Razor view engine

Lesson Objectives

- *Razor view engine*
- *Razor Syntax*
- *Control structures*

Section 1

RAZOR VIEW ENGINE

Which code should you prefer?

```
<ul>
   <%foreach (var item in Products)
      {  %>
         <% if (item.IsInStock)
            { %>
                 <p><%=item.ProductName%> is in stock</p>
             <% }
            else
            { %>
                 <p><%=item.ProductName%> is not in stock</p>
            <% } %>
   <%} %>
</ul>
```

```
<ul>
   @foreach (var item in Products)
   {
            @if(item.IsinStock)
            {
               @item.ProductName is in stock
            } else {
               @item.ProductName is not in stock
            }
   }
</ul>
```

View Engine

- View Engine renders the HTML to the browser.
- By default ASP.Net MVC supports ASPX and the Razor View Engine.
- There are many more third-party view engines, like Spark, Nhaml and so on also available for MVC.

ASPX View Engine

- The syntax used for writing a view with the ASPX View Engine is the same as the syntax used in ASP.Net web forms.
- The file extensions are also the same as for ASP.NET web forms (like .aspx, .ascx, .master).
- ASPX uses "<%= %>" or "<%: %>" to render server-side content.
- Implementing the unit testing framework with the ASPX View Engine is very difficult.

Razor View Engine

- Razor View engine is a mark-up syntax which helps us to write HTML and server-side code in web pages using C# or VB.NET.
- It is server-side mark-up language however it is not at all a programming language.
- It available with MVC 3.0 and later versions.

Razor View

- Razor uses the "@" character to specify code block.
- Razor does not require the code block to be closed, the Razor View Engine parsed itself and it is able to decide during run time that it is a presentation element (content) and that it is a code element.
- The file extension of a Razor view is *.cshtml (for C#) and *.vbhtml (for VB.NET).
- The Razor View Engine is compatible with a unit testing framework.

Razor View

- Razor is not a new language.
- It is easy to learn.
- The main advantage of Razor, is that there is less transition between HTML and code because Razor provides an optimized syntax to generate HTML using a code focused templating approach.

Advantages of Razor View Engine

- Easy to Learn
  - ✓ We can also use our existing C# and HTML skills.
  - ✓ The code looks clean.
- Compact, Expressive, and Fluid
  - ✓ Razor helps us to minimize the coding and provide us a fast and fluid coding work flow.
- Razor does not require any special tool to write mark-up.
  - ✓ We can also write our mark-up code with any old plain text editor like Notepad.

Advantages of Razor View Engine

- ASP.NET MVC has HTML helpers that are methods that can be invoked within a code block.
  - ✓ All existing HTML extension methods can be used with a Razor View Engine without any code changes.
- Powerful built-in validation of markup
  - ✓ Helps us to avoid unwanted runtime exceptions due to errors in the view.
- The @model directive provides a cleaner and more concise way to define a strongly typed model.

Section 2

RAZOR SYNTAX

Razor syntax

- @ symbol
  - ✓ transition from HTML to C#
  - ✓ Example 1: <h2>Hi, my name is @name</h2>
  - ✓ Example 2: <p>@address</p>
- escape an @ symbol
  - ✓ use a second @ symbol
  - ✓ Example 3: <p>@@Username</p>

Razor syntax

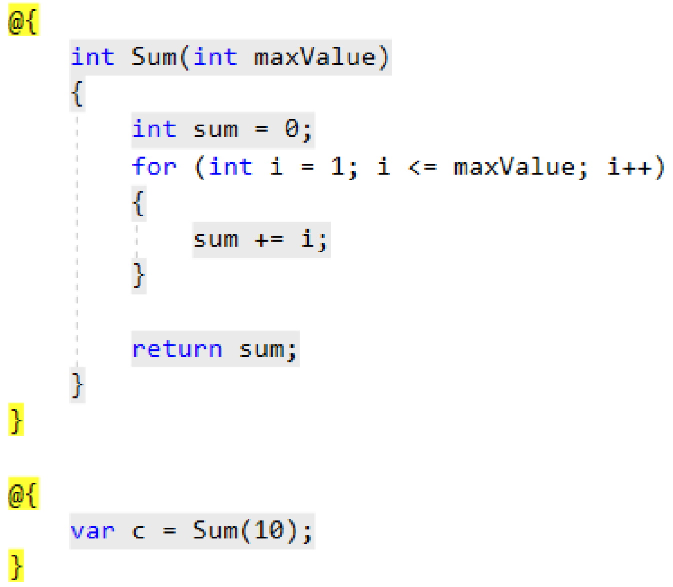
- HTML attributes and content containing email addresses don't treat the @ symbol as a transition character.
  - ✓ Example 4: <a href="mailto:Support@contoso.com">Support@contoso.com</a>

Explicit Razor expressions

- Explicit Razor expressions consist of an @ symbol with balanced parenthesis.
- Any content within the @() parenthesis is evaluated and rendered to the output.
  - ✓ Ex: <p>Last week: @DateTime.Now - TimeSpan.FromDays(7)</p>
- Explicit expressions can be used to concatenate text with an expression result
  - ✓ <input type="text" id="textbox@(index)" />

Razor code blocks

- Razor code blocks start with @ and are enclosed by {}.
- C# code inside code blocks isn't rendered.



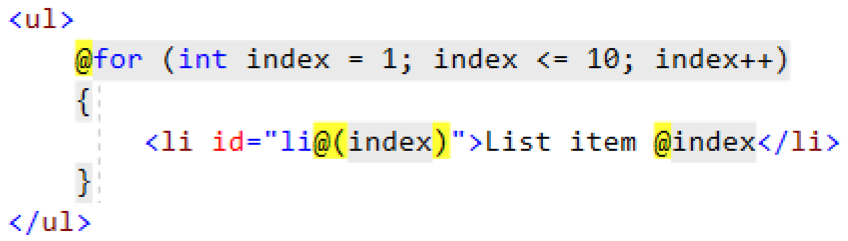
```
@{
    int Sum(int maxValue)
    {
        int sum = 0;
        for (int i = 1; i <= maxValue; i++)
        {
            sum += i;
        }

        return sum;
    }
}

@{
    var c = Sum(10);
}
```

Transition back to HTML

- The default language in a code block is C#
- Razor Page can transition back to HTML in 3 ways:
  - ✓ Implicit transitions
  - ✓ Explicit delimited transition
  - ✓ Explicit line transition

Transition back to HTML

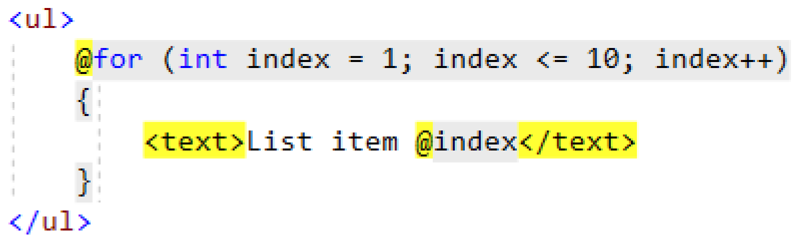
- Implicit transitions
  - ✓ By use HTML tag



```
<ul>
    @for (int index = 1; index <= 10; index++)
    {
        <li id="li@(index)">List item @index</li>
    }
</ul>
```

Transition back to HTML

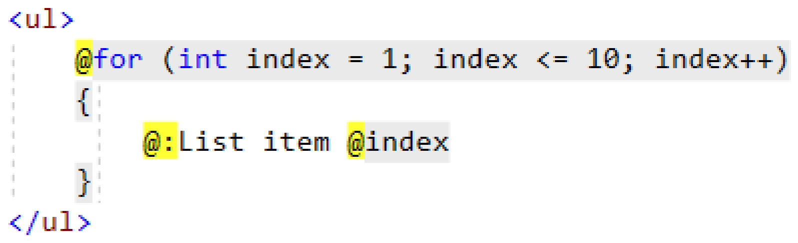
- Explicit delimited transition
  - ✓ surround the characters for rendering with the Razor <text> tag:



```
<ul>
    @for (int index = 1; index <= 10; index++)
    {
        <text>List item @index</text>
    }
</ul>
```

Transition back to HTML

- Explicit line transition
  - ✓ use the ‘@:’ syntax to render the rest of an entire line as HTML inside a code block,



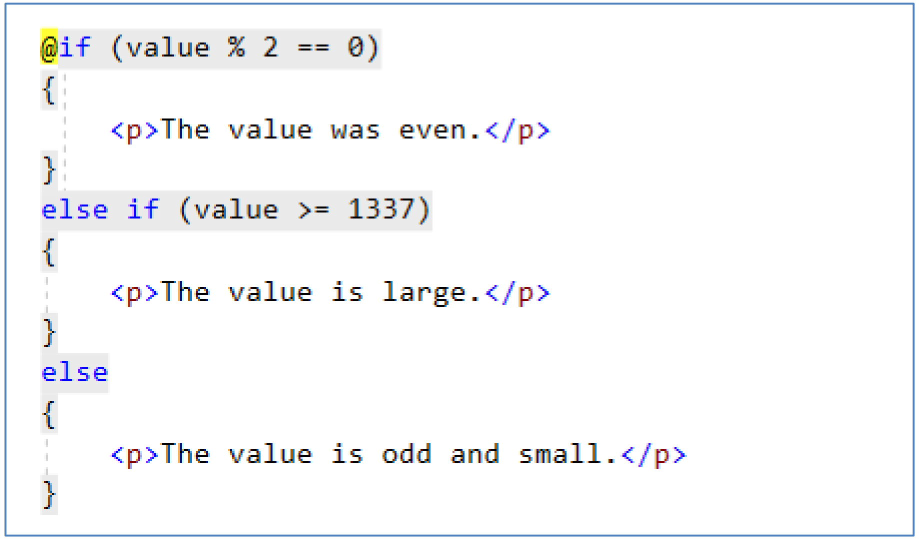
```
<ul>
    @for (int index = 1; index <= 10; index++)
    {
        @:List item @index
    }
</ul>
```

Section 3

CONTROL STRUCTURES

Conditionals

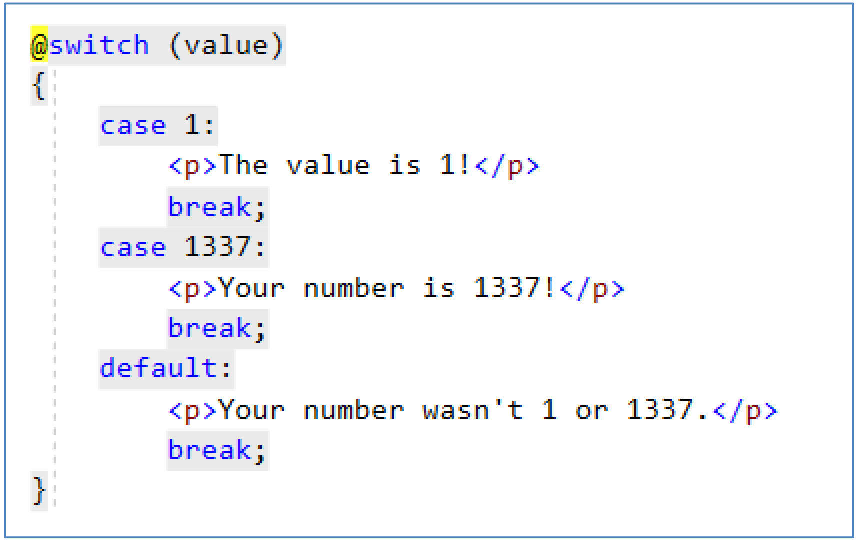
- **@if, else if, else**
  - ✓ Start block with **@if**
  - ✓ **else** and **else if** don't require the @ symbol



```
@if (value % 2 == 0)
{
    <p>The value was even.</p>
}
else if (value >= 1337)
{
    <p>The value is large.</p>
}
else
{
    <p>The value is odd and small.</p>
}
```

Conditionals

- **@switch**
  - ✓ Start block with **@switch**
  - ✓ **case, break, default** don't require the @ symbol



```
@switch (value)
{
    case 1:
        <p>The value is 1!</p>
        break;
    case 1337:
        <p>Your number is 1337!</p>
        break;
    default:
        <p>Your number wasn't 1 or 1337.</p>
        break;
}
```

Looping

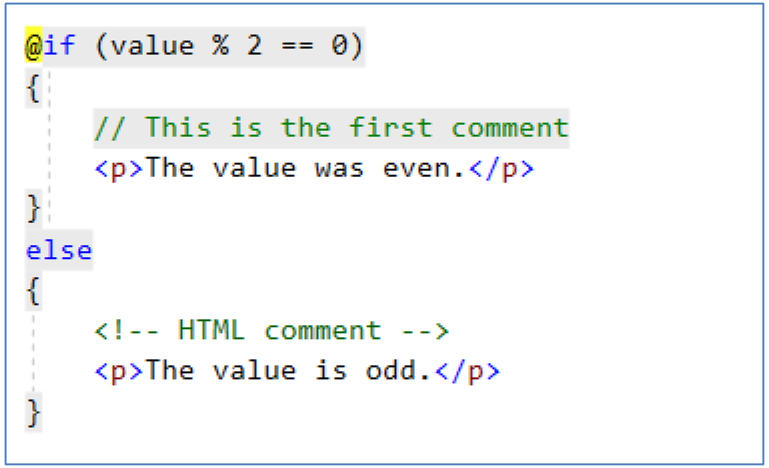
- @for,
- @foreach,
- @while,
- @do while

Handle exception

- **@try, catch, finally**

Comments

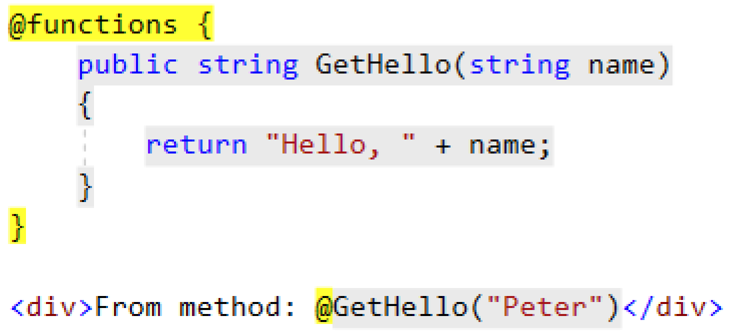
- Razor supports C# and HTML comments
- Razor comments are removed by the server before the webpage is rendered.



```
@if (value % 2 == 0)
{
    // This is the first comment
    <p>The value was even.</p>
}
else
{
    <!-- HTML comment -->
    <p>The value is odd.</p>
}
```

@function

- The **@functions** directive enables adding C# members (fields, properties, and methods)



```
@functions {
    public string GetHello(string name)
    {
        return "Hello, " + name;
    }
}

<div>From method: @GetHello("Peter")</div>
```

@model

- The @model directive specifies the type of the model passed to a view.
- The directive specifies the T in RazorPage<T> that the generated class that the view derives from.
- If the @model directive isn't specified, the Model property is of type dynamic.
- @model is used for Strongly typed mechanism

Thank you