Blazor

Module 3: Components and Layouts

Student Lab Manual

Instructor Edition (Book Title Hidden Style)

**Conditions and Terms of Use**

**Microsoft Confidential**

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2020 Microsoft Corporation. All rights reserved.

**Copyright and Trademarks**

© 2020 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at  
<http://www.microsoft.com/en-us/legal/intellectualproperty/Permissions/default.aspx>

Internet Explorer, Microsoft, SQL Server, Visual Studio, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Contents

[Lab 3: Components and Layouts 1](#_Toc29729390)

[Exercise 1: Components and Layout 3](#_Toc29729391)

# Lab 3: Components and Layouts

#### Introduction

The aim of this exercise is to explore implementing an application using the new application model introduced under ASP.Net Core 3 called Blazor.

Important

**Blazor WebAssembly in preview**

Blazor Server is supported in ASP.NET Core 3.0. Blazor WebAssembly is in preview for ASP.NET Core 3.1.

#### Objectives

After completing this lab, you will be able to:

* Create a Blazor Component
* Create a Blazor Layout

#### Prerequisites

None

#### Scenario

In this scenario, we will explore implementing an online Pizza Delivery application.

#### System Requirements

* Follow the instructions found [on this page](https://docs.microsoft.com/en-us/aspnet/core/blazor/get-started?view=aspnetcore-3.1&tabs=visual-studio) to get started

#### Estimated Time to Complete This Lab

30 minutes

Exercise 1: Components and Layout

#### Objectives

In this exercise, you will:

* Create a Blazor Component
* Create a Layout Component

#### Scenario

You will get started with building a pizza store app using Blazor. You will add a component that displays the list of available pizza specials. You will also introduce a layout component to define a top bar with a branding logo and a nav link for the home page.

Task 1: Create a Blazor Component

1. Start by opening the solution file **BlazingPizza.sln** located under **\Labs\Module03 - Components and Layouts\Begin**.

The solution already contains four projects:

A screenshot of a cell phone

Description automatically generated

* **BlazingPizza.Client**: This is the Blazor project. It contains the UI components for the app.
* **BlazingPizza.Server**: This is the ASP.NET Core project hosting the Blazor app and also the backend services for the app.
* **BlazingPizza.Shared**: Shared model types for the app.
* **BlazingPizza.ComponentsLibrary**: A library of components and helper code to be used by the app in later sessions.

The **BlazingPizza.Server** project should be set as the startup project.

When you run the app, you'll see that it currently only contains a simple home page.

A screenshot of a computer

Description automatically generated

Open Pages/Index.razor in the **BlazingPizza.Client** project to see the code for the home page.

@page "/"

<h1>Blazing Pizzas</h1>

The home page is implemented as a single component. The @page directive specifies that the Index component is a routable page with the specified route.

1. Next, we'll update the home page to display the list of available pizza specials. The list of specials will be part of the state of the Index component.

Add a @code block to Index.razor with a list field to keep track of the available specials:

@code {

List<PizzaSpecial> specials;

}

The code in the @code block is added to the generated class for the component. The PizzaSpecial type is already defined for you in the Shared project.

1. To get the available list of specials we need to call an API on the backend. Blazor provides a preconfigured HttpClient through dependency injection that is already setup with the correct base address. Use the @inject directive to inject an HttpClient into the Index component.

@page "/"

@inject HttpClient HttpClient

The @inject directive essentially defines a new property on the component where the first token specified the property type, and the second token specifies the property name. The property is populated for you using dependency injection.

1. Override the OnInitializedAsync method in the @code block to retrieve the list of pizza specials. This method is part of the component lifecycle and is called when the component is initialized. Use the GetFromJsonAsync<T>() method to handle deserializing the response JSON:

@code {

List<PizzaSpecial> specials;

protected override async Task OnInitializedAsync()

{

specials = await HttpClient.GetFromJsonAsync<List<PizzaSpecial>>("specials");

}

}

The /specials API is defined by the SpecialsController in the Server project.

1. Once the component is initialized it will render its markup. Replace the markup in the Index component with the following to list the pizza specials:

<div class="main">

<ul class="pizza-cards">

@if (specials != null)

{

@foreach (var special in specials)

{

<li style="background-image: url('@special.ImageUrl')">

<div class="pizza-info">

<span class="title">@special.Name</span>

@special.Description

<span class="price">@special.GetFormattedBasePrice()</span>

</div>

</li>

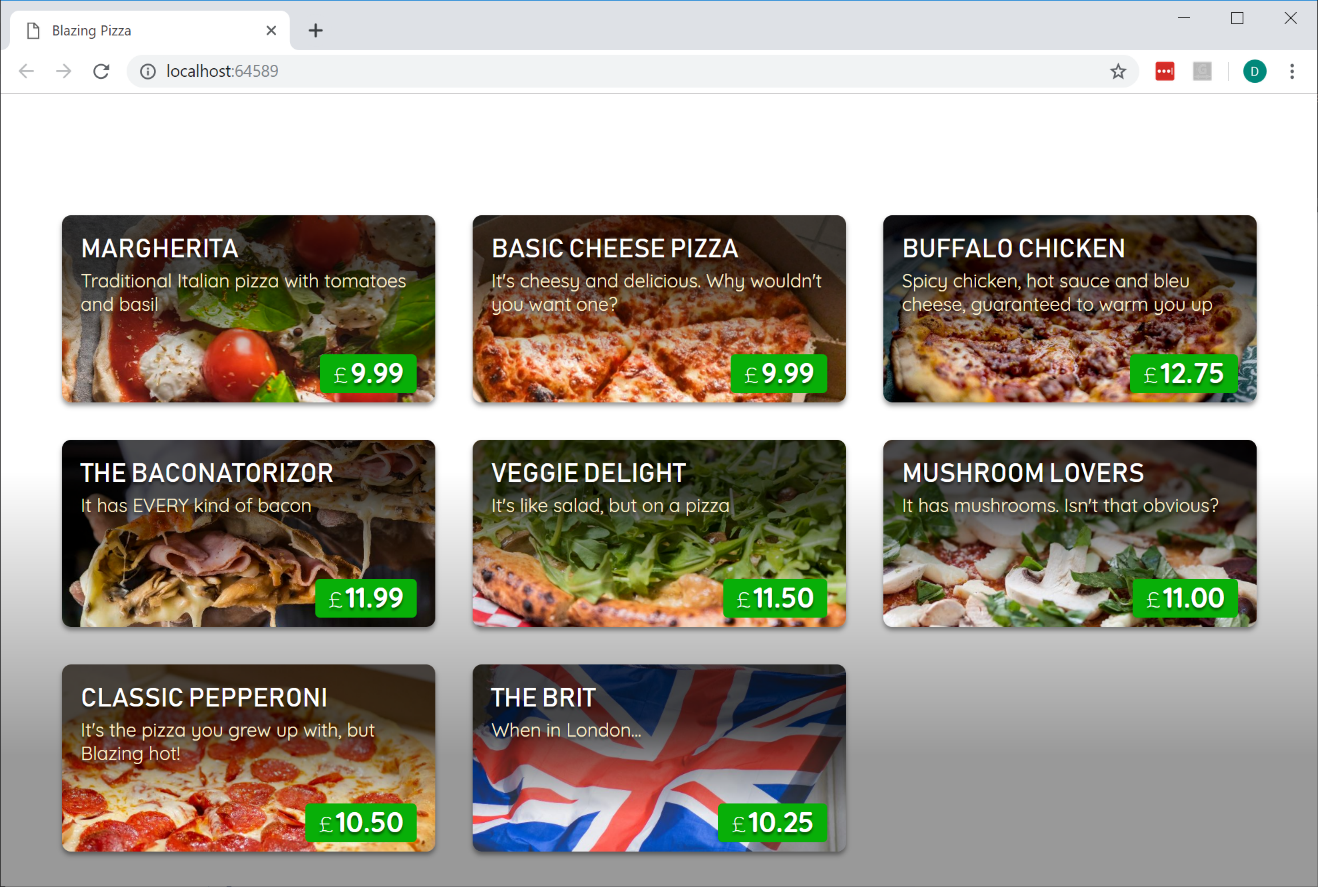
}

}

</ul>

</div>

1. Run the app by pressing Ctrl+F5. You should see a list of the specials available.

[](https://user-images.githubusercontent.com/1874516/57006743-1602e400-6b98-11e9-96cb-ff4829cf459f.png)

Task 2: Create a Layout Component

1. Next, we'll set up the layout for app. Layouts in Blazor are also components. They inherit from LayoutComponentBase, which defines a @Body property that can be used to specify where the body of the layout should be rendered. The layout component for our pizza store app is defined in **Shared/MainLayout.razor** under **BlazingPizza.Client** project.

@inherits LayoutComponentBase

<div class="content">

@Body

</div>

1. To see how the layout is associated with your pages, look at the <Router> component in App.razor. Notice the DefaultLayout parameter which determines the layout used for any page that doesn't specify its own layout directly. You can also override this DefaultLayout on a per-page basis. To do so, you can add directive such as @layout SomeOtherLayout at the top of any \*.razor page component. However, you do not need to do so in this application.
2. Update the MainLayout component to define a top bar with a branding logo and a nav link for the home page:

@inherits LayoutComponentBase

<div class="top-bar">

<img class="logo" src="img/logo.svg" />

<NavLink href="" class="nav-tab" Match="NavLinkMatch.All">

<img src="img/pizza-slice.svg" />

<div>Get Pizza</div>

</NavLink>

</div>

<div class="content">

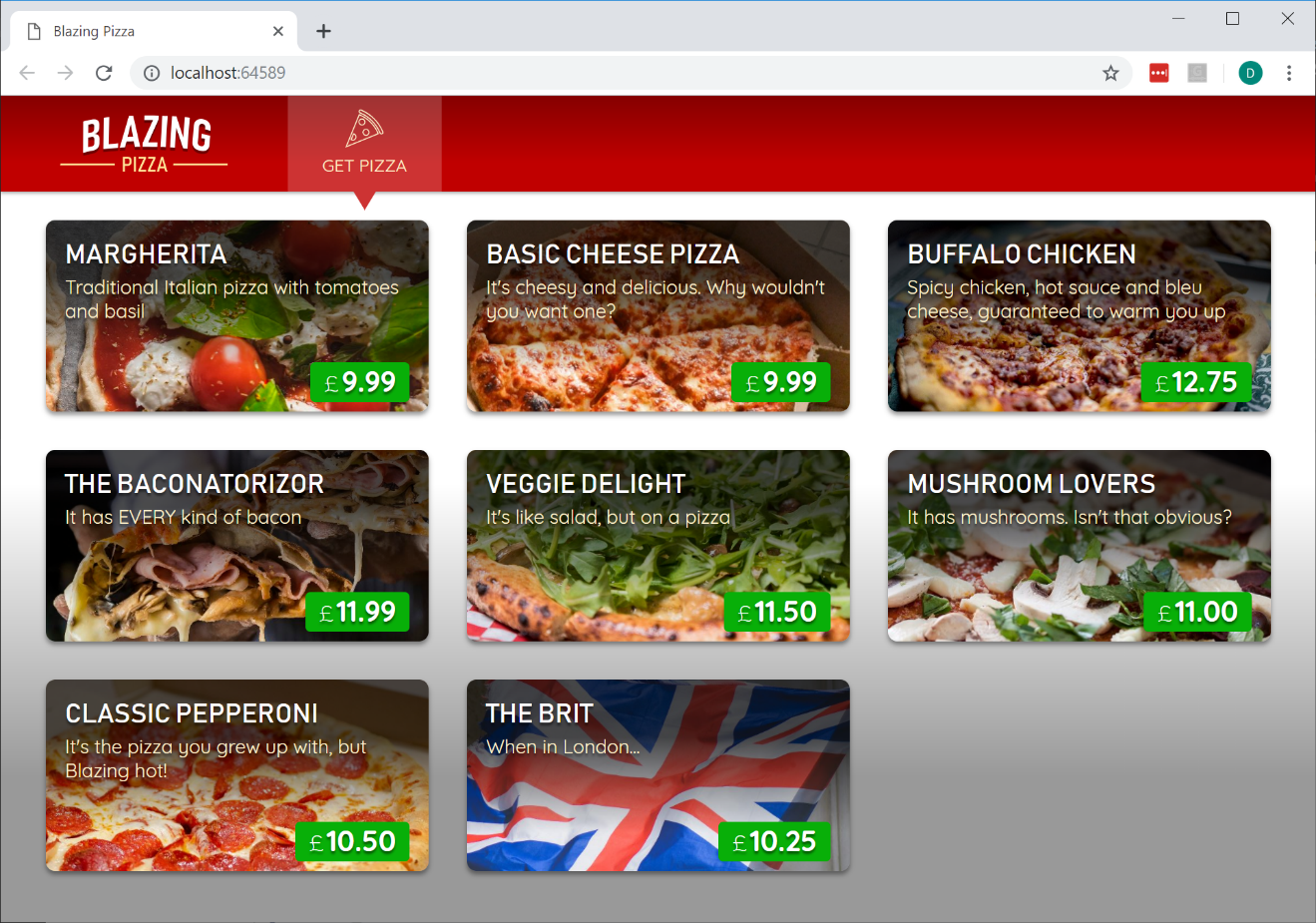
@Body

</div>

The NavLink component is provided by Blazor. Components can be used from other components, which is done by specifying an element with the component's type name along with attributes for any component parameters.

The NavLink component is the same as an anchor tag, except that it adds an active class if the current URL matches the link address. NavLinkMatch.All means that the link should be active only when it matches the entire current URL (not just a prefix). We will examine the NavLink component in more detail in a later session.

1. Run the app by pressing Ctrl+F5. With our new layout our pizza store app now looks like this:

[](https://user-images.githubusercontent.com/1874516/57006730-e81d9f80-6b97-11e9-813d-9c35b62efa53.png)