Exploring Razor Syntax and Tag Helpers

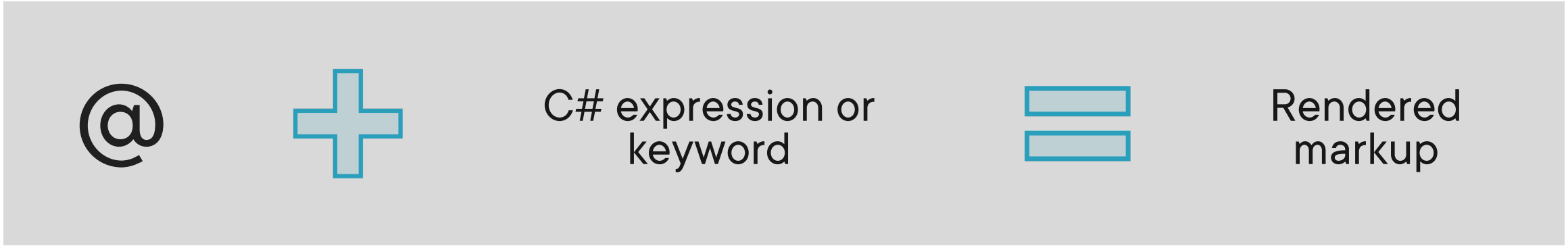
Alex Wolf
.NET Developer

www.thecodewolf.com

Razor

A concise templating syntax designed to generate dynamic markup.

The Anatomy of Razor Expressions



Razor Expressions (Implicit)

Razor Template

```
<p>
Today's date is
@DateTime.Now.Date.ToString("MM/dd/yyyy").
</p>

<p>
The temperature is @weather.Temp degrees.
</p>
```

Rendered HTML

```
<p>
Today's date is 3/31/2022.
</p>

<p>
The temperature is 60 degrees.
</p>
```

Razor Expressions (Explicit)

Razor Template

```
<p>
Last week this time was
@(DateTime.Now - TimeSpan.FromDays(7))
</p>

<p>
The total price is
@(item.Price += (item.Price * salesTax))
</p>
```

Rendered HTML

```
<p>
Last week this time was 3/24/2022.
</p>

<p>
The total price is 10.50.
</p>
```

Razor Code Blocks

Index.cshtml

```
<div>
    @{
        var item = new Product();
        item.Price = 4;
        item.Name = "Coffee";

        decimal salesTax = 0.6;
    }
    <p>
        The price is:
        @(item.Price += (item.Price * salesTax))
    </p>
</div>
```

Use Razor code blocks with caution.

Program Flow with Razor

Conditionals

```
@if (day == "Friday")
{
    <p>Today is payday!</p>
}
else
{
    <p>Today is a normal day.</p>
}
```

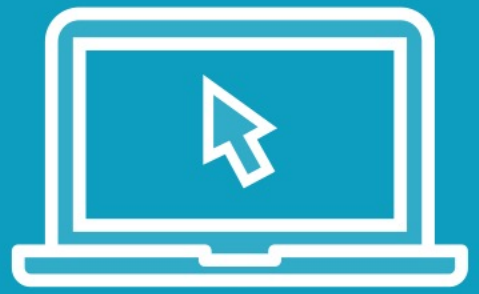
Loops

```
@foreach(var item in productList) {
    <div class="product">

        <p>@item.Name</p>

        <p>@item.Description</p>

        <p>@item.Price</p>

    </div>
}
```

Razor Directives

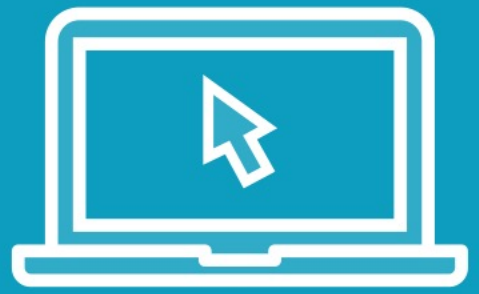
@layout = "_oneColumn.cshtml"	Assign a layout to the page
@page "/dashboard"	Marks the template as a routable Razor Page
@model AddProductModel.cs	Associates the Razor template with a C# code behind file
@using WiredBrain.Services	Pulls a C# namespace into the template
@inject IProductService	Injects services using Dependency Injection

Demo



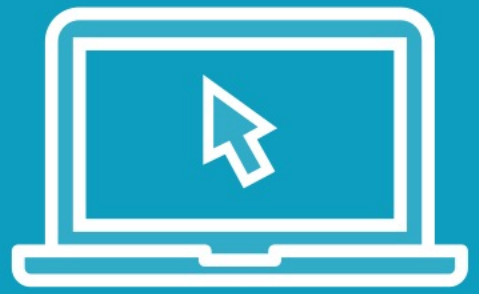
Getting comfortable with Razor syntax

Demo



Creating new Razor Pages

Demo



Adding contextual logic to markup with Razor

Understanding Tag Helpers

Tag Helpers

Reusable, semantically-friendly components that dynamically generate HTML elements

Tag Helper Capabilities

Generate links dynamically

Caching

Manage static assets

Environment features

Render other components

Build smarter forms

Applying Tag Helpers

Tag Helper attributes

Applied to standard HTML elements as attributes

Tag Helper elements

Replace or create new HTML elements

Tag Helpers with Matching HTML Elements

Anchor	**Image**
Script	**Form**
Input	**Label**
Select	**Textarea**

Applying Tag Helpers to Standard HTML Elements

Razor syntax

```
<a class="nav-link"
    asp-page="MenuItemDetails"
    asp-route-id="@Item.Id">
    Edit Item Details
</a>
```

Rendered HTML

```
<a class="nav-link" href="/menu/edit/23">
    Edit Item Details
</a>
```

Stand-alone Tag Helpers

Cache	**Distributed Cache**
Environment	**Partial**
Validation Message	**Validation Summary**

Applying Tag Helpers as New Elements

Razor syntax

```
<environment names="Development">
    <script src="bootstrap.css" />
</environment>

<environment names="Production">
    <script src="bootstrap.min.css" />
</environment>
```

Rendered HTML

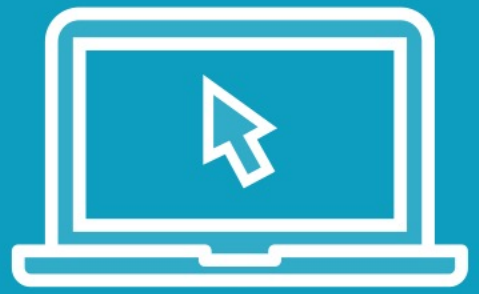
```
<!-- Renders locally -->
<script src="bootstrap.css" />

<!-- Renders in production -->
<script src="bootstrap.min.css" />
```

Semantically Friendly Syntax

Login.cshtml

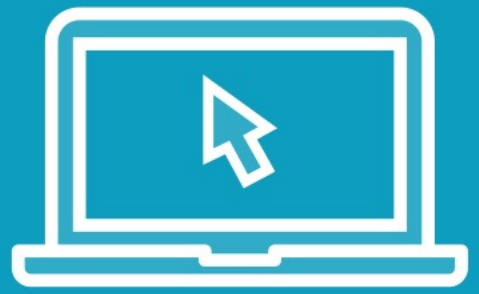
```
<form asp-page="Login" method="post">

    <label asp-for="Email">Email</label>
    <input asp-for="Email" />

    <label asp-for="Password">Password</label>
    <input asp-for="Password" />

    <button type="submit">Login</button>

</form>
```

Demo



Building the navigation using Tag Helpers

Demo



Enhancing markup using Tag Helpers

A Note About HTML Helpers

Example HTML Helpers

@Html.ActionLink("Create", "Product")	Creates an anchor tag
@Html.TextBoxFor(x => x.FirstName)	Creates a form text input
@Html.DisplayFor(x => x.Price)	Renders the value of Price
@Html.ValidationFor(x => x.Email)	Renders the email validation message

Comparing HTML Helpers and Tag Helpers

HTML Helpers

```
@Html.TextBoxFor(p => p.FirstName)

@Html.ActionLink("Home","Index")

<h1>@Html.Display("PageTitle")</h1>

@Html.ValidationSummary()
```

Tag Helpers

```
<input asp-for="Person.FirstName" />

<a asp-page="Index">Home</a>

<h1 asp-for="PageTitle"></h1>

<div asp-validation-summary="all"></div>
```

Tag Helpers are the recommended way forward.

Contact.Razor

```
<!-- Semantically friendly Tag Helper -->
<input asp-for="Contact.Email" class="form-control" placeholder="joe@email.com"/>


<!-- Semantically awkward HTML Helper -->
@Html.TextBoxFor(c => c.Email, new { @class="form-control",
                                     placeholder="joe@email.com" })
```

Summary

Razor syntax utilizes the @ sign and various expressions to dynamically render markup

Razor also provides full support for control flow statements like conditionals and loops

Razor can also define code blocks for writing more traditional C# logic across multiple lines

Tag Helpers dynamically generate markup by enhancing HTML elements

Tag Helpers can also introduce entirely new elements with new capabilities

Tag Helpers are very semantically friendly

HTML Helpers are a more verbose, legacy alternative to Tag Helpers