بسم اللّه الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

آموزش افزودن یک view

مدرس : مهندس افشین رفوآ

آموزش افزودن یک view

در این درس کلاس HelloWorldController را اصلاح کرده تا با استفاده از view template file ها، فرایند ایجاد و ارسال پاسخ های HTML به کاربر سمت سرویس گیرنده را به طور کامل کپسوله سازی کنیم.

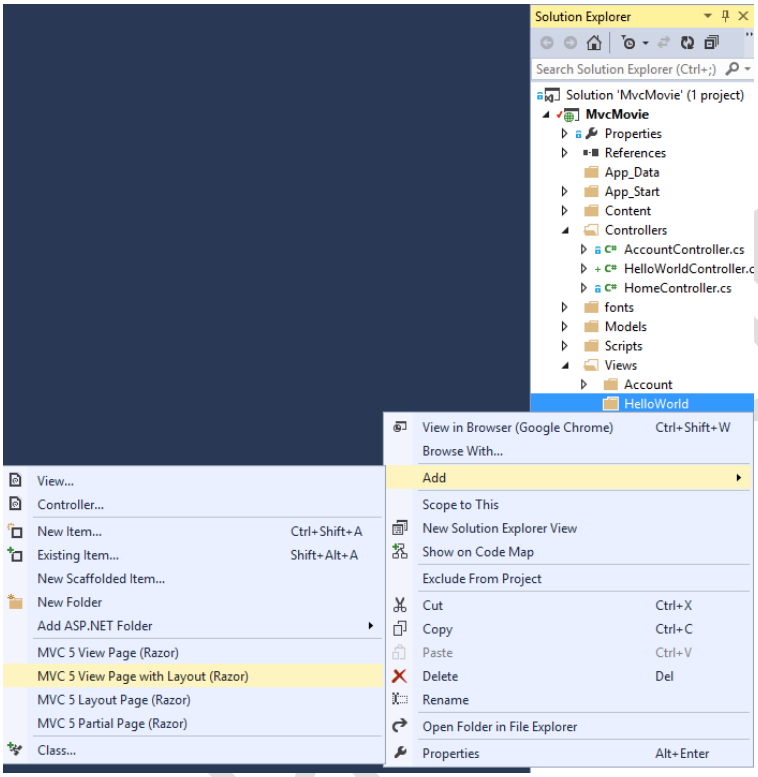
در ابتدا با استفاده از Razor view engine یک template file جدید ایجاد می کنیم. view template های مبتنی بر موتور Razor دارای یک پسوند *.cshtml* هستند و یک روش بسیار موثر در عین حال ساده برای ایجاد خروجی های HTML با استفاده از C# فراهم می نماید. Razor تعداد کاراکترهای تایپی و کلیدهایی که باید به هنگام نوشتن view template فشار دهید را به طور قابل توجهی کاهش می دهد و همچنین تجربه ی کدنویسی روان و سریع را به ارمغان می آورد.

در حال حاضر متد Index به همراه رشته یک پیام ارسال می کند که در کلاس controller، hard-code شده است. متد Index را تغییر داده تا مانند نمونه کد زیر یک شی view برگرداند:

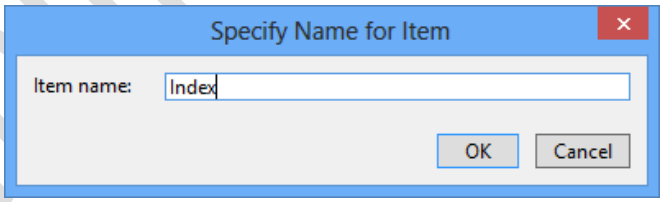
```
public ActionResult Index()
{
  return View();
}
```

متد Index که در مثال فوق مشاهده می کنید از یک template view برای ایجاد پاسخ HTML و ارسال آن به مرورگر استفاده می کند. متدهای controller همچون Index (که از آن ها با نام action methods نیز یاد می شود) به طور معمول یک ActionResult (یا یک کلاس مشتق شده از ActionResult) برمی گردانند و نه نوع های اولیه (primitive types) مانند نوع رشته.

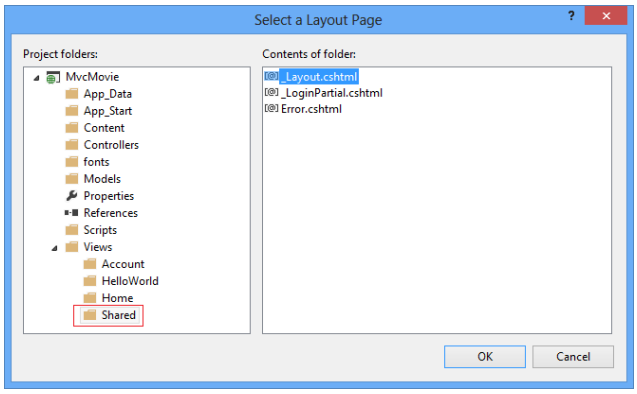
روی پوشه ی *Views\HelloWorld* راست کلیک کرده و Add را انتخاب کنید، سپس MVC 5 View Page with (Layout Razor) را کلیک نمایید.



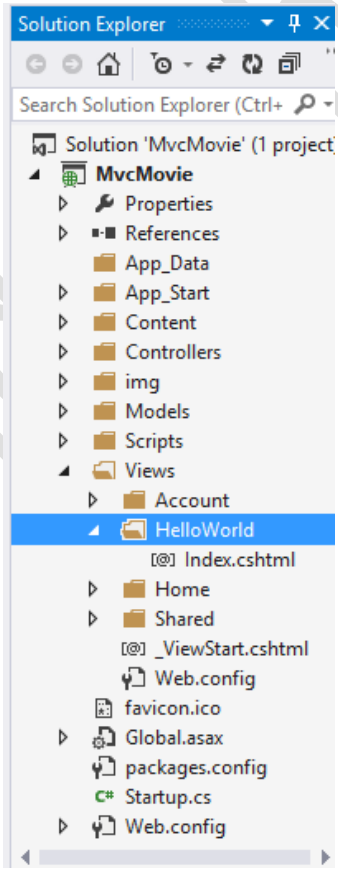
در پنجره ی محاوره ی Specify Name for Item، *Index* را وارد کرده، سپس OK را کلیک نمایید.



در پنجره ی محاوره ی Select a Layout Page، _Layout.cshtml پیش فرض را پذیرفته و را کلیک کنید.



در پنجره ی محاوره ی فوق، پوشه ی *Views\Shared* در قاب سمت چپ انتخاب شده است. اگر از قبل یک layout file سفارشی در پوشه ی دیگر ایجاد کرده باشید، می توانید آن را انتخاب کنید. درباره ی layout file بعدا توضیح خواهیم داد. فایل *MvcMovie\Views\HelloWorld\Index.cshtml* ایجاد شد.

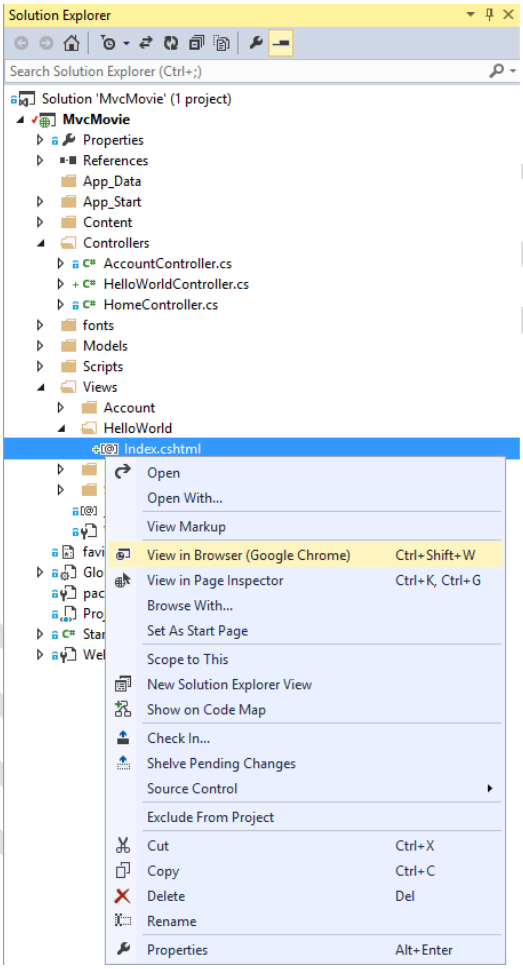


اکنون نشانه گذاری های (markup) رنگی شده را اضافه کنید.

```
@{
  Layout = "~/Views/Shared/_Layout.cshtml";
```

```
}
@{
    ViewBag.Title = "Index";
}
<h2>Index</h2>
<p>Hello from our View Template!</p>
```

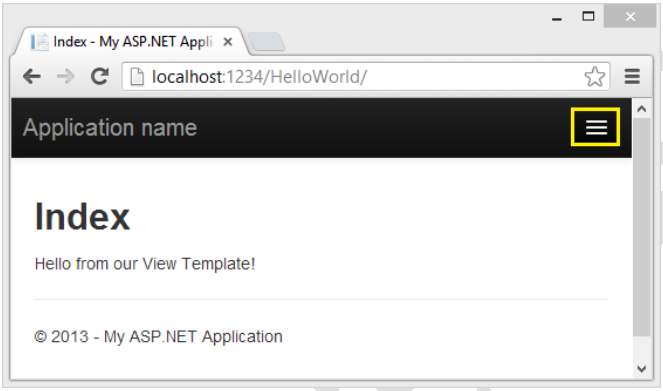
روی فایل *Index.cshtml* راست کلیک کرده و View in Browser را انتخاب کنید.



همچنین می توانید روی فایل *Index.cshtml* راست کلیک کرده و View in Page Inspector را انتخاب نمایید.

یا به روشی دیگر، برنامه را اجرا کرده و به آدرس *http://localhost:xxxx/HelloWorld* (کنترلر HelloWorld) مراجعه نمایید. متد Index در controller کار زیادی انجام نمی داد بلکه صرفا دستور return

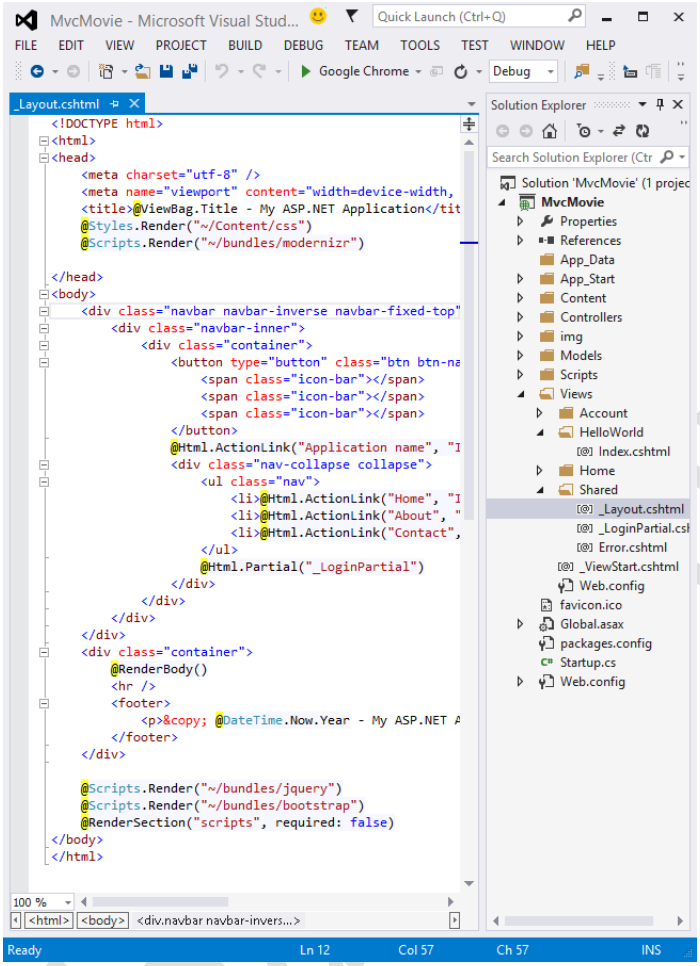
View() را اجرا می کرد. دستور ذکر شده مشخص می کرد که متد موردنظر باید از یک template file برای ارائه ی پاسخ به مرورگر استفاده کند. به این خاطر که شما اسم view template file برای استفاده را به صورت صریح مشخص نکرده اید، ASP.NET MVC به صورت پیش فرض از view file *Index.cshtml* در پوشه ی *Views\HelloWorld* استفاده می کند. تصویر زیر رشته ی "Hello from our View Template!" که در view، hard-code شده را نمایش می دهد.



عالیه نه! اما همان طور که در تصویر بالا مشاهده می کنید، نوار عنوان مرورگر متن "Index My ASP.NET Appli" و لینک بزرگ در بالای صفحه متن "Application name" را نمایش می دهد. بسته به اینکه چقدر صفحه ی پنجره ی مرورگر را کوچک می کنید، ممکن است لازم باشد سه نوار را در سمت راست، بالای صفحه کلیک کنید تا لینک های Home، About، Contact، Register و Log in را مشاهده کنید.

تغییر دادن view ها و صفحات layout

در مرحله ی اول، بایستی لینک "Application name" واقع در قسمت بالای صفحه را تغییر دهید. آن متن بین تمامی صفحات مشترک بوده و در تمامی آن ها موجود می باشد، اما حقیقت این است که تنها در یک قسمت پروژه پیاده سازی شده و در عین حال در تمامی صفحات برنامه نمایش داده می شود. به پوشه ی */Views/Shared* در پنجره ی Solution Explorer مراجعه کرده، سپس فایل *_Layout.cshtml* را باز کنید. فایل نام برده *layout page* نامیده شده و در پوشه ی مشترکی جای گذاری شده که تمامی دیگر صفحات می توانند از آن استفاده کنند.



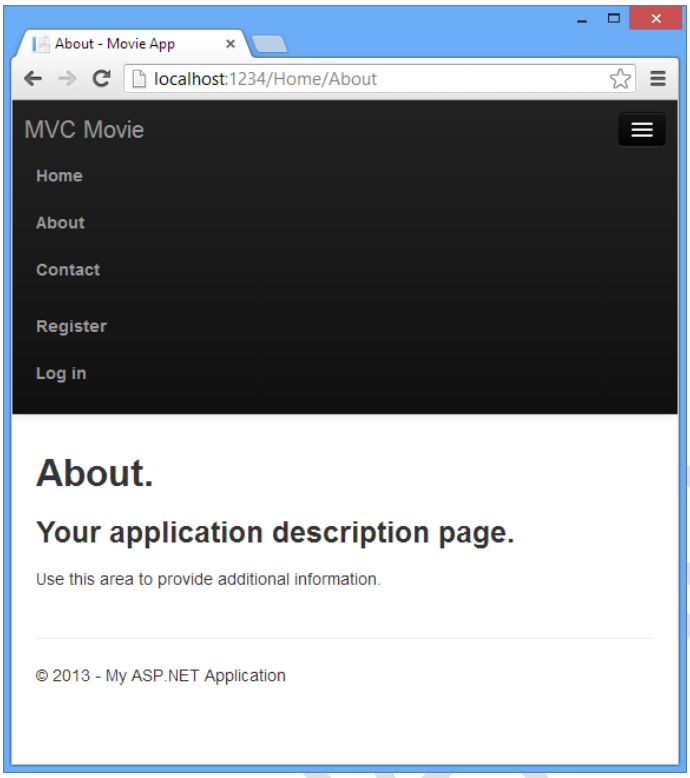
Layout template ها به شما این امکان را می دهند که layout نگهدارنده ی HTML سایت خود را یک مکان واحد مشخص کرده و آن را در چندین صفحه داخل سایت خود اعمال نمایید. خط حاوی @RenderBody() را پیدا کنید. RenderBody یک placeholder است که در آن تمامی صفحات مختص view که ایجاد کرده اید، گنجانده شده در layout page، نمایش داده می شوند (در واقع صفحه محتوا در این قسمت قرار می گیرد. Layout برای نمایش قالب و طرح هر صفحه در برنامه به کار می رود). برای مثال، اگر شما لینک About را انتخاب کنید، *Views\Home\About.cshtml* داخل متد RenderBody، render می شود.

محتوایات المان title را تغییر دهید. ActionLink را در layout template از "Application name" به "MVC Movie" تغییر داده، سپس controller را از Home به Movies تغییر دهید. فایل layout به طور کامل در زیر نمایش داده شده است:

```
<!DOCTYPE html>
<html>
```

```
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>@ViewBag.Title - Movie App</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
</head>
<body>
  <div class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        @Html.ActionLink("MVC Movie", "Index", "Movies", null, new { @class = "navbar-brand" })
      </div>
      <div class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
          <li>@Html.ActionLink("Home", "Index", "Home")</li>
          <li>@Html.ActionLink("About", "About", "Home")</li>
          <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
        </ul>
        @Html.Partial("_LoginPartial")
      </div>
    </div>
  </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>
```

پس از اجرای برنامه مشاهده خواهید کرد که متن "MVC Movie" را نشان می دهد. حال لینک About را کلیک کرده و ببینید که این صفحه نیز مانند صفحه ی قبل متن "MVC Movie" را نمایش می دهد. بنابراین می توان یکبار این تغییر را در layout template اعمال کرد و دید که چگونه این تغییر (عنوان جدید) در تمامی صفحات موجود در سایت منعکس می شود.



هنگامی که برای اولین بار فایل *Views\HelloWorld\Index.cshtm* را ایجاد کردیم، کد زیر را دربرداشت:

```
@{
  Layout = "~/Views/Shared/_Layout.cshtml";
}
```

کد razor بالا، layout page را تعریف یا تنظیم می کند. فایل *Views_ViewStart.cshtml* را بررسی کنید، خواهید دید که دربردارنده ی همان razor markup است. فایل Views_ViewStart.cshtml، layout ای را تعریف می کند که مشترک بوده و مورد استفاده ی تمامی view ها می باشد، بنابراین می توانید توسط comment یا به روش معمولی آن کد را از فایل *Views\HelloWorld\Index.cshtml* حذف کنید.

```
@*@{
    Layout = "~/Views/Shared/_Layout.cshtml";
  }*@
@{
  ViewBag.Title = "Index";
}
<h2>Index</h2>
<p>Hello from our View Template!</p>
```

می توانید با استفاده از خاصیت Layout، یک layout view متفاوت تنظیم کنید و یا آن را روی مقدار null تنظیم نمایید تا هیچ layout file ای مورد استفاده قرار نگیرد.

حال عنوان Index view را تغییر دهید.

MvcMovie\Views\HelloWorld\Index.cshtml را باز کنید. دو مکان برای ایجاد تغییر وجود دارد: اول، آن متنی که در عنوان مرورگر ظاهر می شود و دیگری در header دوم (المان <h2>). آن را تا حدی تغییر دهید که بتوانید تشخیص دهید کدام تکه از کد، کدام بخش از برنامه را تغییر داده است.

```
@{
  ViewBag.Title = "Movie List";
}
<h2>My Movie List</h2>
<p>Hello from our View Template!</p>
```

برای مشخص کردن یک title یا عنوان برای صفحه، تکه کد بالا خاصیت Title شی ViewBag (که در view template فایل *Index.cshtml* قرار دارد) را با مقدار "Movie List" تنظیم می کنیم. دقت داشته باشید که layout template (*Views\Shared_Layout.cshtml*) مقدار موجود در تگ <title> را به عنوان بخشی از قسمت <head> صفحه ی HTML که قبلا مورد اصلاح قرار گرفته بود، استفاده می کند.

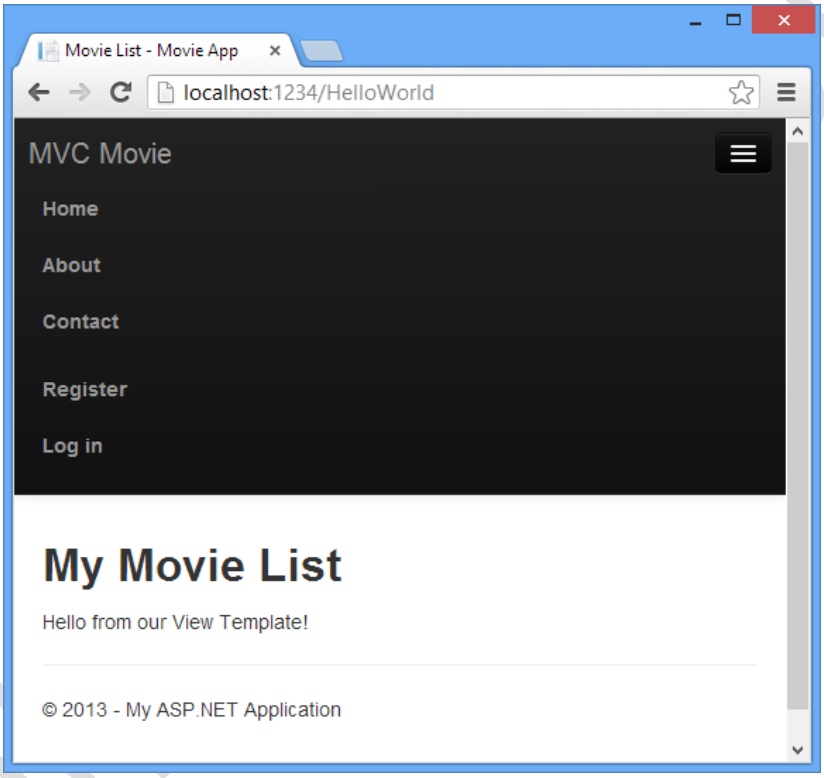
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>@ViewBag.Title - Movie App</title>
  @Styles.Render("~/Content/css")
  @Scripts.Render("~/bundles/modernizr")
</head>
```

با استفاده از ViewBag، می توان به راحتی پارامترهای دیگری را بین view template و layout file خود رد و بدل (پاس داد) کرد.

حال برنامه را اجرا کنید. توجه داشته باشید که عنوان مرورگر، heading اصلی، heading دوم تغییر یافته اند. (در صورت مشاهده نکردن تغییر، نگران نباشید. ممکن است مرورگر شما محتوای حافظه نهان یا cache را خوانده و برای شما به نمایش بگذارد. با استفاده از میانبر Ctrl+F5، کاری کنید که پاسخ از سرور بارگذاری

شود.) عنوان مرورگر توسط ViewBag.Title که در view template (*Index.cshtml*) و همچنین -" Movie App" که در داخل layout file اضافه کردید، ایجاد و تنظیم می شود.

همچنین دقت کنید که چگونه محتوای موجود در view template "*Index.cshtml*" با محتوای view template "*_Layout.cshtml*" ادغام شده و یک پاسخ HTML به مرورگر ارسال شده است. layout template به شما این امکان را می دهد تا به آسانی تغییراتی را وارد کنید که در تمامی صفحات برنامه اعمال می شوند.



اما باید توجه داشته باشید که داده های ما (در این مورد پیغام "Hello from our View Template!") hard-code شده هستند. برنامه ی MVC حاضر یک بخش به نام view یا نما و یک بخش به نام controller یا کنترلگر دارد، اما در آن خبری از یک model نیست. در مبحث بعدی نحوه ی ایجاد یک پایگاه داده و واکشی اطلاعات مدل از آن پایگاه داده را برای شما شرح می دهیم.

ارسال داده ها از controller به view

پیش از پرداختن به پایگاه داده و بحث درباره ی model ها، لازم است نحوه ی ارسال اطلاعات از controller به view را تشریح کنیم. کلاس های controller در پاسخ به درخواست URL دریافتی فراخوانده می شوند. کلاس controller جایی است که در آن کدهایی می نویسید که درخواست های مرورگر را مدیریت می کند ، داده از بانک اطلاعاتی واکشی کرده و در نهایت تصمیم می گیرد چه نوع پاسخی به مرورگر بازگرداند. در پی آن view template ها می توانند از یک controller بکار گرفته شده و پاسخ مناسب HTML به مرورگر را ایجاد و قالب بندی (format) کنند.

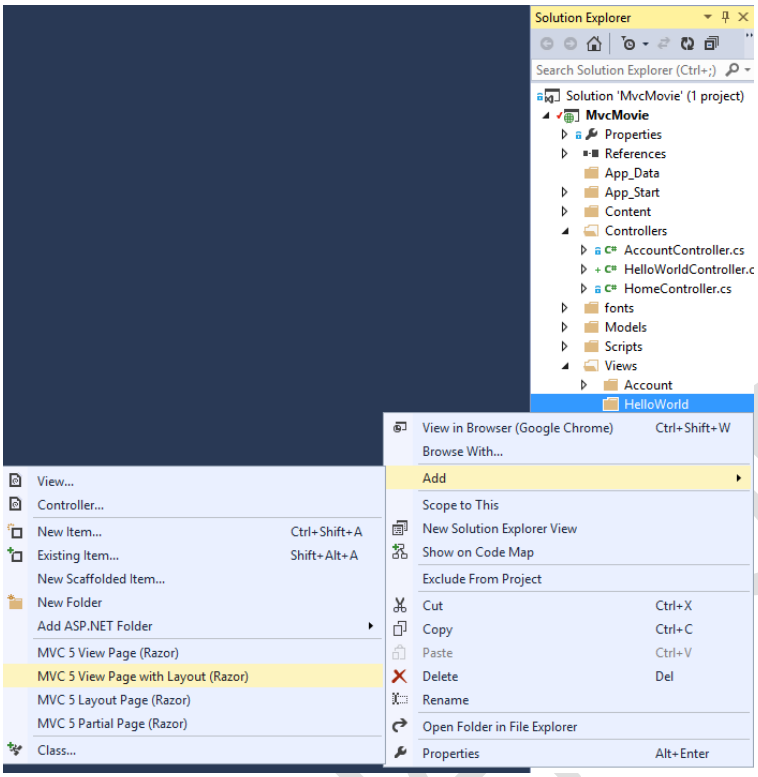
Controller ها مسئول فراهم نمودن تمامی داده ها و اشیا لازم که view template برای ارائه ی خروجی به مرورگر به آن ها نیاز دارد، می باشد (Controller قسمتی از برنامه است که تعامل با کاربر را مدیریت می کند.نوعا controller داده ها را از یک view می خواند، ورودی کاربر را کنترل می کند، و داده های ورودی را به model می فرستد). یک نکته ی بسیار مهم: یک view template هیچگاه نباید منطق business اجرا کند یا با پایگاه داده به طور مستقیم تعامل داشته باشد، بلکه یک view template صرفا باید با داده هایی که توسط controller در اختیار آن قرار گرفته، سروکار داشته باشد. دنبال کردن این رویکرد " جداسازی وظایف/separation of concerns " به شما در مرتب نگه داشتن کد و حفظ قابلیت آزمایش پذیری (testable) آن کمک شایانی می کند.

در حال حاضر، action method موجود در کلاس HelloWorldController که Welcome نام گذاری شده، یک پارامتر name و numTimes می پذیرد، سپس مقادیر را به طور مستقیم به مرورگر (به صورت خروجی) تحویل (output) می دهد. به جای اینکه کاری کنیم تا controller این خروجی یا پاسخ را به صورت رشته ارائه دهد (render کند)، controller را گونه ای تغییر می دهیم که از یک template view استفاده کند. view template یک پاسخ پویا (dynamic response) ارائه می کند، بدین معنا که شما باید فقط دادهای مورد نیاز و مناسب را از controller به view ارسال نمایید تا بدین وسیله پاسخ و خروجی مناسب ایجاد شود. برای این منظور می توان کاری کرد که controller داده های پویایی (پارامترها) که view template به آن ها نیاز دارد را داخل یک شی ViewBag جای گذاری کند. پس از انجام این کار، view template می تواند به راحتی به آن داده های پویا از داخل شی ViewBag دسترسی داشته باشد.

به فایل *HelloWorldController.cs* بازگشته و متد Welcome را اصلاح کنید تا بتوان مقادیر Message و NumTimes را به شی ViewBag افزود. ViewBag یک شی پویا (dynamic object) است، بدین معنا که می توان هر چیزی که دلخواه تان است را داخل آن قرار دهید؛ ViewBag تا زمانی که چیزی درون آن قرار نداده اید، هیچ خاصیت تعریف شده ای نخواهد داشت. ASP.NET MVC model binding system به صورت خودکار پارامترهای نامگذاری شده (name و numTimes) را از query string در نوار آدرس به پارامترهای موجود در متد نگاشت می کند. فایل کامل *HelloWorldController.cs* بدین شکل است:

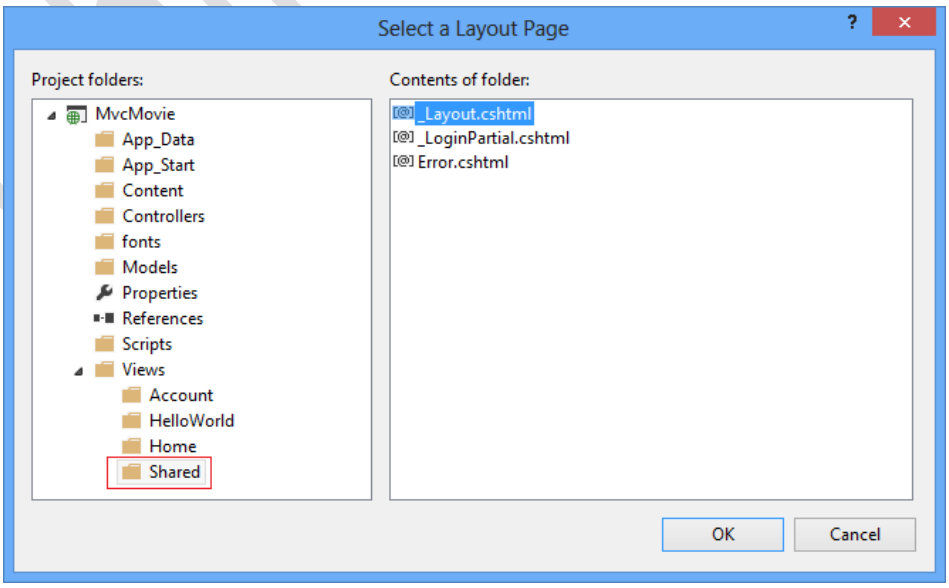
```
using System.Web;
using System.Web.Mvc;

namespace MvcMovie.Controllers
{
  public class HelloWorldController : Controller
  {
    public ActionResult Index()
    {
      return View();
    }
    public ActionResult Welcome(string name, int numTimes = 1)
    {
      ViewBag.Message = "Hello " + name;
      ViewBag.NumTimes = numTimes;
      return View();
    }
  }
}
```

اکنون شی ViewBag دربردارنده ی داده هایی است که به صورت خودکار به view ارسال می شوند. در مرحله ی بعد به یک view template به نام welcome نیاز دارید! در منو Build، Build Solution را انتخاب کرده (و یا دکمه های Ctrl+Shift+B را به طور همزمان فشار دهید) تا پروژه ترجمه/کامپایل شود. روی پوشه ی Views\HelloWorld راست کلیک کرده و Add را کلیک نمایید. حال گزینه ی click MVC 5 View Page (Layout Razor) with را انتخاب نمایید.



در پنجره ی محاوره ی Specify Name for Item، *Welcome* را وارد کرده، سپس OK را کلیک کنید.

در پنجره ی محاوره ی Select a Layout Page، _Layout.cshtml پیش فرض را پذیرفته و OK را کلیک کنید.



فایل *MvcMovie\Views\HelloWorld\Welcome.cshtml* ایجاد شد.

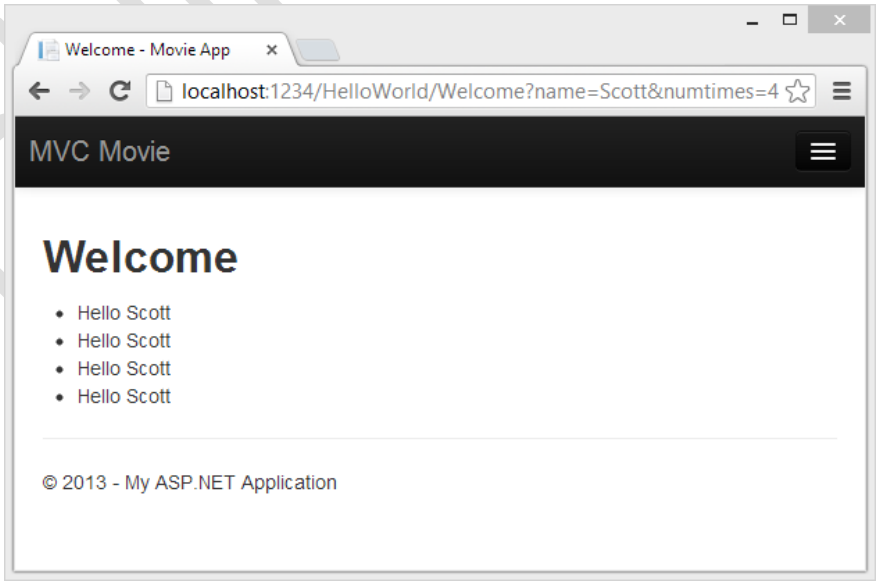
Markup موجود در فایل *Welcome.cshtml* را جایگزین کنید. یک حلقه ایجاد می کنیم که رشته ی "Hello" را به تعداد دفعات مشخص شده از طرف کاربر تکرار کند. فایل کامل *Welcome.cshtml* در زیر به نمایش گذاشته شده است:

```
@{
  ViewBag.Title = "Welcome";
}
<h2>Welcome</h2>
<ul>
  @for (int i = 0; i < ViewBag.NumTimes; i++)
  {
    <li>@ViewBag.Message</li>
  }
</ul>
```

برنامه را اجرا کرده و به آدرس زیر مراجعه کنید:

http://localhost:xx/HelloWorld/Welcome?name=Scott&numtimes=4

اکنون داده ها از URL دریافت شده و از طریق model binder به controller ارسال می گردد. Controller داده های مورد نظر را در شی ViewBag به صورت پکیج درآورده، سپس آن شی را به view پاس می دهد. View نیز داده ها را در قالب HTML به کاربر نمایش می دهد.



در نمونه ی فوق، با استفاده از یک شی ViewBag، اطلاعات را از controller به view ارسال کردیم. در فواصل بعدی آموزش حاضر، از یک view model برای ارسال اطلاعات از controller به view استفاده می کنیم. این رویکرد (ارسال داده از controller به view) ارسال اطلاعات، در مقایسه با روش قبلی از محبوبیت بیشتری برخوردار است.

در این درس یک مدل ایجاد کردیم، ولی ساخت پایگاه داده را در مبحث بعدی مطرح خواهیم کرد. در درس بعدی با استفاده از دانش کسب شده، یک پایگاه داده از فیلم ها ایجاد می کنیم.