Razor Pages与Razor语法

厦门大学信息学院 赵江声

2023-10

目录

Content

01

Razor Pages概述

简要介绍

1.1 什么是ASP.NET Razor页面

参考资料：https://learn.microsoft.com/zh-cn/training/modules/create-razor-pages-aspnet-core/2-why-when-use-razor-pages

Razor Pages 是一种以页面为中心的服务器端编程模型，用于通过 ASP.NET Core 构建 Web UI。

- Razor页面与ASP.NET MVC开发使用的视图组件非常相似，它们具有所有相同的语法和功能。
- 最关键的区别是模型和控制器代码也包含在Razor页面中。它更像是一个MVVM（Model-View-ViewModel）框架，它支持双向数据绑定，更简单的开发体验，具有独立的关注点。

```
@page
@model IndexModel
@using Microsoft.AspNetCore.Mvc.RazorPages

@functions {
    public class IndexModel : PageModel
    {
        public string Message { get; private set; } =
"In page model: ";

        public void OnGet()
        {
            Message += $" Server seconds
{ DateTime.Now.Second.ToString() }";
        }
    }
}

<h2>In page sample</h2>
<p>@Model.Message</p>
```

1.2 Razor Pages优点

- 使用 HTML、CSS 和 C# 轻松设置动态 Web 应用。
  - Razor Pages 使用 Razor 将基于服务器的代码嵌入到网页中。 Razor 语法结合了 HTML 和 C# 来定义动态呈现逻辑。
- 按功能整理文件，以便于维护。
- 使用 Razor 语法将标记与服务器端 C# 代码相结合。
- 关注点分离
  - Razor Pages 强制使用 C# PageModel 类分离关注点
- 通过 Razor Pages 对基于页面的场景编码比使用控制器和视图更轻松、更高效。

1.3 何时使用Razor Pages

在 ASP.NET Core 应用中使用 Razor Pages 的时机：

- 想要生成动态 Web UI。
- 首选以页面为中心的方法。
- 希望减少部分视图的重复。

02

Razor语法

2.1 基本概念

- 参考资料：https://learn.microsoft.com/zh-cn/aspnet/core/mvc/views/razor?view=aspnetcore-7.0

- Razor 是一种标记语法，用于将基于 .NET 的代码嵌入网页中。
- Razor 语法由 Razor 标记、C# 和 HTML 组成。
- 包含 Razor 的文件通常具有 .cshtml 文件扩展名。

2.2 语法

- Razor 支持 C#，并使用 @ 符号从 HTML 转换为 C#。
- Razor 计算 C# 表达式，并将它们呈现在 HTML 输出中。

```
<!-- 注意转义符 -->
@{
    //定义变量
    string Username = "myname";
}
<p>@Username </p>
<p>@@Username</p>
```

myname

@Username

2.2.1 隐式 Razor 表达式 & 显式 Razor 表达式

：

- 隐式 Razor 表达式
  - 隐式 Razor 表达式以 @ 开头，后跟 C# 代码
- 显式 Razor 表达式
  - 显式 Razor 表达式由 @ 符号和平衡圆括号组成。

```
<!-- 隐式razor表达式 ： 隐式 Razor 表达式以 @@ 开头，后跟 C# 代码 ，注意没有平衡圆括号-->
<p>@DateTime.Now</p>
<p>@DateTime.IsLeapYear(2016)</p>

<!-- 显式razor表达式 ：  显式 Razor 表达式由 @@ 符号和平衡圆括号组成。-->
<p>Last week this time: @(DateTime.Now - TimeSpan.FromDays(7))</p>
```

```
2023/10/11 10:36:39

True

Last week this time: 2023/10/4 10:36:39
```

2.2.2 表达式编码

- 计算结果为字符串的 C# 表达式采用 HTML 编码。

```
<!-- 表达式编码 ： 计算结果为字符串的 C# 表达式采用 HTML 编码。-->
@("<span>Hello World</span>")
<br/><span>Hello World</span>
```

```
<span>Hello World</span>
Hello World
```

2.2.3 Razor 代码块

- Razor 代码块以 @ 开始，并括在 {} 中。
- 代码块内的 C# 代码不会呈现，这点与表达式不同。

```
<!-- razor代码块 : Razor 代码块以 @@ 开始，并括在 {} 中 。-->
@{
    var quote = "The future depends on what you do today. - Mahatma Gandhi";
}
<p>@quote</p>
@{
    quote = "Hate cannot drive out hate, only love can do that. - Martin Luther King, Jr.";
}
<p>@quote</p>

<!-- 在代码块中，使用标记将本地函数声明为用作模板化方法。-->
@{
    void RenderName(string name)
    {
        <p>Name: <strong>@name</strong></p>
    }

    RenderName("Mahatma Gandhi");
    RenderName("Martin Luther King, Jr.");
}
```

2.2.4 隐式转换 & 带分隔符的显式转换

- 隐式转换：代码块中的默认语言是 C#，但 Razor 页面可以转换回 HTML；
- 显式转换：若要定义应呈现 HTML 的代码块子节，请使用 Razor<text> 标记将要呈现的字符括起来

```
<!-- 显式转换：若要定义应呈现 HTML 的代码块子节，
请使用 Razor<text> 标记将要呈现的字符括起来。-->
@{
    //Person的定义在Index.cshtml.cs
    var Persons= new Person[]
    {
        new Person("Weston", 33),
        new Person("Johnathon", 41)
    };
}
@for (var i = 0; i < Persons.Length; i++)
{
    var person = Persons[i];
    <text>Name: @person.Name</text><br/>
}
```

```
<!-- 隐式转换：代码块中的默认语言是 C#，但 Razor 页面可以转换回 HTML。-->
@{
    var inCSharp = true;
    <p>Now in HTML, was in C# @inCSharp</p>
}
```

2.2.5 条件属性呈现

- Razor 自动省略不需要的属性。 如果传入的值为 null 或 false，则不会呈现属性。

```
<!-- 条件属性呈现：Razor 自动省略不需要的属性。 如果传入的值为 null 或 false，则不会呈现属性。-->
<div class="@false">False</div>
<div class="@null">Null</div>
<div class="@("")">Empty</div>
<div class="@("false")">False String</div>
<div class="@("active")">String</div>
<input type="checkbox" checked="@true" name="true" />
<input type="checkbox" checked="@false" name="false" />
<input type="checkbox" checked="@null" name="null" />
```

False
Null
Empty
False String
String

```
<div>False</div>
<div>Null</div>
<div class="">Empty</div>
<div class="false">False String</div>
<div class="active">String</div>
<input type="checkbox" checked="checked" name="true">
<input type="checkbox" name="false">
<input type="checkbox" name="null">
```

2.2.6 控制结构

- 代码块的各个方面（转换为标记、内联 C#）同样适用于以下结构：
  - 条件 @if, else if, else, and @switch
  - 循环 @for, @foreach, @while, and @do while
  - 复合语句 @using
  - @try, catch, finally

```
<!--循环-->
@{
    var people = new Person[]
    {
        new Person("Weston", 33),
        new Person("Johnathon", 41)

    };

}
@foreach (var person in people)
{
    <p>Name: @person.Name -- Age: @person.Age</p>
}
```

```
<!--
    控制结构:是对代码块的扩展。 代码块的各个方面
    （转换为标记、内联 C#）同样适用于以下结构:
-->
@{
    int value = 1401;
}
@if (value % 2 == 0)
{
    <p>The value was even.</p>
}

@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>
}
```

2.2.7 注释

```
<!-- 注释 -->
@{
    /* C# comment */
    // Another C# comment
}
<!-- HTML comment -->
```

03

Razor Pages UI开发

案例

3.1 资料

- 参考资料：https://learn.microsoft.com/zh-cn/training/modules/create-razor-pages-aspnet-core/1-introduction

- 在已下载的项目的基础上继续开发。

ContosoPizza
源码fromgithu

3.2 步骤（1）

- 查看 Razor Pages 项目结构；

名称	说明
Pages/	包含 Razor Pages 和支持文件。 每个 Razor 页面都有一个 .cshtml 文件和一个 .cshtml.csPageModel类文件。
wwwroot/	包含静态资产文件，例如 HTML、JavaScript 和 CSS。
ContosoPizza.csproj	包含项目配置元数据，例如依赖项。
Program.cs	充当应用的入口点并配置应用行为，例如路由。

3.2 步骤（2）

- 修改首页；
- 添加新的 Razor 页面；
- 添加新的披萨表单；

谢谢！