WEB SYSTEMS 202

Marinda Taljaard
Office 09 02 29
marinda.taljaard@mandela.ac.za

Model-View-Controller in ASP.NET

Admin

- Behind schedule with test marking
  - Hope to have unmoderated marks on Moodle on Monday
  - Moderator back from sick leave on Monday
  - Scripts available after Moderation process completed
- Missed the test?
  - Sick-test Date and Time to be confirmed later

Practical 07 Feedback

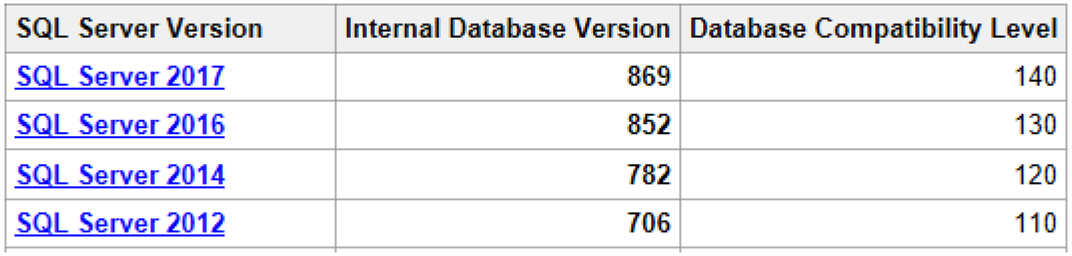
- Only 15 submissions – why
  - Other academic commitments?
  - MVC issues?

- When adding a controller – blank or with scaffolding (skeleton methods and code included)
- Add views to link to methods

- Some feedback aspects discussed now, some other elements discussed during the lecture

Practical 07 Feedback

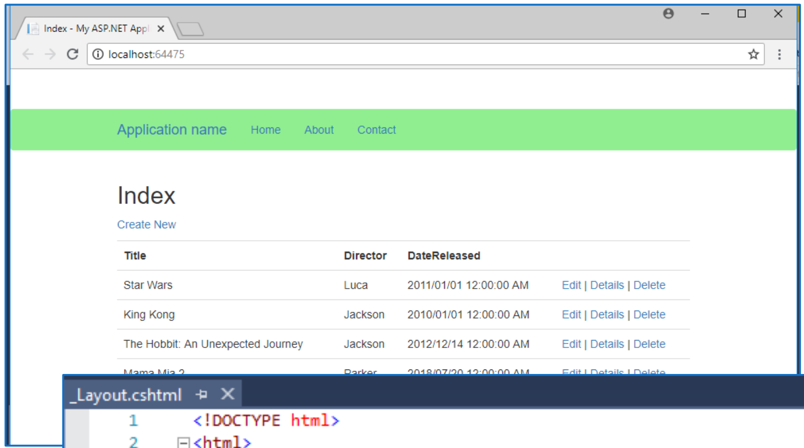
- Problem with both: File, Open, Folder & File, Open, Website
  - Application runs - but
  - VS does not recognise web application as MVC application – cannot add controllers - must open with solution file
- Possible SQL version issues
  - …the database cannot be opened because it is version 852…
  - Version in lab: SQL Server 2014; but possibly 2016 SQL local database
  - Yet to figure out the ideal solution….



SQL Server Version	Internal Database Version	Database Compatibility Level
SQL Server 2017	869	140
SQL Server 2016	852	130
SQL Server 2014	782	120
SQL Server 2012	706	110

Practical 07 Feedback

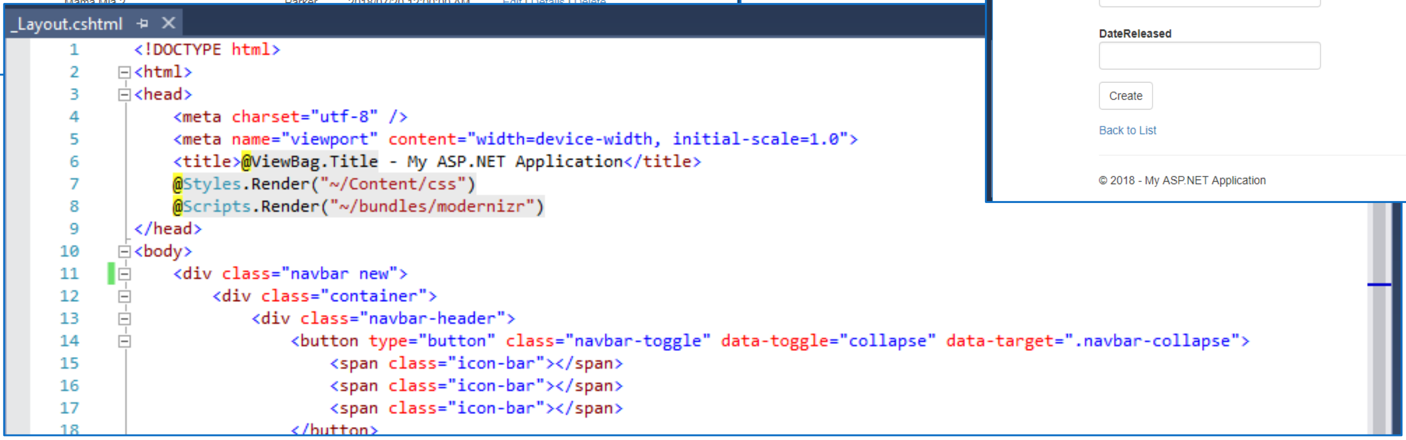
- Updating the "default" settings
  - Mostly in _Layout file
  - Determine whether any changes are required in View



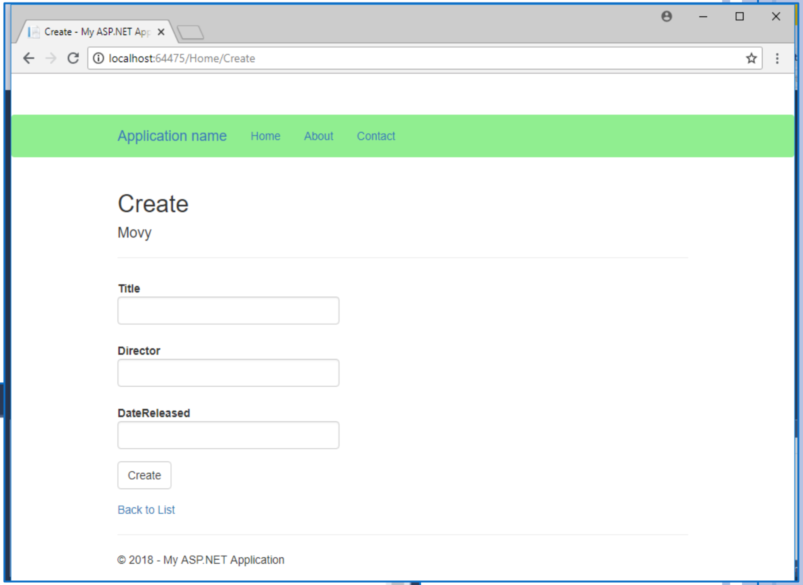
Index

Create New

Title	Director	DateReleased	
Star Wars	Luca	2011/01/01 12:00:00 AM	Edit | Details | Delete
King Kong	Jackson	2010/01/01 12:00:00 AM	Edit | Details | Delete
The Hobbit: An Unexpected Journey	Jackson	2012/12/14 12:00:00 AM	Edit | Details | Delete
Mama Mia 2	Parker	2018/07/20 12:00:00 AM	Edit | Details | Delete



```
<!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 new">
        <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>
```



Create

Movy

Title

Director

DateReleased

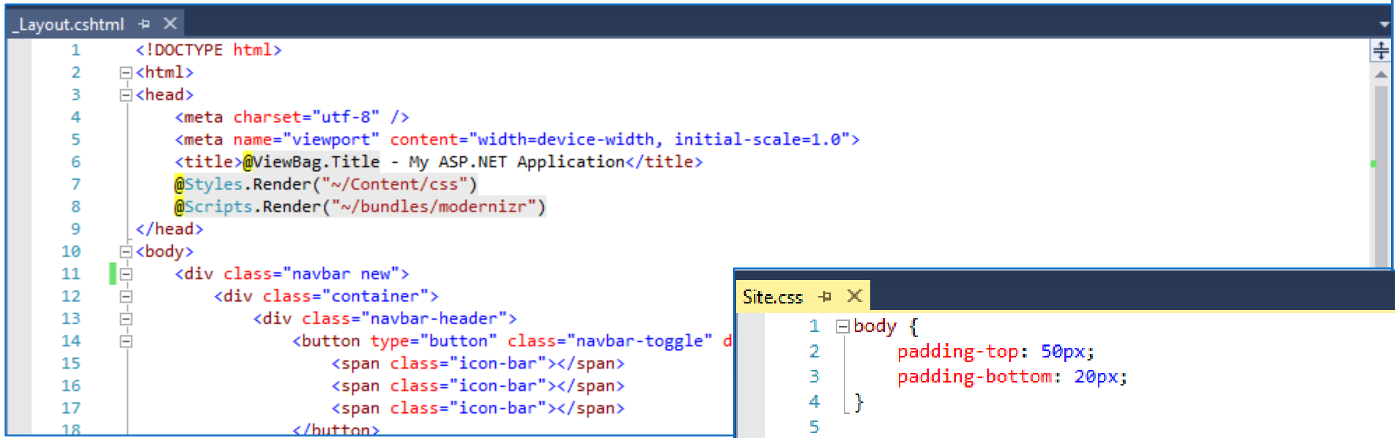
Create

Back to List

© 2018 - My ASP.NET Application

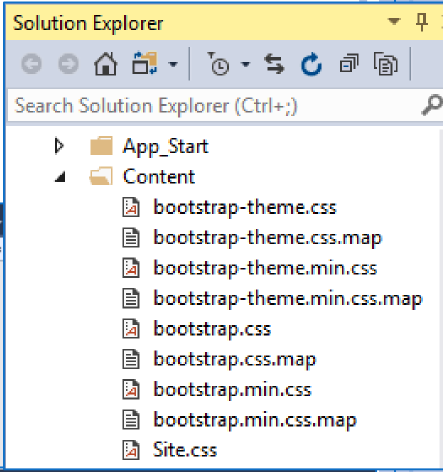
PRACTICAL 07 FEEDBACK – SITE.CSS

- Overriding some bootstrap css instructions

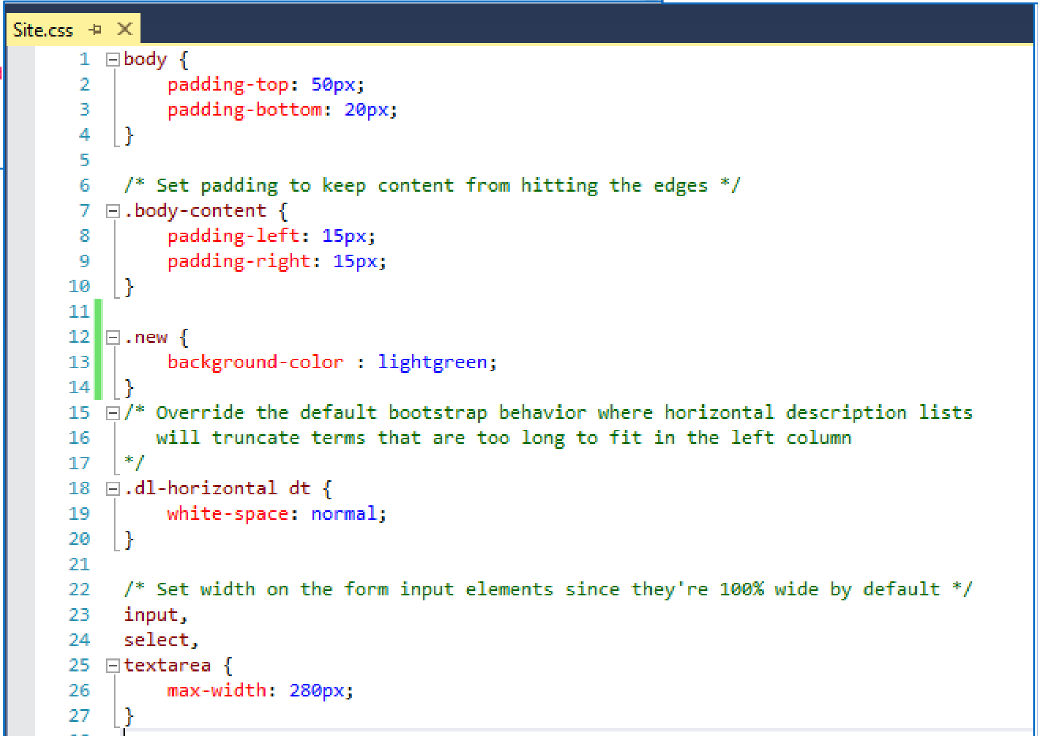


_Layout.cshtml

```
<!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 new">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" d
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
```



```
App_Start
Content
    bootstrap-theme.css
    bootstrap-theme.css.map
    bootstrap-theme.min.css
    bootstrap-theme.min.css.map
    bootstrap.css
    bootstrap.css.map
    bootstrap.min.css
    bootstrap.min.css.map
    Site.css
```

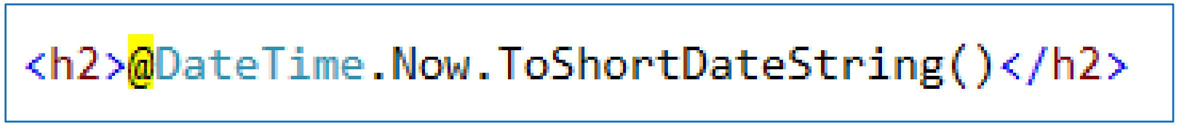


Site.css

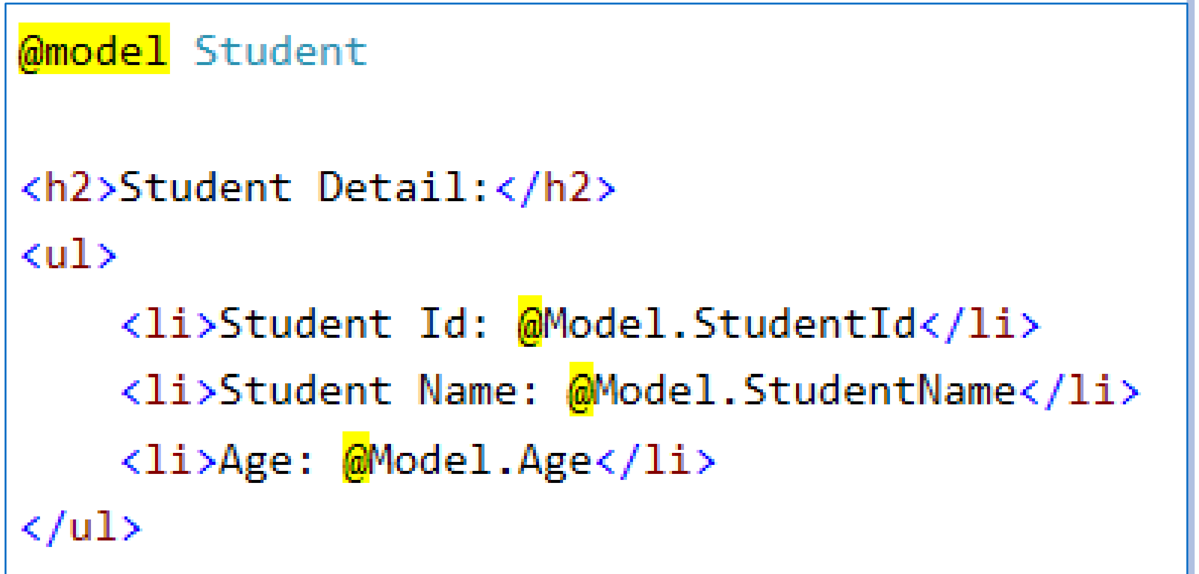
```
body {
    padding-top: 50px;
    padding-bottom: 20px;
}

/* Set padding to keep content from hitting the edges */
.body-content {
    padding-left: 15px;
    padding-right: 15px;
}

.new {
    background-color : lightgreen;
}
/* Override the default bootstrap behavior where horizontal description lists
   will truncate terms that are too long to fit in the left column
*/
.dl-horizontal dt {
    white-space: normal;
}

/* Set width on the form input elements since they're 100% wide by default */
input,
select,
textarea {
    max-width: 280px;
}
```

Razor syntax

- View engine supported in ASP.NET MVC
- Allows a mix of HTML and server side code using C#
  - Files has .cshtml extension
- Start with @ symbol to write server side C# code with html code
- Examples:



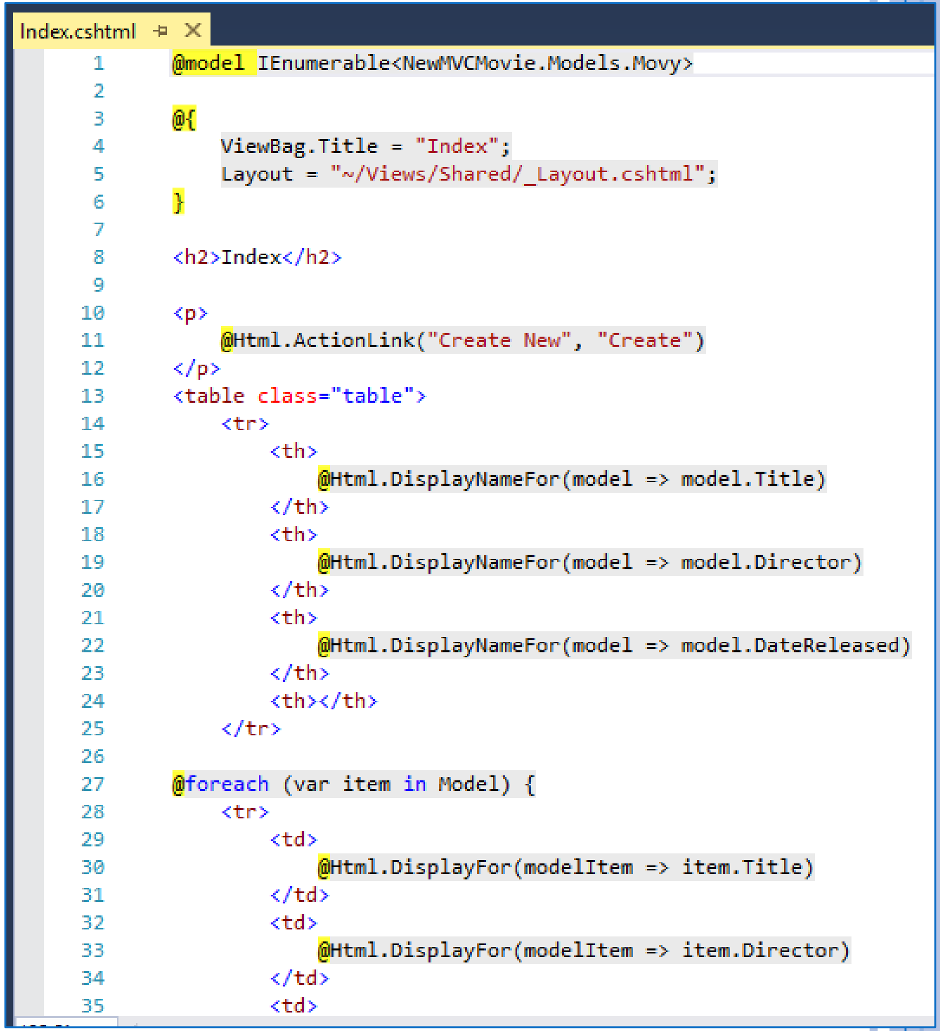
```
<h2>@DateTime.Now.ToShortDateString()</h2>
```



```
@model Student

<h2>Student Detail:</h2>
<ul>
    <li>Student Id: @Model.StudentId</li>
    <li>Student Name: @Model.StudentName</li>
    <li>Age: @Model.Age</li>
</ul>
```

HTML Helper Class

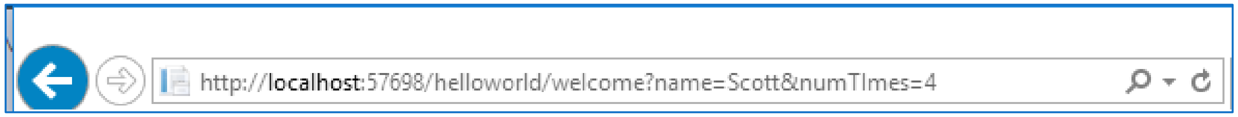
- Generates html elements
- @Html – object of helper class
- DisplayNameFor(); ActionLink() – extension methods
- @Html.Actionlink(…) generates an anchor tag <a…> </a>
- @Html.DisplayFor(…) generates a html string

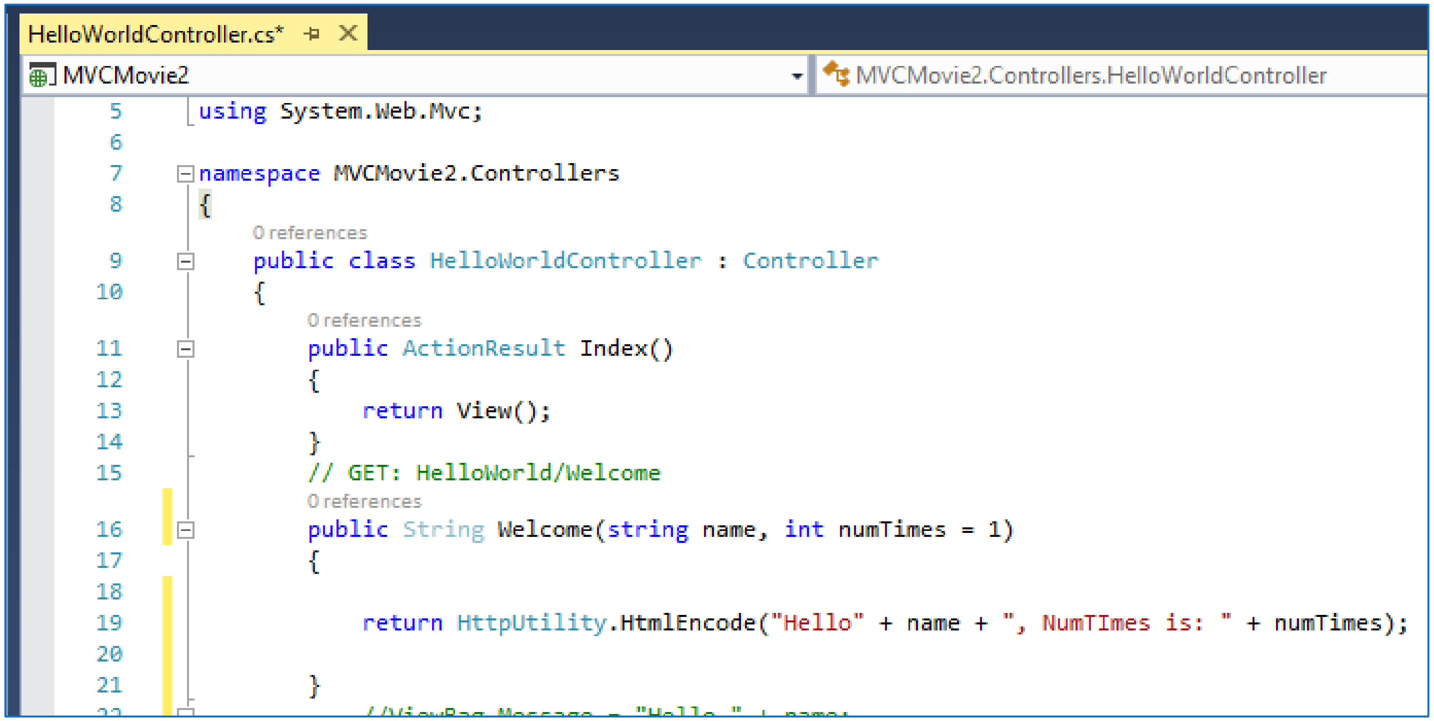

Index.cshtml

```
@model IEnumerable<NewMVCMovie.Models.Movy>

@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Title)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Director)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.DateReleased)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Title)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Director)
        </td>
        <td>
```

PASSING DATA BETWEEN COMPONENTS

- Specifying parameters when routing to a page
  - This example just creates html code which the browser will render (not using a view)

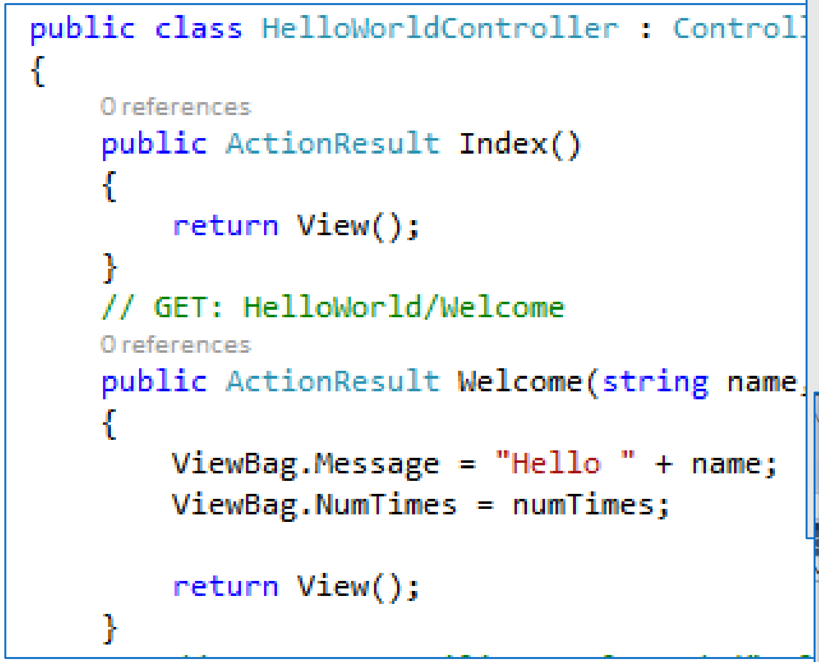

http://localhost:57698/helloworld/welcome?name=Scott&numTImes=4



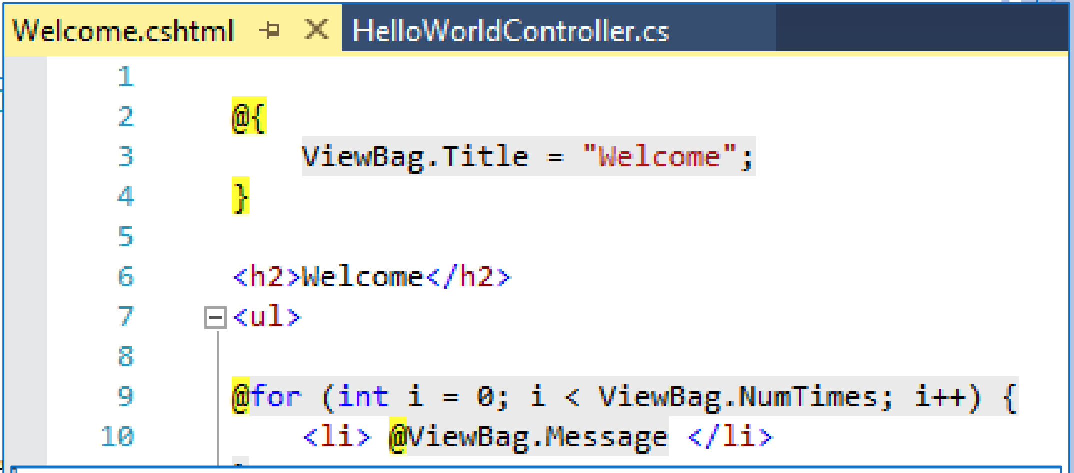
```
using System.Web.Mvc;

namespace MVCMovie2.Controllers
{
    0 references
    public class HelloWorldController : Controller
    {
        0 references
        public ActionResult Index()
        {
            return View();
        }
        // GET: HelloWorld/Welcome
        0 references
        public String Welcome(string name, int numTimes = 1)
        {

            return HttpUtility.HtmlEncode("Hello" + name + ", NumTImes is: " + numTimes);

        }
```

Passing data between components

- Using ViewBag
  - Dynamic object – add any property to it (no compile-time checking though)

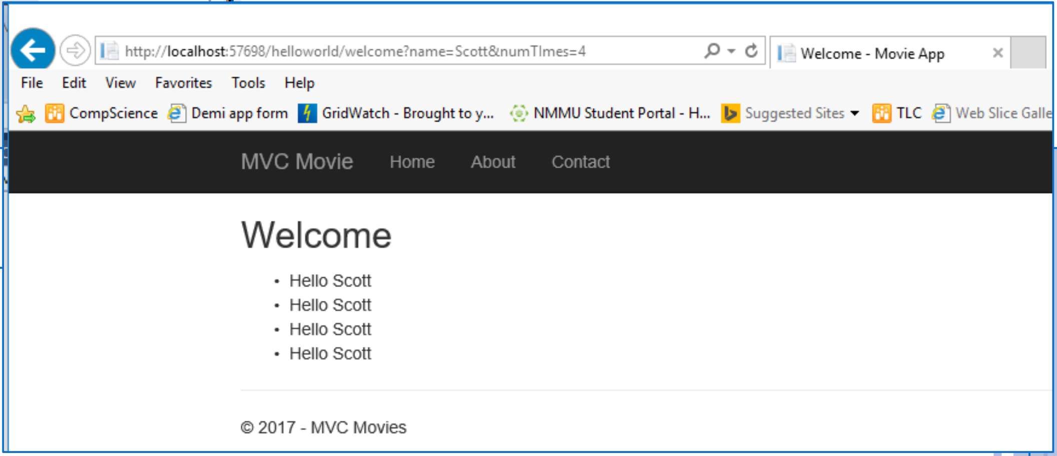


```
public class HelloWorldController : Controll
{
    0 references
    public ActionResult Index()
    {
        return View();
    }
    // GET: HelloWorld/Welcome
    0 references
    public ActionResult Welcome(string name,
    {
        ViewBag.Message = "Hello " + name;
        ViewBag.NumTimes = numTimes;

        return View();
    }
```



Welcome.cshtml HelloWorldController.cs

```
@{
    ViewBag.Title = "Welcome";
}

<h2>Welcome</h2>
<ul>

@for (int i = 0; i < ViewBag.NumTimes; i++) {
    <li> @ViewBag.Message </li>
```



http://localhost:57698/helloworld/welcome?name=Scott&numTimes=4

Welcome - Movie App
File Edit View Favorites Tools Help
CompScience Demi app form GridWatch - Brought to y... NMMU Student Portal - H... Suggested Sites TLC Web Slice Galle
MVC Movie Home About Contact

Welcome

- Hello Scott
- Hello Scott
- Hello Scott
- Hello Scott

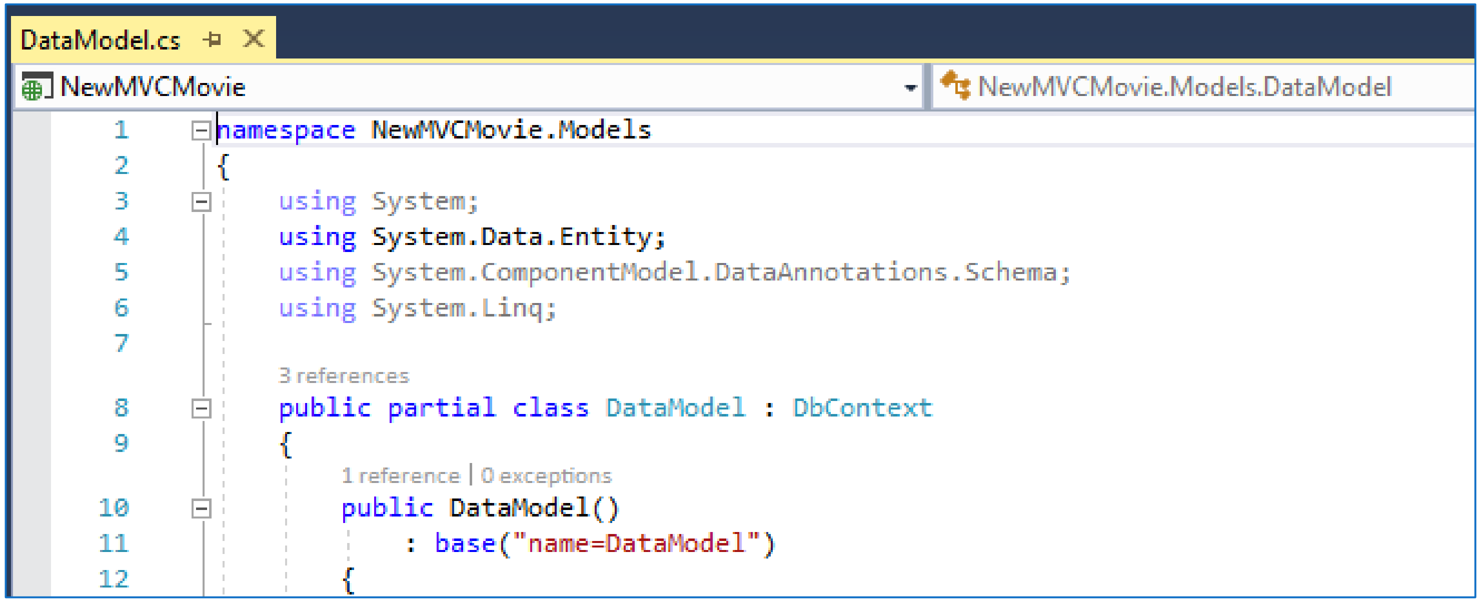
© 2017 - MVC Movies

Strongly typed models

- Better compile time checking
- Scaffolding mechanism in VS uses this approach when using the templates (creating methods and views)
- @model Keyword
  - Used at the top of a View
  - Specifies the type of object the view expects

Closer look at components of the model

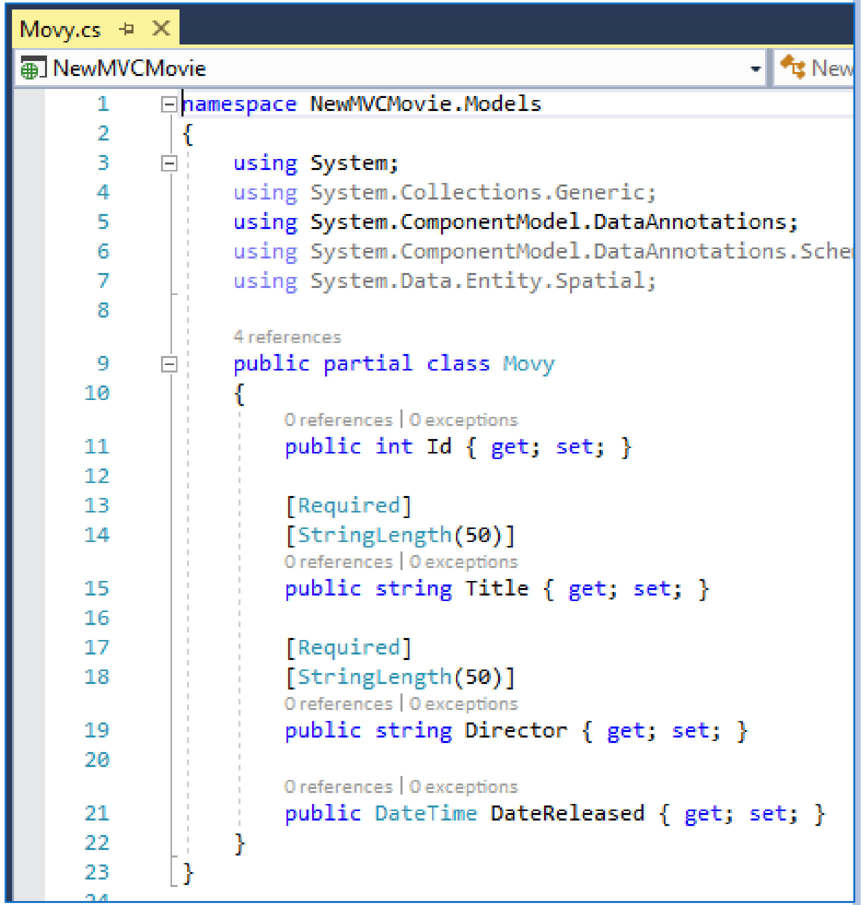
- DbContext – effectively the database connection and a set of tables
- Allows you to link your model properties to your database with a connection string
  - :base(" ") refers to your connection string in web.config



```
namespace NewMVCMovie.Models
{
    using System;
    using System.Data.Entity;
    using System.ComponentModel.DataAnnotations.Schema;
    using System.Linq;

    public partial class DataModel : DbContext
    {
        public DataModel()
            : base("name=DataModel")
        {
```

Closer look at components of the model

- Class that corresponds to fields in database table
  - Take note of *decoration*
  - [Required]
  - [StringLength]



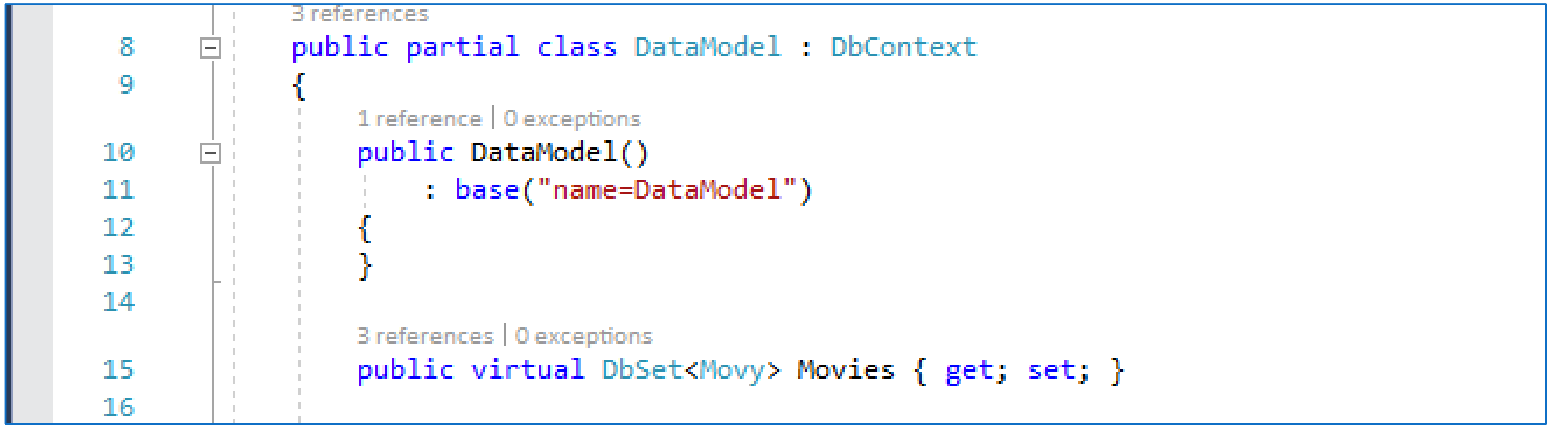
Movy.cs

NewMVCMovie

```
namespace NewMVCMovie.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.ComponentModel.DataAnnotations.Sche
    using System.Data.Entity.Spatial;

    4 references
    public partial class Movy
    {
        0 references | 0 exceptions
        public int Id { get; set; }

        [Required]
        [StringLength(50)]
        0 references | 0 exceptions
        public string Title { get; set; }

        [Required]
        [StringLength(50)]
        0 references | 0 exceptions
        public string Director { get; set; }

        0 references | 0 exceptions
        public DateTime DateReleased { get; set; }
    }
}
```

Closer look at components of the model

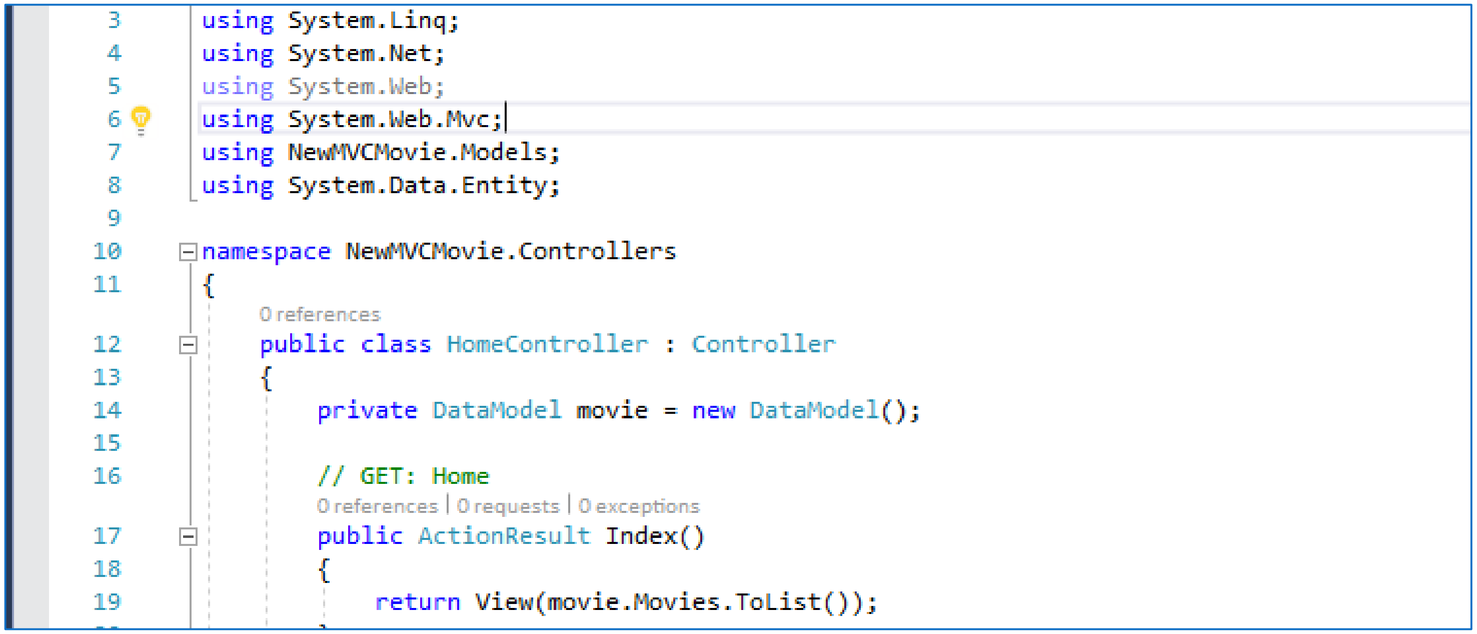
- DbSet – used to query and save instances of the model (class)



```
3 references
public partial class DataModel : DbContext
{
    1 reference | 0 exceptions
    public DataModel()
        : base("name=DataModel")
    {
    }

    3 references | 0 exceptions
    public virtual DbSet<Movy> Movies { get; set; }

```

Closer look at the Controller

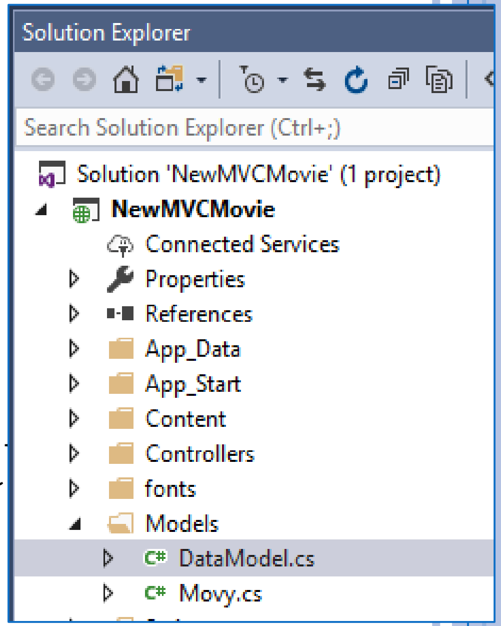
- Create an instance of the DataModel (DBContext) to be able to query, edit and delete records
- A request to the controller returns all the entries in the specific table of the database, and passes the result to the view

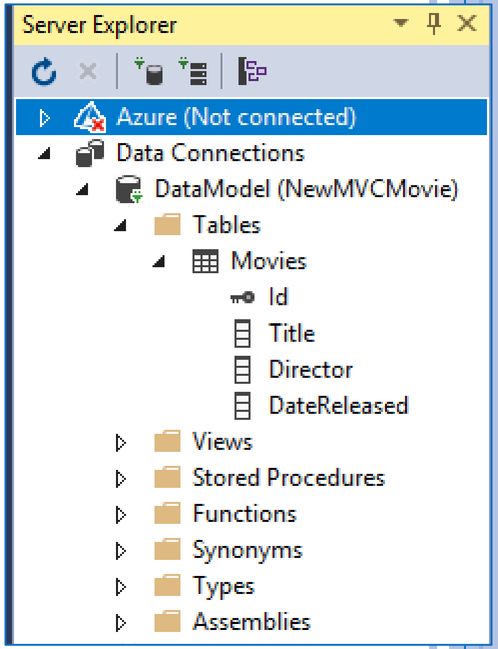


```
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using NewMVCMovie.Models;
using System.Data.Entity;

namespace NewMVCMovie.Controllers
{
    0 references
    public class HomeController : Controller
    {
        private DataModel movie = new DataModel();

        // GET: Home
        0 references | 0 requests | 0 exceptions
        public ActionResult Index()
        {
            return View(movie.Movies.ToList());
```

Working with Database

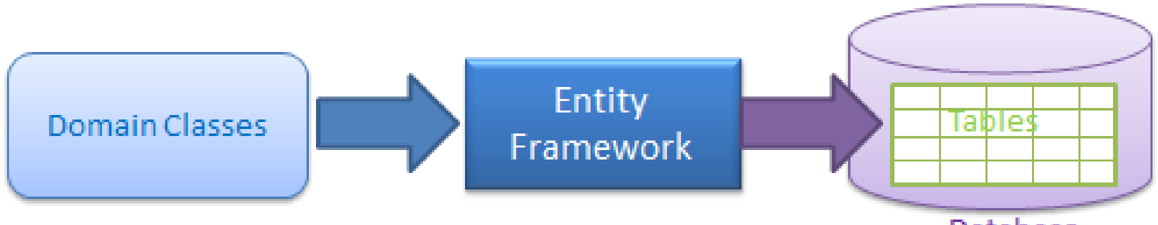
- Must be in App_Data folder
- Can be created from within VS
  - Server Explorer, Add item, SQL Server Data
  - Add table with fields
  - Add records
  - Use wizard to create model (class)
- Use Server Explorer
  - Open tables
  - Open table definition





Working with a Database

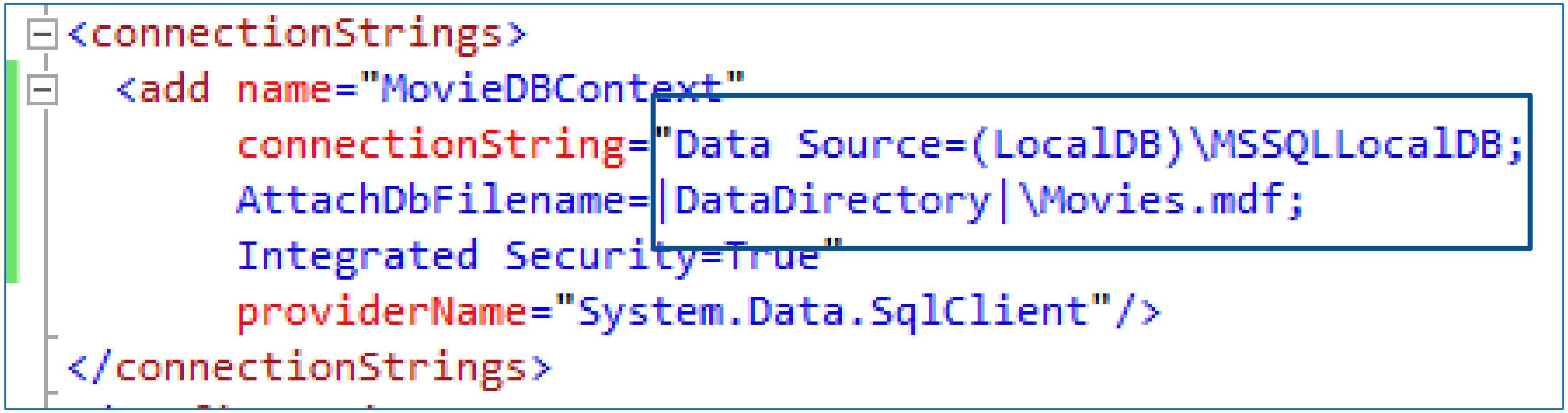
- Entity Framework – Code-First approach
- EF API will create the database based on your classes and configuration
  - Right click Model, Add, New Item, Class
  - Add properties to class
  - Add class to represent the Entity Framework database context – handles fetching, storing and updating of Movie class (MovieDBContext class)
  - Add using Sytem.Data.Entity to .cs file



Code-First Approach

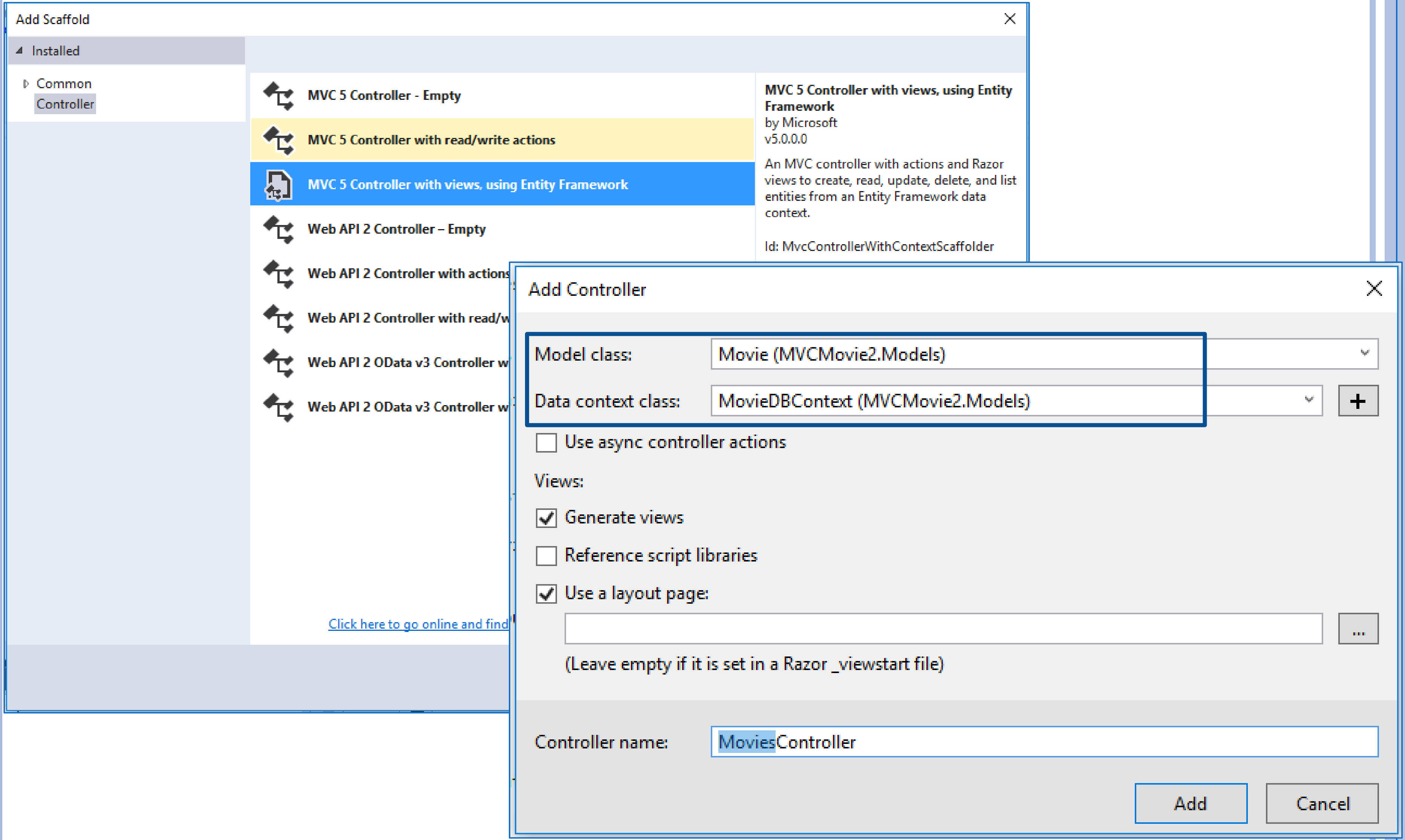
WORKING WITH A DATABASE

- Connection string specified in web.config



```
<connectionStrings>
  <add name="MovieDBContext"
       connectionString="Data Source=(LocalDB)\MSSQLLocalDB;
       AttachDbFilename=|DataDirectory|\Movies.mdf;
       Integrated Security=True"
       providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Controller to access model's data



Add Scaffold

Installed

Common

Controller

MVC 5 Controller - Empty

MVC 5 Controller with read/write actions

MVC 5 Controller with views, using Entity Framework

Web API 2 Controller – Empty

Web API 2 Controller with actions

Web API 2 Controller with read/w

Web API 2 OData v3 Controller w

Web API 2 OData v3 Controller w

Click here to go online and find

MVC 5 Controller with views, using Entity Framework
by Microsoft
v5.0.0.0

An MVC controller with actions and Razor views to create, read, update, delete, and list entities from an Entity Framework data context.

Id: MvcControllerWithContextScaffolder

Add Controller

Model class: Movie (MVCMovie2.Models)

Data context class: MovieDBContext (MVCMovie2.Models)

☐ Use async controller actions

Views:

☑ Generate views

☐ Reference script libraries

☑ Use a layout page:

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

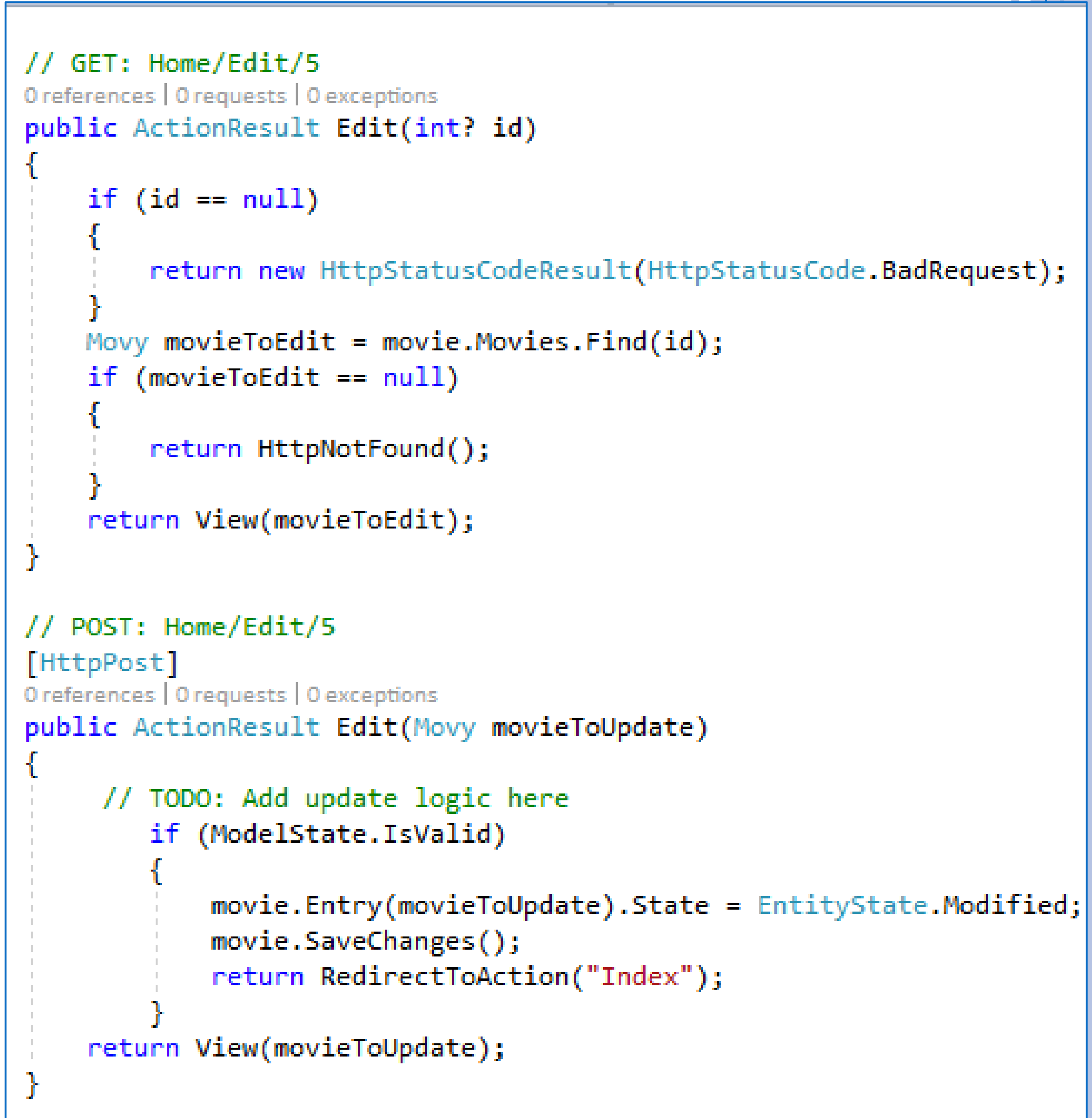
Controller name: MoviesController

HTTP GET AND HTTP POST

- Getting Data from View to Controller
  - E.g. Edit record
- HTTP GET
  - When action method is called by a request URL (by browser)
  - E.g. Display the details of the record that was selected
- HTTP POST
  - When action method is called by something like button click event
  - E.g. Save the updated information back to the database
  - [HttpPost] required to specify that the second method can only be invoked for POST requests

A closer look at the Edit method

- Get:
  - Find the relevant record in the table
  - Display view with that record
  - Some error checking
- Post:
  - Could specify specific fields in Bind attribute
  - If data is valid, save to table
  - Return to Index view
  - Some error checking

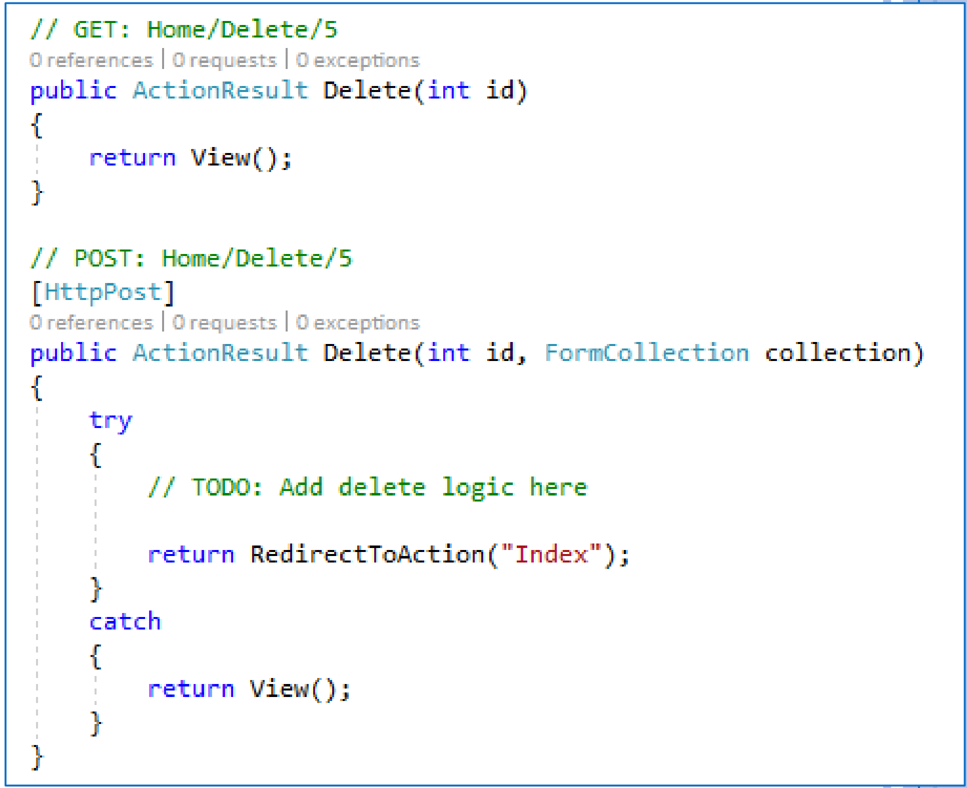


```
// GET: Home/Edit/5
0 references | 0 requests | 0 exceptions
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Movy movieToEdit = movie.Movies.Find(id);
    if (movieToEdit == null)
    {
        return HttpNotFound();
    }
    return View(movieToEdit);
}

// POST: Home/Edit/5
[HttpPost]
0 references | 0 requests | 0 exceptions
public ActionResult Edit(Movy movieToUpdate)
{
     // TODO: Add update logic here
        if (ModelState.IsValid)
        {
            movie.Entry(movieToUpdate).State = EntityState.Modified;
            movie.SaveChanges();
            return RedirectToAction("Index");
        }
    return View(movieToUpdate);
}
```

Examining the the Delete Method

- Get:
  - Find the relevant record in the table
  - Display view with that record
    - Button to confirm delete
  - Some error checking

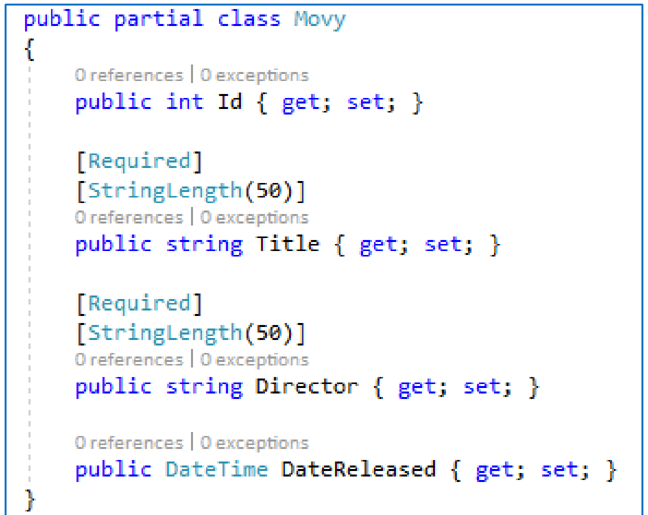
- Post:
  - Find the relevant record in the table
  - Use the Remove method
  - Save the changes
  - Some error checking



```
// GET: Home/Delete/5
0 references | 0 requests | 0 exceptions
public ActionResult Delete(int id)
{
    return View();
}

// POST: Home/Delete/5
[HttpPost]
0 references | 0 requests | 0 exceptions
public ActionResult Delete(int id, FormCollection collection)
{
    try
    {
        // TODO: Add delete logic here

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

Validation

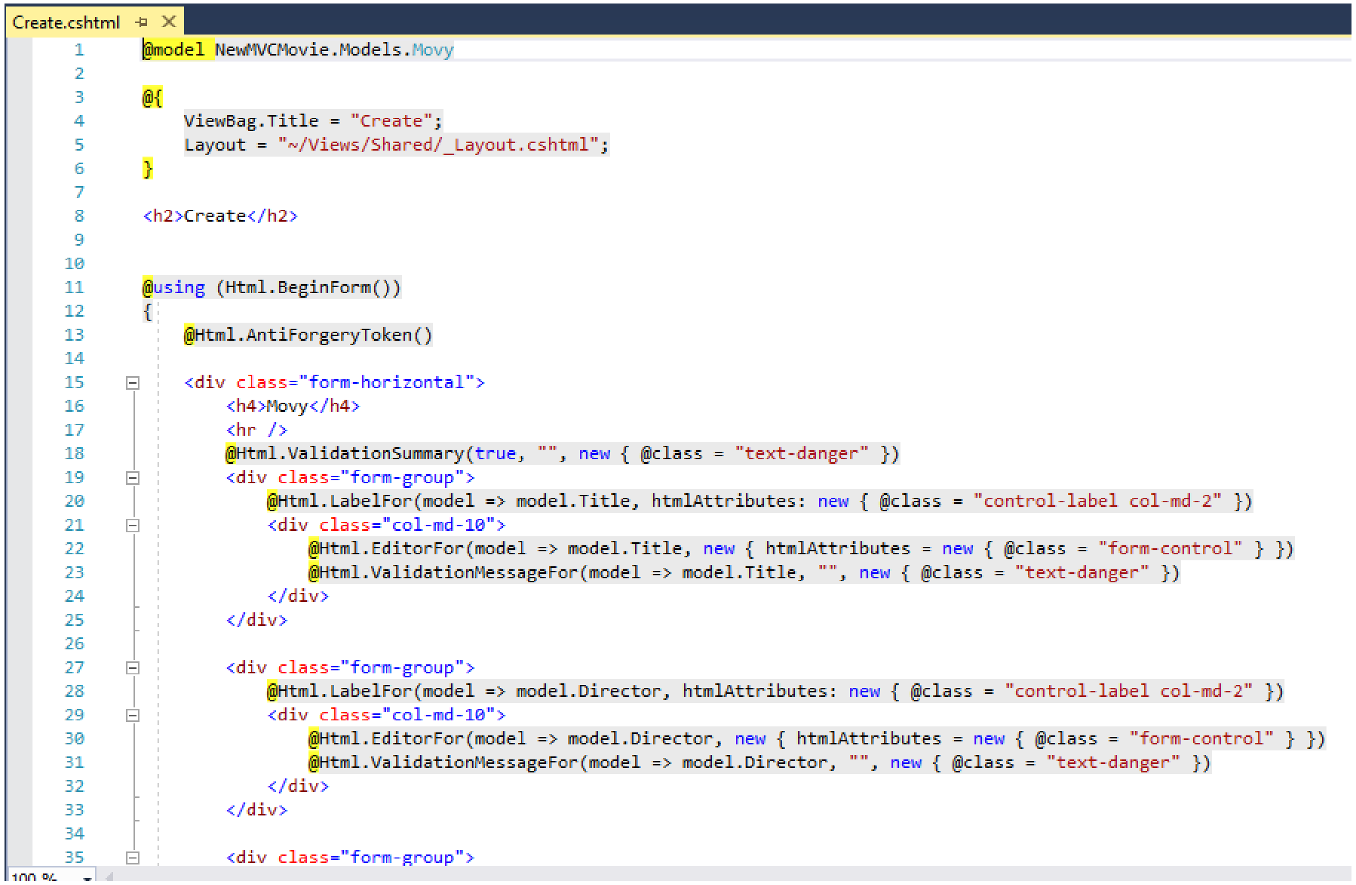
- If you created the database first, and specified “Not Null”
  - [Required] would be visible in model
- Can edit model and update attributes
  - [Required]
  - [StringLength]
  - [Regular Expression(@"^[A-Z]+[a-zA-Z'\s]*$")]



```
public partial class Movy
{
    0 references | 0 exceptions
    public int Id { get; set; }

    [Required]
    [StringLength(50)]
    0 references | 0 exceptions
    public string Title { get; set; }

    [Required]
    [StringLength(50)]
    0 references | 0 exceptions
    public string Director { get; set; }

    0 references | 0 exceptions
    public DateTime DateReleased { get; set; }
}
```

Validation

- Look for corresponding code in View
  - E.g. relevant when creating new records


Create.cshtml

```
@model NewMVCMovie.Models.Movy

@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Movy</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Title, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Title, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Title, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Director, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Director, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Director, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
```

100 %

Authentication

- Next week….

Portfolio practical

- Due 16 October 12:00 midday
  - Submitted on server (details to follow)
- See Portfolio project document
  - 3 topics to choose from
  - Must use the ASP.NET MVC framework
  - Marking rubric provided

Comprehensive Resource

- Getting Started with ASP.NET MVC 5
  - https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/

Other Resources

- Entity Framework:
  - http://www.entityframeworktutorial.net/entityframework6/introduction.aspx
- Examining the Edit methods and Edit View
  - https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/examining-the-edit-methods-and-edit-view
- HTTP GET and HTTP POST
  - https://www.c-sharpcorner.com/UploadFile/3d39b4/getting-data-from-view-to-controller-in-mvc/
- FormCollection
  - https://www.c-sharpcorner.com/UploadFile/dacca2/understand-formcollection-in-mvc-controller/