



Module 3 Day 5

MVC Views 3 – Ends and Odds

What makes an application?

- Program Data

- ✓ Variables & .NET Data Types
- ✓ Arrays
- ✓ More Collections (list, dictionary, stack, queue)
- ✓ Classes and objects (OOP)

- Program Logic

- ✓ Statements and expressions
- ✓ Conditional logic (if)
- ✓ Repeating logic (for, foreach, do, while)
- ✓ Methods (functions / procedures)
- ✓ Classes and objects (OOP)
- ❖ Frameworks (MVC)

- Input / Output

- User
 - ✓ Console read / write
 - ✓ HTML / CSS
 - ❑ Front-end frameworks (HTML / CSS / JavaScript)
- Storage
 - ✓ File I/O
 - ✓ Relational database
 - ❑ APIs

ViewData

- Another way to pass data from Controller to View
 - Not type-safe, no IntelliSense...
 - "Quick and dirty"
 - May be used for messages, for example
- In the Controller:

```
string message = $"{products.Count} {(products.Count == 1 ? "product" : "products")} were found.";
ViewData["message"] = message;
return View(products);
```

- In the View:

```
<h3>@ViewData["message"]</h3>
<h2>All Products</h2>
```

Let's
Code

Even More on Layout

- Partial Views

- A Razor file (*.cshtml*) that renders HTML output *within* another markup file's rendered output
- Used to place duplicate markup into multiple Views
 - E.g., a common form, or a product display tile
- Name starts with `_` by convention (not required)
- “Using” View includes the partial view with:
`<partial name=“_partialVwName” model=“Model” />`

- Example

- `_Filters` partial view
- Used in Index, Tile

Passing QueryString Parameters

- Asp-route-{variable-name} adds {variable-name} to the querystring on the URL
- When that URL is clicked, the framework does [Model Binding](#)
 - Maps URL parameters (query string) to Action parameters
 - Looks for names in the query string that match parameter name
 - But there's more magic: it also looks for public properties of Action parameter types and matches those names if possible.
- Example:
 - Product sort order (int)
 - Product filtering (complex type)



Let's
Code

ActionResult

- Actions are required to return *ActionResult*
- Controller View() method returns a *ViewResult*
- Other types of ActionResult:
 - RedirectResult, RedirectToActionResult
 - OkResult, NotFoundResult, CreatedResult, ForbidResult, StatusCodeResult
 - FileContentResult, FileStreamResult
 - JsonResult
 - And many more
- Examples
 - HomeController.Index returns a re-direct
 - ProductsController.Detail should return a NotFound (404).
Let's do that.



Let's
Code

A little more on Layout Pages

- `_ViewStart.cshtml`
 - Code in this file runs before any view code runs
 - So every view does not need to set its Layout
- You can Render more content than just a single Body
 - `@RenderSection` is used in the Layout file

```
    </main>  
    @RenderSection("sidebar", false)  
  </body>
```

- `@section` is used in the View file to define blocks of code to be rendered

```
@section sidebar {  
    <aside>  
        Here is a side-bar.  
    </aside>  
}
```

Let's
Code

Dependency Injection

- Don't let this blow your mind!
- Inversion of Control pattern
- Removes the task of creating DAO's in the Controllers
- Tells the framework to create and pass DAO's into the controller
- Example:
 - ProductsController constructor



Let's
Code