

PHP CODEIGNITER

GÖKHAN KANDEMİR

İÇİNDEKİLER

BÖLÜM 1: CODEIGNITER'A GİRİŞ	1
Codeigniter ve Framework Nedir?	2
Codeigniter(CI) Nedir?	2
Framework Nedir?	3
Neden Kullanılır?	3
MVC Yapısı Nedir ve Ne için Kullanılır?	4
Peki, MVC Nasıl Çalışır?	5
Klasör Yapısı	6
Codeigniter'ı Bilgisayara İndirmek	6
Model Nedir?	10
View Nedir?	11
Yeni Bir View Oluşturmak	14
Controller Nedir?	17
index() Metodu Nedir ve Ne İşe Yarar?	20
Tasarladığımız View'i Ekranda Göstermek	20
View Üzerinde Dinamik Veriler (Değişken) Göstermek	22
MVC URL Yapısı	26
.htaccess Dosyası Nedir?	28
Metot Parametresi Okumak	31
Metot Tanımı İçerisinde Parametre Belirterek	31
Uri Sınıfı Kullanarak Metot Parametresi Almak	34
Form Verilerini Almak	36
base_url Nedir ve Neden Kullanılır?	45
base_url Tanımlamak	46
Projeye Dışarıdan Dosya Dahil Etmek (CSS, JS, Image)	55
View'e JavaScript Dosyalarının Eklenmesi	58

Routes Nedir ve Nasıl Tanımlanır?	62
Sonuç ve Değerlendirme	68
Bölüm Sonu Sorular	69
Tanım Soruları	69
Boşluk Doldurmalı Sorular	70
Doğru/Yanlış Seçenekli Sorular	70
Uygulama Soruları	71
BÖLÜM 2: DATABASE İŞLEMLERİ	75
Database Nedir?	76
Database Bağlantısı Nasıl Yapılır?	76
Tablo Oluşturma	80
Veri Girişi	81
Active Record Nedir?	81
get() Metodu	83
where() Metotları	88
Birden Fazla Şart Ekleme İşlemi?	92
Birden Fazla where Metodu Ekleyerek	92
Array Oluşturup İçerisinde Şartı Belirtmek	94
Şart İçerisinde OR Kullanmak	95
like() Metotları	98
wildcard Konumu Belirleme	99
order_by() Metodu	101
limit() Metodu	105
query() Metodu	106
insert() Metodu	109
update() Metodu	114
delete() Metodu	117

Sonuç ve Değerlendirme	119
Bölüm Sonu Soruları	120
Tanım Soruları	120
Boşluk Doldurmalı Sorular	121
Doğru/Yanlış Seçenekli Sorular	122
Uygulama Soruları	122
Uygulamanın açıklaması	123
BÖLÜM 3: MODEL İŞLEMLERİ	125
Giriş	126
Model Nedir?	126
Metot Oluşturmak?	129
Controller Üzerinden Model Çağırarak?	130
Kayıt Listeleme İşleminin Model Üzerine Taşınması	131
Kayıt Ekleme İşleminin Model Üzerine Taşınması	132
Kayıt Güncelleme İşleminin Model Üzerine Taşınması	137
Daha Esnek Kayıt Güncelleme İşlemi için Parametre Tanımlaması	142
Kayıt Silme İşleminin Model Üzerine Taşınması	145
Sonuç ve Değerlendirme	149
Bölüm Sonu Soruları	150
Tanım Soruları	150
Boşluk Doldurmalı Sorular	150
Doğru/Yanlış Seçenekli Sorular	150
Uygulama Soruları	151

BÖLÜM 4: PERSONEL KAYIT UYGULAMASI	153
Giriş	154
Uygulamanın Açıklaması	154
Veritabanının Oluşturulması	154
Projenin Çatısının Oluşturulması	156
Codeigniter'ı İndirmek	156
Tabloya Bilgi Girmek	156
Konfigürasyonların Yapılması	158
1. base_url Tanımlama	158
2. autoload İçerisinde Tanımlama	159
3. Database Bağlantısının Yapılması	160
4. .htaccess Dosyası Tanımlama	161
Controller Dosyasının Oluşturulması	162
Model Dosyasının Oluşturulması	165
Config Dosyasında default_controller Tanımlama	167
Listeleme Sayfası	168
HTML Kodlaması	168
PHP Kodlaması	172
Kayıt Ekleme Sayfası	182
HTML Kodlaması	183
PHP Kodlaması	186
Kayıt Düzenleme Sayfası	193
HTML Kodlaması	193
PHP Kodlaması	205
Kayıt Silme Sayfası	212
HTML Kodlaması	213
PHP Kodlaması	213

Kayıt Sıralama	217
HTML Kodlaması	217
PHP Kodlaması	220
Sonuç ve Değerlendirme	224
Bölüm Sonu Soruları	224
Uygulama Soruları	224
BÖLÜM 5: GELİŞMİŞ CODEIGNITER ÖZELLİKLERİ	227
Helper Nedir ve Neden Kullanılır?	228
Session Kullanımı	231
Session'a Veri Atmak	232
Session'dan Veri Almak	234
Session'daki Verileri Listelemek	236
Session'dan Veri Silmek	238
Sunucuya Dosya Aktarımı (Upload)	240
HTML Kodlaması	240
Upload Sınıfı Kullanımı	243
Doğrulama Kodu Oluşturma (Captcha Kodu)	250
Sayfa Bulunamadı Sayfası (Error Page) Yapımı	261
E-Posta Gönderme İşlemi	265
Sonuç ve Değerlendirme	274
Bölüm Sonu Soruları	274
Tanım Soruları	274
Boşluk Doldurmalı Sorular	275
Doğru/Yanlış Seçenekli Sorular	277
Uygulama Soruları	278
Araştırma Soruları	279

BÖLÜM 6: PERSONEL KAYIT UYGULAMASI	281
Personel Kayıt Uygulaması	282
Uygulamanın Açıklaması	282
Kullanıcı Tablosu Oluşturmak	283
Tabloya Bilgi Girmek	284
Projenin Çatısının Oluşturulması	285
Kullanılacak Dosyaların Oluşturulması	285
Helper Oluşturmak	285
Helper Dosyasını Autoload İçerisinde Tanımlama	285
Controller Dosyasının Oluşturulması	286
Model Dosyasının Oluşturulması	287
Bootstrap Framework'ünü İndirme	288
HTML Kodlaması	290
Personel Listeleme Sayfası	290
Personel Ekle Sayfası	294
HTML Kodlaması	295
PHP Kodlaması	300
Personel Listeleme Düzenlemesi	306
Personel Düzenle Sayfası	307
HTML Kodlaması	308
PHP Kodlaması	313
Alert İşlemleri	321
Giriş Sayfası	330
HTML Kodlaması	331
PHP Kodlaması	340
Routes Dosyasında Default_controller Tanımlama	348
Sonuç ve Değerlendirme	359
Bölüm Sonu Soruları	360
Uygulama Soruları	360
Dizin	364

1

CODEIGNITER'A GİRİŞ

BU BÖLÜMDE

Codeigniter ve Framework Nedir?	2
MVC Yapısı Nedir ve Ne için Kullanılır?	4
Klasör Yapısı	6
Model Nedir?	10
View Nedir?	11
Controller Nedir?	17
View Üzerinde Dinamik Veriler (Değişken) Göstermek	22
MVC URL Yapısı	26
.htaccess Dosyası Nedir?	28
Metot Parametresi Okumak	31
Form Verilerini Almak	36
base_url Nedir ve Neden Kullanılır?	45
Projeye Dışarıdan Dosya Dahil Etmek (CSS, JS, Image)	55
View'e JavaScript Dosyalarının Eklenmesi	58
Routes Nedir ve Nasıl Tanımlanır?	62
Sonuç ve Değerlendirme	68
Bölüm Sonu Sorular	69

Bu bölümde Codeigniter ve Framework kavramlarından, MVC yapısının ne olduğu ve nasıl çalıştığından bahsedeceğiz. Ayrıca Codeigniter'in bilgisayara nasıl indirileceğini, klasör yapısının nasıl olduğunu ve çalışma prensibi hakkında bilgi sahibi olacağız.

Daha sonra Model View Controller (MVC) yapısının modüllerinin neler olduğuna, nasıl oluşturulduğuna ve nasıl kullanıldığına değineceğiz.

Ek olarak; .htaccess dosyasının oluşturulmasını, form verilerinin okunmasını, CSS ve JavaScript dosyalarının Codeigniter projesi içerisinde kullanımını, routes tanımlamalarını uygulayıp, sizi bölüm sonunda hazırlamış olduğum alıştırmalar ile baş başa bırakacağım.

CODEIGNITER VE FRAMEWORK NEDİR?

CODEIGNITER(CI) NEDİR?

Codeigniter, web projelerimizi yapmamızı sağlayan çok popüler bir PHP framework'üdür. Yani yapısında PHP kullanılır. Bundan dolayı Codeigniter'ı anlamak için PHP bilmek gerekmektedir. Bu kitap içerisinde maalesef PHP'nin temel kavramlarına değinmeyeceğiz.

Codeigniter kurulum dosyası bulunan, bilgisayarınıza yükleyeceğiniz sıradan bir yapı değil. Belki böyle düşünmüş olabilirsiniz fakat maalesef tahmin ettiğiniz gibi değil. Codeigniter, bir **PHP projesidir**. İçerisinde birçok PHP dosyası mevcuttur. Az önce bahsetmiş olduğumuz binlerce fonksiyon bulunan PHP dosyaları. Codeigniter kullanmak demek aslında bu framework içerisinde bulunan dosyalar, fonksiyonları ve benzerlerini kullanmak demektir. Yapacağımız projeleri Codeigniter'ın bize sağlamış olduğu imkanlarla yapmak demektir.

Codeigniter, bir ara geliştirilmesi durdurulmuş ve daha sonra tekrardan **EllisLab** tarafından geliştirilmeye devam edilmiş ve halen geliştirilmesi için çalışmalar yapılan bir framework'tur. Şuan aktif olarak 3.x versiyonları kullanılmaktadır. Bu kitap içerisinde de 3.x versiyonunu kullanacağız. Eğer Codeigniter gibi bir framework kullanırsak bize sağlayacağı başlıca faydaları şunlardır;

- » Güvenlik Kontrolü
- » Kolay şekilde kullanabileceğimiz SESSION ve COOKIE yapıları,
- » Database işlemlerini kolaylaştıran Active Record aracı,
- » \$_GET ve \$_POST gibi veri transferi açısından önemli olan değişkenleri kullanımı oldukça kolay sınıfı,
- » Form validation yapıları,
- » Cache mekanizmaları,
- » Doğrulama kodları için hazır captcha sınıfı,
- » Dosya upload işlemleri için hazır sınıflar,
- » pagination sınıfı,
- » Sayfaları sıkıştırma algoritmaları ile oluşturulmuş hooks yapıları,
- » String sınıfları,
- » Sayı sınıfları,
- » Crypto sınıfları (şifrelemeler için),
- » Migration yapıları (veritabanı oluşturmak için kullanılan yapılar),
- » Log sınıfları,

- » Dosya okuma sınıfları (XML, XLS...) gibi daha bu listeyi oldukça uzatabileceğimiz birçok özelliği sıralayabiliriz.

Codeigniter'ın bize sunmuş olduğu birçok özelliğin yanında **Open Source** (Açık Kaynak) olduğu için birçok yazılım geliştirici tarafından da farklı sınıflar (modül) yazılmaktadır.

FRAMEWORK NEDİR?

Kabaca anlatılırsa framework; bir proje içerisindeyken işlemlerimizi kolaylaştırmak için, kendimize ait birçok kod bloğu yani fonksiyon ya da sınıf yazarız. Bunlara örnek vermek gerekirse;

- » database sorguları,
- » dosya sistemi okuması,
- » güvenlik için belirli kontroller,
- » form okuma işlemleri,
- » SESSION kontrolleri gibi örnekler denk gelebilir.

Aynı mantıktan yola çıktığımız zaman bir framework içinde; o framework'ün geliştiricileri tarafından, bir proje içerisinde yazılımcının ihtiyacı olabilecek temel ve gelişmiş; fonksiyonlar, modüller, kontrol sistemleri, güvenlik önlemleri gibi arttırabileceğimiz yazılım parçaları kodlanmıştır.

Framework'ü kullanan yazılımcıya ise sadece bunları kullanmak yani kendi sistemine adapte etme kalmaktadır.

Örnek vermek gerekirse; Bootstrap bir CSS Framework'üdür. Biz projelerimizde bu framework'ü kullanarak oldukça kolay bir şekilde responsive ve mobil uyumlu tasarımlar yapabilmekteyiz. Eğer bootstrap kullanmıyor olsaydık, tüm responsive ve mobil uyumlu CSS kodlarını kendimiz yazıyor olacaktık.

NEDEN KULLANILIR?

Framework kullanmak zorunda değiliz, fakat kullanmazsak bize kazandırmış olduğu tüm avantajları kendimiz yapmak zorunda kalacağız. Örneğin; Bootstrap için düşünecek olursak tüm bu CSS içerisinde bulunan class tanımlamalarını kendimiz yazmak durumunda kalacağız.

Framework kullanmadığımızda;

- » Framework'ün bize sunduğu tüm kolaylıkları kendimiz yapacağız demektir.
- » Bunları yaparken tek perspektiften bakacağımız için karşılaşacağımız hata ve süreçsel sıkıntılar da tek perspektiften olacaktır. gibi iki temel durumla karşılaşırız.

VIEW ÜZERİNDE DİNAMİK VERİLER (DEĞİŞKEN) GÖSTERMEK

Bu konu başlığı altında Codeigniter ile View dosyalarına Controller üzerinden nasıl değişken gönderildiğini göreceğiz.

Controller üzerinde oluşturduğumuz bir değişkeni view üzerinde görmek istersek ne gibi bir işlem yapmalıyız? Bu konuda bunun üzerinde duracağız.

Şuana kadar yaptığımız ve gördüğümüz view tasarımlarında genel olarak statik veriler gördük yani sabit olarak HTML üzerinde yazmış olduğumuz <h1> gibi <p> gibi (elbette view içerisinde görmüş olduğumuz her element buna dahildir) elementleri projeyi çalıştırdığımızda görmüştük.

Burada yapacağımız işlem ise; Controller üzerinde oluşturmuş olduğumuz bir değişkeni, view üzerinde göstermek. Bunu yapmamız için ihtiyacımız olan adımlar nelerdir onu görmek.

Peki; buna neden ihtiyacımız var?

Biz birçok mantıksal işlemi controller içerisinde yapacağız. Bu işlemlerin bir sonucu olacaktır. Bazı sonuçları da, view üzerinde kullanıcıya sunmak isteriz. İşte bundan dolayı elde ettiğimiz sonuçları view'e göndermemiz ve view üzerinde bunları görüntülememiz gerekmektedir.

Değişken oluşturmak için **applications/controllers/Welcome.php** isimli controller'a gidelim ve `index()` metodu içerisinde birkaç değişken oluşturalım.

Aşağıdaki kodlamalar **application/controllers/Welcome.php** dosyası üzerine yapılmıştır.

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Welcome extends CI_Controller {

    public function index()
    {

        $sayi_1 = 10;
        $sayi_2 = 20;
        $toplam = $sayi_1 + $sayi_2;

        $this->load->view('test_view');

    }
}
```

MVC URL YAPISI

Bu kısımda Codeigniter ile MVC yapısına ait olan URL yapısını inceleyeceğiz.

MVC yapısının çalışma prensibi farklı olduğu gibi URL sistemi de farklıdır. MVC ile yapılan projelerde; eğer herhangi bir route (ileri konularda route konusunu detaylıca göreceğiz) tanımlaması yapılmamışsa **urunler.php**, **urun-listesi.php**, **hakkimizda.php** gibi dosya isimlerini göremeyiz. MVC yapısının URL yapısı aşağıdaki gibi düşünülmelidir.

http://www.siteadi.com/controllerAdi/metotAdi/parametre1/parametre2/parametreN

Bu şekilde tanımlanmış birçok URL ile günlük hayatta karşılaşıyoruz aslında. Şimdi bu gördüğümüz URL'leri nasıl çözeceğimizi öğrenelim.

Örnek, URL aslında kendini açıklıyor. Bir örnek senaryo ile ilerleyelim;

Sitemizin adı **www.yazilimegitim.net** olsun. Eğitim listesini göreceğimiz bir linke tıklıyoruz ve yukarıdaki URL bir anda değişiyor ve aşağıdaki gibi oluyor;

http://www.yazilimegitim.net/egitim/liste

yazilimegitim.net sitemin adı sonraki / (slash) karakterinden sonra gelen değer ise (yani eğitim) benim MVC yapımdaki controller dosyamıza denk gelmektedir. Bir sonraki / (slash) karakterinden sonra gelen değer ise controller içerisinde yer alan metot adına denk gelir (yani liste). Controller dosyasının görüntüsü aşağıdaki gibi düşünülebilir.

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Egitim extends CI_Controller {

    public function index(){

    }

    public function liste(){

        echo "Burada eğitimler listelenecektir.";

    }

}
```

Görüldüğü gibi **Egitim** adında bir class'ım var ve bu class'ın içerisinde **liste** isimli bir metodum var. **http://www.yazilimegitim.net/egitim/liste** bu şekilde istek

gönderdiğimiz zaman ilgili controller içerisindeki ilgili metod çağırılır ve o metod içerisinde yapılan işlemler gerçekleştirilir. Bizim senaryomuz için düşünersek; **Egitim** isimli controller dosyasının altında bulunan **liste** isimli metod çalıştırılacaktır ve ekrana echo komutu sayesinde **Burada eğitimler listelenecektir** yazacaktır.

Metod adından sonraki her / (slash) karakterinden sonra gelen değerler ise o metoda gönderilen parametreleri oluşturmaktadır (metod parametresi okumaları da detaylıca anlatmaya çalıştım).

Lokal üzerinde çalıştığımız için site adımız **http://localhost/kitap** olacaktır. Yukarıdaki bilgilere göre localhost üzerinde URL yapısını bir deneyelim.



Resimde görebileceğimiz gibi bize böyle bir dosyanın mevcut olmadığını söylüyor. Halbuki böyle bir controller ve metod da projemde bulunmaktadır. Çalışmamasının sebebi ise Codeigniter'in URL yapısının çalışma prensibi.

Codeigniter'in çalışma prensibinden dolayı bu URL yapısı birazcık farklı. İlk bölümde klasör yapısından ve burada ilk klasörün içerisinde bulunan **index.php** dosyasından bahsetmiştik. Codeigniter bütün işlemlerini bu dosya üzerinden yapmaktadır aslında. Yukarıda bahsettiğimiz URL yapısını bu dosya üzerinden gerçekleştirmektedir.

Örneğin, **http://localhost/kitap/egitim/liste** linkini ele alalım. Codeigniter projesinde bu URL'nin çalışması için controller ve metod adını **index.php** dosyasına parametre olarak göndermek gerekmektedir.

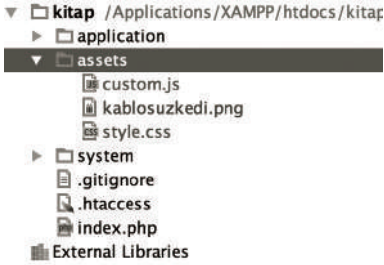
Linkimizi bu şekilde düzenleyip çalıştırdığımızda ise projemizin istediğimiz gibi çalıştığını görebiliriz.

PROJEYE DIŐARIDAN DOSYA DAHİL ETMEK (CSS, JS, IMAGE)

Bu kısımda Codeigniter projemize, proje dıŐarından bir dosyanın (CSS, JavaScript, Resim) nasıl eklendiĐini greceĐiz ve projemizi eklediĐimiz dosyalarla Őekillen-direceĐiz.

Őuana kadar yaptığımız hiŐbir kodlamada resim, stil ya da JavaScript dosyası ile alıŐmadık. Bunları elbette hazırlamıŐ olduĐumuz view dosyalarının ierisinde de yazabilirdik. Fakat bu blmde biz view ierisinde yazmayla deĐil dıŐarıda bulunan CSS, JavaScript ya da Resim dosyalarımızı view de nasıl kullanabiliriz onu greceĐiz.

Bunu yapabilmek iin ilk olarak projemin ana klasr ierisinde **assets** isimli bir klasr oluŐturuyorum. Elbette bu klasrn adını dilediĐiniz gibi Őeebilirsiniz. OluŐturduktan sonra ierisinde **style.css**, **custom.js** dosyaları oluŐturup bir tane de rastgele resim dosyası atıyorum.



Tanımlamaları yaptıktan sonra **style.css** dosyasında sayfamızı Őekillendirmek iin birkaç kodlama yapıyoruz.

```
body{  
    font-family: Arial;  
    background-color: #f0f0f0;  
}
```

Konuyu daha iyi anlayabilmek iin basit tuttum. Bu nedenle sayfamızın font tipini ve arkaplan rengini deĐiŐtirmek, Őuanlık deĐiŐikliĐi grmek aısından yeterli. Daha nce yapmıŐ olduĐumuz **personel_form** view dosyasına gidelim (application/views/personel_form.php). DıŐarından eklenen CSS ve JavaScript dosyalarını <head> blokları ierisinde tanımlıyoruz. Bu yzden tanımlamalarımızı bu blok ierisinde yapacaĐız.

VIEW'E RESİM DOSYALARININ EKLENMESİ

Şuana kadar dışarıdaki dosyaları kullanmış olduğumuz view dosyalarına eklemelerle ilgili bayağı bir yol aldık. Resim dosyalarının eklenmesi ise şuana kadar yapmış olduğumuz işlemlerden farklı değil. Aynı stil ve JavaScript dosyalarının çağırılma şekillerinde olduğu gibi resim dosyalarını gösterme işlemi de HTML etiketlerinden ibaret. Bunun için `` etiketini kullanıyoruz. Formumuzun başına bir resim eklemek için aşağıdaki kodlamayı yapalım.

```
<form action="<?php echo base_url("personel/kaydet"); ?>" method="post">
  " alt="">
  <div>
    <label for="personel_adi">Personel Adı</label>
    <input type="text" name="personel_adi" id="personel_adi">
  </div>
  <br>
  <div>
    <label for="email">E-posta adresi</label>
    <input type="text" name="email" id="email">
  </div>
  <br>
  <div>
    <label for="departman">Departman Adı</label>
    <select name="departman" id="departman">
      <option value="bilgi">Bilgi İşlem</option>
      <option value="yazilim">Yazılım</option>
      <option value="donanim">Donanım</option>
    </select>
  </div>
  <br>
  <div>
    <button type="submit">Kaydet</button>
  </div>
</form>
```

Burada sadece formumuzun içerisine `img` etiketini ekliyoruz ve yolunu belirtiyoruz.

```
" alt="">
```

2

DATABASE İŞLEMLERİ

BU BÖLÜMDE

Database Nedir?	76
Active Record Nedir?	81
get() Metodu	83
where() Metotları	88
like() Metotları	98
order_by() Metodu	101
limit() Metodu	105
query() Metodu	106
insert() Metodu	109
update() Metodu	114
delete() Metodu	117
Sonuç ve Değerlendirme	119
Bölüm Sonu Soruları	120

Bu bölümde, Codeigniter kullanarak Database bağlantısının nasıl yapıldığına değineceğiz. Active Record'un ne olduğunu anladıktan sonra, var olan bir database üzerinde Active Record cümleleri oluşturarak SQL işlemlerini yapacağız.

Bununla birlikte veri seçme işlemleri, veri ekleme işlemleri, veri düzenleme işlemleri, veri silme işlemleri yapacağız. Daha sonra sıralama, limitleme gibi SQL ile yaptığımız birçok işlemi yaptıktan sonra sizi bölüm sorularıyla baş başa bırakacağız.

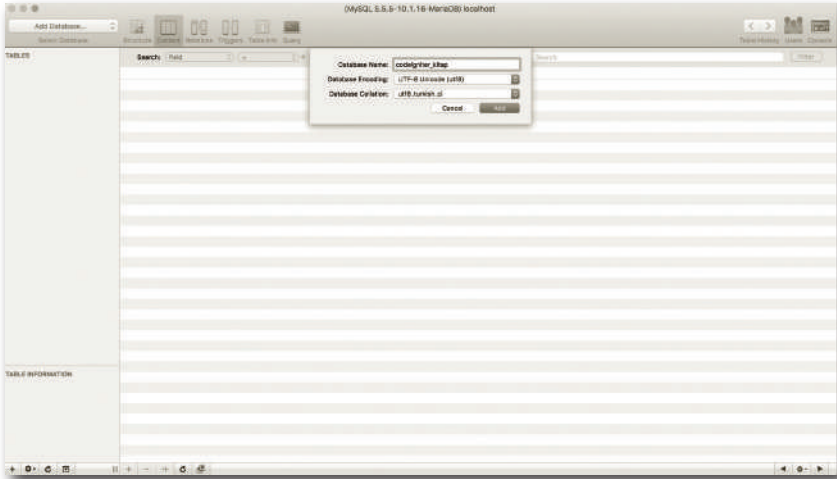
Şimdiden size iyi öğrenmeler.

DATABASE NEDİR?

Database yani veritabanı nedir? sanıyorum bu kitabı alıyorsanız bunun ne olduğuna dair bilginiz vardır. Eğer bu konu ile ilgili bir sıkıntı yaşıyorsanız öncesinde Veritabanı ile ilgili küçük bir araştırma yapmanızı öneririm. Ek olarak konu ile ilgili **kablosuzkedi** isimli YouTube kanalında **MySQL Eğitimi** isimli eğitim serimi izleyebilirsiniz. Bu başlığımız altında Codeigniter ile nasıl veritabanına bağlanırsınız ve ne yapmalıyız? Bu konular üzerinde yoğunlaşacağız.

DATABASE BAĞLANTISI NASIL YAPILIR?

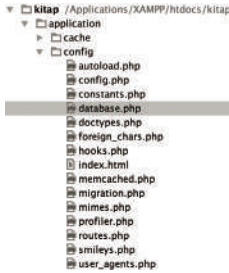
Codeigniter ile veritabanına bağlanmak istiyorsak ilk olarak bir veritabanımızın olması gerekiyor. Biz kitap içerisinde MySQL ile çalışacağız. Elbette Codeigniter çok daha fazla veritabanı türüne destek vermektedir. İlk olarak veritabanına bağlanmak için **codeigniter_kitap** isimli bir database oluşturalım. Ben veritabanı uygulaması olarak Mac üzerinde çalışan **Sequel Pro** isimli uygulamayı kullanmaktayım. Siz PHP MyAdmin, MySQL, Navicat gibi bu işi yapabileceğiniz birçok uygulamayı elbette kullanabilirsiniz.



Veritabanını oluştururken **Database Encoding** isimli alanı **UTF-8 Unicode (utf8)** ve **Database Collaction** isimli alanı **utf8_turkish_ci** olduğundan emin olalım. Çünkü kayıtlarımızda Türkçe karakter olacağı için ileride tablolarımızda sıkıntı vermesini istemeyiz.

Artık veritabanımız oluşmuş durumda henüz içerisinde bir tablo bulunmamaktadır. Tablo oluşturma işlemini ileride yapacağız.

Ben veritabanı sunucuma bağlanırken localhost (yani bu bilgisayar) olarak, kullanıcı adını ise root olarak belirledim. Herhangi bir şifre kullanmadım.



Şimdi veritabanımız hazır olduğuna göre codeigniter ile nasıl bağlanacağımıza bir bakalım. Bunun için `application/config/` klasörü altında bulunan `database.php` isimli dosyayı açıyorum.

`database.php` isimli dosyayı açtığınızda aşağıdaki gibi bir görüntü sizi bekliyor olacaktır.

```

101 // $db['default'] = array(
102 //     'hostname' => 'localhost',
103 //     'username' => 'root',
104 //     'password' => '',
105 //     'database' => 'database',
106 //     'charset' => 'utf8',
107 //     'collat' => 'utf8_general_ci',
108 //     'encrypt' => FALSE,
109 //     'compress' => FALSE,
110 //     'strict' => FALSE,
111 //     'failover' => array(),
112 //     'save_queries' => TRUE
113 // );
  
```

Burada bulunan alanlar bize çok fazla fikir vermektedir. Şimdi buradaki alanlar ne işe yarıyor biraz bundan bahsedelim.

```

$db['default'] = array(
    'dsn' => '',
    'hostname' => '',
    'username' => '',
    'password' => '',
    'database' => '',
  
```


Yine get metodunda karşılaşmış olduğumuz sonuca benzer bir sonuçla karşılaştık. Peki, sonucu nasıl alırız?

Aynı get metodunda yaptığımız gibi result() ekleyeceğiz fakat bu yukarıda yazmış olduğumuz şartın bize oluşturmuş olduğu SQL cümlesine bir göz atalım!

```
$sonuc = $this->db->where("personel_adi", "Gökhan Kandemir");
```

cümlesi bize;

```
WHERE personel_adi = "Gökhan Kandemir"
```

cümlesini üretir. Fakat bu SQL cümlesinde bu şart cümlesinin çalışması için bir SELECT cümlesine ihtiyacımız vardır. Bunu da get() metoduyla yaptığımızı söylemiştik. O zaman yazmış olduğumuz komutu tekrardan düzenleyelim.

```
$sonuc = $this->db->where("personel_adi", "Gökhan Kandemir")->get("personel")->result();
```

şimdi yazdığımız kodu bir analiz edelim;

```
$this->db->where("personel_adi", "Gökhan Kandemir")
```

komutu bize WHERE cümlesini oluşturuyordu. Bu şarta bir SELECT eklemek için sonuna;

```
->get("personel")
```

komutunu ekledik ve direk sonuçları getirmesi için yani kayıtları getirmesi için de;

```
->result();
```

komutunu ekledik. Böylece personel tablosundan personel_adi Gökhan Kandemir olan kayıtları getir. Yani bize;

```
SELECT * FROM personel WHERE personel_adi = "Gökhan Kandemir"
```

cümlesini oluşturmuş oldu. Bu düzenlemelere göre Controller'ın son hali aşağıdaki gibidir.

```

<?php
class Veritabani extends CI_Controller {
    public function index(){
        $liste = $this->db->get("personel")->result();
        print_r($liste);
    }

    public function where(){
        $sonuc = $this->db->where("personel_adi", "Gökhan Kandemir")->get("personel")->result();
        print_r($sonuc);
    }
}

```

Yine localhost/kitap/veritabani/where yazıp bize ne sonuç getirdiğine bakalım;

Bu sefer veritabanından Gökhan Kandemir isimli kaydı bize başarılı bir şekilde getirdi. Peki, şuan kullandığımız komuttan dolayı bize oluşan cümle;

```
SELECT * FROM personel WHERE personel_adi = "Gökhan Kandemir"
```

ifadesiydi. Burada dikkat etmenizi istediğim nokta ise bu şartın sağlanması için personel_adi alanının Gökhan Kandemir'e eşit olması durumu. Biz şartımız içerisinde;

- » maaşı 1000'den büyük/küçük olanları getir
 - » id numarası 5'den küçük olanları getir
- gibi bir şart yazmak istersek bu sefer iş farklı olacaktı.

```
$this->db->where("personel_adi", "Gökhan Kandemir")
```

şeklinde bir ifade yazdığımızda bize doğrudan ("=") bir eşitlik getiriyor. **Büyük-tür/küçüktür** ya da **eşit değildir** gibi bir ifade yazmak istersek sadece bu ifade üzerinde küçük bir değişiklik yapmamız yeterlidir.

```
$this->db->where("personel_adi !=", "Gökhan Kandemir")
```

şeklinde komutumu düzenleyip çalıştırdığımda;

3

MODEL İŞLEMLERİ

BU BÖLÜMDE

Giriş	126
Model Nedir?	126
Controller Üzerinden Model Çağırma?130	
Kayıt Listeleme İşleminin Model Üzerine Taşınması	131
Kayıt Ekleme İşleminin Model Üzerine Taşınması	132
Kayıt Güncelleme İşleminin Model Üzerine Taşınması	137
Daha Esnek Kayıt Güncelleme İşlemi için Parametre Tanımlaması	142
Kayıt Silme İşleminin Model Üzerine Taşınması	145
Sonuç ve Değerlendirme	149
Bölüm Sonu Soruları	150

Bu bölümde MVC yapısının en önemli parçalarından biri olan Model işlemlerinin ne olduğuna, çalışma prensibine ve nasıl kullanıldığına değineceğiz. Bir önceki bölümde yaptığımız SQL işlemlerini Model üzerinden yapacağız. Metod oluşturmak ve Controller üzerinden model çağırma işlemlerini örneklerle anlatmaya çalışacağız.

Bununla birlikte, parametrik model tanımlayıp, model üzerinden işlemlerimizi dinamik şekilde yapacağız. Bu işlemleri yaptıktan sonra sizleri bölüm sorularıyla baş başa bırakacağız.

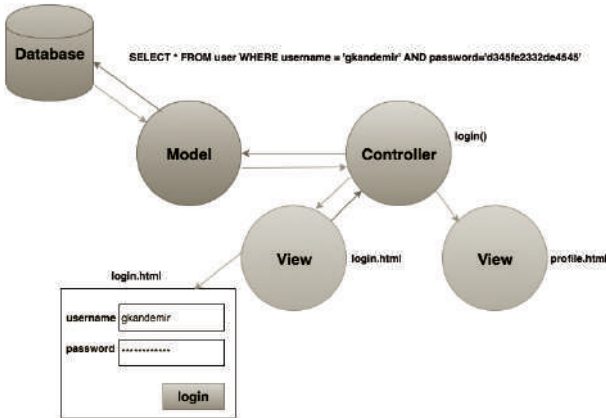
Şimdiden size iyi öğrenmeler.

Giriş

Veritabanı işlemlerinin Codeigniter ile nasıl kolay bir şekilde yapıldığını bir önceki bölümde gördük. Siz de dikkat ettiyseniz işlemlerimizi şuna kadar View ve Controller üzerinde yürüttük. Bu şekilde projemizi kodladığımızda sıkıntı çıkmadı elbette fakat bir MVC yapısı kullanıyoruz ve bunun mantığına göre hareket etmemiz açısından Model kullanmamız, MVC'nin yapısından dolayı gereklidir. Yani Framework'ü amacına göre kullanmak adına ve işlemlerimizi daha düzenli yapmamız adına Model kullanımını bu bölüm üzerinde göreceğiz ve tartışacağız.

MODEL NEDİR?

İlk bölümlerde bahsettiğimiz gibi Framework'lerin bize sağladığı avantajların başında Backend, Frontend ve SQL komutlarını ve dosyalarını birbirlerinden ayırması gerekiyordu. Şuna kadar HTML, CSS, JavaScript komutlarımızı View üzerinde, PHP komutlarımızı ise Controller üzerinde tanımladık. Fakat bir önceki bölümde veritabanı ile ilgili tüm işlemlerimizi controller üzerinden yaptık. Bunun amacı Controller mantığını daha iyi oturtmak ve Active Record'u daha iyi anlayabilmek içindi. Şimdi ise, controller içerisinde yaptığımız tüm Active Record işlemlerini Model'e tanımlayıp controller üzerinden sadece Model'i kullanarak tüm veritabanı işlemlerimizi halledeceğiz. MVC yapısından bahsederken Controller'ın Model'i kullandığından bahsetmiştik.

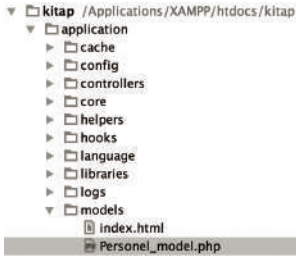


Yani controller'ın veritabanı ile bir işi olduğunda model'e soracak ve model de bu işlemin sonucunu controller'a geri gönderecekti. Daha sonra Controller model üzerinden gelen bu cevaba göre nasıl bir işlem yapmak istiyorsa yapacaktı (ilgili view dosyasını ekrana basacak ya da belirli işlemler yapması gibi).

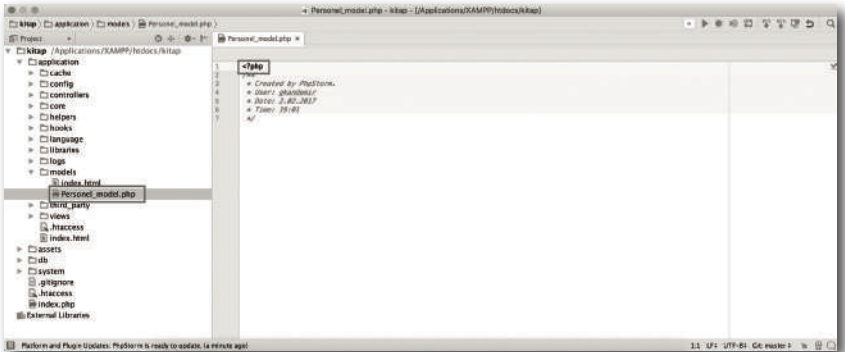
Şimdi bu işlemleri yapmak için projeye bir tane model dosyası ekleyelim. Bunu yapmak için **application/models/** klasörünün altında bir tane model dosyası oluşturalım. Veritabanında **Personel** tablosu oluşturmuştuk bu tablo ile ilgili tüm işlemlerimizi bu oluşturacağımız model dosyası içerisinde yapacağız. Bunun için bu model dosyasının **Personel** tablosuna ait olduğunu belirtmemiz açısından, oluşturacağımız model dosyasının adını **Personel_model** olarak belirliyorum. Eğer başka bir tablomuz daha olsaydı (örneğin **Ogrenci**) bu tablo için de ayrı bir model dosyası belirliyor olacaktım **Ogrenci_model** gibi.

NOT

Model dosyasının ilk harfini Controller dosyasında olduğu gibi **büyük harfle** yazmalıyız. Aksi takdirde Codeigniter bize sorun çıkartacaktır.



daha önce de bahsettiğimiz gibi Codeigniter için temel modül olan tüm dosyalar PHP dosyası olmalıdır. Bundan dolayı oluşturduğumuz model dosyası da **.php** uzantılı olmalıdır.



5

GELİŞMİŞ CODEIGNITER ÖZELLİKLERİ

BU BÖLÜMDE

Helper Nedir ve Neden Kullanılır?	228
Session Kullanımı	231
Sunucuya Dosya Aktarımı (Upload)	240
Doğrulama Kodu Oluşturma (Captcha Kodu)	250
Sayfa Bulunamadı Sayfası (Error Page) Yapımı	261
E-Posta Gönderme İşlemi	265
Sonuç ve Değerlendirme	274
Bölüm Sonu Soruları	274

Bu bölümde,

Codeigniter'da sıklıkla kullandığımız modüllere ve özelliklere değineceğiz. Eğlenceli bir bölüm olacağını düşünüyorum. Lafi fazla uzatmadan bu bölüm içerisinde göreceğimiz konulara göz atalım;

- » Helper nedir ve Nasıl kullanılır?
- » SESSION kullanımı
- » Sunucuya dosya aktarımı (upload)
- » Doğrulama kodu oluşturma (captcha)
- » Sayfa bulunamadı sayfası yapımı (404 Error page)
- » E-posta gönderme işlemi (email)

Bütün bu konuları işledikten ve uyguladıktan sonra sizi bölüm sorularıyla baş başa bırakacağız.

Şimdiden size iyi öğrenmeler!

HELPER NEDİR VE NEDEN KULLANILIR?

Şuana kadar yaptığımız tüm mantık ve işlem içeren dosyalar (yani controller ve model dosyaları) birer OOP mantığının temeli olan class dosyalarıydı. Genel olarak tüm mantık ve operasyonel işlemleri Controller içerisinde yaptık (ekle, sil, düzenle, sırala gibi). Peki, class dosyası olmadan codeigniter ile bir işlem yapmak istersen o zaman ne yapmalıyız?

İşte burada helper dosyaları devreye giriyor. Buna örnek vermek gerekirse `base_url()` komutunu kullanarak sitemizin yolunu rahatlıkla alabiliyorduk. Codeigniter herhangi bir controller yardımı olmadan bunu nasıl yapıyor? Bu sorunun cevabı **helper**. Sitemizin yolunu alabilmek için kullandığımız `base_url()` komutu aslında bir **helper** dosyası içerisinde bulunan fonksiyon. Biz bu fonksiyonu kullanarak sitemizin yolunu kolaylıkla elde edebiliyoruz.

İlk bölümden hatırlayacaksınız; `base_url()` komutunu kullanabilmek için **application/config/autoload.php** dosyası içerisinde helper tanımlaması yapıyorduk.

```
$autoload['helper'] = array("url");
```

burada `base_url()` fonksiyonunu içeren helper dosyasının adı `url` idi. Biz bu helper dosyasını tanımladığımızda `base_url()` fonksiyonunu rahatlıkla kullanabiliyorduk. İşte burada herhangi bir `class` dosyası olmadan sadece PHP fonksiyonlarını kullanarak bunu yapabiliyorduk.

Codeigniter içerisinde biz de kendi helper dosyalarımızı üretebiliyoruz. İçerisinde de fonksiyon/lar oluşturduktan sonra **autoload.php** dosyası içerisinde bu helper dosyasını tanımladığımızda; istediğimiz yerden (Controller, Model, View) bu fonksiyonları çağırıp, kullanabiliriz.

Şimdi kendimize ait bir helper dosyası oluşturalım. Bunu yapmak için **application/helpers/** klasörüne gidelim ve `tools_helper` isimli bir dosya oluşturalım. Elbette siz yine istediğiniz ismi verebilirsiniz fakat; buradaki en önemli nokta, oluşturacağınız dosyanın `_helper` ile bitmesidir. Codeigniter bu dosyanın helper olduğunu `_helper` ekinden anlamaktadır.

NOT

Bu bölümün tüm işlemlerini de ilk bölümde oluşturmuş olduğumuz kitap isimli klasör içerisinde bulunan codeigniter projesi içerisinde yapıyoruz.



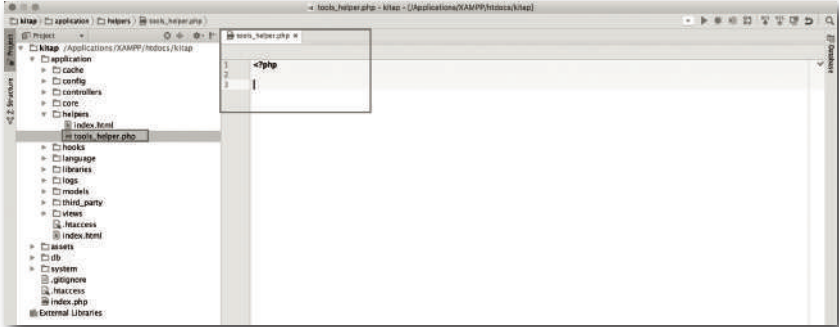
Dosyamızı oluşturduğumuza göre **application/config/** klasörü altında bulunan **autoload.php** dosyasına giderek bu helper dosyamızı kullanmak için tanımlama yapalım.

```
Şautoload['helper'] = array("url");
```

Bu satırı bularak **tools_helper** isimli dosyamızı otomatik olarak yüklenecek şekilde tanımlayalım.

```
Şautoload['helper'] = array("url", "tools");
```

Gördüğünüz gibi burada tanımlama yaparken **_helper** diye eklediğimiz ekin öncesini buraya yazıyoruz. Artık **tools_helper** dosyamızı kullanabiliriz. Fakat içerisinde herhangi bir fonksiyon tanımlı değil.



İlk olarak bir fonksiyon oluşturalım ve içerisine test amaçlı birkaç kodlama yapalım.

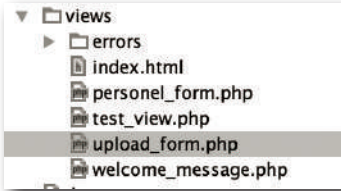
```
<?php\nfunction tarih_getir(){\n    echo date("d M Y");\n}\n?>
```

SUNUCUYA DOSYA AKTARIMI (UPLOAD)

Web sayfalarında en çok yaptığımız işlemlerden biri de sunucuya dosya aktarımıdır. Örnek olarak sosyal medya hesaplarımızdaki profil resimlerimiz, e-posta gönderirken, postamıza ek olarak eklediğimiz dosyalar gibi birçok kullanım yeri vardır hayatımızda. Codeigniter’da bu işlemi yapmak oldukça kolay. Lafı uzatmadan hemen başlayalım bu eğlenceli işlemimize.

HTML KODLAMASI

İlk olarak kullanıcıdan veri almamızı sağlayacak bir HTML tasarımına ihtiyacımız var. MVC yapısında HTML tasarımlarımız View dosyasına denk geldiği için ilk olarak bir view dosyası ekleyelim. Bunu yapmak için `application/views/` klasörü altına gidelim ve `upload_form.php` isimli bir view dosyası oluşturalım.



Şimdi `upload_form.php` isimli view dosyamızın içerisine HTML kodlarımızı yazalım. Peki, bu veri alma işlemi sağlayan elementin adı neydi? Elbette `<form>` elementi. O halde bizim bir forma ihtiyacımız var.

```
<!doctype html>
<html lang="tr">
<head>
  <meta charset="UTF-8">
  <title>Upload Form</title>
</head>
<body>
<form action="" method="post" enctype="multipart/form-data">
<input type="file" name="userFile" placeholder="Upload Edilecek Dosyayı Seçiniz">
  <button type="submit">Yükle</button>
</form>
</body>
</html>
```

Yukarıda gördüğünüz gibi bir <form> elementi oluşturup içerisine dosya yüklemeyi gerçekleştirmek için <input /> elementi ekledik. Bu elementin name alanının da dolu olduğuna dikkat ediniz. Çünkü form gönderilirken yani submit edilirken eğer bu name alanını doldurmazsak hiçbir veri sunucuya gönderilmez. Ek olarak da formu göndermek için type alanı submit olan bir button elementi ekliyoruz.

NOT

Formun action alanının henüz boş olduğuna dikkat edelim. Bu alanı az sonra formun gönderileceği metodu belirledikten sonra dolduracağız. Dosya yüklemek için dikkat etmemiz gereken 2 durum vardır.

Bunlardan birincisi; eğer bir dosya göndermek istiyorsak, o formun enctype alanının multipart/form-data olması gerekir. Aksi takdirde göndermek istediğimiz dosya sunucuya gönderilmez. Diğer önemli durum ise; input elementinin type alanının file olması gerekir. Böylece bize dosya seçmemiz için gerekli elementleri kendisi oluşturmuş olacaktır.

Şimdi HTML formunu oluşturduğumuza göre bu formu ekrana getirmek için bir metoda o metodu da içerecek bir controller dosyasına ihtiyacımız var. O halde **application/controllers/** klasörünün altına gidip **UploadTest.php** isimli bir PHP dosyası oluşturalım.

```

▼ controllers
  ● Baseurl.php
  ● Egitim.php
  ● Helper.php
  📄 index.html
  ● Model.php
  ● Personel.php
  ● Sessiontest.php
  ● Test.php
  ● Uploadtest.php
  ● Veritabani.php
  ● Welcome.php

```

Şimdi bu oluşturmuş olduğumuz PHP dosyasının Codeigniter tarafından bir controller olarak görülebilmesi için ilk olarak class ve daha sonra extends tanımlamalarımızı yapalım.

```

<?php
class Uploadtest extends CI_Controller {
}

```

6

PERSONEL KAYIT UYGULAMASI

BU BÖLÜMDE

Personel Kayıt Uygulaması	282
Kullanıcı Tablosu Oluşturmak	283
Model Dosyasının Oluşturulması	287
Bootstrap Framework'ünü İndirme	288
Personel Listeleme Sayfası	290
Personel Ekle Sayfası	294
Personel Listeleme Düzenlemesi	306
Personel Düzenle Sayfası	307
Alert İşlemleri	321
Giriş Sayfası	330
Routes Dosyasında Default_controller	
Tanımlama	348
Sonuç ve Değerlendirme	359
Bölüm Sonu Soruları	360
Dizin	364

Bu bölümde 4. Bölümde yapmış olduğumuz Personel Listesi uygulamasını birazcık zenginleştirip, birkaç bölüm daha ekleyeceğiz.

Bu ekleyeceğimizi bölümler ise 5.Bölüm içerisinde görmüş olduğumuz konuları temel alacaktır. İlk olarak Bootstrap framework'ünü projemize dahil edip tüm view dosyalarımızı daha görsel hale getireceğiz.

5. Bölümde görmüş olduğumuz konulardan yola çıkarak; personel için bir resim ekleme işlemi yapacağız. Yaptığımız database işlemlerinin sonuçlarını ekranda bir mesaj kutusu içerisinde göstermek için alert sistemi ekleyip en sonda projeye girebilmek için doğrulama kodu eklenmiş bir kullanıcı giriş sistemi ekleyeceğiz. Daha sonra sizi bölüm sorularıyla başa bırakacağız.

Şimdiden size iyi öğrenmeler.

PERSONEL KAYIT UYGULAMASI




UYGULAMANIN AÇIKLAMASI

4. Bölüm içerisinde yapmış olduğumuz Personel Listesi uygulamasına artık bir giriş ekranı sayesinde giriş yapabiliyor olacağız. Bunun yanında personelleri girerken personele ait olan bir de fotoğraf girmesini sağlayacağız. Bu uygulama içerisinde, 5. Bölüm içerisindeki gördüğümüz birçok konuyu kullanacağız. Bu yüzden bu uygulama önceki bölümün tekrarı ve pekiştirmesi açısından mükemmel bir tekrar niteliğindedir.

Bu uygulamanın sahip olacağı modüller;

- » Kullanıcı giriş sayfası
 - √ Doğrulama kodu ile beraber kullanıcıya bir login ekranı gelecektir.
- » İşlemlerin durumunu belirten bir alert yani uyarı sistemi olacaktır.
- » Personel girişi sırasında kullanıcı personelin fotoğrafını yükleyebilecek

Bu sefer 4. Bölümün aksine direk görselliği de işin içine katacağız. Bu yüzden biraz daha eğlenceli bir uygulama sizleri bekliyor olacak. Bölüm bittiğinde uygulamamızın görüntüsü aşağıdaki gibi olacaktır.

#ID	Resim	Ad Soyad [A-C] [2-4]	E-mail [A-C] [2-4]	Telefon [0-9] [3-4]	Departman [A-C] [2-4]	Adres [A-C] [2-4]	İşlevler
#1		Gökhan Karadeniz	info@yuzmagitim.net	0507118108	Yatırım	İstanbul / Orançaya	Sil Gör
#2		Baran Karabulut	baranm@yuzmagitim.com	05321059922	Frontend	İstanbul	Sil Gör
#5		Gökhan Karadeniz	gokhan@yuzmagitim.com	0507118108	Kırtiya	Yapılmak Süreli	Sil Gör

gösterileri olarak giriş yapabilir. Çıkış yapın

Personel Ekle
Personel Adı
Bir Adres
info@yuzmagitim.com
Telefon
0507118108
Adres
Yapılmak Süreli
Departman
Kırtiya
Personel Resim
Personel Fotoğrafı
Gör Sil

Kullanıcı Giriş
E-Posta Adres
info@yuzmagitim.com
Şifre
1234
Giriş Yap

Personel Düzenle
Personel Adı
Bir Adres
info@yuzmagitim.com
Telefon
0507118108
Adres
Yapılmak Süreli
Departman
Kırtiya
Personel Resim
Personel Fotoğrafı
Gör Sil

Sizi bu kadar heyecanlandırmak yeterli bence. Artık kodlamaya başlayalım.

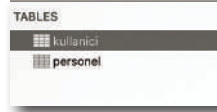
KULLANICI TABLOSU OLUŞTURMAK

Artık sistemimize bir kullanıcı denetimiyle gireceğiz. Bunu yapmak için ilk olarak kullanıcıların bilgilerini tutmak amacıyla bir tabloya ihtiyacımız var. Ben bu tablonun adını kullanıcı olarak belirledim.

NOT

Normal bir proje içerisindeyken tüm alan ve tablo isimlerini evrensel bir çalışma olması adına İngilizce belirtiyoruz. Burada yapının daha iyi anlaşılması için bu tarz isimlendirmeleri Türkçe olarak belirledik.

Tabloyu eklemek için daha önce oluşturmuş olduğumuz `personel_db` isimli veritabanımıza gidip kullanıcı isimli bir tablo ekleyelim.



Bu tablonun alanlarını ise basit tutup; id, kullanıcı_adi, sifre, email olarak belirliyorum.

Field	Type	Length	Unsigned	Zerofill	Binary	Allow Null	Key	Default	Extra	Increment	Encoding	Collation	Comment
id	INT	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	PK		auto_increment	0	0	0	
kullanici_adi	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None	0	ISO 8859-9 Turkish	utf8_turkish_ci	
sifre	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None	0	ISO 8859-9 Turkish	utf8_turkish_ci	
email	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None	0	ISO 8859-9 Turkish	utf8_turkish_ci	

NOT

id isimli alanının auto_increment ve primary_key olarak işaretlendiğine lütfen dikkat ediniz.

Field	Type	Length	Unsigned	Zerofill	Binary	Allow Null	Key	Default	Extra	Increment	Encoding	Collation	Comment
id	INT	11	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	PK		auto_increment	0	0	0	
kullanici_adi	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None	0	ISO 8859-9 Turkish	utf8_turkish_ci	
sifre	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None	0	ISO 8859-9 Turkish	utf8_turkish_ci	
email	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>		NULL	None	0	ISO 8859-9 Turkish	utf8_turkish_ci	

Tabloyu da oluşturduğumuza göre artık veritabanına bağlanıp kodlamalar yapabiliriz demektir. Ek olarak aşağıdaki tablo yapısını SQL kodu olarak kodlayıp veritabanı uygulamanızda direk bu tabloyu otomatik olarak oluşturabilirsiniz. Ya da bölümle ilgili dosyalarda ilgili SQL dosyasını çalıştırabilirsiniz.

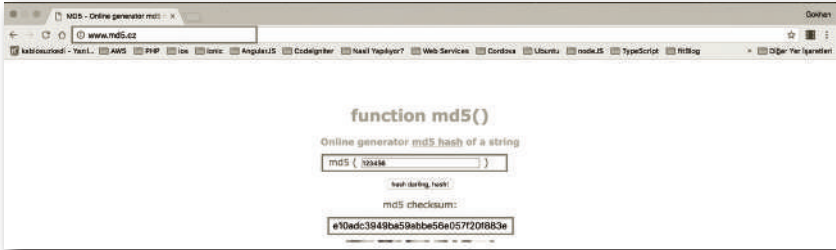

```
CREATE TABLE `kullanicı` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `kullanicı_adi` varchar(255) DEFAULT NULL,
  `sifre` varchar(255) DEFAULT NULL,
  `email` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin5;
```

TABLOYA BİLGİ GİRMEK

Giriş yapabilmek adına ben, veritabanı uygulamasını kullanarak bir tane kullanıcı oluşturacağım. Bu kullanıcıda en çok dikkat edilmesi gereken yer ise şifre alanı. Şifrelemeyi **database** üzerinde md5 algoritmasına göre şifreleyip tutacağımız için kayıt girerken şifre alanındaki veri md5 ile şifrelenmiş bir veri olmalıdır. Şifre olarak 123456 belirleyeceğim fakat bunun md5 deki karşılığı elbette daha karmaşık bir kod olacaktır. Şifremizin md5 halini bulmak için internet üzerinde birçok web sayfası bulabilirsiniz. Ben aşağıdaki site üzerinden gerçekleştirdim bu işlemi.

NOT

Normal şartlarda kullanıcı kaydını biz kendi sistemimiz üzerinden yapacağız (aslında bu bir ödev olacak sizin için. Burada kullanıcının kaydını oluştururken; kullanıcının belirlediği şifreyi bir PHP içerisinde md5() fonksiyonunu kullanarak otomatik olarak yapacağız.



123456 şifresinin md5 algoritmasına göre karşılığı e10adc3949ba59abbe56e057f20f883e. Ben de site üzerinden elde ettiğim bu kodu yeni oluşturacağım kaydın şifre alanına yazıyorum.

id	kullanicı_adi	sifre	email
1	gkandemir	e10adc3949ba59abbe56e057f20f883e	gokhan@gkandemir.com

Gördüğünüz gibi dosyamızın adı başarılı bir şekilde tablomuza aktarılmış durumdadır. Ben daha önceki kayıtlar için kendim rastgele birkaç resim adı girdim ve bu dosyaları da **uploads** klasörü içerisine aktardım ki listelemede bu dosyalar da gözüksün. Bunlara aldırış etmeyiniz.

PERSONEL LİSTELEME DÜZENLEMESİ

Personele ait resim dosyasını da başarılı bir şekilde girdirdiğimize göre bu resimlerin listeleme sayfasında da görüntülenmesi gerekiyor. Bundan dolayı **application/views/** klasörü altında bulunan **personel_liste.php** isimli dosyamıza gidelim ve aşağıdaki güncellemeleri yapalım.

```
<thead>
<th>#id</th>
<th>Resim</th>
```

İlk olarak **#id** alanımızın hemen yanında yer alması için **Resim** isimli bir başlık ekliyoruz ki tablomuzun hemen üstünde başlık olarak **Resim** de yazsın.

```
imgUrl");?>" alt="">
```

Daha sonra bu başlığın sırasına denk gelecek şekilde **<tbody>** içerisinde bulunan **<td>** elementlerine de bir ekleme yapıyoruz ki oluşturulacak her kayıt için resim dosyalarını gösterebileyim. Bu işlem için **id** bilgisini yazdırdığımız **<td></td>** elementinden hemen sonra bir **<td>** daha açıp, resim göstermesi için içerisine **** elementi ekliyoruz.

```
<td>#<?php echo $row->id; ?></td>
<td width="40">
    imgUrl");?>" alt="">
</td>
<td><?php echo $row->ad_soyad; ?></td>
```

**** elementinin de bootstrap'a uygun olması açısından bu elementin de class alanının değerleri bootstrap class ifadelerinden oluşmalıdır.

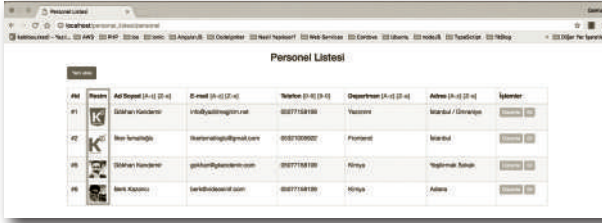
```
class="img-responsive"
```

son olarak da `` elementimizin resim dosyasını gösterebilmesi adına `src` alanının değerini; **uploads** klasörünün içerisindeki resim dosyalarından, tablo-muzdan gelen `imgUrl` alanındaki değere göre almasını söyledik.

```
<?php echo base_url("uploads/$row->imgUrl");?>
```

Burada `src` alanında yazacak URL değeri aşağıdaki gibi olacaktır.

http://localhost/personel_listesi/uploads/berk.png gibi dosya isimleri her kayıta göre değişecek şekilde düzenlenecektir. Böylece personel listemizde kayıtlarımızın resimlerini gösterme işlemi de gerçekleştirmiş olduk. Bunun için; adres çubuğuna **localhost/personel_listesi/** yazmamız yeterlidir.

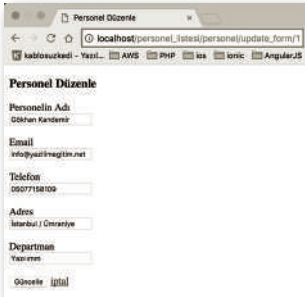


Adı	Fotoğraf	E-mail	Telefon	Departman	Adres	İşlemler
Öğretmen Karadeniz		info@yazilma.com.tr	05077158109	Yasemen	İstanbul / Çeremçe	Gör Sil
Bank Aksoy		bankaksoy@gmail.com	05077158109	Frontend	İstanbul	Gör Sil
Öğretmen Karadeniz		gokhan@yazilma.com	05077158109	Frontend	Yedigöller Mahallesi	Gör Sil
Bank Aksoy		bankaksoy@gmail.com	05077158109	Frontend	Adana	Gör Sil

Gördüğümüz gibi listemize kayıtlarımızın resimleri de gelmiş oldu.

PERSONEL DÜZENLE SAYFASI

Şuan için 4. Bölüm içerisinde oluşturmuş olduğumuz personel düzenle sayfa-mızın görüntüsü aşağıdaki gibidir. Herhangi bir CSS işlemi ve görsellik henüz burada yok.



Personel Düzenle

Personelin Adı
Öğretmen Karadeniz

Email
info@yazilma.com.tr

Telefon
05077158109

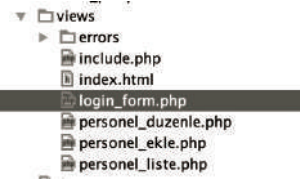
Adres
İstanbul / Çeremçe

Departman
Yasemen

[Gözet](#) [İptal](#)

HTML KODLAMASI

Yeni bir view oluşturmak için `application/views/` klasörü altında `login_form.php` isimli bir view dosyası oluşturalım.



Dosyayı oluşturduğumuza göre HTML kodlamalarını gerçekleştirelim.

HTML Kodları

```

<!doctype html>
<html lang="tr">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <?php $this->load->view("include"); ?>
</head>
<body class="login">
<div class="container">
    <div class="row">
        <div class="col-md-6 col-md-offset-3">
            <h3 class="text-center">Kullanıcı Giriş</h3>
            <hr>
            <?php
                $alert = $this->session->userdata("alert");
                if($alert){ ?>
                    <
!- Mesaj Kutusunun HTML Kodları ->
                <div class="alert <?php echo $alert["type"]; ?>" role="alert">
                    <!-- Alert içeriği -->
                    <strong><?php echo $alert["title"]; ?></strong>
                    <span><?php echo $alert["message"]; ?></span>
                </div>
                <?php } ?>
                <form class="login_form" method="post" action="<?php echo base_
url("kullanici/login");?>">
                    <div class="form-group">
                        <label for="exampleInputEmail1">E-posta Adresi</label>
                        <input type="email" name="email" class="form-control"
id="email" placeholder="E-posta Adresinizi giriniz.">
                    </div>
                    <div class="form-group">
                        <label for="exampleInputPassword1">Şifre</label>

```

```
        <input type="password" name="sifre" class="form-control"
id="sifre" placeholder="Şifrenizi giriniz.">
        </div>
        <div class="row">
            <div class="form-group col-md-5">
                <input type="text" name="captcha" class="form-control"
id="captcha" placeholder="Doğrulama kodunu giriniz.">
            </div>
            <div class="col-md-3">
                <!-- captcha Kodu Gelecek -->
            </div>
        </div>
        <button type="submit" class="btn btn-default">Giriş Yap</button>
    </form>
</div>
</div>
</div>
</body>
</html>
```

Bu view içerisinde tanımlamış olduğumuz kodlamaları teker teker ele alalım.

```
<head>
```

```
    <meta charset="UTF-8">
    <title>Login</title>
    <?php $this->load->view("include"); ?>
</head>
```

Yine bootstrap'in login_form.php sayfasında aktif olması için tanımlamış olduğumuz include.php dosyasını <head></head> blokları arasında çağırıyoruz.

```
<body class="login">
```

<body> elementine bir class ataması yaptık. Bu login isimli class ifademiz sayesinde bu view dosyamızın arka plan rengini değiştireceğiz.

```
<div class="container">
    <div class="row">
        <div class="col-md-6 col-md-offset-3">
            <!-- Login formumuzun HTML kodları -->
```

```

        </div>
    </div>
</div>

```

Burada yine bootstrap'in hiyerarşisini kullanıyoruz ve giriş yapmak için kullanacağımız kodlamalar bu elementlerin içerisinde olacak.

```

<h3 class="text-center">Kullanıcı Giriş</h3>
<hr>

```

Formumuzun üzerinde basit bir şekilde **Kullanıcı Giriş** başlığı atıyoruz ki formun ne işe yaradığını söylesin kullanıcımıza.

```

<?php
$alert = $this->session->userdata("alert");
if($alert){ ?>
    <!-- Mesaj Kutusunun HTML Kodları -->
    <div class="alert <?php echo $alert["type"]; ?>" role="alert">
        <!-- Alert içeriği -->
        <strong><?php echo $alert["title"]; ?></strong>
        <span><?php echo $alert["message"]; ?></span>
    </div>
<?php } ?>

```

Personel Listesi sayfasında yapmış olduğumuz mesaj kutusu gösterme işlemini olduğu gibi burada da yapıyoruz. Oradaki kodlamayı aynen buraya da yapıştırabilirsiniz. Eğer SESSION içerisinde alert isimli bir indis varsa. Bu indisin başlığını, açıklamasını ve türünü alıp, mesaj kutumuzu bu değerlere göre şekillendiriyoruz.

```

<form class="login_form" method="post" action="<?php echo base_url("kullanici/login");?>">
    <!-- form elementleri -->
</form>

```

Kullanıcının email adresi ve şifre bilgisini girdikten sonra bu bilgilerin bizim Kullanici controller dosyamıza göndermesi için form elementinin method alanını post, gönderileceği metodu belirlemek için ise action alanını kullanici/login olarak belirledik.