In the “Hello World” application in Chapter 2, Razor, C#, and ASP.NET Web Pages were used to perform relatively complex server-side tasks with ease. Now that you have a base knowledge of C#, either from experience or from the previous chapter, it is time to take a more in-depth look at Razor and ASP.NET Web Pages.

C#, Razor, and ASP.NET Web Pages combine perfectly to create an extremely powerful framework for creating sophisticated and dynamic web pages. In this chapter, some of the features of this framework that have been designed to provide a lightweight, fast, and scalable way to build web applications will be looked at. Topics covered in this chapter include:

- Razor syntax
- Layout
- Helpers
- Functions
- Maintaining state
- Uniform Resource Locators (URLs) and Routing

Razor Syntax

Razor is a simple to use, yet extremely powerful, programming syntax for inserting server-side code into web pages. When you create a new ASP.NET Web Page (.CSHTML or .VBHTML) in WebMatrix, you are creating a file that has two totally separate types of content: client-side and server-side. The client-side content of an ASP.NET Web Page can contain any of the usual code and markup that you would expect to see in a normal HTML page, including HTML, JavaScript, and CSS. The server-side content contains instructions to the web server in C# (or VB.NET), which are used to create dynamic content and interact with other resources, such as databases and file systems.

The server-side code is inserted directly into the page amongst the client-side content; use of the Razor syntax is your way of telling the web server where the client-side content ends and where the server-side content begins, and vice-versa.

The most common use of Razor is to dynamically create HTML and other client-side content. When IIS receives a request for a .CSHTML or .VBHTML page it recognizes it as an ASP.NET Web Page,
executes any server-side instructions marked with the Razor syntax, and then returns the generated page to the browser.

---

Note The Razor syntax was actually invented for WebMatrix. However, it proved to be so popular amongst developers and such a great improvement over previous syntaxes, that it has now also been adopted as the default view engine for ASP.NET MVC.

---

Adding a single line of server-side code

A single line of server-side code can be added to an ASP.NET Web Page using the @ symbol. The code is added in-line with the client-side content, such as HTML, etc., and is recognised and processed automatically by the web server, substituting the Razor code with relevant dynamic content wherever this is the intention, such as

```html
<p>The current date and time is: @DateTime.Now</p>
```

The parser has an intimate in-built knowledge of C# syntax, which makes it able to recognize where a C# statement ends and the HTML begins.

When a single-line expression is used in this way, the output is rendered directly to the page. If the single line of server-side code is there to perform some other function and is not intended for display, you can inform the parser by wrapping the statement in a pair or curly braces, thereby creating a statement block. This is seen in the following code

```csharp
@{ var PageTitle = "My Home Page"; }
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>@PageTitle</title>
</head>
<body>
    <h1>@PageTitle</h1>
    <p>The current date and time is: @DateTime.Now</p>
</body>
</html>
```

This mix of code and HTML markup would output the result, as seen in Figure 4-1.