

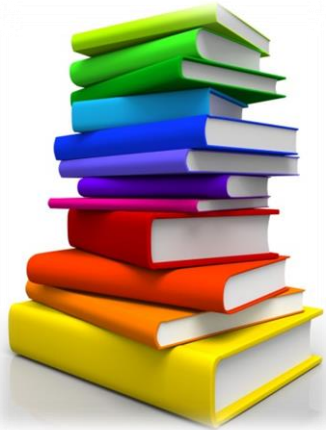
YZM 3215

İleri Web Programlama

Yrd. Doç. Dr. Deniz KILINÇ
Celal Bayar Üniversitesi
Hasan Ferdi Turgutlu Teknoloji Fakültesi
Yazılım Mühendisliği

BÖLÜM - 6

ASP.NET MVC I. Bölüm



Bu bölümde;

- ASP.NET MVC
 - MVC nedir?
 - ASP.NET MVC
 - ASP.NET MVC Avantajları
 - İlk ASP.NET MVC Uygulaması
 - Controller ve View Arasında Veri Taşıma
 - ViewData, ViewBag, TempData, Session

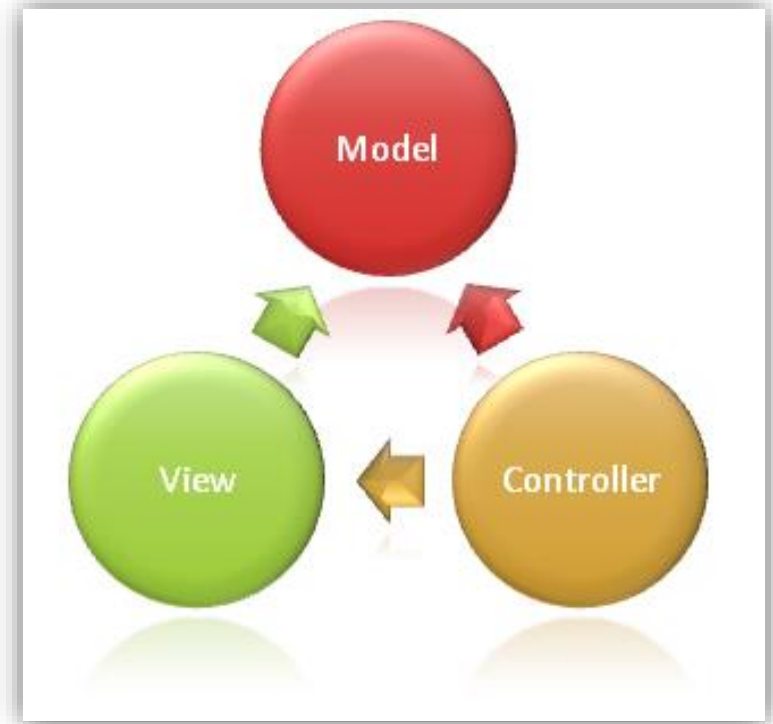
ile ilgili konular anlatılacaktır.

MVC Nedir?

- MVC (**Model-View-Controller**), ilk olarak 1979 yılında Trygve Reenskaug tarafından tanımlandıktan sonra **yazılım alanında kullanılmaya başlayan** önemli bir **mimari patern/kalıp** (architectural pattern)'dir.
- Ortaya atıldığı ilk yıllarda ismi “**Thing-Model-View-Controller**” sonradan basitleştirilerek şu anki halini almıştır.
- MVC Java'da, C++'ta, PHP'de ve bir çok köklü dilde desteklenmektedir.
- MVC mimari kalıbı yazılım katmanlarının,
 - Örnek olarak **presentation logic'ten data access logic'in**, ayrı tutulduğu uygulamalar oluşturur.

MVC Nedir? (devam...)

- MVC mimari kalıbı 3 kısımdan oluşur:
 - **Model:** Veri ve veri etkileşimi sağlanan kısımdır.
 - **View:** Kullanıcı ile etkileşimin sağlandığı kullanıcı ara yüzünün oluşturulduğu kısımdır.
 - **Controller:** Kullanıcı ara yüzündeki işlemler ile veri veya iş katmanını arasındaki iletişimi sağlar.



ASP.NET MVC

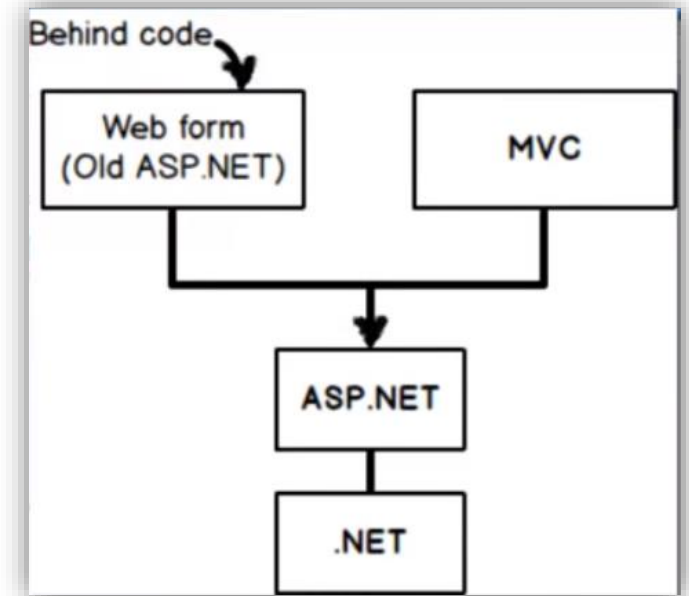
- Microsoft tarafından ilk defa **2007 yılında** duyurulmuştur.
- ASP.NET **Web Forms'un** yetersizlikleri nedeniyle ortaya çıkmıştır.
 - ASP.NET içerisinde geliştirilmiş hazır server kontrollerini sürükle-bırak yöntemi ile sayfalarımıza atıp, otomatik olarak bizim için oluşturulmuş HTML çıktılarına sahip oluruz.
 - Çalışmasını istediğimiz sayfadaki işlemleri handler'lar, modüller vasıtasıyla çalıştırırız.
 - Dolayısı ile ASP.NET yerleşik olarak bir **mimari kalıba sahip değildir.**

ASP.NET MVC (devam...)

ASP.NET Web Forms

VS.

ASP.NET MVC



<https://www.youtube.com/v/bGpBgDDDV1>

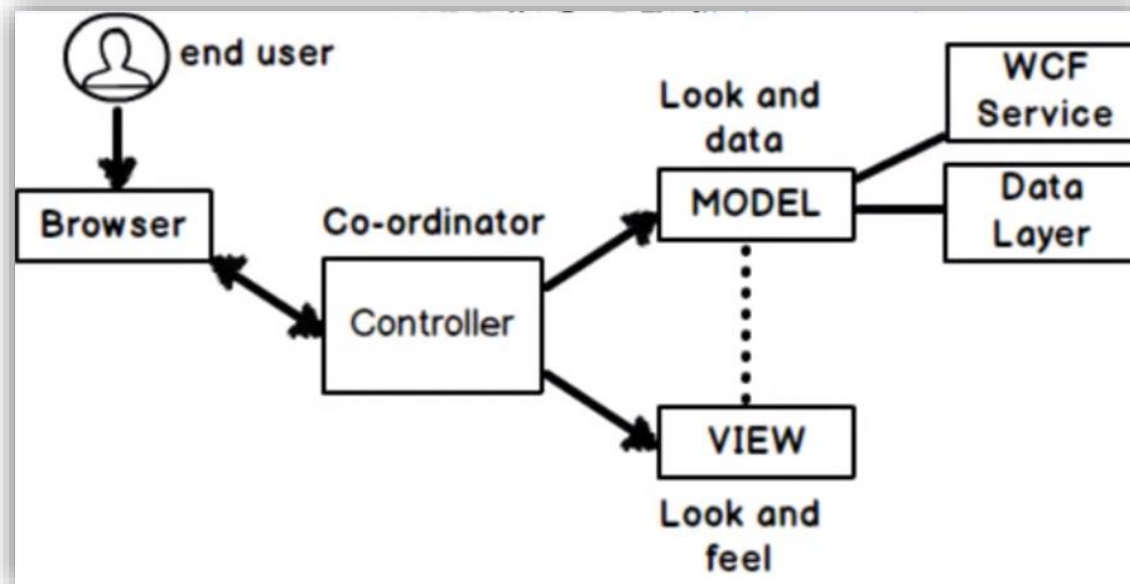
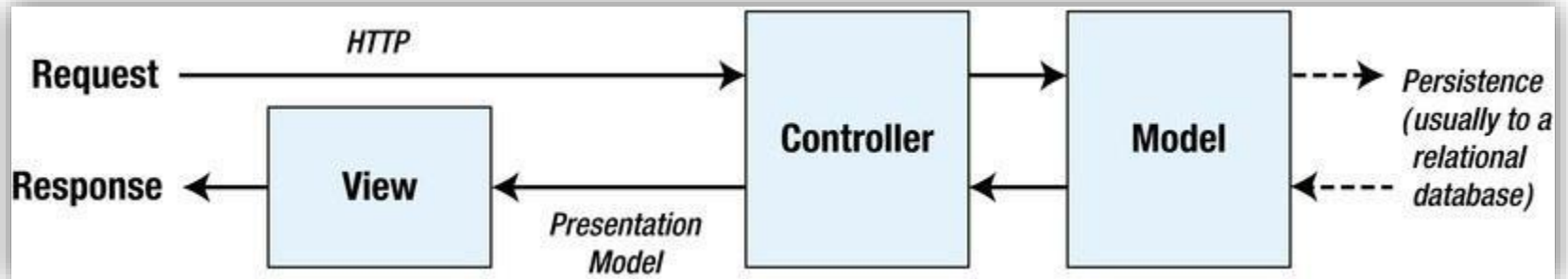
M

ASP.NET MVC (devam...)

MVC Versiyonları:

- MVC 1
- MVC 2 (**Visual Studio 2008**)
- MVC 3 (**Visual Studio 2012**)
- MVC 4 (**Visual Studio 2012**)
- MVC 5 (**Visual Studio 2013**)
- MVC 6 **beta 8**

ASP.NET MVC (devam...)



ASP.NET MVC (devam...)

- ***M (Model)***, İş Mantığını ve *veri işleme süreçlerini* yürütür. C (Controller) tarafından gönderilen yönlendirmelere göre hareket eder.
 - Bilgi işleme sürecinden sonra veriyi C'ye, diğer modellere veya doğrudan V (View)'ye gönderir. Veritabanına erişim, class'lar, ayrıca data access layer (DAL) yani veritabanı işlemleri için kullanacağımız Ado.Net, Nhibernate veya EntityFramework ile veri işlemleri burada yer alır.
- ***V (View)*** son kullanıcıya gösterilecek olan verinin sunumu ile ilgilenir. View, bu bilgiyi Controller'dan alır, aynı zamanda son kullanıcıdan gelen talepleri Controller'a iletir. Dinamik olarak HTML kodları burada üretilir.

ASP.NET MVC (devam...)

- **Controller** sistemin ana kısmıdır. Gelen talepleri kontrol eder ve sistemin diğer elemanlarınının (M,V) bilgiyi uygun şekilde alıp, göndermelerini sağlar.
 - Client (istemci) tarafından yapılan **request (istek)** controller'lar tarafından yakalanır ve *işleme tabi tutulur*.
 - Özetle Controller; iş akışının gerçekleştiği, ara yüzden gelen kullanıcı etkileşimlerinin değerlendirildiği, işlendiği, gerekli metotların çalıştırıldığı, değişkenlerin ve nesnelerin oluşturulduğu, M ile V bölümleri arasında iletişimin sağlandığı yerdir.

ASP.NET MVC Avantajları

- **Performans**

- Asp.Net MVC’de **ViewState** kavramı bulunmadığı için **durum yönetimi** *yazılım geliştirici* tarafından yapılmak durumundadır.

- **Geliştirmesi / bakımı kolay ve hızlı**

- Katmanlar birbirinden farklı olduğundan farklı yazılım geliştiriciler *farklı katmanlarda* aynı işlem için **eş zamanlı** olarak kodlama yapabilirler.
- Problemin nerede olduğunu bulmak daha kolay.

- **Test Driven Developmet (TDD) iş sürecine uygun**

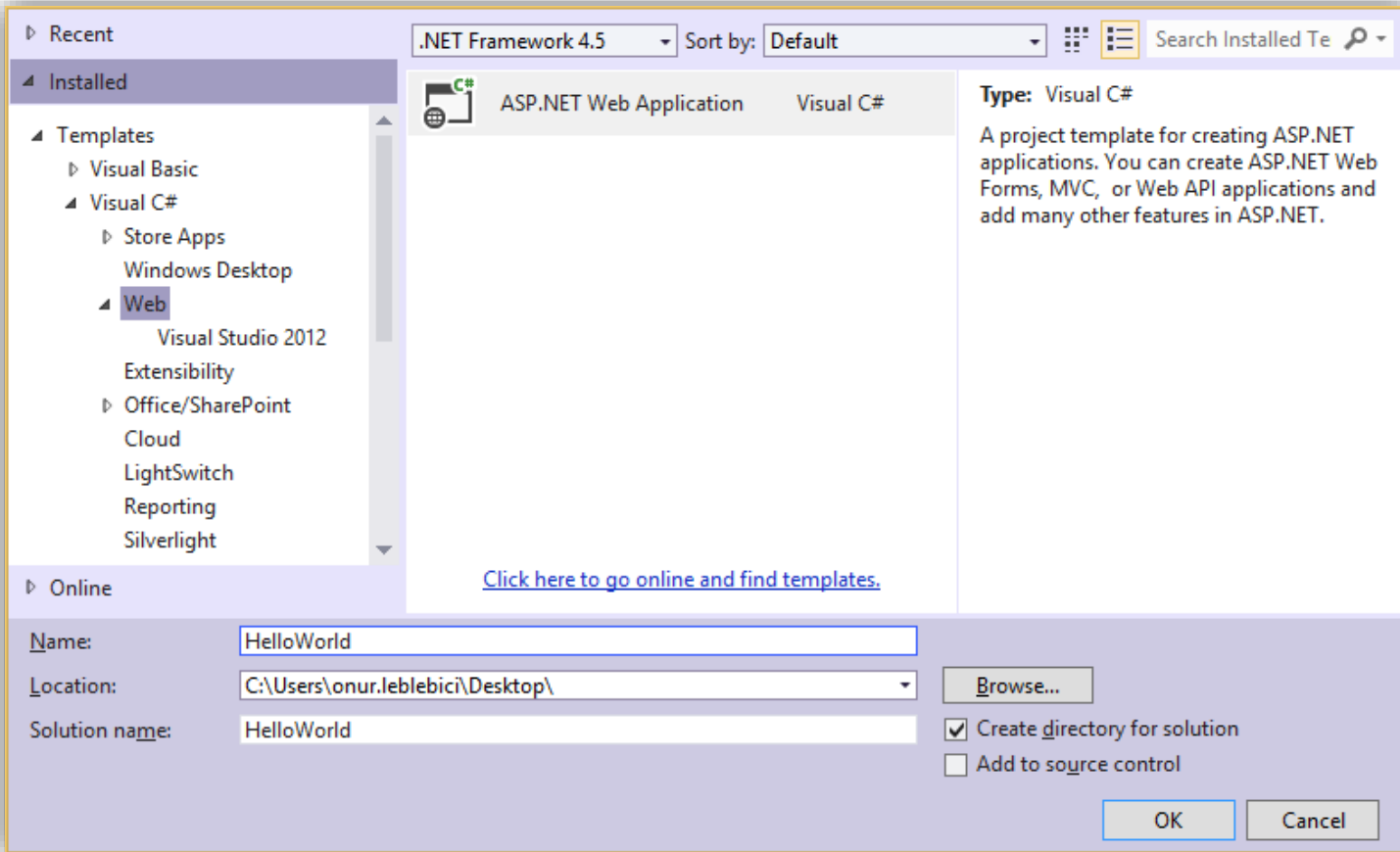
- Asp.Net MVC’de yazılan kodlar kontrollere ya da sayfaya özgü olmadığı (bağımsız yazıldığı) için kolaylıkla test edilebilir durumdadır.

ASP.NET MVC Avantajları (devam..)

- **Kontrollerin Yönetimi İçin Client Side Development (İstemci Taraflı Geliştirme)**
 - Sayfa içerisine yerleştirilen HTML kontrollerinin yönetimi ve sunucuya istek gönderimi için Asp.Net MVC’de Javascript ve Javascript tabanlı istemci taraflı teknolojiler kullanılmaktadır.
- **Ölçeklenebilir ve Genişletilebilir**
- **Microsoft destekli olarak, Codeplex üzerinde açık kaynak kodlu**
- **Arama motoru dostu içerik ve URL üretimi desteği**
 - SEO dostu URL üretimi desteklenmektedir.

İlk MVC Uygulaması – Adım 1

MVC 5 versiyonu için...



İlk MVC Uygulaması – Adım 2

Select a template:

Empty Web Forms **MVC** Web API

Single Page Application Facebook

Step 1

Add folders and core references for:

Web Forms MVC Web API

Add unit tests → **Unit Test**

Test project name: MyFirstHelloWorld.Tests

Step 2

Change Authentication

Authentication: **Individual Use**

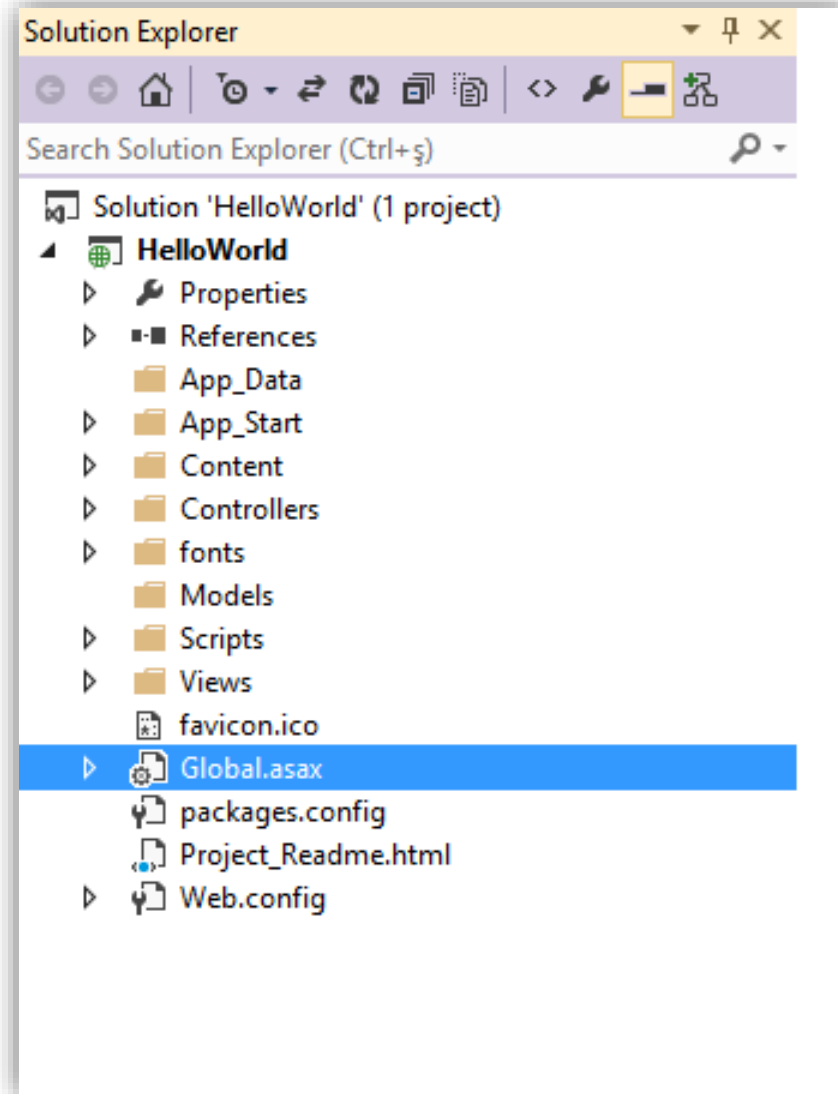
A project template for creating *A* ASP.NET MVC allows you to build Model-View-Controller architecture. It includes many features that enable development for creating application standards.

[Learn more](#)

İlk MVC Uygulaması – Adım 2

- **Step 1:** MVC şablonunu ve checkbox'ını seç. MVC şablonu seçilince bu checkbox büyük ihtimalle disable olacaktır..
- **Step 2:** Change Authentication butonuna tıkla ve "No Authentication" seçeneğini işaretle.
- **Unit Tests:** MVC uygulamalarının en önemli avantajlarından birisi "Unit" testlerdir. Şimdilik işaretlemiyoruz. (Nedir bu **Unit Test**?)

İlk MVC Uygulaması – Adım 3



KLASÖRLER

İlk MVC Uygulaması – Adım 3

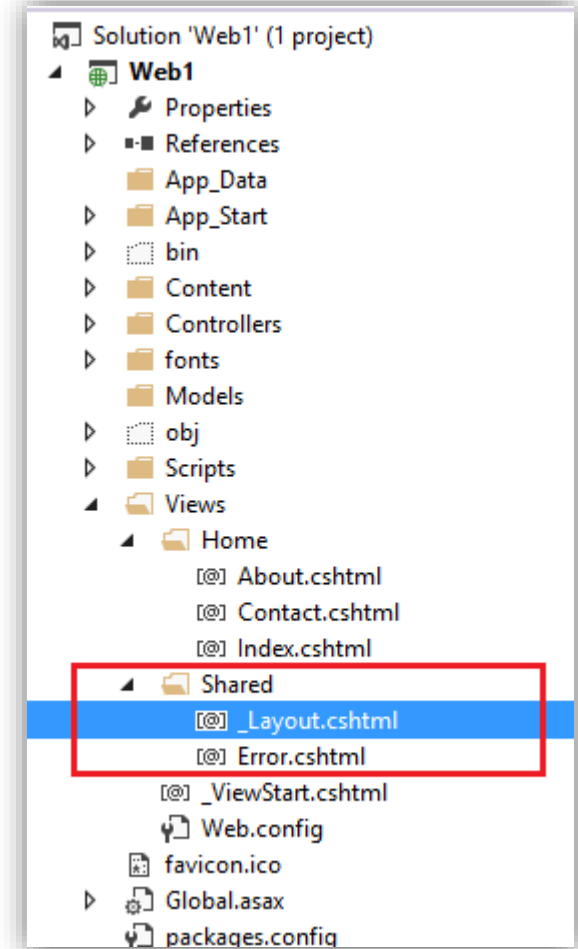
View	MVC View'lerinin bulunduğu klasördür. HTML, Razor veya Webform formatında olabilirler.
Model	Model class'larının bulunduğu klasördür. Örneğin: Müşteri sınıfı ve validasyonları.
Controller	Action'lara sahip Controller'ların yer aldığı klasördür. İstemciden request'i alır, uygun action'ı tetikler, modelden doğru nesneyi yaratır ve View'a bind eder. View kullanıcıya response olarak gösterilir.
Script	Javascript dosyalarının yer aldığı klasördür. Default jQuery dosyaları yer alır.
Content	Css dosyalarının bulunduğu klasördür.
App_Data	Veriler XML, txt gibi dosyalarda saklanıyorsa, bu dosyaların yer aldığı klasördür.

KLASÖRLER

İlk MVC Uygulaması – Adım 3

Layout View

- Uygulamalarda **genelde değişmeyen UI elemanları** bulunmaktadır (Logo, üst menü, alt menü, navigasyon bar).
- ASP.NET MVC içerisindeki **Layout View’lar** sayesinde her sayfada bu ortak kısımları kodlamak zorunda kalmayız. ASP.NET Web Forms içerisindeki Master Page’lere benzer.



İlk MVC Uygulaması – Adım 3

Layout View

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>@ViewBag.Title - My ASP.NET Application</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
</head>
<body>
  <div class="navbar navbar-inverse navbar-fixed-top">...</div>
  <div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
      <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
    </footer>
  </div>
  @Scripts.Render("~/bundles/jquery")
  @Scripts.Render("~/bundles/bootstrap")
  @RenderSection("scripts", required: false)
</body>
</html>
```

Dinamik render işlemi

Diğer view'lar için placeholder

İlk MVC Uygulaması – Adım 3

Application name Home About Contact

_Layout.cshtml

ASP.NET

Index.cshtml

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

[Learn more »](#)

Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

[Learn more »](#)

Get more libraries

NuGet is a free Visual Studio extension that makes it easy to add, remove, and update libraries and tools in Visual Studio projects.

[Learn more »](#)

Web Hosting

You can easily find a web hosting company that offers the right mix of features and price for your applications.

[Learn more »](#)

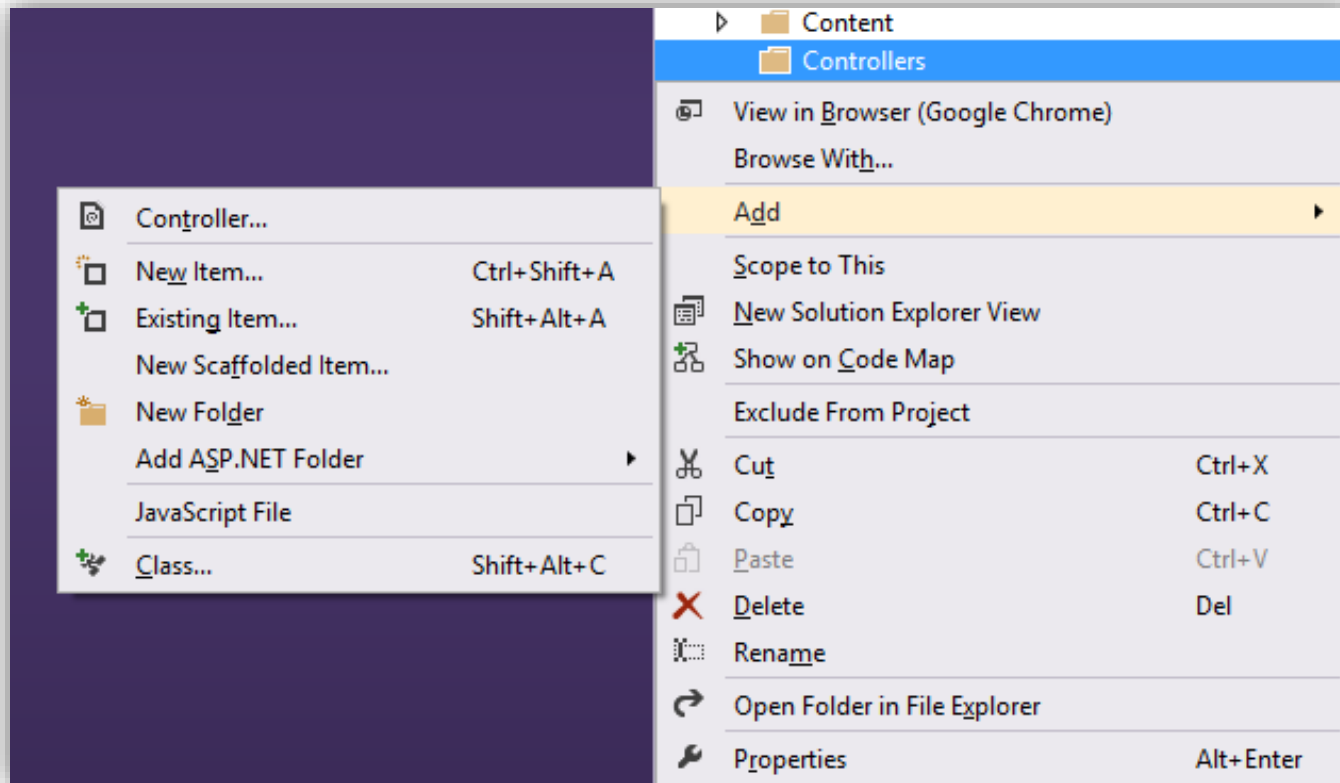
İlk MVC Uygulaması – Adım 4



*Controller ve View klasörleri altındaki dosyaları silelim, **en baştan** Controller ve View ekleyerek devam edelim.*

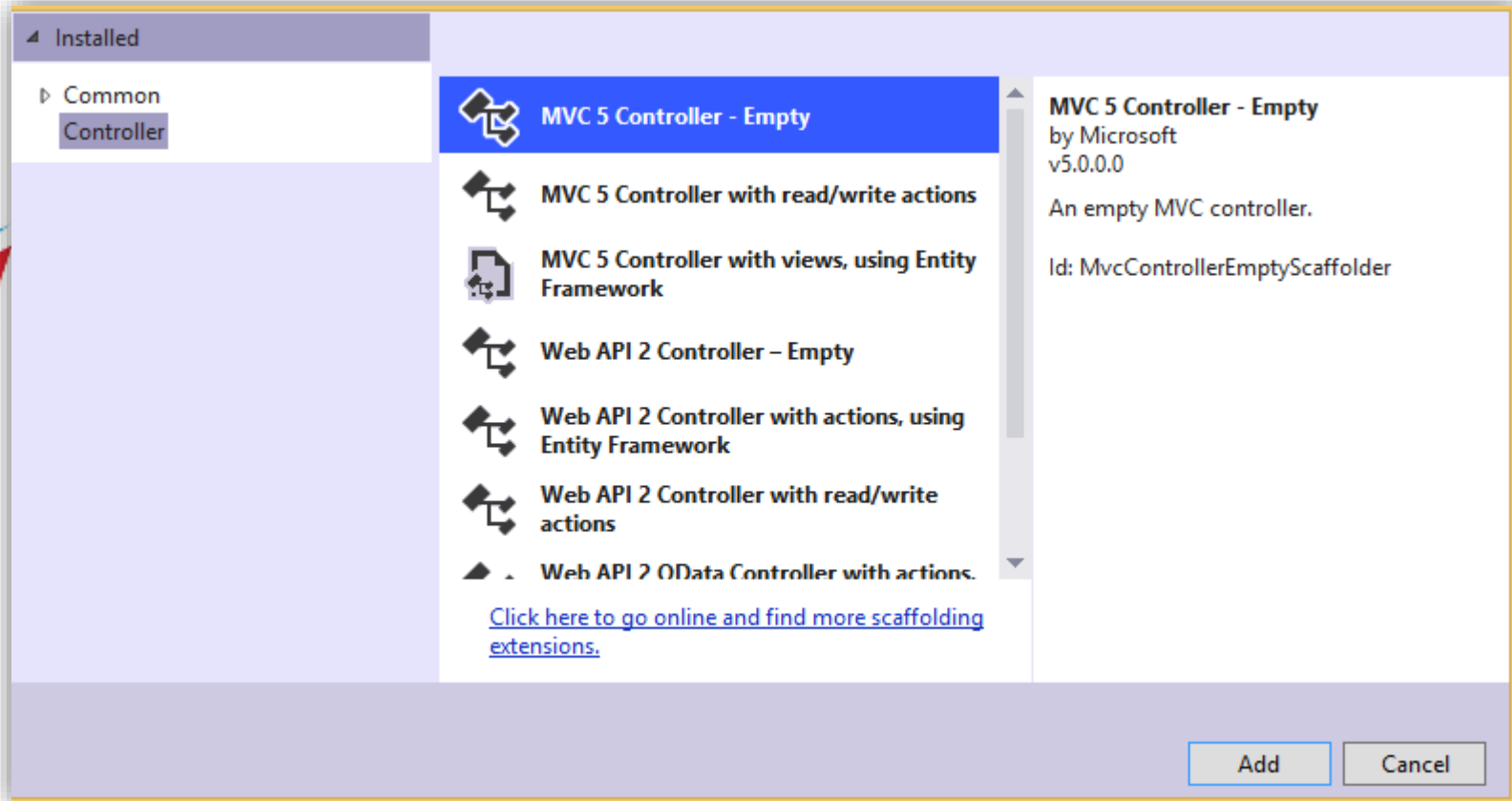
İlk MVC Uygulaması – Adım 5 (C+)

FirstController isimli bir Controller ekleyelim.



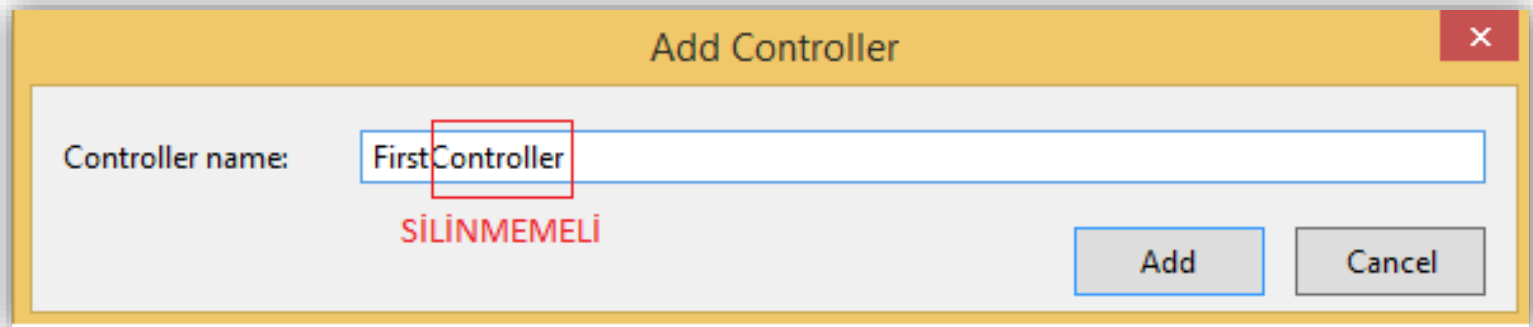
İlk MVC Uygulaması – Adım 5 (C+)

Empty Controller ekliyoruz. Baştan öğreniyoruz.



İlk MVC Uygulaması – Adım 5 (C+)

- Controller kelimesi **anahtar** kelimedir. Kesinlikle silmiyoruz.
- Controller'ımızın gerçek adı **First**.



The screenshot shows a dialog box titled "Add Controller". It has a yellow header bar with a close button (X) in the top right corner. The main area is light gray and contains a label "Controller name:" followed by a text input field. The input field contains the text "FirstController" and is highlighted with a red rectangular box. Below the input field, the text "SİLİNMEMELİ" is written in red. At the bottom right of the dialog, there are two buttons: "Add" and "Cancel".

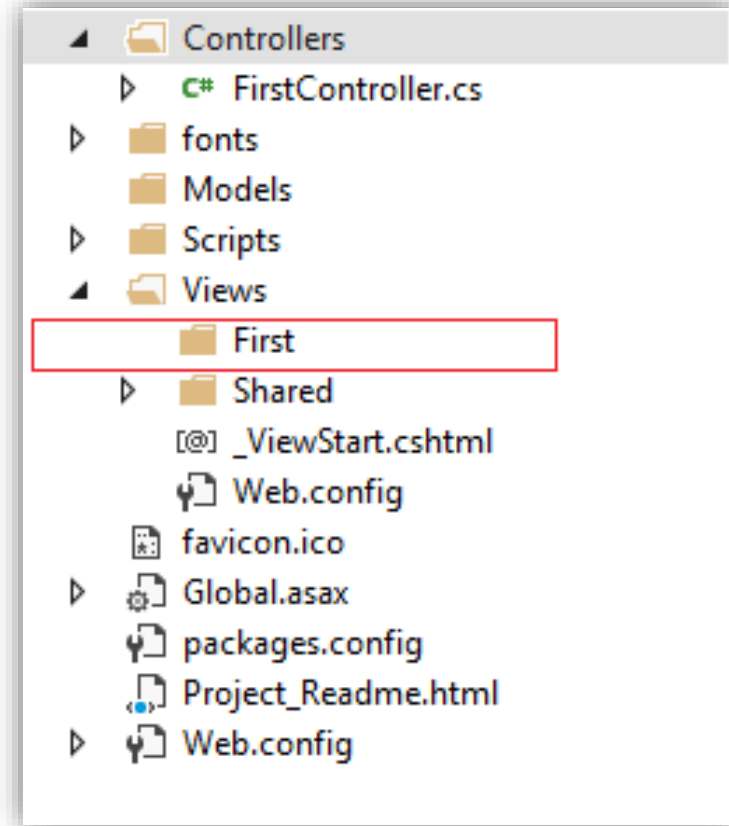
İlk MVC Uygulaması – Adım 5 (C+)

- FirstController sınıfı içerisine `SayHello()` isimli Action ekleyelim.

```
//GET: SayHello  
0 references  
public ActionResult SayHello()  
{  
    return View();  
}
```

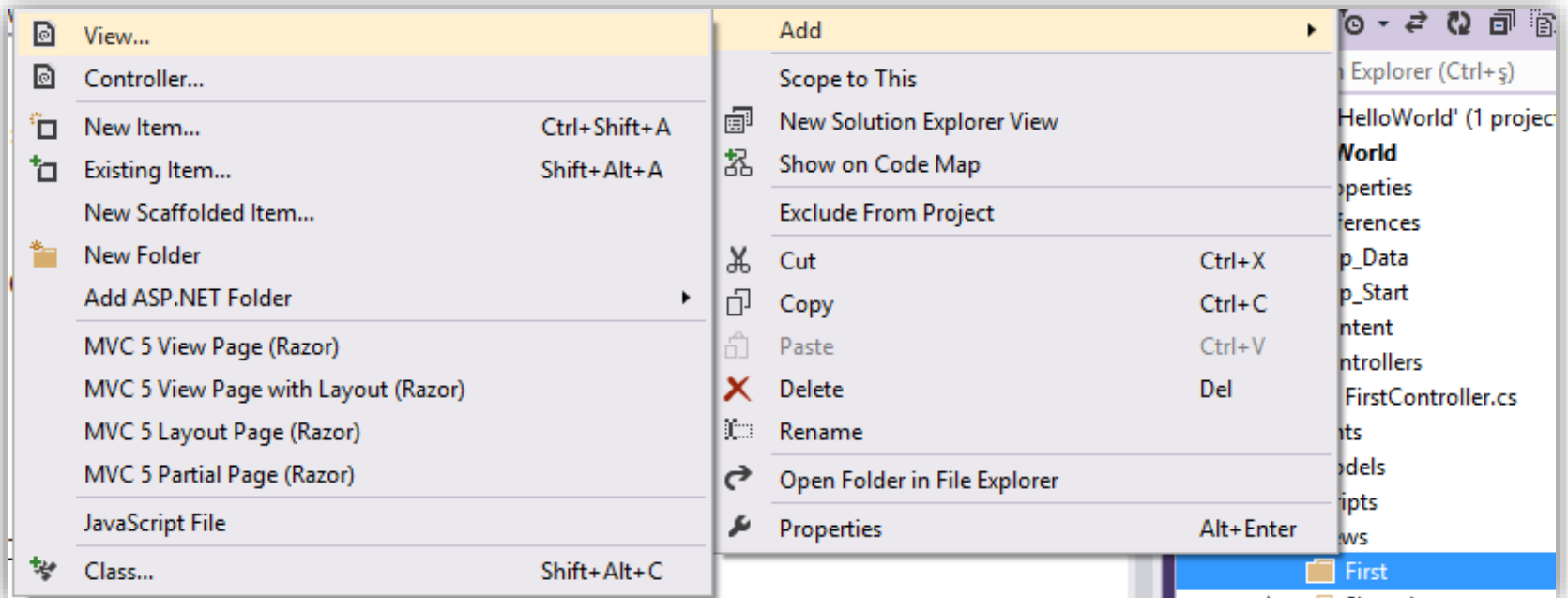
İlk MVC Uygulaması – Adım 6 (V+)

- Controller ve Action eklendikten sonraki adım, kullanıcı requestine cevap olarak üretilecek *View*'in yaratılmasıdır.
- Views klasörü altında **First** klasörü **otomatik** olarak yaratılmıştır.



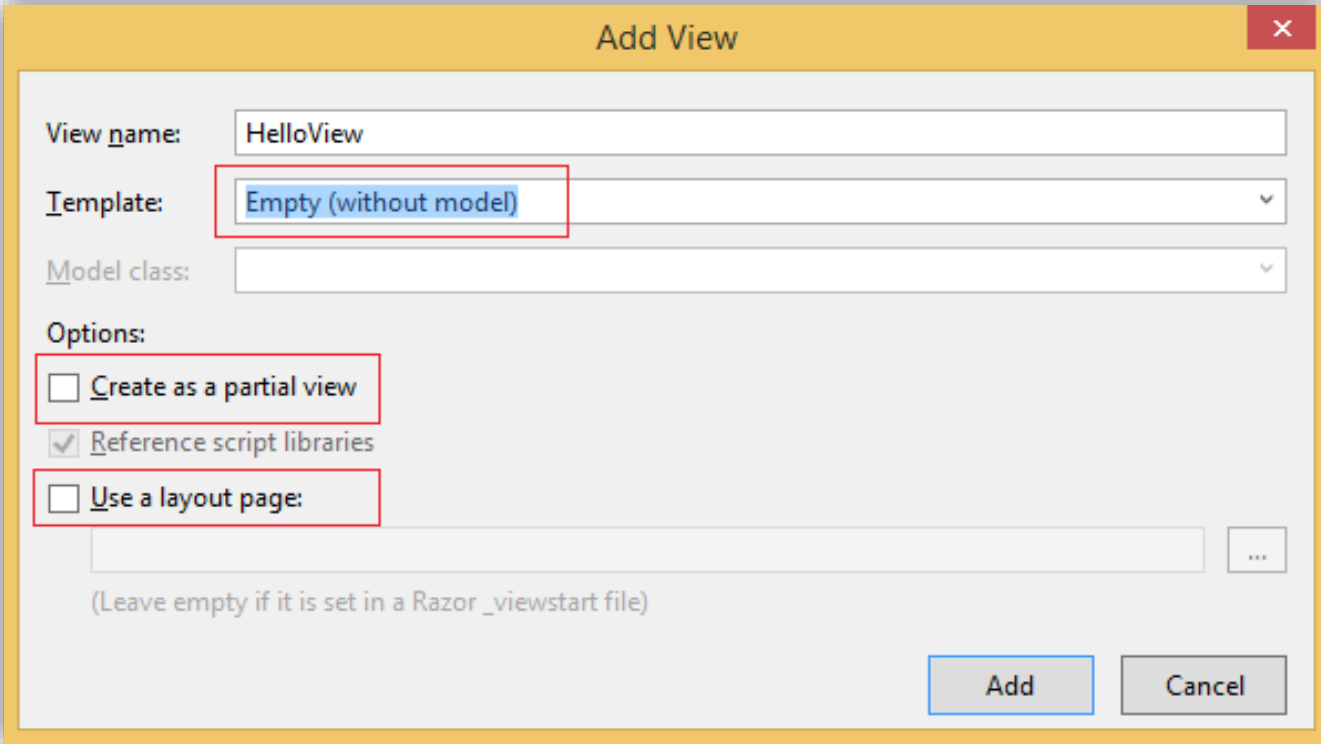
İlk MVC Uygulaması – Adım 6 (V+)

- **HelloView** isimli bir View yaratalım.



İlk MVC Uygulaması – Adım 6 (V+)

- **HelloView** isimli bir View yaratalım.
- KISS prensibi: KEEP IT SIMPLE.



View name: HelloView

Template: Empty (without model)

Model class:

Options:

Create as a partial view

Reference script libraries

Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

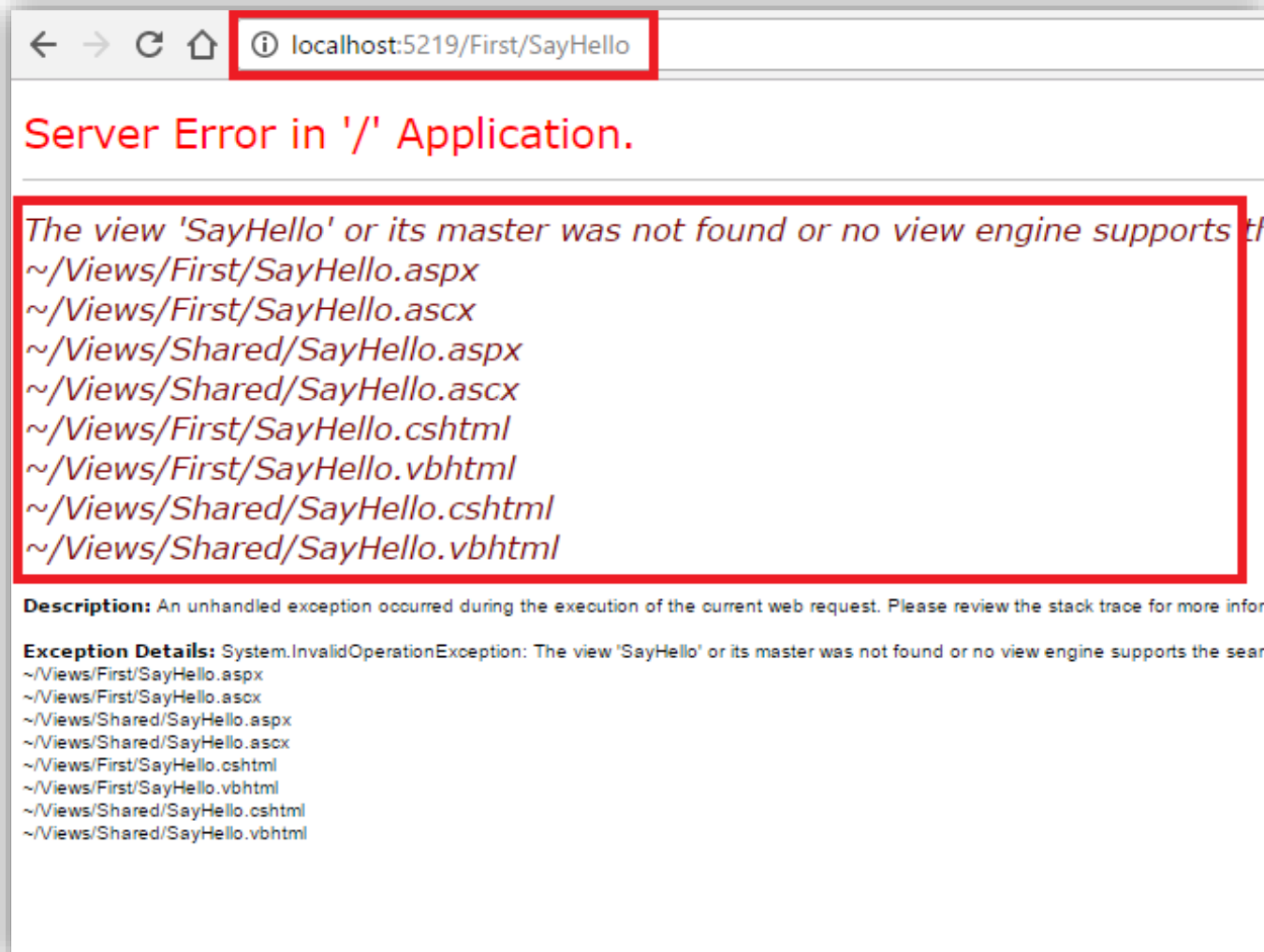
İlk MVC Uygulaması – Adım 6 (V+)

- **.cshtml** uzantılı **Razor HTML** sayfamızı aşağıdaki gibi değiştirelim.

```
@{  
    Layout = null;  
}  
  
<!DOCTYPE html>  
<html>  
<head>  
    <meta name="viewport" content="width=device-width" />  
    <title>HelloView</title>  
</head>  
<body>  
    <div>  
        <h1>İlk ASP.NET MVC uygulamamız.</h1>  
    </div>  
</body>  
</html>
```

İlk MVC Uygulaması – Adım 6 (V+)

- **View** yaratıldıktan sonra artık çalıştırabilir miyiz?



The screenshot shows a web browser window with the address bar containing `localhost:5219/First/SayHello`. The main content area displays a red error message: "Server Error in '/' Application." Below this, a list of search paths for the view is shown, including `~/Views/First/SayHello.aspx`, `~/Views/First/SayHello.ascx`, `~/Views/Shared/SayHello.aspx`, `~/Views/Shared/SayHello.ascx`, `~/Views/First/SayHello.cshtml`, `~/Views/First/SayHello.vbhtml`, `~/Views/Shared/SayHello.cshtml`, and `~/Views/Shared/SayHello.vbhtml`. A description and exception details are also visible at the bottom.

← → ↻ 🏠 ⓘ localhost:5219/First/SayHello

Server Error in '/' Application.

The view 'SayHello' or its master was not found or no view engine supports the search paths:

- ~/Views/First/SayHello.aspx
- ~/Views/First/SayHello.ascx
- ~/Views/Shared/SayHello.aspx
- ~/Views/Shared/SayHello.ascx
- ~/Views/First/SayHello.cshtml
- ~/Views/First/SayHello.vbhtml
- ~/Views/Shared/SayHello.cshtml
- ~/Views/Shared/SayHello.vbhtml

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information.

Exception Details: System.InvalidOperationException: The view 'SayHello' or its master was not found or no view engine supports the search paths.

- ~/Views/First/SayHello.aspx
- ~/Views/First/SayHello.ascx
- ~/Views/Shared/SayHello.aspx
- ~/Views/Shared/SayHello.ascx
- ~/Views/First/SayHello.cshtml
- ~/Views/First/SayHello.vbhtml
- ~/Views/Shared/SayHello.cshtml
- ~/Views/Shared/SayHello.vbhtml

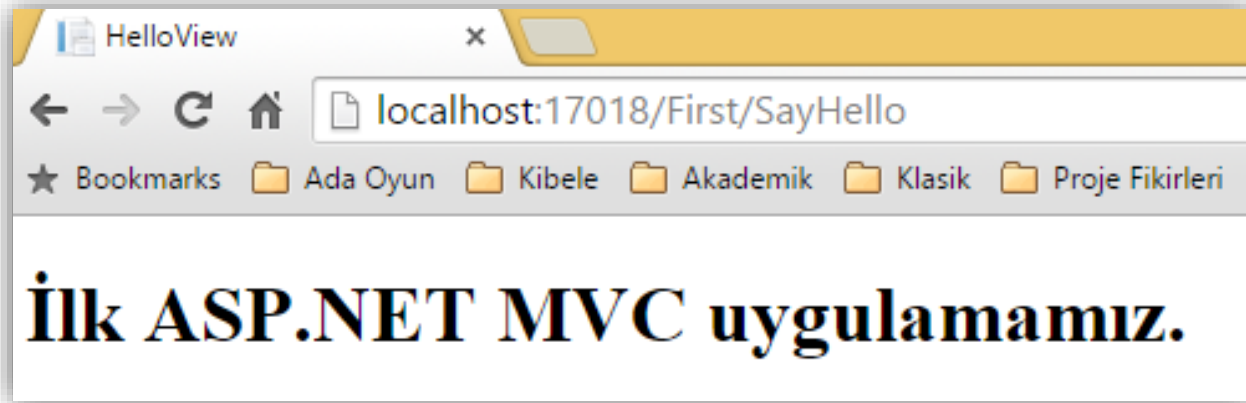
İlk MVC Uygulaması – Adım 7 (CV+)

- **Controller** içerisinde **View** ile **bağlantı kuralım**.
- FirstController sınıfının **SayHello** action'ı HelloView'ı *geri döndürecek* şekilde değiştirelim.

```
//GET: SayHello  
0 references  
public ActionResult SayHello()  
{  
    return View("HelloView");  
}
```

İlk MVC Uygulaması – Adım 8 (Run)

- **CTRL + F5** ile çalıştırılım.
- URL aşağıdaki gibi oluşturulmalı.
 - *localhost:portnumber/ControllerName/ActionName*
 - Controller **anahtar kelimesi eklenmemeli**.

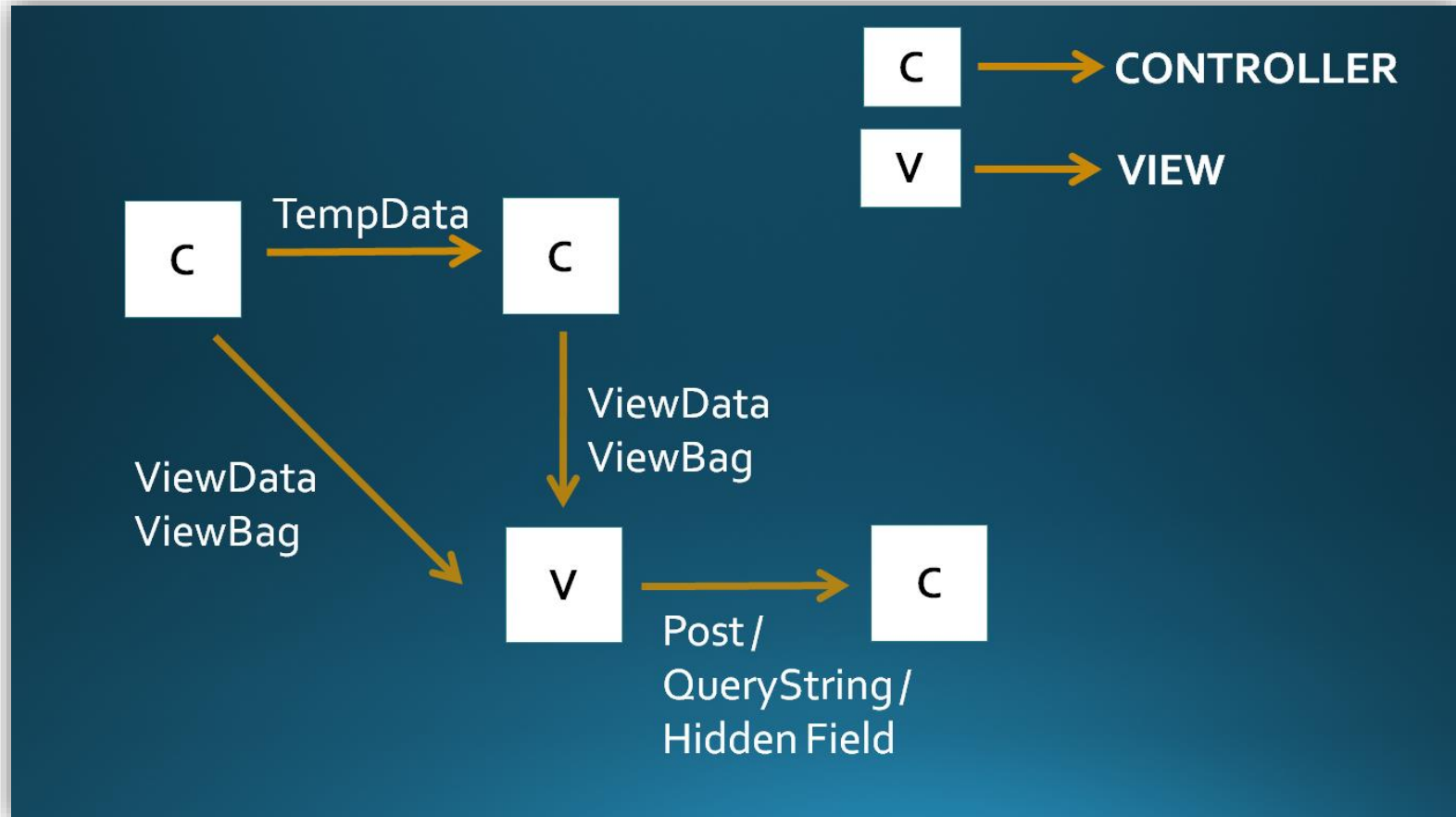


Controller ve View Arasında Veri Taşıma

Controller ve View - Veri Taşıma

- Controller ve View arasında veri taşıma için kullanılan önemli nesnelere aşağıdaki gibidir:
 - **ViewData**
 - **ViewBag**
 - **TempData**
 - **Session**

Controller ve View - Veri Taşıma (devam...)



C<->V Veri Taşıma – ViewData

- **ViewDataDictionary** sınıfından türetilen bir **dictionary** nesnesidir.
- ViewData **ControllerBase** sınıfının bir özelligidir.
- Sadece geçerli istek sırasında okunabilir. Yönlendirme sonrasında **kendini imha eder**, **null değer alır**.
- Veriyi değişkene aktarmak için **tip dönüşümü yapmanız gerekmektedir**.

Örnek: ViewData["Tarih"] = DateTime.Now;

C<->V Veri Taşıma – ViewBag

- ViewBag C# 4.0 ile gelen **dinamik veri tipi** özelliklerinden faydalanır.
- ViewData nın **dinamik** (run time binding) halidir. **Compile time** da *hata vermez* **run time**'da hata verir.
- ViewBag **ControllerBase** sınıfının bir özelligidir.
- Sadece geçerli istek sırasında okunabilir. Yönlendirme sonrasında **kendini imha eder**, **null değer alır**.
- Veriyi değişkene aktarmak için **tip dönüşümü** yapmanız gerekmez.

Örnek: ViewBag.Tarih = `DateTime.Now`;

Örnek1: Controller ve View - Veri Taşıma

- Aşağıdaki 3 değişkeni **PassDataController** isimli bir **Controller**'da dolduralım ve **View**'da değişik şekillerde gösterelim.
 - **isim**: string
 - **yas**: int
 - **renkler**: List<string>

Örnek1: Controller ve View - Veri Taşıma (devam...)

- Yeni bir **ASP.NET MVC** projesi oluşturalım.
- **PassController** isimli bir **Controller** ekleyelim.
- Bu controller'a **Index** isimli bir **View** ekleyelim.

```
public ActionResult Index()
{
    ViewData["isim"] = "Ada KILINÇ";
    ViewBag.yas = 6;

    List<string> renkler = new List<string> { "siyah", "beyaz" };
    ViewData["renkler"] = renkler;

    return View();
}
```

Örnek1: Controller ve View - Veri Taşıma (devam...)

- **ViewData** ile **1. Gösterim**

```
<div>
  <p>İsim: @ViewData["isim"]</p>
  <p>Yaş: @ViewData["yas"]</p>
  <p>
    Sevdiği Renkler:
    <ul>
      @foreach (var renk in (List<string>)ViewData["renkler"])
      {
        <li>@renk</li>
      }
    </ul>
  </p>
</div>
```

- **DİKKAT:** ViewBag ile doldurduğumuz **yas** değişkenine **ViewData** ile ulaşıyoruz.

Örnek1: Controller ve View - Veri Taşıma (devam...)

- **ViewData** ile **2. Gösterim (Değişkenlere atarak)**

```
<div>
  @{
    string isim = (string)ViewData["isim"];
    int yas = (int)ViewData["yas"];
    <p>Isim: @isim</p>
    <p>Yaş: @yas</p>
    <p>
      Sevdği Renkler:
      <ul>
        @foreach (var renk in (List<string>)ViewData["renkler"])
        {
          <li>@renk</li>
        }
      </ul>
    </p>
  }
</div>
```

- **DİKKAT:** ViewData ile değişkene atarken **mutlaka typecast** yani **tip dönüşümü** yapmalıyız.

Örnek1: Controller ve View - Veri Taşıma (devam...)

- **ViewBag** ile **1. Gösterim**

```
<div>
  <p>Isim: @ViewBag.isim</p>
  <p>Yaş: @ViewBag.yas</p>
  <p>
    Sevdiği Renkler:
    <ul>
      @foreach (var renk in ViewBag.renkler)
      {
        <li>@renk</li>
      }
    </ul>
  </p>
</div>
```

- **DİKKAT:** ViewBag ile **renkler** **değişkenine** **ulaşırken** **typecast yapmıyoruz.**

Örnek1: Controller ve View - Veri Taşıma (devam...)

- **ViewBag** ile **2. Gösterim**

```
<div>
  @{
    string isim = ViewBag.isim;
    int yas = ViewBag.yas;
    <p>Isim: @isim</p>
    <p>Yaş: @yas</p>
    <p>
      Sevdği Renkler:
      <ul>
        @foreach (var renk in ViewBag.renkler)
        {
          <li>@renk</li>
        }
      </ul>
    </p>
  }
</div>
```

- **DİKKAT:** ViewBag ile değişkene atarken **typecast** yani **tip dönüşümü** yapmamıza gerek yok.

C<->V Veri Taşıma – TempData

- **TempDataDictionary** sınıfından türetilen kısa ömürlü **session** da depolanan bir **dictionary** nesnesidir.
- ViewData **ControllerBase** sınıfının bir özelligidir.
- Yönlendirme sonrasında **bir defaya mahsus** hafızada kalır daha sonraki yönlendirmede **kendini imha eder**, **null değer alır**.
- Veriyi değişkene aktarmak için **tip dönüşümü** yapmanız gerekmektedir.

Örnek: TempData["temp"] = "temp";

Örnek1: Controller ve View - Veri Taşıma (devam...)

- **TempData** ile **1. Gösterim (Sadece Test)**

```
public ActionResult Index()
{
    //TempData değişken set
    TempData["temp"] = "temp";

    //TempData Redirect Testi için açıldı
    return RedirectToAction("About");
}

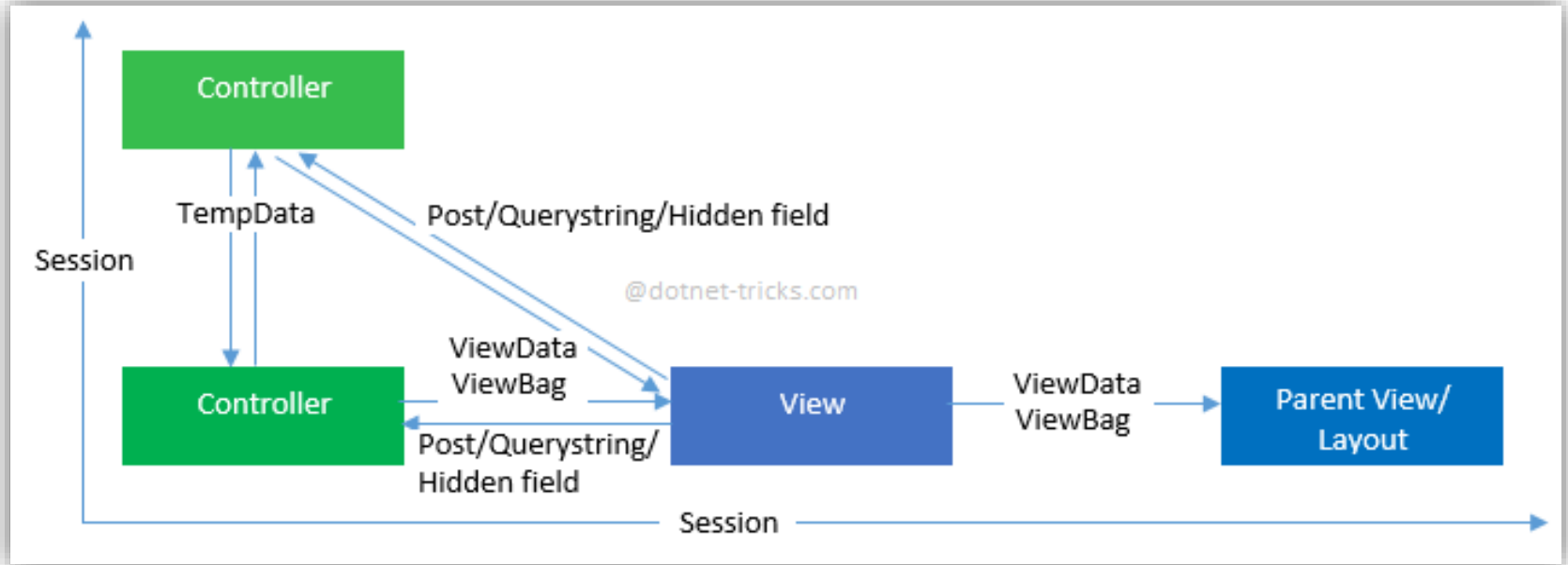
// GET: About
0 references
public ActionResult About()
{
    //Redirect olduktan sonra TempData'yı kontrol ediyoruz.
    var test = TempData["temp"];

    return View();
}
```

- **DEBUG EDEREK GÖRELİM**

Controller ve View - Veri Taşıma (devam...)

ÖZET GÖSTERİM



Yararlanılan Kaynaklar

- <http://www.codeproject.com/Articles/207797/Learn-MVC-Model-View-Controller-step-by-step-in>
- <http://www.questpond.com>
- **Kitap:** Programming ASP.NET MVC 5 A Problem Solution Approach, Nimit Joshi
- <http://www.dotnet-tricks.com/Tutorial/mvc/9KHW190712-ViewData-vs-ViewBag-vs-TempData-vs-Session.html>
- <http://www.mvcsharp.com/MakaleDetay/ViewData-ViewBag-TempData-Nedir>

İYİ ÇALIŞMALAR...

Yrd. Doç. Dr. Deniz KILINÇ

drdenizkilinc@gmail.com

deniz.kilinc@cbu.edu.tr