Web Applications

Dominic Duggan
Stevens Institute of Technology

1

WEB FORMS

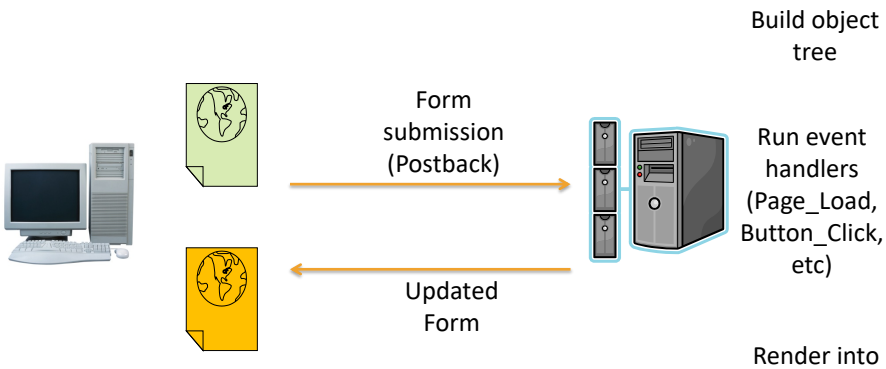
2

ASP.NET Web Forms

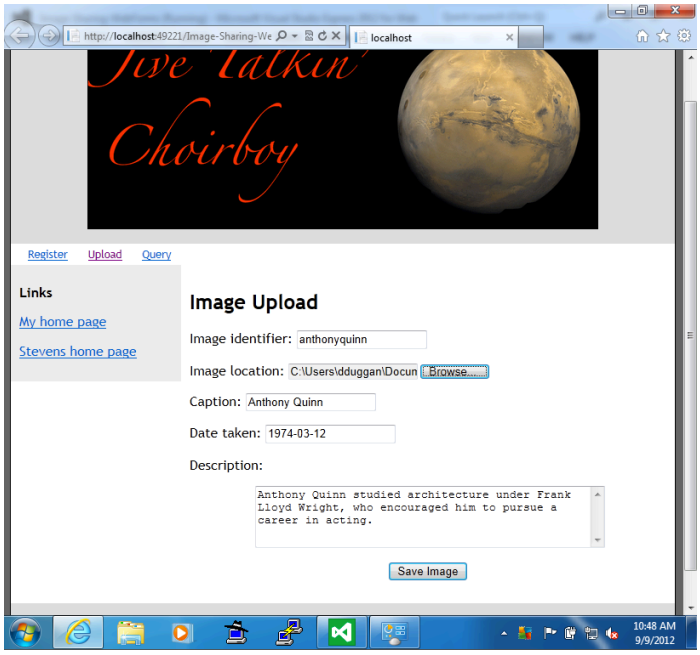
- Web interaction as GUI interaction
- Stateful GUI
- Event handlers
- Problem: Web is stateless
- Web forms: virtual GUI state

3

ASP.NET Web Forms



4


Jive Talkin'
Choirboy
Register
Upload
Query
Links
My home page
Stevens home page
Image Upload
Image identifier: anthonyquinn
Image location: C:\Users\dduggan\Docun
Browse...
Caption: Anthony Quinn
Date taken: 1974-03-12
Description:
Anthony Quinn studied architecture under Frank Lloyd Wright, who encouraged him to pursue a career in acting.
Save Image
10:48 AM
9/9/2012

5

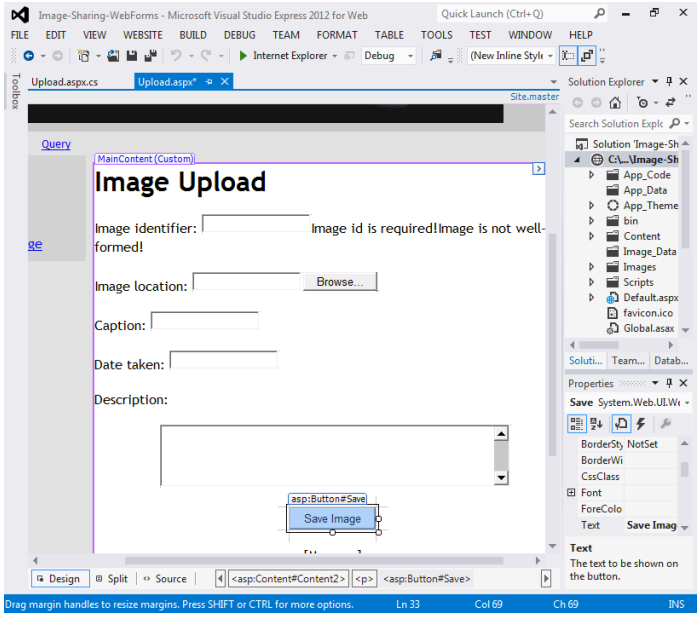
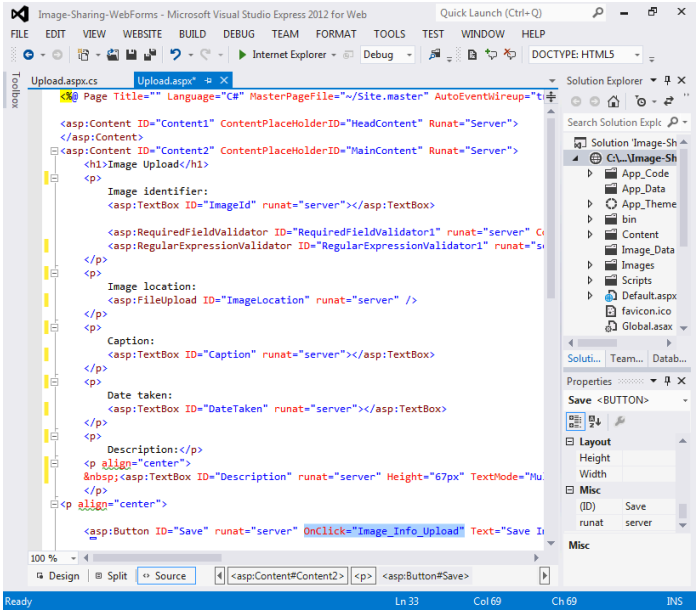


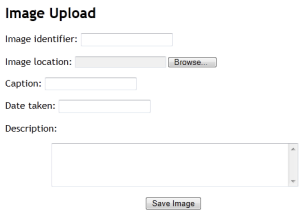
Image-Sharing-WebForms - Microsoft Visual Studio Express 2012 for Web
Upload.aspx.cs
Upload.aspx*
Site.master
Query
MainContent (Custom)
Image Upload
Image identifier:
Image id is required!Image is not well-formed!
Image location:
Browse...
Caption:
Date taken:
Description:
asp:Button#Save
Save Image
Solution Explorer
Properties
Text
The text to be shown on the button.
Drag margin handles to resize margins. Press SHIFT or CTRL for more options.

6



7

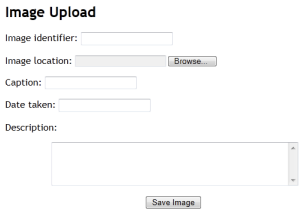
ASP.NET Markup




```
<h1>Image Upload</h1>
<p> Image location:
   <asp:FileUpload ID="ImageLocation" runat="server" />
   </p>
<p> Caption:
   <asp:TextBox ID="Caption" runat="server" />
   </p>

<asp:Button ID="Save" runat="server"
    OnClick="Image_Info_Upload"
    Text="Save Image" />

<asp:Label ID="Message" runat="server" />
```

8

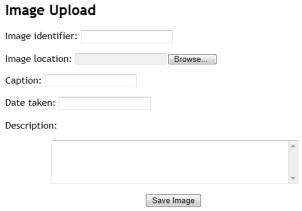
ASP.NET Markup




```
<h1>Image Upload</h1>
<p> Image location:
   <asp:FileUpload ID="ImageLocation" runat="server" />
   </p>
<p> Caption:
   <asp:TextBox ID="Caption" runat="server” />
   </p>
...
<asp:Button ID="Save" runat="server"
    OnClick="Image_Info_Upload"
    Text="Save Image" />

<asp:Label ID="Message" runat="server” />
```

9

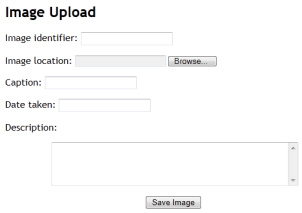
Rendered HTML




```
<p> Image location:
<input type="file"
   name="ctl00$MainContent$ImageLocation"
   id="MainContent_ImageLocation" /> </p>
<p> Caption:
<input type="text" name="ctl00$MainContent$Caption"
   id="MainContent_Caption" /></p>
...
<input type="submit" name="ctl00$MainContent$Save"
   value="Save Image"
   onclick=
        "javascript:WebForm_DoPostBackWithOptions(...)"
   id="MainContent_Save" />
```

10

ASP.NET Markup




```
<h1>Image Upload</h1>
<p> Image location:
   <asp:FileUpload ID="ImageLocation" runat="server" />
   </p>
<p> Caption:
   <asp:TextBox ID="Caption" runat="server" />
   </p>

<asp:Button ID="Save" runat="server"
    OnClick="Image_Info_Upload"
    Text="Save Image" />

<asp:Label ID="Message" runat="server" />
```

11

Code Behind

```
public partial class Upload : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Image_Info_Upload(object sender, EventArgs e)
    {
        ImageData data = new ImageData();
        data.caption = Caption.Text;
        ...
        // Save the image data on the server.

        // Updated form by default remembers previous inputs in fields.
        Message.Text = "Image information successfully uploaded!";
        Caption.Text = "";
        ...
    }
}
```

12

Viewstate in Postback Form

```
<form name="form1" method="post" action="Default.aspx"
                id="form1">
  <div>
    <input type="hidden" name="__VIEWSTATE"
           id="__VIEWSTATE" value="viewStateValue" />
  </div>

  ... Rendered output of inner Web controls ...

</form>
```

The page's *view state* is encoded and stored in a hidden `<input>` field.

13

ASP.NET Web Forms Summary

- Pattern: Page controller
- Similar to GUI programming
  - Wysiwyg editor
  - Event handling
  - Postbacks
- Problems
  - Unit testing
  - Event handler interactions
  - Viewstate
  - Control over rendered HTML

14

ASP.NET MVC

15

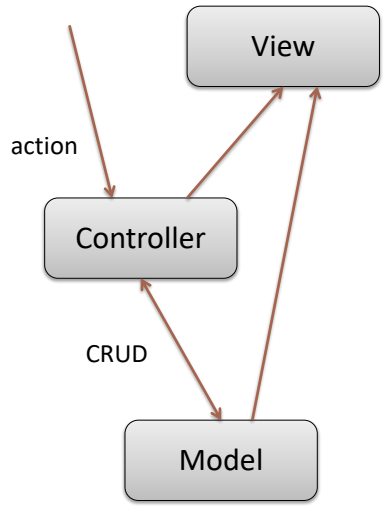
ASP.NET MVC

- Model-View-Controller
- Motivation: Separation of Concerns
- Benefits over Web Forms
  - Program control of app behavior
  - Test-driven development (TDD)
  - Routing
- Disadvantages
  - Relative (app) complexity

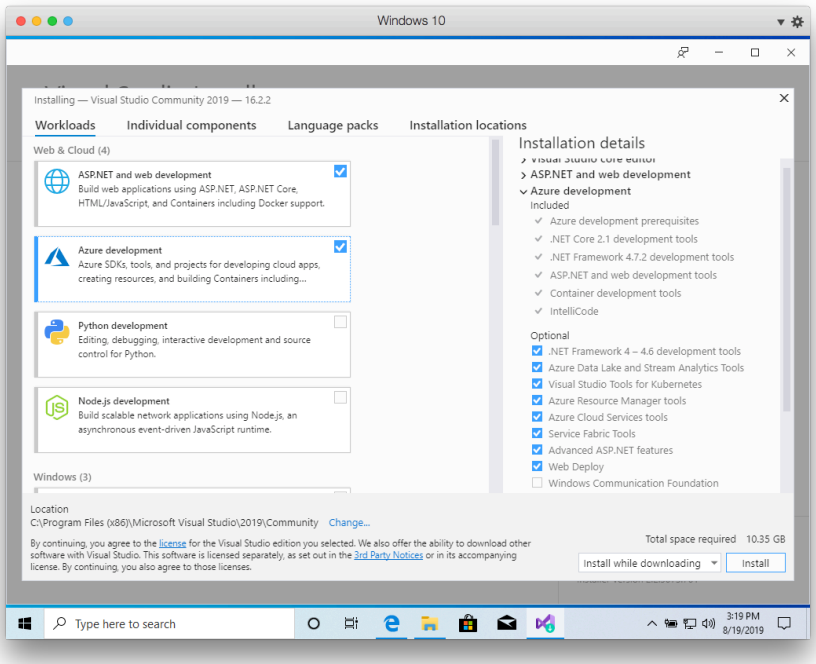
16

Model-View-Controller

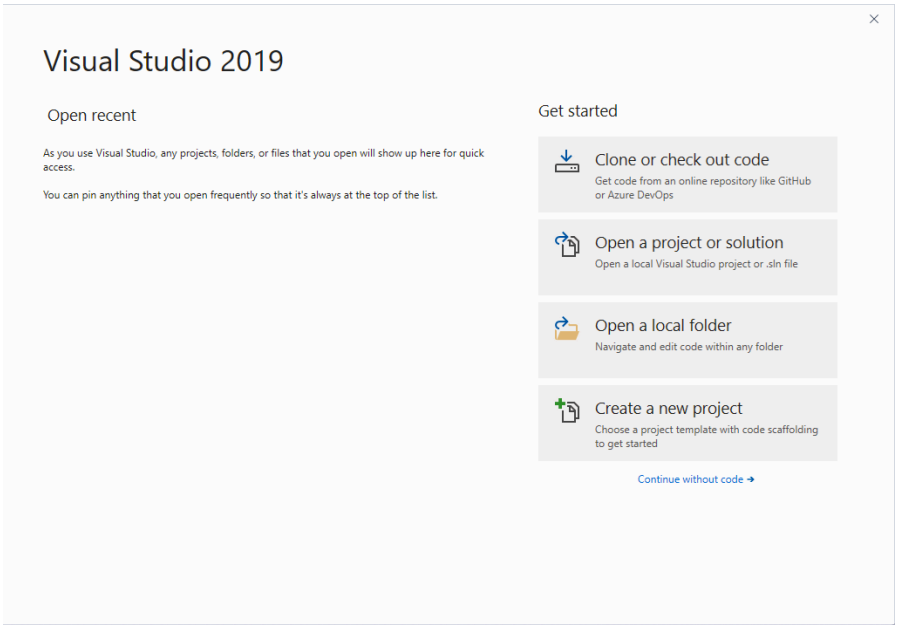
- Controller: business logic
- Model: application state
  - Database
- View: presentation logic
  - HTTP response



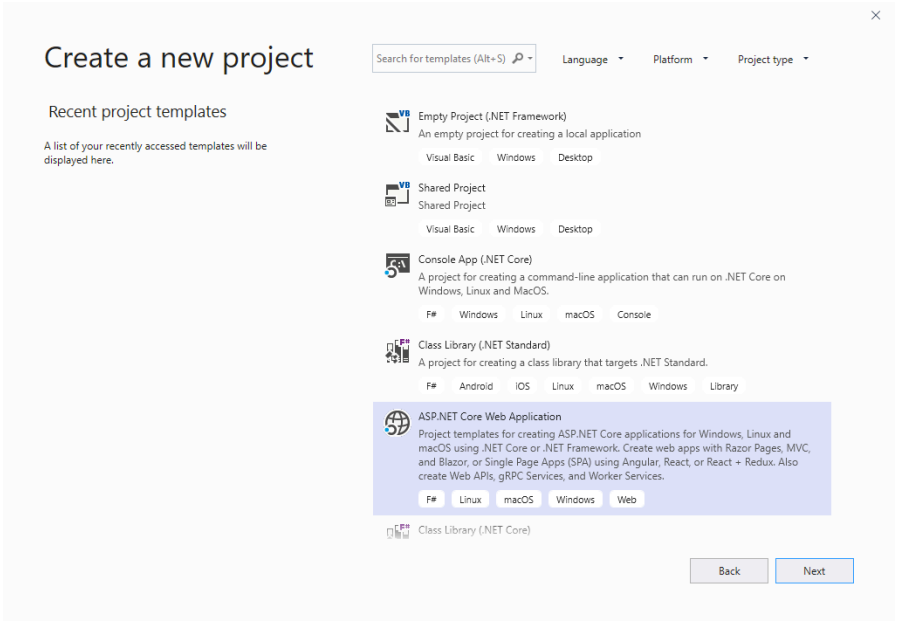
17



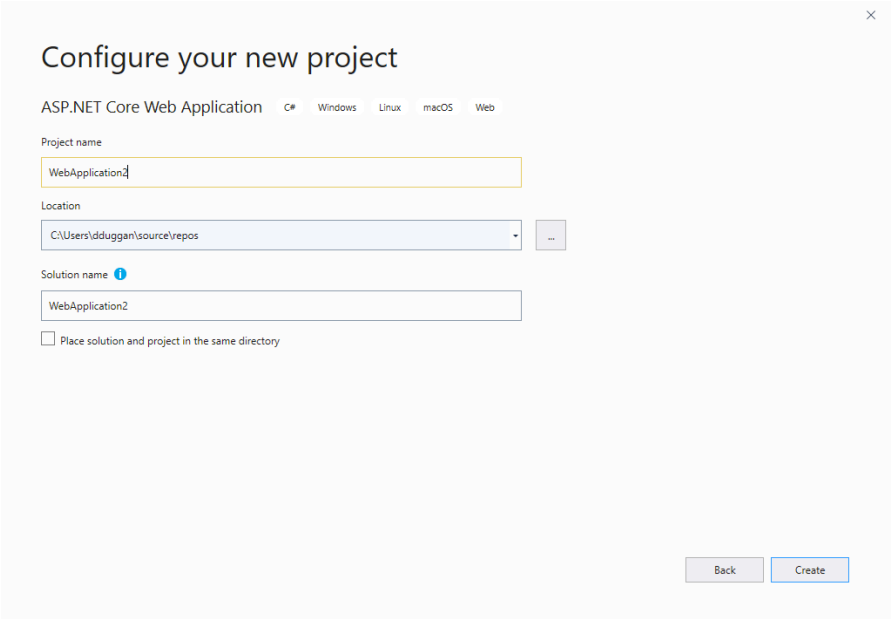
18


Visual Studio 2019
Open recent
As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.
You can pin anything that you open frequently so that it's always at the top of the list.
Get started
Clone or check out code
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

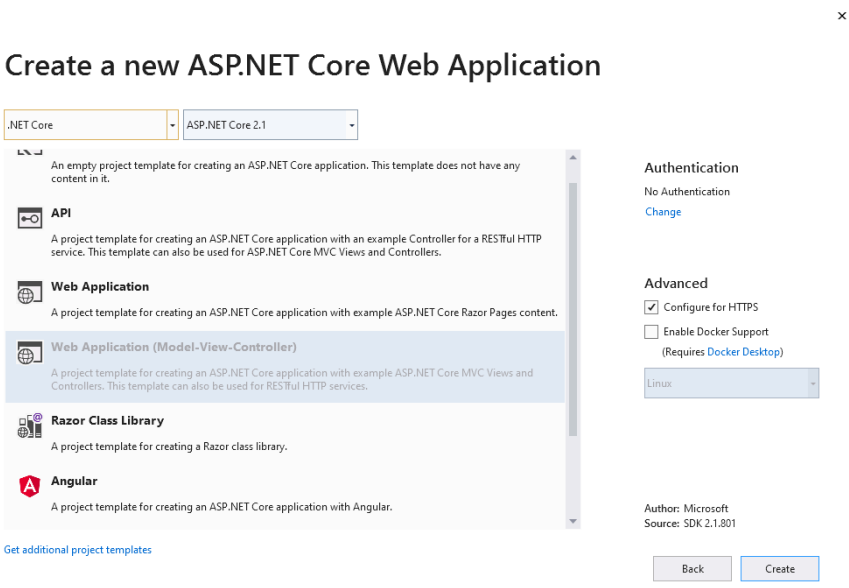
19


Create a new project
Search for templates (Alt+S)
Language
Platform
Project type
Recent project templates
A list of your recently accessed templates will be displayed here.
Empty Project (.NET Framework)
An empty project for creating a local application
Visual Basic Windows Desktop
Shared Project
Shared Project
Visual Basic Windows Desktop
Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
F# Windows Linux macOS Console
Class Library (.NET Standard)
A project for creating a class library that targets .NET Standard.
F# Android iOS Linux macOS Windows Library
ASP.NET Core Web Application
Project templates for creating ASP.NET Core applications for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, and Blazor, or Single Page Apps (SPA) using Angular, React, or React + Redux. Also create Web APIs, gRPC Services, and Worker Services.
F# Linux macOS Windows Web
Class Library (.NET Core)
Back
Next

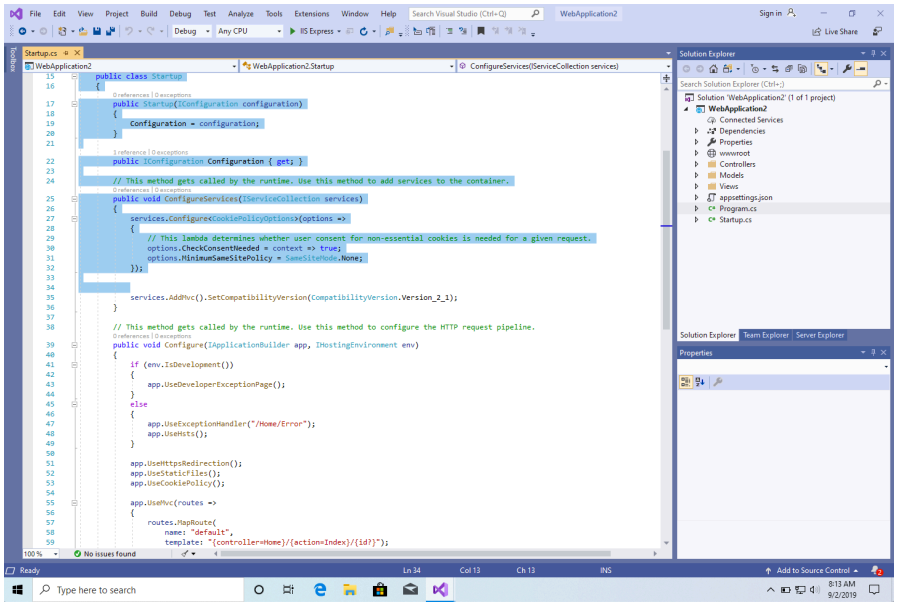
20


Configure your new project
ASP.NET Core Web Application
C#
Windows
Linux
macOS
Web
Project name
WebApplication2
Location
C:\Users\dduggan\source\repos
...
Solution name
WebApplication2
Place solution and project in the same directory
Back
Create

21


Create a new ASP.NET Core Web Application
.NET Core
ASP.NET Core 2.1
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Core Razor Pages content.
Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
Razor Class Library
A project template for creating a Razor class library.
Angular
A project template for creating an ASP.NET Core application with Angular.
Authentication
No Authentication
Change
Advanced
Configure for HTTPS
Enable Docker Support
(Requires Docker Desktop)
Linux
Author: Microsoft
Source: SDK 2.1.801
Get additional project templates
Back
Create

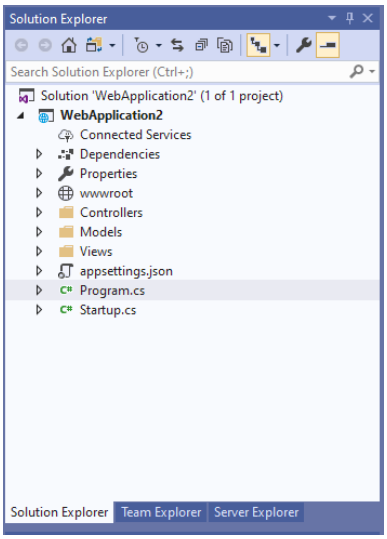
22



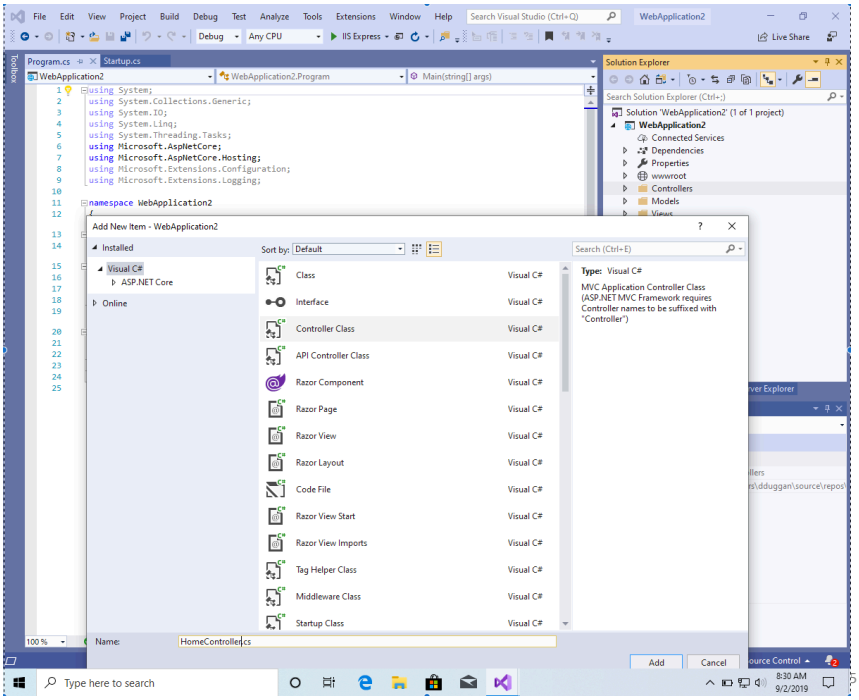
23

Structure of a Core MVC Project

- wwwroot
- appsettings.json
- Program.cs
- Startup.cs
- Controllers
- Models
- Views



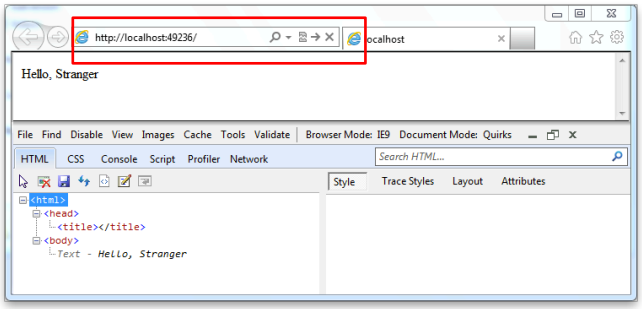
24

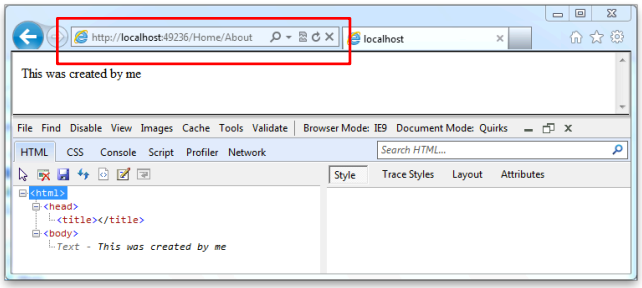


Sample Controller

```
namespace HelloWorld.Controllers
{
    public class HomeController : Controller
    {
        public String Index()
        {
            return “Hello, Stranger”;
        }

        public String About()
        {
            return “This was created by me.”;
        }
    }
}
```

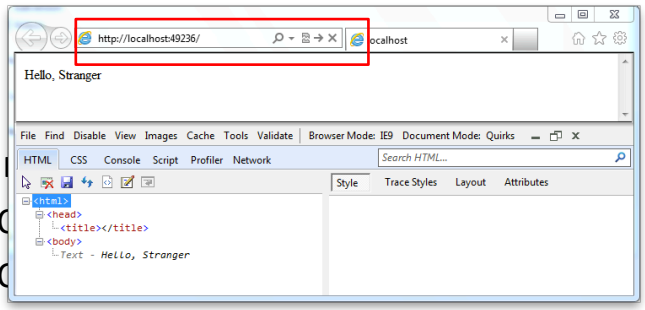
26





27

- Na
  - C
  - C



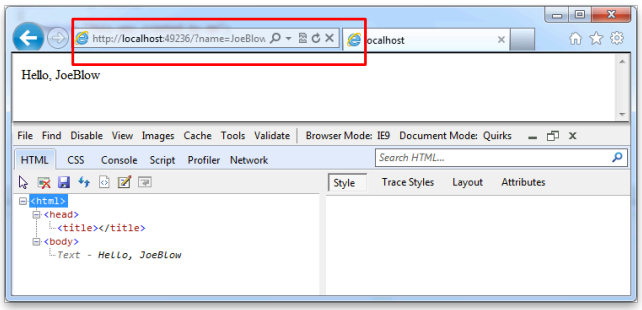
- Routing Convention
  - URL: `http://host:port/control-id/action-id`
  - Ex: `http://localhost:49236/Home/About`
  - Default: `http://localhost:49236/Home/Index`

28

Sample Controller

```
namespace HelloWorld.Controllers
{
    public class HomeController : Controller
    {
        public String Index(String name="Stranger")
        {
            return "Hello, "+name;
        }

        public String About()
        {
            return "This was created by me.";
        }
    }
}
```

29



Danger, Will Robinson!

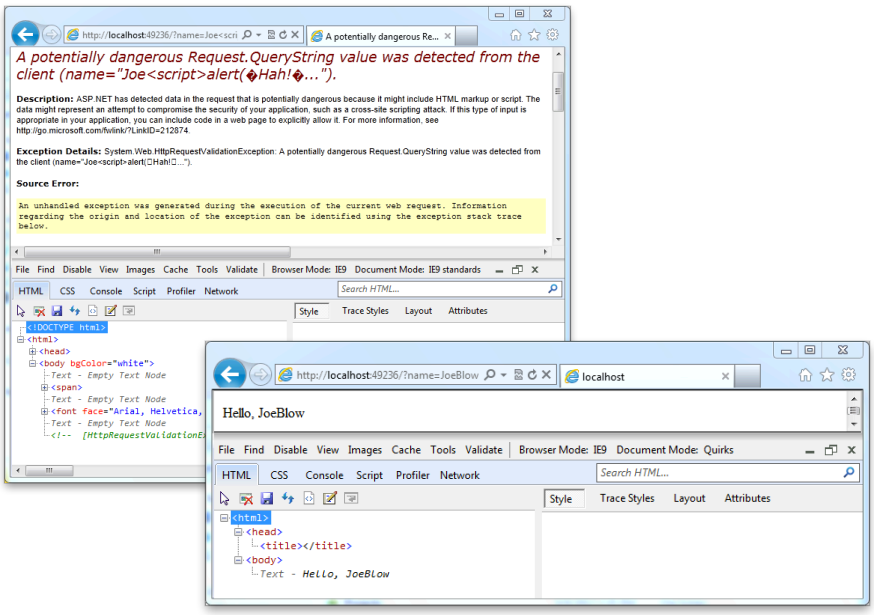
http://localhost:49236/?name=Joe<script>alert("Hah!");</script>

30

Sample Controller

```
namespace HelloWorld.Controllers
{
    public class HomeController : Controller
    {
        public String Index(String name="Stranger")
        {
            return HttpUtility.HtmlEncode("Hello, "+name);
        }

        public String About()
        {
            return HttpUtility.HtmlEncode("...");
        }
    }
}
```

31



32

VIEWS

33

Presentation Logic

- Programmatic definition
  - e.g. `HttpUtility.HtmlEncode()`
- Declarative: ASPX markup (*`view`*`.aspx`)
  - XML
  - Designed for Web Forms
- Combined: Razor (*`view`*`.cshtml`)
  - Not XML
  - Integrated with C#

34

Sample Controller

```
namespace HelloWorld.Controllers
{
    public class HomeController : Controller
    {
        public String Index()
        {
            return “Hello, Stranger”;
        }

        public String About()
        {
            return “This was created by me.”;
        }
    }
}
```

35

Views for Results

```
namespace HelloWorld.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

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

36

ViewData

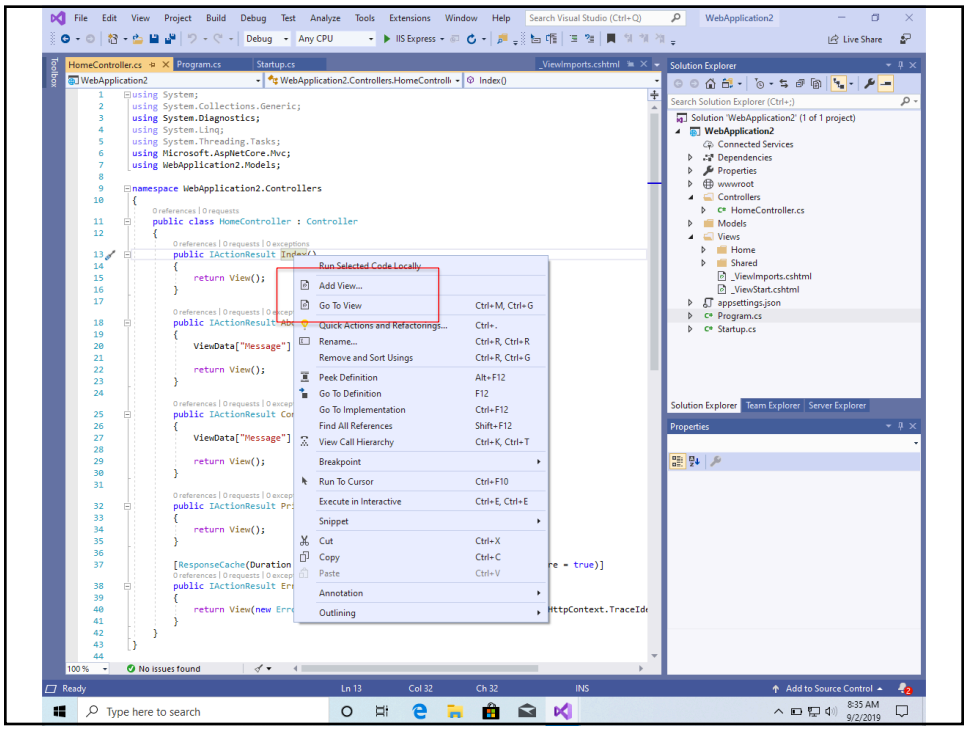
```
namespace HelloWorld.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index(String name="Stranger")
        {
            ViewData["name"] = name;
            return View();
        }

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

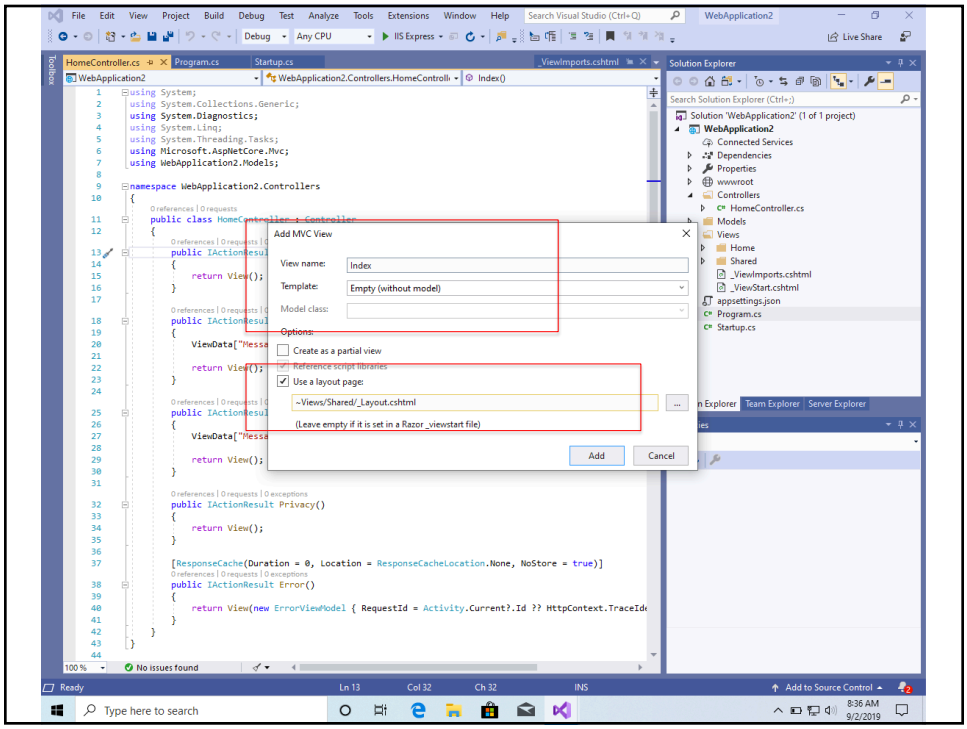
37

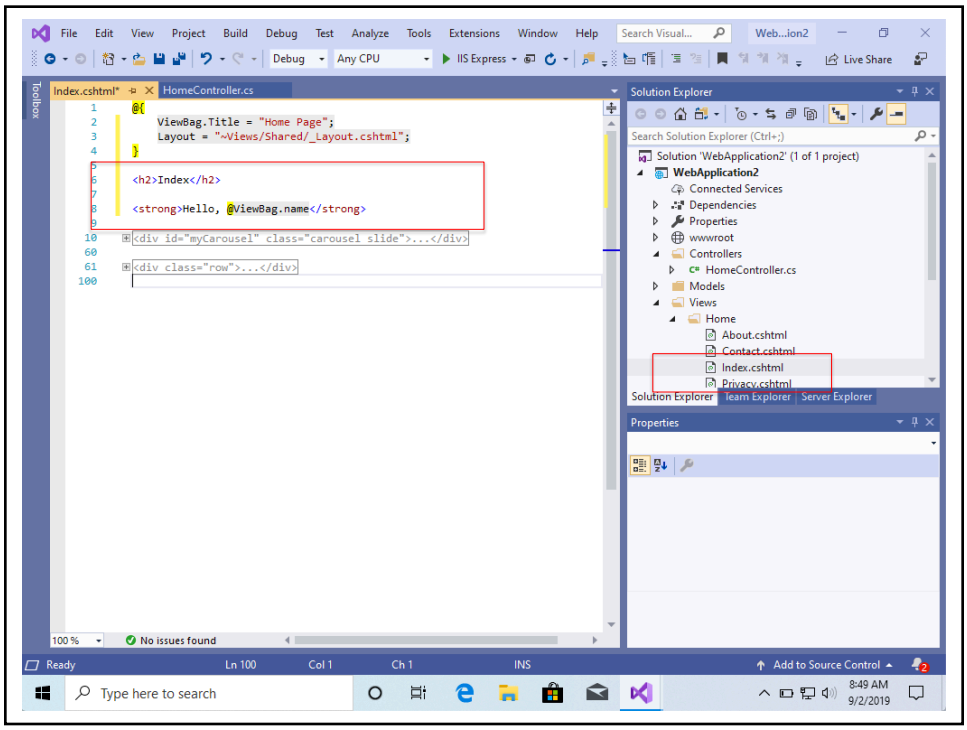
ViewBag

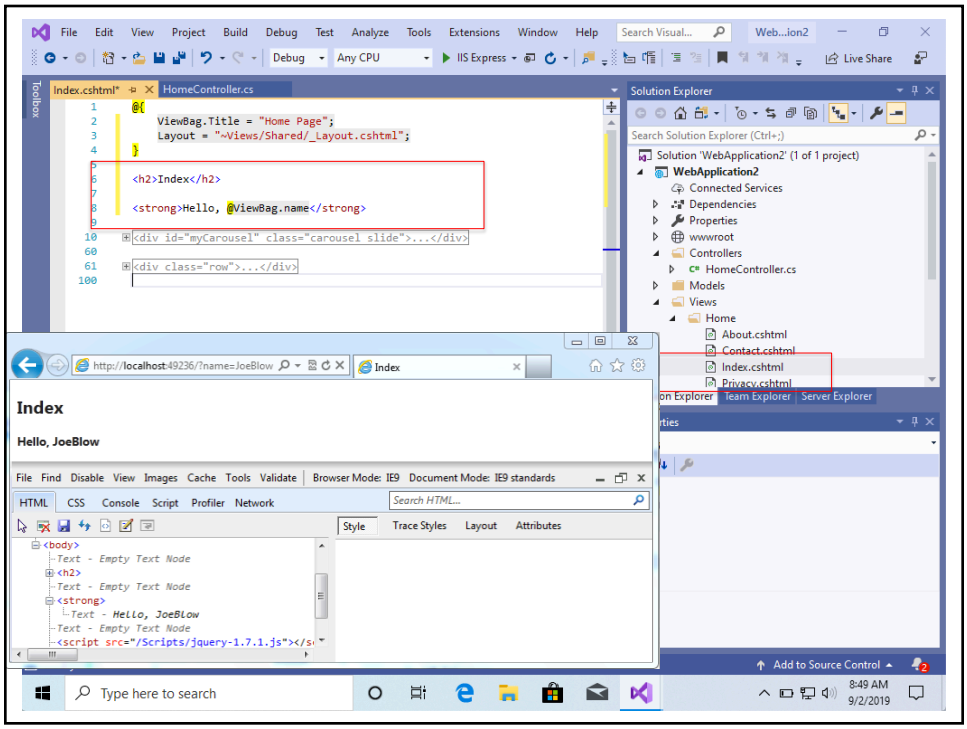
```
namespace HelloWorld.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index(String name="Stranger")
        {
            ViewBag.name = name;
            return View();
        }

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

38


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
WebApplication2
Debug
Any CPU
IIS Express
Live Share
HomeController.cs
Program.cs
Startup.cs
_ViewImports.cshtml
WebApplication2
WebApplication2.Controllers.HomeControll
Index()
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication2.Models;
namespace WebApplication2.Controllers
public class HomeController : Controller
public IActionResult Index()
return View();
ViewData["Message"]
[ResponseCache(Duration
public IActionResult Privacy()
public IActionResult Error()
Run Selected Code Locally
Add View...
Go To View Ctrl+M, Ctrl+G
Quick Actions and Refactorings... Ctrl+.
Rename... Ctrl+R, Ctrl+R
Remove and Sort Usings Ctrl+R, Ctrl+G
Peek Definition Alt+F12
Go To Definition F12
Go To Implementation Ctrl+F12
Find All References Shift+F12
View Call Hierarchy Ctrl+K, Ctrl+T
Breakpoint
Run To Cursor Ctrl+F10
Execute in Interactive Ctrl+E, Ctrl+E
Snippet
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Annotation
Outlining
Solution Explorer
Solution 'WebApplication2' (1 of 1 project)
WebApplication2
Connected Services
Dependencies
Properties
wwwroot
Controllers
HomeController.cs
Models
Views
Home
Shared
_ViewImports.cshtml
_ViewStart.cshtml
appsettings.json
Program.cs
Startup.cs
Solution Explorer
Team Explorer
Server Explorer
Properties
No issues found
Ready
Ln 13
Col 32
Ch 32
INS
Add to Source Control
Type here to search
8:35 AM
9/2/2019

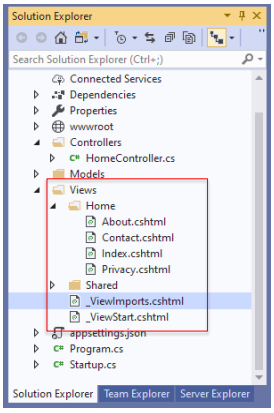

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
WebApplication2
Debug
Any CPU
IIS Express
Live Share
HomeController.cs
Program.cs
Startup.cs
_ViewImports.cshtml
WebApplication2
WebApplication2.Controllers.HomeControll
Index()
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication2.Models;
namespace WebApplication2.Controllers
public class HomeController : Controller
public IActionResult Privacy()
public IActionResult Error()
Add MVC View
View name: Index
Template: Empty (without model)
Model class:
Options:
Create as a partial view
Reference script libraries
Use a layout page:
~/Views/Shared/_Layout.cshtml
(Leave empty if it is set in a Razor _viewstart file)
Add
Cancel
Solution Explorer
Solution 'WebApplication2' (1 of 1 project)
WebApplication2
Connected Services
Dependencies
Properties
wwwroot
Controllers
HomeController.cs
Models
Views
Home
Shared
_ViewImports.cshtml
_ViewStart.cshtml
appsettings.json
Program.cs
Startup.cs
Team Explorer
Server Explorer
No issues found
Ready
Ln 13
Col 32
Ch 32
INS
Add to Source Control
Type here to search
8:36 AM
9/2/2019


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Debug
Any CPU
IIS Express
Live Share
Index.cshtml*
HomeController.cs
@{
ViewBag.Title = "Home Page";
Layout = "~Views/Shared/_Layout.cshtml";
}
<h2>Index</h2>
Hello, @ViewBag.name
...
...
Solution Explorer
Solution 'WebApplication2' (1 of 1 project)
WebApplication2
Connected Services
Dependencies
Properties
wwwroot
Controllers
HomeController.cs
Models
Views
Home
About.cshtml
Contact.cshtml
Index.cshtml
Privacy.cshtml
Solution Explorer
Team Explorer
Server Explorer
Properties
No issues found
Ready
Ln 100
Col 1
Ch 1
INS
Add to Source Control
Type here to search
8:49 AM
9/2/2019


http://localhost:49236/?name=JoeBlow
Index
Index
Hello, JoeBlow
File Find Disable View Images Cache Tools Validate Browser Mode: IE9 Document Mode: IE9 standards
HTML CSS Console Script Profiler Network
Style Trace Styles Layout Attributes
<body>
Text - Empty Text Node
<h2>
Text - Empty Text Node

Text - Hello, JoeBlow
Text - Empty Text Node
<script src="/Scripts/jquery-1.7.1.js"></s
Add to Source Control
Type here to search
8:49 AM
9/2/2019

Naming Views



- Default view

```
public ActionResult Index(..
{
    ViewBag.name = name;
    return View();
}
```

- Named view

```
return View("Index2");

return View("~/Views/Example/Index.shtml");
```

43

VIEWS AND MODELS

44

Image Model

- Model for photographic image

```
public class Image {
   public String Id { get; set; };
   public int TagId { get; set; };
   public String Caption { get; set; };
   public String Description { get; set; };
   public Date DateTaken { get; set; };
   public String UserId { get; set; };
}
```

45

Dynamically Typed Views

- Controller Action

```
public ActionResult Index(...)
{
   List<Image> images;
   ...
   ViewBag.images = images;
   return View();
}
```

- View

```
IEnumerable<Image> images =
       ViewBag.images as IEnumerable<Image>;
foreach (Image img in images)
  <li> @img.Caption </li>
```

46

Statically Typed Views

- Controller Action

```
public ActionResult Index(...)
{
   List<Image> images;
   ...
   return View(images);
}
```

- View

```
@model IEnumerable<ImageSharing.Models.Image>
...
foreach (Image img in Model)
  <li> @img.Caption </li>
```

47

RAZOR

48

Razor

- Presentation language for ASP.NET MVC
- HTML Markup
  - Delimited by tags
- C# code fragments
  - Delimited by '@'

49

Code in Presentation

- Web Forms View

```
<title> <%: Page.Title %> </title>
```

- Razor

```
<title> @ViewBag.Title </title>
```

50

Code in Presentation

- Web Forms View

```
<title> <%: Page.Title %> </title>
```

- Razor

```
@{
    var title = ViewBag.Title;
}
<title> @title </title>
```

51

Code in Presentation

- Shifting between code and literal

```
@{
  IEnumerable<Image> images =
     ViewBag.images as IEnumerable<Image>;
  foreach (Image img in images)
     <li>
        @img.Caption
     </li>
}
```

52

Code in Presentation

- Shifting between code and literal

```
@foreach (Image img in
      (ViewBag.images as IEnumerable<Image>))
      <li>
         @img.Caption
      </li>
```

53

Edge Cases

```
@{ String root = "MyApp"; }
```

```
<span> @root.Models </span>
```

Output: Error (`root.Models` undefined)

```
<span> @(root).Models </span>
```

Output: `MyApp.Models`

`@@foobar` evaluates to "@foobar"

54

HTML Encoding

```
@{ var mesg =
   “<script>alert(‘Hah!’);</script>”
}
<span> @mesg </span>
```

outputs

```
<span> &lt;script&gt;...&gt; </span>
```

- HtmlString versus String
- Html.Raw(model.Message)

55

Mixing code and text

```
@if (showMessage) {
   This is plain text.
}

@if (showMessage) {
   <text> This is plain text. </text>
}

@if (showMessage) {
   @: This is plain text.
}
```

56

Comments

```
@{
   var title = ViewBag.title;
}

@*
   This is a comment.
   @if (showMessage)
      <h1>
         @title
      </h1>
*@
```

57

LAYOUTS

58

Layouts

- Purpose: Templates for views
  - Main body
  - Sections
- Consistent look-and-feel
  - ADA
- Maintenance

59

Layout

```
<!DOCTYPE html>
<html>
<head>
  <title> @ViewBag.Title </title>
</head>
<body>
  <h1> @ViewBag.Title </h1>
  <div id="main">
    @RenderBody()
  </div>
</body>
```

60

Instance

```
@{
  Layout = "~/Views/Shared/Layout.cshtml";
  ViewBag.Title = "The Index!";
}
<p> This is the main content! </p>
```

61

Response Document

```
<!DOCTYPE html>
<html>
<head>
  <title> The Index! </title>
</head>
<body>
  <h1> The Index! </h1>
  <div id="main">
    <p>This is the main content!</p>
  </div>
</body>
```

62

Layout

```
<!DOCTYPE html>
<html>
<head>
  <title> @ViewBag.Title </title>
</head>
<body>
  <h1> @ViewBag.Title </h1>
  <div id=“foo”>@RenderSection(“Foobar”)</div>
  <div id=“main”>@RenderBody()</div>
</body>
```

63

Instance

```
@{
  Layout = "~/Views/Shared/Layout.cshtml";
  ViewBag.Title = "The Index!";
}
@section Foobar {
  <p>This is featured.</p>
}
<p>This is the main content!</p>
```

64

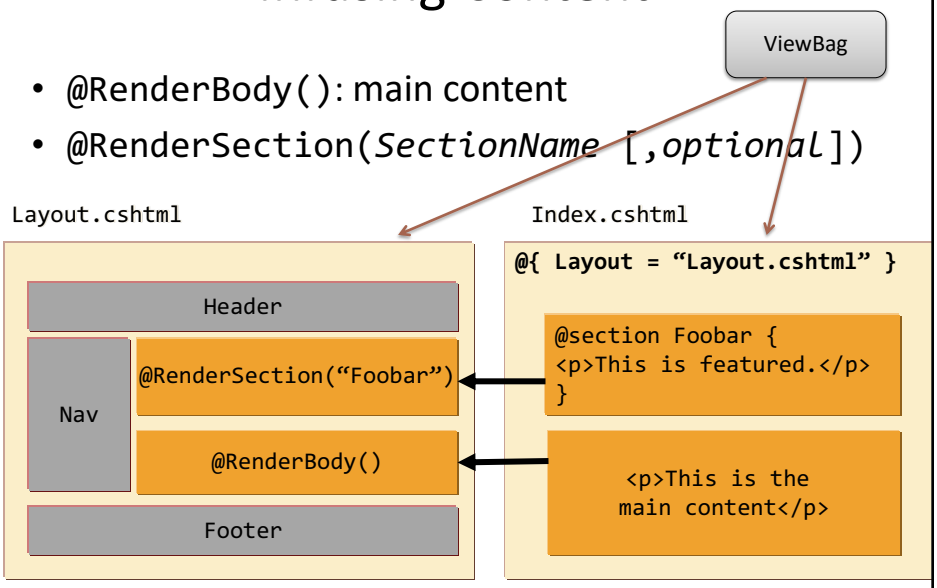
Response Document

```
<!DOCTYPE html>
<html>
<head>
  <title> The Index! </title>
</head>
<body>
  <h1> The Index! </h1>
  <div ...><p>This is featured.</p></div>
  <div id="main">
    <p>This is the main content!</p>
  </div>
</body>
```

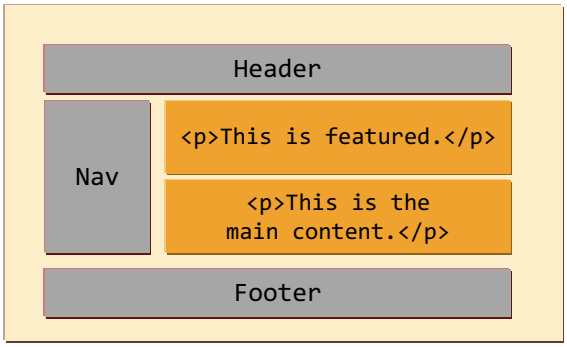
65

Infusing Content

- @RenderBody(): main content
- @RenderSection(*SectionName* [,*optional*])



Result of Infusion



67

Default Content

```
<footer>
@if (IsSectionDefined("Footer")) {
  RenderSection("Footer");
} else {
  <span>
    This is the default footer.
  </span>
}
</footer >
```

68

ViewStart

- Default layout in `~/Views/_ViewStart.cshtml`:

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

- Override default layout for all views
- Consolidate common custom settings

69

PARTIAL VIEWS

70

Partial View

- Server-side:

```
public class HomeController : Controller {
  public ActionResult Message() {
    ViewBag.Message = "This is partial.";
    return PartialView();
  }
```

- Client-side: Web service call

```
<div id =“result”> </div>
<script type=“text/javascript”>
$(function(){
    $(‘#result’).load(‘/home/message’);
});
</script>
```

71

Partial View

- Server-side:

```
public class HomeController : Controller {
  public ActionResult Message() {
    ViewBag.Message = "This is partial.";
    return PartialView();
  }
```

- Client-side: Web service call

```
<div id =“result”> </div>
<script type=“text/javascript”>
$(function(){
    $(‘#result’).load(‘/home/message’);
});
</script>
```

72

FORMS

73

HTML Form

- Collect user data for submission to server

```
<form action="http://www.bing.com/search"
      method="GET">
  <input type="text" name="q"/>
  <input type="submit" value="Search!"/>
</form>
```

- Request URL:
  `http://www.bing.com/search?q=`*`search-str`*

74

HTML Form

- Collect user data for submission to server

```
<form action="http://www.bing.com/search"
      method="GET">
  <input type="text" name="q"/>
  <input type="submit" value="Search!"/>
</form>
```

- GET method: Result can be cached
- POST method: May cause state change in server

75

HTML Form

- Define view for search form (in `/Home/Views/Index.cshtml`):

```
<form action="/Home/Search"
      method="GET">
  <input type="text" name="q"/>
  <input type="submit" value="Search!"/>
</form>
```

- Problem: hard-coded routing logic

76

HTML Form

- Define search view with HTML helpers:

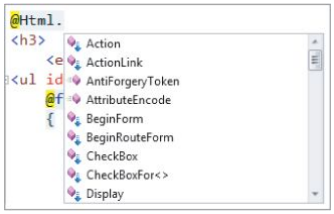
```
@{ Html.BeginForm("Search", "Home",
                  FormMethod.Get); }
  <input type="text" name="q"/>
  <input type="submit" value="Search!"/>
@{ Html.EndForm(); }
```

- HTML helpers encapsulate routing logic

77

HTML Helpers

- Automatic HTML Encoding
  - Protection against XSS
- Close to the metal
  - Unlike Web Forms controls



- `Html`: property of every view
  - Type `System.Web.Mvc.HtmlHelper<`*`model-type`*`>`
  - Namespace `System.Web.Mvc.Html`
  - Override by removing NS from `Views/Web.config`

78

HTML Form

- Add attributes:

```
@{ Html.BeginForm("Search", "Home",
                  FormMethod.Get,
                  new { target="_blank" }))
  <input type="text" name="q"/>
  <input type="submit" value="Search!"/>
@{ Html.EndForm(); }
```

- may produce:

```
<form action="/Home/Search" method="get"
      target="_blank"> ... </form>
```

79

HTML Form

```
@{ Html.BeginForm(); }
  <fieldset>
    <legend> Edit Image</legend>
    <p> @Html.Label("TagId")
        @Html.DropDownList("TagId",
           ViewBag.Tags as SelectList) </p>
    <p> @Html.Label("Caption")
        @Html.TextBox("Caption", Model.Caption)
        @Html.ValidationMessage("Caption") </p>
    <input type="submit" value="Save" />
  </fieldset>
@{ Html.EndForm(); }
```

80

Actions and Forms

GET /Home/Upload

```
[HttpGet]
public ActionResult Upload()
{
   ... return View(...);
}
```

```
<form ...>
  <input id="Caption"
    type="text" .../>
  <input id="DateTaken"
    type="text" .../>
</form>
```

```
@Html.BeginForm(...)
@Html.TextBox("Caption")
@Html.TextBox("DateTaken")
@Html.EndForm(...)
```

POST /Home/Upload

```
[HttpPost]
public ActionResult Upload(...)
{
   ... return View(...);
}
```

81

Consuming Form Input

- Use Request property from the context:

```
[HttpPost]
public ActionResult Upload () {
   Image data = new Image();
   data.Id = Request.Form["Id"];
   data.Caption = Request.Form["Caption"];
   ...
}
```

82

Consuming Form Input

- Use FormCollection:

```
[HttpPost]
public ActionResult Upload
                    (FormCollection values) {
   Image data = new Image();
   data.Id = values["Id"];
   data.Caption = values["Caption"];
   ...
}
```

83

Consuming Form Input

- Use named parameters (preferred!):

```
[HttpPost]
public ActionResult Upload (String Id,
          String Caption, ...) {
   Image data = new Image();
   data.Id = Id;
   data.Caption = Caption;
   ...
}
```

84

Consuming Form Input

- Use a model (preferred!):

```
[HttpPost]
public ActionResult Upload (Image image) {
   ...
   // Save the data to a file
   RedirectToAction("Query",
                    new {id=image.Id})
}
```

85

Consuming Form Input

- Use a model (preferred!):

```
[HttpPost]
public ActionResult Upload (Image image) {
   if (ModelState.IsValid()) {
      ...
      // Save the data to a file
      RedirectToAction("Query",
                       new {id=image.Id})
   } else {
      return View(image);
   }
}
```

86

MODELS IN VIEWS

87

Models in Views

- Controller Action

```
public ActionResult Index(...)
{
    Image image;
    ...
    return View(image);
}
```

- View

```
@model ImageShare.Models.Image
...
@{ Image img = Model; }
```

88

Controller

```
public ActionResult Edit (...)
{
   Image image;
   ...
   return View(image);
}
```

View

```
@model ImageShare.Models.Image

@Html.BeginForm(...)
@Html.TextBox("Caption")
@Html.TextBox("DateTaken")
@Html.EndForm(...)
```

```
class Image
String Caption
Date DateTaken
```

I took this.

2012-09-12

Model

```
<form ...>
  <input id="Caption"
    type="text" .../>
  <input id="DateTaken"
    type="text" .../>
</form>
```

Form

89

TextBox Helper

- View code:

```
@Html.TextBox("Caption", Model.Caption)
```

- HTML output:

```
<input id="Caption" name ="Caption"
       type ="text"
       value ="Image caption here" />
```

90

TextBox Helper

- View code:

```
@Html.TextBox(“Caption”)
```

- HTML output:

```
<input id=“Caption" name =“Caption”
       type =“text”
       value =“Image caption here” />
```

91

TextArea Helper

- View code:

```
@Html.TextArea(“Description",
               “write <br/> here”,
               5, 80, null)
```

- HTML output:

```
<textarea id=“Description”
          name=“Description”
          cols=“80” rows=“5” >
   write &lt;br/&gt; here
</textarea>
```

92

Label Helper

- View code:
```
@Html.Label("TagId")
@Html.TextBox("TagId")
```
- HTML output:
```
<label for="TagId">Tag</label>
<input id="TagId" type="text" .../>
```
- Purpose:
  - Attach info
  - Transfer focus

93

HTML HELPERS AND MODEL STATE

94

Image Model

- Model for photographic image

```
public class Image {
    public String Id { get; set; };
    public String Caption { get; set; };
    public String Description { get; set; };
    public Date DateTaken { get; set; };
    public String UserId { get; set; };
}
```

95

Controller

```
public ActionResult Edit (...)
{
    Image image;
    ...
    return View(image);
}
```

View

```
@model ImageShare.Models.Image

@Html.BeginForm(...)
@Html.TextBox("Caption")
@Html.TextBox("DateTaken")
@Html.EndForm(...)
```

```
class Image
String Caption        I took this.
Date DateTaken        2012-09-12
```

Model

```
<form ...>
  <input id="Caption"
    type="text" .../>
  <input id="DateTaken"
    type="text" .../>
</form>
```

Form

96

Helpers and ViewBag

- Controller:
```
public ActionResult Edit(int imageId) {
   ViewBag.Caption = "I took this.";
   return View();
}
```
- View:
```
@Html.TextBox("Caption");
```
- HTML Output:
```
<input id="Caption" name="Caption"
       type="text" value="I took this."/>
```

97

Helpers and ViewBag

- Controller:
```
public ActionResult Edit(int imageId) {
   ViewBag.Image = ...;
   return View();
}
```
- View:
```
@Html.TextBox("Image.Caption");
```
- HTML Output:
```
<input id="Image_Caption" name="Image_Caption"
       type="text" value="I took this."/>
```

98

Helpers and Model

- Controller:

```
public ActionResult Edit(int imageId) {
   Image image = ...;
   return View(image);
}
```

- View:

```
@Html.TextBox("Caption");
```

- HTML Output:

```
<input id="Caption" name="Caption"
       type="text" value="I took this."/>
```

99

Helpers and Model

- Controller:

```
public ActionResult Edit(int imageId) {
   Image image = ...;
   return View(image);
}
```

- View:

```
@Html.TextBox("Caption", Model.Caption);
```

- HTML Output:

```
<input id="Caption" name="Caption"
       type="text" value="I took this."/>
```

100

Validation

- Controller:

```
public ActionResult Edit(int imageId) {
   Image image = ...;
   return View(image);
}
```

- View:

```
@Html.ValidationMessage("Caption",
                        "A caption is required!");
```

- HTML Output:

```
<span class="field-validation-error"
      data-valmsg-for="Caption"
      data-valmsg-replace="true">
 A caption is required!</span>
```

101

Validation

- Model:

```
public class Image {
   [Required]
   public String Caption { get; set; };
}
```

- View:

```
@Html.ValidationMessage("Caption",
                        "A caption is required!");
```

- HTML Output:

```
<span class="field-validation-error"
      data-valmsg-for="Caption"
      data-valmsg-replace="true">
 A caption is required!</span>
```

102

Validation

- Model:

```
public class Image {
   [Required(ErrorMessage=
             "A caption is required")]
   public String Caption;
}
```

- View:

```
@Html.ValidationMessage("Caption");
```

- HTML Output:

```
<span class="field-validation-error"
      data-valmsg-for="Caption"
      data-valmsg-replace="true">
   A caption is required!</span>
```

103

ModelState

- Used for state of input validation
- Helpers get field values from ModelState
  - Display current values for editing
  - Otherwise from ViewData, ViewBag
- Preserves validation errors
- Preserves bad input for editor

104

EXPLICITLY TYPED HELPERS

105

TextBox Helper

- View code:

```
@Html.TextBox(“Caption”)
```

- HTML output:

```
<input id=“Caption" name=“Caption”
       type=“text”
       value=“Image caption here” />
```

106

```
@model ImageShare.Models.Image;

delegate String GetProperty(Image m);

void TextBoxFor (GetProperty lambda);
```

- View code:

```
@Html.TextBoxFor(m ⇒ m.Caption)
```

- HTML output:

```
<input id="Caption" name="Caption"
       type="text"
       value="Image caption here" />
```

107

Templated Helper

- View code:

```
@Html.EditorFor(m ⇒ m.Caption)
```

- HTML output:

```
<input id="Caption" name="Caption"
       type="text"
       value="Image caption here" />
```

108

Templated Helper

- View code:

```
@Html.EditorFor(m ⇒ m.Description)
```

- HTML output:

```
<input id="Description"
       name="Description"
       type="text"
       value="I took this picture." />
```

109

```
public class Image {
   ...
   [DataType(DataType.MultiLineText)]
   public String description;
}
```

- View code:

```
@Html.EditorFor(m ⇒ m.Description)
```

- HTML output:

```
<textarea id="Description"
          name="Description"
          class="text-box multi-line" >
   I took this picture.
</textarea>
```

110

Explicitly Typed Form

```
@{ Html.BeginForm(); }
  <fieldset>
    <legend> Edit Image</legend>
    <p> @Html.LabelFor(m ⇒ m.TagId)
        @Html.DropDownListFor(m ⇒ m.TagId,
           ViewBag.Tags as SelectList) </p>
    <p> @Html.LabelFor(m ⇒ m.Caption)
        @Html.EditorFor(m ⇒ m.Caption)
        @Html.ValidationMessageFor
           (m ⇒ m.Caption) </p>
    <input type="submit" value="Save" />
  </fieldset>
@{ Html.EndForm(); }
```

111

OTHER HELPERS

112

Other Input Helpers

- `Html.Hidden`, `Html.HiddenFor`
- `Html.Password`, `Html.PasswordFor`
- `Html.RadioButton`, `Html.RadioButtonFor`
- `Html.CheckBox`, `Html.CheckBoxFor`

113

Rendering Helpers

- Html.ActionLink

```
@Html.ActionLink (link-text, action)
@Html.ActionLink
    (link-text, action, controller)
@Html.ActionLink
    (link-text, action, controller,
     argument-list, attribute-list)
```

- Html.RouteLink

```
@Html.RouteLink (link-text, route)
```

114

URL Helpers

- Action

```
<span>
@Url.Action("Query", "Image",
            new { id = "anthonyquinn" },
            null )
</span>

<span> /Image/Query?id=anthonyquinn </span>
```

- RouteUrl
- Content

115

URL Helpers

- Action
- RouteUrl
- Content

```
<script
   src="@Url.Content(
         "~/Scripts/jquery-1.5.1.min.js")",
   type="text/javascript">
</script>
```

116

Other HTML Helpers

- Partial view as a string

```
Html.Partial( view-name
              [, model]
              [, view-data] )
```

- Partial view to response output stream

```
Html.RenderPartial(...)

@{Html.RenderPartial("ImageData");}
@Html.Partial("ImageData")
```

117

Action HTML Helper

- `Html.Partial`: insert external partial view
- `Html.Action`: call external action

```
public class MyController {
   public ActionResult Index() {
      return View();
   }
   [ChildActionOnly]
   public ActionResult Menu() {
      var menu = GetMenuFromSomewhere();
      return PartialView( menu);
   }
}
```

118

Action HTML Helper

- View for Menu

```
[ChildActionOnly]
public ActionResult Menu() {
   var menu = GetMenuFromSomewhere();
   return PartialView( menu);
}

@model Menu
<ul>
@foreach (var item in Model.MenuItem) {
   <li> @item </li> } </ul>
} </ul>
```

119

Action HTML Helper

- View for Index:

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

<html>
   <head> <title> ... </title> </head>
   <body>
      @Html.Action("Menu")
      <h1>Welcome to the Index View</h1>
   </body>
</html>
```

120

Summary

- ASP.NET MVC
  - Powerful abstractions for Web applications
  - Strong typing possible
  - Still “close to the metal”
- Next: Models
  - Entity Framework
  - LINQ

121