What is ASP.Net MVC

What is MVC

* Architectural pattern that separates an app into 3 parts

1. Models : classes that represents the data of an app. Uses Validation logic to enforce business rules for the data. Typically model object retrieve and store model state in database.
2. Views : display the app UI
3. Controller : Handles the browser request. Retrieve model and data and call view templates that returns the response.

Adv

1. Separate different aspects of the app
2. Provide loosely coupling between these elements.
3. Helps to manage the complexity

Improved the testability of code.

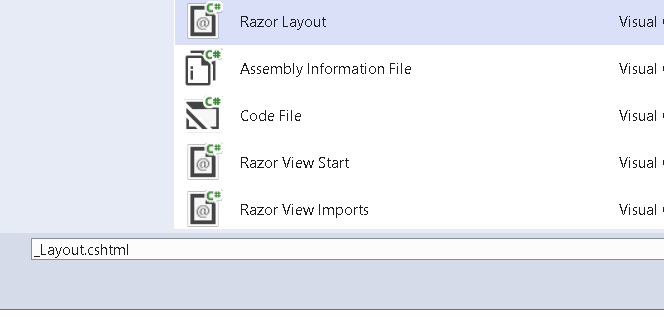
Explain How to Enable ASP.Net Controller and View Parts in the Web application

Layout

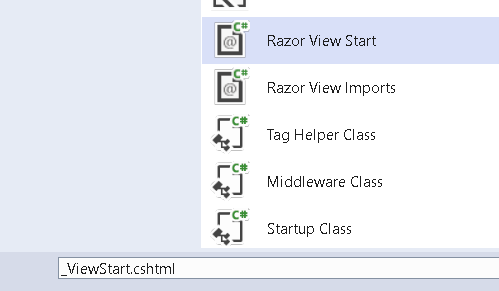
* Consistent look and feel
* RenderBody :inserts contents of other views.

Demo : How to add Layout

1. Add Shared folder in Views folder , add new Item with Razor Layout option



1. To Apply view to All the View in the web application, add \_viewstart.cshtml



1. Download Bootstrap from the following links and Put files in wwwroot

<https://getbootstrap.com/docs/4.0/getting-started/introduction/>

1. Change the Layout code

<body>

<div class="bg-dark text-white p-2">

<span class="navbar-brand ml-2">PAINTING STORE</span>

</div>

<div class="row m-1 p-1">

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

Put something useful here later

</div>

<div class="col-9">

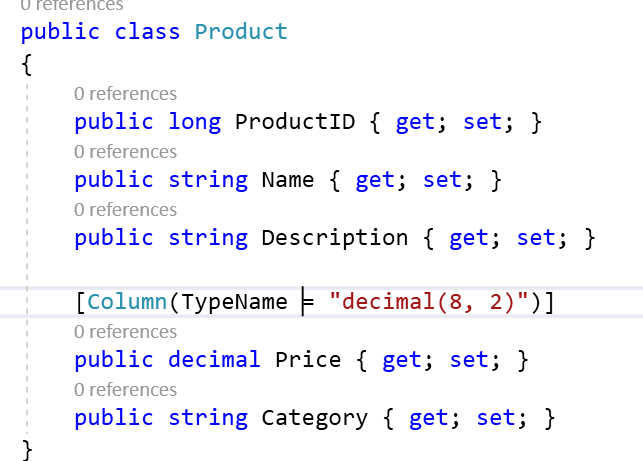
@RenderBody()

</div>

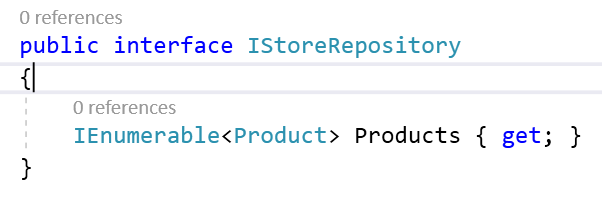
</div>

</body>

1. Add Models Folder and Class Product



1. Add IstoreRepository in models folder



1. Add ProductInMemoryRepository class in model and add some mock data

public class ProductsInMemoryRepository : IStoreRepository

{

static List<Product> products = new List<Product>

{

new Product

{

ProductID=1,Name="Chess Board",Category="Chess"

,Description="Brown Chess Board"

,Price=20.3m

},

new Product

{

ProductID=10,Name="Chess Board Black",Category="Chess"

,Description="Black Chess Board"

,Price=20.3m

},

//https://www.google.com/url?sa=i&url=https%3A%2F%2Fshopee.ph%2F-Filtered-Dreams-Printed-Artwork-i.2494212.2403709528&psig=AOvVaw1Ab6fdkXluRq6n7ijqywO1&ust=1613241650272000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLj8o87\_5O4CFQAAAAAdAAAAABAd

new Product

{

ProductID=2,Name="Chess Notes",Category="Chess",Description="Chess Notes",Price=210.3m

},

new Product

{

//https://www.google.com/url?sa=i&url=https%3A%2F%2Froyalthaiart.com%2Fproduct%2Fpeacock-artwork%2F&psig=AOvVaw1Ab6fdkXluRq6n7ijqywO1&ust=1613241650272000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLj8o87\_5O4CFQAAAAAdAAAAABAO

ProductID=3,

Name="Chess Timer",

Category="Chess"

,Description="Chess Timer "

,Price=10.3m

},

new Product

{

ProductID=4,Name="Ball",Category="Cricket"

,Description="Leather Ball"

,Price=20.3m

},

new Product

{

ProductID=15,Name="Ball Tennis",Category="Cricket"

,Description="TesnnisBall"

,Price=20.3m

},

//https://www.google.com/url?sa=i&url=https%3A%2F%2Fshopee.ph%2F-Filtered-Dreams-Printed-Artwork-i.2494212.2403709528&psig=AOvVaw1Ab6fdkXluRq6n7ijqywO1&ust=1613241650272000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLj8o87\_5O4CFQAAAAAdAAAAABAd

new Product

{

ProductID=5,Name="Bat",

Category="Cricket",Description="Bat with Sticket",Price=210.3m

},

new Product

{

//https://www.google.com/url?sa=i&url=https%3A%2F%2Froyalthaiart.com%2Fproduct%2Fpeacock-artwork%2F&psig=AOvVaw1Ab6fdkXluRq6n7ijqywO1&ust=1613241650272000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLj8o87\_5O4CFQAAAAAdAAAAABAO

ProductID=6,

Name="Stumps",

Category="Cricket"

,Description="3 Stumps Along with bails"

,Price=10.3m

},

new Product

{

ProductID=7,Name="Soccer Ball",Category="Soccer"

,Description="Soccer Ball"

,Price=20.3m

},

//https://www.google.com/url?sa=i&url=https%3A%2F%2Fshopee.ph%2F-Filtered-Dreams-Printed-Artwork-i.2494212.2403709528&psig=AOvVaw1Ab6fdkXluRq6n7ijqywO1&ust=1613241650272000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLj8o87\_5O4CFQAAAAAdAAAAABAd

new Product

{

ProductID=8,Name="Soccer Cards",

Category="Soccer",Description="Soccer Cards",Price=210.3m

},

new Product

{

//https://www.google.com/url?sa=i&url=https%3A%2F%2Froyalthaiart.com%2Fproduct%2Fpeacock-artwork%2F&psig=AOvVaw1Ab6fdkXluRq6n7ijqywO1&ust=1613241650272000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCLj8o87\_5O4CFQAAAAAdAAAAABAO

ProductID=9,

Name="Net",

Category="Soccer"

,Description="Soccer Net"

,Price=10.3m

},

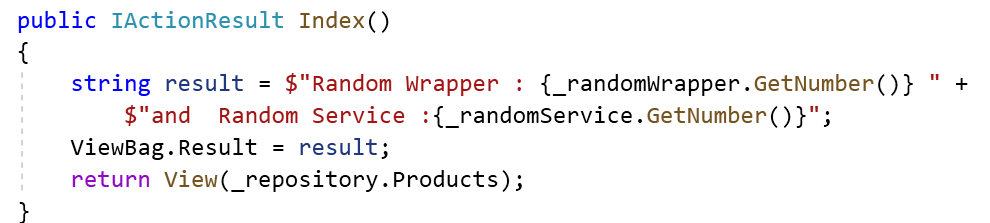
};

IEnumerable<Product> IStoreRepository.Products => products;

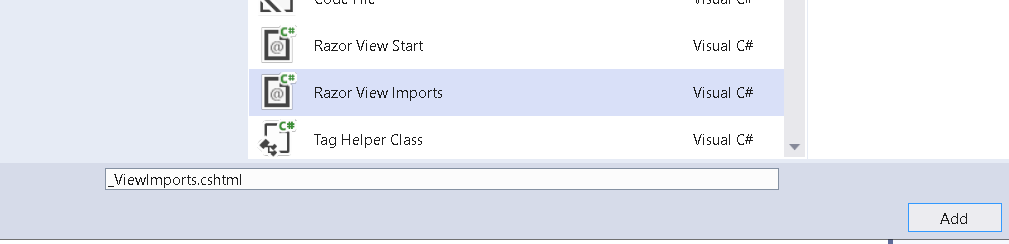
1. Add into Services collection in startup.cs

services.AddScoped<IStoreRepository, ProductsInMemoryRepository>();

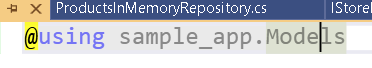
1. Add Code to in HomeController to get Products



1. Add ViewImport file in Views folder



1. Add Code to make available models within the application



1. Add code in Home Index.cshtml to display data

@model IEnumerable<Product>

@foreach (var p in Model)

{

<div class="card card-outline-primary m-1 p-1">

<div class="bg-faded p-1">

<h4>

@p.Name

<span class="badge badge-pill badge-primary" style="float:right">

<small>@p.Price.ToString("c")</small>

</span>

</h4>

</div>

<div class="card-text p-1">@p.Description</div>

</div>

}

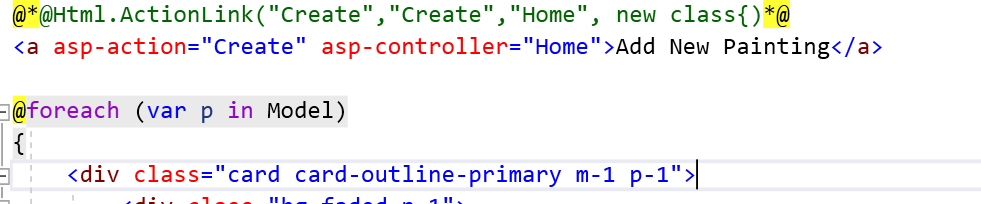
* Tag Helpers enable server-side code to participate in creating and rendering HTML elements in Razor files.

What does it provides

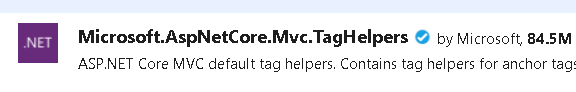
* HTML friendly development experience for most part of Razor markup.
* Front end designed who know HTML css and JS can edit Razor without learning C# razor syntax.
* Asp-action
* Asp-controller

Demo TagHelper

1. Add asp-action and asp-controller link in Index of HomeController



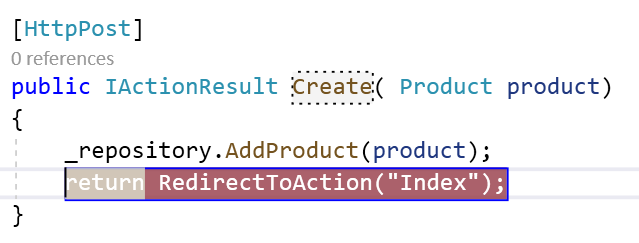
1. We need to add library in order to make this link work



1. Make Available this TagHelpers to the view



1. Add Action Method with name Create



1. Add one more View Create.cshtml

@model sample\_app.Models.Product

<h3>Add New Painting</h3>

<form **asp-action**="Create" **asp-controller**="Home">

<div class="form-group">

<label **asp-for**="@Model.ProductID">ID</label>

<input **type**="text" **asp-for**="@Model.ProductID" class="form-control" placeholder="ID">

</div>

<div class="form-group">

<label **asp-for**="@Model.Name">Name</label>

<input **type**="text" **asp-for**="@Model.Name" class="form-control" placeholder="Name">

</div>

<div class="form-group">

<label **asp-for**="@Model.Price">Price</label>

<input **type**="text" **asp-for**="@Model.Price" class="form-control" placeholder="Price">

</div>

<div class="form-group">

<label **asp-for**="@Model.Category">Category</label>

<input **type**="text" **asp-for**="@Model.Category" class="form-control" placeholder="Category">

</div>

<div class="form-group">

<label **asp-for**="@Model.Description">Description</label>

<input **type**="text" **asp-for**="@Model.Description" class="form-control" placeholder="Description">

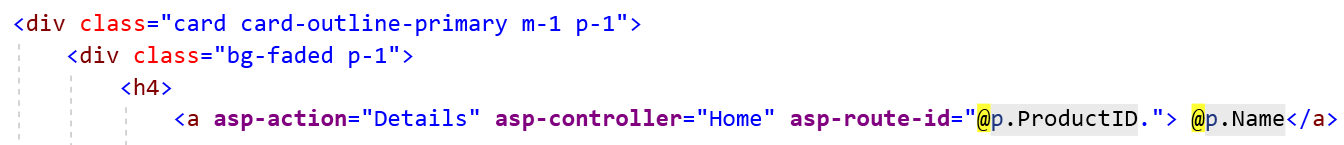
</div>

<button type="submit" class="btn btn-default">Submit</button>

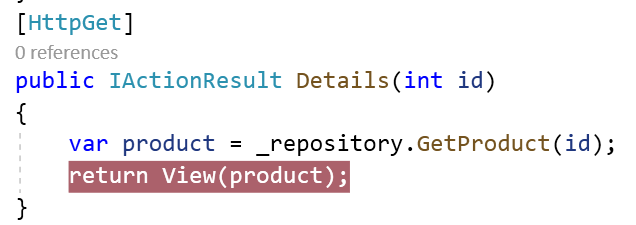
</form>

Model Binding Example 2

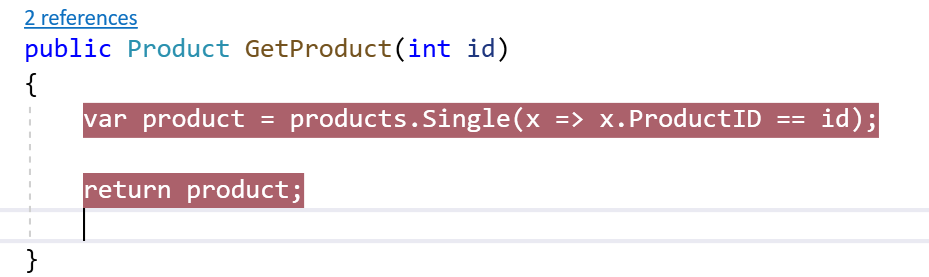
1. Modify Index.cshtml



1. Create Detaisl.cshtml that can take the Parameter.



1. Add GetProduct Method in ProductInMemoryRepository



1. Add Details View

@model Product

@\*@Html.ActionLink("Create","Create","Home", new class{)\*@

<a **asp-action**="Index" **asp-controller**="Home">Back To Home</a>

<div class="card card-outline-primary m-1 p-1">

<div class="bg-faded p-1">

<h4>

@Model.Name

<span class="badge badge-pill badge-primary" style="float:right">

<small>@Model.Price.ToString("c")</small>

</span>

</h4>

</div>

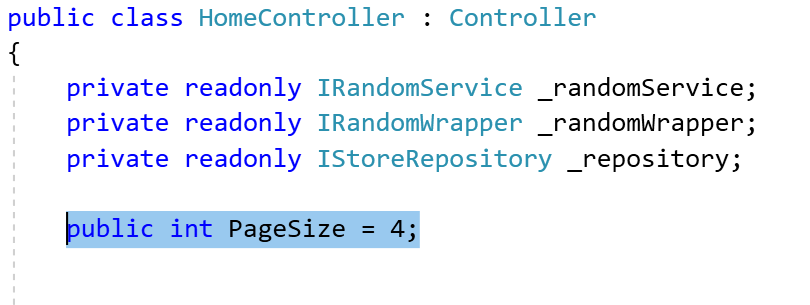
<div class="card-text p-1">@Model.Description</div>

</div>

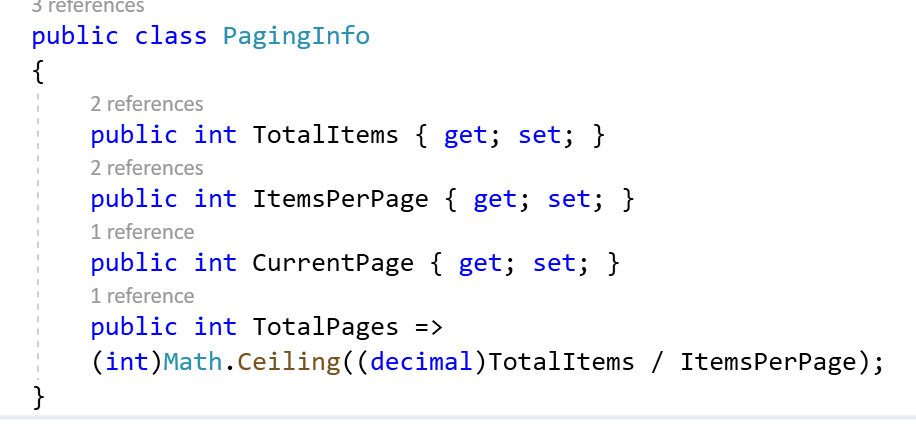
1. Check the Output

Demo How to Add Pagination

1. Add Default PageSize in the Controller



1. In Order to Calculate number of Pages Available, Current Page and Total number of Products available. Lets Create a model



1. Add PageLinkTagHelper class that contains logic to Display the links . Add this into infrastructure folder

[HtmlTargetElement("div", Attributes = "page-model")]

public class PageLinkTagHelper : TagHelper

{

// Tag Helpers used IUrlHelperFactory Object to Generate URL

private IUrlHelperFactory urlHelperFactory;

public PageLinkTagHelper(IUrlHelperFactory helperFactory)

{

urlHelperFactory = helperFactory;

}

// Object that provides access to HttpContext, Request and Response

// Need to Set [ViewContext ] In order to gain such access

[ViewContext]

public ViewContext ViewContext { get; set; }

public PagingInfo PageModel { get; set; }

public string PageAction { get; set; }

// TagHelperContext : Get the information Related execution of TagHelper

// TAgHelperOutput : Represents output of the tagHlper

public override void Process(TagHelperContext context,

TagHelperOutput output)

{

// Current ITagHelper Information for the request associated with context

IUrlHelper urlHelper = urlHelperFactory.GetUrlHelper(ViewContext);

// Build URLs

TagBuilder result = new TagBuilder("div");

for (int i = 1; i <= PageModel.TotalPages; i++)

{

TagBuilder tag = new TagBuilder("a");

tag.Attributes["href"] = urlHelper.Action(PageAction,

new { productpage = i });

tag.InnerHtml.Append(i.ToString());

result.InnerHtml.AppendHtml(tag);

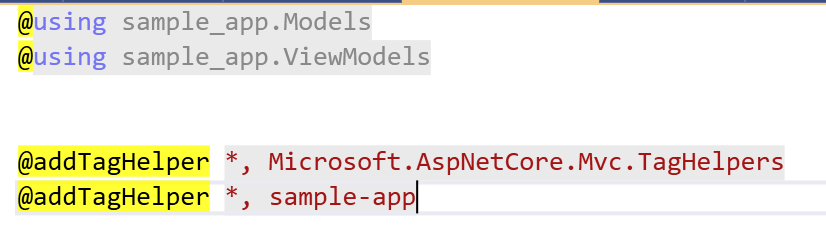
}

output.Content.AppendHtml(result.InnerHtml);

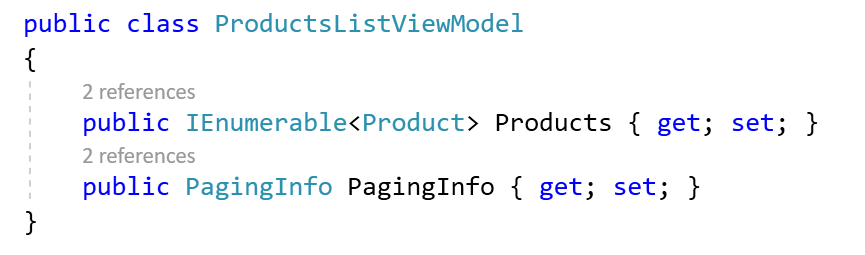
}

}

1. Expose this TagHelper through \_viewImports.cshtml



1. Create ProductListViewModel



1. Write a Code to Create appropriate model for view.

public IActionResult Index(int productPage = 1)

{

string result = $"Random Wrapper : {\_randomWrapper.GetNumber()} " +

$"and Random Service :{\_randomService.GetNumber()}";

ViewBag.Result = result;

var productListViewModel = new ProductsListViewModel

{

Products = \_repository.Products

.OrderBy(p => p.ProductID)

.Skip((productPage - 1) \* PageSize)

.Take(PageSize),

PagingInfo = new PagingInfo

{

CurrentPage = productPage,

ItemsPerPage = PageSize,

TotalItems = \_repository.Products.Count()

}

};

return View(productListViewModel);

}

1. Change Index.chtml to work with new ProductListViewModel

@model sample\_app.ViewModels.ProductsListViewModel

@\*@Html.ActionLink("Create","Create","Home", new class{)\*@

<a **asp-action**="Create" **asp-controller**="Home">Add New Painting</a>

@foreach (var p in Model.Products)

{

<div class="card card-outline-primary m-1 p-1">

<div class="bg-faded p-1">

<h4>

<a **asp-action**="Details" **asp-controller**="Home" **asp-route-id**="@p.ProductID"> @p.Name</a>

<br/>

@\*<img width="100px" height="100px" src="https://d1zdxptf8tk3f9.cloudfront.net/ckeditor\_assets/pictures/171/content\_5899676716\_ff0d235315\_z.jpg"/>\*@

<span class="badge badge-pill badge-primary" style="float:right">

<small>@p.Price.ToString("c")</small>

</span>

</h4>

</div>

<div class="card-text p-1">@p.Description</div>

</div>

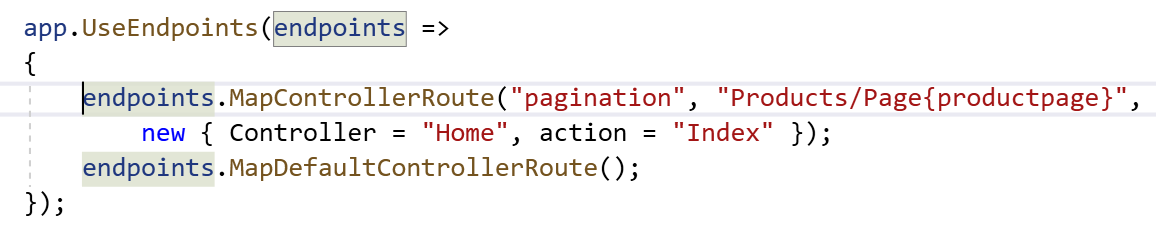
}

<div **page-model**="@Model.PagingInfo" **page-action**="Index"></div>

1. Check the Output

Demo : Add Conventional Routing

1. Add Route to the Startup.cs



1. Check the Output
2. Change the Styling of Pagination

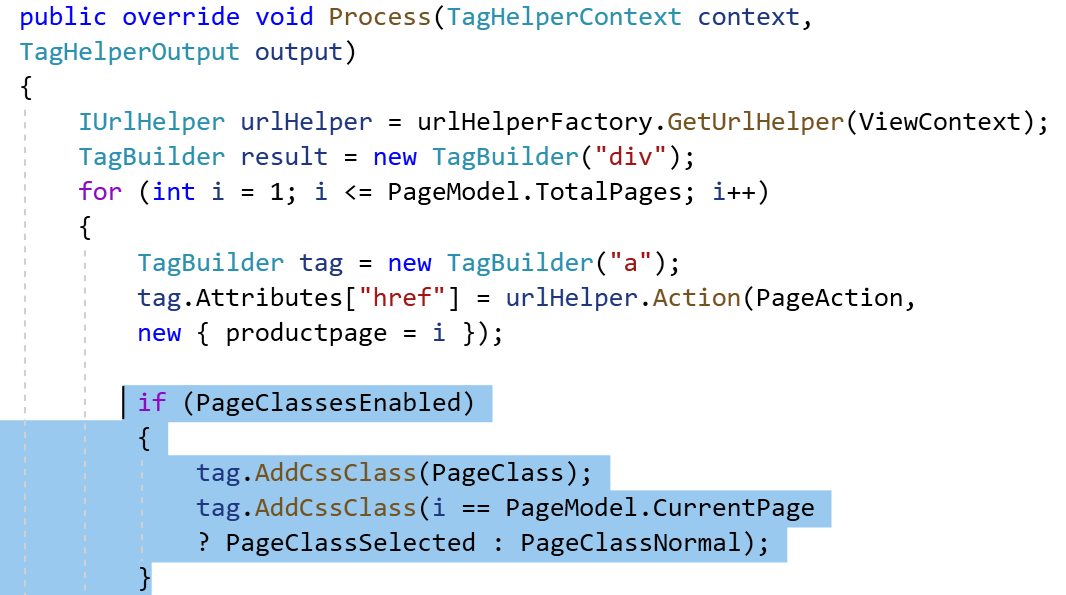
<div page-model="@Model.PagingInfo" page-action="Index" page-classes-enabled="true"

page-class="btn" page-class-normal="btn-outline-dark"

page-class-selected="btn-primary" class="btn-group pull-right m-1">

</div>

1. Modify te Tag Helper to use the Classes as the selection is made

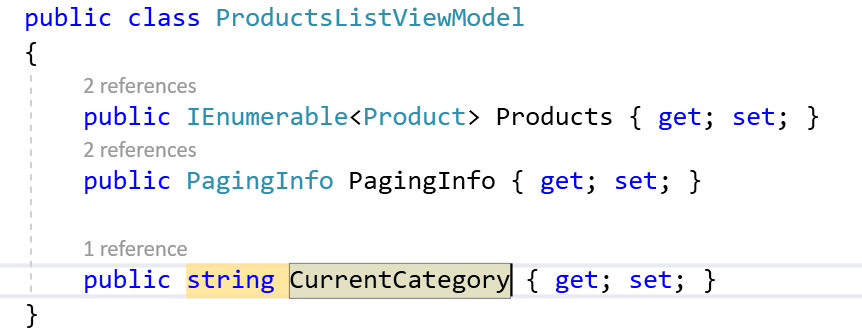


1. Check the Output

We want to allow use to Filter the Products By Category. We want to set the current Category

Keep DomainModel and ViewModel Separate

1. Domain model represents the object that represents the data in database. I
2. Domain model is related to data access layer of the application.
3. ViewMode refers the objects which hold the data that needs to be show to the user.
4. ViewModel is related to the presentation layer of our application.
5. They are based on how the data is presented to the user rather than how they are stored.
6. Avoid using Domain model as Viewmodel you could expose potentially dangerous properties in the view.
7. The domain models are persisted to the database using the data access layer .Hence the exposed properties may be updated / inserted into the database accidentally.
8. Create ViewModel per view
9. ViewModel should contain onle data and no logic
10. Very useful when data needs to be pulled from different data sources
11. Since viewModels are disconnected from the domain model . that gives the flexibility to use it the ways you see it.
12. ViewModels makes the application more secure as you don’t have to expose the potentially dangerous properties like UserRole , isAdmin in the viewModel
13. Add CurrentCategory in ProductListViewModel



1. Change the Index action to accommodate Category Parameter

public IActionResult Index(string category , int productPage = 1)

{

string result = $"Random Wrapper : {\_randomWrapper.GetNumber()} " +

$"and Random Service :{\_randomService.GetNumber()}";

ViewBag.Result = result;

var productListViewModel = new ProductsListViewModel

{

Products = \_repository.Products

.Where(p=> category == null || p.Category == category)

.OrderBy(p => p.ProductID)

.Skip((productPage - 1) \* PageSize)

.Take(PageSize),

PagingInfo = new PagingInfo

{

CurrentPage = productPage,

ItemsPerPage = PageSize,

TotalItems = \_repository.Products.Count()

},

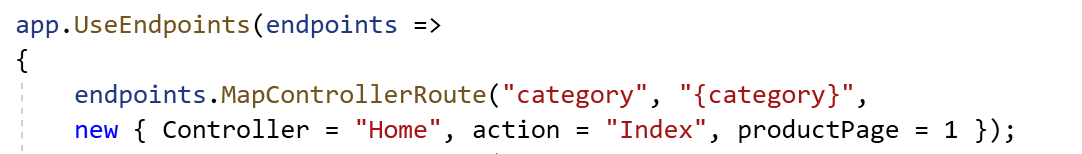
CurrentCategory = category

};

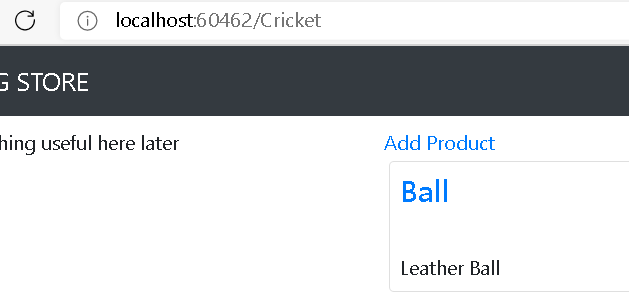
return View(productListViewModel);

}

1. Lets Add Route Where Category can be passed from URL

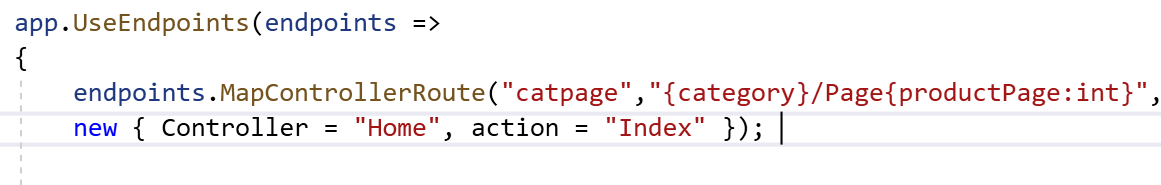


1. Check the Output



We might also want to Get the Category Information PageWise

1. Add Route for the same



1. Check the Output

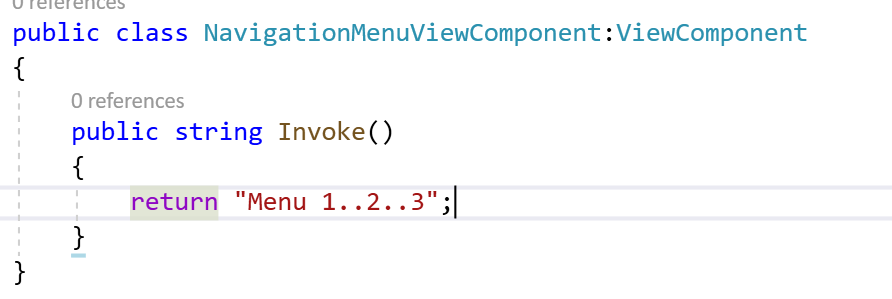


Building Navigational Component

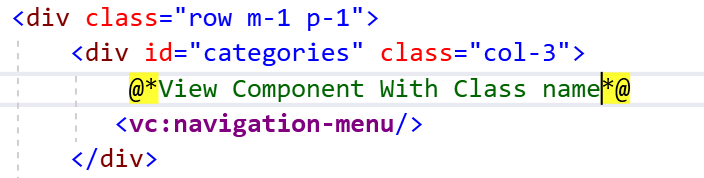
* Need to Provide users with a way to select category that doesn’t involve in typing URLs,that is display list of available categories and indicating which is currently selected
* ASP.NEt Cor has the concepts of View Components

Demo : Navigational components

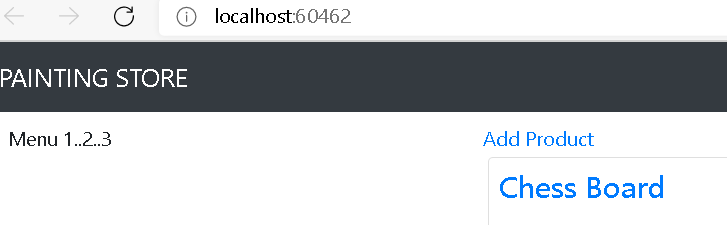
1. Add Components folder and add new component



1. Include this component in Layout



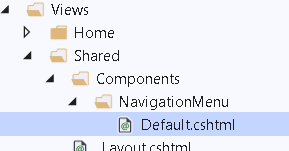
1. Check the Output



1. Modify Navigation Component to Get the Category Values



1. Create Components folder inside shared folder and add view in it



1. Add code in Default

@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 btn-outline-secondary"

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

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

**asp-route-productPage**="1">

@category

</a>

}

1. Check the Output

Current Issue Page URL is not changing as per the Selected Category

1. Change the Code Count Total Item as per Category, in Home Controller

PagingInfo = new PagingInfo

{

CurrentPage = productPage,

ItemsPerPage = PageSize,

//TotalItems = \_repository.Products.Count()

TotalItems = category == null ?

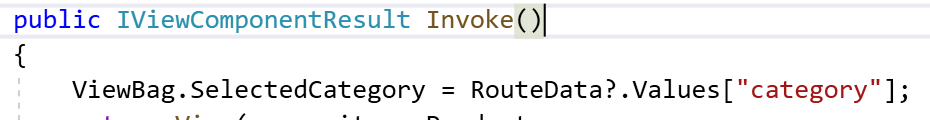
\_repository.Products.Count() :

\_repository.Products.Where(e =>

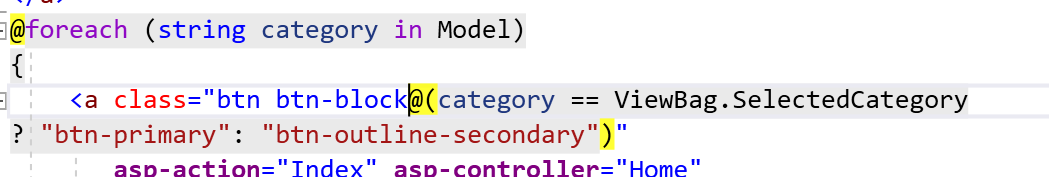
e.Category == category).Count()

},

1. Check the Output
2. Highlight the Selecte Category
3. Get the the Selected Category From Route Data



1. Apply Particular Style , for that Particular Category in Default.cshtml



1. Check the Output

Routing in Core

* Routing is responsible for matching incoming HTTP requests and dispatching it those requests to the apps executable endpoints.
* Endpoints are the apps unit of executable request handling code.
* Endpoints are defined in the app and configured in the app when its starts.
* Endpoint matching process can extract values from the requests URL and provide those values for request processing.
* ASP.Net Core Controller use Routing middleware to match the URLS of incoming requests and map them to actions.

It can be route in 2 ways

1. Conventionally routed
2. Attribute Routes

Asp.Net Core 3.0 added Endpoint Routing : <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-3.1>

Conventional Routing

* Routing is registered in the middleware pipeline.
* Routing uses pair of middleware

1. Use Routing : add route matching to the middleware pipeling. It looks at the set of endpoints defined in the app and select best match based on the request.
2. UseEndpoints : Adds endpoint execution to the middleware pipeling. It runs the delegate associated with the selected endpoint.

Endpoint

MapGet Method is used to define an endpoint

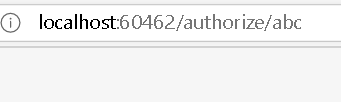
1. It selected
2. Executed by running delegate.

Demo : How to Add Custom Routes

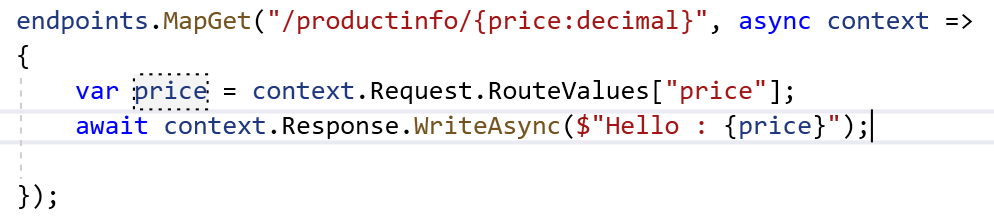
1. Add Custom Route in order to allow alphabet of length 4



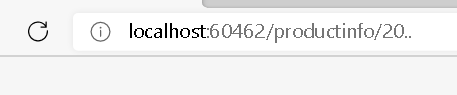
1. Check the Output



1. Add Endpoint for Price



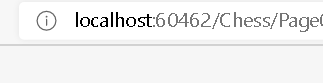
1. Check the output



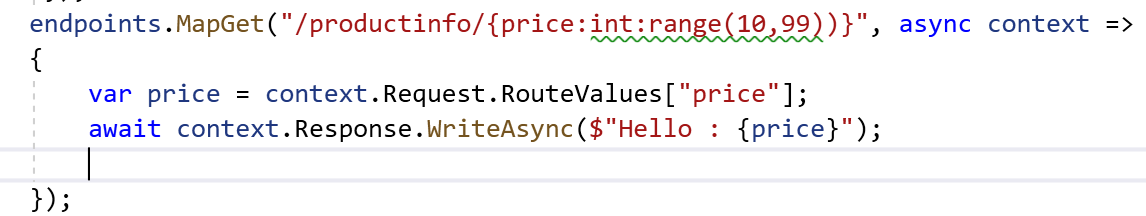
1. Add more route



1. Check Output



1. Add more routes



1. Check the Output. It wont work for other ranges