Passing Data Lab

Lab 1: View Data

Open **\Views\Home\Index.cshtml**

Change the Title at the top of the file

```
@{
    ViewData["Title"] = "Adventure Works";
}
```

Open **ProductMaintenance.cshtml**

Add the following at the top of the file

```
@{
    ViewData["Title"] = "Product Maintenance";
}
```

Open **CategoryMaintenance.cshtml**

Add the following at the top of the file

```
@{
    ViewData["Title"] = "Category Maintenance";
}
```

Open **ColorMaintenance.cshtml**

Add the following at the top of the file

```
@{
    ViewData["Title"] = "Color Maintenance";
}
```

Try it Out

Run the application

Click on each link and notice that the title in your browser's tab is now that same as the ViewData value you added to each page

Lab 2: View Bag

Open **ProductMaintenance.cshtml**

Remove the code you added in the last lab

```
@{
    ViewData["Title"] = "Product Maintenance";
}
```

Open **CategoryMaintenance.cshtml**

Remove the code you added in the last lab

```
@{
    ViewData["Title"] = "Category Maintenance";
}
```

Open **ColorMaintenance.cshtml**

Remove the code you added in the last lab

```
@{
    ViewData["Title"] = "Color Maintenance";
}
```

Open the **ProductMaintController.cs** file

Add the code shown in bold you see below:

```
[HttpGet]
public IActionResult ProductMaintenance()
{
  ViewBag.Title = "Product Maintenance";

  return View();
}

[HttpGet]
public IActionResult ColorMaintenance()
{
  ViewBag.Title = "Color Maintenance";

  return View();
}

[HttpGet]
public IActionResult CategoryMaintenance()
{
  ViewBag.Title = "Category Maintenance";

  return View();
}
```

Try it Out

Run the application

Click on each link and notice that the title in your browser's tab is now that same as the ViewBag value you added to each page

Lab 3: Add Configuration Settings

Add some configuration settings to store application-wide settings.

Open **\appsettings.Development.json** and add the following code shown in bold.

```
{
  "AdvWorks": {
    "DefaultCost" : 100,
    "DefaultPrice" : 200,
    "DefaultColor" : "Red"
  }
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```

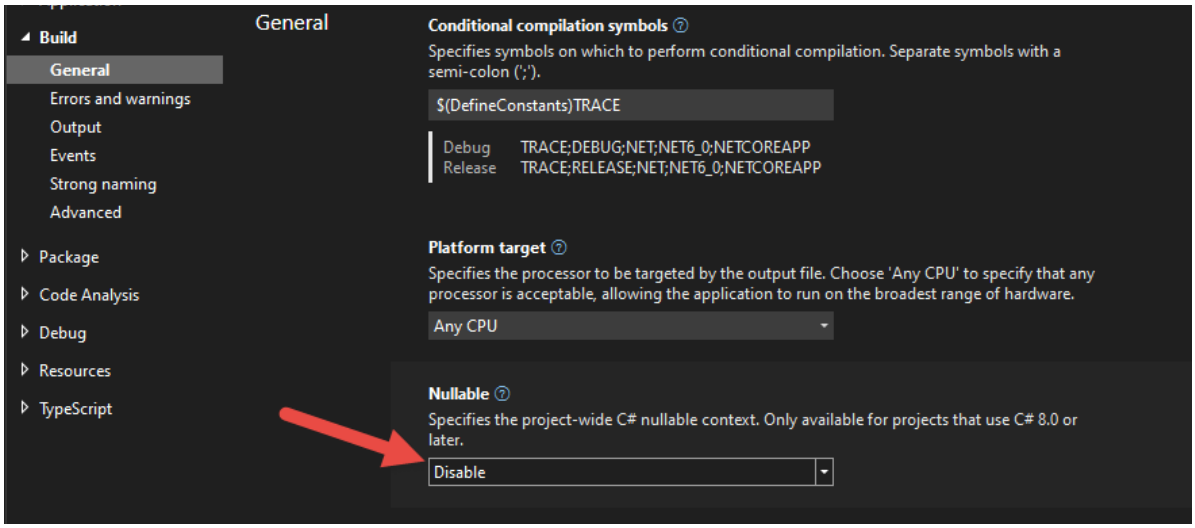
Lab 4: Add View Model Properties

Right mouse-click on the \Models folder and add a new class named **ProductViewModel.cs** and add three properties.

```
public class ProductViewModel
{
  public decimal DefaultCost { get; set; }
  public decimal DefaultPrice { get; set; }
  public string DefaultColor { get; set; }
}
```

Global Nullable Setting

After adding this code, you are going to see that the *DefaultColor* property has a squiggly underneath it. This is because the default for nullable values has been changed in .NET 6. Right mouse-click on the **AdvWorks** project and select Properties from the menu. Expand the Build tab and click on the General tab. Modify the Nullable setting to Disable as shown in the following screen shot.



Right mouse-click on the **\Views\ProductMaint** folder and add a new view named **ProductDetail.cshtml**.

```
@model AdvWorks.Models.ProductViewModel

<h1>Product Information</h1>

<form>
  <div class="form-group">
    <label for="name">Product Name</label>
    <input class="form-control" id="name" name="name" />
  </div>
  <div class="form-group">
    <label for="productNumber">Product Number</label>
    <input class="form-control" id="productNumber"
name="productNumber" />
  </div>
  <div class="form-group">
    <label for="color">Color</label>
    <input class="form-control" id="color" name="color"
value="@Model.DefaultColor" />
  </div>
  <div class="form-group">
    <label for="standardCost">Cost</label>
    <input class="form-control" type="number" id="standardCost"
name="standardCost" value="@Model.DefaultCost" />
  </div>
  <div class="form-group">
    <label for="listPrice">Price</label>
    <input class="form-control" type="number" id="listPrice"
name="listPrice" value="@Model.DefaultPrice" />
  </div>
  <div class="form-group">
    <button class="btn btn-primary">Save</button>
  </div>
</form>
```

Open the **ProductMaintController.cs** file and add some using statements

```
using Microsoft.Extensions.Configuration;
using AdvWorks.Models;
```

Add a new field

```
private readonly IConfiguration _config;
```

Add a Constructor and inject the IConfiguration object

```
public ProductMaintController(IConfiguration config)
{
  _config = config;
}
```

Add a ProductDetail() method

```
[HttpGet]
public IActionResult ProductDetail()
{
  ViewBag.Title = "Product Information";

  ProductViewModel vm = new()
  {
    // Retrieve items from configuration settings file
    DefaultCost = _config.GetValue<decimal>("AdvWorks:DefaultCost"),
    DefaultPrice =
_config.GetValue<decimal>("AdvWorks:DefaultPrice"),
    DefaultColor = _config.GetValue<string>("AdvWorks:DefaultColor")
  };

  return View(vm);
}
```

Open the **ProductMaintenance.cshtml f**ile

Add the following immediately after the <h2> tag

```
<div class="row">
  <div class="col">
    <a asp-action="ProductDetail"
       asp-controller="ProductMaint"
       class="btn btn-primary">
      Add
    </a>
  </div>
</div>
```

Try it Out

Run the application

Click on the Product Maintenance link

Click on the **Add** button

You should see the default values from the appsettings.Development.json file appear in the input fields

Demo 5: Dependency Injection

Right mouse-click on the **\Models** folder and add a new class named **AppSettings.cs**

```
namespace AdvWorks.Models
{
  public class AppSettings
  {
    public decimal DefaultCost { get; set; }
    public decimal DefaultPrice { get; set; }
    public string DefaultColor { get; set; }
  }
}
```

Open the **Program.cs** file and add a using statement at the very top of the file

```
using AdvWorks.Models;
```

Immediately after the builder.Services.AddControllersWithViews() code add the following:

```
// Create an instance of the AppSettings configuration class
builder.Services.AddSingleton<AppSettings>();
```

Just after the app.UseAuthorization() line of code add the following:

```
// Get the configuration settings class
AppSettings settings =
app.Services.GetRequiredService<AppSettings>();
// Read settings from configuration file
settings.DefaultCost  =
  app.Configuration.GetValue<decimal>("AdvWorks:DefaultCost");
settings.DefaultPrice =
  app.Configuration.GetValue<decimal>("AdvWorks:DefaultPrice");
settings.DefaultColor =
  app.Configuration.GetValue<string>("AdvWorks:DefaultColor");
```

Open the **ProductMaintController.cs** file

Change the _config field you added earlier to be an AppSettings object

```
private readonly AppSettings _settings;
```

Modify the Constructor to inject the AppSettings object

```
public ProductMaintController(AppSettings settings)
{
  _settings = settings;
}
```

Modify the ProductDetail() method

```
[HttpGet]
public IActionResult ProductDetail()
{
  ViewBag.Title = "Product Information";

  ProductViewModel vm = new()
  {
    // Retrieve items from settings class
    DefaultCost = _settings.DefaultCost,
    DefaultPrice = _settings.DefaultPrice,
    DefaultColor = _settings.DefaultColor,
  };

  return View(vm);
}
```

Try it Out

Run the application

Check to make sure you are still getting the same values in the product form