# COIT 12204, T2 2020

# Assessment 2 Guide

### Introduction

* This guide is a simplified process for adding the side navigation and displaying the products
* As the product grid layout is a little more complex than the tutorials, I am walking you through it

Ensure you have followed the Database guide to add the database to the project

### Adding the Side Navigation

First, we need to create our navigation ***View Component***

* Create the ***Components*** folder in the project root
* Add the ***NavigationMenuViewComponent.cs*** file

**Note**: When we insert this component into our view, Razor drops the ViewComponent.cs part of the filename and identifies the component as ***navigation-menu***

<vc:navigation-menu />

### Navigation Component

In the nav component we need to access the route parameters. We can do this outside the controller, by using the ***RouteData*** object

RouteData?.Values["Parameter-name"];

Add the following code to the ***NavigationMenuViewComponent.cs*** file

using Microsoft.AspNetCore.Mvc;

using System.Linq;

using KoalaBeach.Models;

namespace KoalaBeach.Components

{

public class NavigationMenuViewComponent : ViewComponent

{

private IStoreRepository repository;

public NavigationMenuViewComponent(IStoreRepository repo)

{

repository = repo;

}

public IViewComponentResult Invoke()

{

ViewBag.SelectedCategory = RouteData?.Values["category"];

//string type = RouteData?.Values["type"].ToString();

return View(repository.Products

//.Where(x => x.SubCategory == type)

.Select(x => x.Category)

.Distinct()

.OrderBy(x => x));

}

}

}

The Invoke method is called when the component is encountered in the Razor script. This method creates the component as a view and returns it to the parent view

There are two options for the menu

* One is to assume we have common categories for all genders such as beach wear. This is what I am doing in the code above
* The other is to filter each gender for specific gender-based categories such as bikinis. If you wish to do this, then uncomment the extra query parameter. You would also need to handle Sale which may simply just not filter by gender

### Navigation Component View

Next create the ***Razor View*** for the component

* Create the folder ***Views/Shared/Components/NavigationMenu***
* Create the component’s ***Razor View*** file ***Default.cshtml***
* Add the following code to the file

@model IEnumerable<string>

<a class="btn btn-block btn-outline-secondary" **asp-action**="Index"

**asp-controller**="Home" **asp-route-category**="">

Home

</a>

@foreach (string category in Model)

{

<a class="btn btn-block

@(category == ViewBag.SelectedCategory

? "btn-primary": "btn-outline-secondary")"

**asp-action**="Catalog" **asp-controller**="Home"

**asp-route-type**="@ViewBag.Type"

**asp-route-category**="@category">

@category

</a>

}

This code is the same as the book with the page number removed and an extra parameter in the route for the subcategory (type).

### Tag Helper

Next, we have to enable tag helpers for the project as we will be using the tag helper format to add the component to the view. All this is doing is to allow us to insert the menu using the <**vc:navigation-menu** /> tag

Open ***Views/Shared/\_ViewImports.cshtml*** and add the line below

@addTagHelper \*, KoalaBeach

### Catalog Page

Then we need to begin reconstructing the Catalog page

* Open ***Views/Home/Catalog.cshtml*** file
* Replace ***ALL*** code from the ***first div*** with the code below (leave just the razor code for title at the top)

@model IEnumerable<Product>

<div class="container-fluid">

<div class="row flex-column flex-md-row">

<div id="categories" class="col-3">

<**vc:navigation-menu** />

</div>

<div class="col-9">

Products go here

</div>

</div>

</div>

### Test the Side Nav

* Run the project and ensure the side nav displays on the Catalog page

### Product List

Next, we need to replace the ***Products go here*** comment above with the code between the divs below to add in our grid of products (do not copy the div tags)

<div class="col-9">

@{

int col = 0;

foreach (var p in Model)

{

if (col == 0)

{

@:<div class="row">

}

<div class="col-3">

<div class="card bg-white">

<img class="card-img-top" src="/images/cap.png" alt="Card image">

<div class="card-body text-center">

<p class="card-title"> Product Name </p>

<p class="card-text">Some text inside the first card</p>

<!-- Add to cart button goes here -->

</div>

</div>

</div>

if (col == 3 || col == Model.Count() - 1)

{

@:</div><br />

}

col = ++col % 4;

}

}

</div>

You will now need to use the product ***p*** in the ***foreach*** to populate the name, description and price for each product. I will leave that for you to do. Leave the image as cap for now unless you want to drop some more images in the images folder

### Controller

Next, we need to update the controller Catalog method to fetch the filtered product list and pass it to the view

* Open ***Controllers/HomeController.cs***
* Update the ***Catalog*** method for the route containing ***type***

[Route("/Catalog/{type}")]

[Route("/Catalog/{type}/{category}")]

public ViewResult Catalog(String type, string category)

{

ViewBag.Type = type; // men, women, sale

ViewBag.Category = category; // shirts, caps, shoes etc

IEnumerable<Product> products = repository.Products

.Where(p => (category == null || p.Category == category) && p.SubCategory == type)

.OrderBy(p => p.ProductID);

return View(products);

}

Note that we can define the routes either here, or in Startup.cs as we did in the lab. As our needs are simpler (and this method is clearer), we will define the route here (above the method call in the square brackets). Remember this is how we did it in assignment 1 to pass a variable in the route for the page type.

### Test the Catalog Page

***Run the project*** and you should see the product card deck for each. The seed data only had two products in men so it will be limited. You will need to put some more products in the database to test it fully.

## Shopping Cart

When you have the nav and products done, follow the tutorial from the section ***Building the Shopping Cart*** in the ***SportsStore: Navigation and Cart*** chapter of the updated text.

I will not be providing a detailed guide from this point on so you will be following the textbook once you have added the cart button.

**Note**

* Most files require you to change ***SportsStore*** to ***KoalaBeach*** (or your project namespace)
  + If your project will not run check the error messages at the bottom of the IDE as you may have missed one
* Do ***NOT*** run the project after **Listing 8.17** as it says. Wait till after ***Listing 8.18***.
  + This is simply because we have nothing in the Infrastructure namespace yet
  + which is because we did a few things differently
* Don’t miss Listing 8.19 and make sure you put it in the imports file in ***Views*** not Pages

Use the code below for the ***Add to Cart*** button in Listing 8.20. If you are using a partial view you will have to check the book as the partial view only deals with a single product not the list in a foreach loop

* Do this in place of the instructions and code in ***Listing 8.20*** (as we are not using a partial view)
* Open ***Views/Home/Catalog.cshtml***
* Replace the ***Add to cart button goes here*** comment with the following code for the add to cart button

<form id="@p.ProductID" **asp-page**="/Cart" method="post">

<input type="hidden" name="ProductID" value="@p.ProductID" />

<input type="hidden" name="returnUrl"

value="@ViewContext.HttpContext.Request.PathAndQuery()" />

<span class="card-text">

<button type="submit"

class="btn btn-success btn-sm">

Add To Cart

</button>

</span>

</form>

### Check the Add to Cart Button

You can run the project to check the add to cart button. However, the button will not do anything yet.

### Update the Bootstrap Path

Before you hit run at the end of the chapter, go to the cart layout and correct the path to your bootstrap if it is incorrect. For the sample solution the path is shown below

* Open ***Pages/ \_CartLayout.cshtml***
* Update the Bootstrap path to match the path used in your project i.e.

<link href="/lib/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet" />

### Test the Add to Cart

***Run the project*** and you should see the Add to cart button. Select the button and items should be added to the cart and the cart page displayed as it did in the tutorial.

* Note that you may now take some of the code out into a partial view if you wish (but not required)