Module05

Developing ASP.NET MVC Views

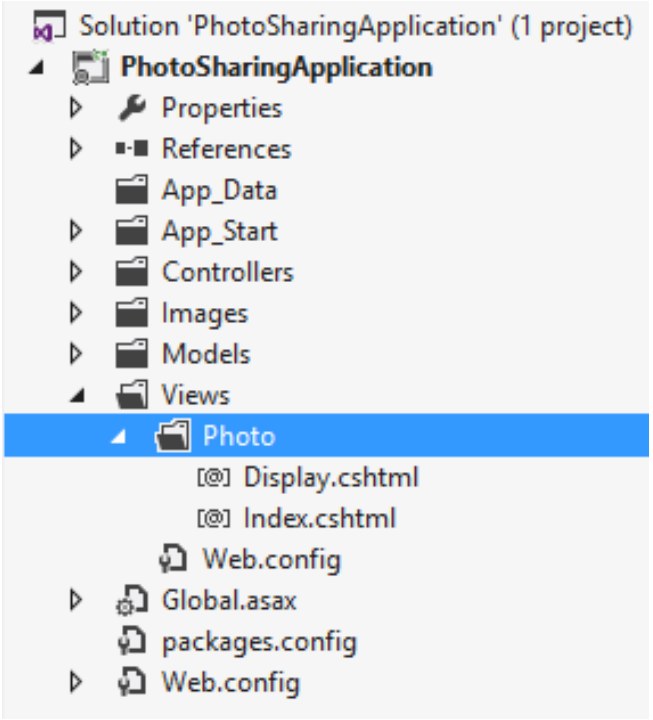
Module Overview

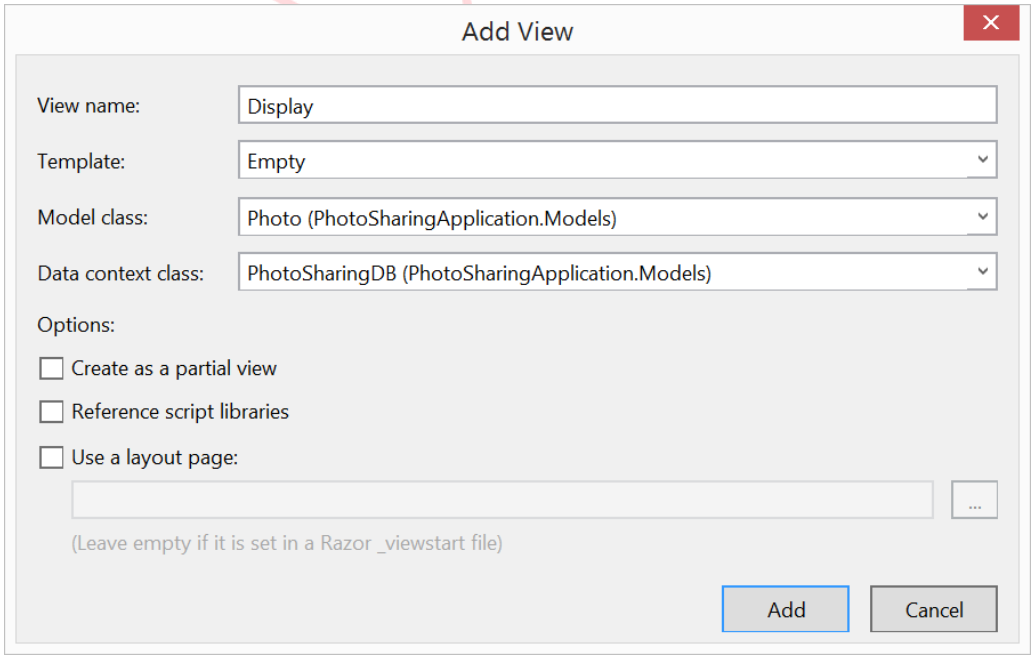
- Creating Views with Razor Syntax
- Using HTML Helpers
- Re-using Code in Views

Lesson 1: Creating Views with Razor Syntax

- Adding Views
- Differentiating Server Side Code from HTML
- Features of Razor Syntax
- Binding Views to Model Classes and Displaying Properties
- Rendering Accessible HTML
- Alternative View Engines

Adding Views





Differentiating Server Side Code from HTML

- Razor identifies server-side code by looking for the **@** symbol.
- In Razor syntax, the **@** symbol has various uses. You can:
  - Use **@** to identify server-side C# code.
  - Use **@@** to render an @ symbol in an HTML page.
  - Use **@:** to explicitly declare a line of text as content and not code.
  - Use **<text>**to explicitly declare several lines of text as content and not code.
- To render text without HTML encoding, you can use the **Html.Raw()** helper.

Features of Razor Syntax

- A sample code block displaying the features of Razor.

```
@* Some more Razor examples *@

<span>
Price including Sale Tax: @Model.Price * 1.2
</span>
<span>
Price including Sale Tax: @(Model.Price * 1.2)
</span>

@if (Model.Count > 5)
{
 <ol>
   @foreach(var item in Model)
   {
      <li>@item.Name</li>
   }
 </ol>
}
```

Binding Views to Model Classes and Displaying Properties

- You can use strongly-typed views and include a declaration of the model class. Visual Studio helps you with additional IntelliSense feedback and error-checking as you write the code.
- Binding to Enumerable Lists:

```
@model IEnumerable<MyWebSite.Models.Product>

<h1>Product Catalog</h1>

@foreach (var Product in Model)

{

   <div>Name: @Product.Name</div>

}
```

- You can use dynamic views to create a view that can display more than one model class.

Rendering Accessible HTML

- You can ensure that your content is accessible to the broadest range of users by adhering to the following guidelines:
  - Provide **alt** attributes for visual and auditory content
  - Do not rely on color to highlight content
  - Separate content from structure and presentation code:
    - Only use tables to present tabular content
    - Avoid nested tables
    - Use ** ** elements and positional style sheets to lay out elements on the page
    - Avoid using images that include important text
    - Put all important text in HTML elements or **alt** attributes

Alternative View Engines

The following table describes and compares four popular view engines.

View Engine	Description
Razor	This is the default view engine for MVC 4. It is easy to learn for anyone who has experience in HTML and C# or Visual Basic.
ASPX	This was the default view engine for MVC 2. It is easy to learn for developers with experience in ASP.NET Web Forms and is also known as the Web Forms View Engine.
NHaml	NHaml is a .NET Framework version of the Haml view engine used with Ruby on Rails, a competitor to the ASP.NET MVC technology.
Spark	This view engine uses view files that are easily readable and similar to static HTML files.

Lesson 2: Using HTML Helpers

- Using Action Helpers
- Using Display Helpers
- The Begin Form Helper
- Using Editor Helpers
- Using Validation Helpers
- Demonstration: How to Use HTML Helpers

Using Action Helpers

- **Html.ActionLink()**

```
@Html.ActionLink("Click here to view photo 1",
  "Display", new { id = 1 })
```

```
<a href="/photo/display/1">
  Click here to view photo 1
</a>
```

- **Url.Action()**

```
<img alt="This image came from an action"
  src="@Url.Action("GetImage", new { id = 1 })" />
```

```
<img
  alt="This image came from an action"
  src="/photo/getimage/1" })"
/>
```

Using Display Helpers

- **Html.DisplayNameFor()**

```
@Html.DisplayNameFor(model => model.CreatedDate)
```

Created Date

- **Html.DisplayFor()**

```
@Html.DisplayFor(model => model.CreatedDate)
```

03/12/2012

The Begin Form Helper

Html.BeginForm()

```
@using (Html.BeginForm("Create", "Photo",
   FormMethod.Post,
   new { enctype = "multipart/form-data" }))
{
   @* Place input controls here *@
}
```

```
<form action="/Photo/Create" method="post“
   enctype="multipart/form-data">

</form>
```

Using Editor Helpers

Html.LabelFor()

```
@Html.LabelFor(model => model.ContactMe)
```

```
<label for="ContactMe">
  Contact Me
</label>
```

Html.EditorFor()

```
@Html.EditorFor(model => model.ContactMe)
```

```
<input type="checkbox"
  name="Description">
```

Using Validation Helpers

Html.ValidationSummary()

```
@Html.ValidationSummary()
```

```
<ul>
  <li>Please enter your last name</li>
  <li>Please enter a valid email address</li>
</ul>
```

Html.ValidationMessageFor ()

```
@Html.ValidationMessageFor(model => model.Email)
```

Please enter a valid email address

Demonstration: How to Use HTML Helpers

In this demonstration, you will see how to:

- Create a new view.
- Use the **Html.BeginForm** helper to render an HTML form.
- Use the **Html.LabelFor** helper to render a label for a model property.
- Use the **Html.EditorFor** helper to render an editor control for a model property.
- Use the **Html.ValidationMessageFor** helper to render validation errors.
- Use the **Html.ActionLink** helper to render a link to an action.

Lesson 3: Re-using Code in Views

- Creating Partial Views
- Using Partial Views
- Discussion: Partial View Scenarios

Creating Partial Views

You can use partial views to render the same HTML content in different locations in your web application

- Creating and Naming Partial Views:
  - Create a partial view by using the **Add View** dialog
  - Name partial views with an underscore prefix to keep to convention
- Strongly-typed and dynamic partial views:
  - Create strongly-typed partial views if you are certain that the partial view will always display the same model class
  - Create dynamic partial views if you are not sure if the partial view will always display the same model class

Using Partial Views

- Using HTML helpers, you can use partial views within other views in a web application:
  - To pass the same model object to a partial view from the parent view, use **Html.Partial()**
  - To pass a model object to a partial view, which is different from the parent view or of a different model class, use **Html.Action()**

- Use the **ViewBag** and **ViewData** collections to share data between the controller action, parent view, and partial view

Discussion: Partial View Scenarios

Determine the appropriate use of HTML helpers in these scenarios:

- In a product web application:
  - You want to display the ten latest comments in a new webpage when users click the **Latest Comments** link on the home page
- In a product web application:
  - You want to display the comments for an article in the **Article** view
  - You also want to display the comments for a product in the **Product** view
  - There are separate **ArticleComment** and **ProductComment** classes in your model, but they have similar properties
- In a photo sharing web application:
  - You want to display a gallery of thumbnail images on both the **AllPhotos** view and the home page **Index** view.
  - You want the **AllPhotos** view to display every **Photo** object, but you want the home page **Index** view to display only the three most recent photos uploaded

Lab: Developing ASP.NET MVC 4 Views

- Exercise 1: Adding a View for Photo Display
- Exercise 2: Adding a View for New Photos
- Exercise 3: Creating and Using a Partial View
- Exercise 4: Adding a Home View and Testing the Views

Estimated Time: 60 minutes

Lab Scenario

You have been asked to add the following views to the photo sharing application:

- *A Display view for the Photo model objects*. This view will display a single photo in a large size, with the title, description, owner, and created date properties.
- *A Create view for the Photo model objects*. This view will enable users to upload a new photo to the gallery and set the title and description properties.
- *A Photo Gallery partial view*. This view will display many photos in thumbnail sizes, with the title, owner, and created date properties. This view will be used on the **All Photos** webpage to display all the photos in the application. In addition, this view will also be used on the home page to display the three most recent photos.

After adding these three views to the photo sharing application, you will also test the working of the web application.

Lab Review

- How can you improve the accessibility of the HTML that your photo views render?
- In the lab, how did you ensure that the Create view for Photo model objects could upload photo files when the user clicked the Create button?

Module Review and Takeaways

- Review Question(s)
- Best Practice
- Common Issues and Troubleshooting Tips