

## Web Tasarımı ve Programlama

### Ders 8:MVC (Model – View – Controller)

Dr. Öğr. Üyesi Elvan DUMAN

eduman@mehmetakif.edu.tr  
Yazılım Mühendisliği Bölümü

- 1. MVC'ye Giriş
- 2. MVC Yapısı
- 3. Controller Katmanı (İş Mantığı)
- 4. Route Katmanı (Yönlendirme)
- 5. Model Katmanı
- 6. View Katmanı (EJS Şablonları)
- 7. MVC'nin Sağladığı Avantajlar

**Amaç:** Web uygulamalarında kodu **mantıksal katmanlara ayırmak**

- Kod sadece dosyalara bölünmez; görevlerine göre ayrılır.
- Bu yaklaşım separation of concerns (sorumluluk ayrımı) olarak bilinir.
- Her bölümün görevi netleşir: hangi kod veriyle ilgileniyor, hangisi görüntüyle, hangisi akışı yönetiyor.
- MVC bu düzeni sağlar: Model – View – Controller

| Katman            | Görev                                                        | Örnek                                             |
|-------------------|--------------------------------------------------------------|---------------------------------------------------|
| <b>Model</b>      | Veriyi temsil eder, kaydetme/getirme işlemlerini yapar.      | Ürün listesini dosyadan okuma veya bellekte tutma |
| <b>View</b>       | Kullanıcıya gösterilecek içeriği oluşturur (EJS şablonları). | Ürün sayfası HTML çıktısı                         |
| <b>Controller</b> | İstekleri karşılar, modeli kullanır, uygun view'ı döndürür.  | Ürün detayını alıp ekrana render eder             |

**Amaç:** Kodun düzenli, test edilebilir ve genişletilebilir hale gelmesi.

**MVC**, uygulama kodunu üç ayrı katmana ayırarak **düzenli, bakımı kolay ve genişletilebilir** bir yapı oluşturmayı amaçlayan bir mimari desendir.

- Node.js + Express + EJS projesinde:
- **Controller** /controllers klasöründeki dosyalar
- **View** /views klasöründeki .ejs şablonları
- **Models** klasöründe model dosyaları

| Klasör       | Görevi                                           |
|--------------|--------------------------------------------------|
| controllers/ | İş mantığı (ör. ürün ekleme, hata yönetimi)      |
| routes/      | URL yönlendirmeleri (hangi controller çalışacak) |
| views/       | EJS şablon dosyaları                             |
| includes/    | Tekrarlanan kısımlar (header, footer, nav)       |
| public/      | Statik dosyalar (CSS, görseller)                 |
| util/        | Yardımcı modüller (ör. path ayarları)            |

Bu yapı Express projelerinde MVC'nin doğal karşılığıdır.

Controller, uygulamanın “beyni” dir.

İstekleri alır, gerekli işlemleri yapar ve res.render() ile görünümü döndürür.

- Controller:
- Route tarafından çağrılır.
- Form verilerini işler (req.body).
- View dosyasını render eder (res.render()).

```
exports.getUrunEkle = (req, res, next) => {  
  //res.sendFile(path.join(rootDir, 'views', 'urun-ekle.html'));  
  res.render('urun-ekle', {  
    pageTitle: 'Ürün Ekle',  
    path: '/admin/urun-ekle'  
  });  
};
```

```
exports.postUrunEkle = (req, res, next) => {  
  urunler.push({urun_adi:req.body.urun_adi});  
  res.redirect('/');  
};
```

Route, gelen isteğin **hangi controller fonksiyonuna gideceğini** belirler.

## Route:

- URL : Controller eşleşmesini yapar.
- İş mantığı içermez, sadece yönlendirir.
- app.js içinde `app.use('/admin', adminRoutes);` ile aktif hale gelir.

## admin.js

```
const path = require('path');  
const express = require('express');
```

```
const urunController = require('../controllers/urun');  
const router = express.Router();
```

```
router.get('/urun-ekle', urunController.getUrunEkle);  
router.post('/urun-ekle', urunController.postUrunEkle);
```

```
module.exports = router;
```

- Model, veriyi temsil eder ve bu verinin işlenmesinden sorumludur.
- Controller, Model üzerinden veri ekler, siler, günceller veya okur.
- Veritabanında saklama ve işlemlerini bilmediğimizden ilk olarak veritabanı yerine bellekte ve dosya sisteminde veri saklama örnekleri yapılabilir.
- MVC’de Model, uygulamanın “veri kaynağı katmanı” dır.

Bu örnekte veriler bellek içi dizide saklanır. Sayfa yenilendiğinde veriler kaybolur.

```
const urunler = [];
```

```
module.exports = class Urun {  
  constructor(a) {  
    this.urun_adi = a;  
  }  
  save() {  
    urunler.push(this);  
  }  
  static fetchAll() {  
    return urunler;  
  }  
};
```

- Gerçek uygulamalarda veriler ilişkisel veya ilişkisel olmayan veritabanında saklanır. Ancak o aşamaya gelmeden önce dosya sisteminde saklama ile pratik yapılabilir.
- Node.js'te fs modülü kullanarak veriler JSON dosyasına yazılabilir.
- Model katmanında dosya okuma ve yazma işlemleri yapılır.

```
const fs = require('fs');
const path = require('path');
const dosya = path.join(
  path.dirname(require.main.filename),
  'data',
  'urunler.json'
);
const dosyadanUrunAl = cb => {
  fs.readFile(dosya, (err, fileContent) => {
    if (err) cb([]);
    else cb(JSON.parse(fileContent));
  });
};
```

Bu yapıda, her yeni ürün eklendiğinde veriler urunler.json dosyasına kaydedilir. Uygulama kapanıp açılrsa bile kayıtlar korunur.

- Controller, Model'in metotlarını çağırarak veriyle etkileşir.
- Model veriyi **nasıl saklayacağını** bilir, Controller ise **ne zaman** saklayacağını belirler.
- `Urun.save()` → veriyi dosyaya yazar
- `Urun.fetchAll()` → veriyi okur ve Controller'a gönderir.

```
exports.post_urun_ekle = (req, res, next) => {  
  const urun = new Urun(req.body.urun_adi);  
  urun.save();  
  res.redirect('/');  
};  
  
exports.get_urunler = (req, res, next) => {  
  Urun.fetchAll(urunler => {  
    res.render('index', { urunler: urunler, pageTitle:  
      'Anasayfa' });  
  });  
};
```

Controller ile Model arasındaki ilişki uygulamanın iş mantığını ve veri akışını net biçimde ayırır.

View, kullanıcıya gönderilen arayüzdür.  
Controller tarafından gönderilen verilerle dinamik hale gelir.

View:

- EJS değişkenleriyle (<%= %>) dinamik içerik gösterir.
- includes/ altındaki parçaları kullanarak tekrar eden kısımları tek dosyada toplar.

**örnek:** /views/urun-ekle.ejs

```
<%- include('includes/head.ejs') %>
</head>
<body>
  <%- include('includes/navigation.ejs') %>
  <form action="urun-ekle" method="POST">
    <input type="text" name="urun_adi">
    <button type="submit">Ürün Ekle</button>
  </form>

  <%- include('includes/end.ejs') %>
```

- **Modülerlik:** Her dosya kendi işini yapar.
- **Okunabilirlik:** Kod karmaşası azalır.
- **Tekrar kullanılabilirlik:** Aynı Controller farklı View'larla çalışabilir.
- **Kolay genişleme:** Yeni sayfa eklemek için sadece route + controller + view eklemek yeterlidir.

| Katman     | Örnek Dosya         | Görev                          |
|------------|---------------------|--------------------------------|
| Model      | (henüz yok)         | Veri yönetimi                  |
| View       | views/urun-ekle.ejs | Kullanıcıya sunulan arayüz     |
| Controller | controllers/urun.js | İş mantığı                     |
| Route      | routes/admin.js     | Yönlendirme                    |
| App.js     | app.js              | Tüm katmanları bağlayan merkez |

## Ders 8 Sunum Sonu

**İletişim : [eduman@mehmetakif.edu.tr](mailto:eduman@mehmetakif.edu.tr)**

