solution* da oluşturulmaktadır. Eğer *solution* içinde çok sayıda *proje* yer almayacak ise *solution* ismi ile *proje* ismi genellikle aynı olmaktadır. *New Project* diyalog kutusunda proje ismi ve konumu belirtilerek *OK* düğmesine basıldıktan sonra Resim 6.4'teki diyalog kutusu açılmaktadır.



Resim 6.4 New ASP.NET Project Diyalog Kutusu

ASP.NET ile uygulama geliştirmenin temelde iki yöntemi olduğu söylenebilir. Bunlar; Web Forms ve MVC'dir. MVC (Model-View-Controller) yöntemiyle ilgili kitabın diğer ünitelerinde daha detaylı bilgi verilmiştir. MVC yöntemi *model* (veritabanı), *görünüm* (HTML, CSS, JavaScript kodları) ve *kontrolcü* (süreç yönetimi) işlerinin birbirinden ayrılarak yönetilmesi mantığına dayanmaktadır. ASP.NET ile MVC yöntemiyle uygulama geliştirmek mümkündür. MVC yapısı, Web Formalarına göre daha karmaşıktır ve öncelikle çeşitli tanımlamaların yapılmasını gerektirir. Daha basit projeler için Web Formları ile hızlı geliştirme yapmak üretkenliği arttıracaktır. Ancak daha büyük projelerde ise daha esnek bir yapı sunan, test edilebilir ve daha kolay yönetilebilen MVC yapısı tercih edilebilir. Geliştirilecek projenin yapısına göre MVC ya da Web Formları tercih edilmelidir. ASP.NET geliştiricileri arasında hangisinin daha iyi bir geliştirme yöntemi olduğu tartışmalı bir konu olmayı sürdürmektedir. Bu bakımdan bir yöntemin diğer yöntemden daha iyi olduğunu söylemek doğru olmayacaktır. Bu ünitede ASP.NET Web Formları konusu anlatılmaktadır.

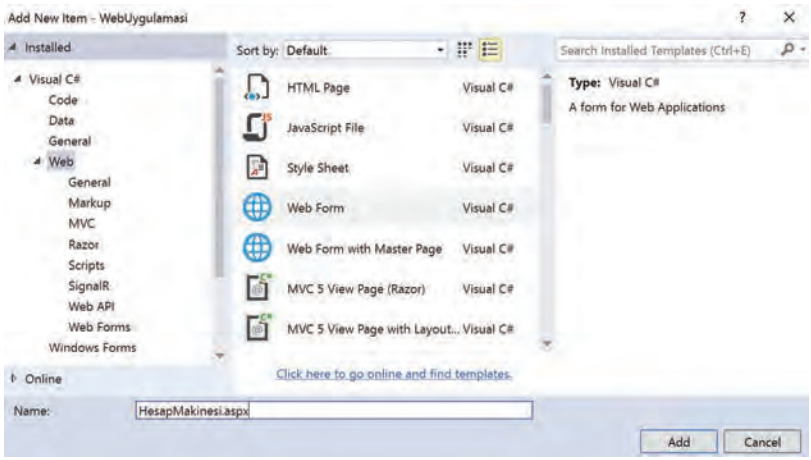
Web Formları ile birbirinden bağımsız veya belli ölçülerde bağımlı web sayfaları geliştirilmektedir. Web formlarında MVC'de olduğu gibi bir ayrım söz konusu değildir. *New ASP.NET Project* diyalog kutusunda *Microsoft Azure* onay kutusu bulunmaktadır. Microsoft Azure, Microsof'un bulut teknolojisinin adıdır. Eğer üyelik var ise projeler Microsoft Azure üzerinde saklanabilir, dağıtılabilir ve yönetilebilir. Microsoft Azure seçeneği iptal edilerek *OK* düğmesine tıklandığında Resim 6.5'teki gibi yeni bir proje oluşturulmuş olur.



Resim 6.5 Yeni Proje Ekranı

Yeni *Web Forms* projesi oluşturulduğunda varsayılan olarak birçok dosya da projeye eklenmiş olacaktır. Şekildeki görüntü incelendiğinde sağ bölümde *Solution Explorer* ve *Properties* panelleri görülecektir. *Solution Explorer* panelinde projede bulunan dosyalar listelenmektedir. Bu dosyalar üzerine çift tıklanarak orta bölümdeki editör panelinde açılması sağlanabilir. *Properties* paneli ise editör panelinde seçilen nesne ile ilgili özelliklerin seçilebildiği alandır. Seçilen nesnenin türüne göre özellikler değişiklik gösterecektir. Pencerenin sol bölümünde bulunan *Tookbox* paneli kullanılarak orta panelde açık olan web sayfasına *TextBox, Button* gibi çeşitli elemanlar eklemek mümkündür. Visual Studio ile çalışmanın en önemli avantajlarından biri de web sayfalarını (HTML, CSS) kod yazmadan görsel olarak oluşturulabilmesidir.

Web Forms olarak oluşturulan web sayfaları *aspx* uzantılı olmaktadır. Visual studio ile geliştirme yapıldığında ayrıca *aspx* dosyası ile aynı isimde bir adet *cs* uzantılı ve bir adet de *designer.cs* adlı dosya oluşturulmaktadır. Genellikle HTML, CSS ve JavaScript kodları *aspx* dosyasına, C# kodları ile *cs* dosyasına yazılmaktadır. *Designer.cs* dosyası sistem tarafından otomatik oluşturulmakta ve *toolbox* kullanılarak form üzerine eleman eklendiğinde otomatik olarak güncellenmektedir. HTML, CSS, JavaScript ve C# kodları tek bir dosyaya yazılarak web sayfası oluşturulabilir. Ancak Visual Studio'nun yönlendirdiği şekilde kullanmak ve C# kodlarını farklı dosyalarda barındırmak kodların yönetimi açısından daha iyi bir yöntem olacaktır. Kullanıcılar web sayfasını çağırmak istediklerinde aspx uzantılı dosyanın yolunu vermeleri gerekir. Örneğin *HesapMakinesi.aspx* dosyası, *http://localhost//HesapMakinesi.aspx* şeklinde çağrılabilir. *Cs* ve *designer.cs* dosyaları son kullanıcı tarafından ulaşılamaz. *Projeye* yeni bir *web form* eklemek için *Solution Explorer*'da *proje* seçildikten sonra sağ tıklanarak ya da *Project* menüsü kullanılarak *Add New Item* seçeneği seçilmelidir (Resim 6.6). Add New Item diyalog penceresinden Web Form seçildikten sonra dosya adı bölümüne *HesapMakinesi.aspx* yazılarak projeye yeni bir web form eklenmiş olur.



Resim 6.6 Add New Item Diyalog Kutusu

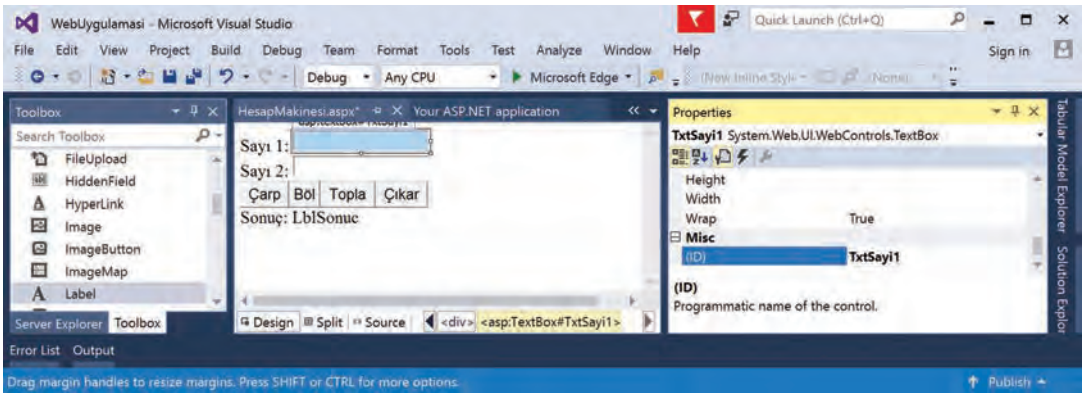
Projeye yeni bir web form eklendiğinde aspx dosyasının içeriği aşağıdaki gibi olur.

```
1   <%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="HesapMakinesi.aspx.cs"
    Inherits="WebUygulamasi.HesapMakinesi" %>
2   <!DOCTYPE html>
3   <html xmlns="http://www.w3.org/1999/xhtml">
4   <head runat="server">
5       <title></title>
6   </head>
7   <body>
8       <form id="form1" runat="server">
9       <div>
10      </div>
11      </form>
12  </body>
13  </html>
```

Aspx uzantılı dosyanın içeriği görüldüğü üzere HTML, CSS ve JavaScript kodları şeklinde olmaktadır. Sayfanın birinci satırında bu *aspx* dosyası ile ilişkili C# dosyasının adının yazıldığı bir tanımlama bulunmaktadır. Ayrıca web sayfasına otomatik olarak bir *<form>* etiketi eklenmektedir ve bu etikete *runat="server"* özelliği eklenmiştir. Böylece bu formun içine eklenecek tüm kontrollerin yönetimi C# dosyası tarafından server üzerinde yönetilebilmektedir. cs uzantılı dosyasının içeriği ise aşağıdaki gibi olur.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Web;
5   using System.Web.UI;
6   using System.Web.UI.WebControls;
7   namespace WebUygulamasi
8   {
9       public partial class HesapMakinesi: System.Web.UI.Page
10      {
11          protected void Page_Load(object sender, EventArgs e)
12          {
13          }
14      }
15  }
```

Görüldüğü üzere *cs* dosyasının içeriği C# kodları şeklindedir. Sistem otomatik olarak *System.Web.UI.Page* sınıfını miras alan *aspx* dosya adı ile aynı isimli bir sınıf tanımlaması yapmaktadır. Web sayfasında gerçekleşen olaylar ile ilgili olarak yazılacak C# kodlar genellikle yukarıda belirtilen sınıfa birer üye fonksiyonu olarak yazılması gerekmektedir. *Page_Load* fonksiyonu ise web sayfası ilk yüklenirken tetiklenen fonksiyondur. *Designer.cs* dosyası ise Visual Studio ile yapılan görsel tasarım sırasında otomatik olarak oluşturulan bir dosyadır. Bu dosya üzerinde değişiklik yapılmaması tavsiye edilmektedir. Çünkü bu dosyada yapılacak değişiklik sonrasında görsel tasarım ekranının doğru şekilde çalışmaması söz konusu olabilir. Dosyanın kodlarında sınıflar *partial class* olarak tanımlanır. *Partial* ifadesi kullanılarak bir sınıfın tanımı birden fazla dosyaya bölünebilmektedir. Derleyici daha sonra bu *partial* olarak tanımlanmış ismi aynı olan sınıfları birleştirecek ve tek bir sınıf gibi çalışmasını sağlayacaktır. Aşağıda Resim 6.7'de basit bir hesap makinesinin tasarımı yer almaktadır.



Resim 6.7 Hesap Makinesi Web Sayfasının Tasarımı

Resimde görüldüğü gibi basit bir web sayfası tasarımı yapılmıştır. *ToolBox* kullanılarak iki adet *TextBox,* dört adet *Button* ve bir adet *Label* nesnesi web sayfasının içine eklenmiştir. Daha sonra eklenen bu nesneler seçilerek özellikleri sağ taraftaki Properties panelinden değiştirilmiştir. *TextBox* metin kutusudur ve kullanıcının metin girişi yapmasını sağlar. *Button* tıklanabilen düğmedir. *Label* ise etikettir, kod ile istenilen metin ifadeler *label* içine yazdırılarak kullanıcıya gösterilebilmektedir. *TextBox*'ların *ID*'leri *TxtSayi1* ve *TextSayi2* olarak belirlenmiştir. *Button*'ların *ID*'leri ise *BtnCarp, BtnBol, BtnTopla* ve *BtnCikar* olarak belirlenmiştir. Ayrıca butonların *Text* özellikleri *Çarp, Böl, Topla* ve *Çıkar* olarak belirlenmiştir. *Label*'ın *ID*'si ise *LblSonuc* olarak belirlenmiştir. Görüldüğü üzere web sayfasına eklenen nesneler isimlendirilirken belli bir sistematik kullanılmıştır. Böylece bu tür değişkenleri anlamlandırmak ve kontrolünü sağlamak kolaylaşacaktır. Faklı bir yaklaşım ile farklı şekilde isimlendirme de yapılabilir. Resim 6.7'nin orta bölümünün altında bulunan *Design, Split, Source* düğmeleri kullanılarak farklı görünümlere geçilebilir. *Design,* görsel tasarım ekranıdır. *Source* kaynak kod düzenleme editörüdür. *Split* ise hem tasarım ekranı hem de kaynak düzenleme editörünü alt alta gösterilmesini sağlar. Aşağıda *HesapMakinesi.aspx* dosyasının düzenlenen bölümleri yer almaktadır.

```
1   Sayı 1:
2   <asp:TextBox ID="TxtSayi1" runat="server"></asp:TextBox>
3   <br />
4   Sayı 2:
5   <asp:TextBox ID="TxtSayi2" runat="server"></asp:TextBox>
6   <br />
7   <asp:Button ID="BtnCarp" runat="server" Text="Çarp" />
8   <asp:Button ID="BtnBol" runat="server" Text="Böl" />
9   <asp:Button ID="BtnTopla" runat="server" Text="Topla" />
10  <asp:Button ID="BtnCikar" runat="server" Text="Çıkar" />
11  <br />
12  Sonuç:
13  <asp:Label ID="LblSonuc" runat="server" Text=""></asp:Label>
```

Görüldüğü üzere tasarım ekranında yapılan işlemler için otomatik olarak html kodları oluşturulmaktadır. *Toolbox*'tan eklenen nesneler için ise *asp:TextBox* gibi bir etiket eklenmektedir. Bu nesneler aynı zamanda *HesapMakinesi* sınıfının birer üyesi olmaktadırlar. *HesapMakinesi.aspx.designer.cs* dosyasının içeriği aşağıdaki gibidir.

```
1   public partial class HesapMakinesi {
2   protected    global::System.Web.UI.HtmlControls.HtmlForm form1;
3   protected    global::System.Web.UI.WebControls.TextBox TxtSayi1;
4   protected    global::System.Web.UI.WebControls.Button BtnCarp;
5   protected    global::System.Web.UI.WebControls.Button BtnBol;
6   protected    global::System.Web.UI.WebControls.Button BtnTopla;
7   protected    global::System.Web.UI.WebControls.Button BtnCikar;
8   protected    global::System.Web.UI.WebControls.Label LblSonuc;
9   }
```

Görüldüğü üzere tasarım ekranında web sayfasına eklenen nesneler için otomatik olarak *designer.cs* dosyasında ilgili eleman için bir satır eklenmiştir. Böylece bu nesneler *HesapMakinesi* sınıfının birer üye değişkeni olmuşlardır. *HesapMakinesi* sınıfı içindeki fonksiyonlar ile bu üye değişkenlere ulaşılabilir. Bu web sayfasında kullanıcı metin kutularına sayısal değerler girdikten sonra *Çarp, Böl, Topla* veya *Çıkar* düğmelerine bastığında *LblSonuc* isimli *Label*'da hesaplama sonucunu görmelidir. Örneğin metin kutularına sırasıyla 12 ve 4 girildiğinde ve *Çarp* düğmesine basıldığında sonuç 48 olarak *LblSonuc* üzerinde görülmelidir. Düğmenin tıklanması bir olaydır. Bu olay gerçekleştiğinde belirlenen kodlar çalıştırılmalı ve istenilen işlemleri yapmalıdır. Bir nesnenin varsayılan olayını eklemek için o nesne üzerine çift tıklanabilir. Düğmelerin varsayılan olayı tıklama olayıdır. Farklı olaylar eklemek için ise nesne seçildikten sonra *Properties* panelinin *Event* bölümünden ilgili olayın üzerine çift tıklamak gerekir. Bir nesneye olay eklendiğinde *cs* dosyasında sınıfa bir üye fonksiyon eklenecektir. Bu üye fonksiyon o olay tetiklendiğinde çalışacak olan fonksiyondur. Genellikle bu fonksiyon *BtnCarp_Click* (BtnCarp isimli düğme için Click-Tıklama olayı) gibi otomatik olarak isimlendirilir. Aşağıda *HesapMakinesi.aspx.cs* dosyasının tamamlanmış şekli yer almaktadır.

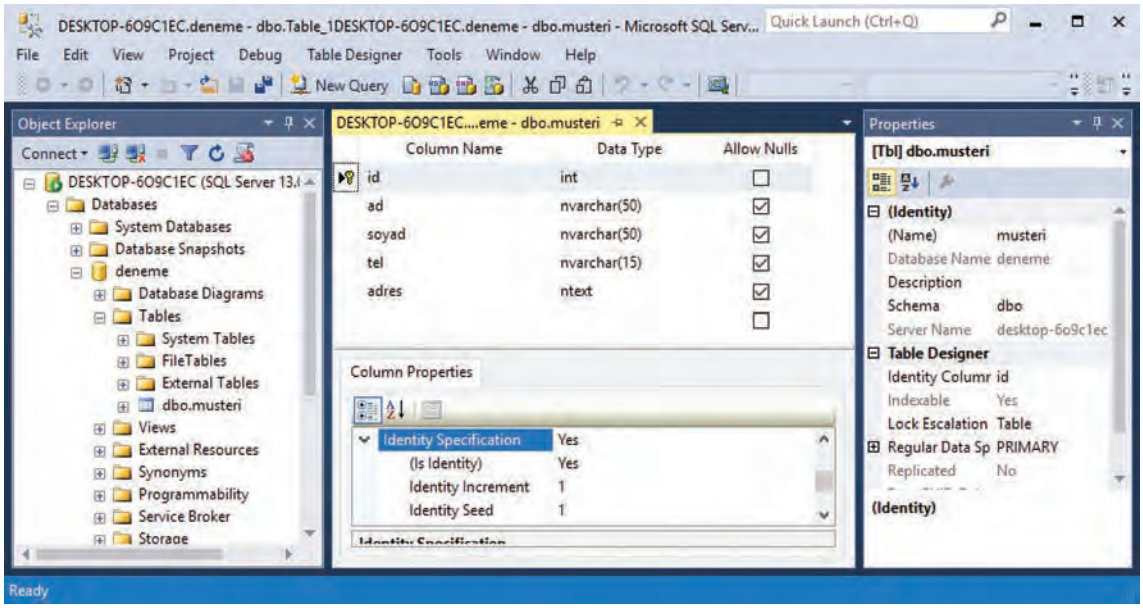
```
1   public partial class HesapMakinesi : System.Web.UI.Page
2   {
3       private double sayi1;
4       private double sayi2;
5       protected void Page_Load(object sender, EventArgs e)
6       {
7       }
8       private void SayilariOku()
9       {
10          sayi1 = Convert.ToDouble(TxtSayi1.Text);
11          sayi2= Convert.ToDouble(TxtSayi2.Text);
12      }
13      protected void BtnCarp_Click(object sender, EventArgs e)
14      {
15          SayilariOku();
16          double sonuc = sayi1 * sayi2;
17          LblSonuc.Text = sonuc.ToString();
18      }
```

```
19      protected void BtnBol_Click(object sender, EventArgs e)
20      {
21          SayilariOku();
22          double sonuc = sayi1 / sayi2;
23          LblSonuc.Text = sonuc.ToString();
24      }
25      protected void BtnTopla_Click(object sender, EventArgs e)
26      {
27          SayilariOku();
28          double sonuc = sayi1 + sayi2;
29          LblSonuc.Text = sonuc.ToString();
30      }
31      protected void BtnCikar_Click(object sender, EventArgs e)
32      {
33          SayilariOku();
34          double sonuc = sayi1 - sayi2;
35          LblSonuc.Text = sonuc.ToString();
36      }
37  }
```

Kodlar incelendiğinde öncelikle sınıfa iki adet *sayi1* ve *sayi2* isminde üye değişken eklenmiştir. Daha sonra *SayilariOku* isminde bir üye fonksiyon yazılmıştır. Bu üye fonksiyonun amacı kullanıcının girmiş olduğu değerleri okuyarak *sayi1* ve *sayi2* üye değişkenlerine aktarmaktır. Böylece bu kodlar tüm olay fonksiyonlarında tekrar tekrar yazılmamıştır. Kullanıcının girmiş olduğu değer metin kutusunun *Text* özelliği ile elde edilmektedir. Gelen bu değer *string* olduğu için doğrudan *double* türü bir değişkene atanamaz. O yüzden *Convert* sınıfı kullanılarak *string* değerin *double*'a dönüştürülmesi sağlanmıştır. Eğer kullanıcı metin kutusuna sayısal bir değer girmez ise çalışma hatası (runtime error) meydana gelir ve programın çalışması sonlanır. Bu tür hataları önlemek için tedbir almak uygun olacaktır. Düğmelere ait olay fonksiyonlarında ise öncelikle *SayilariOku* üye fonksiyonu çağrılmıştır. Bu fonksiyon sayesinde kullanıcın girmiş olduğu değerler *sayi1* ve *sayi2* üye değişkenlerine aktarılır. Daha sonra *sonuc* isminde bir değişken oluşturulmuş ve hesaplama işlemi gerçekleştirmiştir. Son olarak hesaplanan değer *LblSonuc*'un *Text* özelliğine yazılmıştır. *Text* özelliği sadece *string* türü değerleri kabul ettiği için *sonuc* değişkeninin değeri *ToString()* ile *string*'e dönüştürülmüştür.

Veritabanı İşlemleri

ASP.NET ile birçok farklı veritabanı sunucu yazılımını kullanmak mümkündür. Bu bölümde ASP.NET ile Microsoft SQL Server veritabanı sunucu yazılımını ile veritabanı işlemlerinin nasıl yapılabileceği hakkında bilgi verilecektir. Öncelikle Microsoft SQL Server Management Studio (MSSMS) kullanılarak bir veritabanı ve bu veritabanı içinde tablolar oluşturmak gerekir. MSSMS ile SQL Server'a bağlandıktan sonra *Databases* bölümü üzerine sağ tıklanarak *New Database* seçeneği seçilir. Açılan diyalog penceresinde veritabanı adı yazılır ve gerekli görülen ayarlar yapıldıktan sonra *Ok* düğmesi tıklanarak yeni veritabanı oluşturulmuş olur. Oluşturulan veritabanına yeni tablo eklemek için öncelikle yeni veritabanının açılması gerekmektedir. Veritabanı açıldıktan sonra *Tables* bölümüne sağ tıklanır ve açılan menüden *New/Table* seçeneği seçilerek yeni *Tablo* oluşturma ekranının açılması sağlanır (Resim 6.8).



Resim 6.8 Yeni Tablo Oluşturma Ekranı

Resimde görüldüğü gibi, *deneme* veritabaninda *musteri* isminde bir tablo oluşturulmaktadır. Tabloda *id* alanı *int* türünde birincil anahtar olarak tanımlanmıştır. *Is Identity* özelliği *Yes* yapılarak otomatik artan sayı olması sağlanmıştır. Ad, soyad ve tel alanları *nvarchar* olarak tanımlanmıştır. *Nvarchar* türü değişken sayıdaki unicode alfa numerik verileri saklamak için kullanılır. Aynı şekilde *ntext* ise unicode metinleri saklamak için kullanılır.

ASP.NET ile veritabanı bağlantısı yapmak için genel olarak iki yöntem olduğu söylenebilir: ADO.NET ve Entity Framework. ADO.NET aslında veritabanı bağlantılarını yönetmek amacıyla kullanılan .NET içindeki sınıfları temsil etmektedir. ADO.NET kullanıldığında temel olarak veritabanı SQL sorguları ile yönetilmektedir. Entity Framework ise Microsoft tarafından geliştirilen ver veritabanı bağlantılarını yönetmek için kullanılan farklı bir yöntem sunmaktadır. Entity Framework ile veritabanındaki tablolar sınıflara modellenmektedir. Veritabanı üzerindeki işlemler ise bu model sınıfları üzerinden gerçekleştirilmektedir. Özellikle büyük projelerde Entity Framework tercih edilmektedir. Bu bölümde ise ADO.NET sınıfları basit bir şekilde tanıtılacaktır.

Veritabanı üzerinde SQL komutları çalıştırmak için öncelikle *System.Data* ve *System.Data.SqlClient* isim uzaylarını çalışmaya eklemek uygun olacaktır. SQL komutları çalıştırmadan önce veritabanına bağlantı yapmak gerekir. Daha sonra bağlantı yapılan veritabanına SQL komutları gönderilebilir. Aşağıdaki INSERT SQL komutunu çalıştırma ile ilgili bir örneğe yer verilmiştir.

```
1   string baglantiMetni = "User ID=sa;Password=deneme123;Initial
    Catalog=deneme; Data Source=localhost";
2   SqlConnection sqlBaglantisi = new SqlConnection
    (baglantiMetni);
3   sqlBaglantisi.Open();
4   SqlCommand sqlKomutu = new SqlCommand();
5   sqlKomutu.Connection = sqlBaglantisi;
6   sqlKomutu.CommandText = "INSERT INTO dbo.musteri
    (ad,soyad,tel,adres) VALUES('Ali','Veli','222','Eskişehir')";
7   sqlKomutu.ExecuteNonQuery();
```

İlk olarak bir bağlantı metni oluşturulmuştur. Bu bağlantı metninde kullanıcı adı (sa), şifresi (deneme123), veritabanı adı (deneme) ve veritabanının adresi (localhost) yer almaktadır. Farklı durumlarda bağlantı metninde değişiklikler olabilir. Alternatif olarak *Toolbox*'ta *SqlDataSource* nesnesi kullanılarak SQL

Server bağlantısı oluşturulabilir. Bağlantı metnini oluşturduktan sonra bu bağlantı metni ile bir *SqlConnection* nesnesi oluşturulmuş ve bu nesnenin *Open* fonksiyonu çağrılarak veritabanına bağlantı yapılması sağlanmıştır. Daha sonra *SqlCommand* nesnesi oluşturulmuştur. *SqlCommand* nesnesinin *Connection* özelliğine *sqlBaglantisi* atanmıştır. Böylece SQL komutlarının hangi veritabanı üzerinde çalıştırılacağı belirlenmiş olur. Aynı anda birçok veritabanına bağlantı yapmak mümkünüdür. *CommandText* özelliğine ise SQL komutları yazılmaktadır. SQL komutlarını veritabanında çalıştırmak için *ExecuteNonQuery* fonksiyonundan yararlanılmıştır. Bu fonksiyon geriye değer döndürmeyen SQL komutlarının çalıştırılmasında kullanılmaktadır. Veritabanından SELECT SQL komutu ile bilgi okumak için aşağıdaki yöntem kullanılabilir.

```
1   string baglantiMetni = "User ID=sa;Password=deneme123;Initial
    Catalog=deneme; Data Source=localhost";
2   SqlConnection sqlBaglantisi = new SqlConnection(baglantiMetni);
3   sqlBaglantisi.Open();
4   SqlCommand sqlKomutu = new SqlCommand();
5   sqlKomutu.Connection = sqlBaglantisi;
6   sqlKomutu.CommandText = "SELECT id,ad,soyad,tel,adres FROM dbo.mus
    teri WHERE ad=@adParametresi";
7   SqlParameter adParam = new SqlParameter("adParametresi",
    SqlDbType.NVarChar, 50);
8   adParam.Value = "ali";
9   sqlKomutu.Parameters.Add(adParam);
10  SqlDataReader okuyucu = sqlKomutu.ExecuteReader();
11  while (okuyucu.Read())
12  {
13      int id = okuyucu.GetInt32(0);
14      string ad = okuyucu.GetString(1);
15      string soyad = okuyucu.GetString(2);
16      string tel = okuyucu.GetString(3);
17      string adres = okuyucu.GetString(4);
18      Response.Write(ad + " " + soyad + "<br />");
19  }
```

SQL Server ile bağlantı bölümleri önceki örnek ile aynıdır. *SqlCommand* nesnesi olan *sqlKomutu*'nun *CommandText* özelliğine SELECT SQL sorgusu yazılmıştır. Sorguya dışarıdan bir parametre eklenmek için eklenecek parametre adı @ işareti ile isimlendirilmelidir. Örnekte *@adParametresi* ifadesi kullanılmıştır. Bu parametreye değer eklemek için *SqlParameter* nesnesi oluşturulmuş ve *Value* özelliğine "ali" atanmıştır. Daha sonra bu parametre nesnesi *sqlKomutu.Parameters.Add* fonksiyonu ile eklenmiştir. Böylece sorgu "SELECT * FROM dbo.Musteri WHERE ad='ali'" şekline dönüşmüştür. Bir başka deyişle *ad* alanı *ali* olan kayıtlar tablodan çekilecektir. *SqlCommand* nesnesi hazırlandıktan sonra okuma işlemini başlatmak için *sqlKomutu.ExecuteReader()* fonksiyonunu *SqlDataReader* nesnesine atanmıştır. Bir *while* döngüsü içinde *okuyucu.Read()* ifadesi ile kayıtların okunması sağlanır. Okunacak alanın türüne göre *GetInt* ve *GetString* gibi fonksiyonlar kullanılabilir. Bu fonksiyonlar parametre olarak okunan alana sıfırdan başlayarak sıra numarası vermektedir. Örneğin ilk sırada *id* olduğu ve *id* de *int* türünde olduğu için *GetInt32(0)* ifadesi kullanılmıştır.

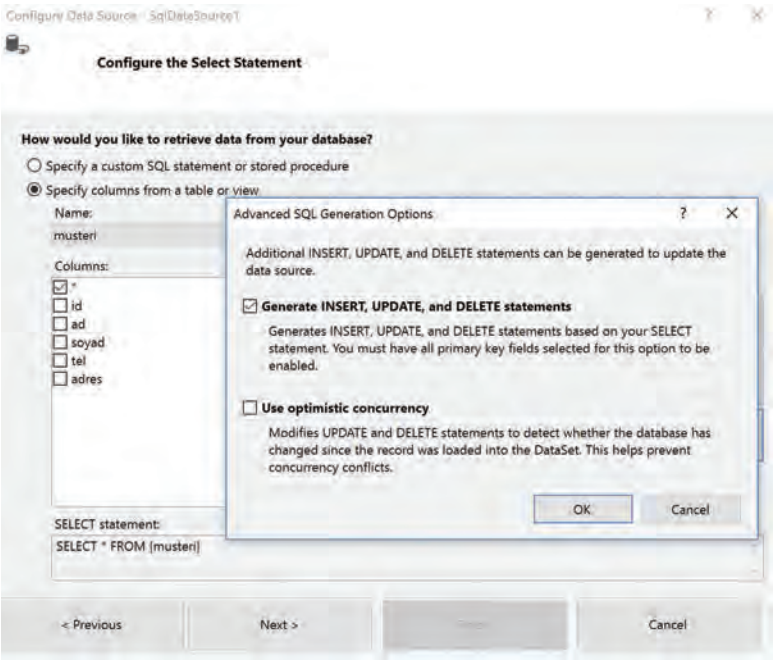
dikkat

Bu işlem SqlParameter kullanmadan da yapılabilirdi. Ancak SqlParameter sınıfı, parametreleri güvenlik kontrolünden geçirdiği için daha güvenlidir. Mümkün olduğunca kullanıcıdan gelen verileri doğrudan SQL metninin içinde kullanmaktan kaçınmak gerekir.

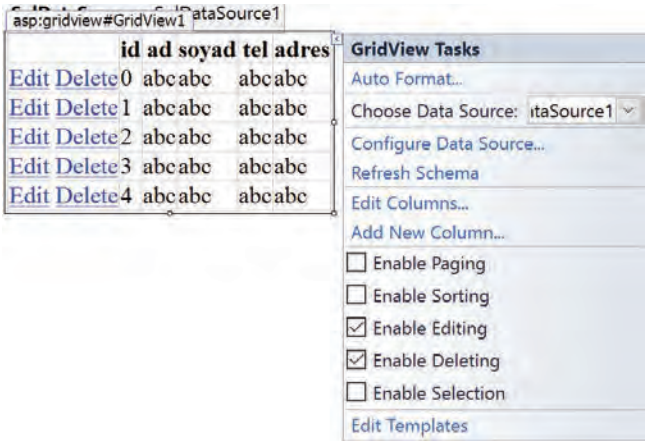
Musteri tablosundaki bilgileri listelemek, silmek ve yeni kayıt eklemek için aşağıdaki örnek yapılmıştır. Tablodaki kayıtları listelemek için *Musteriler* isminde yeni bir Web Form eklenmiştir. Tablodaki kayıtları listelemek için *GridView* nesnesinden yararlanılabilir. *GridView* nesnesnin çalışması için aynı zamanda *SqlDataSource* nesnesinin de web sayfasında bulunması gerekmektedir. *SqlDataSource* nesnesi web sayfa-

sına ekledikten sonra *Cofigure Data Source* seçeneği seçilerek ayarlaması yapılmalıdır. Açılan diyalog penceresinden *New Connection String* düğmesi kullanılarak SQL Server ile bağlantının kurulması sağlanmalıdır. SQL Server bağlantısı tamamlandıktan sonra *Next* ile bir sonraki sayfaya geçilmelidir. Bu sayfadaki "Save the Connection String to the Application Configuration File" seçeneğine tıklanabilir. ASP.NET uygulamalarında kök dizinde bulunan *web.config* dosyası uygulamanın ayar dosyasıdır. Bu seçenek seçildiğinde bağlantı metni bu dosya içine kayıt edilecek diğer web sayfaları tarafından kullanılabilir hale gelmiş olacaktır. Bir sonraki sayfada ise "Specify columns from table or view" seçeneğinin altından *musteri* tablosu seçilmelidir. Daha sonra Aynı pencerede *Advanced* seçeneği seçilerek "Generate INSERT, UPDATE and DELETE statements" seçilmelidir (Resim 6.9). Böylece gridview üzerinden kayıtları silmek ve güncellemek mümkün hale gelir.

SqlDataSource nesnesinin ayarları tamamlandıktan sonra web sayfasına *GridVew* nesnesi eklenir. *GridView* eklendikten sonra Resim 6.10'da görülen *Choose Data Source* seçeneğinden biraz önce ayarlamış olan *SqlDataSource* nesnesinin adı seçilir. Daha sonra *Enable Editing* ve *Enable Deleting* seçenekleri seçilerek *GridView* üzerinde silme ve güncelleme işlemlerinin yapılabilmesi sağlanır.



Resim 6.9 SqlDataSource Nesnesinin Ayarları



Resim 6.10 GridView Nesnesinin Ayarları

Oluşturulan Web Form çalıştırıldığında tablodaki kayıtların listelendiği ve kayıtlar üzerinde güncelleme ve silme işlemlerinin yapılabildiği görülecektir. Görüldüğü üzere Visual Studio kullanılarak tek satır kod yazmadan birçok işi yapabilen bir Web Form geliştirilmiştir. *GridView*'in özellikleri değiştirilerek güzel görünümlüm bir tablo elde edilebilir. *Musteri* tablosuna yeni bir kayıt eklemek için ise *MusteriEkle* Web Formu oluşturulmuştur. Görsel olarak oluşturulan Formun kodları aşağıda verilmiştir.

```
1   <strong>Yeni Kayıt Ekle<br />
2   </strong>Adı:<asp:TextBox ID="TxtAd" runat="server"></asp:TextBox>
3   <br />
4   Soyadı:<asp:TextBox ID="TxtSoyad" runat="server"></asp:TextBox>
5   <br />
6   Telefon:<asp:TextBox ID="TxtTel" runat="server"></asp:TextBox>
7   <br />
8   Adres:<asp:TextBox ID="TxtAdres" runat="server"></asp:TextBox>
9   <br />
10  <asp:Button ID="BtnKaydet" runat="server" OnClick="BtnKaydet_Click"
    Text="Kaydet" />
```

Web sayfasında kullanıcının giriş yapabileceği *ad, soyad, telefon* ve *adres* metin kutuları ve *kaydet* düğmesi bulunmaktadır. *Kaydet* düğmesi tıklandığında çalışan fonksiyonun kodları aşağıda verilmiştir.

```
1   protected void BtnKaydet_Click(object sender, EventArgs e)
2   {
3   string baglantiMetni = "User ID=sa;Password=deneme123;Initial
    Catalog=deneme; Data                    Source=localhost";
4   SqlConnection sqlBaglantisi = new SqlConnection(baglantiMetni);
5   sqlBaglantisi.Open();
6   SqlCommand sqlKomutu = new SqlCommand();
7   sqlKomutu.Connection = sqlBaglantisi;
8   sqlKomutu.CommandText = "INSERT INTO dbo.musteri
    (ad,soyad,tel,adres) VALUES(@ad,@soyad,@tel,@adres)";
9   sqlKomutu.Parameters.Add("ad", SqlDbType.NVarChar, 50).Value =          TxtAd.
    Text;
10  sqlKomutu.Parameters.Add("soyad", SqlDbType.NVarChar, 50).Value = TxtSoyad.
    Text;
11  sqlKomutu.Parameters.Add("tel", SqlDbType.NVarChar, 15).Value = TxtTel.Text;
12  sqlKomutu.Parameters.Add("adres", SqlDbType.NText).Value = TxtAdres.Text;
13    sqlKomutu.ExecuteNonQuery();
14    sqlBaglantisi.Close();
15    Response.Redirect("Musteriler.aspx");
16  }
```

Kodlar incelendiğinde öncelikle veritabanı bağlantısı gerçekleştirilmiştir. Daha sonra *SqlCommand* nesnesi ile INSERT SQL sorgusu hazırlanmıştır. Sorguya a*d, soyad, tel* ve *adres* metin kutularına girilen değerler parametre olarak eklenmiş ve sorgu çalıştırılmıştır. 14. satırda veritabanı bağlantısı kapatılmış ve 15. satırda *Musteriler.aspx* dosyasına yönlendirme yapılmıştır.

Öğrenme Çıktısı

6 Web Form kullanarak uygulama geliştirebilme
7 Veritabanı işlemleri yapabilme

Araştır 4

ASP.NET'te Web formları oluşturulduğunda neden kodlar tek bir sayfada değil ayrı belgelerde kaydedilir?

İlişkilendir

ASP.NET Web formlarını PHP Web formları ile karşılaştırınız.

Anlat/Paylaş

Visual Studio ile ASP.NET Web formu geliştirme deneyimlerinizi paylaşınız.

1 ASP.NET'in ve .NET Framework'ün çalışma prensiplerini açıklayabilme

ASP.NET

1 ASP.NET, Microsoft firmasının sunucu temelli uygulama geliştirme çözümüdür. ASP.NET, .NET Framework mimarisi üzerine inşa edilmiştir. Herhangi bir .NET dili ile ASP.NET uygulama geliştirmek mümkündür. .NET Framework, Microsoft tarafından geliştirilmiş bir uygulama geliştirme platformudur. Bu platform ile geliştirilen yazılımlar işletim sistemi ve donanımdan bağımsız olmakta ve .NET Framework'in Common Language Runtime ismindeki yapısı üzerinde çalışmaktadır. .NET Framework için geliştirilen bir yazılımın çalışması için sistemde mutlaka .NET Framework kurulu olmalıdır. Microsoft son yıllarda .NET Core isminde açık kaynak kodlu bir yapı üzerinde çalışmaktadır. Bu yapının gelişmesiyle birlikte .NET uygulamaları çok farklı işletim sistemlerinde de çalışabilir hale gelmektedir. Örneğin .NET Core ile bir ASP.NET uygulamasını bir Linux sunucu üzerinde çalıştırmak mümkün olacaktır.

2 ASP.NET kurulum gereksinimlerini sıralayabilme

Kurulum

2 ASP.NET uygulamalarını çalıştırabilmek için işletim sisteminde öncelikle IIS yazılımının çalışıyor olması gerekmektedir. Böylece ASP.NET uygulamaları test edilebilecek ve yayınlanabilecektir. Microsoft Windows Server ve diğer Microsoft Windows işletim sistemlerinde IIS yazılımı bütünleşik olarak gelmektedir. IIS'i açmak için Denetim Masasındaki "Program kaldır veya değiştir" penceresinden yararlanılır. IIS kurulduktan sonra SQL Server gibi bir veri tabanı sunucu yazılımı kurularak gerekli bağlantılar yapılır. Son olarak Microsoft Visual Studio bütünleşik geliştirme ortamına ihtiyaç vardır.

öğrenme çıktıları ve bölüm özeti

3 Değişken türleri ve tür dönüşümlerini kullanabilme

4 Dizileri ve ArrayList'leri kullanabilme

5 Kontrol yapılarını kullanabilme

6 Sınıf tanımlaması yapabilme

C# Programlama Dili

3 C# programlama dili değişkenler konusunda katı kurallara sahip bir dildir. Basit türler ve karmaşık türler bulunmaktadır. C# ile kullanılabilecek basit türler şu şekildedir: bool, byte, sbyte, char, decimal, double, float, int, uint, long, ulong, short, ushort, string. C# programlama dilinde değişkenler kullanılmadan önce mutlaka tanımlanmalıdır. Değişkenlerin hafızadan kapladıkları alan değişiklik gösterebilmektedir. Bu bakımdan kapsam olarak daha küçük değişken türleri, daha büyük değişken türlerine doğrudan atama yapılabilir. Ancak tam tersi mümkün değildir. Bunun için tür dönüşümü yapılmalıdır. Tür dönümüne cast işlemi adı verilmektedir.

4 C# ile birçok değeri içinde barındıran dizeler tanımlanabilir. C# ile dizi tanımlarken dizin boyutunun belirtilmesi gerekir. Dizilerin boyutları daha sonradan değiştirilemez. Dolayısıyla dizilere yeni eleman eklemek veya diziden eleman silmek mümkün değildir. Ayrıca yine dizi boyutları sabit olarak tanımlanmak kaydıyla çok boyutlu diziler oluşturmak da mümkündür. C# ile dizi tanımlamanın esnek bir yapısı olmaması dolayısıyla daha esnek yapıya sahip ArrayList sınıfı geliştirilmiştir. ArrayList sınıfı System. Collections'ın bir üyesidir ve aslında bir dizi tanımlamasıdır. ArrayList ile tanımlanan diziler ekleme, çıkarma ve sıralama gibi birçok işlem gerçekleştirilebilmektedir.

5 C# programlama dilinde de diğer dillerde olduğu gibi if, for, while, do while ve foreach kontrol yapıları mevcuttur. If kontrol yapısı ile bir bloğun çalıştırılıp çalıştırılmayacağı bir karşılaştırma ifadesine bağlı hale getirilebilmektedir. Karşılaştırma ifadelerinin sonuçları bool türünde olmaktadır. Bool türü ise sadece true ve false değerlerini barındırabilmektedir. Eğer karşılaştırma ifadesinin sonu true ise blok içindeki kodlar çalıştırılır, false ise çalıştırılmaz. For, while, do while ifadeleri ise döngü oluşturmak için kullanılmaktadır. Bu ifadelerdeki en önemli yer karşılaştırma ifadesi bölümüdür. Karşılaştırma ifadesinin sonucuna göre döngü içindeki kodlar çalıştırılır ya da çalıştırılmaz. Kodlar çalıştırıldıktan sonra döngü başa döner ve tekrar kontrol yapılır. Eğer karşılaştırma ifadesi tekrar true sonucunu üretir ise döngü bloğundaki kodlar tekrar çalıştırılır. Karşılaştırma ifadesi false değerini üretene kadar döngü içindeki kodlar tekrar tekrar çalıştırılacaktır.

6 Sınıflar nesne yönelimli programlamanın temel taşıdır. C# programlama dili de nesne yönelimli programlama konusunda son derece gelişmiş bir programlama dilidir. Miras alma, sanallaştırma, arayüz tanımlama gibi gelişmiş özellikleri destekler. Sınıf tanımlaması yaparken class kelimesi kullanılmaktadır. Sınıfların üye değişkenleri, üye fonksiyonları ve üye özellikleri bulunmaktadır. Sınıflara üye tanımlaması yaparken private, public ve protected kelimeleri kullanılmaktadır. Bu kelimeler tanımlanan üyenin erişim hakları ile ilgilidir. Private olarak tanımlanan bir üyeye sadece sınıf içinde erişilebilir. Public olarak tanımlanan bir üyeye ise sınıf dışından da erişilebilir. Genellikle üye değişkenler private olarak tanımlanmaktadır. Üye değişkenleri sınıfın veri saklama yerleridir. O yüzden kontrollü erişilmesi tercih edilmektedir. Sınıflarda üye değişkenlerine kolay bir şekilde değer ataması yapabilmek için özellik adında bir sistem mevcuttur. Bu özelliklerin get ve set blokları kullanılarak üye değişkenlere değer ataması ve değer okuması yapmak mümkündür.

7 Web Form kullanarak uygulama geliştirebilme

8 Veritabanı işlemleri yapabilme

WEB Formları

7 ASP.NET ile uygulama geliştirmek için genel olarak iki yol olduğu söylenebilir: MVC ve Web Form. MVC yapısı kullanıldığında tasarım model, görünüm ve kontrolcü olarak üç parçada ele alınmaktadır. MVC yapısı Web Form'larından daha karmaşık bir yapıdır, ancak büyük karmaşık projelerde tercih edilebilir. Web Form'lar ise olay tabanlı çalışan birbirinden bağımsız web sayfaları oluşturmak için kullanılan bir yapıdır. Visual Studio'nun gelişmiş görsel tasarım yapma olanakları sayesinde çok hızlı yazılımlar geliştirmek mümkündür. Bir Web Form oluşturulduğunda aspx, cs, ve designer.cs olmak üzere üç adet dosya oluşturulur. Aspx dosyasında HTML, CSS ve JavaScript kodları yer alır. Cs dosyası C# kodlarının yazıldığı dosyadır. Designer.cs dosyası ise sistem tarafından otomatik olarak oluşturulur ve programcı tarafından değişiklik yapılmaması gerekir.

8 .NET Framework ile veritabanı erişiminde temel olarak iki yaklaşımdan söz etmek mümkündür: ADO.NET ve Entity Framework. Entity Framework, Microsoft tarafından geliştirilmiştir ve projeye internetten indirilerek eklenmesi gerekir. Temel olarak veritabanındaki tabloların sınıflara modellenmesi mantığı ile çalıştır. ADO.NET'e göre daha karmaşık olsa da bir çok avantajı bulunmaktadır. Özellikle kapsamlı projeler için Entity Framework'ün tercih edilmesi uygun olacaktır. ADO.NET ise veritabanı bağlantısı konusunda daha basit bir yapı sunmaktadır. ADO.NET ile rahatlıkla SQL komutları oluşturup bu komutların veritabanı üzerinde çalıştırılması sağlanabilir. SQL komutları çalıştırmak için System.Data ve System.Data.SqlClient isim uzaylarını using ile eklemek uygun olacaktır. SqlConnection, SqlCommand, SqlParameter, SqlDataReader kullanılabilecek sınıflardır. Farklı ihtiyaçlara yönelik farklı sınıflar da mevcuttur.

neler öğrendik?

1 Aşağıdakilerden hangisi dizi oluşturmak için kullanılır?

A. ( )
B. String
C. Count
D. { }
E. ->

2 Aşağıdakilerden hangisi temel değişken türlerinden biri **değildir?**

A. Int32
B. Arraylist
C. String
D. Uint
E. Float

3 Birbirine bağlı çok sayıda karşılaştırma yapmak ve bu karşılaştırmalar sonucunda tanımlanmış çok sayıdaki bloktan birinin çalışmasını sağlamak için hangi yapı kullanılmalıdır?

A. if-else if
B. for each
C. do while
D. do
E. else

4 xKonum ve yKonum değişkenlerinin değerlerinin koordinat düzleminde (50,60) ile (100,120) arasında olup olmadığını belirmek amacıyla aşağıdaki karşılaştırma ifadelerinden hangisi kullanılabilir?

A. 50<=xKonum<=100 || 60<=yKonum<=120
B. xKonum>=50 || xKonum<=100 &&yKonum>=60 || yKonum<=120
C. 50<=xKonum<=100 && 60<=yKonum<=120
D. xKonum>=50 && xKonum<=100 &&yKonum>=60 && yKonum<=120
E. 50<xKonum<100 && 60<yKonum<120

5 for(int i=1; i<=3; i+=2){ Console. WriteLine("Merhaba Dünya"); } döngüsü çalıştırıldığında "Merhaba Dünya" yazısı ekrana kaç defa yazılır?

A. 3
B. 4
C. 2
D. 5
E. 0

6 Çalışan bir döngüyü başa almak için hangi komut kullanılmalıdır?

A. Foreach
B. While
C. Break
D. Continue
E. Switch

7 Dizi elemanları içinde bir döngü oluşturmak için aşağıdakilerden hangisi **kullanılmaz?**

A. Foreach
B. For
C. Do while
D. While
E. Switch

8 Sınıf üyesi değişkenlerin dışarıdan erişilememesi için hangi kelime kullanılmalıdır?

A. Private
B. Public
C. Void
D. Protected
E. Return

9 Veritabanında SQL komutları çalıştırmak için hangi sınıf kullanılmalıdır?

A. Sqlconnection
B. Sqladapter
C. Sqlparameter
D. Sqlcommand
E. Sqlnet

10 Tasarım ekranında vertabanı bağlantısı yapmaya yarayan toolbox öğesinin adı nedir?

A. Sqldatasource
B. Listview
C. Gridview
D. Sqlconnection
E. Sqloption

1. D	Yanıtınız yanlış ise "Diziler" konusunu yeniden gözden geçiriniz.	6. D	Yanıtınız yanlış ise "Break ve Continue" konusunu yeniden gözden geçiriniz.
2. B	Yanıtınız yanlış ise "Değişkenler ve Türler" konusunu yeniden gözden geçiriniz.	7. E	Yanıtınız yanlış ise "Switch" konusunu yeniden gözden geçiriniz.
3. A	Yanıtınız yanlış ise "If" konusunu yeniden gözden geçiriniz.	8. A	Yanıtınız yanlış ise "Sınıflar" konusunu yeniden gözden geçiriniz.
4. D	Yanıtınız yanlış ise "Operatörler" konusunu yeniden gözden geçiriniz.	9. D	Yanıtınız yanlış ise "Veritabanı İşlemleri" konusunu yeniden gözden geçiriniz.
5. C	Yanıtınız yanlış ise "For" konusunu yeniden gözden geçiriniz.	10. A	Yanıtınız yanlış ise "Veritabanı İşlemleri" konusunu yeniden gözden geçiriniz.

6 Araştır Yanıt Anahtarı

Araştır 1

Yeni adı ile Asp.Net Core 1.0 ile geliştirilen web uygulamaları Windows'un yanısıra MacOS ve Linux işletim sisteminde de çalıştırılabilmektedir. Cross-Platform adı verilen bu yaklaşım ile farklı platformlarda ve platformlar arasında sorunsuz uygulama geliştirmek ve kullanmak mümkün hale gelmektedir. Microsoft ayrıca Asp.Net Core 1.0 ile web uygulamaları geliştirmek için Visual Studio Code aracını ücretsiz olarak sağlamaya başlamıştır. Böylece Visual Studio Code cross-platform çalışabilen bir uygulama geliştirme aracı haline gelmiştir.

Araştır 2

ASP.NET uygulamaları geliştirmek için Microsoft Visual Studio bütünleşik geliştirme ortamından yararlanılır. Visual Studio yazılımının Community, Professional ve Enterprise sürümü bulunmaktadır. Visual Studio'nun Professional ve Enterprise sürümleri ücretli, Community sürümü ise ücretsizdir. Microsoft Visual Studio 2017 sürümünde önmeli yenilikler yapılmıştır. Bu yeniliklerin başında bulut bilişim entegrasyonu ve mobil uygulama geliştirme desteği gelmektedir.

Araştır 3

final>=50 && ortalama>=45 karşılaştırma ifadesi ile kontrol edilebilir.

Araştır 4

ASP.NET'te Web formları oluşturulduğunda aspx dosyası ile aynı isimde bir adet cs uzantılı ve bir adet de designer.cs adlı dosya oluşturulmaktadır. HTML, CSS, JavaScript ve C# kodları tek bir dosyaya yazılarak web sayfası oluşturulabilir. Ancak kodları farklı dosyalarda barındırmak kodların yönetimini kolaylaştırmaktadır

Kaynakça

Aktaş, V. ve Sevinç, T. (2013). *ASP.Net MVC 4.* İstanbul: Kodlab Yayın.

Çelik, A.İ. (2016). *ASP.Net ile SQL Server veri tabanı uygulamaları.* Bursa: Ekin Kitabevi Yayınlari

Erdinç, F. (2016). *C# ile nesne tabanlı programlama.* İstanbul: Abaküs.

Erdinç, F. (2015). *Yeni başlayanlar için C# ile algoritmalara ve programcılığa giriş.* Ankara: Pusula Yayıncılık ve İletişim.

Kaplan, F. (2016). *ASP. Net ile proje geliştirme oku, izle, dinle, öğren.* İstanbul: Kodlab Yayın.

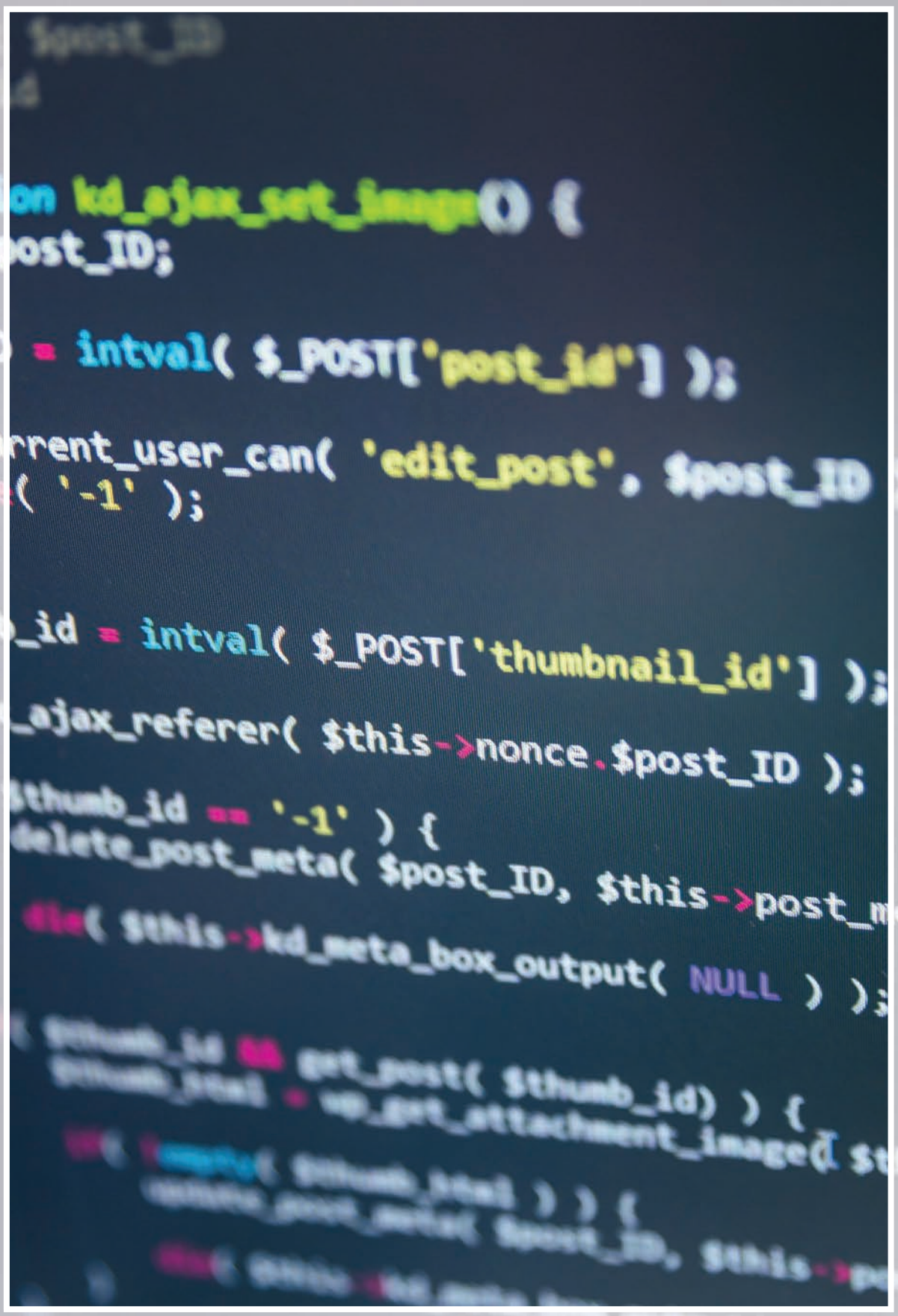
Kızmaz, V.U. (2017). *Uygulamalarla ASP.Net 4.5 oku, izle, dinle, öğren.* İstanbul: Kodlab Yayın.

Kızmaz, V.U. (2016). *Asp.Net Mvc 5 oku, izle, dinle, öğren.* İstanbul: Kodlab Yayın.

Schildt, H. (2011). *C# 4.0.* İstanbul: Alfa Yayınları.

Temur, M. (2013). *ASP.Net 4.5.* İstanbul: Dikeyeksen Yayincilik.

Uzunköprü, S. (2016). *Projeler ile C# 6.0 ve SQL Server 2016.* İstanbul: Kodlab Yayın.


function kd_ajax_set_image() {
$post_ID;
= intval($_POST['post_id']);
current_user_can('edit_post', $post_ID
('-1');
_id = intval($_POST['thumbnail_id']);
_ajax_referer($this->nonce.$post_ID);
thumb_id == '-1') {
delete_post_meta($post_ID, $this->post_
die($this->kd_meta_box_output(NULL));
get_post($thumb_id)) {
wp_get_attachment_image(

Bölüm 7

Python ve Django

öğrenme çıktıları

1 Python Programlama Dili

1 Python dilini kurabilme

2 Python Program Denetimi

2 Python dilinde program denetimi yapılarını açıklayabilme

3 Python ile Nesneye Yönelik Programlama

3 Python dilinde nesneleri kullanabilme

4 Python dilinde tkinter kullanarak masaüstü görsel programlama yapabilme

4 Django Web Çatısı

5 Django web çatısını kurabilme

6 Django şablon kullanabilme

5 Django Modeller

7 Django ile model oluşturabilme

8 Django ile modeller üzerinde işlem yapabilme

Anahtar Sözcükler: • Python Programlama Dili • Nesneye Yönelik Programlama • Tkinter • Django

GİRİŞ

Kitabımızın bu bölümünde, son yıllarda popüleritesi artmış bir dil olan "Python" programlama dilinden bahsedilecektir. Ayrıca, masaüstü programlar yapmamıza olanak veren "tkinter" paketine ve python programlama diliyle yazılmış bir web sayfası geliştirme çatısı olan "Django" ya, kısaca yer verilecektir. Bölüm, konu hakkında ön bilgisi olmayan bireylere yönelik giriş seviyesinde bilgiler içermektedir.

Python, Guido Van Rossum isimli bir programcı tarafından 1991 yılında duyurulmuş zamanla kullanımı artmıştır. Dünya'da kabul görmüş Python, artık Türkiye'de birçok üniversitesinin ilgili bölümlerinde öğretim programlarında yer almaya başlamıştır. Rossum, programlama dilinin ismini The Monty Python adlı bir İngiliz komedi grubunun, Monty Python's Flying Circus adlı gösterisinden esinlenerek vermiştir. İsminin esin kaynağı farklı olmasına rağmen, günümüzde dilin simgesi olarak piton yılanı kullanılmaktadır. Python kelimesinin telaffuzu esas olarak "paytın" şeklinde olmasına karşın kelimeyi "piton" olarak da telaffuz edenlerin sayısı da az değildir.

Guido Van Rossum python hakkında yaptığı röportajı https://www.youtube.com/watch?v=rTTFh7HOlC0 adresinden izleyebilirsiniz.

Açık kaynak kod felsefesi ile geliştirilen Python programlama dili günümüzde Python Yazılım Vakfı desteği ile gelişim sürecine devam etmektedir. İlk sürümü olan Python 1.0' ın 1994'te yayınlanmasından sonra zaman içerisinde Python 2.x ve Python 3.x serileri hayata geçirilmiştir. Bu bölümde, daha güncel seri olan Python 3.x serisine yer verilmiştir. Python 3.x serisinin şu an için tek dezavantajı 2.x serisinde yer alan kütüphanelerinin tümünün aktarılmamış olması olarak görülmektedir. Fakat bu dezavantajın zaman içerisinde giderileceği beklenmektedir.

Python, nesne yönelimli, yorumlamalı, modüler ve etkileşimli yüksek seviyeli bir programlama dili olarak tanımlanmaktadır. Python dili yorumlanan bir dil olma özelliği ile JavaScript programlama dilinin, yapısal sağlamlığı ile de java dilinin özelliklerini birleştiren modern, basit ve temiz söz dizimine sahip, iyi tasarlanmış bir programlama dilidir. Bu programlama dilinin avantajlarını şu şekilde sıralayabiliriz.

- Açık kaynak kodlu ve ücretsiz bir dil olması,
- Yorumlanan ve dinamik bir dil olması,
- Etkileşimli bir dil olması (Yorumlayıcı penceresinde komutlar veri sonuçlarını hemen alabilirsiniz),
- Söz diziminin sade ve okunabilirliğinin kolay olması,
- Modüler bir yapıya sahip olması,
- Hızlı bir şekilde öğrenilebilmesi, yeni başlayanlar için uygun olması,
- Nesneye yönelik programlama dili olması ama bunu mecbur tutmaması,
- Öğretici kaynaklarının oldukça fazla olması,
- Bilimsel araştırmalarda yaygın olarak kullanılması,
- Farklı alanlara özel çok geniş bir kütüphane, modül, paket desteğinin olması,
- Gelişmiş bir grafik kütüphanesinin olması,
- Farklı işletim sistemleri (GNU/Linux, Windows, Mac OS X, AS/400, BeOS, MorphOS, MS-DOS, OS/2, OS/390, z/OS, RiscOS, S60, Solaris, VMS, Windows CE, HP-UX, iOS, Android vb.) üzerinde çalışabilmesi (kodu bir kez yazmak yeterli)
- Farklı dillerde yazılmış kodların entegrasyonunu desteklemesi,
- Uluslararası saygınlığının olması,
- Az kod yazarak çok iş yapmayı desteklemesi,
- 3.x serisinden itibaren UTF-8 desteğinin olması (kodlama da ğ,ı,ç,ü,ö,ş,İ,Ş,Ğ,Ü,Ö,Ç gibi karakterler kullanılabilir).

Python dilinde kod bloklarını belirlemede begin/end veya süslü parantez {, } gibi ifadeler yerine girintilime kullanılmaktadır.

Python programlama dili ile web uygulamaları, ağ ve soket programcılığı, veri analizi programları, makina öğrenmesi ile ilgili programlar, doğal dil işleme üzerine programlar, bilimsel içerikli programlar (SciPy), veritabanı programları gibi birçok farklı program kodlanabilir. Bunlar dışında masaüstü programcılığı ve oyun programcılığı için de Python kullanılabilir. Fakat Python'u bu tip programlamada kullanmak çok uygun ve verimli olmayacaktır. Python, tıpkı Java gibi yorumlanan bir dil olduğundan dolayı bir yorumlayıcıya ihtiyaç duymaktadır. Python bu yapısının getirdiği avantaj nedeni ile platform bağımsızdır.

PYTHON PROGRAMLAMA DİLİ

Python Kurulumu

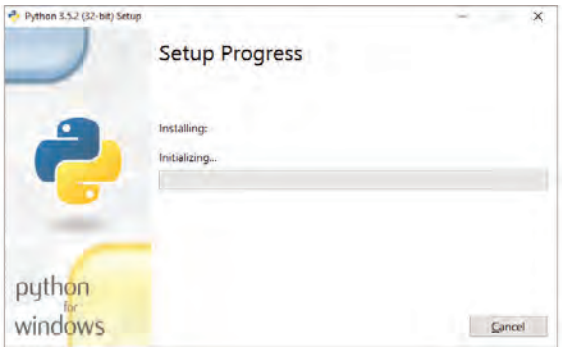
Python dili ile programlama yapabilmek için program dilinin bilgisayarınıza kurmanız gerekir. Daha öncede belirtildiği gibi, python farklı işletim sistemleri üzerinde çalıştırılabilir bir programlama dilidir. Yani, farklı işletim sistemleri için farklı kurulum durumlarına sahiptir. Fakat bu bölümde sadece Windows kullanıcıları için kurulum gösterilecektir.

> ✓ Kurulum için 'web-based installer' seçeneği de kullanılabilir. Bu seçenek ile gerekli dosyalar kurulum esnasında indirilecektir.

Kurulum için öncelikle, "http://www.python.org/downloads" adresinden kurulum dosyalarını indirmemiz gerekmektedir. Sayfada şu an itibari ile en güncel sürümler olan python 3.5.2 ve python 2.7.12 sürümleri bulunmaktadır. Bu bağlantıları kullanarak istediğiniz sürümü bilgisayarınıza indiriniz. İndirdiğiniz dosyayı kullanarak kuruluma başlayabilirsiniz.

> ✓ Kurulumun sorunsuz olabilmesi için işletim sisteminizin güncel olmasına dikkat ediniz.

Kuruluma başladığınızda ilk olarak aşağıdaki ekran karşınıza gelmektedir. Bu ekranın alt kısmında bulunan 2 seçeneğinde işaretli olması gerekmektedir. Eğer kurulum ile ilgili detaylı ayarlar yapmak isterseniz "Customize Installation" seçeneğini seçebilirsiniz. Fakat "Install now " seçeneği yeni başlayanlar için sorunsuz bir kurulum sağlayacaktır.



Resim 7.1 Python Kurulumunun İlk Ekran Görüntüleri

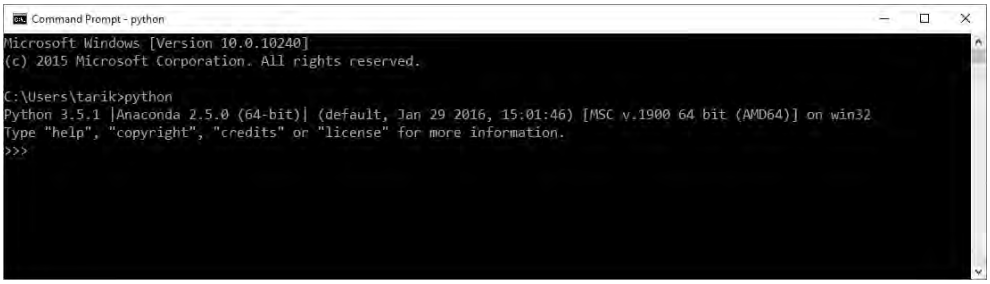
Bu ayarlardan sonra kurulum başlayacak ve sonlanacaktır. Python, eğer "Install now" seçeneği ile kurduysanız, "%LOCALAPPDATA%\Programs\Python" dizini içerisine kurulacaktır.

> internet
> Eğer bilgisayarınıza Python kurmadan denemek istiyorsanız, http://www.pythontutor.com/ adresini ziyaret ediniz.

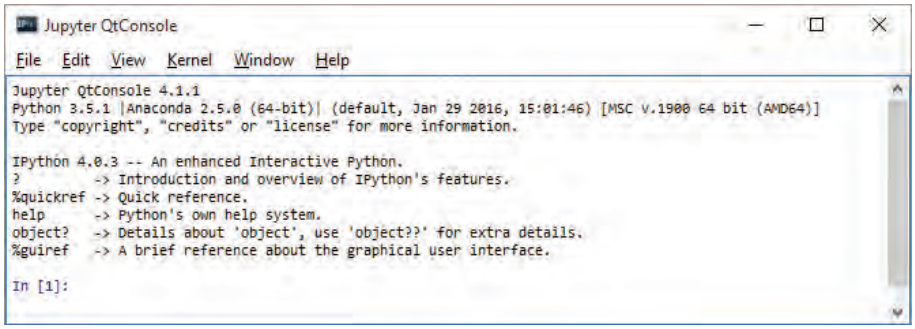
Python ile Çalışma

Kurulum aşamasından sonra python ile çalışma için iki l bulunmaktadır. Bunlardan ilki Başlat düğmesi altında Tüm programlar>Python seçeneği altında Python(Command Line) seçeneğini kullanmaktır. Diğer ise MS-DOS komut istemcisi arayüzünde python komutu vererek çalıştırmaktır. Bu işlemlerin ardından

kodunuzu yazabileceğiniz bir arayüze sahip olabilirsiniz. Bu arayüzde ilk Python kodlarımızı ">>>" işaretinden sonra, hiç boşluk bırakmadan yazmamız gerekmektedir.

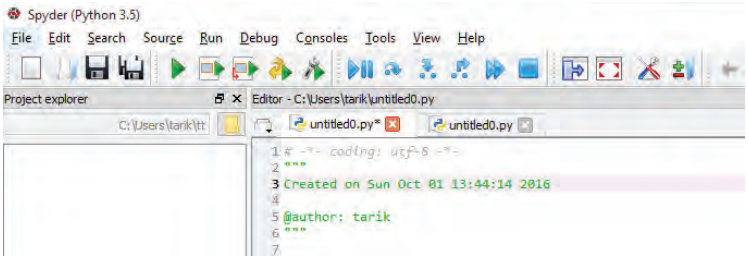


Resim 7.2 Python İçin Etkileşimli Kabuk



Resim 7.3 Python İçin Jupyter QtConsole Etkileşimli Kabuk

Farklı işlevleri destekleyen ve kod yazmayı kolaylaştıran daha gelişmiş ortamlar da bulunmaktadır. Bu bölümde yer alan kodlar ilk etapta python'un etkileşimli kabuğunda daha sonra ise "spyder" isimli bir editör ile yazılmıştır. Oldukça işlevsel bir arayüze (resim 7.4) sahip bu editör kod yazımını oldukça kolaylaştırmaktadır.



Resim 7.4 Spyder Arayüzü İçin Bir Ekran Alıntısı

Bölümün ilerleyen kısımlarında daha farklı editörlerden ve arayüzlerden de bahsedilecektir. Ama önce bu kabuğu kullanarak ilk programımızı yazalım.

internet

"Spyder"ı indirmek ve gerekli belgeler ulaşmak için https://pypi.python.org/pypi/spyder linkini kullanabilirsiniz.

internet

Python editörlerinin listesine https://wiki.python.org/moin/PythonEditors adresinden ulaşabilirsiniz.

Bu noktada bir geleneği gerçekleştirerek Python programlama dili ile "Merhaba Dünya" örneğini yapalım. Python dilinin söz diziliminin basit yapısını karşılaştırabilmek için aynı örneği JAVA ve C++ programlama dili ile altı yedi satırda kodlarken Python ile bunu tek satırda kodlamak mümkündür.

```
print("...Merhaba Dünya...")
```

Python programlama dili yaklaşık 30 adet saklı sözcük barındırmaktadır. Bu sözcükler değişken ve metot ismi olarak kullanılamaz. Saklı sözcükler şunlardır: and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield.

Bir programlama dilinde koda açıklamalar koymak, sağlıklı bir kodlama yapabilmek açısından çok önemlidir. Her programlama dilinde olduğu gibi Python'da da yorum satırları oluşturabilmek için bazı özel karakterler bulunmaktadır. Eğer yapacağımız yorum tek satırdan oluşuyorsa # işaretini kullanabiliriz.

```
# bu bir yorum satırıdır.
```

Python Programlama Dilinde Değişkenler

Programlama dilinde değişkeni, bir veriyi içerisinde depolayan birim olarak tanımlayabiliriz. Değişkeni genel anlamı ile boş bir kutuya benzetebiliriz. Kutu, içerisine koyduğumuz veriyi bizim için saklar ve üzerinde işlem yapmamıza izin verir. Birçok programlama dilinde verinin saklanacağı değişken tipinin önceden tanımlanması gerekmektedir. Fakat Python'da tıpkı PHP de olduğu gibi veri tipleri önceden belirlenmez, değişkene yapılan ilk atama esnasında atanan veri tipine göre belirlenir. Bu programcı için büyük bir özgürlük sağlar. Fakat bu durum değişken değerlerinin izlenmesini zorlaştırır. Aşağıda python dilinde yapılmış birkaç değişken atamasını görebilirsiniz.

```
>>>tamsayı=10
>>>ad="Ali"
>>>ondalik=7.2
```

Değişken isimlerini verirken daha önce de bahsettiğimiz saklı kelimeleri ve kütüphanelerde yer alan fonksiyon isimlerini kullanamayız. Bunun dışında da bazı kurallar bulunmaktadır. Bu kurallardan bazıları programlama ortamları tarafından bize dikte edilirken, bazıları da genel tavsiye niteliğindedir. Değişken isimleri bir harf veya "_" ile başlamalıdır. Python da değişkenler isimleri Türkçe karakter içerebilir, ama şu an itibari ile bundan kaçınmakta yarar vardır. Değişken isimlerinde boşluk kullanılmaz. Değişken adı birden fazla kelimeden oluşuyorsa kelime geçişlerinde büyük harf veya _ kullanılabilir. Örneğin degisken_adi veya degiskenAdi gibi. Değişken isimlerini seçerken anlaşılabilir yorumlanabilir isimler seçmekte yarar bulunmaktadır. Değişken ataması "=" işareti ile yapılmaktadır. Aynı değere sahip birden fazla değişken aynı satırda tanımlanabilir. Python' da programlamada çok kullanılan değişken takas işlemi için kısa yapılar tanımlanmıştır.

```
>>>a=5
>>>b=8
>>>a, b=b, a
```

Görüldüğü gibi bu işlev bizi daha fazla satır yazmaktan kurtarmaktadır. Python dilinde değişkenlerini birbirine dönüştürmek tip dönüştürücü fonksiyonları kullanarak mümkündür. Bu fonksiyonlara hazır fonksiyonlar anlatılırken değinilecektir.

Python Veri Tipleri

Python programlama dilinde değişken tanımlamada tip belirtmeye gerek yoktur. Python ilk atama yapıldığı anda değişkenin tipini otomatik olarak belirlemektedir. Python ile kullanacağımız veri tipleri, integer, float, long integer, boolean, complex, string, list, dictionary ve tuples olarak sıralanabilir. Ayrıca python'da fonksiyonlar, lambdalar, nesneler, dosya tanıtıcıları gibi topluluklar da veri tipi olarak kabul edilirler.

Integer: Tamsayıları saklamak için kullanılmaktadır. 32 bit genişliğindedir. ± 2147483647 sayı aralığında değer alabilirler.

Long Integer: Daha büyük tam sayılar için kullanılmaktadır. 64 bit genişliğindedir.

Float: Ondalık sayıları saklamak için kullanılmaktadır. Örneğin, 3.54 veya 7.8e-28 gibi. Python ile birçok dilin aksine çok büyük ondalıklı değerler tanıtılabilmekte ve üzerlerinde işlem yapılabilmektedir. Bu nedenle özellikle büyük sayılarla çalışılan matematik problemlerinde avantaj sağlamaktadır.

Complex sayılar: Matematikte sıkça kullanılan karmaşık sayıların tanımlanmasında kullanılmaktadır. Gerçek ve sanal sayılardan oluşur. Sanal sayılar j veya J soneki ile belirtilir. 5+4j veya 1.35+0.003j gibi.

Boolean: Doğru ve yanlış mantıksal değerleri saklamak için kullanılır. İki boolean değer vardır: 'True' ve 'False'.

String: Karakter katarlarını tanımlamak için kullanılır. " veya ' işaretleri arasında tanımlanır.

```
>>> metin1="Python öğreniyorum"
>>> metin2='Python çok kolay'
```

Eğer metin içerisinde " karakteri kullanılmak istenirse kaçış karakteri (\) ile birlikte kullanılmalıdır.

```
>>> metin1='İzmir\'de doğdum'
```

String tipindeki iki değişkeni yan yana birleştirme işlemi (Concatenate) toplama operatörü ile gerçekleştirilmektedir.

```
>>>metin1='Python'
>>>metin2='Programlama Dili'
>>>metin3=metin2+" "+metin3
```

Yukarıda verilen örnekte metin3 değişkeninin içerisinde "Python Programlama Dili" yer alacaktır. "+" operatörü dışında "*" (çarpma) operatörünün de string değişkenler açısından farklı bir anlamı vardır.

```
>>>metin="yavaş "
>>>print (metin * 2)
yavaş yavaş
```

Örnekte de görüldüğü üzere "*" operatörü ile metin tekrar sayısını belirleyebiliriz.

List: Ardışık verilerin tutulduğu listeler olarak tanımlanabilir. Esasen diğer dillerdeki dizi değişkenlerin python'daki karşılığıdır. Liste elemanları köşeli parantez kullanılarak aralarında virgül ile tanımlanır. Liste elemanına ulaşmak için indis (anahtar) kullanılmaktadır.

```
>>>ogrenciler=["ali", "ayşe", "veli", "fatma"]
>>>print(ogrenciler)
['ali', 'ayşe', 'veli', 'fatma']
>>>print(ogrenciler[0])
ali
>>>print(ogrenciler[2])
veli
>>>print(ogrenciler[-1])
fatma
>>>print(ogrenciler[1:3])
['ayşe', 'veli']
```

Örnekte görüldüğü üzere indis 0'dan (toplam eleman sayısı -1) e kadar değer almaktadır. İndis değeri negatif değer alabilmektedir. Bu listenin sonundan başlayarak indisin belirlenmesi anlamında gelir. Yani, -1 indisi listenin son elemanını göstermektedir. Ayrıca liste veri tipinde aralık vererek de listenin bir parçasına erişim mümkün olmaktadır. Listede yer alan elemanların türlerinin aynı olmasına da gerek bulunmamaktadır.

```
>>>ogrenci=["ali", "taş", 15, 70.5, True]
```

Örneğin yukarıdaki gibi tanımlana listede liste elemanları farklı tiplerde tanımlanmıştır. Listenin sonuna eleman eklemek için append(), istediğimiz bir noktasına eleman eklemek için insert() metodu, iki listeyi birleştirmek için ise extend() metodu kullanılmaktadır. Listeden belirtilen bir eleman silmek için ise remove() ve indisi verilen bir elemanı silmek için ise pop() metodu kullanılabilir.

```
>>>liste=["ali", "ayşe", "veli", "ahmet"]
>>>liste.append("mehmet");
>>>print (liste)
['ali', 'ayşe', 'veli', 'ahmet', 'mehmet']
>>>liste.insert(1, "mine")
>>>print(liste)
['ali', 'mine', 'ayşe', 'veli', 'ahmet', 'mehmet']
>>>liste.remove("mehmet")
>>>print(liste)
['ali', 'mine', 'ayşe', 'veli', 'ahmet']
>>>liste.pop(1)
['ali', 'ayşe', 'veli', 'ahmet']
```

Bunlar dışında;

- ***.index():*** Bir öğenin indisini öğrenmek için,
- ***.sort():*** Listemizdeki öğeleri alfabe sıralamak için,
- ***.reverse():*** sırayı ters çevirmek için
- ***.count():*** listedeki eleman sayısını öğrenmek için

gibi fonksiyonlar kullanılabilir.

Tuples: Tüpler içeriği değiştirilmeyen listeler olarak tanımlanabilir. Elemanları (ve) karakterleri içerisinde virgülle ayrılarak tanımlanır.

```
>>>ProgDilleri=("Python", "c", "Java", "PHP", "Pascal")
```

Dictionary: Bu veri tipide listeye benzemektedir. Fakat dictionary (sözlük) veri tipinde elemanlar (anahtar, değer) ikilileri şeklinde tanımlanırlar. Anahtar ve değer için istenilen veri tipi kullanılabilir. Değere ulaşmak için daha önce belirtilen anahtar kullanılmaktadır. Elemanları { ve } karakterleri arasında anahtar:değer sözdizimi ile virgül ile ayrılarak tanımlanır.

```
>>>ogrenciler = {155: "ali", 223: "ayşe", 255:"veli", 321: "fatma"}
>>>print (ogrenciler[155])
ali
```

Sözlüğe veri eklemek için sadece yeni anahtarı kullanarak değer atamamız yeterli olacaktır.

```
>>>ogrenciler = {155: "ali", 223: "ayşe", 255:"veli", 321: "fatma"}
>>>ogrenciler[200]="melih"
>>>print (ogrenciler)
{200: 'melih', 321: 'fatma', 155: 'ali', 255: 'veli', 223: 'ayşe'}
```

Sözlükten veri silmek istediğimizde ise "del" fonksiyonunu kullanabiliriz.

```
>>>ogrenciler = {155: "ali", 223: "ayşe", 255:"veli", 321: "fatma"}
>>>del ogrenciler[155]
>>>print (ogrenciler)
{321: 'fatma', 255: 'veli', 223: 'ayşe'}
```

Sözlük veri tipinde de kullanabileceğimiz başka metotlar bulunmaktadır:

- ***.clear():*** Sözlüğün tüm elemanlarını siler. Kullanımı: liste.clear()
- ***.keys():*** Sözlüğün sadece anahtarlarını almamızı sağlar. Kullanımı: liste.keys()
- ***.values():*** Sözlüğün sadece değerlerini almamızı sağlar. Kullanımı: liste.values()
- ***.items():*** Sözlüğün anahtarlarını ve değerlerini almamızı sağlar. Özellikle for döngülerinde oldukça işimize yarayacak bir metottur. Kullanımı: liste.items()
- ***.get():*** Eğer yazılan anahtar varsa onun değerini getirirken, anahtar arrayde bulunmaz ise ikinci parametreyi döndürmektedir. Kullanımı: ogrenciler.get(155, "aradığınız öğrenci yok.")
- ***.copy():*** Sözlüğü kopyalamak için kullanılır. Kullanımı: yedek_ogrenciler = ogrenciler.copy()
- ***.pop():*** listelerde olduğu gibi verilen anahtarı listeden silme işlevine sahiptir. Kullanımı ogrenciler.pop(155)
- ***.setdefault():*** Yeni anahtar ve değer ikilisi eklemek için kullanılır. Eklenmek istenen ikili sözlükte mevcut ise ekleme işlemini gerçekleştirmez. Daha önce gösterdiğimiz sözlüğe ekleme işleminde ise anahtara ait değer son gelen değer ile değiştirilir. Kullanımı: ogrenciler.setdefault(175, "onur")

None Nesnesi: Var olmayan değerleri ifade etmekte kullanılır. Diğer dillerde yer alan "null ve nil" ifadelerinin pythondaki karşılığıdır.

Python Programlama Dilinde Operatörler

Operatörler sağında ve solunda bulunan değer/değerler üzerinde bir işlem geçekleştiren veya bir ilişki kuran işaretler olarak tanımlanabilir. Birçok kaynakta operatörler için işleç, işlemci gibi isimlerde kullanılmaktadır. Programlama dilinde operatörler farklı kategoriler altında sınıflandırılmışlardır: atama, aritmetik ve karşılaştırma operatörleri.

Atama Operatörleri: Python'da diğer diller dillerde olduğu gibi atama operatörü "=" dir. Bu operatör dışında birleşik atama operatörleri de bulunmaktadır: +=, -=, /=, *=, %=, **=, //=

Aritmetik Operatörler: Genel olarak matematiksel işlemleri gerçekleştirmemizi sağlamaktadırlar. Bu operatörler: + (toplama), - (çıkarma), * (çarpma), / (bölme), ** (kuvvet), % (modülüs), // (taban bölme) şeklinde sıralanabilir. Daha öncede açıklandığı üzere "+" ve "*" string veri tipleri için farklı bir anlamı daha bulunmaktadır. Aşağıdaki örnekte bölme işlemi ile taban bölme işlemi gösterilmektedir.

```
>>>print(9/2)
4.5
>>> print (9//2)
4
```

Karşılaştırma Operatörleri: iki veri arasında karşılaştırma ilişkisi kuran operatörler, karşılaştırma operatörleri olarak adlandırılmaktadır. Bunlar; == (eşitlik), != (eşit değil), > (büyüktür), < (küçüktür), >= (büyük eşit), <= (küçük eşit) olarak sıralanabilir.

Python'da kullanılan bir başka karşılaştırma operatörü de "is" kelimesidir. Bu kelime genellikle == ile karıştırılmaktadır. "is" operatörü nesneler arasında karşılaştırma yapılırken kullanılmaktadır. İlerleyen bölümlerde bu operatöre daha detaylı değineceğiz.

Son olarak ise in operatörüne değinelim. "in" operatörü bir sayının veya karakter katarının, herhangi bir veri tipi içinde olup olmadığını sorgulamaktadır.

```
>>>metin = "pyhton"
>>>"p" in metin
True
>>>"a" in metin
False
>>>liste=[3,5,7,13,56]
>>>print (3 in liste)
True
```

Python Programlama Dilinde Hazır Fonksiyonlar

Python dilinde işlerimizi kolaylaştıracak birçok hazır fonksiyon ve metot bulunmaktadır. Bunlardan sıkça kullanacağımızı düşündüklerimiz bazılarını şu şekilde listeleyebiliriz.

print() Fonksiyonu

Tüm dillerde ilk öğrenilen fonksiyonlardan olan print fonksiyonu, istediğimiz verinin ekrana basılmasını sağlamaktadır. Aslında print fonksiyonunu şu ana kadar birçok örneğimizde kullandık. Bu bölümde ise biraz daha detaya ineceğiz. Print fonksiyonu ile yazdıracağımız string katarları tek tırnak (‘), çift tırnak (“) veya üç tane çift tırnak kullanılarak yazdırılabilir. Üç çift tırnak gösterimi genelde ekrana basmak istediğimiz karakter katarı içerisinde “ karakteri bulunduğu zaman ve yazdırılacak veri birden fazla satırdan oluşuyorsa tercih edilir.

```
>>> print('Python programlama dili')
Python programlama dili
>>> print("Python programlama dili")
Python programlama dili
>>> print("""Python "programlama" dilidir""")
Python "programlama" dili
```

Print fonksiyonu birden fazla parametre alabilmektedir. Örneğin aşağıdaki kullanımda girilen tüm parametreler aralarında boşluk ile ekrana basılır. İstersek “sep” değerini kullanarak boşluk yerine istediğimiz karakterin parametreler arasına yazılmasını sağlayabiliriz.

```
>>> print('Python', 'programlama', 'dili')
Python programlama dili
>>> print('www', 'anadolu', 'edu', 'tr', sep='.')
www.anadolu.edu.tr
```

Print fonksiyonu içerisinde bir alt satıra geçmek amacı ile \n, bir tab (sekme) boşluk bırakmak için \t, aynı satırda satır başı yapmak için \r karakterleri kullanılabilir. Print fonksiyonunda kullanılabilecek yararlı bir işlev ise * karakteri ile sağlanmaktadır. Aşağıdaki örnekte görüldüğü üzere * parametresine sahip string tipindeki paramatre harf harf yazılmış aralarına varsayılan olarak boşluk karakteri koyulmuştur. Varsayılan karakteri değiştirmek için ise yine sep ifadesini kullanabiliriz.

```
>>> print(*'Python')
P y t h o n
>>> print(*'python', sep='.')
p.y.t.h.o.n
```

✓ print fonksiyonunun formatlı yazdırma seçenekleri de bulunmaktadır.

input() Fonksiyonu

Konsol ekranında kullanıcıdan veri almak için kullandığımız fonksiyondur. Bu fonksiyon sayesinde kullanıcıdan gelen veriyi istediğimiz bir değişkene atabiliriz.

```
>>> isim=input('İsminizi giriniz')
>>> print('Merhaba', isim)
```

Burada dikkat etmemiz gereken nokta, kullanıcıdan alınan bilginin (kullanıcı sayısal bir bilgi girse bile) string tipinde olduğudur. Eğer kullanıcıdan gelen bilgi ile sayısal işlem yapacaksak muhakkak veriyi dönüştürmemiz gereklidir. Dönüştürücü fonksiyonlar bir sonraki başlık altında açıklanmıştır. Aşağıdaki örneği yapmadan muhakkak dönüştürücü fonksiyonları inceleyiniz.

Kullanıcıdan aldığı çap bilgisi ile dairenin çevresini ve alanını hesaplayan bir program yazalım.

```
>>>cap=input('Dairenin çapını giriniz')
>>>yaricap=int(cap)/2
>>>print('Dairenin çevresi : ', 2*yaricap*3.14)
>>>print('Dairenin alanı : ', yaricap *3.14*3.14)
```

Dönüştürücü Fonksiyonlar

Verinin tipini değiştiren fonksiyonlar olarak tanımlanmaktadır. Özellikle kullanıcıdan aldığımız verilerle matematiksel işlemler gerçekleştireceksek muhakkak dönüştürülmesi gerekmektedir. Aşağıdaki örnekte eğer dönüşüm yapılmaz ise nasıl bir sonuç ile karşılaşacağımız örneklendirilmiştir.

```
>>>s1=input('ilk sayıyı giriniz')
>>>s2=input('ikinci sayıyı giriniz')
>>>print(s1,"+", s2," = ", s1+s2)
78+4 = 784
```

Örnekte kullanıcı 78 ve 4 sayılarını girmiştir. Sonuç 82 olması gerekirken 784 olarak yazılmıştır. Bunun nedeni tip dönüşümünün yapılmamasıdır. Doğru kodlama şu şekilde olmalıdır.

```
>>>s1=input('ilk sayıyı giriniz')
>>>s2=input('ikinci sayıyı giriniz')
>>>print(s1,"+",s2,"=",int(s1)+int(s2))
78+4 = 82
```

Python dilinde var olan dönüştürücü fonksiyonların bazıları şu şekilde sıralanabilir:

- *int(x):* veriyi tam sayıya dönüştürür.
- *long(x):* veriyi uzun tam sayı ifadeye dönüştürür.
- *float(x):* veriyi ondalıklı sayıya dönüştürür.
- *complex(x):* veriyi karmaşık sayıya dönüştür.
- *str(x):* veriyi string ifadeye dönüştürür.
- *tuple(s):* veriyi/leri tuple veri tipine dönüştürür.
- *list(s):* veriyi list veri tipine dönüştürür.
- *set(s):* String ifadeyi set eder.
- *dict(d):* veriyi sözlük oluşturur.
- *chr(x):* bir tamsayıyı karaktere dönüştürür.
- *unichr(x):* bir tamsayıyı unicode olarak karaktere dönüştürür.
- *ord(x):* bir karakterin ASCII kodunu verir.
- *hex(x):* tam sayıyı hexadecmal bir sayıya dönüştürür.
- *oct(x):* tam sayıyı octale dönüştürür.

abs() Fonksiyonu parametre olarak girilen sayının mutlak değerini döndürmektedir.

len() Fonksiyonu

Parametre olarak girilen karakter kadarının eleman sayısını veya girilen dizi tipi verilerin (tuple, list, dictionary vb.) eleman sayısını döndürmektedir.

```
>>>metin="python"
>>>print("metnin uzunluğu=",len(metin))
metnin uzunluğu= 6
>>>liste=["ali", "ayşe", "veli", "ahmet"]
>>>print("listenin uzunluğu=",len(liste))
listenin uzunluğu= 5
```

pow() Fonksiyonu

Bu fonksiyona parametre olarak 2 veya 3 veri girilebilmektedir. Eğer iki parametre girilirse pow(x,y)=xy, eğer 3 parametre girilirse pow(x,y,z)= xy%z şeklinde değer döndürmektedir.

round() Fonksiyonu

Bu fonksiyon parametre olarak 1 veya 2 veri almaktadır. Eğer tek parametre ile çalışırsa girilen sayıyı en yakın tam sayıya yuvarlamaktadır. İkinci parametre (eğer girilirse) virgülden sonra yuvarlanacak basamak sayısını göstermektedir.

```
>>>print(round(1.4))
1
>>>print(round(1.5))
2
>>>print(round(1.657, 2))
1.66
>>>print(round(1.657123, 4))
1.6571
```

Python'da yer alan tüm hazır fonksiyonların listesine https://docs.python.org/3.6/library/functions.html ve https://docs.python.org/3.6/library/stdtypes.html adreslerinden ulaşabilirsiniz. Ayrıca merak ettiğiniz fonksiyonları bu sayfada bulunan arama kutucuğunu kullanarak inceleyebilirsiniz.

Öğrenme Çıktısı

1 Python dilini kurabilme

Araştır 1

Bir öğrenci listesinin elemanlarını tek tek ekleyip, sonra bu listeyi büyükten küçüğe sıralayıp ekrana yazdırınız. Ayrıca İngilizce kelimelerin anahtar, Türkçe karşılıklarının değer olduğu bir sözlük yapısı oluştu¬rup içerisine 10 adet veri giriniz.

İlişkilendir

Python kurulumu süreçlerini diğer internet tabanlı programlama dillerinin kurulum süreçleriyle ilişkilendirerek karşılaştırınız.

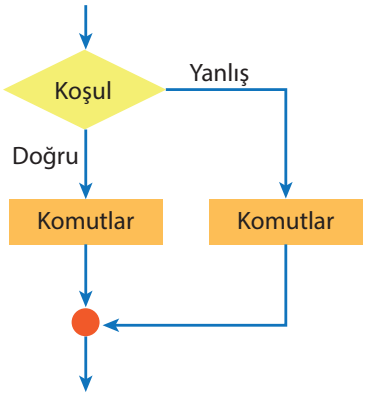
Anlat/Paylaş

Python kurulumundan sonra Python'da denemeleri yapınız. Python(Command Line) ve MS-DOS komut istemcisi arayüzünde Python ile çalışma deneyimlerinizi paylaşınız.

PYTHON PROGRAM DENETİMİ

Programlama dillerinde program denetimi, şartlara bağlı olarak program akışını düzenlememize imkân veren, programın oluşturacağı sonuçları değiştirmemizi sağlayan yapılar olarak tanımlanabilir. Akış denetimi ve döngüler iyi bir programcının derinlemesine bilgi sahibi olması gereken yapılardır. Program akışının denetlenmesi ve kontrolü kod yazımının en önemli parçasıdır. Bu başlık altında tüm dillerde var olan kontrol ve döngü yapılarının python'daki sözdizimlerini inceleyeceğiz.

Kontrol Yapıları: if, else, elif İfadeleri: Kod içerisinde, koşul kullanarak yön değiştirme için tasarlanmış en basit yapıdır. Aşağıdaki şekilde if else yapısının akış şeması yer almaktadır.



Resim 7.5 If else Yapısının Akış Şeması.

Python dilinde kod bloklarını belirlemede begin/end veya süslü parantez {, } gibi ifadeler yerine girintileme kullanılmaktadır.

Python'da koşulları belirtmek için if, else, elif isimli 3 deyim kullanılmaktadır. Şimdi bu 3 deyimi sırasıyla örneklendirerek açıklayalım.

if Deyimi: if deyiminin genel söz dizimi aşağıdaki şekildedir.

```
if KOŞUL:
    komutlar
```

Kullanıcının girdiği sayının 100' den büyük olmasını denetleyen programı şu şekilde yazabiliriz.

dikkat
Bu noktadan sonra verilen kod örnekleri python etkileşimli kabuğu yerine spyder ortamında yazılmıştır.

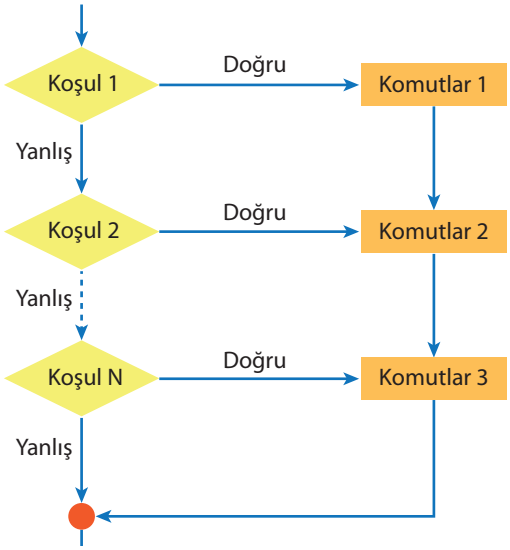
```
girdi=input("bir tamsayı giriniz")
n=int(girdi)
if n>100:
   print("girilen sayı 100'den büyüktür")
```

else Deyimi: else deyimi if ile deyimi ile birlikte kullanılmaktadır. if ile belirtilen koşul gerçekleşmediğinde else deyiminde yer alan komutlar işletilmektedir. "else" deyiminin genel yapısı aşağıdaki şekildedir.

```
if KOŞUL:
    komutlar
else:
    komutlar
```

elif Deyimi: Bu deyim, if deyimi ile birlikte kullanılmaktadır. Diğer dillerden aşna olduğumuz "elseif" yapısının python'daki şeklidir. Akış şeması ve sözdizimi genel yapısı aşağıda verilmiştir.

```
if KOŞUL 1:
    komutlar 1
elif KOŞUL 2:
    komutlar 2
...
elif KOŞUL 3:
    komutlar 3
```



Resim 7.6 If elif Yapısının Akış Şeması

Bu noktada yukarıdaki örneği biraz geliştirelim. Kullanıcının girdiği sayının 100'den büyük, küçük veya eşit olma durumunu gösteren bir örnek yapalım.

```
girdi=input("bir tamsayı giriniz")
n=int(girdi)
if n>100:
    print("girilen sayı 100'den büyüktür")
elif n<100:
    print("girilen sayı 100'den küçüktür")
else:
    print("girilen sayı 100'e eşittir")
```

Döngü Yapıları: for ifadesi

Herhangi bir programlama dilinde döngü deyimleri deyince akla ilk gelen for ifadesi olacaktır. Tüm programların kodlarının içinde her zaman bir for döngüsüne rastlamak mümkündür. For döngüsü belli bir kod bloğunun belirtilen aralıkta tekrarlanmasını sağlayan ifadelerdir. Python'daki, for döngüsü, PHP, JavaScript, PERL gibi dillerdeki "foreach" döngülerinin yaptığı işlevide yüklenirler. Bir başka deyişle string, listeler (list), tüpler (tuples), sözlükler (dictionary) vb. veriler ile de kullanılabilir. *for* ifadesi ile ilgili örneklere geçmeden önce *range()* fonksiyonu öğrenmekte yarar bulunmaktadır.

range() Fonksiyonu

Range([başlangıç değeri],sonuç değeri,[artım değeri]) şeklinde gösterilmektedir. Yani toplamda 3 parametre almaktadır. İlk parametre aralığın başlangıç değerini, ikinci parametre sonlanma değerini, üçüncü parametre ise artış değerini belirtmektedir. Fonksiyon eğer sadece tek parametre kullanılarak çağrılısa başlangıç değeri 0, artış değeri ise 1 olarak kullanılır. Şimdi bu fonksiyonun for ifadesi içerisinde kullanımını inceleyelim.

```
for i in range(5):
    print(" i = ", i ," | ")
```

Yukarıda verilen örneğin çıktısı

```
i = 0 | i = 1 | i = 2 | i = 3 | i = 4 |
```

Çıktıdan da görüldüğü üzere tek parametre ile çalışan range fonksiyonu başlangıç değeri olarak 0, artış değeri olarakta 1 almıştır. Aynı örneği 2 parametre vererek gerçekleştirelim.

```
for i in range(2,5):
    print(" i = ", i ," | ")
i = 2 | i = 3 | i = 4 |
```

Görüldüğü gibi döngü 2 den 4' e kadar, birer artış ile gerçekleşmiştir. Şimdi ise artış miktarını belirleyerek yapalım.

```
for i in range(0,5,2):
    print(" i = ", i ," | ")
i = 0 | i = 2 | i = 4 |
```

Görüldüğü üzere döngü 0'dan başlamış 5'ten küçük olmak şartı ile ikişer artarak devam etmiştir. "range" fonksiyonundaki artış değeri negatifte olabilir.

```
for i in range(5,2, -1):
    print(" i = ", i ," | ")
i = 5 | i = 4 | i = 3 |
```

Bu şekilde geriye doğru sayan döngüler yapabiliriz.

Şimdi de for ifadesinin listelerde nasıl kullanıldığını bir örnek ile açıklayalım.

```
liste=["ali", "ayşe", "veli", "ahmet"]
for x in liste:
    print(x ," | ")
ali | ayşe | veli | ahmet |
```

Aynı örneği bir de sözlük veri yapısı için yapalım.

```
ogrenciler = {155: "ali", 223: "ayşe", 255:"veli", 321: "fatma"
for i in ogrenciler:
    print(i ," => ", ogrenciler[i], "\n")
321 => fatmma
155 => ali
255 => veli
223 => ayşe
```

Koşullu Döngü Yapıları: while ifadesi

while ifadesi neredeyse tüm dillerde olduğu gibi python'da da koşullu döngüler için kullanılmaktadır. while döngüsü belirtilen koşul gerçekleştiği (True) müddetçe içinde belirtilen kod bloğunu tekrar etmektedir. while döngüsünün genel yapısı aşağıda gösterilmiştir.

```
while i<10:
  KOMUTLAR
```

while döngüsünü anlamak için 25'den 200'e kadar olan sayılar içerisinde 6'ya tam bölünenleri listeleyen bir program yazalım.

```
a=25
while (a<=200):
    if a%6==0:
        print(a, "|")
    a+=1
```

dikkat

while döngüsündeki koşulu belirlerken çok dikkatli olmamız gerekir. Burada yapacağımız hata programımızın sonsuz döngülere girmesine neden olabilir. Böyle bir durumda programı durdurmak için ctrl+c veya ctrl+z tuş kombinasyonları kullanılır.

Döngü Çıkış Bildirimleri: pass/break/continue

pass: Bu bildirim kullanıldığı noktadan itibaren içinde yer aldığı bloğu es geçmemizi sağlar. Program çalışmaya bloğun sona erdiği satırdan devam eder. Aşağıdaki örnekte kullanılan pass deyimi programı, print("devam") satırına göndermiştir. Yani while içerisindeki hiçbir ifade çalışmamıştır.

```
a=25
while (a<=200):
    pass
    if a%6==0:
        print(a, "|")
    a+=1
print("devam")
```

continue: Bu bildirim, bir döngünün gövdesi içerisinde çalışmaktadır. Bildirim çağrıldığında, o noktadan sonraki komutlar atlanır, döngü bir sonraki iterasyondan devam eder.

```
for i in range(1,10):
    if i==5:
        continue
    print (i)
```

Bu örnekte çalışan program 5 hariç 1'den 9'a kadar rakamları ekrana basacaktır. Bunun nedeni, i'nin 5 değeri aldığında if bloğundaki koşulun gerçekleşmesi, continue bildirimin çalışması ve bu noktadan sonraki kodları es geçerek iterasyona bir sonraki i değeri ile devam etmesidir.

break: Bu bildirim ile program çağrıldığı döngüden çıkarak, bir sonraki program adımından devam eder. Aşağıdaki örnekte ekrana sadece 1, 2, 3, 4 değerleri basılacaktır. Çünkü i değeri 5 olduğunda break ifadesi çalışacak ve döngüyü kıracaktır.

```
for i in range(1,10):
    if i==5:
        break
    print (i)
```

İçiçe döngüler ile çalışıldığında, break bildirimi sadece içerisinde çalıştığı döngüden çıkar. Eğer dış döngü varsa bu döngü devam eder.

Python Programlama Dilinde Fonksiyonlar

Fonksiyon, bir amaç doğrultusunda bir araya gelmiş kodlar olarak tanımlanabilir. Aslında şu ana kadar kodlarımızda print(), len(), range() vb. birçok hazır fonksiyon kullandık. Bu bölümde ise kendi fonksiyonlarımızı nasıl yazacağımızı öğreneceğiz. İsterseniz ilk önce fonksiyonların özelliklerini maddeler halinde inceleyelim:

- Fonksiyonlar programlamada kullanacağımız ve sürekli tekrar eden komutların kullanımını daha da kolaylaştırmak için oluşturacağımız yapılardır.
- Fonksiyonlar aynı zamanda kodumuzun okunabilirliğini ve anlaşılabilirliğini yükseltmektedir.
- Fonksiyonlar bir isme sahiptir.
- Fonksiyonlar gerçekleştireceği işe göre dışarıdan parametre/ler alabilirler veya almayabilirler. Bu parametreler "(", ")" karakterleri arasında virgül ile ayrılarak iletilirler. Eğer fonksiyonun hiç parametresi yoksa () ifadesi fonksiyon adından sonra kullanılır.
- Parametre listesinin tanımlandığı şekilde sıralı verilmesi gerekmektedir. Aksi takdirde fonksiyon istenilen şekilde çalışmayacaktır. Python'da parametreler isimleri ile de çağrılabilirler. Bu durumlarda sıranın bir önemi olmaz.
- Fonksiyonları çoğu zaman bir şablon yapı olarak düşünebiliriz. Bu şablon yapı girilen parametrelere göre şekillendirilebilir.
- Fonksiyonlar gerçekleştridiği işlem sonrasında bir veri geri döndürebilir veya döndürmez.

Fonksiyonların Tanımlanması ve Çağırılması

Python' da fonksiyon kullanımı iki aşamadan oluşmaktadır: tanımlama ve çağırma. Fonksiyon tanımlamanın genel şekli aşağıda gösterilmiştir.

```
def fonksiyonAdi(parametre/ler):
    Komut Listesi
```

Görüldüğü üzere fonksiyon tanımlanan anahtar kelimesi "def"'dir. Bu deyimden sonra fonksiyonun adı verilir. Fonksiyon adında değişken adı belirleme kuralları geçerlidir. Fonksiyon adı, fonksiyon çağırılırken kullanılacaktır. Fonksiyonun adının belirlenmesinin ardından ise parametre listesi parantez içerisinde virgül ile ayrılarak tanımlanır. Daha öncede belirtildiği üzere fonksiyonlar 0 veya n adet parametre alabilirler. Fonksiyon tanım satırı ":" karakteri ile bitirildikten sonra, bir alt satırdan itibaren fonksiyonun işlevini tanımlayacak kod listesi yazılır.

dikkat

Tıpkı ana program kodları yazarken olduğu gibi, fonksiyon tanımı yaparken de hizalama/girintilime çok önemlidir.

Python'da bir fonksiyonun çağırılabilmesi için öncelikle tanımlanması şarttır. Tanımlanan bir fonksiyon, kodun istenen bir bölümünde fonksiyonun adı ve parametre listesi ile çağrılabilir. İsterseniz şimdi birkaç fonksiyon örneği yazalım. İlk örneğimiz hiç bir parametre almayan bir fonksiyon olsun.

```
def merhaba():
    print ("merhaba dünya")
```

Tanımlamasını yaptığımız fonksiyonu çağırmak için ise kodun istediğimiz bir yerinde,

```
merhaba()
```

yazmamız yeterlidir. İkinci örneğimiz tek parametre alsın.

```
def merhaba(prmtr1):
    print ("merhaba", prmtr1)
```

✓ Fonksiyon tanımlarken kullanabileceğimiz maksimum parametre sayısı 256'dır.

Bu fonksiyonu çağırmak için ise fonksiyonun muhakkak bir parametre ile çağrılması gerekmektedir. Eğer fonksiyon tanımlanan parametre adeti ile çağrılmaz ise "missing 1 required positional argument" hatası ile karşılaşırız.

```
merhaba("PYTHON")
```

Bu fonksiyon çağrıldığında ekrana "merhaba PYTHON" yazacaktır. Şimdi yazacağımız fonksiyon ise bize bir değer geri döndürsün. Bu örnek için üs alma işleminin fonksiyonunu yazalım.

```
def üsAl(x, y):
    sonuc=x**y
  return sonuc
```

```
def üsAl(x, y):
    return x**y
```

Yukarıdaki fonksiyon tanımları incelendiğinde her iki kodunda aynı işlemi gerçekleştirdiği fakat sağdaki kodun bunu daha az satırda ve daha az değişken ile gerçekleştirdiği görülmektedir. Bu nedenle programcılık açısından sağ taraftaki kod daha iyi bir kod örneği olarak gösterilebilir. Fonksiyon tanımını incelediğimizde, fonksiyonun verilen iki parametreyi (x ve y) kullanarak bir işlem gerçekleştirdiği ve elde edilen sonucu «return» anahtar kelimesi ile geri döndürdüğü görülmektedir. Geri dönen değer fonksiyonun çağırıldığı noktaya iade edilmektedir. Geri dönen değer ile ne yapılacağı yazılan koda bağlıdır. Geri dönen değer istenirse bir değişken içerisine atılabilir veya bir başka fonksiyona parametre olarak sunulabilir.

```
a=2
b=5
print(üsAl(a,b))
```

Örnekte "üsAl" fonksiyonunun a (=2) ve b(=5) parametrelerini alarak ürettiği değer (32) başka bir fonksiyona (print) parametre olarak aktarılmıştır. Sonuçta program ekrana 32 değerini basar.

Parametre olarak verilen bir kelime içerisinden yine parametre olarak verilen bir karakteri silen fonksiyonu yazalım.

```
def sil(kelime, silinecek):
return kelime.replace(silinecek, "")
k=input("kelimeyi giriniz")
s=input("silinecek ifade")
k=sil(k,s)
print(k)
```

Python'da fonksiyon çağırılırken parametreler ismi ile birlikte sunulabilir, bu durumda parametre sırası önemli değildir.

```
a=2
b=5
print(üsAl(y=b,x=a))
```

Yukarıdaki kodda görüldüğü üzere x ve y parametreleri fonksiyon tanımında belirtilen sıradan farklı bir şekilde sunulmuştur. Ama parametreler isimli olarak verildiğinden dolayı fonksiyon bir önceki örnekte olduğu gibi çalışır ve aynı sonucu üretir.

Python programlama dilinde fonksiyon tanımlarken istediğimiz parametrelere varsayılan değer atayabiliriz. Bu sayede fonksiyon çağrımında varsayılanı tanımlanmış parametrelere değer girmezse işlemlerimiz varsayılan değer ile yürütülür. Önceki örnekte yazdığımız fonksiyonu aşağıdaki şekilde değiştirelim.

```
def üsAl(x, y=2):
    sonuc=x**y
    return sonuc
```

Bu şekilde tanımladığımız fonksiyonu "üsAl(2, 5)" şeklinde çağırırsak bize 32 değerini verecektir. İstersek fonksiyonu "üsAl(2, y=5)" şeklinde de çağırabiliriz. Bu iki çağrım tamamen aynı şekilde çalışacaktır. Tanımlamada y parametresi için verdiğimiz varsayılan değer sayesinde bu fonksiyonu "üsAl(2)" şeklinde de çağırabiliriz. Bu durumda fonksiyon y değeri yerine varsayılan olarak atanan 2 değerini kullanacak ve sonuç olarak 4 değerini üretecektir.

Şu ana kadar yaptığımız örneklerde parametre sayısını hep fonksiyon tanımlanırken belirledik. Peki, kullanacağımız parametre sayısı belli değilse ne yapacağız? Böyle bir sorun karşısında python bize belirsiz sayıda isimsiz parametre tanımlama seçeneği sunmaktadır. Bunun için parametre isminin sol tarafına koyacağımız * karakterinden faydalanıyoruz. Aşağıdaki örnekte girilen rastgele sayıdaki integer değerlerin toplamını veren bir kod örneği verilmiştir.

```
def topla(*sayilar):
    toplam=0
    for x in sayilar:
        toplam+=x
    return toplam
```

Fonksiyon tanımlarken sayılar isimli parametrenin solundaki * karakteri sayılar parametresinin birden fazla değer alabileceğini belirtmektedir. Tanımladığımız bu fonksiyon farklı şekillerde çağrılabilir durumdadır.

```
print (topla (1, 2))
print (topla (1, 5, 8))
print (topla (12, 34, 32, 69, 98, 433))
```

Görüldüğü gibi topla fonksiyonu 3 defa farklı sayıda parametreler ile çağrılmıştır.

Python Modüller ve Paketler

Bölümün başında python'un çok farklı alanlarda, çok sayıda modüllerin, paketlerin var olmasının önemli özelliklerinden olduğunu ve python'un modüler bir programlama dili olduğunu belirtmiştik. Şimdi bu noktada modül ve paket kavramından biraz bahsedeceğiz.

Modül, birtakım fonksiyonları ve nitelikleri barındıran araçlar olarak tanımlanabilir. Modüller bizim için tekrar tekrar kod yazmamızın önüne geçen yapılardır. Bu noktada Python'un temel olarak parçalardan bir bütün elde etmek için tasarlanmış bir dil olduğunu söylemek yanlış olmaz. Aslında şu ana kadar kullandığımız hazır fonksiyonlar ile birçok defalar Python'un hazır kütüphanesinde bulunan modüllerden yararlandık. Standart python kütüphane modülleri kurulumla beraber sistemimize yüklenmiştir. Standart modüllere ek olarak Python ile uğraşan programcılar farklı amaçlara uygun birçok modül geliştirmişler ve yayınlamışlardır. Bunların birçoğu açık kaynak kodludur ve kurulum gerektirmektedir. İlerleyen bölümlerde bu modüllerden bahsedilecektir.

Python kütüphanesindeki hazır modülleri http://docs.python.org/3/library/ adresinden öğrenebilirsiniz.

İsterseniz şimdi bir modülü kodumuza ekleme ve modülde yer alan bir fonksiyonu kullanma üzerine bir örnek yapalım.

```
import random
print (random.randint(1, 100))
```

Bu kod parçacığında 1 ile 100 arası rasgele bir sayı üretmemizi sağlayan "randint(1,100)" fonksiyonu kullanılmış. Bu hazır fonksiyon aslında bir paketin içerisinde daha önce tanımlanmış bir fonksiyondur. O nedenle kodumuza bu paketin eklen-

mesi gereklidir. Bu ekleme işlemi için ise kullanılan anahtar kelime "import" tur. Bu kelime sayesinde random modülü kodumuza eklenmiş ve bu modül içerisinde bulunan randint() fonksiyonu kodumuzda kullanılır hale gelmiştir. Eğer bu eklemeyi yapmadan fonksiyonu kullanma çalışırsak "name 'random' is not defined" şeklinde bir uyarı alırız.

Bir modül içerisinde hangi fonksiyonların olduğunu "dir()" ile öğrenebiliriz. Aşağıdaki kod ile random modülünün içerisindeki fonksiyonların isim listesini elde edebiliriz. Aşağıdaki kodun çalışması ile ekrana ['BPF', 'LOG4', 'NV_MAGIC-CONST', ...] ile başlayan uzun bir liste dökülür.

```
import random
print(dir(random))
```

Eğer modül içerisinden belli başlı bazı fonksiyonları kullanacaksak, tüm modülü yüklememize gerek olmayabilir. Bunun için, "from modülAdi import fonksiyonListesi" ifadesi kullanılır.

```
from random import randint
print(randint(1,100))
```

Temel olarak bakıldığında yazdığımız her kod ve .py uzantılı dosya bir modüldür ve import deyimi ile başka kodla entegre edilebilir.

Tabiki yazdığımız bir kodu herhangi bir koda eklemek istiyorsak kodumuzun uygun yere taşınması gerekmektedir. Sistemimizdeki varsayılan dizini öğrenmek için gerekli import işleminden sonra "sysconfig.get_python_lib()" komutu kullanılabilir.

Kısaca modüllerden bahsettikten sonra şimdide "Paket" kavramını inceleyelim. Paket, bir dizin yapısı içerisinde bir araya getirilmiş birbiri ile ilgili modüllerden oluşan yapıya verilen isimdir. Python'u tam anlamı ile etkili kullanabilmek ve hızlı kod üretebilmek için modüllerden amacımıza uygun olanları bilmek çok önemlidir. Örneğin amacımız bir web sayfası yapmak ise "Django", bir masaüstü uygulaması geliştirmek ise "tkinter" veya bir android veya iOS için program yazmak ise ise "kivy" paketlerini iyi bilmemiz gerekmektedir.

Bir paketin kodumuza dahil edilmesi modül dahil etme ile aynıdır. Kodumuza bir paket içerisindeki tek bir modülü eklemek istediğimizde ise "import paketAdi.modülAdi" veya "from paketAdi import modülAdi" komutu kullanılmalıdır. Bir pakette yer alan bir modülün belli fonksiyonlarını eklemek istiyorsak ise "from paketAdi.modülAdi import fonksiyonListesi" satırı kullanılmalıdır.

Bir paketin kurulması ile ilgili bilgi için https://packaging.python.org/installing/ adresini ziyaret ediniz.

Python' da yer alan yararlı modül ve paket listesi kısa açıklamalar ile beraber şu şekilde listeleyebiliriz.

- *Kivy:* Mobil cihazlara program yazabilmek için geliştirilmiş bir pakettir. Kivy'i kulanarak yazılan programlar tüm platformlarda çalışabilmektedir. Temel olarak PyGame paketi üzerine kurulmuş bir yapıdır.
- *Pygame:* Python'ı kullanarak platform bağımsız olarak oyun ve çokluortam uygulamaları geliştirmemizi sağlayan bir kütüphanedir. SDL kütüphanesi üzerine kurulmuştur. SDL (Simple DirectMedia Layer) C programlama dili ile yazılmış grafik, ses, klavye ve mouse etkileşimi sunan bir kütüphanedir.

PyGame ile düzenlenen oyun geliştirme turnuvalarına https://pyweek.org/ linkinden ulaşabilirsiniz.

- *PIL (Python Imaging Library):* PIL, Kullanıcısına grafik işleme için birçok hazır nesne ve metot sunar. Birçok grafik türünü açıp çizim, düzenleme, filitreleme benzeri işlemleri yapmamıza olanak verir.

- *NumPy:* Sayısal (Numerical) python'un kısaltılmış halidir. Özellikle veri odaklı çalışan programcılar için önemli nesne ve metotları içerir. Çok boyutlu dizi/vektör/matris hesaplamaları için özelleşmiş, bunlar üzerinde çok hızlı işlem yapılabilen nesnelere ve metotlara sahiptir. Ayrıca lineer matematik işlemlerini de desteklemektedir.
- *SciPy:* Matematiksel fonksiyonları ve farklı bilim dallarındaki (fizik, kimya, biyoloji vb.) problemlerin çözümü için gerekli hazır fonksiyonları içerir. Scipy kütüphanesi, Numpy dizilerini kullanmaktadır.
- *Pandas:* Veri yapılarını sık kullanan bireyler için uygun bir pakettir. Excel ve SQL verilerini kolaylıkla işleyebilir. Finansal işlemleri yüksek performans ile destekleyen çok sayına nesne ve metot barındırır.
- *Matplotlib:* Verileri görselleştirmede kullanılan önemli bir pakettir. NumPy paketi ile uyumlu bir şekilde çalışabilir. Bu iki paket ile MATLAB programının kapasitesine yakın bir kapasite yakalanabilir.
- *Polyglot:* Doğal dil işleme alanında çalışanların kullanabileceği temel fonksiyonları birçok dil için destekleyen bir pakettir.
- *PyQtGraph:* 2 ve 3 boyutlu grafikler oluşturmak, matematiksel, bilimsel ve mühendislik yazılımları için gerekli işlevleri içeren, PyQt, NumPy, SciPy ve PyOpenGL kütüphanelerinden yararlanan grafik ve GUI (Graphical User Interface/ grafiksel kullanıcı arayüzü) kütüphanesidir.
- *PySonic:* Platform bağısız ses kütüphanesidir.
- *SQLAlchemy:* Python için bir veritabanı erişim kütüphanesidir.
- *MySQLdb:* Python ile MySQL veritabanını kullanma için tasarlanmış Python modülüdür.
- *PyQT:* Programcılara, GUI oluşturma için gerekli altyapıyı sunan bir kütüphanedir. PyQT platform bağımsız bir kütüphanedir. Dünya çapında birçok firmanın yazılım geliştirme için seçtiği kütüphanedir.
- *PyGtk:* Tıpkı PyQt gibi, kullanıcılara GUI arabirimleri oluşturmamıza imkân veren bir kütüphanedir. Platform bağımsız olarak uygulama geliştirilebilir. GTK+ kütüphanesini kullanır.
- *Tkinter*: GUI uygulamaları geliştirmek için tasarlanmış bir kütüphanedir. En öneli özelliği python programlama dili ile beraber gelmesidir. İlerleyen kısımlarda Tkinter'a biraz daha detaylı değinilecektir.
- *WxPython*: Bir başka platform bağımsız GUI uygulamaları geliştirmek için kullanabileceğimiz kütüphane.
- *Scapy*: Network programlama için gerekli yapıları içeren python kütüphanesidir.

PyQT hakkında ayrıntılı bilgi için https://riverbankcomputing.com/software/pyqt/intro adresini ziyaret edebilirsiniz.

PyGTK+ hakkında ayrıntılı bilgi için www.gtk.org/ adresini ziyaret edebilirsiniz.

- *PyQuery*: XML belgeleri üzerinde Jquery sorgularının gerçekleştirmemizi sağlayan kütüphanedir.
- *Web2Py*: Python dilinde geliştirilmiş web uygulaması çatısıdır. Web2Py, web sayfası geliştiricilerine dinamik web içeriklerini kolayca oluşturmaları için gerekli alt yapıyı sunmaktadır.
- *Django*: Django, tıpkı web2Py gibi bir web uygulama çatısıdır. MTV (Model-View-Template) mimarisini destekleyen django'nun temel amacı genelde karmaşık bir yapıya sahip veritabanı kullanabilen web sayfalarını hızlıca tasarlanmasıdır. Modüler ve yeniden kullanılabilir (re-usability) felsefelerini temel almaktadır. Bu kütüphane hakkında daha geniş bilgi ileride sunulacaktır.

Öğrenme Çıktısı

2 Python dilinde program denetimi yapılarını açıklayabilme

Araştır 2

Kullanıcının girdiği 100'lük sistemdeki notu harf sisteminde yazdıran programı yazınız. Ayrıca 1'den 10' a kadar olan sayılar için çarpım tablosunu ekrana döken bir program yazınız.

İlişkilendir

Diğer internet tabanlı programlama dillerinde de var olan kontrol ve döngü yapılarının Python'daki sözdizimlerini avantaj ve zorlukları açısından ilişkilendirerek karşılaştırınız.

Anlat/Paylaş

: Python'da kullandığınız fonksiyon ve döngü yapıları ile ilgili deneyimlerinizi paylaşınız.

PYTHON İLE NESNEYE YÖNELİK PROGRAMLAMA

Bölümün başlarında da bahsettiğimiz gibi python neneye yönelik programlamayı desteklemektedir. Aslında Python programlama dilinde, tüm program öğeleri, nesne yapısındadır. Fakat python programcıya nesneye yönelik programlamayı zorunlu kılmaz. Fakat hazır kütüphanelerin neredeyse tümü bu yaklaşım ile yazılmıştır. Yani python kullanıyorsanız nesneye yönelik programlama kullanıyorsunuz demektir. Dilerseniz şimdi python kullanarak sınıf ve nesnelerin tanımlanmasını ve kullanılmasını inceleyelim.

Sınıfı, nesnelerin özelliklerini, davranışlarını ve başlangıç durumlarını tanımlamak için kullanılan şablon tanımlamalar olarak belirtebiliriz. Bir sınıf tanımını kullanarak türettiğimiz nesne de aslında o sınıfın bir örneğidir. Başka bir deyişle sınıf, nesneleri türetmemizi sağlayan veri tipleri şeklinde de tanımlanabilir.

Python'da Sınıf ve Nesne

Her programlama dilinde sınıf ve nesne tanımlamaları farklı şekillerde yapılmaktadır. Python'da ise içi boş bir sınıf şu şekilde tanımlanabilir.

```
class DortIslem:
    pass
yeniİslem=dortIslem()
```

Görüldüğü gibi sınıf tanımlaması için "class" ifadesini kullandık. Sınıf tanımlama gövdesinde ise sadece pass yazdık. Bu şimdilik "DortIslem" sınıfı hakkında daha fazla bilgi vermeyeceğimiz anlamını taşımaktadır. Sınıf tanımlamasının ilk satırı "class DortIslem():" şeklinde parantez kullanılarak da yazılabilir. Şimdi "DortIslem" sınıfını biraz geliştirelim ve değişken tanımlamalarını yapalım. Birçok dilin aksine python, sınıftan türetilecek nesnelerin değişken tanımlarını "__init__" isimli bir metot içerisinde gerçekleştirir.

```
class DortIslem:
    def __init__(self):
        self.sayi1=15
        self.sayi2=25
ilkIslem=DortIslem()
print(ilkIslem.sayi1, ilkIslem.sayi2)
```

Örnekten de görüldüğü üzere nesne değişkeni olarak sayi1 ve sayi2 değişkenlerinin tanımı __init__ metodu içerisinde gerçekleştirilmiştir. Yani, "ilkIslem=DortIslem()" satırında ilkIslem nesnesi türetilirken ilk olarak __init__ metodu çalışmakta ve o nesne için sayi1 ve sayi2 değişkenleri tanımlanarak ilk atamaları yapılmaktadır. Bu tanımlamada dikkati çeken ilk nokta "self" kelimesidir. Bu kelime esasen o an için değişken tanımlaması yapılacak nesneyi temsil etmektedir. Yani, metot içerisinde tanımlayacağımız her türlü değişken sınıftan türetilecek nesneler için ayrı ayrı tutulacaktır. "self"

kelimesi yerine istenen herhangi kelime kullanılabilir. Tek koşuk __init__ metodunun ilk parametresi olmaktır. Genel olarak kodlarda self kelimesi tercih edilmektedir.

Yukarıdaki örneğimizde sayi1 ve sayi2 değişkenlerine sabit değerler atanmıştır. Aşağıdaki örnekte ise nesne türetirken değişken değerleri __init__ metoduna gönderilen parametreler ile belirlenmiştir.

```
class DortIslem:
    aciklama="Bu metot iki sayı üzerinde tanımlanan islemleri yapar."
    def __init__(self, a, b):
        self.sayi1=a
        self.sayi2=b
islem1=DortIslem(13, 54)
islem2=DortIslem(4, 12)
```

Örnekte görüldüğü üzere __init__ metodunun parametre sayısını arttırdık ve aldığımız a ve b parametrelerini sayi1 ve sayi2 nesne değişkenlerinin başlangıç değerleri olarak atadık. Bu şekilde nesne türetme kodumuzda "islem1 = DortIslem(13, 54)" şeklinde değiştirdik. Bu şekilde türetilen islem1 nesnesinin sayi1 ve sayi2 değişkenleri sırası ile 13 ve 54 değerlerini alırken, islem2 nesnesinin sayi1 ve sayi2 değişkenleri de 4 ve 12 değerlerini almış oldu.

Yukarıdaki örnekte __init__ metodu dışında "aciklama" ismi de bir değişken tanımlama daha yapılmıştır. __init__ metodu dışında yapılan tanımı yapılan değişkenler, esasen nesnenin değil sınıfın değişkeni şeklinde tanımlanırlar. Bunu açıklamak için sınıf içerisinde yazdığımız değişkenleri yazdıralım.

```
print ("islem1 nesnesi \n\t sayi1 =",islem1.sayi1,"\n\t sayi2="
    ,islem1.sayi2, "\n\t aciklama =",islem1.aciklama)
print ("islem2 nesnesi \n\t sayi1 =",islem2.sayi1,"\n\t sayi2 ="
                        ,islem2.sayi2, "\n\t aciklama =",islem2.aciklama)
print ("DostIslem sınıf değişkeni \n\t aciklama= ",DortIslem.aciklama)
```

Kodların ekran çıktısı şu şekilde olacaktır.

```
islem1 nesnesi
       sayi1 = 13
       sayi2 = 54
       aciklama = Bu metot iki sayı üzerinde tanımlanan islemleri yapar.
islem2 nesnesi
       sayi1 = 4
       sayi2 = 12
       aciklama = Bu metot iki sayı üzerinde tanımlanan islemleri yapar.
DostIslem sınıf değişkeni
       aciklama =  Bu metot iki sayı üzerinde tanımlanan islemleri yapar.
```

Görüldüğü üzere, "aciklama" değişkeni DortIslem sınıfından türeyen tüm nesneler için aynı değeri alırken sayi1 ve sayi2 değişkenleri nesneye göre farklılaşmaktadır. Bunun nedeni "aciklama" değişkeninin nesnenin değil sınıfın değişkeni olmasıdır. Sınıf değişkenleri bu özellikleri nedeni ile sınıf üzerinden de çağrılabilirler (örnekte "DortIslem.aciklama" satırı). Fakat nesne değişkenleri sınıf üzerinden çağrılamazlar. Yani eğer DortİslemDortIslem.sayi1 ifadesi geçersizdir ve "AttributeError: type object 'DortIslem' has no attribute 'sayi1'" hatası verir.

Metotlar

init sınıf içerisinde yazdığımız ilk metottu. Eğer sınıfımıza farklı kabiliyetler kazandırmak istiyorsak metot sayısını istediğimiz kadar artırabiliriz.

```
class DortIslem:
    aciklama="Bu metot iki sayı üzerinde tanımlanan islemleri yapar."
    def __init__(self, a, b):
        self.sayi1=a
        self.sayi2=b
    def toplama(self):
        return self.sayi1+self.sayi2
    def cikarma(self):
        return self.sayi1-self.sayi2
    def carpma(self):
        return self.sayi1*self.sayi2
    def bolme(self):
        return self.sayi1/self.sayi2
    def us(self):
        return self.sayi1**self.sayi2
    def toplamininUssu(self, a):
        return (self.sayi1+self.sayi2)**a
    def yazdir(self):
        print ("sayi1=",self.sayi1," \n sayi2=",self.sayi2)
```

Örnekte görüldüğü üzere sınıfımıza 7 adet metot daha ekledik. Bu metotlardan toplama, çıkarma, çarpma, bölme ve us nesne içerisinde tanımlı iki sayı ile işlem gerçekleştirip sonucu "return" ifadesi ile geri döndürmektedirler. toplamininUssu metodu ise nesne değişkenlerini toplayıp dışarıdan aldığı bir parametreyi kullanarak üs işlemi gerçekleştirmiş ve çıkan sonucu döndürmüştür. Son metodumuz yazdir ise diğerlerinden farklı olarak, değişkenleri ekrana yazdırmıştır. Herhangi bir veriyi geri döndürmemiştir. Aşağıdaki kodlarda ise tanımlanan bazı metotların kullanımı yer almaktadır.

```
islem=DortIslem(4, 8)
print(islem.toplama())
print(islem.toplamininUssu(2))
islem.yazdir()
```

Kodların çıktısı şu şekilde olacaktır.

```
12
144
sayi 1=4
sayi 2=8
```

Şu ana kadar tanımladığımız metotların hepsi nesne metotlarıdır. Bu metotlar tıpkı nesne değişkenleri gibi nesne üzerinden çağırılırlar ve nesne değişkenleri üzerinde işlem gerçekleştirirler. Eğer biz sadece sınıf değişkeni üzerinde işlem yapacaksak bir başka deyişle nesne değişkenlerini kullanmayacaksak nesne metotları yerine sınıf metotlarını kullanmamız daha doğru olacaktır.

```
class DortIslem:
    aciklama="Bu metot iki sayı üzerinde tanımlanan islemleri yapar."
    def __init__(self, a, b):
        self.sayi1=a
```

```
        self.sayi2=b
    @classmethod
    def aciklamaSil(cls):
        cls.aciklama=""
    @classmethod
    def aciklamaDegistir(cls, yeniaciklama):
        cls.aciklama=yeniaciklama
```

Görüldüğü üzere yukarıdaki sınıf tanımında aciklamaSil ve aciklamaDegistir isimli iki yeni metot tanımlamıştır. Sınıf metodunu yazarken, metot tanımından önce "@classmethod" isimli bir ifade kullanılmıştır. Bu ifade bu metodun bir sınıf metodu olduğunu işaretlemektedir. Yine metodun parametre listesinin ilk terimi "cls" olarak kullanılmıştır. "cls" sınıf belirteci olarak kullanılmıştır. "self" ifadesi gibi "cls" ifadesi de genel olarak kabul görmüş bir ifade olsa da yerine başka bir isimde kullanılabilir. Şimdi bu metotların nasıl kullanıldığına bakalım.

```
DortIslem.aciklamaDegistir("YENİ AÇIKLAMA")
print (DortIslem.aciklama)
```

Bu kod ile ekrana "YENİ AÇIKLAMA" yazısı basılacaktır. Görüldüğü üzere sınıf metodunu kullanmak için nesne metodlarında olduğu gibi bir nesne türetmek zorunda kalmadık. Dolayısıyla, sınıf metotları direk olarak sınıf üzerinden çağrıldığını söyleyebiliriz.

Eğer metotta nesne veya sınıf değişkenleri kullanılmayacaksa o zaman statik metotlar tercih edilmelidir. Aşağıdaki kodda iki adet statik metot tanımlanmıştır.

```
class DortIslem:
    aciklama="Bu metot iki sayı üzerinde tanımlanan islemleri yapar."
    def __init__(self, a, b):
        self.sayi1=a
        self.sayi2=b
    @staticmethod
    def topla(a, b):
        return a+b
    @staticmethod
    def mesaj(a):
        print ("Statik metodun mesajı: "+a)
```

Görüldüğü gibi tanımlanan iki metotta ne nesnenin ne de sınıfın değişkenleri üzerinde işlem yapmaktadır. Statik metot işaretlenirken "@staticmethod" belirleyicisi kullanılmaktadır. Nesne ve sınıf metotlarından farklı olarak statik metotlarda ilk parametre nesne veya sınıf belirteci olarak kullanılmaz.

Statik metotları, mantıksal olarak sınıf ile alakalı, fakat sınıfın veya nesnenin hiç bir değişkenini veya metodunu ihtiyacı olamayan kodlar için kullanılmalıdır. Static metotlar kullanılarak bu tip metotlar sınıf içerisine dâhil edilebilir. Statik metotlara erişimi hem sınıf üzerinden hem de sınıftan türetilecek nesne üzerinden gerçekleştirebiliriz.

Erişim Belirleyicileri

Bir sınıfın üyelerini yani değişkenlerini ve metotlarını, erişim durumlarına göre ikiye ayırmak mümkündür: public (genel erişime açık), private (genel erişime kapalı).

Erişime açık olma (public), eğer bir sınıfın üyesine direk olarak dışarıdan erişebiliyorsa buna "public" denmektedir. Şu ana kadar yazılan bütün kod parçalarındaki tüm public erişimine sahiptirler.

Erişime kapalı olma (private), bir sınıfın üyelerine direk olarak ulaşılamama durumunu belirtir. Aşağıda private erişim belirleyicisine sahip üyeleri olan bir sınıf tanımlanmıştır.

```
class YeniSinif:
    __aciklama="Yeni sınıftan merhaba..."
    def __init__(self,a):
        self.__deger=a
        self.__hosgeldin()
    def __hosgeldin(self):
        print ("hoşgeldin")
    def aciklamaGoster(self):
       print ("Aciklama=",self.__aciklama)
```

Bir sınıfta tanımlı private üyeler "__" iki adet alt çizgi ile başlamaktadır. Yukarıdaki kodda görüldüğü üzere "__aciklama" sınıf değişkeni, "__deger" nesne değişkeni ve "__hosgeldin" nesne metodu private erişim belirleyicisi ile tanımlanmışlardır. Şimdi bu üyelere erişimin nasıl olacağını inceleyelim. İlk olarak "__deger" değişkenine erişelim.

```
sinif= YeniSinif(15)
sinif.__deger=20
```

Private bir üyeye erişim isteğimiz "AttributeError: 'YeniSinif' object has no attribute '__deger'" şeklinde bir hata alacaktır.

```
print(YeniSinif.__aciklama)
```

Benzer şekilde "__aciklama" sınıf değişkenine erişimde de "AttributeError: type object 'YeniSinif' has no attribute '__aciklama'" hatası ile karşılaşacağız. En son olarak da private nesne metoduna erişmeyi deneyelim.

```
sinif= YeniSinif(15)
sinif.__hosgeldin();
```

Görüldüğü üzere benzer hata "AttributeError: 'YeniSinif' object has no attribute '__hosgeldin'" tekrar alınacaktır.

Sınıf içerisinde tanımlanan private üyelere erişim görüldüğü üzere bulunmamaktadır. Bu tip üyelere erişmek için ayrıca erişim metotlarının yazılması gerekmektedir.

> Aslında python dilinde private üyelere de ulaşılabiliriz. Örneğin __deger üyesine erişim için sinif._YeniSinif__deger kullanabiliriz. Fakat kod yazan kişiye saygı duymamız ve private olarak belirlenen üyelere bu şekilde ulaşmamamız daha iyi olacaktır.

Public ve private erişim belirleyicilerin yanında semi-private üyelerde bulunmaktadır. Bu üyeler isimlerinin önüne eklenen "_" tek alt çizgi karakteri ile tanımlanırlar. Fakat bu üyelere private üyelerde olduğu gibi dışarıdan erişimi engelleyen bir mekanizma bulunmamaktadır. Aslında bu yapı programcıların birbirine gönderdiği bir mesaj olarak algılanabilir. Mesaj, bize bu tip bir üyeyi değiştirmemiz gerektiğini belirtmektedir.

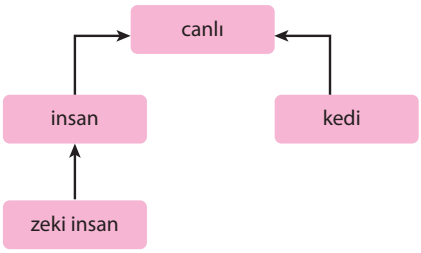
Miras (Inheritance)

Nesne Yönelimli Programlama dillerindeki en temel prensiplerinden biri Miras(Inheritance) kavramıdır. Bu prensip, bir sınıfın özelliklerinin ve metotlarının başka sınıflara aktarılmasına dayanır. Bu şekilde birbirlerine kodsal açıdan benzeyen sınıfları tek tek yazmak yerine, ortak noktaları belirlenerek temel bir sınıf (base class) oluşturulur. Daha sonra ise temel sınıfın ortak özellikleri üzerine yeni özellikler metotlar eklenerek türetilmiş sınıflar (derived class) oluşturulmaktadır. Aşağıdaki örnekte, bir taban sınıf tanımlanmış ve sınıflar türetilmiştir.

```
class canli():
    aciklama="Ben bir canlıyım"
    def __init__(self, e_s, a_s):
        self.elSayisi=e_s
        self.ayaksayisi=a_s
class insan(canli):
    def konus(self):
       print("Benim ", self.elSayisi, " elim ", self.ayaksayisi, " ayağım var." )
class zekiInsan(insan):
    def __init__(self, e_s, a_s):
        super().__init__(e_s, a_s)
        self.zekapuani=100
    def dusun(self):
        print("Benim ", self.elSayisi, " elim ", self.ayaksayisi, " ayağım var.
Aynı zamanda zeka puanım ", self.zekapuani," dır" )
class kedi(canli):
    def miyavla(self):
         print("miyavvvv... elsayım ", self.elSayisi, " ayak sayım ise ", self.
ayaksayisi, " dır." )
```

Yukarıdaki kod incelendiğinde toplam 4 adet sınıf (canli, insan, zekiInsan, kedi) tanımlandığı görülmektedir. Bu sınıfların hiyerarşisi şu şekildedir.

Görüldüğü üzere "canli" sınıfı temel sınıfından insan ve kedi sınıfı, insan sınıfından da zeki insan sınıfı türetilmiştir. Aşağıdaki tabloda bu dört sınıfın üyeleri listelenmiştir.



Resim 7.7 Sınıf Hiyerarşisi

Tablo 7.1 4 Sınıfın Üye Listesi

	canli	insan	zekiInsan	kedi
Sınıf Değişkenleri	elSayisi ayakSayisi	elSayisi ayakSayisi	elSayisi ayakSayisi zekapuani	elSayisi ayakSayisi
Nesne değişkenleri	aciklama	aciklama	aciklama	aciklama
Nesne Metotları		__init__() konus()	**__init__()** konus() dusun()	__init__() miyavla()

Görüldüğü üzere insan sınıf canli sınıfından türetilmiştir. Dolayısı ile canli sınıfının üyeleri insan sınıfı içerisinde geçerli olmakta ve insan sınıfında tekrar tanıtılmamaktadır. İnsan sınıfına canlı sınıfının üyeleri dışında konus() isimli bir metot da eklenmiştir. zekiInsan sınıfı insan sınıfından türetilmiş, ekstradan zekapuanı değişkeni ve dusun() metodu eklenmiştir. Zeki insan sınıfında ayrıca *__init__* metodu da tekrar tanımlanmıştır. init metodu tekrar tanımlanırken super().*__init__*(*e_s*, *a_s*)" komutu ile temel sınıfının *__init__* metodu çağırılmış, ardındanda istenen kodlar yazılmıştır. Bu şekilde *__init__*metodu yeniden yazılmıştır.

Tkinter

Tkinter, python ile beraber gelen masaüstü programlarda kullanılan kullanıcı arayüzlerini oluşturmamıza imkân veren bir pakettir. Daha önce de bahsedildiği gibi farklı paketler ile de grafik arayüzüne sahip programlar yazabiliriz. Bu paketlerin en bilinenleri PyQti PyGI, PyGtk, Kivy gibi paketlerdir. Tkinterin avantajı, paketlerinin kurulu gelmesi, sağlam, kararlı ve oturmuş bir arayüz paketi olmasıdır. Bu kısımda temel olarak tkinter paketinin kullanımı anlatılacaktır.

İlk olarak boş bir pencere oluşturalım.

```
import tkinter
ekran = tkinter.Tk()
ekran.geometry('300x300+100+100')
ekran.title("ilk ekran")
ekran.mainloop()
```

Tkinter paketini kullanmak için öncelikle dosyamız içerisine paketi eklememiz gerekmektedir. Bunun için "import tkinter" kodunu kullanmaktayız. İstenirse "as" ifadesi ile tkinter paketine bir takma isim de verilebilir. Bu komutun ardından Tk() metodu ile bir pencere tanımlanır. Ardından ise geometry metodu kullanılarak pencerenin ebatları ve konumunu, title metodu ile pencerenin başlığı ayarlanabilir. geometry metodunda parametre olarak girilen "300x300" pencerenin ebatını verirken, parametrenin "100+100" kısmı ise x ve y ekseninde konumunu vermektedir. En son ise "`ekran.mainloop()`" komutu ile pencere görünür hale getirilir.

Eğer pencerenin simgesini değiştirmek istiyorsak, "`ekran.wm_iconbitmap(ikon dosyasının yolu)`" metodunu kullanabiliriz. Varsayılan olarak oluşturduğumuz pencere kullanıcı tarafından boyutlandırılabilir. Eğer biz penceremizin boyutlandırılmasını istemiyorsak, "`ekran.resizable(False, False)`" metodunu kullanabiliriz. Metottaki ilk parametre enine ikinci parametre boyuna boyutlandırmayı kontrol etmektedir. Yani metodu "`ekran.resizable(True, False)`" şeklinde kullanırsak, bu oluşturulan ekranın enine boyutlandırabildiğini fakat boyuna boyutlandırılamadığını gösterir. Tkinter paketinde yer alan minsize() ve maxsize() metotları ise pencere maksimum ve minimum ebatlarını belirlememizi sağlamaktadır.

Tkinter hakkında daha fazla özellik için https://docs.python.org/2/library/tkinter.html adresini ziyaret edebilirsiniz.

Tkinter kütüphanesi, pencere içerisinde kullanabileceğimiz birçok nesneye sahiptir: label, button, entry, checkbutton, listbox, menu, scrollbar vb. İsterseniz bu noktada pencere tasarımında kullanacağımız nesnelerden bazılarını inceleyelim.

Label()

Tasarımlarda en fazla kullanılan nesnelerden biri olan Label'in pencere kullanım kodları şu şekildedir.

```
import tkinder as tk
ekran = tk.tk()
ekran.title("Label Kullanımı")
ekran.geometry('300x300')
ilkEtiket=tk.Label(text="Tkinder'e hoş-
geldiniz.", fg="blue", bg="yellow")
ilkEtiket.pack()
ekran.mainloop()
```

Kod içerisinde label tanımlanması "`ilkEtiket=tk.Label()`" komutu ile gerçekleştirilmiştir. Bu tanımlama yapılırken parametre olarak verilen "text" ile etiket yazısı "bg" değişkeni ile yazı arkaalan rengi, "fg" değişkeni ile de yazı rengi belirlenmiştir. Renk değerleri hexadecimal renk kodları kullanılarak da verilebilmektedir. Ayrıca fg yerine foreground, bg yerine background ifadeleri de kullanılabilmektedir. Label nesnesi oluşturulurken kullanılabilecek başka parametreler de bulunmaktadır.

Örneğin, label'da kabartma ayarı için "relief" parametresi flat (kabartmasız), sunken (alçak kabartma), raised (yüksek kabartma), groove (oluklu kabartma), ridge (kenar kabartma) ve solid (düz kenar kabartma) değerleri ile kullanılabilir. Kenarlık için "border" parametresi (bd olarakta kullanılabilir) kalınlık değeri ile verilebilir. Label nesnesinin boyutu için "width" ve "heigth" parametreleri kullanılabilir. Eğer Label içerisine resim eklemek istiyorsak ise kodumuz aşağıdaki şekilde yazabiliriz.

```
resim = tk.PhotoImage(file="python.gif)
ilkEtiket=tk.Label(text="Tkinder'e hoşgeldiniz.", image=resim,
compound="top", font=("Verdana",30, "bold"))
```

Yukarıdaki kodda öncelikle "PhotoImage()" komutu (istenirse "BitmapImage" komutu da kullanılabilir.) ile resim değişkeni oluşturulmuş, ardından ise bu resim değişkenini "image" parametresine değer olarak atılmıştır. "compound" parametresi, resmin yazıya göre pozisyonunu belirlemektedir. "font" parametresi ile de yazı tipini değiştirebiliriz.

Oluşturduğumuz label nesnenin pencere içerisinde görünebilmesi için pencere içerisinde nereye yerleşeceğinin belirlenmesi gerekmektedir. Bunun için " ilkEtiket.pack()" komutu kullanılmıştır. Nesnelerin pencere içerisine yerleşmesi amacı ile farklı yapılar kullanılabiliriz. Bu yapılara (pack, grid ve place) ilerleyen kısımlarda kısaca değinilecektir. Kodun çalıştırılması ile elde edilen ekran görüntüsü aşağıdadır.

Resim 7.8 Ekran Çıktısı

Button()

Grafiksel arayüz tasarımında bir diğer önemli pencere elamanı da button'lardır. Button'ların kullanımı da label kullanımına oldukça benzerdir. İsterseniz ufak bir program ile button nesnesini inceleyelim. Aşağıdaki kod ile button'a basıldığında etiketin metninin ve metin renginin değişimi gerçekleştirilmiştir.

```
import tkinder as tk
ekran = tk.Tk()
ekran.title("Düğme Kullanımı")
ekran.geometry('300x300')
e=tk.Label(text="Düğmeye basın...")
e.pack()

def degistir():
    e["text"]="Etiketi değiştirdin. Tebrikler..."
    e["foreground"]="red"

ilkDugme=tk.Button(text="BAS", command=degistir)
ilkDugme.pack()
eran.mainloop()
```

Kodda görüldüğü üzere bir label nesnesinin yanı sıra bir button nesnesi "`Button(text="BAS", command=degistir)`" kodu ile üretilmiş ve "`ilkDugme.pack()`" kodu ile pencere üzerine yerleştirilmiştir. Button fonksiyonu içerisinde "text" parametresi ile button üzerindeki yazı, "command" parametresi ile de button'a basıldığında çalıştırılması istenen fonksiyon verilmektedir. Kod içerisinde tanımlanan button'da "degistir()" fonksiyonu çağırılmaktadır. değiştir() fonksiyonu tanımlı etiketin yazısını değiştirmektedir. Sonuç olarak programın ekran görüntüsü aşağıdaki şekilde olacaktır.

Resim 7.9 Ekran Çıktısı

Button fonksiyonunun, tıpkı label fonksiyonunda olduğu gibi, renk (background, foreground), kabartma (relief), kenarlık (border), boyut (width, height), yazı tipi (font) vb. özellikleri bulunmaktadır.

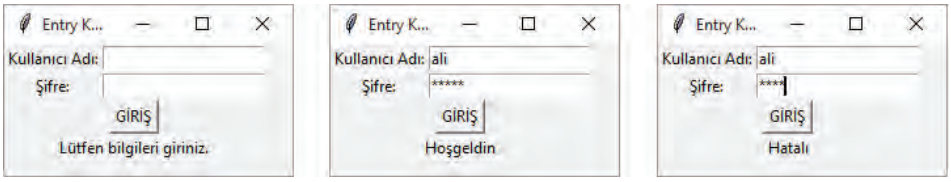
Entry()

Entry, kullanıcıdan veri almamızı sağlayan nesnelerdir. Entry nesnesi "`Entry()`" komutu ile tanımlanır. Renk (background, foreground), kabartma (relief), kenarlık (border), boyut (width, height), yazı tipi (font) vb. özellikleri bulunmaktadır. Aşağıdaki kod da kullanıcı adı ve şifre bilgisini alan ve bilgilerin doğruluğuna göre mesaj veren bir kod parçası yer almaktadır.

```
import tkinter as tk
def kontrol():
    if (kulAd.get()=="ali") and (sifre.get()=="12345"):
        l3["text"]="Hoşgeldin Ali"
    else:
        l3["text"]="Hatalı Şifre"

ekran = tk.Tk()
ekran.title("Entry Kullanımı")
ekran.geometry('220x100')

l1=tk.Label(text=" Kullanıcı Adı ")
l1. grid(row=0,column=0)
kulAd=tk.Entry()
kulAd.grid(row=0, column=1)
l2=tk.Label(text="Şifre:")
l2.grid(row=1,column=0)
sifre=tk.Entry(show="*")
sifre.grid(row=1, column=1)
b=tk.Button(text="GİRİŞ")
b.grid(row=2, column=0, columnspan=2)
l3=tk.Label(text="Lütfen bilgileri giriniz.")
l3.grid(row=3, column=0, columnspan=2)

ekran.mainloop()
```

Tkinter'da bulunan checkbutton, listbox, menu, scrollbar vb. diğer nesneler hakkında detaylı bilgi için https://docs.python.org/2/library/tkinter.html adresini ziyaret ediniz.

Yukarıda görüldüğü üzere, tanımlanan iki adet "entry" nesnesinden ikincisinde kullanılan "`show="*"`" parametresi sayesinde şifre girişi için uygun hale getirilmiştir. Button'a basılması ile çalışan değiştir fonksiyonunda entry içerisinde kullanıcının girdiği veriye "get()" metodu ile ulaşılmıştır. Entry içerisindeki metne veri eklemek için ise entry nesnesi üzerinden "`insert(index, metin)`" metodu, entry içerisindeki metinden veri silinmesi isteniyor ise "`delete(ilkindex,son index)`" metodunun çağrılması gerekir. Bu kodda daha öncekilerden farklı olarak nesnelerin pencere yerleşiminde "pack()" yerine "grid()" pencere yöneticisi kullanılmıştır. Kodun çalıştırılması ile elde edilen ekran görüntüleri aşağıda gösterilmiştir.



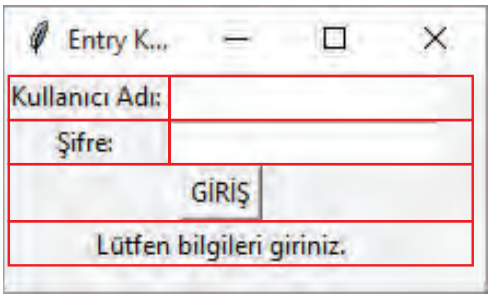
Resim 7.10 Ekran Çıktısı

Tkinter'da Geometri Yönetimi

Tkinter'da nesnelerimizi bir pencere içerisine veya başka bir nesne içerisine yerleştirirken kullanabileceğimiz 3 farklı yaklaşım bulunmaktadır: pack, grid ve place. Bunlardan ilk ikisini şu ana kadar yazdığımız kodlar içerisinde kullandık. Şimdi ise biraz daha detaylandıralım.

Pack kullanarak, nesnelerimizi alt alta (varsayılan) veya yan yana yerleştirmek mümkündür. Pack, side, expand, fill, padx, pady vb. parametreler alabilir.

- "side" parametresi ile nesneyi yerleştireceğimiz konumu belirleyebiliriz. "side" parametresine left (sol), right (sağ), top (üst), bottom (alt) gibi değerler atayabiliriz.
- Varsayılan olarak, pencere üzerine yerleştirdiğimiz nesneler pencere boyutu büyüse dahi konumlarını değiştirmezler. "expand" parametresi, elemanın konumunu pencere büyütüldüğünde göreceli olarak değiştirir.
- "fill" parametresi ile nesnenin x, y ekseninde veya her iki eksende de pencereyi doldurması için kullanılır.
- "anchor" parametresi, nesnenin yerini belirler ve bu noktada sabit kalmasını sağlar. Bu sayede nesnenin konumu, pencerenin boyut değişiminde değişmez. Parametrenin alabileceği değerler n (kuzey), s (güney), e (doğu), w (batı), ne (kuzeydoğu), nw (kuzeybatı), se (güneydoğu), sw (güneybatı) ve center (orta) şeklindedir.
- "padx", "pady" parametreleri, nesnenin belirlenen yerleşim noktasından x ve y ekseninde uzaklığını belirler. Bu uzaklık varsayılan olarak piksel cinsindendir. Fakat istenirse c (santimetre), i (inc), m (milimetre) veya p (point) cinsinde de verilebilir.



Resim 7.11 Grid Yapısı

Grid, pencere alanımızı bir tablo (table) gibi kullanmamızı sağlar. Bu sayede nesneleri daha düzenli bir şekilde yerleştirebiliriz. "Entry" nesnesini anlatırken kullandığımız kodu dikkatlice incelerseniz ekran tasarımında aşağıdaki gibi bir tablonun varlığını görebilirsiniz.

Grid, row, column, rowspan, columnspan, padx, pady, sticky vb. parametrelere sahiptir.

- "row" ve "column" parametreleri genelde birlikte kullanılır. Bu parametreler nesnenin tasarımda hangi hücrede olacaklarını gösterirler. Örneğin kullanıcının şifre girişinde kullandığı sifre isimli entry için yerleştirme " `sifre.grid(row=1, column=1)`" kodu ile sağlanmıştır.
- "rowspan" ve "columnspan" parametresi ise satır veya sütunda birleştirme işlemi yapılmak için kullanılır. Örneğimizde button nesnesinin yerleştiği bölüm "`b.grid(row=2, column=0, columnspan=2)`" kodu kullanılarak, iki sütunun birleştirilmesiyle oluşturulmuştur.
- "sticky" parametresi ile de nesneleri yaslamak için kullanılır. n (kuzey), s(güney), w (batı) ve e (doğu) değerlerini alır.

Place, nesneleri pencere üzerinde istediğimiz konuma yerleştirmemizi sağlar. Kullanımı basit olmasına rağmen çok tercih edilmemektedir. Bunun nedeni yapılan tasarımın farklı çözünürlüğe sahip bilgisayarlarda sorun yaratabilmesidir. Place, x, y, relx, rely, width, height, relwidth, relheight vb. parametrelere sahiptir.

- "x" ve "y" parametresi nesnenin konumunu belirlemek için kullanılmaktadır. Pencere boyutlandırılmasında konum değişmez.
- "relx" ve "rely" parametresi ise benzer şekilde konumu belirler. Fakat pencerenin boyutu değiştirildiğinde göreceli olarak pozisyonu değiştirir. Bu parametreler sadece 0.0 ile 1.0 arasında değerleri kabul eder.
- "width" ve "height" parametreleri nesnenin konumundaki bloğun genişlik ve yüksekliğini ayarlar.
- "relwidth" ve "relheight" parametreleri ise genişlik ve yüksekliği pencere boyutuna göre düzenler.

Öğrenme Çıktısı

3 Python dilinde nesneleri kullanabilme
4 Python dilinde tkinter kullanarak masaüstü görsel programlama yapabilme

Araştır 3

Koordinat düzleminde bir noktayı ve doğruyu temsil eden aşağıdaki listelenen metodlara sahip iki sınıf tasarlayınız. Nokta Sınıfı: noktanın koordinatlarını ekrana yazan noktaYazdir metodu. Doğru Sınıfı: doğrunun uzunluğunu hesaplayan dogruUzunluk metodu, doğrunun orta noktasını döndüren ortaNokta Metodu ve doğrunun iki noktasına ait koordinatları ekrana basan dogruYazdir metodu.

İlişkilendir

Nesne yönelimli programlama yapısının temelinde yer alan ve Python dilinde de başvurulan Miras prensibini anlamsal Web bileşeni olan ontolojilerle ilişkilendiriniz.

Anlat/Paylaş

Tkinter paketi kullanımına ilişkin deneyimlerinizi paylaşınız.

DJANGO WEB ÇATISI

Django, Python programlama diliyle yazılmış bir web sayfası geliştirme çatısıdır (web framework). Çatı, hızlı, güvenilir, modüler ve sağlam bir kodlama yapabilmemiz için kullanılan altyapı olarak tanımlanabilir. Django MVC (Model-View-Controller) mimarisine benzer MTV (Model-View-Template) mimarisi ile çalışmaktadır. Genelde bir program yazılım süreci temel olarak aynı işlemleri içermektedir. Bu nedenle aynı kodları tekrar tekrar yazmak yerine çatının sağladığı altyapı olanakları kullanmak sürece olumlu yönde katkı yapmaktadır.

internet

MTV mimarisi hakkında ayrıntılı bilgi için http://djangobook.com/model-view-controller-design-pattern/ adresiniz ziyaret edebilirsiniz.

Temel olarak django'yu, farklı dillerde hazırlanan wordpress, joomla, openCart, nukePHP'ye benzetmek çok yanlış olmaz. Django, python kurulumu ile birlikte gelmemektedir. Kurulumun işletim sistemine göre yapılması gerekmektedir. Bu nedenle isterseniz ilk önce windows işletim sistemi için django kurulumunu inceleyelim.

internet

https://www.djangosites.org/ adresinde django kullanılarak tasarlanan siteleri bulabilirsiniz.

Kurulum

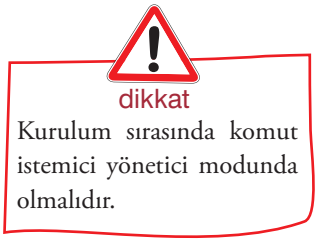
Django kurulumu için öncelikle bilgisayarınızda python yüklü olması gerekir. Ardından https://www.djangoproject.com/download/ sayfasından en son sürümü indirmeniz ve herhangi bir klasör içerisine açmanız gerekmektedir. Django kurulumu komut istemcisinin açılması django klasörüne gidilmesi ve aşağıdaki kodların yazılması gereklidir.

```
c:\Django> python setup.py install
```

Kurulum başlayacak ve hiçbir ayar gerektirmeden sonlanacaktır. Kurulumun doğru gerçekleşip gerçekleşmediğini öğrenmek için

```
import django
print (django.VERSION)
```

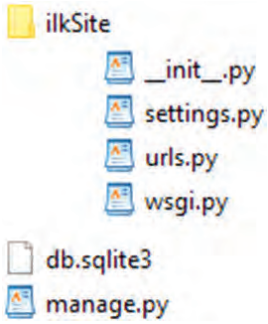
yazılır. Kod çalışırsa kurulum başarı ile gerçekleşmiştir. Django ile kod yazmak için farklı editörler kullanılabilir: aptana studio, wingware, spyder, pycharm vb.



Proje Oluşturma

Django ile proje oluşturmak için aşağıdaki işlemlerin dos kabuğunda sırayla uygulanması gerekir.

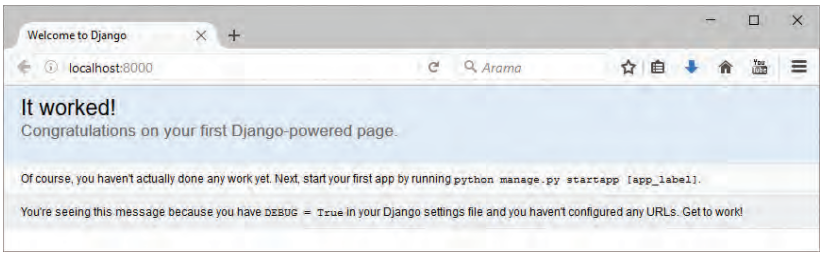
```
C:\Django>mkdir projeler
C:\Django>cd projeler
C:\Django\projeler>django-admin startproject ilkSite
```



Resim 7.12 Django Proje Dizin Yapısı

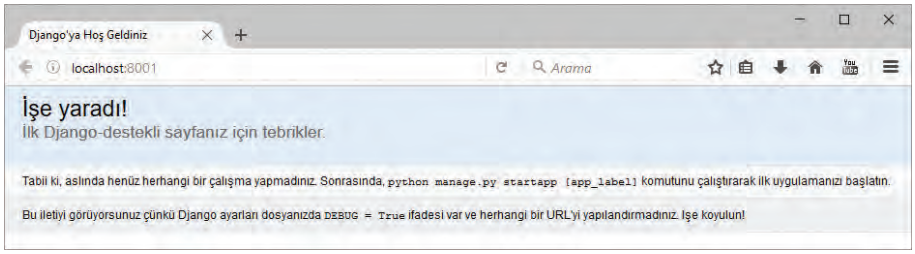
Bu işlemden sonra proje klasörünün içerisine bazı klasör ve dosyalar otomatik olarak oluşturulacaktır.

"manage.py", projemiz ve uygulamalar ile ilgili komutların yer aldığı yönetim dosyadır. Bu dosya genellikle düzenlenmez. "init.py" dosyası, esas olarak bir python projesi olduğunu belirtir. İçine özel bazı komutlar eklenebilir. "setting.py" projemiz ile ilgili yapılandırmaların yer aldığı dosyadır. "url.py", projedeki kullandığımız adresleri içerir. Boş bir şekilde gelir. "wsgi.py" dosyası WSGI (Web Server Gateway Interface) uygulamasını içermektedir. İlk projemizi çalıştırmak için konsolu kullanmamız gerekmektedir. Sunucu üzerinde bir sitenin hizmet verebilmesi için "`C:\Django\projeler\ilkSite> python manage.py runserver`" komutu kullanılır. Eğer sitemizin başka bir porttan hizmet vermesini istersek komutumuzu "`C:\Django\projeler\ilkSite> python manage.py runserver 127.0.0.1:8748`" şeklinde yazmamız gerekir. Sitemiz yayına başladığında aşağıdaki ekran görüntüsü alınacaktır.



Resim 7.13 İlk Site Ekran Alıntısı

Görüldüğü üzere sayfanın dili varsayılan olarak İngilizcedir. Eğer sayfanın dilini değiştirmek istersek setting.py dosyasındaki "LANGUAGE_CODE" değişkeninin değeri 'tr-TR' şeklinde değiştirilmelidir. Bu değişiklikten sonra site tazelendiğinde aşağıdaki ekran görüntüsü elde edilir. Bu kodda aynı zamanda port ayarı da yapılmış, yeni port numarası 8001 olarak ayarlanmıştır.



Resim 7.14 Türkçe Dil Ayarı Yapıldıktan Sonra İlk Site Ekran Alıntısı

Bu noktadan sonra artık yavaş yavaş web sitesini tasarlamaya başlayalım. Django'nun çalışma mantığını anlamak için "Merhaba Dünya" gibi bir örnekle başlayalım. Django'da herhangi bir sayfanın görüntülenebilmesi için görünümün hazırlanıp, görünüme ulaşılacak adresle bağlama işleminin yapılması gereklidir. İlk önce görünümü hazırlayalım. Bunun için bir python dosyası oluşturmamız gerekiyor. Dosyanın isminin ne olduğu çok önemli olmasa da genel olarak views.py ismi kullanılmaktadır.

```
from django.http import *

def hello(request):
    text=""
    for i in range(10):
        text+="<b>Merhaba</b> <br>"
    return HttpResponse(text)
```

Görüldüğü üzere ilk önce django.http paketi kodumuza dahil edilmiş ardından da "hello" fonksiyonu tanımlanmıştır. "hello" fonksiyonu içerisinde "`HttpResponse`" fonksiyonu kullanılmıştır. Bu fonksiyon parametre olarak verilen metin ile web sayfasını oluşturur. "views.py" dosyası projenin kök dizinine atılabilir. Sayfanın görüntülenebilmesi için dosyanın proje ile url bağlantısını kurmamız gerekmektedir. Bu işlem için "urls.py" dosyası aşağıdaki gibi düzenlenmelidir.

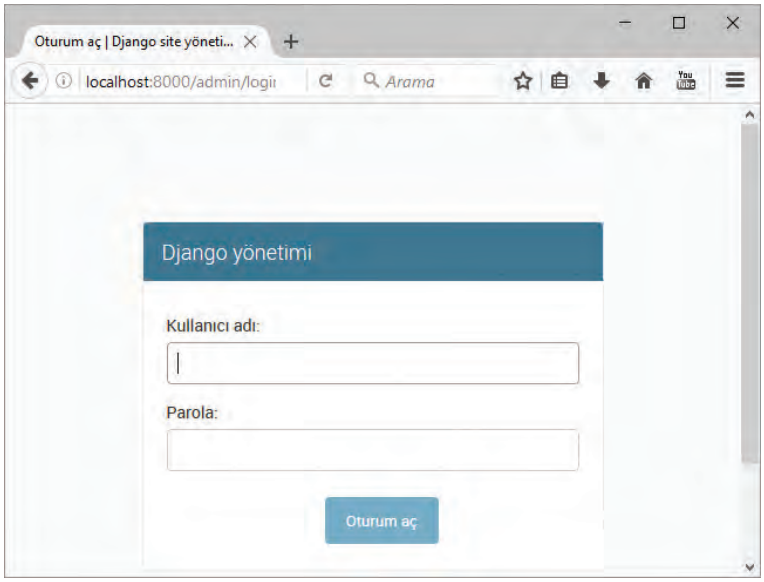
ÖNCEKİ

```
from django.conf.urls import url
from django.contrib import admin

] = urlpatterns
url(r'^admin/',  admin.site.
,(urls
[
```

SONRAKİ

```
from django.conf.urls import url
from django.contrib import admin
import views

] = urlpatterns
url(r'^admin/',  admin.site.
,(urls
(url(r'^selam/', views.hello
[
```

"urls.py" proje ile bağlantı kurulacak sayfaların tanımlamaları yapılmaktadır. Görüldüğü üzere dosyanın önceki halinde varsayılan olarak bir url tanımlaması bulunmaktadır. Bu tanımlama django'nun projemiz için otomatik oluşturduğu "admin" sayfasıdır.



Resim 7.15 Projenin Otomatik Oluşturulan Admın Giriş Ekranı

"Url" tanımlaması yapılırken öncelikle adres çubuğunda sayfanın çağrılacağı isim, ardından ise sayfa çağrıldığında çalıştırılacak olan fonksiyon belirtilmiştir. "views.py" dosyasını projeye bağlamak amacı ile " `url(r'^selam/', views.hello)` " kodu kullanılmıştır. Burada "selam" çağrılma kelimesi olarak seçilmiş selam çağrıldığında çalışacak fonksiyon "views.selam_ver" olarak belirlenmiştir. Tabiki "urls.py" dosyasına "import views" satırının eklenmesi unutulmamalıdır. Bu düzenlemeden sonra "localhost:8000/selam" adresi aşağıdaki gibi olacaktır.

internet

Admin sayfası hakkında detaylı bilgiye https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Admin_site ve http://first-django-admin.readthedocs.io/en/latest/ bağlantıları üzerinden ulaşılabilir.

Resim 7.16 İlk Sayfanın Ekran Görüntüsü

Bir projede ilk sayfamızı tanımladığımız zaman, "localhost:8000" adresi artık " Page not found (404)" hatası vermeye başlayacaktır. Bu nedenle tıpkı index. html, index.php gibi projemize bir ana sayfanın tasarlanması ve tanımlanması gerekmektedir. Bunun için öncelikle "views.py" içerisinde bir fonksiyon tanımlamalı (örneğim main_page gibi) ve ardından "urls.py" dosyasında bağlantısı "url(r'^$',views.main_page)" kodu ile kurulmalıdır.

Eğer sitemizde dinamik bir sayfa tasarlayacaksak her bir sayfayı "urls.py" dosyasında tanımlamamız gerekmektedir. Bazen sayısını bizimde bilmediğimiz bu sayfaları tanıtırken düzenli ifadeler kullanmamız gerekmektedir. Düzenli ifadeler ile joker karakterler kullanıp birden fazla sayfa tanımlamasını tek bir komutla gerçekleştirebiliriz. İsterseniz bunu bir örnek ile açıklayalım. Örneğimizde, öğrenci numarası url adresinde verilen bir öğrencinin bilgilerini sunacağımız bir sayfa tasarlayalım. İlk önce views.py dosyasına bir fonksiyon tanımlamak ile işe başlayacağız.

```
def ogrenci(request,ogrenci_no):
    ogrenci_no = int(ogrenci_no)
    ogrenciler = {
        155: ("Ahmet","Tan"),
        223: ("Ayşe","Can"),
        255: ("Mehmet","Sam"),
        321: ("Yasemin","Ay")
        }
    if ogrenci_no in ogrenciler:
        html="""<table border=1>
    <tr><td colspan=2><strong>Öğrenci Bilgileri</strong></td></tr>
    <tr><td>Ad</td><td>%s</td></tr>
```

```
<tr><td>Soyad</td><td>%s</td></tr>
</able>"""
%(ogrenciler[ogrenci_no][0], ogrenciler[ogrenci_no][1])
else:
html="<strong>%s</strong>Öğrenci bulunamadı."%(ogrenci_no)
return HttpResponse('<strong>Merhaba Dünya</strong>')
```

Görüldüğü üzere ogrenci fonksiyonu daha önce yazdıklarımızdan farklı olarak iki parametre almaktadır. "öğrenci_no" değeri adres çubuğundan girilecek verinin içine atanacağı değişkendir. İlk satırda bu değişken integer veri tipine dönüştürülmüştür. Ardından öğrenci verilerini tutulduğu sözlük (dictionary) veri tipinde "öğrenciler" isimli bir değişken tanımlanmıştır. Girilen öğrenci_no değerinin sözlükte tanımlı olup olmadığı sorgulanmış ve eğer böyle bir öğrenci mevcut değilse "html" değişkenine "öğrenci bulunamadı" değeri atanmıştır. Eğer öğrenci numarası mevcutsa, "html" değişkeninin değerine 3x2 boyutunda bir tablo oluşturup içerisine sözlükten okuduğu değerleri yazan formatlı bir metin ifadesi atanmıştır. "ogrenci" fonksiyonunun tanıtılmasından sonra bu dinamik sayfanın proje ile url bağlantısının yapılması gerekmektedir. Bu işlem için "urls.py" dosyasına aşağıdaki satırın yazılması gerekir.

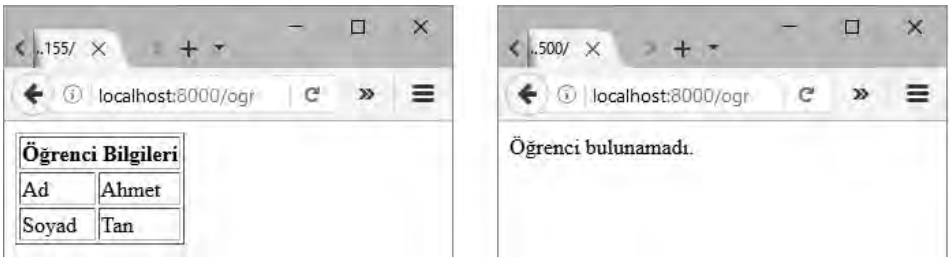
```
url(r'^ogrenci/([0-9]+)/$', views.ogrenci)
```

Yazılan bu kod ile "http://localhost:8000/ogrenci/[ogrenci_no]" şeklindeki url adresler tanımlanmış olacaktır. Kodda, düzenli ifade olarak "([0-9]+)" kullanılmıştır. [0-9] ile 0 ile 9 arasında rakamlar, + ile de bu ifadenin 1 vey a daha fazla tekrarı ifade edilmiştir. Bu şekilde, "http://localhost:8000/ogrenci/155" veya "http://localhost:8000/ogrenci/4585" gibi sayfalar tanımlanmış olmaktadır. Aşağıdaki tabloda kullanabileceğimiz düzenli ifade sembolleri ve açıklamaları yer almaktadır.

Tablo 7.2 Düzenli İfadeler

Sembol	Açıklama
. (nokta)	Herhangi bir karakter
\d	Herhangi bir rakam
[A-Z]	Herhangi bir karakter (büyük harf)
[a-z]	Herhangi bir karakter (küçük harf)
[A-Za-z]	Herhangi bir karakter (büyük - küçük harf duyarsız)
+	Sonrasında yer aldığı ifadeden 1 veya daha fazla tekrar olabileceğini belirtir.
[^/]+	Eğik çizgi hariç tüm karakterler
?	Sonrasında yer aldığı ifadeden 1 veya daha fazla tekrar olabileceğini belirtir.
{1,5}	Sonrasında yer aldığı ifadeden 1-5 arası tekrar olabileceğini belirtir.

Örnek çalıştırıldığında aşağıdaki gibi bir çıktı alınacaktır.



Resim 7.17 Sayfanın Ekran Alıntısı

Şablon Kullanımı

Az önceki örnekte görüldüğü gibi sayfamızı tasarlarken kod ile tasarımı aynı sayfa içerisinde içiçe gerçekleştirdik. Oysaki python ve HTML/CSS yazmak farklı bir iştir ve olması gereken şey tasarım ile kodların birbirinden ayrılmasıdır. Şablonlar, sayfalarda kod ile tasarımı birbirinden ayırmamızda bize yardımcı olurlar. Şablonlar yardımı ile tasarımda veya kodda yapılacak bir değişiklik diğerini etkilemeyecek ve kod yazım işlemi paralel olarak yürüyebilecektir.

Django'da şablon kullanmak için üç temel komut bulunmaktadır: template.Template, template.Context ve render.

```
from django import template
sablon = template.Template("Öğrenci İsmi: {{isim}}")
c = template.Context({"isim":"Ali"})
print (sablon.render(c))
```

Şablonları kullanabilmek için öncelikle template modülünün eklenmesi gerekmektedir. Ardından template.Template ile tasarım kodları tanımlanmıştır. Bu kodların içerisinde "{{isim}}" ifadesi bu kısmın değişeceğini belirtmektedir. Şablonlar içerisinde benzer birçok etiket kullanılabilir. Bu etiketler ileride tanıtılacaktır. Ardından şablon için bağlam (context) tanımlaması yapılmıştır. Bağlam anahtar ve değerlerin tanımlandığı bir sınıftır. Yapı olarak bir sözlük (dictinary) gibi çalışmaktadır. Kodda, "c" isminde bir bağlam tanımlanmıştır. En son satırda ise bağlamın şablon içerisinde tanımlı ifadelere yerleştirildiği render fonksiyonu çalıştırılmaktadır. Kodun çalışması sonucunda; `Öğrenci İsmi: Ali` çıktısı elde edilecektir. Yukarıdaki örnekte görüldüğü üzere, bağlama sadece bir metin gönderileceği gibi bir sözlükte gönderilebilmektedir.

```
from django.http import *
from django import template

def ogrenci(request)
    ogrenci={"isim": "Ali", "soyad": "Can", "sinif": 2}
    sablon = template.Template("Öğrenci İsmi: {{ogrenci.isim}}<br>
    Öğrenci soyadı: {{ogrenci.soyad}}<br>
    Sınıf: {{ogrenci.sinif}}")
    c = template.Context({"ogrenci":ogrenci})
    html=sablon.render(c)
    return HttpResponse(html)
```

Kodda görülen ogrenci fonksiyonu, "views.py" dosyası içerisine yazılmıştır. Görüldüğü üzere, şablon ile bağlam ilişkilendirildikten sonra, HttpRespnse fonksiyonu ile görüntülenmesi sağlanmıştır. "urls.py" dosyasına yapılacak url ilişkilendirmeden sonra elde edilen ekran çıktısı şu şekildedir.

Resmi 7.18 Şablon Örneği Ekran Alıntısı

Temel Etiketler

Şu ana kadar django şablonları içerisinde sadece veri aktarmayı gördük. Şimdi ise farklı etiketlerden bahsedeceğiz. Django'da *if else* etiketinin, yapısı aşağıdaki şekildedir.

```
{% if Koşul %}
....
{% else %}
....
{% endif %}
```

dikkat

Sayaç indisinde ulaşmak için forloop.counter etiketi kullanılmalıdır.

internet

Django içerisinde yer alan etiketlere https://docs.djangoproject.com/en/1.10/ref/templates/builtins/ adresinden ulaşabilirsiniz.

if etiketinin koşul kısmında kullanılabilecek karşılaştırma ve mantıksal operatörleri python ile aynıdır. ifequal/ifnotequal etiketi ise iki değişkenin eşit olup olmadığını kontrolde kullanılabilir. for etiketinin söz dizimi ise; `{% for %}...{% endfor %}`şeklindedir. For döngü etiketinde tıpkı python da olduğu gibi aralık belirtmek için range, liste, sözlük vb. veri yapılarının elemanlarını dolaşmak için in ifadesi kullanılır. Şablonlarda, yorum satırları için ise "`{# Bu bir yorumdur. #}`" kullanılır.

Filitreler

Şablonda kullanılan filitreler, değişkenleri görüntülemeden önce değiştirmemize imkan veren yapılardır. Kullanımı; `{{ baglam_degiskeni|filtre:filtre_argumani}}` şeklindedir. Bazı filitreleri şu şekilde sıralayabiliriz.

Tablo 7.3 Django Template Filitrelerinden Bazıları

Filitre	Kullanım	Açıklama
lower	{{ value|lower }}	değişkenin tüm harflerini küçük harfe dönüştürür.
upper	{{ value|upper }}	değişkenin tüm harflerini büyük harfe dönüştürür.
length	{{ value|length }}	değişkenin uzunluğunu döndürür
add	{{ value|add:"2" }}	değişkenin değerine 2 ekler
capfirst	{{ value|capfirst }}	değişkenin ilk harfini büyük harfe dönüştürür.
center	{{ value|center:"15" }}	Değişkeni verilen genişliğin ortasına yerleştirerek gösterir.
first	{{ value|first }}	Liste değişkeninin ilk elemanını gösterir.
wordcount	{{ value|wordcount }}	Değişkendeki kelime sayısını gösterir.
urlize	{{ value|urlize }}	Adresleri url biçimine dönüştürmemizi sağlar

Buraya kadar şablonlar ile verdiğimiz örneklerde şablon tanımlamalarını ve bağlam atamalarını aynı yerde yaptık. Oysaki daha önce de söylediğimiz gibi şablon ile bağlamın birbirinden ayrılması gerekmektedir. Bunu gerçekleştirebilmek için yapacağımız tasarımları farklı dosyaya kaydedip bu dosyayı fonksiyonumuzda çağırmamız ve bağlam ile eşleştirmemiz gerekmektedir. İsterseniz bunu bir örnek üzerinde gösterelim. İlk önce şablon dosyası;

internet

Django içerisinde yer alan etiketlere https://docs.djangoproject.com/en/1.10/ref/templates/builtins/#built-in-filter-reference adresinden ulaşabilirsiniz.

```
<!DOCTYPE html>
<html>
<head>
 <title>Öğrenci Bilgi Sayfası</title>
</head>
<body>
    <table border="1">
      <tr><td>Öğrencinin Adı: </td><td>{{ ogrenci.isim }}</td> </tr>
      <tr><td>Öğrencinin Soyadı:</td><td>{{ ogrenci.soyad }}</td></tr>
      <tr><td>Öğrencinin sınıfı:</td><td>{{ ogrenci.sinif }}</td></tr>
    </table>
</body>
</html>
```

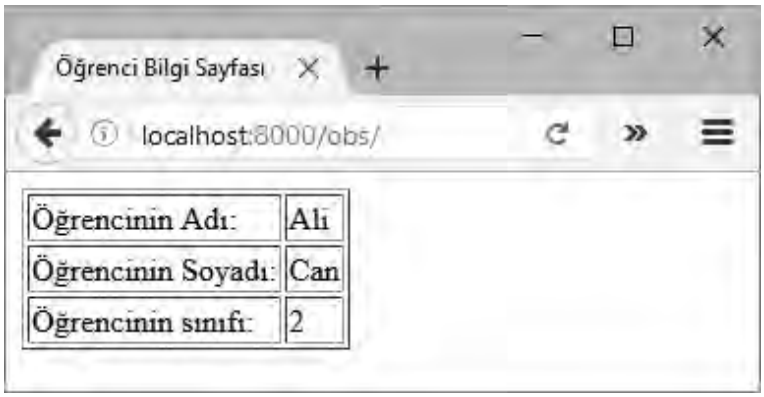
Şablon dosyasını proje ana dizini içerisinde oluşturduğumuz "templates" isimli "obs.html" klasöre kaydedelim. Ardından "views.py" dosyamıza fonksiyonumuzu yazalım.

```
from django.http import *
from django import template
from django.template.loader import get_template

def obs(request)
    ogrenci={"isim": "Ali", "soyad": "Can", "sinif": 2}
    sablon = get_template('obs.html')
    c = template.Context({"ogrenci":ogrenci})
    html=sablon.render(c)
    return HttpResponse(html)
```

Görüldüğü üzere, dış dosya olarak hazırladığımız obs.html şablon dosyamızı "get_template" fonksiyonu ile çağırdık ve ardından bağlam ile ilişkilendirdik. Sayfamızı çalıştırabilmek için son bir işimiz daha kaldı. O da şablon dosyalarımızın nerede olacağını tanımlamak. Bu tanımlama için "settings.py" dosyasındaki TEMPLATES altında tanımlı `'DIRS': []`, satırını `'DIRS': [os.path.join(BASE_DIR, 'templates')]`, şeklinde değiştirmemiz gerekmektedir. Bu sayede aslında sisteme şablon dosyaları nerede bulabileceğini söylemiş olduk. Sonuç olarak yazdığımız kodun ekran çıktısı şu şekilde olacaktır.

dikkat

Şablonların içi içe kullanmak için {% include dosya_adı %} etiketini kullanabilirsiniz.



Resim 7.19 Dış Şablon Örneği Ekran Alıntısı

Formlar

Formlar, kullanıcıdan veri alabilmek için tasarlanmış html nesneleridir. Formlarda kullanıcı tarafından girilen veriler istenilen sayfaya POST veya GET metodu kullanılarak gönderilirler. GET metodunda gönderilecek veri(ler)nin değişken adı ve içeriği adres çubuğunda görülürken, POST metodunda gösterilmemektedir. Buna ek olarak GET metodu ile göndereceğimiz veri 2KB'tan fazla olamazken POST metodunda böyle bir sınır bulunmamaktadır. Dilerseniz Django ile form ve POST metodunun kullanımını bir örnek üzerinden açıklayalım. İsterseniz örnekte POST ifadesi gördüğünüz yerleri GET ile değiştirerek örneği GET metodu ile çalışabilecek bir hale getirebilirsiniz.

Örneğimizde ilk sayfada ismini aldığımız kullanıcıya ikinci sayfada merhaba diyelim. İlk sayfamızın tasarımı şu şekilde olacaktır.

```
<!DOCTYPE html>
<html>
<head>
    <title>Merhaba Uygulaması</title>
</head>
<body>
    {% if error %}
        <p style="color: red;">Lütfen Bir İsim Giriniz.</p>
    {% endif %}
<form action="/merhaba/" method="post">
    {% csrf_token %}
    İsminiz : <input type="text" name="isim" />
    <input type="submit" value="Giriş" />
</form>
```

Görüldüğü üzere öncelikle bir hata parametresi gelirse sayfada görülmesini istediğimiz uyarıyı yazdıktan sonra bir metin kutusu ve butondan oluşan formumuzu tasarladık. Sırada göndereceğimiz veriyi tutacak bir koda ihtiyacımız var. Bunu için "views.py" içerisine aşağıdaki kodları yazalım.

Kodda yer alan "{% csrf_token %}" ifadesi CSFR saldırılarına önlem olarak kullanılmaktadır. Lütfen sayfa kaynak kodunu inceleyiniz ve ayrıntılı bilgi için https://www.bilgiguvenligi.gov.tr/web-guvenligi/webdeki-buyuk-tehlike-csrf.html adresini ziyaret ediniz.

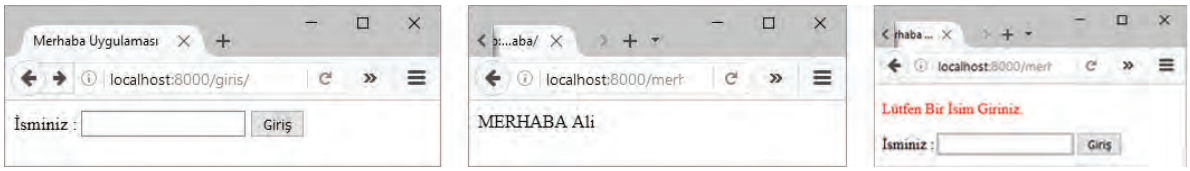
```
def merhaba_form(request):
    return render(request, 'merhaba.html')

def merhaba(request):
    if 'isim' in request.POST and request.POST['isim']:
        isim = request.POST['isim']
        return HttpResponse('MERHABA %s'%(isim))
    else:
        return render(request, 'merhaba.html', {'error': True})
```

Görüldüğü üzere iki sayfamız için views.py içerisine iki fonksiyon yazdık. Bunlardan ilki sadece merhaba html sayfamızı görüntülemektedir. İkinci fonksiyon ilk önce "isim" değişkeninin POST metodu ile gelip gelmediğini kontrol edilmektedir. Eğer "isim" değişkeni mevcut ise ekrana yazdırmakta, değişken mevcut değilse hata mesajının görüntülenmesi için merhaba.html dosyası error değişkeni ile çağrılmaktadır. En son olarak, sayfalarımızı "urls.py" dosyasına ekleyelim.

Render işlemlerimizi daha hızlı yapmamızı sağlayan render() fonksiyonunun çalışabilmesi için dosyanın başına `"from django.shortcuts import render"` eklemeyi unutmayın.

```
url(r'^giris/',views.merhaba_form),
url(r'^merhaba/',views.merhaba),
```

Sonuç olarak sayfaların ekran çıktıları aşağıdaki gibi olacaktır.



Resim 7.20 Form Uygulaması Ekran Alıntıları

İstenirse "views.py" içerisine yazdığımız iki fonksiyon tek bir fonksiyonda birleştirilebilir.

```
def merhaba(request):
    error=False
    if 'isim' in request.POST:
        isim = request.POST['isim']
        if not isim:
           error=True
    else:
           return HttpResponse('MERHABA %s'%(isim))
    return render(request, 'merhaba.html', {'error': error})
```

Burada öncelikle "isim" değişkeninin POST ile gelip gelmediği kontrol edilmektedir. Eğer gelmedi ise error değişkeni False olarak merhaba.html çalıştırılmaktadır. Eğer "isim" değişkeni POST ile geldiyse içeriğinin boş olup olmadığı kontrol edilmektedir. Eğer içerik boş ise error değişkeni True olarak Merhaba.html dosyası çalıştırılmakta, eğer "isim" değişkeni veri içeriyorsa istenilen şekilde ekrana basılmaktadır.

Öğrenme Çıktısı

5 Django web çatısını kurabilme
6 Django şablon kullanabilme

Araştır 4

Django ile kullanıcının girdiği ifadeyi ve girdiği font büyüklüğü ile ekrana basan kodları yazınız.

İlişkilendir

Django web sayfası geliştirme çatısı ile JavaScript kütüphanesi olan jQuery ile karşılaştırınız.

Anlat/Paylaş

Django kurulumu sırasında yaşadığınız problemleri ve çözüm önerilerini paylaşınız.

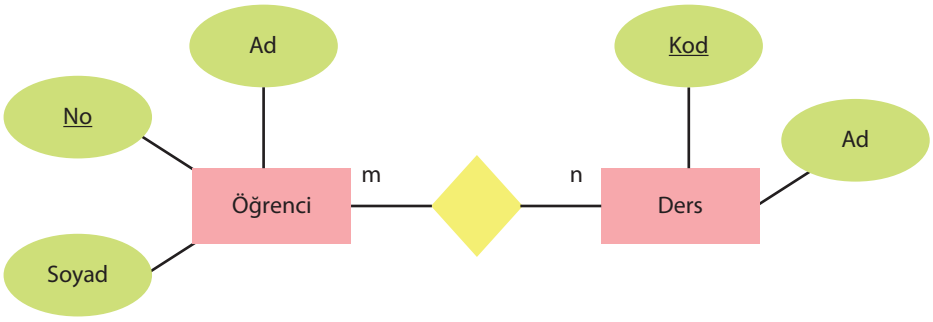
DJANGO MODELLER

Günümüzde web sitelerinin birçoğu en az bir veritabanı kullanmaktadır. Django bir web sitesinin veritabanını gereksinimini karşılamak için gerekli tüm yapılara sahiptir. Sağladığı fonksiyonlar sayesinde, veritabanı kullanırken SQL komutlarını bilmemize de gerek kalmaz. İsterseniz öncelikle django ile veritabanı konfigürasyonumuzu nasıl yapabileceğimizi inceleyelim. Veritabanı ayarları "settings.py" dosyasında yapılmaktadır. Varsayılan şekli aşağıdaki gibidir.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Burada, ENGINE ile kullanılacak veritabanı sistemi belirlenir. Varsayılan olarak sqlite3 veritabanı kullanılmaktadır. Burada farklı veritabanı sistemleri de tanıtabiliriz: PostreSQL (django.db.backends.postgresql_psycopg2), mySQL (django.db.backends.mysql), Oracle (django.db.backends.oracle) vb. NAME ise veritabanının yolunu ve ismini içermektedir. Bunlar dışında, USER (veritabanı kullanıcısı ismi), PASSWORD (veritabanı kullanıcı şifresi), HOST (veritabanı yerel makina dışında tanımlı ise sunucu adresi), PORT (port numarası) gibi tanımlamalarda yapılabilir.

"settings.py" dosyasında gerekli ayarlar yapıldıktan sonra projemiz içerisinde veritabanı işlemlerini yapabilmek için bir uygulama oluşturmamız gereklidir. Bunun için konsolda şu kodu yazmamız gereklidir: python manage.py startapp vt_yonetim. Bu komutun işlenmesi ile proje dizinimizde vt_yonetim isimli bir klasör oluşacaktır. Klasör admin.py, apps.py, models.py, tests.py ve __init__.py gibi dosyalar içermektedir. Bu dosyalardan üzerinde çalışacağımız models.py ve views.py dosyası boş olarak gelmektedir. Oluşturduğumuz vt_yonetim uygulamasının projemiz tarafından tanınabilmesi için INSTALLED_APPS değişkenine "vt_yonetim," satırının eklenmesi gereklidir. MTV mimarisinin temel elemanı olan model, veritabanı yapısının python kodu olarak temsil edilen hali şeklinde tanımlanabilir. Modeller sayesinde veritabanının kontrolünü ve yönetimini kolaylıkla yapabiliriz. Projede veritabanını kullanabilmemiz için düşündüğümüz veritabanı yapısının model tanımlamasını "models.py" dosyasına yapmamız gerekmektedir. İsterseniz aşağıda ER (Entity/relationship) diyagramı verilen veritabanı yapısının modelini "models.py" dosyası içerisine yazalım.



Resim 7.21 Örnek Veritabanı ER Diyagramı

```
from django.db import models

class Ogrenci(models.Model):
    ad = models.CharField(max_length=30)
    soyad = models.CharField(max_length=30)

class Ders(models.Model):
    ad = models.CharField(max_length=100)
    ogrenciler = models.ManyToManyField(Ogrenci)
```

Görüldüğü üzere öğrenci ve ders için düşündüğümüz veritabanı yapısı model'e aktardık. Fakat tablolarda biricil anahtar olarak düşündüğümüz öğenci için "no", ders için "kod" alanlarını tanımlamadık. Çünkü django bu alanları otomatik olarak tanımlamaktadır. Sonuç olarak, Django, tanımladığımız her model için bir tablo oluşturulmaktadır. Tablolar arasında yer alan m-n tipinde ilişkiyi ise " `models.ManyToManyField` " ifadesini kullanarak tanımladık. İlişki tipi m-n lik olduğu için django yine otomatik olarak

bir tablo daha oluşturmaktadır. Modelde alan tanımlamalarında kullanılabilecek referanslardan bazıları aşağıdaki tabloda verilmiştir.

Tablo 7.4 Django Model Veri Tipi Referansları

Veri Tipleri		
AutoField	DurationField	NullBooleanField
BigIntegerField	EmailField	PositiveIntegerField
BinaryField	FileField	SlugField
BooleanField	FilePathField	SmallIntegerField
CharField	FloatField	TextField
DateField	ImageField	TimeField
DateTimeField	IntegerField	URLField
DecimalField	GenericIPAddressField	UUIDField

Modeli tanımladıktan sonra tablolarda hata kontrolü için `python manage.py check` komutu kullanılmaktadır. Eğer bir hata bulunamazsa sistem "System check identified no issues (0 silenced)." yanıtı verecektir. Ardından, `python manage.py makemigrations vt_yonetim` komutu ile django'ya veritabanındaki değişiklik olduğu bilgisi verilmelidir. Bu komutun çıktısı aşağıdaki gibi olacaktır.

```
Migrations for 'yonetim':
  0001_initial.py:
    - Create model Ders
    - Create model Ogrenci
    - Add field ogrenciler to ders
```

Modelimiz için oluşan SQL komutlarını görmek için ise şu komut kullanılır: `python manage.py sqlmigrate yonetim 0001`. Komut çıktısı şu şekildedir.

```
BEGIN;
--
-- Create model Ders
--
CREATE TABLE "vt_yonetim_ders" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
"ad" varchar(100) NOT NULL);
--
-- Create model Ogrenci
--
CREATE TABLE " vt_yonetim_ogrenci" ("id" integer NOT NULL PRIMARY KEY AUTOINCRE-
MENT, "ad" varchar(30) NOT NULL, "soyad" varchar(30) NOT NULL);
--
-- Add field ogrenciler to ders
--
CREATE TABLE " vt_yonetim_ders_ogrenciler" ("id" integer NOT NULL PRIMARY KEY
AUTOINCREMENT, "ders_id" integer NOT NULL REFERENCES " vt_yonetim_ders" ("id"), "og-
renci_id" integer NOT NULL REFERENCES " vt_yonetim_ogrenci" ("id"));
CREATE UNIQUE INDEX " vt_yonetim_ders_ogrenciler_ders_id_5eb7203b_uniq" ON " vt_
yonetim_ders_ogrenciler" ("ders_id", "ogrenci_id");
CREATE INDEX " vt_yonetim_ders_ogrenciler_49544093" ON " vt_yonetim_ders_ogren-
ciler" ("ders_id");
CREATE INDEX " vt_yonetim_ders_ogrenciler_eabeca19" ON " vt_yonetim_ders_ogren-
ciler" ("ogrenci_id");
COMMIT;
```

Artık veritabanımızı oluşturmak için şu kodu yazabiliriz: `python manage.py migrate`. Bu komut ile veritabanımızı ve django modelimizi oluşturmuş olduk.

Model ile Veri Erişimi

Bu bölümde oluşturduğumuz modeli kullanarak veritabanındaki verilere nasıl erişeceğimizden bahsedeceğiz. İsterseniz önce etkileşimli kabuk kullanarak birkaç öğrenci verisi ekleyelim.

Veri Listeleme

Kabuğu açmak için komut istemcisinde projenin bulunduğu klasöre geçip "python manage.py migrate" komutunu yazalım. Ardından aşağıdaki kodları yazarak Ogrenci tablosuna 3 öğrenci verisi ekleyelim. Görüldüğü üzere veri eklemek için ilk önce bir Ogrenci nesnesi oluşturup bunu ".save()" metodu ile kaydediyoruz.

```
In [1]: from yonetim.models import *
In [2]: ogrenci1=Ogrenci(ad='Ali', soyad='Can')
In [3]: ogrenci1.save()
In [4]: ogrenci2=Ogrenci(ad='Veli', soyad='Bal')
In [5]: ogrenci2.save()
In [6]: ogrenci3=Ogrenci(ad='Ayşe', soyad='Tek')
In [7]: ogrenci3.save()
```

Şimdi ise tüm öğrenci bilgilerini ekrana basacak bir sayfa tasarlayalım. Bunun için "vt_yonetim" klasörü içerisinde yer alan "views.py" dosyasına "ogrenci_listesi" isimli bir fonksiyon yazarak işe başlayalım.

```
def ogrenci_listesi(request):
        ogrenci_listesi = Ogrenci.objects.all()
        t = get_template('ogrenciler.html')
        c=template.Context({'ogrenci_list': ogrenci_listesi})
        html = t.render(c)
        return HttpResponse(html)
```

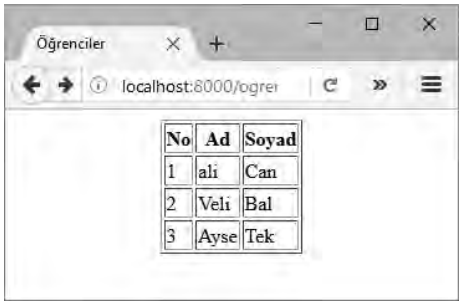
Burada, Ogrenci tablosunda yer alan tüm kayıtlar, "`Ogrenci.objects.all()`" komutu ile ogrenci_listesi değişkenine atanmakta, ardından bu değişken şablona parametre olarak gönderilmektedir. Şimdi, daha önce ayarladığımız "templates" klasörü içerisine şablonumuzu oluşturabiliriz. Şablonumuzun ismini "ogrenciler.html" olarak belirledik.

dikkat

Tabiki "views.py" dosyası içerisine gerekli dosyaları (from yonetim.models import Ogrenci, from django.http import * , from django import template, from django.template.loader import get_template) eklemeyi unutmamamız gerekir.

```
<!DOCTYPE html>
<html>
<head>
    <title>Öğrenciler</title>
</head>
<body>
<table align=center>
 <tr><th>No</th><th>Ad</th><th>Soyad</th></tr>
 {% for ogr in ogrenci_list %}
 <tr>
<td>{{ forloop.counter }}</td><td>{{ ogr.ad }}</td><td>{{ ogr.soyad }}</td></tr>
 {% endfor %}
</table>
</body>
</html>
```

En son olarak "urls.py" içerisine sayfa tanımlamamızı yaparak ("url(r'^ogrenci_list/',yonetim.views.ogrenci_listesi),") tasarımı tamamlıyoruz. Sonuç olarak öğrenci listesi sayfasının ekran çıktısı aşağıdaki şekilde olacaktır.



Resim 7.22 Öğrenci Listesi Ekran Alıntısı

Veri Ekleme

Şimdi kullanıcının öğrenci ekleyebilmesi için bir sayfa tasarlayalım. Tabiki ilk önce kullanıcının veri girebilmesi için bir form tasarımı ile işe başlamalıyız.

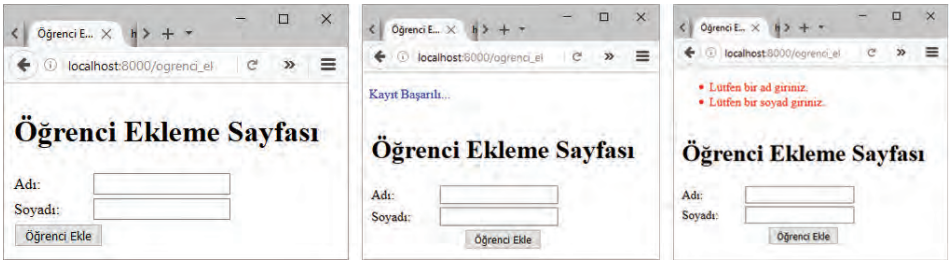
```
<!DOCTYPE html>
<html>
<head> <title>Öğrenci Ekle</title> </head>
<body>
{% if errors %}
    <p style="color:red">
    <ul style="color:red">{{ errors|unordered_list }}</ul>
    </p>
{% else %}
    {% if success %}
        <p style="color:blue">Kayıt Başarılı...</p>
    {% endif %}
{% endif %}
<form action="/ogrenci-ekle/" method="post">
  {% csrf_token %}
  <table>
    <tr><td colspan=2><h1>Öğrenci Ekleme Sayfası</h1></td></tr>
    <tr><td>Adı:</td><td><input type="text" name="ad"/></td></tr>
    <tr><td>Soyadı:</td><td><input type="text" name="soyad"/></td></tr>
    <tr><td colspan=2><input type="submit" value="Öğrenci Ekle"/></td></tr>
  </table>
</form>
</body> </html>
```

Formumuzu "ogrenci_ekle.html" ismi ile şablonları barındıran klasörümüze (templates) kaydedelim. Form tasarımının ardından ise POST metodu ile gönderilen verileri alıp veritabanına işleyecek kodları vt_yonetim klasörü içerisinde yer alan "views.py" dosyamız içerisine aşağıdaki şekilde yazalım.

```
def ogrenci_ekle(request):
    errors=[]
    if request.method == "POST":
        ad = request.POST['ad']
        soyad = request.POST['soyad']
        if not ad:
            errors.append("Lütfen bir ad giriniz.")
        if not soyad:
            errors.append("Lütfen bir soyad giriniz.")
        if errors:
    return render(request, 'ogrenci_ekle.html', {'errors': errors})
        else:
```

```
            ogrenci=Ogrenci(ad=ad,soyad=soyad)
            ogrenci.save()
    return render(request, 'ogrenci_ekle.html', {'success': True})
    return render(request, 'ogrenci_ekle.html', {'errors': errors})
```

Yukarıda yer alan kodda öncelikle POST ile gelen verinin olup olmadığı kontrol ettik. Eğer POST ile bir veri geliyorsa gelen verinin geçerliğinin kontrolü gerekmektedir. Bu amaçla örnekte ad ve soyad verileri için sadece var olma kontrolünü gerçekleştirdik. (Eğer kullanıcıdan e-mail, telefon numarası veya ders notu gibi farklı verilerde alınsaydı farklı kontroller ihtiyacımız olacaktı.) Birden fazla veri kontrolü yaptığımızdan dolayı burada en iyi yol hataların bir dizi yapısı (errors) içerisinde tutulmasıdır (errors isimli dizi şablon kodumuzda kullanılmıştı.). Bu nedenle biz de oluşabilecek her hatada diziye yeni bir hata cümlesi ekledik. Ardından hata dizisinin kontrolünü gerçekleştirdik. Eğer bir hata varsa "errors" dizisini göndererek ogrenci_ekle.html sayfası çalıştırdık. Eğer bir hata ile karşılaşılmadıysa veritabanına kayıt işlemini gerçekleştirerek ogrenci_ekle.html sayfasını "success" değişkeni True değerini vererek çağırdık. Sonuç olarak elde edilen ekran çıktıları aşağıda verilmiştir.



Resim 7.23 Öğrenci Ekleme Sayfası Ekran Alıntıları

Veri Seçme

Veritabanı sistemlerinde belki de en fazla kullanılan işlem veri seçme işlemidir. Veritabanı ile çalışan programlar genelde verilerin arasından belli kriterlere uygun kayıtlar üzerinde işlemler gerçekleştirirler. Tabloda yer alan tüm verilerin seçilmesi için all() metodu kullanılırken, filtreleyerek veri seçme işlemi için "filter()" metodu kullanılır. "filter()" metodu sonuç olarak bir sorgu seti geri döndürmektedir. Eğer tek bir kayıt çekmek istiyorsak "get()" metodu kullanmamız gerekmektedir. İsterseniz bu noktada bir örnek verelim.

```
def sec1(request):
    ogrenciler=Ogrenci.objects.filter(ad="Ali")
def sec2(request):
    ogrenciler=Ogrenci.objects.filter(ad="Ali", soyad="Can")
```

Örnekte görüldüğü üzere ilk fonksiyon ismi "Ali" olan tüm öğrencileri ogrenciler isimli değişkene atarken, ikinci fonksiyon ise ismi "Ali" ve soyadı "Can" olan öğrencileri filtreleyerek ogrenciler değişkenine atmaktadır. Bu fonksiyonların SQL karşılıklarını şu şekilde gösterebiliriz.

```
SELECT * FROM ogrenci
WHERE ad='Ali'

SELECT * FROM ogrenci
WHERE ad='Ali' and soyad='Can'
```

Eğer sorgumuz tek bir kayıt döndürecekse *get()* metodunu kullanabiliriz.

```
def sec3(request):
    try:
    ogrenci=Ogrenci.objects.get(ad="Ali")
    except ogrenci.DoesNotExists:
    print("Sonuç Bulunamadı.")
```

Bu kodda kullandığımız try-except bloğu, sorgumuz sonucunda hiçbir kayıt dönmezse çalışacaktır. Filtreleme sırasında birçok sorgulama seçeneğimiz bulunmaktadır. Aşağıdaki tabloda bu seçenekler verilmiştir.

Tablo 7.5 Filtreleme Seçenekleri

İsim	Açıklama	Kullanımı	SQL karşılığı
exact	Tam bir eşleştirme yapar.	Ogrenci.objects.filter(ad__exact='Ali')	SELECT * FROM ogrenci WHERE ad='Ali'
iexact	Harf duyarsız	Ogrenci.objects.filter(ad__iexact='ali')	
contains	Eşleştirme işleminde içermeyi göz önünde bulundurur.	Ogrenci.objects.filter(ad__contains='a')	SELECT * FROM ogrenci WHERE ad LIKE '%a%'
icontains	Harf duyarsız	Ogrenci.objects.filter(ad__icontains='a')	
startswith / endswith	Başı/sonu girilen değer ile eşleşen verileri getirir.	Ogrenci.objects.filter(ad_startswith='a') Ogrenci.objects.filter(ad_endswith='a')	SELECT * FROM ogrenci WHERE ad LIKE 'a%'
istartswith / iendswith	Harf duyarsız	Ogrenci.objects.filter(ad_istartswith='a') Ogrenci.objects.filter(ad_iendswith='a')	
in	Liste içerisindeki verilerle eşleştirir	Ogrenci.objects.filter(ad__in=['ali','veli'])	SELECT * FROM ogrenci WHERE ad in ('ali','veli')
gt / lt	Belirtilen değerden büyük/küçük eşleştirmeleri getirir.	Ogrenci.objects.filter(id__gt=3) Ogrenci.objects.filter(id__lt=3)	SELECT * FROM ogrenci WHERE id>3
gte / lte	Büyük/küçük ve eşit eşleştirmeleri getirir.	Ogrenci.objects.filter(id__gte=3) Ogrenci.objects.filter(id__lte=3)	SELECT * FROM ogrenci WHERE id>=3
range	İstenilen aralıktaki verileri getirir.	Ogrenci.objects.filter(id__range=(3,5))	SELECT * FROM ogrenci WHERE id between (3,5)

Sorgularımız sonucunda dönen satırları sıralamak istersek ise order_by() metodu kullanılabilir.

```
def sec4(request):
    ogrenciler=Ogrenci.objects.filter(ad__contains="a").order_by("soyad")
```

Bu şekilde sorgu sonucu dönen satırlar soyad kolonuna göre artan sıralamaya sahip olacaktır. Azalan sıralama için fonksiyona parametreyi "-soyad" şeklinde gönderebiliriz. Sorulamalarımızda ayrıca, sayma (count), ortalama (avg), maksimum (max), minimum (min), toplam (sum) gibi kümeleme fonksiyonları da kullanabiliriz. Bu fonksiyonların genel kullanımı şu şekildedir:

```
[Model Adi].objects.all().count()
[Model Adi].objects.all().aggregate(Avg('[ortalaması alınacak kolon adi]'))
[Model Adi].objects.all().aggregate(sum('[toplamı alınacak kolon adi]'))
[Model Adi].objects.all().aggregate(min('[kolon adi]'))
[Model Adi].objects.all().aggregate(max('[kolon adi]'))
```

Yukarıdaki ifadeler tüm kolonlar üzerinde işlem yapmaktadır. İstenirseniz "all()" metodu yerine "filter()" kullanılarak filtreleme sonrası elde edilen veriler üzerinde de işlemleri gerçekleştirilebilirsiniz. Modeller üzerinde filter() metodunun yanısıra eğer istenirse SQL cümleleri kullanılarak da işlemler gerçekleştirilebiliriz.

```
ogrenciler=Ogrenci.objects.raw('select * from ogrenci where ad=%s'%('Ali'))
```

Veri Güncelleme

Veri güncelleme işlemi kaydetme işlemi ile benzerlik göstermektedir. Tek farkı güncellemek istediğimiz satırın seçilmesidir.

```
ogrenci=Ogrenci.objects.get(ad="Ali")
ogrenci.soyad="Tek"
ogrenci.save()
```

Bu şekilde "Ali" ismindeki öğrencinin soyadını değiştirmiş olduk. İstersek bu kodu şu şekilde de yazabiliriz.

```
Ogrenci.objects.filter(ad="Ali").update(soyad="Tek")
```

"update()" fonksiyonu sorgulama kümesi üzerinden çalışan bir fonksiyondur. Yani, birden fazla satır üzerinde değişiklik gerçekleştirebilir.

Veri Silme

Veri silme işlemi için "delete()" metodu kullanılmaktadır. Örneğin aşağıdaki kod ismi "Ali" olan öğrencileri silecektir.

```
ogrenciler=Ogrenci.objects.filter(ad="Ali")
ogrenciler.delete()
```

İki satırdan oluşan bu kodu tek satırda yazmak da mümkündür. `ogrenciler=Ogrenci.objects.filter(ad="Ali").delete()`.Eğer tüm tabloyu silmek istersek ise `ogrenciler=Ogrenci.objects.all().delete()` yazmamız yeterlidir.

Öğrenme Çıktısı

7 Django ile model oluşturabilme
8 Django ile modeller üzerinde işlem yapabilme

Araştır 5

ER diyagramı verilen veritabanı yapısının modelini oluşturun.

İlişkilendir

Django'da veritabanı ilişkilendirme süreçlerini PHP çatılarında veritabanı ilişkilendirme süreçlerini karşılaştırınız.

Anlat/Paylaş

Django'da Web formu oluşturma ve kullanım süreçlerini, veri ekleme, veri silme ve veri güncelleme seçeneklerini anlatınız.

1 Python Dilini Kurabilme

Python Programlama Dili

1 Python dili ile kod yazmanın ilk adımı gerekli yazılımların bilgisayara kurulabilmesidir. Python, farklı işletim sistemleri üzerinde çalışabilen bir dildir. Bu nedenle çalışma ortamının kurulumu işletim sistemine göre farklılıklar göstermektedir. Bölümde Python kurulumu sadece Windows işletim sistemi için anlatılmıştır. Windows işletim sistemi üzerinde Python kurulumu görselleştirilmiş arayüzü sayesinde oldukça basittir.

2 Python dilinde program denetimi yapılarını açıklayabilme

Python Program Denetimi

2 Program denetimi, şartlara ve girdilere bağlı olarak program akışını düzenleme işidir. Program denetimi öğelerinin kullanımını bilmeden kodlama yapmak söz konusu değildir. Akış denetimi ve döngüler bir programcının bilmesi gereken temel unsurlardır. Python da diğer diller gibi, if, else, elif gibi kontrol yapılarına, for, while gibi döngü yapılarına, pass, break, continue gibi döngü çıkış elemanlarına sahiptir.

öğrenme çıktıları ve bölüm özeti

3 Python dilinde nesneleri kullanabilme

4 Python dilinde tkinter kullanarak masaüstü görsel programlama yapabilme

Python ile Nesneye Yönelik Programlama

3 Python, nesneye yönelik bir programlama dilinin tüm unsurlarını içermektedir (Bunları kullanmayı mecbur kılmaz). Python içerisinde yer alan tüm kütüphaneler nesneye yönelik programlama felsefesi ile yazılmıştır. Bu nedenle python dili ile kodlama yapacaksanız python ile nesne kullanımını iyi bilmeniz gerekmektedir. Sınıf, nesnelerin özelliklerini, davranışlarını ve başlangıç durumlarını tanımlamak için kullanılan şablonlar olarak tanımlanırken, nesneleri ise sınıflardan türetilmiş örnekler olarak tanımlayabiliriz. Yaklaşım olarak tüm dillerde bu tanımlar aynı olmasına rağmen, farklı dillerde sözdizimler farklılaşmaktadır. Nesneye yönelik yaklaşımda, kodlarımızı nesne veya sınıf üzerinden çağıracağımız metotlar içerisine yazar ve bunlar üzerinden tetikleriz. Buna ek olarak, tanımladığımız sınıflar, metotlar ve değişkenler için bir erişim belirleyicisi tanımlayabiliriz. Python'da erişim belirleyicileri: public, private ve semi-private olarak belirlenmektedir. Nesne Yönelimli Programlama dillerindeki en temel prensiplerinden olan Miras kavramı da python tarafından desteklenmektedir. Miras kavramı, bir sınıfın özelliklerinin ve metotlarının başka sınıflara aktarılmasına dayanmaktadır.

4 Python dili için yazılmış masaüstü görsel programlama yapmamızı sağlayan PyQti PyGI, PyGtk, Kivy vb. birçok paket bulunmaktadır. Bölümde, python kurulum dosyaları ile beraber gelen tkinter paketi incelenmiştir. Tkinter paketi, görsel programlamanın label, button, entry vb. birçok üyesini kullanmamıza imkân vermektedir. Ayrıca, tkinter nesnelerimizin yerleştirilmesinde pack, grid ve place gibi 3 farklı yaklaşım sunmaktadır.

5 Django web çatısını kurabilme

6 Django şablon kullanabilme

Django Web Çatısı

5 Django, Python programlama diliyle yazılmış bir web sayfası geliştirme çatısı (web framework) olarak tanımlanmaktadır. Django kurulumu oldukça basit olmasına rağmen dikkatli olmamızı gerektirir. Django kurulumun ilk adımı python kurulumudur. Django kurulumu için herhangi bir görsel arayüz bulunmamaktadır. Kurulum, komut istemcisi üzerinden gerçekleştirilmektedir.

6 Şablonlar, web sayfalarında tasarım ile kodu ayırmamızı sağlayan yapılardır. Şablonlar yardımı ile tasarımda veya kodda yapılacak bir değişiklik diğerini etkilememesini sağlayabiliriz. Bu, işlemleri paralel olarak gerçekleştirmemize yardımcı olur.

7 Django ile model oluşturabilme

8 Django ile modeller üzerinde işlem yapabilme

Django Modeller

7 Günümüzde, web sitelerinin birçoğu kayıtlarını saklayabilmek için en az bir veritabanı ihtiyaç duymaktadır. Web sitesinin veritabanını gereksinimini karşılamak için tüm gerekli yapılar modeller sayesinde Django›da desteklenmektedir. Modeller üzerinden kullanılan metotlar ile SQL komutlarına gerek kalmadan veritabanı işlemlerini gerçekleştirebiliriz.

8 Django'da modelleri tanımlayarak ve kullanarak, veri listeleme, veri ekleme, farklı filtreler uygulayarak veri seçme, veri güncelleme ve veri silme işlemlerini kolaylıkla gerçekleştirebiliriz. Bu işlemeleri gerçekleştirmek için tanımladığımız modeller üzerinden birtakım fonksiyonları çağırmak yeterli olacaktır.

neler öğrendik?

1 print(*“TBMM”, sep=”.”) fonksiyonunun ekran çıktısı aşağıdakilerden hangisidir?

A. T.B.M.M.
B. TBMM
C. T.B.M.M
D. .TBMM.
E. unexpected EOF while parsing

2 Aşağıdaki seçeneklerden hangisinde değer ataması doğru yapılmıştır?

A. isim=ayşe
B. isim:= “ayşe”
C. isim= “ayşe”
D. isim: “ayşe”
E. isim:= ayşe

3 Liste=[“ali”, “ayşe”, “fatma”, “mehmet”] şeklindeki bir liste düşünüldüğünde print (Liste[-1]) fonksiyonunun çıktısı aşağıdakilerden hangisidir?

A. ali
B. ayşe
C. fatma
D. mehmet
E. name ‘Liste’ is not defined

4 Python’da fonksiyon tanımlama için aşağıdakilerden hangisi kullanılmaktadır?

A. defination
B. def
C. function
D. procedure
E. özel bir saklı sözcük yoktur.

5 Aşağıdakilerden hangisi 10’dan 1’e kadar 2’şer 2’şer ilerleyen döngü oluşturan kod parçasıdır?

A. for i in range(1,10)
B. for i=10 down to 1 by 2
C. for (i=10;i>1;i=-2)
D. for i in range (10,1,-2)
E. for i in range (1,10, -2)

6 Sınıftan türetilecek nesnelerin değişken tanımlarını hangi metot içerisinde gerçekleştirilir?

A. __init__
B. constructor
C. nesne ismi ile aynı isme sahip metot
D. launch
E. starter

7 Aşağıdakilerden hangisi tkinter paketine ait nesne yerleştiricisidir?

A. locater
B. cell
C. inserter
D. placer
E. grid

8 Django’da proje ile bağlantı kurulacak sayfaların tanımlamaları hangi dosyada yapılmaktadır?

A. views.py
B. urls.py
C. settings.py
D. wsgi.py
E. manage.py

9 Django’da kod ile tasarımı birbirinden ayrılmasını sağlayan yapılara ne isim verilir?

A. Model
B. Web çatısı
C. Şablon
D. Etiketler
E. Filtreler

10 Django’da veritabanı erişimi için kullanılan yapıya verilen isim aşağıdakilerden hangisidir?

A. Render
B. Tkinter
C. Şablon
D. Form
E. Model

1. C	Yanıtınız yanlış ise "Python Programlama Dilinde Hazır Fonksiyonlar" konusunu yeniden gözden geçiriniz.	6. A	Yanıtınız yanlış ise "Python ile Nesneye Yönelik Programlama" konusunu yeniden gözden geçiriniz.
2. C	Yanıtınız yanlış ise "Python Programlama Dilinde Değişkenler" konusunu yeniden gözden geçiriniz.	7. E	Yanıtınız yanlış ise "Tkinter" konusunu yeniden gözden geçiriniz.
3. D	Yanıtınız yanlış ise "Python Veri Tipleri" konusunu yeniden gözden geçiriniz.	8. B	Yanıtınız yanlış ise "Django Web Çatısı, Proje Oluşturma" konusunu yeniden gözden geçiriniz.
4. B	Yanıtınız yanlış ise "Python Programlama Dilinde Fonksiyonlar" konusunu yeniden gözden geçiriniz.	9. C	Yanıtınız yanlış ise "Django Web Çatısı, Şablon Kullanımı" konusunu yeniden gözden geçiriniz.
5. D	Yanıtınız yanlış ise "Python Program Denetimi" konusunu yeniden gözden geçiriniz.	10. E	Yanıtınız yanlış ise "Django Web Çatısı, Modeller" konusunu yeniden gözden geçiriniz.

7 Araştır Yanıt Anahtarı

Araştır 1

```
>>>liste=[]
>>>liste.append("ali")
>>>liste.append("ayşe")
>>>liste.append("zeynep")
>>>liste.append("murat")
>>>liste.append("kerem")
>>>liste.append("veli")
>>>liste.append("gül")
>>>liste.sort()
>>>print (liste)
['ali', 'ayşe', gül, 'kerem', 'murat', 'veli', 'zeynep']

>>>Sozluk={}
>>>sozluk["book"]="kitap"
>>>sozluk["car"]="araba"
>>>sozluk["house"]="ev"
>>>sozluk["computer"]="bilgisayar"
>>>sozluk["phone"]="telefon"
>>>sozluk["box"]="kutu"
>>>sozluk["bottle"]="şişe"
>>>sozluk["chair"]="sandalye"
>>>sozluk["fish"]="balık"
>>>sozluk["monkey"]="maymun"
>>>print (sozluk)
```

7 Araştır Yanıt Anahtarı

Araştır 2

```
dersNotu=input("Notu Giriniz")
n=int(dersNotu)
if n>87:
    print("AA")
elif n>80:
    print("BA")
elif n>73:
    print("BB")
elif n>66:
    print("CB")
elif n>59:
    print("CC")
elif n>52:
    print("DC")
elif n>45:
    print("DD")
elif n>38:
    print("FD")
else:
    print("FF")
```

Araştır 3

```
from math import sqrt
class Nokta:
    def __init__(self, x, y):
        self.x=x
        self.y=y
    def noktaYazdir(self):
        print("x koordinatı: ",self.x," y koordinatı: ",self.y)
class Dogru:
    def __init__(self, x1, y1, x2, y2):
        self.n1=Nokta(x1, y1)
        self.n2=Nokta(x2, y2)
    def dogruUzunluk(self):
        return sqrt(((self.n1.x-self.n2.x)**2)+((self.n1.y-self.n2.y)**2))
    def ortaNokta(self):
        return Nokta((self.n1.x+self.n2.x)/2,(self.n1.y+self.n2.y)/2 )
    def dogruYazdir(self):
        print(" ilk nokta x koordinatı:      ",self.n1.x," y koordinatı: ",self.n1.y,
              "\n ikinci nokta x koordinatı: ",self.n2.x," y koordinatı: ",self.n2.y)
```

7 Araştır Yanıt Anahtarı

Araştır 4

```
<!DOCTYPE html>
<html>
<head>
    <title>Örnek Uygulama</title>
</head>
<body>
    {% if error %}
        <p style="color: red;">Lütfen bir ifade gi-
riniz.</p>
    {% endif %}
<form action="/goster/" method="post">
    {% csrf_token %}
    İfade : <input type="text" name="ifade" /><br>
    Font büyüklüğü : <select name="buyukluk" />
            <option>1</option>
            <option>2</option>
            <option>3</option>
            <option>4</option>
            <option>5</option>
            <option>6</option>
            <option>7</option>
            </select>
    <input type="submit" value="Goster" />
</form>
```

urls.py dosyası

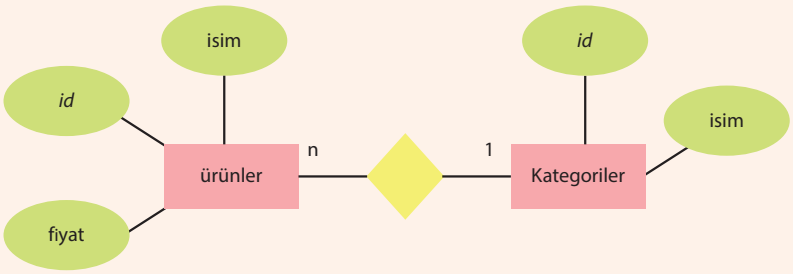
```
url(r'^ornek/',views.ilk_form),
url(r'^sonuc/',views.sonuc_form),
```

Views.py dosyası

```
def ilk_form(request):
    return render(request, 'ornek.html')

def sonuc_form(request):
  if 'ifade' in request.POST and request.POST['ifade']:
        isim = request.POST['ifade']
        buyukluk  = request.POST['buyukluk']
    return    HttpResponse('<font    size=%s    >$s</
font>'%(ifade),%(buyukluk))
    else:
       return render(request, 'ornek.html', {'error':
True})
```

7 Araştır Yanıt Anahtarı

Araştır 5




```
from django.db import models

class Kategoriler(models.Model):
    isim = models.CharField(max_length=100)

class Urunler(models.Model):
    isim = models.CharField(max_length=40)
    fiyat = models.IntegerField()
    kategori = models.ForeignKey(Kategoriler)
```

Kaynakça

Aksoy A., (2016). Yeni Başlayanlar İçin Python, ISBN: 978-605-912-950-3, Abaküs yayıncılık

Başer, M., (2014). Python, 3. Baskı, ISBN: 978-605-87588-7-2, Dikeyeksen Yayıncılık.

Karaçay, T., (2014). Python 3: Veri Yapıları. 2. Baskı, ISBN: 978-975-023-093-6 Seçkin Yayıncılık.

Özgül F., (2013). Her yönü ile Python, ISBN: 978-605-5201-265 Kodlab Yayıncılık.

Summerfield, M. (2011). Programming in Python 3, 2. baskı. ISBN: 978-032-168-056-3. Addison-Wesley

internet kaynakları

http://djangobook.com/

https://www.djangoproject.com/

https://pythonprogramming.net/

http://www.pythontutor.com

https://www.python.org/

https://docs.python.org

Bölüm 8

Ruby ve Ruby On Rails

öğrenme çıktıları

1 Ruby Nedir ve Ne İşe Yarar?

1 Ruby dilinin geliştirilme amaçlarını, temel felsefi prensiplerini de göz önüne alarak açıklayabilme

2 Ruby Dili Söz Dizimi ve Bileşenleri

2 Ruby dilinin temel söz dizimi kuralları ve kullandığı yapıları açıklayabilme

3 Ruby On Rails Çatısı

3 Ruby on Rails çatısının temel çalışma prensiplerini açıklayabilme

4 Ruby On Rails ile Uygulama Geliştirme

4 Ruby dilinin temel teknik özelliklerini göz önüne alarak ne tür uygulamalarda kullanılabileceğini ifade edebilme

Anahtar Sözcükler: • Ruby • Ruby On Rails • Web Uygulama Geliştirme Çatısı • Kendi Kendini Tekrarlama (DRY) • Konfigürasyon Üzerine Kabuller (CoC)

GİRİŞ

Teknolojinin hızına yetişmemizin neredeyse imkânsız olduğu günümüzde, bu duruma paralel olarak gün geçmiyor ki yeni bir programlama dili ortaya çıkmasın. Bu yeni yetme dillerin bir kısmı gün yüzü göremeden teknoloji çöplüğünü boyluyor olsa da, aralarından bazıları bu çetin mücadeleden galip çıkarak yazılım geliştiricilerin depolama birimlerinde kendilerine yer edinip, yeni yazılımların geliştirilmesi sürecine dahil olabiliyorlar. Belli bir dilde uzmanlaşmasının oldukça fazla zaman gerektiriyor olması, her programlama dilinin genelde bir alana veya metodolojiye odaklanıyor olması, farklı platformlar için farklı dillere ihtiyaç duyulması gibi pek çok etkenin rol oynadığı bir ortamda; bir programcının bir dilden diğerine geçmeyi tercih etmesi, diğer bir deyişle alışkanlıklarının bir kısmını kökünden değiştirmesi için bu yeni programlama dilinin kullanmakta olduğu dile oranla çok daha üstün özellikler vaat ediyor olmasının gerektiği de aşikardır.

Ruby yüksek okunabilirliğe sahip olması, esnek kod yazmaya imkân tanıması, kolaylıkla genişletilebilmesi, platform bağımsız olması gibi pek çok nedenden ötürü günümüzün popüler dilleri arasında yer almaktadır. Tüm bunların yanı sıra Ruby dilinin popülerliğini kazanmasındaki asıl itici gücün Ruby on Rails çatısı olduğu da su götürmez bir gerçektir. 2000'li yılların başında diğer dillerle oluşturulması günler alabilecek bir uygulamanın, Ruby on Rails ile sadece on beş dakika içerisinde tamamlanabiliyor olması pek çok yazılım geliştiricinin bu dile geçiş yapmasına neden olmuştur. Ruby, günümüzde dünyanın en çok kazandıran ve bilişim dünyasının kalbi olan Silikon Vadisindeki yazılım geliştiriciler tarafından en çok tercih edilen programlama dillerinin başında gelmektedir.

RUBY NEDİR VE NE İŞE YARAR?

Ruby, Japon bilgisayar bilimcisi *Yukihiro "Matz" Matsumoto* tarafından geliştirilmiş bir programlama dilidir. Basitlik ve verimlilik dengesi üzerine inşa edilmiş olan tamamen nesne yönelimli bir dildir. *Perl, Pyhon, SmallTalk, Ada* gibi pek çok programlama dilinden esinlenmiş olan Ruby, aynı zamanda *Rust, Swift, Clojure, Crystal* gibi birçok programlama diline de esin kaynağı olmuştur.

1993 yılında geliştirilmeye başlanmış ve 1995 yılının sonlarında ilk versiyonu olan Ruby 0.95 yerel bir Japon haber grubunda duyurularak kullanıma sunulmuştur. 2000 yılına kadar Japonya sınırlarının dışında pek ses getirememiş olsa da; bu yılda İngilizce olarak ilk kitabın piyasaya çıkması, haber grupları ve e-posta listeleri kurulması gibi etkenlerin de yardımıyla dünya çapında ilgi uyandırmaya başlamıştır. Ruby dilinin asıl manada dünya çapında kullanılmaya başlanması ise Ruby on Rails isimli web çerçevesi ile birlikte olmuştur. Bu ünitenin yazımı sırasında Ruby'nin kullanılabilir durumda olan en son kararlı hali 2.3.1 sürümüdür.

Daha önce bahsettiğimiz gibi, Ruby dilinin geliştirilmesi aşamasında pek çok programlama dilinden esinlenilmiştir. Ruby, gerek bu dillerin bazılarının doğasında var olan, gerekse Matz tarafından adapte edilmiş olan pek çok felsefi temel üzerine inşa edilmiştir. Bunları birkaç madde ile şöyle özetleyebiliriz:

- *İnsan Faktörü Prensibi*: Matz'in diğer dillerle ilgili en çok yakındığı nokta: bu dilleri kullanan programcıların, özellikle bilgisayar mühendislerinin, program yazarken işlevsellik yerine makinenin daha hızlı ve verimli çalışmasına odaklanmalarıydı. Bunu yaparken insan faktörü sürekli göz ardı ediliyordu. Halbuki bizler, bilgisayar programlarının büyük bir bölümünü insanlarla etkileşim halinde olması için tasarlamaktayız. Bu nedenle programlama dilleri makineden çok insan faktörü düşünülerek tasarlanmalıdır. Matz bunu şu ifadeyle açıkça dile getirmektedir: "*Onlar (bilgisayarlar) köle, bizler efendiyiz.*"
- *Asgari Sürpriz Prensibi*: Bilgisayar bilimleri haricinde başka alanlarda da uygulaması bulunan bu prensip gereği bir işlemin sonucu tahmin edilebilir, açık ve tutarlı olmalıdır. Matz bu durumu birkaç cümlede şöyle ifade etmektedir: "*İnsanlar program yazarken programlama dili ile savaşmak yerine kendilerini ifade etmek isterler. Bu nedenle programlama dilleri, programcılara doğal görünmelidir. Ben, Ruby'i tasarlarken sürprizleri en aza indirgemeye çalıştım ve bundaki amacım program yazarken harcanan eforu en aza indirgemekti.*".

- *Özgürlük ve Rahatlık Prensibi*: Yazılım geliştiricilerin geçmişte farklı programlama dillerini kullanmış olmaları, farklı metodolojileri benimsemeleri ve hatta farklı kültürlerden gelmeleri gayet olağan bir durumdur. Tüm bunlardan dolayı her programcının kendine has sorun çözme yöntemlerinin olması da kaçınılmazdır. Bu durumu göz önünde bulunduran Ruby, programcıların kendilerini özgür ve rahat hissetmeleri adına çoğu zaman alternatif yollar sunar.

> Python gibi diller bu prensibin aksine minimal felsefeyi benimseyecek şekilde tasarlanmıştır ki: bu dillerde bir işi yapmanın sadece ve sadece tek bir yolu bulunmaktadır. Aslında minimal felsefenin kullanımı herkesin aynı kodlama stilini kullanıyor olması nedeniyle kodun daha okunabilir hale gelmesini sağlasa da; Ruby bu noktada özgürlük ve rahatlığı tercih eden konumunda yer almayı seçmiş olup, kod okunabilirliğini arttırmak amacıyla daha farklı yöntemler kullanmıştır.

- *Maksimum Eğlence Prensibi*: Program yazmanın genelde zor ve sıkıcı bir iş olduğu söylenir. Halbuki hepimizin diğer hayat amaçlarının yanı sıra eğlenmeye de ihtiyacının olduğunu söyleyebiliriz. Ruby programcının sıkıcı işlemler ile uğraşmak yerine sadece o anda yapılması gereken asıl işe odaklanabilmesine imkan tanır. Diğer dillerde satırlarca ifade ile yaptığınız bir işlemi Ruby ile bir veya iki satırda halledebilirsiniz.

Ruby Dilinin Temel Özellikleri

Ruby dilinin söz dizimi ve diğer yapılarından bahsetmeden önce sahip olduğu temel özelliklerden kısaca bahsetmek yerinde olacaktır. Bu özellikleri şöyle sıralayabiliriz:

- *Tamamen nesne yönelimli bir dildir*: Ruby, tamamen nesne yönelimli bir dil olarak tasarlanmıştır. Ruby içerisinde her şey bir nesnedir ve kendisine ait özellikleri ve metotları bulunmaktadır. Programcının fazladan bir şeyler yapmasına gerek duyulmadan "sarmalama" ilkesinin gereği olarak nesnelere ait özelliklerin ve metotların dış ortamdan gizlenmesine imkân tanır. Programcıya sadece ihtiyaç duyduğu özelliklere erişmek için gerekli metotları etkinleştirmek kalır. "Kalıtım" ilkesinin gereği olarak bir sınıfın başka bir sınıfın özelliklerini ve metotlarını kullanmasına izin verir. Yine bir başka nesne yönelimli programlama özelliği olan "çok biçimlilik" ilkesini de başarıyla uygulayabilen dillerden birisidir.
- *Çok sayıda programlama paradigmasını destekler*: Ruby nesne yönelimli özelliklerinin yanında fonksiyonel, zorunlu (imperatif) ve yansıtıcı (reaktif) programlama paradigmalarını da destekleyen bir yapıya sahiptir.
- *Yorumlanan bir dildir*: Yorumlanan programlama dillerinde yazılmış olan bir kodun çalıştırılması bilgisayarda yer alan yorumlayıcı adında bir yazılım sayesinde gerçekleştirilir. Bu yazılım derlenen dillerin aksine kodumuzun tamamını tek seferde makine diline dönüştürmek yerine; bu dönüştürme işlemini satır satır gerçekleştirmektedir. Yine derlenen dillerde kodumuzda yapacağımız en ufak bir değişiklik bütün program kodunun tekrar derlenmesi anlamına gelmektedir ki bu da geliştirme sürecini oldukça yavaşlatmaktadır. Yorumlanan dillerde ise bu durumun aksine yapılan değişiklik herhangi bir zaman kaybına yol açmaz.
- *Platform bağımsız bir dildir*: Yorumlanan dillerin en büyük artılarından birisi de makine kodunun çalışma zamanında oluşturulması sayesinde herhangi bir platforma bağlı kalma zorunluluğunun ortadan kalkmasıdır. Yani Ruby hemen hemen tüm platformlarda sorunsuz çalışabilmektedir. Derlenen dillerde ise kodun derlenmesi için hangi platforma ilişkin derleyici yazılımı kullanıldıysa o platforma bağımlı kalınmalıdır. Aynı yazılım farklı platformlarda çalıştırılmak istenirse kodun o platforma ilişkin bir derleyici yazılımı kullanılarak yeniden derlenmesi gerekecektir. Ruby, aynı zamanda işletim sisteminden bağımsız olarak sistem çağrıları gönderebilme yeteneğine de sahiptir. Bu yeteneği sayesinde sistem yönetimi araçlarının geliştirilmesi için de tercih edilebilecek diller arasında yer almaktadır.

Derlenen dillerin çıktısı makine kodu olduğu için çalıştırılabilir kodun yorumlanan dillere göre bir miktar daha performanslı olduğu söylenebilir. Ancak günümüzde çok gelişmiş mikroişlemciler kullanıldığından bu performans artışından çok büyük bir kazanç olarak bahsedemeyiz. Ayrıca burada bahsedilen Ruby'nin standart yorumlayıcısı olan MRI yani "Matz'in Ruby Yorumlayıcısı"dır. Halihazırda MRI'ya alternatif olarak geliştirilen farklı Ruby uyarlamaları da bulunmaktadır: Rubinius, JRuby, RubyMotion, IronRuby gibi…

- *Dinamik ve güçlü tiplemeli bir dildir:* Ruby çalışma zamanında değişkenlerin tipini kendi başına belirleyebilmektedir. Böylece herhangi bir tip tanımlamasına ihtiyaç duyulmamaktadır. Ayrıca programın çalışması sırasında bir değişkene atanan değer veya bir metodun geriye döndürdüğü değer gibi tipler de değiştirilebilir. Ruby veri türleri konusundaki bu üstün esnekliğinin yanında hatalı kullanımın da önüne geçebilmek adına güçlü tipleme sistemi kullanmaktadır. Bu sayede birbiriyle işleme sokulamayacak olan bazı tipler arasında işlem yapılması gibi hataların oluşması da engellenmektedir.

Saydıklarımızın haricinde özetlemek gerekirse Ruby dilinin diğer özelliklerini de şöyle sıralayabiliriz:

- Temiz ve test edilebilir kod yazmayı destekler.
- Meta-programlama sayesinde çalışma zamanında kod üretimine (kod yazan kodlar oluşturmaya) imkân tanır.
- Alana özel dil (DSL) oluşturabilme kabiliyeti sayesinde Ruby içerisinde farklı söz dizimi kullanan apayrı bir dil oluşturabiliriz. Ünitenin ilerleyen bölümünde bahsedeceğimiz Ruby on Rails web çatısı bunun uygulamadaki en iyi örneklerinden birisidir.

internet

Ruby topluluklarına ilişkin bilgi edinmek ve katılım sağlamak amacıyla https://www.ruby-lang.org/tr/community/ adresine başvurabilirsiniz.

Ruby birbirinden farklı pek çok alan için en uygun dillerden birisidir. Ruby gerçek dünyada: internet tabanlı uygulamalar, yapay zeka ve keşifsel matematik, CGI programlama, GUI (Grafiksel Arayüz) uygulamaları, ağ ve telekomünikasyon uygulamaları, robotik, sistem programlama, güvenlik uygulamaları, metin işleme yazılımları, genel programlama ve programlama eğitimleri gibi alanlarda başarıyla kullanılmaktadır. Bu alanlarda geliştirilen uygulamalara birkaç örnek vermek gerekirse: NASA simülasyon uygulamalarının yönetilmesi, Motorola simülatör uygulamasının geliştirilmesi, Google SketchUp 3D modelleme uygulamasının makro geliştirme API'si geliştirilmesi, MORPHA projesinde, Siemens servis robotunun reaktif denetim kısmının uygulaması ve Open Domain Server IP ayarlarının gerçek zamanlı olarak güncellenmesini ve böylece statik domainler ile eşlenebilmesini sağlayan "Dynamic DNS" istemcilerinin yazılması sırasında Ruby dilini kullanması örnek verilebilir.

internet

Ruby dilinin kullanıldığı başka gerçek dünya başarı öyküleri için https://www.ruby-lang.org/tr/documentation/success-stories/ adresi ziyaret edilebilir.

Öğrenme Çıktısı

1 Ruby dilinin geliştirilme amaçlarını, temel felsefi prensiplerini de göz önüne alarak açıklayabilme

Araştır 1

Size göre Ruby programlama dilinin temel felsefi özellikleri göz önüne alındığında başarısının arkasında yatan asıl neden ne olabilir?

İlişkilendir

Japonya'da geliştirilen Ruby'nin dünya çapında nasıl yayıldığını Ruby on Rails ile ilişkilendirerek açıklayınız.

Anlat/Paylaş

Birçok programlama dilinden esinlenen ve birçoğuna da esin kaynağı olan Ruby'nin felsefi temellerini tartışınız.

RUBY DİLİ SÖZ DİZİMİ VE BİLEŞENLERİ

Bu başlık altında Ruby dilinin söz dizimine ilişkin temel kurallardan ve dilin kendisini oluşturan yapılardan basitçe bahsedilecektir.

Ruby Kodunun Çalıştırılması

Ruby ile birlikte program yazabilmek için Ruby yorumlayıcısına ihtiyacımız olacaktır. Daha önce bahsettiğimiz gibi Ruby platform bağımsız bir dil olduğundan Windows, MAC OS, Linux gibi en çok tercih edilen işletim sistemlerinin yanı sıra NEWS-OS, SunOS, AIX, SVR4, Solaris, NEC UP-UX, NeXTSTEP gibi pek fazla bilinmeyen işletim sistemleri tarafından da çalıştırılabilmektedir. Ayrıca Windows Phone, Symbian OS gibi mobil platformlar için kullanılabilecek versiyonları da mevcuttur.

internet

Ruby kurulum işlemleri, kullanmakta olduğunuz işletim sistemine göre farklılık gösterdiğinden https://www.ruby-lang.org/tr/downloads/ adresini ziyaret ederek kendi işletim sisteminiz için neler yapmanız gerektiğini öğrenebilirsiniz.

Ruby kodunu çalıştırmak için komut satırı üzerinden Ruby Yorumlayıcısını (MRI) veya Etkileşimli Ruby (IRB) yazılımını kullanabiliriz. İlk olarak ruby yorumlayıcısını kullanarak programlarımızı nasıl çalıştırabileceğimizi inceleyeceğiz.

Programı komut satırında çalıştırmadan önce yazmış olduğumuz kodları içeren bir dosyaya ihtiyacımız olacaktır. Bu amaçla aşağıdaki kodu bir metin editörü yazılımı kullanarak *merhaba.rb* adıyla kaydediniz.

dikkat

".rb" uzantısı Ruby dosyalarının varsayılan uzantısıdır. Kullanılmaması bir sorun teşkil etmese de bilgisayarımızda yüklü programların bu dosyaları tanıyabilmesi için kullanılması önerilmektedir.

```
class Ornek
  def merhaba
    puts "Merhaba Dünya"
  end
end

ornek1 = Ornek.new
ornek1.merhaba
```

Komut satırında dosyayı kaydettiğimiz dizine geçtikten sonra aşağıdaki komutu vermemiz programın çalıştırılmasını sağlayacaktır

dikkat

$: karakterleri terminal komut satırının başlangıç bölümünü göstermek için kullanılmıştır, bu nedenle komutları çalıştırırken yazılmaması gereklidir. Ünitenin bundan sonraki sayfalarında da aynı amaçla kullanılacaktır.

```
$: ruby merhaba.rb
Merhaba Dünya
```

Eğer herhangi bir yazım hatası yapmadıysanız; yukarıdaki gibi ekranda "Merhaba Dünya" çıktısını görüyor olmalısınız. Programlarımızı çalıştırmanın diğer yolu ise yine Ruby kurulumuyla birlikte gelen *IRB* aracının kullanılmasıdır. Programımızda yer alan tüm ifadeler ekrana bir şeyler yazdırmasa bile geriye bir değer döndürmektedir. Ruby diline has bir özellik olan bu durum *IRB* kullanılarak test edilebilir, zaten bu aracın kullanım amacı program yazmaktan ziyade kod parçalarını test etmemize imkân tanımasıdır.

```
$: irb
irb(main):001:0> 3+5
=> 8
```

Görüldüğü gibi *IRB* yazılımını çalıştırmak için komut satırında adını yazmamız yeterlidir. Bu noktadan itibaren komut satırının başlangıcı değişecek ve bize *IRB* içerisinde olduğumuzu gösterecektir. Bu örnekte *IRB* bir hesap makinesi gibi kullanılmış ve bir toplama işlemi yapılmıştır. Bu işlemin hemen altındaki satırda yer alan => karakterleri de bu işlemden geriye dönen değeri göstermektedir.

Ruby Dili Sözdizimi

Ruby dilinin söz dizimine ilişkin temel kuralları şöyle özetleyebiliriz:

- Bir Ruby programı art arda gelen ifadelerden oluşur. İfadeleri birbirinden ayırmak için yeni satır veya noktalı virgül karakteri kullanılabilir. Bir ifade birden fazla satıra bölünmek istenirse; bir önceki satır +, - gibi sağ taraftaki operandı eksik bir metot ile ya da ters bölü (\) karakteri ile bitirilebilir.

> Söz dizimi Türk Dil Kurumu sözlüğünde "bir cümleyi oluşturan kelime türlerinin arasındaki ilişkileri inceleyen dil bilgisi kolu, cümle bilgisi, nahiv" şeklinde tanımlanmaktadır. Bir programlama dili için ise söz dizimi; o dilin yapıları kullanılarak geçerli bir ifadenin nasıl oluşturulabileceğine ilişkin konuları kapsamaktadır.

- Boşluk karakterleri (space, tab gibi) sadece metin türünden verilerin içerisinde yer aldıklarında önem arz ederler, bunun dışında Ruby bu karakterleri görmezden gelir.
- Diyez (#) karakteri ile başlayan metinler o satırın sonuna kadar, =*begin* ve =*end* ifadeleri arasında yer alan satırlar ise tamamıyla yorum satırlarıdır ve Ruby yorumlayıcısı tarafından görmezden gelinir.
- Değişken, metot veya sabitler gibi belirteçler büyük-küçük harf duyarlıdır. Yani; *AOF* ile *aof* tamamen farklı belirteçlerdir.
- Çeşitli amaçlarla Ruby dilinin kendisi tarafından kullanılan kelimelere *anahtar kelimeler* denir. Bu kelimeler değişken veya sabit isimleri olarak kullanılamaz.

> **dikkat**
> Ruby içerisinde yer alan anahtar kelimelerin bir listesine göz atmak için http://ruby-doc.org/core-2.3.1/doc/keywords_rdoc.html adresine başvurabilirsiniz.

- Parantez karakterlerinin kullanımı çoğu durumda isteğe bağlıdır. Bu özellik genellikle kodun daha doğal görünmesini sağlarken; bazı zorunlu durumlarda(örneğin aritmetiksel işlemlerdeki işlem önceliğinin ayarlanması gibi) ve karmaşıklık yaratabilecek hallerde kullanılması gereklidir.

Temel kurallara ilişkin kısa bir bilgilendirmeden sonra hızlıca Ruby dilinin içerisine bir göz atalım.

Değişkenler

Değişkenleri birtakım verileri isimlendirmek ve daha sonra bu isimle onlara erişmek için kullanmaktayız. Tüm veriler çeşitli veri tiplerinde (metin, sayı, vb.) bulunduklarından dolayı bazı programlama dillerinde bu verilerin hangi türden olduğunun belirtilmesi gereklidir. Ruby ise bu konuda oldukça esnektir; değişkenin türünü kendisi belirler ve onunla ilgili işlemleri kendisi yaparak bizi bu zahmetten kurtarır. Yine bazı dillerde değişkenler kullanılmadan önce tanımlanmalıdır. Ruby dilinde ise bir değişkene değer atandığı anda o değişken otomatik olarak tanımlanmaktadır. Bir örnek ile bunu şöyle gösterebiliriz:

```
plaka = 26
```

Bu atama işlemi sonucunda plaka isimli değişkene 26 değeri atanması sonucu bu değişken oto-

matik olarak oluşturulmuş ve ilgili hafıza bölgesine 26 değeri yerleştirilmiştir. Artık bu değere erişmek için değişkenin ismini yazmamız yeterli olacaktır. Bu değişkene istenildiği anda aynı ya da farklı türden bir değer atayabiliriz. Örneğin aşağıdaki atama işlemi sonucunda plaka değişkeni artık metin türünden bir değer taşıyacaktır:

```
plaka = "Eskişehir"
```

Değişkenler isimlendirilirken bazı kurallara dikkat etmemiz gerekmektedir. Değişken isimleri her zaman küçük harf (sabitler için büyük harf) veya _(alt tire) karakteri ile başlamalıdır. Sonrasında harf, alt tire veya rakam kullanılabilir. Değişken isimleri basılamayan karakterleri (boşluk, sekme karakterleri gibi.) içeremez. Değişken isimleri anahtar kelimeler olamaz.

Değişkenleri geçerli oldukları kapsama göre türlere ayırabiliriz. Ruby içerisindeki değişken türlerini şöyle gösterebiliriz:

- *Yerel Değişkenler*: Sadece tanımlandıkları blok içerisinde kullanılabilirler.
- *Global Değişkenler*: Tüm program içerisinde kullanılabilir durumda olan değişkenlerdir. Bu değişkenlerin isimlendirilmesine $ karakteri ile başlanır.
- *Örnek Değişkenleri*: Nesnelere ait özellikleri saklamak için kullanılır. Değişken ismi @ karakteri ile başlamalıdır.
- *Sınıf Değişkenleri*: Bir sınıf içerisinde veya bu sınıftan oluşturulan tüm nesneler tarafından erişilebilir olan değişkenlerdir. Sınıf değişkenleri gibi her nesne için farklı bir değer taşımazlar, sadece tek bir değerleri olabilir. Yazımına @@ karakterleri ile başlanmalıdır.
- *Sabitler*: Yerel değişkenler gibidirler ancak değerleri değiştirilmeye çalışıldığında bir uyarı alınmasına neden olurlar. Değeri program boyunca değişmeyecek olan veriler için kullanılmalıdırlar. İsimleri büyük harf ile başlamalıdır. Ruby'de bazı yerlerde (sınıf ismi, modül ismi gibi) sabitlerin kullanılması zorunludur.

Veri Türleri

Ruby dilinde kullanılan veri türlerini basit veri türleri (sayılar, metinler, semboller ve mantıksal) ve birleşik veri türleri (diziler ve sözlükler) olarak ikiye ayırabiliriz.

Sayılar: Ruby dilinde diğer pek çok dilde olduğu gibi temel olarak iki tür sayı bulunmaktadır: tamsayılar ve ondalık sayılar. Bu türlerin yanı sıra rasyonel sayılar, karmaşık sayılar gibi sıra dışı sayı türlerine de bütünleşik olarak destek sunulmaktadır.

Ruby, sayılar ile birlikte kullanabileceğimiz pek çok metot sunmaktadır. Bu metotların bir listesine IRB üzerinde "*methods*" metodunu kullanarak aşağıdaki gibi erişebiliriz.

```
irb(main):001:0> 12.methods
=> [:%, :&, :*, :+, :-, :/, :<, :>, :^, :|, :~, :-@, :**, :<=>, :<<, :>>, :<=, :>=,
:==, :===, :[], :inspect, :size, :succ, :to_s, :to_f, :div, :divmod, :fdiv, :modulo,
:abs, :magnitude, :zero?, :odd?, :even?, :bit_length, :to_int, :to_i, :next, :upto,
:chr, :ord, :integer?, :floor, :ceil, :round, :truncate, :downto, :times, :pred,
:to_r, :numerator, :denominator, :rationalize, …]
```

Metot isimlerinin başında yer alan ":" karakteri metot isimlerinin sembol olarak saklanmasından kaynaklanmaktadır. Tüm bu metotlardan burada bahsetmemiz mümkün değil ancak yine de birkaçına değinmeden geçmeyelim. Aşağıda bazı örnek kullanımlar gösterilmektedir:

```
# Mod alma
16 % 5            #=> 1
# Üs alma
3 ** 5            #=> 243
# Mutlak değer
-345.abs          #=> 345
# İki tamsayı arasında bölme işlemi yine tamsayı olacaktır. Eğer ondalık
# sonuç isteniyorsa sayılardan birisi ondalık olarak verilmelidir
```

```
21/4              #=> 5
21/4.0            #=> 5.25
# Normal yuvarlama, aşağı yuvarlama, yukarı yuvarlama
2.45.round        #=> 2
2.45.floor        #=> 2
2.45.ceil         #=> 3
# Tamsayılar Fixnum sınıfından, ondalık sayılar ise Float sınıfından
# oluşturulmuş nesnelerdir
2321.class        #=> Fixnum
23.21.class       #=> Float
```

Metinler: Metinler tek veya çift tırnak karakterleri arasına yazılmış karakter dizileridir: “Merhaba”, “213. Sokak”, ‘34+65’, “Anadolu Üniversitesi” gibi. Sayılarda olduğu gibi metinlerle birlikte de kullanabileceğimiz pek çok metot bulunmaktadır. Biz sadece bunlardan bazılarına değineceğiz.

Bir metin içerisindeki alt metne veya karaktere erişmek için [ ] metodunu kullanabiliriz. Metinler içerisindeki karakterler, metin içerisinde bulundukları yere göre bir indis numarasına sahiptirler. Örneğin “AOF” metninde “A” karakteri birinci sırada yer almaktadır. Ancak Ruby dilinde indisler 0 rakamından başladığı için sırasıyla bu karakterlerin indisleri şu şekilde olacaktır: A:0, O:1, F:2. Bu bilgiler ışığında aşağıdaki örneklere göz atabilirsiniz:

```
“Anadolu Üniversitesi”[3]        #=> “d”
# Negatif indis kullanımı: Son karakterden başa doğru -1, -2, -3, ...
“Anadolu Üniversitesi”[-2]       #=> “s”
# Belli bir karakter aralığı
# 8. karakterden başla, 3 karakter al
“Anadolu Üniversitesi”[8, 3]     #=> “Üni”
# 3. karakterden, 6. karaktere kadar al
“Anadolu Üniversitesi”[3..6]     #=> “dolu”
“Anadolu Üniversitesi”[3...7]    #=> “dolu”
```

Örnekteki .. ve ... karakterlerinin kullanımına aralık (range) adını vermekteyiz. Aralık belirtirken iki nokta kullandığımızda son karakter aralığa dahil olurken, üç nokta kullanımında son karakter aralığa dahil edilmemektedir.

Ruby’nin sağladığı nesne yönelimli programlama özelliklerinden birisinin de metotların aşırı yüklenmesi olduğundan bahsetmiştik. Buna örnek olarak sayılarla birlikte kullandığımız “+” ve “*” metotlarını gösterebiliriz. Toplama işlemi metinler ile birlikte kullanıldığında metinlerin peş peşe eklenmesini, çarpma işlemi ise metinlerin belirtilen adette tekrar edilmesini sağlar.

```
“Ruby” + “ on Rails”   #=> “Ruby on Rails”
“Ruby” * 3             #=> “RubyRubyRuby”
```

Metinlerin içerisinde değişkenlerin kullanılması da mümkündür. Bunun için yine “+” metodundan yararlanabileceğimiz gibi ilgili değişkenleri veya istenilen herhangi bir Ruby kodunu #{} karakterlerinin arasına yazabiliriz. Burada dikkat edilmesi gereken nokta ise metni oluştururken çift tırnak kullanılması gerekmesidir, tek tırnak kullanılması durumunda her şey metin olarak algılanacaktır.

```
“3 + 5 işleminin sonucu #{3+5}’dir.”
#=> “3 + 5 işleminin sonucu 8’dir.”

“#{8 * 5}, #{42/2}’den büyüktür.”
#=> “40, 21’den büyüktür.”
```

Terminal ekranına bir şeyler yazdırmak amacıyla *puts* veya *print* metodu kullanılabilir. *puts* metodu yazma işleminden sonra yeni satıra geçerken, *print* metodu aynı satırdan devam etmektedir.

Metinler içerisinde “, ‘, \ gibi özel karakterler kullanılmak istenirse; bu karakterler kaçış karakteri adı verilen \ karakterinden sonra yazılmalıdır.

```
puts "Atatürk'ün dediği gibi \"Yurtta sulh, cihanda sulh\"."
# Atatürk'ün dediği gibi "Yurtta sulh, cihanda sulh".
```

Metinlerle birlikte kullanabileceğimiz bazı metotlarla ilgili aşağıdaki örnekleri inceleyiniz.

```
# Tümünü büyük harfe dönüştür
"abc".upcase    #=> "ABC"
# Tümünü küçük harfe dönüştür
"DEF".downcase    #=> "def"
# Metni ters çevir
"abcdef".reverse    #=> "fedcba"
# İlk harfi büyüt, diğerlerini küçülт
"ABCdef".capitalize    #=> "Abcdef"
# Metindeki karakter sayısı
"kent park".length    #=> 9
```

Metinler üzerinde yapılan harf değiştirme işlemlerinde Türkçe dili için bütünleşik destek sunulması 2.4.0 versiyonunun vaat ettiği değişikliklerden birisidir. Önceki versiyonlar için Türkçe dil desteğine ihtiyaç duyarsanız https://github.com/sbagdat/turkish_support adresinden ilgili paketi yükleyebilirsiniz.

Semboller: Ruby içerisinde metinlere oldukça benzeyen sembol adında bir veri türü daha bulunmaktadır. Sembollerin metinlerden farkı bir kere oluşturulduklarında program boyunca aynı bellek bölgesini işgal etmeleridir. Oysa metinler aynı içeriğe sahip olsalar dahi her kullanıldıklarında tekrar oluşturulmaktadır. Bu da bellek kullanım miktarını ve performansı bir miktar düşürmektedir. Bu nedenle programımızda sık kullanacağımız metinleri sembol haline dönüştürmek yararlı olacaktır. Sembollerin oluşturulması da metinlere oldukça benzerdir; tek farkı sembollerin başında iki nokta üst üste “:” karakterinin yer almasıdır. Aşağıdaki örnekte görülebileceği gibi aynı içeriğe sahip metin nesneleri için farklı bellek alanı kullanılırken, semboller için bu durum söz konusu değildir.

```
"metin".object_id      #=> 70270242151520
"metin".object_id      #=> 70270242134400

:"metin".object_id     #=> 1144028
:metin.object_id       #=> 1144028
```

Mantıksal: Çoğu programlama dilinde olduğu gibi Ruby'de de mantıksal veri türleri yer almaktadır. Bunlar doğru, evet, ikilik sistemde 1 rakamı gibi değerleri temsil etmek için *true*, yanlış, hayır ve 0 rakamını temsil etmek için *false* ve hiçbir verinin olmaması durumunu temsil etmek için kullanılan *nil* yapılarıdır. Bu yapılar akış kontrolü sağlanması konusunda kullanışlıdır. Akış kontrol yapılarından ileride ayrıca bahsedeceğiz.

Diziler: Diziler, sıralı listelerden başka bir şey değildir. Tıpkı metinlerde bahsettiğimiz gibi dizilerde de indisler sıfır rakamı ile başlar. Bir dizinin elemanları temel veri türlerinden olabileceği gibi; başka bir dizi veya sözlük olabilir. Bir dizi oluşturmak için çok sayıda olası yoldan bahsedebiliriz ancak bunların içerisinde en basit ve kolay anlaşılabilir olanı [ ] karakterlerinin kullanılmasıdır. Örnek üzerinde bir dizi oluşturulmasını söyle gösterebiliriz:

```
dizi1 = [1,3,5,7]
dizi2 = ["bir", "iki", 3, true, nil, ["elma", "armut", "kiraz"]]
```

Bir dizinin elemanlarına erişim için yine metinlerde olduğu gibi indis değerlerini kullanabiliriz:

```
dizi2 [0]          #=> "bir"
dizi2 [2]          #=> 3
dizi2 [5]          #=> ["elma", "armut", "kiraz"]
dizi2 [5] [2]      #=> "kiraz"
```

Programlama dünyasının en çok kullanılan veri yapılarından birisi olan diziler ile ilgili diğer birkaç metodunun kullanımı aşağıda örneklendirilmiştir:

```
ornek_dizi1 = [1,2,3]
ornek_dizi2 = ['a','b','c']

# Dizinin sonuna eleman ekleme
ornek_dizi1.push(4)        # [1,2,3,4]
ornek_dizi1.push([5,6])    # [1,2,3,4,[5,6]]
ornek_dizi1.push(7,8)      # [1,2,3,4,[5,6],7,8]
ornek_dizi1 << 9           # [1,2,3,4,[5,6],7,8,9]

# Dizinin istenilen yerine eleman ekleme veya değerini değiştirme
ornek_dizi1[8] = 10        # [1,2,3,4,[5,6],7,8,9,10]
ornek_dizi1[0] = 101       # [101,2,3,4,[5,6],7,8,9,10]

# Dizinin sonundan eleman alma
ornek_dizi1.pop      #=> 10
ornek_dizi1.pop(2)   #=> [8,9]
ornek_dizi1          #=> [101, 2, 3, 4, [5, 6], 7]

# Dizileri birleştirme
ornek_dizi1 + ornek_dizi2
#=> [101, 2, 3, 4, [5, 6], 7, "a", "b", "c"]

# Dizi elemanlarını sıralama
[8, 0, 15, -42].sort     #=> [-42, 0, 8, 15]

# Belli bir karaktere göre metni parçalayarak dizi oluşturmak
"Eskişehir Anadolu Üniversitesi".split(' ')
#=> ["Eskişehir", "Anadolu", "Üniversitesi"]
"Fizik,Kimya,Biyoloji".split(',')
#=> ["Fizik", "Kimya", "Biyoloji"]
```

Diziler, ayrıca döngü oluşturma ve yönetmede de yoğun olarak kullanılmaktadır. İlerleyen sayfalarda bu konuya ilişkin örnekler verilecektir.

Sözlükler: Sözlükler de tıpkı diziler gibi bir liste yapısıdır. Aralarındaki en temel fark; bir dizinin elemanlarına erişmek için indisleri kullanmamız gerekirken, sözlüklerde bunun yerine onlara vermiş olduğumuz isimleri, diğer bir deyişle anahtar değerleri kullanırız. Diğer bir fark ise sözlükler oluştururken kullandığımız [ ] karakterleri yerine { } karakterlerini kullanmamız gerektiğidir. Bir örnek üzerinde görelim:

```
sozluk = {"elma" => 3, "mandalina" => 5}
sozluk ["elma"]    #=> 3
```

Sözlük oluştururken anahtar değerler için sembol kullanımının tercih edilmesi durumunda daha okunabilir bir kod elde etmemiz de mümkün olacaktır. Alternatif sözlük sentaksı olarak adlandırabileceğimiz bu kullanım şekli şöyledir:

```
sozluk = {elma: 3, mandalina: 5}
sozluk [:elma]    #=> 3
```

Sözlüklerle birlikte kullanabileceğimiz diğer bazı metotlarla ilgili aşağıdaki örnekleri inceleyiniz:

```
urunler = {elma: 3, armut: 5, kiraz: 8}
puts "Elimizde #{urunler [:armut]} kilo armut bulunmaktadır."
# Elimizde 5 kilo armut bulunmaktadır.

urunler.keys       #=> [:elma, :armut, :kiraz]
urunler.values     #=> [3, 5, 8]
```

Diğer Yapılar

Metotlar: Metotlar nesne yönelimli programlamada oldukça önemli bir yere sahip olmalarının yanında, NYP kullanılmasa dahi programlarımızda aynı kod bloğunu olduğu gibi veya farklı parametrelerle bir defadan fazla çalıştırmamız gerektiğinde de oldukça kullanışlı yapılar olarak karşımıza çıkmaktadır. Ruby'de metot tanımlamaları *def* anahtar kelimesi ile başlar ve *end* anahtar kelimesi ile biter. Aşağıda, ekrana "merhaba" metnini yazan basit bir metot tanımlaması görülmektedir:

```
def merhaba
  puts "Merhaba"
end
```

Bir metodun oluşturulması, o metot içerisinde tanımlananların yerine getirilmesi anlamına gelmez. Metodun işini yerine getirebilmesi için ona bir çağrı gönderilmesi, yani çağırılması gerekmektedir. Bir metodu çağırmak için ismini yazmamız yeterli olacaktır:

```
merhaba     # Merhaba
```

Metotlarla birlikte kullanmak istediğimiz parametreler varsa; bunları metot isminden hemen sonra, aralarına virgül koyarak, herhangi bir tip tanımlamasına ihtiyaç olmadan belirtebiliriz.

```
def merhaba(isim)
  puts "Merhaba #{isim}!"
end

merhaba('Nedim')     # Merhaba Nedim!
merhaba('Nazlı')     # Merhaba Nazlı!
```

Daha önce Ruby'de her ifadenin geriye bir şey döndürdüğünden bahsetmiştik, tabi ki bu duruma metotlar da dahildir. Bu geri dönüş değerlerine bazen ihtiyacımız olmasa da çok kullanışlı olduğu durumlarda bulunmaktadır. Pek çok dile benzer olarak Ruby'de bu amaçla *return* anahtar kelimesini kullanmaktadır. Bununla birlikte eğer döndürülecek değer metot gövdesinin son satırında yer alıyorsa; *return* kelimesinin kullanılmasına gerek yoktur. Örneği inceleyiniz:

```
def topla(a, b)
  a + b
end

puts "34 ile 26 sayılarının toplamı #{topla(34,26)}'dır."
# 34 ile 26 sayılarının toplamı 60'dır.
```

Metodun son satırı şöyle de yazılabilirdi:

```
def topla (a, b)
    return a + b
end
```

Ruby'e özgü özelliklerden birisi de bazen metotların son karakterlerinin onlar hakkında fazladan bilgi içerebilmesidir. Örneğin daha önce öğrendiğimiz metinlerle birlikte kullanılan *capitalize* metodunu ele alalım. Bu metot ilgili metnin ilk harfini büyütüp, diğer harflerini küçültmekte, bu şekilde elde ettiği yeni metni ise metodun geriye dönüş değeri olarak kullanmaktadır. Ancak bu işi yaparken orijinal metin nesnesi üzerinde herhangi bir değişiklik yapmamaktadır. Aşağıdaki örnekte bu durum gözlenebilir:

```
isim = "erHAN"
isim.capitalize   #=> "Erhan"
isim              #=> "erHAN"
```

Bu metot ile aynı işi yapan, bununla birlikte orijinal nesne üzerinde de değişiklik yapılmasını sağlayan *capitalize!* adında bir başka metot daha bulunmaktadır. Metodun bu şekilde isimlendirilmesi, programcının kullanırken daha dikkatli olmasını sağlayacaktır.

```
isim = "erHAN"
isim.capitalize!  #=> "Erhan"
isim              #=> "Erhan"
```

Bir diğer metot son eki ise "?" karakteridir. Bir metot isminin son harfinin "?" olması; bu metodun geriye true veya false bir değer döndürdüğü anlamına gelmektedir:

```
# Tek sayı mı?
12.odd?  #=> false
13.odd?  #=> true

# Pozitif mi?
-3.positive? #=> false
3.positive?  #=> true
```

Görüldüğü gibi, bu özel son ekleri kendi oluşturduğumuz metotlarda da kullanmamız işimizi oldukça kolaylaştıracaktır.

Bloklar: Bloklar, ruby içerisinde oldukça sık kullanılan yapılardır. Blokları isimsiz metotlar olarak düşünebiliriz ki; metotlarda olduğu gibi bu yapıları da genellikle aynı işlemin tekrarlanması veya farklı bir noktada kullanılması amacıyla kullanmaktayız. Bununla birlikte bir bloğun tek başına kullanımı sık karşılaşabileceğimiz bir durum değildir. Ruby içerisinde bloklar, bir metoda parametre olarak gönderilmekte ve çalışmaları sağlanmaktadır.

Bir bloğun başlangıç ve bitiş noktalarını belirtmek için *do* ve *end* anahtar kelimelerini kullanırız. Bu amaçla süslü parantezleri de kullanabiliyor olsak da, bu kullanım şekli bloğun içeriğinin bir satırdan uzun

olması durumunda pek tercih edilmemektedir. Aşağıdaki örnek her iki kullanım şeklini de göstermektedir:

```
do
  # blok içeriği
  # blok içeriği
end

{ # blok içeriği }
```

Blokların metotlar ile birlikte kullanımına ilişkin basit bir örnek olarak; tam sayılar ile birlikte kullanabileceğimiz *times* metodunu ele alalım. Aşağıdaki kod parçası ekrana 3 defa "Merhaba Ruby!" metninin yazdırılmasını sağlayacaktır.

```
3.times do
  puts "Merhaba Ruby!"
end
```

Bloklarla birlikte parametrelerin de kullanılması mümkündür; aslında bloklar, gerçek gücünü parametre kullanımı ile birlikte göstermektedirler. Örneğin bir dizi içerisindeki tüm elemanlara sırasıyla erişmemizi sağlayan *each* metodunun bir blok ile birlikte kullanımı aşağıda görülebilir:

```
["Aylin", "Demir", "Zeki"].each do |kisi|
  puts "Merhaba, #{kisi}!"
end
```

Yukarıda kod parçasında | ve | karakterleri arasında yer alan *kisi* isimli parametre veya değişken, metodun o anda erişmekte olduğu dizi elamanın içeriğine sahip olacaktır. Bu sayede each metodu ekrana her bir dizi elemanı için ayrı ayrı selamlama cümlesini yazdıracaktır.

Örnek vermiş olduğumuz times ve each metotları döngü oluşturmak için sıklıkla kullanılan metotlardandır. Farklı bir amacı olan diğer bir metotla devam edelim. Bir metin içerisinde başka bir metni aramamız ve/veya değiştirmemiz gerektiği durumlarda oldukça kullanışlı olan *gsub* metodunun bloklarla birlikte kullanımı aşağıda gösterilmektedir:

```
"ankara".gsub("a") { |harf| harf.upcase }
#=> "AnkArA"
```

Akış Kontrolü: Bir program yazabilmek için en çok ihtiyacımız olan ifadelerin akış kontrolü veya karşılaştırma ifadeleri olduğunu söylersek, yanılmış olmayız. Bu ifadelerin sonucu her zaman true veya false mantıksal değerlerinden birisine eşit olacaktır.

Ruby'de karşılaştırma ve mantıksal değerlendirme amacıyla kullanabileceğimiz operatörlerden bazıları Tablo 8.1'de gösterilmektedir.

Tablo 8.1 Karşılaştırma Operatörleri ve Mantıksal Operatörler.

Operatör	Açıklama
==	Eşittir
>, <, >=, <=	Büyüktür, küçüktür, büyük veya eşit, küçük veya eşit
!=	Eşit değil
&&	Ve
||	Veya
!	Değil

Bunların yanında daha önce geriye *true* veya *false* değer döndüren metotlar olduğundan bahsetmiştik. Doğal olarak bu metotları da mantıksal ifadeler ile birlikte kullanabiliriz.

Mantıksal ifadeleri programlarımızda gerçekleşen durumlara göre uygun ifadelerin çalıştırılması yani kontrolün sağlanması amacıyla kullanmaktayız. Pek çok programlama dilinde olduğu gibi Ruby'de de bu amaçla *if / elsif / else* yapısından yararlanmaktayız. Örneğin hava durumuna göre kullanıcıya uygun mesajı gösteren aşağıdaki programı inceleyiniz:

```
hava = "yağışlı"

if hava == "güneşli"
  puts "Piknik zamanı!"
elsif hava == "yağışlı"
  puts "Şemsiyeni unutma!"
elsif hava == "karlı"
   puts "Böyle zamanlarda Uludağ ayrı
bir güzel oluyor."
else
  puts "Malesef bu hava durumu hakkın-
da bilgim yok"
end
```

Mantıksal ifadelerle birlikte if/else yapısı dışında *unless* ve *case/when* yapıları da kullanılabilir. Ayrıca *while* ve *until* gibi döngü oluşturma yapılarıyla birlikte de mantıksal ifadeler kullanılmaktadır.

Aşağıda bu yapıların kullanımına ilişkin yer alan örneği inceleyiniz:

```
sayac = 1
while sayac <= 5
  print "#{sayac} "
  sayac = sayac + 1
end

sayac = 1
until sayac > 5
  print "#{sayac} "
  sayac = sayac + 1
end
# Her iki örneğin de çıktısı: "1 2 3 4 5" şeklinde olacaktır.
```

Sınıflar ve Nesneler: Ruby'nin tamamen nesne yönelimli bir dil olduğundan ve dolayısıyla etkileşimde bulunduğumuz her şeyin, her verinin bir nesne olduğundan bahsetmiştik. Nesneler tıpkı gerçek hayattaki gibi bir takım özelliklere ve eylemlere sahiptir ki biz bu özellikler için değişkenleri, eylemler için de metotları kullanmaktayız.

Bir nesne oluşturabilmek için öncelikle o nesnenin ait olacağı bir sınıf tanımlamamız gerekmektedir. Sınıf tanımlaması class anahtar kelimesi ile başlayıp end kelimesi ile sonlandırılır.

```
class Bilgisayar
  attr_accessor :islemci, :bellek

  def konfigurasyon
    puts "Bu bilgisayar #{islemci} işlemciye ve #{bellek} belleğe
          sahiptir!"
  end
end
```

Yukarıdaki kod parçasında *Bilgisayar* isminde bir sınıf tanımladık. Sınıf isimlerinin bir sabit olduğundan ve sabitlerin büyük harf ile başlaması gerektiğinden daha önce bahsetmiştik. Sınıfa ait özellikleri belirtirken *attr_accessor* isimli ön tanımlı bir metottan yararlandık. Bu metot *islemci* ve *bellek* isminde iki özellik ve bu özelliklere aynı isimlerle erişmemizi sağlayan ikişer metodu da bizim için tanımlamış oldu. Sonrasında da oluşturacağımız bilgisayar nesnesi ile ilgili bilgileri ekrana yazdırmamızı sağlayacak olan *konfigurasyon* isimli bir metot tanımlaması yaptık.

Oluşturmuş olduğumuz sınıftan yeni bir nesne yaratmak için sınıf adı ile birlikte *new* metodundan yararlanmaktayız.

```
b1 = Bilgisayar.new
b2 = Bilgisayar.new
```

Bu nesnelere ilişkin özelliklere aşağıdaki gibi değerler atayabiliriz:

```
b1.islemci = "2,7 GHz Intel Core i5"
b1.bellek = "8 GB 1867 MHz DDR3"
b2.islemci = "3,2 GHz Intel Core i7"
b2.bellek = "4 GB 1867 MHz DDR3"
```

Benzer şekilde istenilen herhangi bir özelliğe, o özelliğin ismini kullanarak erişebiliriz:

```
b1.bellek      #=> "8 GB 1867 MHz DDR3"
b2.islemci     #=> "3,2 GHz Intel Core i7"
```

Tanımlamış olduğumuz konfigurasyon isimli metodu da doğrudan bu nesneler üzerinde kullanabiliriz:

```
b1.konfigurasyon
# Bu bilgisayar 2,7 GHz Intel Core i5 işlemciye ve 8 GB 1867 MHz DDR3
# belleğe sahiptir!
b2.konfigurasyon
# Bu bilgisayar 3,2 GHz Intel Core i7 işlemciye ve 4 GB 1867 MHz DDR3
# belleğe sahiptir!
```

Sınıflarla ilgili son olarak, kalıtım özelliğinden de bahsetmeden geçmeyelim. Kalıtım bir sınıfın istenilen özellik ve metotlarının, kendisinden oluşturulan alt sınıflar tarafından da kullanılabilmesini sağlamaktadır. Bu konuda Ruby on Rails'dan bahsederken çok sayıda örnekle karşılaşacak olduğumuzdan burada basit bir örnek vermek yeterli olacaktır. Mesela elimizde Makine adında bir sınıf olduğunu ve ve Bilgisayar sınıfının da bu sınıfın alt sınıfı olmasını istediğimizi düşünelim. Bu durumda sınıf tanımlaması aşağıda gösterildiği gibi olacaktır:

```
class Bilgisayar < Makine
  ...
end
```

Ruby Paket Yöneticisi (RubyGems)

Ruby ile geliştirilen uygulamalara ekstra özellikler ve işlevsellik katmak amacıyla kullandığımız paketlere "gem" adını vermekteyiz. Bu paketleri yönetmek için kullandığımız uygulamaya ise RubyGems denilmektedir. Ruby yüklemesi ile birlikte zaten sistemimize kurulmuş olan RubyGems paket yöneticisi aşağıda gösterildiği şekilde kullanılabilir:

Yeni bir paket yüklemek için:

```
$: gem install paket_adi
```

Yüklü paketi son sürümüne güncellemek için:

```
$: gem update paket_adi
```

Bir paketi sistemimizden kaldırmak için:

```
$: gem uninstall paket_adi
```

komutlarını terminal üzerinde çalıştırabiliriz.

Çoğunlukla paketleri bu şekilde tek tek yüklemeye ihtiyaç duymayız. Bunun yerine bu işi bizim yerimize halledecek başka paketler de bulunmaktadır. Örneğin; Ruby on Rails başlığı altında ayrıca bahsedeceğimiz Bundler gibi.

RubyGems hakkında daha fazla bilgi edinmek ve tüm paketler içeresinde arama yapmak amacıyla https://rubygems.org/ adresini ziyaret edilebilir. Daha hiyerarşik bir yapıda paketleri incelemek amacıyla da https://www.rubytoolbox.com/ adresini ziyaret edebilirsiniz.

Ruby paketlerinin yanı sıra çekirdek içerisinde yer almayan ama istenilmesi durumunda programlarımıza dahil edebileceğimiz pek çok kütüphane de mevcuttur. Bu kütüphaneleri incelemek amacıyla http://ruby-doc.org/stdlib-2.3.1/ adresini ziyaret edilebilir.

Öğrenme Çıktısı

2 Ruby dilinin temel söz dizimi kuralları ve kullandığı yapıları açıklayabilme

Araştır 2

Bir işletim sistemi programlamayı düşünüyorsunuz. Bu durumda Ruby dilini tercih eder misiniz? Nedenlerini açıklayınız.

İlişkilendir

Ruby'nin çalışma zamanında kod üretme (kod yazan kodlar oluşturma) özelliğini yapay zekâ mantığı ile ilişkilendiriniz.

Anlat/Paylaş

Ruby topluluklarına katılarak bu topluluklarda öğrendiğiniz Ruby bileşenlerini ve söz dizimlerini paylaşınız.

RUBY ON RAILS ÇATISI

Ruby on Rails (bundan sonra sadece Rails olarak bahsedilecektir), Ruby dili ile geliştirilmiş bir web uygulama geliştirme çatısıdır (framework). İnternet tabanlı uygulamalarda ihtiyaç duyulabilecek şeylere ilişkin varsayımlar kullanarak, uygulamaların çok kısa sürede ve kolayca geliştirilmesine imkan tanımaktadır. Bu sayede az kodla, çok iş yapılması mümkün olmaktadır.

Rails felsefesi ya da daha çok bilinen ismiyle Rails yolu iki ana prensibi benimsemektedir:

- *Kendi Kendini Tekrarlama:* DRY(Don't Repeat Yourself) olarak bilinen bu prensip; her bilgi parçasının bir sistem içerisinde tek, kesin ve güvenilir bir ifadesinin olması gerektiğini savunmaktadır. Aynı bilginin tekrar tekrar yazılmıyor olması kodumuzun daha yönetilebilir, esnek ve az hatalı hale gelmesini sağlayacaktır.
- *Konfigürasyon Üzerine Kabuller*: Rails, bir web uygulamasında gerekli olan pek çok şeyi yapmak için kullanılabilecek en iyi yolun hangisi olduğuna ilişkin fikirlere sahiptir ve bunu varsayılan olarak sunmaktadır. Bu sayede programcılar, sonsuz konfigürasyon ayarlamaları ile uğraşmak yerine; ya hiç ya da çok az değişikliğe ihtiyaç duyacaklardır.

Rails, MVC adı verilen, Model-Görünüm(View)-Kontrolör(Controller) yazılım mimarisi desenini kullanmaktadır. MVC, çevik(agile) yazılım geliştirme süreçlerine uygun olarak sıklıkla tercih edilen bir yazılım mimarisi desenidir. İlk olarak 1979 yılında SmallTalk programlama dili ile birlikte ortaya çıkmış olan MVC'nin ardındaki ana düşünce; yazılımın kullanıcıya hitap eden bölümü yani kullanıcı arayüzü ile arka plandaki işlemlerin birbirinden ayrı bölümler olarak ele alınmasına dayanmaktadır. Bu amaçla sistemi üç ana rol başlığı altında geliştirmeyi öngörmektedir:

- *Model*: Uygulamaya özel verilerin ve uygulamanın iş mantığına uygun biçimde bu veriler arasındaki ilişkilerin, erişim ve düzenleme kurallarının bulunduğu bölümdür. Diğer bir deyişle uygulamamızın kullandığı tüm verilerin yaşadığı yer burasıdır. Model, bu verilerin kullanıcıya ne şekilde sunulacağı ile ilgilenmez.
- *View (Görünüm)*: Uygulamanın kullanıcılarına sunulacak olan arayüz elemanlarını içeren yani kullanıcıların gördüğü yegane bölümdür. Kullanılan verilerin nereden geldiği ile ilgili bir bilgisi yoktur.
- *Controller (Kontrolör)*: Kullanıcıdan gelen isteklerle ilgili işlemleri gerçekleştirmek üzere gerekli görevleri yerine getiren bölümdür. Uygulamayı kullanan kişi ne zaman bir istekte bulunacak olsa, bu isteği ele alacak olan bölüm burasıdır. İsteğe uygun olarak gerekli yanıtı (genellikle uygun verilerle birlikte bir görünüm oluşturarak), istemciye ulaştırmakla yükümlüdür.

MVC kullanan standart bir uygulamada çok sayıda model, görünüm ve kontrolör bulunur. Bu bölümler arasındaki ilişki; aracı rolünü oynayan kontrolör tarafından gerçekleştirilmektedir. Yani model ve görünüm arasında doğrudan bir ilişki söz konusu değildir. Eğer uygulamamız bir veri tabanı kullanmaktaysa; veri tabanına erişmekle yükümlü olan bölüm modeldir. Kullanıcı, görünümleri kullanarak veya doğrudan bir istek yaptığında, bu istek kontrolöre iletilir. Bu isteğe karşılık olarak,

kontrolör gerekliyse model ile iletişime geçtikten sonra kullanıcıya sunulacak görünümü oluşturur ve oluşturulan bu görünüm kullanıcıya sunulur.

Tüm internet tabanlı uygulamalarda kullanıcının uygulama ile iletişime geçmesi istemci-sunucu ilişkisi olarak adlandırılan bir yapı içerisinde yer alır. Rails açısından bu ilişkinin geçtiği basamakları şu şekilde özetleyebiliriz:

1. Bir istemcinin uygulamamız ile iletişime geçmesi sırasında gerçekleşecek olan ilk adım, sunucudan bir HTTP isteğinde bulunulmasıdır. Bir HTTP isteği kaynağın URL adresi, istek metodu ve başlık bilgilerinden oluşur.
2. İsteği alan sunucu, onu Action Dispatch bileşenine iletir. Bu bileşen, istenilen kaynağa ilişkin URL adresi ile uygulamanın rota tablosunda yer alan kayıtları karşılaştırır. Eşleşme bulması durumunda bu rotaya karşılık olarak belirtilmiş olan kontrolöre ilişkin eylemi çalıştırır ve (varsa) parametreleri de gönderir. Böylece kontrol Action Controller bileşenine geçer.
3. Eylemin çalıştırılması sırasında veri tabanı ile etkileşim gerekliyse; Action Controller bileşeni, Active Record bileşeni ile etkileşime geçer ve bu bilesen veri tabanındaki gerekli işlemleri gerçekleştirir.
4. Action Controller daha sonra kontrolü Action View bileşenine devrederek görünümlerin oluşturulmasını sağlar. Oluşturulan görünümler tekrar Action Controller bileşenine iletilir.
5. İletişimin son adımı da HTTP yanıtının oluşturularak, istemciye gönderilmesi olacaktır. Bu adımda; Action Controller, Action View'dan gelen çıktıyı (HTML, XML, JSON gibi) mesajın gövdesine ekleyerek, diğer bilgilerle (baslık bilgileri, durum kodu) beraber web sunucusuna gönderir. Web sunucusu da tüm bunları tarayıcıya gönderir ve süreç tamamlanır.

Birazdan geliştireceğimiz örnek uygulamada bu yapının nasıl çalıştığı daha iyi anlaşılabilecektir. Bunun öncesinde Rails ile uygulama geliştirmeye başlamak için rails paketini sistemimize yüklememiz gerekecektir. Bu amaçla terminalde aşağıdaki komutu çalıştırmalıyız:

```
$: gem install rails
```

Eğer Windows altında çalışmayı planlıyorsanız; ayrıca DevKit adındaki geliştirici alet çantası uygulamasını da yüklemeniz gerekecektir. DevKit uygulaması, http://rubyinstaller.org/add-ons/devkit adresinden indirilip, yüklenebilir.

Yükleme işleminin başarılı olup olmadığı aşağıda gösterildiği gibi kontrol edilebilir. Çıktıdan görülebileceği gibi biz bu ünitede Rails 5.0.0.1 versiyonunu kullanacağız:

```
$: rails --version
Rails 5.0.0.1
```

Sistemimize rails yüklemesi ile birlikte çok sayıda harici paketin de yüklendiği(daha önce yüklü değilse) gözlenecektir. Bu yeni yüklenen paketlerin bazılarından ünitenin ilerleyen sayfalarında bahsedeceğiz.

Öğrenme Çıktısı

3 Ruby on Rails çatısının temel çalışma prensiplerini açıklayabilme

Araştır 3

Ruby on Rails'in iki temel prensibinden ilki olan "Don't Repeat Yourself" kodlamada ne anlama gelmektedir.

İlişkilendir

Ruby on Rails çatısını MVC mimarisi ile ilişkilendirerek anlatınız.

Anlat/Paylaş

Ruby on Rails ile oluşturulmuş bir internet tabanlı uygulamanın çalışma adımlarını paylaşınız.

RUBY ON RAILS İLE UYGULAMA GELİŞTİRME

Rails paketini yüklediğimizde, terminal üzerinden kullanabileceğimiz kendisiyle aynı isimli bir aracı da yüklemiş olduk. Rails ile birlikte çalışırken çoğu zaman kendimizi bu aracı kullanırken bulacağız ve uygulamamızın ilk adımından son adımına kadar pek çok eylemi bu komutu kullanarak gerçekleştireceğiz. Bu aracın kullanımı hakkında daha fazla bilgi sahibi olmak isterseniz terminalde "rails --help" komutunu çalıştırmayı deneyebilirsiniz.

Rails'in temel özelliklerini anlamak için basit bir uygulama geliştireceğiz. Bu uygulama eşliğinde yeri geldikçe ilgili konu hakkında daha detaylı bilgilendirmelere yer vereceğiz.

Rails, beraberinde üretici (generator) adı verilen pek çok hazır komut dosyası ile birlikte gelir. Üreticiler, belirli görevleri hızlı ve kolayca yerine getirebilmemiz için tasarlanmışlardır ve rails uygulamalarında sıklıkla kullanılmaktadır. Kullanacağımız ilk üretici, uygulamamıza ilişkin temel yapıyı oluşturacak olan "new" üreticisi olacaktır. Bu üreticinin kullanımı terminal üzerinde "rails new uygulama_adı" komutunun çalıştırılması şeklindedir. Aşağıdaki komut "blog" adında yeni bir rails uygulaması oluşturmamızı sağlayacaktır:

```
$: rails new blog
create
      create  README.md
      create  Rakefile
...
         run  bundle install
Fetching gem metadata from https://rubygems.org/..........
...
Bundle complete! 15 Gemfile dependencies, 62 gems now installed.
Use `bundle show [gemname]` to see where a bundled gem is installed.
         run  bundle exec spring binstub --all
* bin/rake: spring inserted
* bin/rails: spring inserted
```

Konsol çıktısı incelendiğinde bu kısacık komut ile birlikte Rails'in çok sayıda işlem yaptığını görmekteyiz. Yapılan bu işlemleri şöyle sıralayabiliriz:

- "create" komutu ile uygulamamızın ihtiyaç duyacağı tüm dosya ve dizinler oluşturuldu.
- "run bundle install" komutu ile uygulamamızın ihtiyaç duyduğu tüm paketlerin yüklenmesi sağlandı.
- "run bundle exec spring binstub --all" komutu ile uygulamamızı geliştirirken kullanacağımız rake ve rails araçları, daha hızlı çalışmalarını sağlamak amacıyla spring isimli uygulama ön yükleyicisi paketi ile entegre hale getirildi.

Bu noktada çokça karşımıza çıkmış olan "bundle" aracından bahsetme ihtiyacımız doğmuştur. Daha önce Ruby ile geliştirilen uygulamalara ekstra özellikler ve işlevsellik katmak amacıyla kullandığımız paketlere *gem* adının verildiğini, sistemimize bir *gem* dosyası yüklemek için terminal uygulamasındayken "gem install paket_adi" şeklinde bir komut çalıştırmamız gerektiğinden bahsetmiştik. Yukarıdaki ekran çıktısını incelediğimizde uygulamamızın toplamda 62 adet (bu sayı rails versiyonuna bağlı olarak değişiklik gösterebilir) paketin yüklenmesine ihtiyaç duyduğunu görebiliriz. Böyle bir durumda her bir paketin "gem install" komutu kullanılarak yüklenmesi; hem geliştirme hem de uygulamanın yayınlanması sırasında tekrar yüklenmesi, paketlerin versiyonları arasında uyumsuzluk yaşanması gibi durumlardan dolayı oldukça zahmetli bir iş haline gelecektir. Bu nedenle; Rails'i kurmamızla birlikte bundle adında bir paket otomatik olarak yüklenmektedir. Bu paket, bundler yani paketçi olarak isimlendirilmektedir. Bundler

Bundler hakkında daha detaylı bilgi edinmek amacıyla http://bundler.io/ adresi ziyaret edilebilir.

sayesinde uygulamamızın ihtiyaç duyduğu paketlerin tamamını tek seferde ve versiyon uyumsuzluğu gibi sorunlar yaşamadan yükleyebilmekteyiz. Eğer rails ile birlikte gelen paketlere ek olarak başka bir paket kullanmak istersek veya yüklü olan bir paketi kaldırmak istersek uygulamamızın ana dizininde yer alan "Gemfile" isimli dosya üzerinde bu değişikliği yapmalı ve ardından terminalde "bundle install" komutunu çalıştırmamız gerekecektir. Yüklü olan bir pakete ait versiyon bilgisinin güncellenmesi durumunda ise "bundle update" komutu kullanılabilir.

Uygulamanın ilerleyen adımlarında yeniden rails komutuna ihtiyaç duyacağız. Ancak bu noktadan itibaren global durumda olan rails komutunu kullanmak yerine, uygulamamızın bin/ dizini içerisinde yer alan rails dosyasını uygulamanın ana dizinindeyken "bin/rails" şeklinde çağırarak kullanacağız.

Oluşturmuş olduğumuz boş rails uygulaması aslında tüm gereklilikleri sağlayan yani çalıştırılabilir bir uygulamadır. Bu uygulamayı tarayıcı üzerinde görüntülemek ve her şeyin çalışır durumda olduğundan emin olmak için terminal üzerinde "blog" klasörüne geçtikten sonra "bin/rails server" komutunu vermemiz yeterli olacaktır:

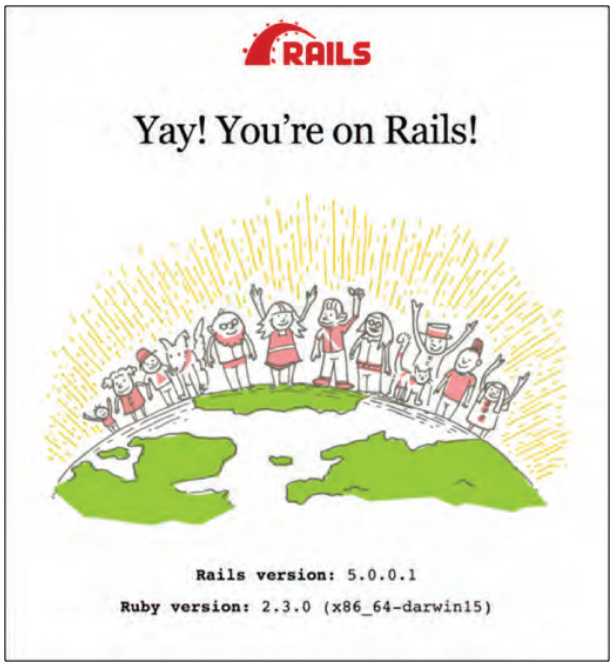
dikkat

Eğer işletim sistemi olarak Windows kullanıyorsanız; rails aracını kullanarak çalıştıracağınız tüm komutları "ruby bin/rails server" şeklinde doğrudan ruby yorumlayıcısına göndererek çalıştırmanız gerekmektedir.

```
$: cd blog
$: bin/rails server
=> Booting Puma
=> Rails 5.0.0.1 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
Puma starting in single mode...
* Version 3.6.0 (ruby 2.3.0-p0), codename: Sleepy Sunday Serenity
* Min threads: 5, max threads: 5
* Environment: development
* Listening on tcp://localhost:3000
Use Ctrl-C to stop
```

Bu komut, rails tarafından varsayılan olarak kullanılan Puma isimli web uygulama sunucusunun çalıştırılmasını ve yine varsayılan port olan 3000 portu üzerinden uygulamamızı servis etmesini sağlayacaktır. Bu komuttan sonra web tarayıcımızda http://localhost:3000 adresini ziyaret ederek Resim 8.1'dekine benzer bir karşılama sayfası görebiliriz.

Uygulama sunucusunun çalışmasını durdurmak isterseniz terminalde Ctrl+C tuşlarına basmamız yeterli olacaktır. Rails new üreticisi ile oluşturduğumuz klasörler ve dosyalardan bazılarına ilişkin açıklamalar Tablo 8.2'de gösterilmektedir.



Resim 8.1 Rails Varsayılan Karşılama Sayfası

Tablo 8.2 Ruby on Rails Uygulamasında Yer Alan Dizin ve Dosyalar

Klasör/Dosya	Açıklama
app/	Bir geliştiricinin üzerinde en çok çalışacağı yer burasıdır. Uygulamamızda kullanacağımız modeller, görünümler, kontrolörler ve postacılar gibi yapılara ilişkin hemen hemen her şey bu klasör içerisinde uygun alt klasörlere yazılacaktır.
bin/	Bu klasör çalıştırılabilir dosyaları barındırmaktadır. İçerisinde bundle, rails, spring ve rake gibi dosyaları barındırmaktadır. Örneğin biraz evvel çalıştırdığımız bin/rails komutuna ilişkin dosya burada yer almaktadır.
config/	Uygulamamıza ilişkin yapılandırma bilgilerinin yer aldığı klasördür. Örneğin uygulamamızda kullanacağımız veri tabanı ile ilgili ayarlar ve erişim bilgileri, URL isteklerinin yönlendirileceği rotalar, yerelleştirme ayarları gibi tüm yapılandırma bilgilerini bu klasör içerisinde saklarız.
db/	Veri tabanı ile ilgili şema, aktarımlar, çekirdek veriler gibi dosyalar bu klasörde yer alır.
Gemfile Gemfile.lock	Uygulamamızda kullandığımız paketlere ait isim, versiyon gibi bilgilerin yer aldığı dosyalardır. Yeni bir paket kullanacağımızda veya kullandığımız bir paketi kaldırmak/güncellemek istediğimizde Gemfile dosyası üzerinde değişiklik yaparız.
lib/	Modeller, görünümler ve kontrolörlere doğrudan ait olmayan uygulamaya özel ve paket haline getirilmemiş kütüphane kodları, rake görevleri vb. dosyaları barındırır. Daha çok uzman kullanıcılar tarafından kullanılan bir dizindir.
log/	Uygulamaya ait günlük dosyalarının bulunduğu dizindir. Uygulamanın çalışması sırasında meydana gelen tüm olaylar bu dosyalara kaydedilmektedir.
public/	Uygulamamızın dış dünyaya acık olan yegâne dizinidir. İçerisinde statik dosyalar bulunmaktadır.
test/	Uygulamamıza ait testlerin bulunduğu klasördür. Uygulamamızın, daha önceden belirlenmiş beklentilere uygun olarak çalışıp çalışmadığını test etmemizi sağlar.
tmp/	Önemsiz ve geçici dosyalar bu dizinde yer almaktadır.
vendor/	Tipik bir rails uygulamasında kullanılan paketlere ait üçüncü parti yazılım kodlarının bulunduğu dizindir.

Daha önce Rails'in MVC mimarisini temel aldığından bahsetmiştik. Bununla birlikte bir web sayfasının illa ki veri tabanı yani model ile ilişkide olması da gerekmeyebilir. Bu durumda Rails'e bir şeyler yaptırmak için en azından bir kontrolör ve görünüme ihtiyacımız olduğunu söyleyebiliriz. Bu amaçla bir başka üretici olan "generate" komutundan yararlanacağız. Bu komutu kullanarak kontrolörler ve ilişkide olacakları görünümler, modeller ve yine ilişkili veri tabanı aktarımları, model-görünüm ve kontrolörü bir arada barındıran iskeleler, e-postacılar, vb. ve hatta başka bir üretici bile oluşturabiliriz. Blog uygulamamızın giriş sayfası olarak kullanmak amacıyla bir görünüm ve kontrolör oluşturmak için aşağıdaki komutu çalıştıralım.

```
$: bin/rails generate controller Welcome index
Running via Spring preloader in process 1661
      create  app/controllers/welcome_controller.rb
       route  get 'welcome/index'
      invoke  erb
      create    app/views/welcome
```

dikkat

Rails kontrolör, model gibi bileşenler için belirlenen isimleri zaman zaman tekil ve çoğul hale getirmektedir. Rails'in kabulleri bu temel üzerinde çalıştığından biz de uygulamamızda Türkçe isimlendirme kullanmayacağız. Tabi ki uygulamanın kullanıcılar tarafından görüntülenecek olan kısımlarında Türkçe kullanılması sorun oluşturmayacaktır.

```
create    app/views/welcome/index.html.erb
invoke  test_unit
create    test/controllers/welcome_controller_test.rb
invoke  helper
create    app/helpers/welcome_helper.rb
invoke    test_unit
invoke  assets
invoke    coffee
create      app/assets/javascripts/welcome.coffee
invoke    scss
create      app/assets/stylesheets/welcome.scss
```

Komutun ekran çıktısı incelendiğinde *app/controllers/* dizini içerisinde belirttiğimiz isimde yani *welcome_controller.rb* adında bir dosya oluşturulduğunu görebiliriz. Söz konusu dosya yine aynı dizinde yer alan *ApplicationController* isimli kontrolör sınıfını da kalıtım yoluyla içermektedir ki varsayılan olarak tüm kontrolörler bu sınıftan kalıtım yoluyla oluşturulurlar. Bu sayede tüm kontrolörler için ortak olarak kullanmak istediğimiz kodları, bu sınıf içerisine yazarak kodumuzda oluşması muhtemel tekrarlardan kurtulabiliriz. *WelcomeController* sınıfı içerisinde, *generate* üreticisini çalıştırırken belirttiğimiz *index* eylemi de yer almaktadır. Görüleceği gibi kontrolör eylemleri, örnek metotlarından başka bir şey değildirler.

```ruby
#app/controllers/welcome_controller.rb
class WelcomeController < ApplicationController
  def index
  end
end
```

Şimdi Rails'in üzerinde değişiklik yaptığı başka bir dosyaya bakalım. Bir kullanıcının uygulamamıza ait olan bir URL adresini ziyaret etmesi durumunda hangi kontrolöre ait, hangi metodun çalıştırılması gerektiğini Rails'e bildirmemiz gerekmektedir. Bu bildirimlerin yer aldığı dosya *config/* dizini içerisindeki *routes.rb* dosyasıdır. Biraz önce oluşturduğumuz kontrolör sayesinde Rails otomatik olarak bu dosyaya aşağıda görüldüğü gibi bir rota eklemiştir:

```ruby
Rails.application.routes.draw do
  get 'welcome/index'
end
```

Rails yönlendiricisi (router) de tıpkı diğer Rails bileşenleri gibi kendine has bir DSL kullanmaktadır. Bununla birlikte metot isimleri oldukça anlaşılır şekilde isimlendirilmişlerdir. Örneğin burada görülen "get" metodu aynı isimli HTTP metodunun da ismidir. Bu metodun parametresinde yer alan metin "welcome/index" şeklinde bir URL isteği geldiğinde "welcome" kontrolörünün, index eylemine yönlendirilme yapılmasını sağlayacaktır. Bu durumu tarayıcınızda http://localhost:3000/welcome/index adresini ziyaret ederek gözlemleyebilirsiniz.

> Rails yönlendiricisi ile birlikte kullanabileceğimiz HTTP metotları GET, POST, PUT, PATCH ve DELETE olmak üzere beş adettir. GET metodu ile yapılan istekler sunucudaki bir kaynağı elde etmek için kullanılırken, POST metodu sonucunda sunucuda yeni bir kaynak oluşturulması hedeflenmektedir. Sunucudaki bir kaynağın güncellemesi amacıyla kullanılabilecek olan PUT ve PATCH metotlarından ilki kaynağın tamamıyla güncellenmesi durumunda kullanılırken, patch metodu kaynağın kısmen güncellenmesinin istenildiğini gösterir. DELETE metodu ile yapılan istekler ise sunucudaki bir kaynağın silinmesini hedef almaktadır.

Biraz önce ziyaret ettiğimiz sayfa yine Rails üreticisi tarafından otomatik olarak oluşturulmuş olan görünüm sayfamızın html çıktısıydı. Bu dosyayı "app/views/welcome" dizini içerisinde index.html.erb adıyla bulabilirsiniz. Anlaşılacağı üzere klasör ismi kontrolörün ismi ile, görünüm dosyasının ismi de eylem ile eşleşmektedir. Bu isimlendirme sayesinde Rails neyi, nerede arayacağını kendi kendine bulabilmektedir.

```
# app/views/welcome/index.html.erb
<h1>Welcome#index</h1>
<p>Find me in app/views/welcome/index.html.erb</p>
```

Bu dosya içeriği kullanıcıya gösterilmek istenilen html çıktısına göre düzenlenebilir. Dosya ismine bakıldığında *.erb* uzantısı dikkat çekmektedir. ERB, farklı dosya türleri içerisine Ruby kodlarını gömmemizi sağlayan bir şablon oluşturma sistemidir. Yani örneğimizdeki *index.html.erb* dosyası sayfayı talep eden ziyaretçiye gönderilmeden önce içerisindeki Ruby kodları çalıştırılacak ve asıl index.html dosyası oluşturulacaktır.

Erb etiketleri her zaman <% karakterleri ile baslar ve %> karakterleri ile sonlandırılır. Bu karakterler arasına yazılacak kodlar Ruby yorumlayıcısı tarafından değerlendirilecektir, ancak herhangi bir çıktı oluşturmayacaklardır. Eğer hem kodların değerlendirilip, hem de çıktı olarak dosyamıza eklenmesini istiyorsak kodumuzu gömmeye <%= karakterleri ile başlamamız gereklidir. Yandaki kodları görünümümüze yazarak deneyebiliriz:

```
<h1>Blog Sayfama Hoş geldiniz</h1>
<%
  yil = Time.now.year
  dogum_yili = 1995
  yas = yil - dogum_yili
%>
<p>Merhaba, Ben <%= yas %> yaşındayım.</p>
```

Yazmış olduğumuz kodun çıktısı Resim 8.2'de görüldüğü şekilde olacaktır:

Blog Sayfama Hoş geldiniz

Merhaba, Ben 21 yaşındayım.

Resim 8.2 Örnek Ekran Çıktısı

Uygulamanın giriş sayfasını da oluşturduğumuza göre blog girdileri oluşturup, yayınlayabilmemiz için bir alt yapı oluşturmamızın zamanı geldi. Bu işlem için ayrı ayrı model, kontrolör vs. oluşturabiliriz ancak biz bunun yerine iskele (scaffold) üreticisini kullanacağız.

```
$: bin/rails generate scaffold Article title:string body:text
Running via Spring preloader in process 5714
      invoke  active_record
      create    db/migrate/20161016000241_create_articles.rb
      create    app/models/article.rb
      ...
      invoke  resource_route
       route    resources :articles
      invoke  scaffold_controller
      create    app/controllers/articles_controller.rb
      invoke    erb
      create      app/views/articles
      create      app/views/articles/index.html.erb
    ...
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/articles.coffee
      invoke    scss
      create      app/assets/stylesheets/articles.scss
...
```

Üreticinin neler oluşturduğuna daha yakından bakalım. ilk satırda active_record çağrısını görmekteyiz. Active Record MVC mimarisinin model ile ilgili olan katmanıdır ve bir **ORM** (Object Relational Mapping) yani Nesne İlişkisel Haritalama yazılımıdır.

> *ORM* yazılımları, veri tabanımızın kullandığı uyumsuz tip sistemleri ile nesne yönelimli yazılımlar arasındaki dönüşümü sağlamaktadırlar.

Web uygulamalarında genellikle verilerimizi saklamak ve yönetmek için, ilişkisel veri tabanlarını kullanırız. İlişkisel veri tabanları üzerinde işlem yapmak için ise SQL dilinden yararlanmaktayız. Ancak SQL sorguları ile nesne yönelimli programlama kavramlarını ilişkilendirmemiz oldukça zordur; yani SQL sorgularının oluşturulma mantığıyla, nesne yönelimli programlamanın mantığı birbiriyle pek uyuşmamaktadır. İşte bu noktada ORM kavramı ve ORM yazılımları ile tanışmamız gerekecektir.

Bir ORM yazılımı olarak Active Record'ın yapmakla yükümlü olduğu şeyleri şu şekilde sıralayabiliriz: modellerin ve ilişkili verilerin yönetilmesi, modeller arasındaki ilişkilerin kurulması, modeller arasındaki kalıtım hiyerarşisinin sağlanması, modellerdeki verilerin veri tabanında kalıcı hale gelmeden önce doğruluklarının kontrol edilmesi ve veri tabanı işlemlerinin nesne yönelimli programlamaya uygun olarak yerine getirilmesi.

Tekrardan uygulamamıza dönecek olursak Active Record çağrısıyla birlikte db/migrate/ dizini içerisinde bir veri tabanı aktarımı ve app/models/ dizini içerisinde de modelimize ait sınıfı içeren bir dosyanın oluşturulduğunu görürüz.

Biraz önce oluşturduğumuz veri tabanı aktarım dosyasının içeriğine bakacak olursak; articles isminde bir tablo ve bu tablo içerisinde üretici ile birlikte belirtmiş olduğumuz title ve body alanlarının oluşturulmasına dair kodların yer aldığını görebiliriz. Bunun yanında timestamps isminde bir metodun kullanıldığını da görmekteyiz ki; bu metot otomatik olarak veri tabanındaki söz konusu tabloda her kaydın oluşturulma ve değiştirilme tarihlerinin tutulmasını sağlayacak olan created_at ve updated_at alanlarının oluşturulmasını sağlayacaktır.

> Veri tabanı aktarımları, Rails'in veri tabanı şeması üzerinde ekleme, çıkarma, güncelleme gibi işlemleri yapmak için kullandığı yapıdır. Bunun yanında aktarımlar kullanılması, veri tabanı üzerindeki şema değişikliklerinin tümünün tek seferde gerçekleştirilmesi yerine, bu değişikliklerin artımlı şekilde gerçekleştirilmesini sağlar. Bu sayede veri tabanının versiyon değişikliklerini takip etmemizi sağlayan bir zaman çizelgesini de kullanmamıza imkan tanır. Bunu her veri tabanı aktarım dosyasını oluştururken tarih ve saat bilgisini de dosyanın başına ekleyerek yapmaktadır.

```
# db/migrate/20161016000241_create_articles.rb
class CreateArticles < ActiveRecord::Migration[5.0]
  def change
    create_table :articles do |t|
      t.string :title
      t.text :body

      t.timestamps
    end
  end
end
```

Veri tabanı aktarımlarının fiziksel veri tabanımızda kalıcı hale gelmesi, yani söz konusu tabloların ve alanların kullanılabilmesi için atmamız gereken son bir adım daha vardır. Terminal üzerinde aktarıma ait kodun çalıştırılması için aşağıdaki komutu girmeliyiz:

```
$: bin/rails db:migrate
== 20161016000241 CreateArticles: migrating ==========================
-- create_table(:articles)
   -> 0.0012s
== 20161031000241 CreateArticles: migrated (0.0013s) ==================
```

Bu komut sayesinde config/database.yml dosyası içerisinde belirtilen veri tabanı oluşturuldu ve içerisine *articles* tablosu eklenmiş oldu. Kullandığımız iskele üreticisi çıktısında *resource_route* isimli bir çağrı daha görmekteyiz. Tahmin edileceği gibi bu çağrı, birazdan inceleyeceğimiz gibi ilgili kontrolör içerisindeki eylemlerin hangi URL'ler ile eşleşeceğinin ayarlanmasına ilişkin rotaların oluşturulmasını amaçlamaktadır. İlgili dosyayı açtığımızda içerisine sadece bir yeni satırın eklendiğini görürüz:

```
# config/routes.rb
Rails.application.routes.draw do
  resources :articles
  get 'welcome/index'
end
```

Bu tek satırlık kod, klasik kayıt oluşturma-okuma-güncelleme-silme işlemlerine ilişkin sekiz adet rotanın tanımlanarak, kontrolör/eylem ilişkisinin kurulmasını sağlamaktadır. Uygulamamızın kullandığı tüm rotaları görüntülemek için terminalde "bin/rails routes" komutunu çalıştırabiliriz:

```
$: bin/rails routes
         Prefix  Verb     URI Pattern    Controller#Action
         articles GET     /articles(.:format)  articles#index
                  POST    /articles(.:format) articles#create
     new_article GET     /articles/new(.:format)    articles#new
     edit_article GET    /articles/:id/edit(.:format) articles#edit
         article GET     /articles/:id(.:format)     articles#show
               PATCH     /articles/:id(.:format)     articles#update
                 PUT     /articles/:id(.:format)     articles#update
              DELETE     /articles/:id(.:format)     articles#destroy
welcome_index GET /welcome/index(.:format)   welcome#index
```

Üreticinin sonraki adımında kontrolör dosyasının oluşturulduğunu görmekteyiz ancak bu sefer boş metotlar yerine her metoda ilişkin Ruby kodları da dosyada yer almaktadır. Kontrolör sınıfımızı içeriğini *app/controllers* dizininde *articles_controller.rb* dosyasını açarak görebilirsiniz. Buradaki kodlar bir article nesnesin oluşturması, görüntülenmesi, güncellenmesi ve silinmesine dair gerekli olan tüm işlevselliği uygulamamıza kazandırmaktadır. Uygulamamızın görünümlerine ilişkin dosyalar da yine üretici tarafından sağlanmıştır. Tüm bu dosyalar da *app/views/articles* dizini içeresinde görüntülenebilir. Şu anda kullanılabilir durumda olan uygulamamızda yeni bir kayıt oluşturmak için http://localhost:3000/articles/new adresine, tüm kayıtları listelemek için http://localhost:3000/articles adresine vs. gidebiliriz.

Rails uygulamamızla ilgili son olarak *assets/* dizininden bahsedeceğiz. Bu dizinde uygulamamıza ait JavaScript dosyaları, CSS dosyaları ve resim dosyalarının yer almaktadır. Rails, varsayılan olarak JavaScript yazmamız amacıyla CoffeeScript ve CSS yazmamız amacıyla da SASS dilini kullanmaktadır. Tabi ki bu bir zorunluluk değil, tercih meselesidir ve kolaylıkla pasif duruma getirilebilir.

> Ruby ve Rails ile ilgili daha detaylı bilgi edinmek ve profesyonel uygulamalar geliştirebilmek için bu konudaki Türkçe kaynaklar arasında en kapsamlı olan Ruby ve Ruby on Rails kitaplarını inceleyebilirsiniz.

Öğrenme Çıktısı

4 Ruby dilinin temel teknik özelliklerini göz önüne alarak ne tür uygulamalarda kullanılabileceğini ifade edebilme

Araştır 4

Bir kullanıcının uygulamanızda var olmayan bir rotaya istek gönderdiğini düşünün. Bu durumda neler olacaktır? Bu sorunu çözmek için neler yapabilirsiniz?

İlişkilendir

Rails ile beraber gelen hazır kod dosyaları olan üreticileri (generator) ile diğer internet tabanlı programlama dillerindeki çatı, paket ve kütüphanelerle ilişkilendirerek karşılaştırınız.

Anlat/Paylaş

Ruby on Rails ile bir internet tabanlı rails uygulaması oluşturup bu süreçteki deneyimlerinizi paylaşınız.

1 Ruby dilinin geliştirilme amaçlarını, temel felsefi prensiplerini de göz önüne alarak açıklayabilme

Ruby Nedir ve Ne İşe Yarar?

Ruby dilinin temel felsefi prensipleri "İnsan Faktörü", "Asgari Sürpriz", "Özgürlük ve Rahatlık" ve "Maksimum Eğlence" şeklindedir. Aslında özüne bakılacak olursa bütün bu prensiplerin insan ya da daha özel bir grubu işaret edersek programcıların hayatını kolaylaştırmak ve verimliliklerini arttırmaya dayalı olduğunu görmekteyiz. Ruby dilinin geliştiricisi olan Matz, diğer dillerin insanlar yerine makineleri önemsediklerini ve bu yüzden kendisine hitap etmediklerini ve bu duruma paralel olarak kendi programlama dilini geliştirmeye başladığını belirtmektedir. Bunu yaparken de tekerleği yeniden icat etmek yerine hali hazırda var olan dillerin kendisine hitap eden yönlerini de bu yeni dile dahil etmiştir. Bu amaçla ondan fazla farklı programlama dilinden etkilenmiştir. Ruby'nin bu yaklaşımı programlama dünyasında beğeni görmüş ve en az etkilendiği diller sayısınca başka dillerin geliştirilmesine de katkıda bulunmuştur.

2 Ruby dilinin temel söz dizimi kuralları ve kullandığı yapıları açıklayabilme

Ruby Dili Söz Dizimi ve Bileşenleri

Ruby dili, noktalı virgül ve gereksiz parantezler gibi söz dizimi öğelerinin kullanımının isteğe bırakılması, boşlukların görmezden gelinmesi sayesinde istenilen noktadan kod yazmaya başlanabilmesi ve uzun ifadelerin kolayca birden fazla satıra bölünebilmesi, neredeyse İngilizce metin yazar gibi kod yazmaya ve okumaya imkan tanıması sayesinde oldukça esnek bir söz dizimine sahiptir. Tüm programlama dillerinde yer alan basit ve birleşik veri yapılarına kendine has sözlük (hash), aralık (range), bloklar gibi farklı yapıları da ekleyerek dilin kullanışlılığını ve zenginliğini arttırmaktadır. Döngü oluşturma, akış kontrolü sağlama ve nesne yönelimli programlama öğeleri açısından bakıldığında da diğer birçok dile oranla oldukça zengin bir yapıdadır.

3 Ruby on Rails çatısının temel çalışma prensiplerini açıklayabilme

Ruby On Rails Çatısı

Rails, kendi kendini tekrarlama (DRY = Don't-Repeat-Yourself) ve konfigürasyon üzerine kabuller (CoC = Convention-over-Configuration) prensiplerini temel alan, tek başına internet tabanlı uygulamalar geliştirmek için gerekli olan her şeyi sunan bir web uygulama geliştirme çatısıdır. MVC yazılım mimarisi desenini temel almaktadır. Bu mimari ardındaki ana düşünce; yazılımın kullanıcıya hitap eden bölümü ile arka plandaki işlemlerin birbirinden ayrı bölümler olarak ele alınmasına dayanmaktadır. M harfi model yani veri uygulamaya özel verilerin ve uygulamanın iş mantığına uygun biçimde bu veriler arasındaki ilişkilerin, erişim ve düzenleme kurallarının bulunduğu bölümdür. V harfi yani görünüm (view), uygulamanın kullanıcılarına sunulacak olan arayüz elemanlarını içeren yani kullanıcıların gördüğü bölümdür. C harfi ise kontrolör (controller) olup, kullanıcıdan gelen isteklerle ilgili işlemleri gerçekleştirmek ve yine kullanıcıya yanıt vermek üzere gerekli görevleri yerine getiren bölümdür. Çok sayıda bileşenin bir araya gelmesiyle oluşan Rails'in temel olarak çalışma prensibi şu adımlarla açıklanabilir. Rails, *Action Dispatch* bileşeniyle kullanıcının isteklerini alıp kontrolü uygun kontrolör bileşenine yani Action Controller'e devreder. İstek veri tabanı ile iletişim gerektiriyorsa kontrolör, *Active Record* bileşeni aracılığıyla veri tabanı işlemlerinin yapılmasını sağlar. *Action Controller* daha sonra kontrolü *Action View* bileşenine devrederek görünümlerin oluşturulmasını sağlar. Oluşturulan görünümler tekrar *Action Controller* bileşenine iletilir. *Action Controller*'ın kullanıcıya HTTP yanıtını göndermesiyle iletişim tamamlanır.

4 Ruby dilinin temel teknik özelliklerini göz önüne alarak ne tür uygulamalarda kullanılabileceğini ifade edebilme

Ruby On Rails ile Uygulama Geliştirme

Ruby dilinin teknik özelliklerinden bazılarını, bir uygulamanın geliştirilmesi konusunda yetersiz olabilme potansiyeli açısından incelediğimizde göze en çok çarpan onun yorumlanan bir dil olmasıdır. Yorumlanan dillerin, derlenen dillere oranla daha esnek olmaları nedeniyle bir miktar performans kaybının yaşanmasına sebep oldukları gerçektir. Sadece bu özelliği nedeniyle bile işletim sistemi, derleyici yazılımları ve çok yüksek trafikli uygulamalar yazmak için çok kullanışlı değildir. Bunun dışında Ruby ile hemen hemen her tür uygulama geliştirmek mümkündür.

1 Aşağıdakilerden hangisi Ruby programlama dilinin temel felsefi prensiplerinden birisi **değildir?**

A. İnsan Faktörü Prensibi
B. Çağdaş Tasarım Prensibi
C. Asgari Sürpriz Prensibi
D. Maksimum Eğlence Prensibi
E. Özgürlük ve Rahatlık Prensibi

2 I. Nesne Yönelimli Programlama

II. Fonksiyonel Programlama

III. Zorunlu (İmperatif) Programlama

IV. Yansıtıcı (Reaktif) Programlama

Yukarıdaki programlama paradigmalarından hangisi ya da hangileri Ruby dili tarafından desteklenmektedir?

A. Yalnız I
B. I ve II
C. I, III, IV
D. I, II, III, IV
E. II, ve IV

3 Ruby dili ile ilgili olarak söylenen aşağıdaki ifadelerden hangisi doğru **değildir?**

A. Dinamik ve zayıf tiplemeli bir dildir
B. Platformdan bağımsız bir dildir
C. Tamamen nesne yönelimli bir dildir
D. Temiz ve test edilebilir kod yazmayı destekler
E. Yorumlamalı bir dildir

4 Ruby dilinin aşağıdaki alanlardan hangisi için kullanılması **önerilmez?**

A. Robotik
B. Yapay Zeka
C. İnternet Tabanlı Uygulamalar
D. Ağ ve Telekomünikasyon Uygulamaları
E. İşletim Sistemleri

5 I. IRB Aracı

II. Ruby Yorumlayıcısı

III. Ruby Derleyicisi

Yukarıdaki araçlardan hangisi ya da hangileri kullanılarak Ruby kodlarının çalıştırılması sağlanabilir?

A. Yalnız I
B. I ve II
C. I ve III
D. II ve III
E. I, II ve III

6 Ruby dili söz dizimine ilişkin aşağıdaki ifadelerden hangisi doğrudur?

A. Parantez kullanımı her zaman zorunludur
B. Satır sonları her zaman noktalı virgül ile bitirilmelidir
C. Belirteçlerin isimlendirilmesinde büyük/küçük harf ayrımı yapılır
D. """ karakterleri çok satırlı yorum ifadeleri oluşturmak için kullanılır
E. Ruby kodları girintilere dikkat edilerek yazılmalıdır

7 Ruby dilinde kullanılan paket yönetici uygulaması hangi isimle anılmaktadır?

A. RubyPackages
B. RubyPkg
C. RubyPip
D. RubyGems
E. RubyPipMgr

8 "*Uygulamaya özel verilerin ve uygulamanın iş mantığına uygun biçimde bu veriler arasındaki ilişkilerin, erişim ve düzenleme kurallarının bulunduğu bölümdür.*"

Yukarıdaki tanımda hangi MVC tasarım deseni öğesinden bahsedilmektedir?

A. Model
B. Görünüm
C. Kontrolör
D. Vekil
E. Yönlendirici

9 Rails beraberinde uygulama geliştirme sürecini oldukça hızlandıran pek çok hazır komut dosyası ile birlikte gelir. Bu dosyalara genel olarak ne ad verilir?

A. Hızlandırıcılar
B. Kolaylaştırıcılar
C. Üreticiler
D. Yanıtlayıcılar
E. Emrediciler

10 Uygulamanın ihtiyaç duyduğu paketlerin tamamını tek seferde ve versiyon uyumsuzluğu gibi sorunlar yaşamadan yükleyebilmemizi sağlayan Ruby paketinin adı nedir?

A. Active Support
B. Active Report
C. Action Controller
D. ERB
E. Bundler

neler öğrendik yanıt anahtarı

1. B	Yanıtınız yanlış ise "Ruby Nedir ve Ne İşe Yarar?" konusunu yeniden gözden geçiriniz.
2. D	Yanıtınız yanlış ise "Ruby Nedir ve Ne İşe Yarar?" konusunu yeniden gözden geçiriniz.
3. A	Yanıtınız yanlış ise "Ruby Nedir ve Ne İşe Yarar?" konusunu yeniden gözden geçiriniz.
4. E	Yanıtınız yanlış ise "Ruby Nedir ve Ne İşe Yarar?" konusunu yeniden gözden geçiriniz.
5. B	Yanıtınız yanlış ise "Ruby Dili Söz Dizimi ve Bileşenleri" konusunu yeniden gözden geçiriniz.
6. C	Yanıtınız yanlış ise "Ruby Dili Söz Dizimi ve Bileşenleri" konusunu yeniden gözden geçiriniz.
7. D	Yanıtınız yanlış ise "Ruby Paket Yöneticisi (RubyGems)" konusunu yeniden gözden geçiriniz.
8. A	Yanıtınız yanlış ise "Ruby On Rails Çatısı" konusunu yeniden gözden geçiriniz.
9. C	Yanıtınız yanlış ise "Ruby On Rails ile Uygulama Geliştirme" konusunu yeniden gözden geçiriniz.
10. E	Yanıtınız yanlış ise "Ruby On Rails ile Uygulama Geliştirme" konusunu yeniden gözden geçiriniz.

8 Araştır Yanıt Anahtarı

Araştır 1

Ruby'nin tüm felsefi temelleri, merkezine insanı yani programcıyı oturtmuş olan prensiplere dayanmaktadır. Dilin tüm yapı taşları programcının daha hızlı ve üretken olması, kendini özgür ve rahat hissetmesi, sürprizlerle karşılaşmaması ve program yazarken eğlenceden uzak kalmaması gibi insani ilkeler üzerine kurulmuştur. Diğer pek çok dilin aksine makine yerine insanı temel alan bir yaklaşım sergiliyor olması Ruby'nin başarısının arkasındaki asıl nedendir.

Araştır 2

Ruby'nin yorumlanan ve dinamik bir dil olması onu, işlem hızı ve bellek tüketimi konularında C, C++ gibi bazı dillerden daha zayıf hale getirmiştir. İşlem hızının yüksek ve bellek tüketiminin az olması bir işletim sistemi için hayati önem taşıdığından; Ruby dili bu iş için doğru bir seçim olmayacaktır.

Araştır 3

"Kendini tekrarlama" anlamına gelen bu prensip kod yapısındaki parçaların ek, kesin ve güvenilir bir ifadesinin olması gerektiğini vurgular. Tekrar tekrar yazılmayan kod parçaları da kod yapısının daha sade, anlaşılır, daha iyi yönetilebilir, esnek ve az hatalı hale gelmesini sağlar.

Araştır 4

Kullanıcının uygulamada var olmayan bir URL adresini kullanarak istek göndermesi Rails yönlendiricisinin böyle bir URL ile ilişkilendirilmiş olan bir rota olmadığına dair Routing Error (rota hatası) mesajını üretmesiyle sonlandırılacaktır. Bu sorunun çözümü için ilgili rota config/routes.rb dosyasına eklenip, sonrasında kontrolör ve gerekliyse görünüm oluşturularak çözülebilir. Tabi ki rotayı varolan bir kontrolör ve eylem ile eşleştiriyorsak bu durumda rotanın routes.rb dosyasına eklenmesi yeterli olacaktır. Hem kontrolör, hem görünüm hem de rotanın aynı anda oluşturulması için *bin/rails generate controller kontrolör_ismi eylem_ismi* şeklinde bir üreticiyi çalıştırabiliriz.

Kaynakça

Bağdat, S. (2013). *Ruby*, Dikeyeksen Yayınevi, İstanbul

Bağdat, S. (2014). *Ruby on Rails*, Dikeyeksen Yayınevi, İstanbul

Flanagan, D. ve Matsumoto, Y. (2008). *The Ruby Programming Language*, O'reilly Media Inc, USA

Özyılmazel, U. (2015). Ruby 101. https://vigo.gitbooks.io/ruby-101/content/, Erişim Tarihi: 25 Ekim 2016

internet kaynakları

Ruby Programlama Dili Resmi Sitesi: https://www.ruby-lang.org/en/