COMP 2139
How to work with Razor Views

Agenda

- Distinguish between a Razor code block and inline expressions, loops, and if statements
- Distinguish between an inline conditional statement and an inline conditional expression
- Describe how an MVC web application typically maps its views to the action methods of its controllers
- Describe the use of tag helpers to generate a URL
- Describe the difference between a relative URL and an absolute URL
- Describe how to pass a model to a view
- Distinguish between the @model directive and the @Model property
- Describe how to bind an HTML element to a property of a view's model
- Descibe how to bind a <select> element to a list of items
- Describe how to display a list of items in a <table> element
- Describe how to create and apply a layout
- Describe how to build layouts by nesting one layout within another
- Describe how a layout can use the ViewContext property to get data about the route of the current view.
- Describe how to code a section in a view and render that section in layout.

Razor Views

Razor Views

- Razor is an ASP.NET programming syntax used to create dynamic web pages with the C#
- The Razor syntax is a template markup syntax, based on the C# programming language, that enables the programmer to use an HTML construction workflow.
- Razor syntax starts code blocks with an **@** character.

The Syntax for Razor code block

```
@{
    // one or more C# statements
}
```

The syntax for an inline expression

```
@(csharp_expression)
```

Example HomeController

The Index() action method of an example HomeController

```
public IActionResult Index()
{
    ViewBag.CustomerName = "John";
    return View();                          // returns → Views/Home/Index.cshtml
}
```

The Views/Home/Index.cshtml file

```
@{
    string message = "Welcome!";
    if (ViewBag.CustomerName != null)
    {
        message = "Welcome back!";
    }
}

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Home</title>
</head>
<body>
    <h1>@message</h1>
    <p>Customer Name: @ViewBag.CustomerName</p>
    <p>2 + 2 = @(2 + 2)</p>
</body>
</html>
```

Razor Block

Razor inline expression

Drop-Down List

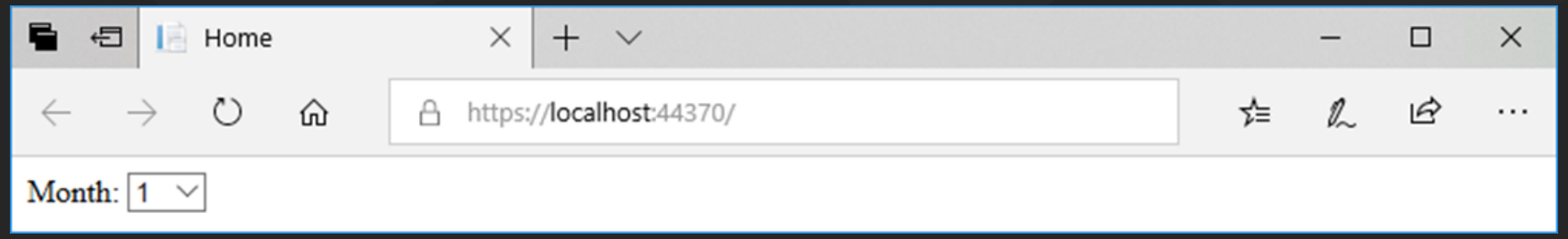
Example - For Loop

A For Loop that displays a drop-down list of month numbers

```
<label for="month">Month:</label>
<select name="month" id="month">
    @for (int month = 1; month <= 12; month++)
    {
        <option value="@month">@month</option>
    }
</select>
```

- Inline loop within a cshtml view.
- Within these loops you can use HTML tags to send HTML to the view.

The result displayed in a browser



Displaying a Collection List

Example – Creating a List in a Controller

Code in a controller that creates a list of strings

```
public IActionResult List()
{
    ViewBag.Categories = new List<string>
    {
        "Guitars", "Basses", "Drums"
    };
    return View();
}
```

- Loop inside List() action of Home controller sets up the list of categories to display.

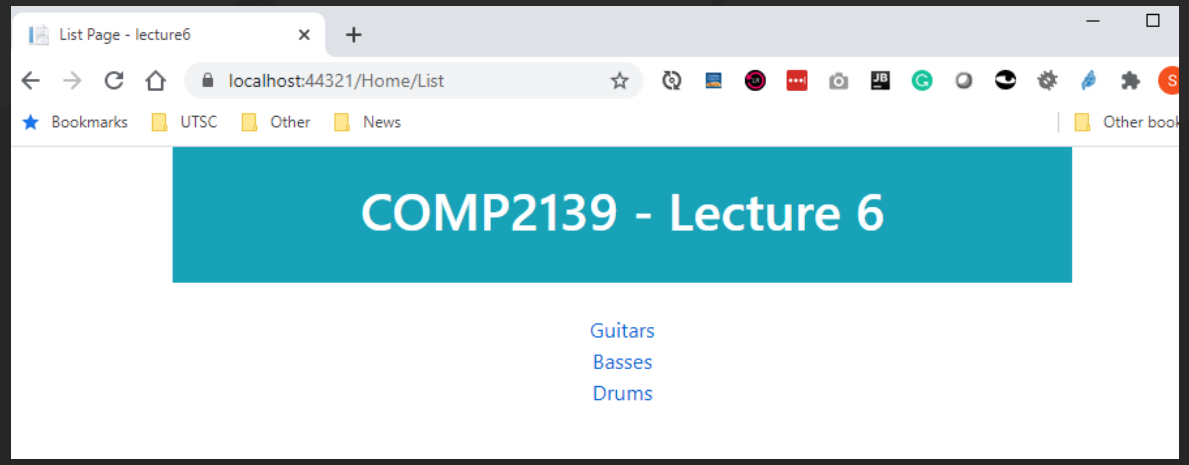
Example – Creating a List in a Controller…

A foreach loop that displays a list of links

```
@forach(string category in ViewBag.Categories)
{
    <div>
        <a href="/Product/List/@category/">@category</a>
    </div>
}
```

- foreach loop inside List.cshtml view
- foreach loop declare a category variable to is set to the value of the Category stored in the Categories property of the ViewBag

The result displayed in a browser



if-else statement

if-else statement

An if-else statement in a view

```
@if (ViewBag.ProductID == 1)
{
    <p>Fender Stratocaster</p>
}
else if(ViewBag.ProductID ==2)
{
    <p>Gibson Les Paul</p>
}
else
{
    <p>Product Not Found</p>
}
```

- You can use inline statement such as an **if-else** statement to send HTML to a view

switch statement

switch statement

A switch statement in a view

```
@switch (ViewBag.ProductID)
{
  case 1:
     <p>Fender Stratocaster</p>
  break;

  case 2:
     <p>Gibson Les Paul</p>
  break;

 default:
     <p>Product Not Found</p>
  break;
}
```

More if conditional Examples

More Examples

An if statement that adds bootstrap CSS if condtional is true

```
<a asp-controller="Product" asp-action="List" asp-route-id="All" class="list-group-item
    @if(ViewBag.SelectedCategoryName == "All")
    {
        <text>active</text>
    }">
    All
</a>
```

- If statement to check whether the SelectedCategoryName property of the ViewBag is equal to a string “All”.
- If true, it adds the Bootstrap CSS class named “active” (ex class = “list-group-item active”) to the class attribute of the <a> tag element.
- To make this work, the example uses the **<text>** tag element to send the “active” class attribute value as plain text

More Examples…..

A conditional expression that adds bootstrap CSS if condtional is true

```
<a asp-controller="Product" asp-action="List" asp-route-id="All" class="list-group-item
    @(ViewBag.SelectedCategoryName == "All" ? "active" : "")">
    All
</a>
```

- If statement to check whether the SelectedCategoryName property of the ViewBag is equal to a string “All”.
- If true, it adds the Bootstrap CSS class named “active” (ex class = “list-group-item active”) to the class attribute of the **<a>** tag element.

Example Project Guitar Shop Application

The Project Setup

The Guitar Shop

- The example application follows the conventions for storing the models, controllers, and static files for an application.
- folder and file structure maps the actions of each controller to the appropriate view etc..

```
GuitarShop
    /Controllers
        /HomeController.cs
        /ProductController.cs
    /Models
        /Category.cs
        /Product.cs
    /Views
        /Home
            /Index.cshthml    -- the view for the Home/Index action
            /About.cshthml    -- the view for the Home/About action
        /Product
            /List.cshthml     -- the view for the Product/List action
            /Details.cshthml  -- the view for the Product/Details action
            /Update.cshthml   -- the view for the Product/Update action
        /Shared
            /_Layout.cshtml   -- a layout that can be shared by views
        _ViewImports.cshtml   -- imports models and tag helpers for views
        _ViewStart.cshtml     -- specifies the default layout for views
    /wwwroot
        /css
            /custom.css
        /lib
            /bootstrap/css/bootstrap.min.css
    Program.cs                -- sets up the app
                              -- configures middleware that may impact views
```

The Project Routing

The routing that's specified in the Program.cs file

```
app.UseEndpoints(endpoints =>
{
    // map route for Admin area
    endpoints.MapAreaControllerRoute(
        name: "admin",
        areaName: "Admin",
        pattern: "Admin/{controller=Home}/{action=Index}/{id?}");

    // map default route
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}/{slug?}");
});
```

Maps admin routing to HomeController, the second segment to the controllers Index action, third segment to optional id arguement.

Maps default routing to HomeController, the second segment to the controllers Index action, third and fourth to optional id and slug arguements.

- The Program.cs file contains code that configures middleware for the application, including routing that specifies how controllers are their action methods are mapped to URLs.

Controllers that return Views

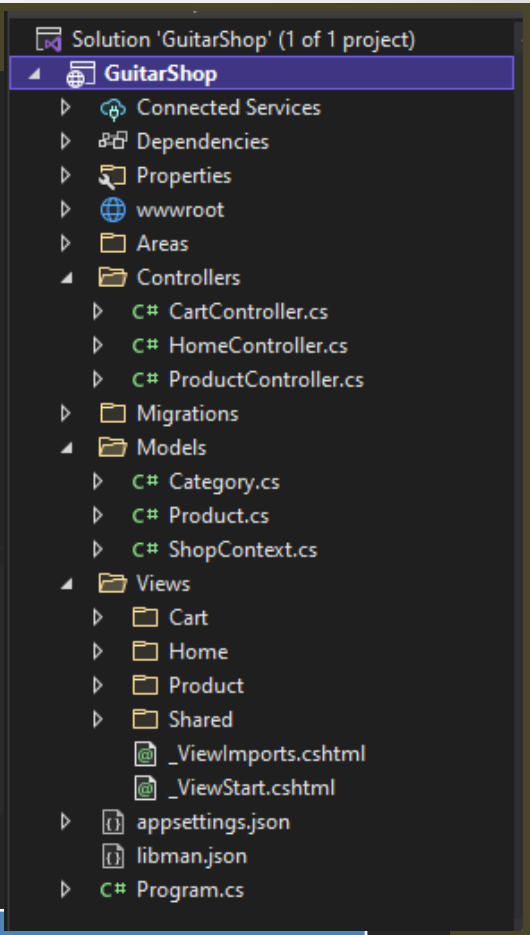
The HomeController

```
public class HomeController : Controller
{
    0 references
    public IActionResult Index()
    {
        return View();
    }

    0 references
    public IActionResult About()
    {
        return View();
    }
}
```

/Views/Home/Index.cshtml

/Views/Home/About.cshtml



Method	Description
View()	Creates a ViewResult object that corresponds to the name of the current controller and action method
View(name)	Creates a ViewResult object that corresponds to the current controller and the specified view name.

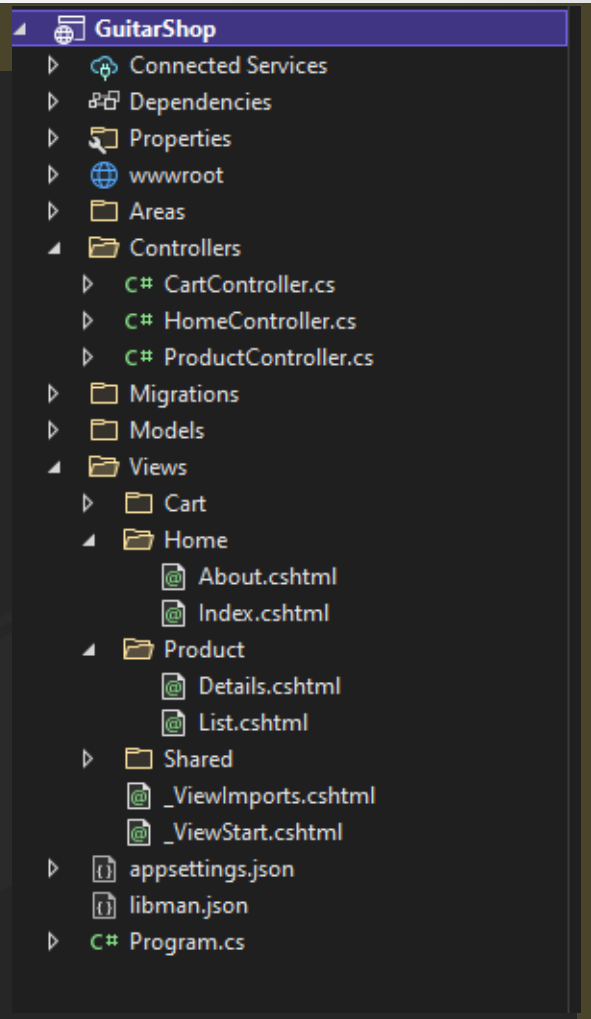
Controllers that return Views…

The ProductController

```
public class ProductController : Controller
{
    public IActionResult Index()
    {
        return View("List");          // Views/Product/List.cshtml
    }

    public IActionResult List(string id = "All")
    {
        ViewBag.Category = id;
        return View();                // Views/Product/List.cshtml
    }

    public IActionResult Details(string id)
    {
        ViewBag.ProductSlug = id;
        return View();                // Views/Product/Details.cshtml
    }
}
```



- You can also specify the name of the view in the View() method. For example both the Index() and List() action methods in the above example, create a ViewResult object from the Product/List.cshtml file.

Controllers that return Views…

The ProductController

```
public class ProductController : Controller
{
    public IActionResult Index()
    {
        return View("List");          // Views/Product/List.cshtml
    }

    public IActionResult List(string id = "All")
    {
        ViewBag.Category = id;
        return View();               // Views/Product/List.cshtml
    }

    public IActionResult Details(string id)
    {
        ViewBag.ProductSlug = id;
        return View();               // Views/Product/Details.cshtml
    }
}
```

- The List() and Details() action methods in this example, store the id argument in the ViewBag property, so it can later be accessed by the resultant view.
- All actions in this example, specify a return type of the IActionResult interface.
- ViewResult implements the IActionResult interface.

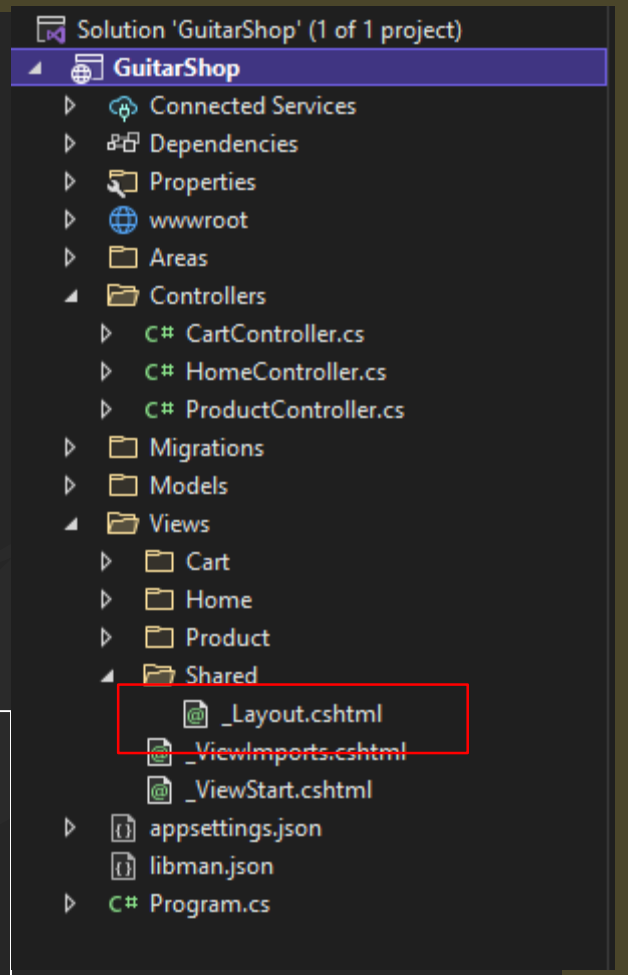
How to create a default Layout

The _Layout.cshtml (in Views/Shared)

The _Layout.cshtml

```
<!DOCTYPE html>

<html>
<head>
    <title>@ViewBag.Title</title>
    <link rel="stylesheet" href="~/lib/bootstrap/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
<body>
    @RenderBody()
</body>
</html>
```



- A Razor layout typically specifies code that shared between multiple views, such as **<html>, <head>, <body>** elements for a view.
- A Razor layout also typically imports CSS and JavaScript files for a view.
- Within a layout , the ViewBag is often used to display a title set in the view **(@ViewBag.Title)**.
- The **RenderBody()** method renders the body of the view (.cshtml forwarded to)

Default Layout…

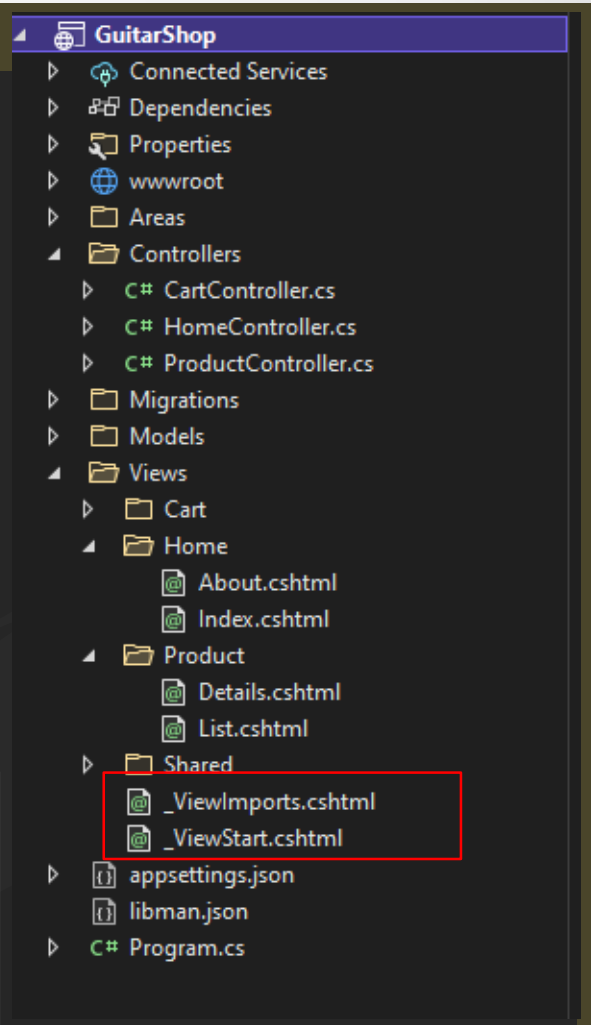
The _ViewStart.cshtml

```
@{
    Layout = "_Layout";
}
```

The _ViewImport.cshtml

```
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@using GuitarShop.Models
```

- In a typical web application, most of the views share the same layout. As a result, it usually makes sense to configure the default layout in the _ViewStart.chtml
- When working with Views, its common to use the tag helpers that are available from ASP.NET Core MVC. The easiest way to do that is to add a _ViewImports file to your project.



Tag Helpers for Generating URLs

Tag Helpers for Generating URLs

Three tag helpers for generating URLs

Tag Helper	Description
asp-controller	Specifies the controller. This is only necessary if you want to specify a URL for an action method from another controller.
asp-action	Specifies the action method
asp-route-*param_name*	Specifies a route parameter where ***param_name*** is the name of the parameter. If you specify a parameter name that exists on one of the applications routes, the application uses the parameter value as a segment of the URL. Otherwise, it adds the parameter name and value to the end of the URL as part of the URLs **query string** (?name1=value&name2=value).

Coding links

Use HTML to hard code the URL in the href attribute

```
<a href="/Product/List/Guitars">View guitars</a>
```

Use ASP.NET tag helpers to generate the URL

```
<a asp-controller="Product" asp-action="List" asp-route-id="Guitars">
   View guitars
</a>
```

The URL for both links

```
/Product/List/Guitars
```

Coding links …

How to code a link to an action method in the same controller

```
<a asp-action="About">About Us</a>
```

/Home/About

How to Code a link to an action method in a different controller

```
<a asp-controller="Product" asp-action="List">
   View all products
</a>
```

/Product/List

Coding links …

How to code a link that includes a parameter

```
<a asp-controller="Product" asp-action="List" asp-route-id="Guitars">
   View guitars
</a>
```

/Product/List/Guitars

A link that specifies a route parameter name that doesn't exist in the apps route

```
<a asp-controller="Product" asp-action="List" asp-route-page="1"
   asp-route-sort_by="price">
   Products - Page 1
</a>
```

/Product/List?page=1&sort_by=price

Example: Views that use the Default Layout and Tag Helpers

Example 1: Home/Index

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

<h1>Home</h1>

<div class="list-group">
    <a asp-controller="Product" asp-action="List">
        View all products
    </a>

    <a asp-controller="Product" asp-action="List" asp-route-id="Guitars">
        View guitars
    </a>

    <a asp-controller="Product" asp-action="Details" asp-route-id="Fender-Stratocaster">
        View Fender Stratocaster
    </a>
</div>
```

Razor code block stores title for page in ViewBag

/Product/List

/Product/List/Guitars

/Product/Details/Fender-Stratocaster

Example 2: Product/List view

```
@{
    ViewBag.Title = "Product List";
}

<h1>Product List</h1>

<div class="list-group">
    <p>Category: @ViewBag.Category</p>

    <a asp-action="Details" asp-route-id="Fender-Stratocaster">
        View Fender Stratocaster</a>

    <a asp-controller="Home" asp-action="Index">Home</a>
</div>
```

Razor code block stores title for page in ViewBag

/Product/Details/Fender-Stratocaster

/Home/Index

Example 3: Product/Details view

```
@{
    ViewBag.Title = "Product Details";
}

<h1>Product Details</h1>

<div class="list-group">
    <p>Slug: @ViewBag.ProductSlug</p>

    <a asp-controller="Home" asp-action="Index">Home</a>
</div>
```

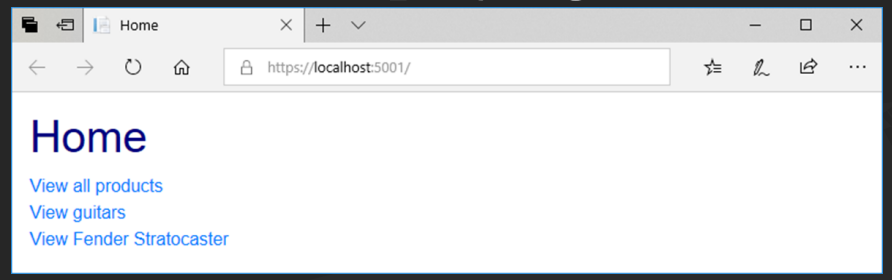
Razor code block stores title for page in ViewBag

/ prints ViewBag ProductSlug property

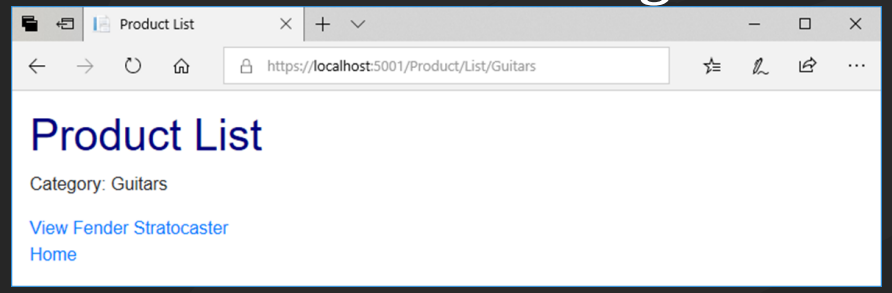
/Home/Index

Display Output

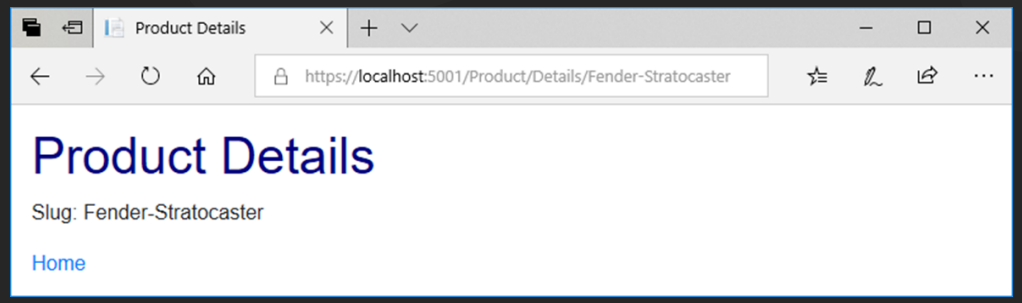
A browser displaying the Home Page



A browser after clicking the View Guitars link



A browser after requesting the View Fender Stratocaster



More tag helpers

Tag Helper	Description
asp-area	Specifies the area for the URL.
asp-fragment	Specifies the placeholder that you want to jump to.
asp-protocol	Specifies the protocol. By default, it is set to HTTP. However, its common for applications to redirect to HTTPS, even if you specify HTTP.
asp-host	Specifies the name of the server

- A URL fragment allows you to jump to a specified placeholder on a web page. In a URL, a fragment is preceded by the hash mark (#)

More tag helpers… Area Link

How to code a link to an area

```
<a asp-area="Admin" asp-controller="Product" asp-action="List">
   Admin - Product Manager</a>
```

The URL that's generated

```
/Admin/Product/List
```

More tag helpers… HTML Placeholder

How to code an HTML placeholder

```
<h2 id="Fender">Fender Guitars</h2>
```

How to code a URL that jumps to an HTML placeholder on the same page

```
<a asp-fragment="Fender">View Fender Guitars</a>
```

Causes the browser to move up or down until it displays the part of the page that contains the placeholder (element with matching id)

The URL that's generated

```
/#Fender
```

More tag helpers… Absolute URLs

How to an absolute URL

```
<a asp-protocol="https"
   asp-host="domain.com"
   asp-controller="Shop"
   asp-action="Details"
   asp-route-id="net-programming"
   asp-fragment="course">COMP2139</a>
```

The URL that's generated

```
https://domain.com/Shop/Details/net-programming/#course
```

Format Numbers in Views

Format Specifiers used to format numbers

Format Specifiers

Specifier	Name	Description
C	Currency	Formats a number as a currency value for the current locale
N	Number	Formats a number using a separator for the thousandths (,) place
P	Percent	Formats a number as a percentage

Format Specifiers used to format numbers

Example Code that stores numbers in the ViewBag

```
ViewBag.Price = 12345.67;
ViewBag.DiscountPercent = .045;
ViewBag.Quantity = 1234;
```

Expression	Result
`@ViewBag.Price.ToString("C")`	$12,345.67
`@ViewBag.Price.ToString("C1")`	$12,345.7
`@ViewBag.Price.ToString("C0")`	$12,346
`@ViewBag.Price.ToString("N")`	12,345.67
`@ViewBag.Price.ToString("N1")`	12,345.7
`@ViewBag.Price.ToString("N0")`	12,346
`@ViewBag.DiscountPercent.ToString("P")`	4.50%
`@ViewBag.DiscountPercent.ToString("P1")`	4.5%
`@ViewBag.DiscountPercent.ToString("P0")`	5%

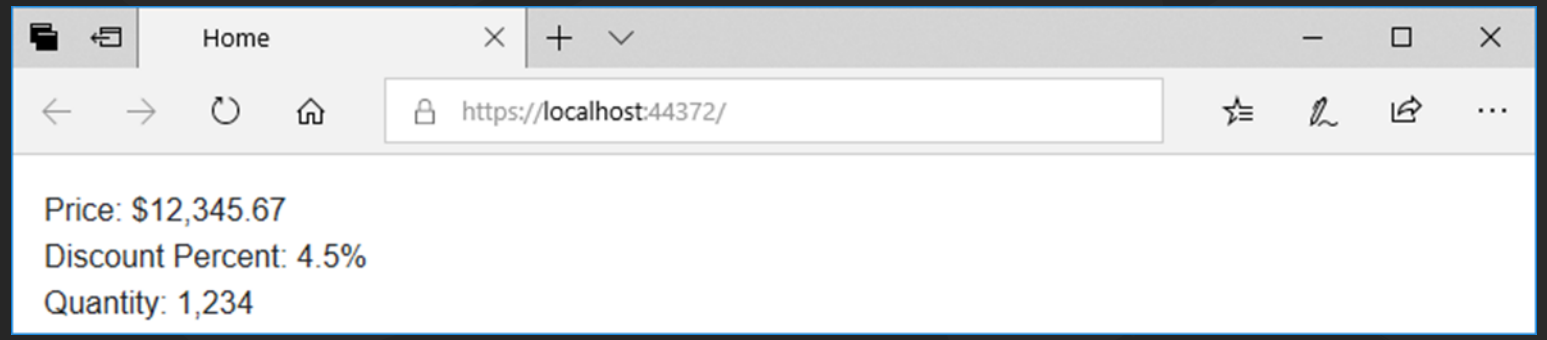
Specifier	Name
`C`	Currency
`N`	Number
`P`	Percent

Format Specifiers used to format numbers….

Code in a view that displays the nunbers

```
<div>Price: @ViewBag.Price.ToString("C")</div>
<div>Discount Percent: @ViewBag.DiscountPercent.ToString("P1")</div>
<div>Quantity: @ViewBag.Quantity.ToString("N0")</div>
```

The View in the browser



Questions?