**Where we are:**

|  |  |
| --- | --- |
| Intro to MVC | Navigation, Searching and Filtering |
| Controllers and Views | Forms and Tag Helpers |
| Unit Testing, Publishing to Azure | Annotation and validation |
| Dependency Injection, Models | Authentication |
| Entity Framework and Scaffolding | Authorization |

Intro

Why do we need forms? It’s the primary way to get user input other than having users click on links. It’s also the main (only?) way to generate an HTTP Post to send user data to the server.

Review

Last week’s topics:

* Searching (filtering)
* Navigation links on a shared layout (we didn’t format them as buttons)
* We didn’t and won’t cover components, we may have time to briefly talk about Bootstrap

Last week’s SportsStore example:

* Category filtering with buttons
  + Buttons are in a Bootstrap NavBar, see: <http://getbootstrap.com/components/#nav>
* Pagination (not a focus for us)
* Cart
  + Model which includes:
    - a list of Cart Line objects
    - AddItem, RemoveItem, and other methods that manage operations related to the collection of line items.
  + Unit tests for the Cart model
  + ProductSummary.cshtml which is a partial view
    - Form, created with a tag helper  
      *See Ch. 24, Form Tag Helpers*
    - Hidden input elements created with tag helpers  
      Q: Why are the hidden elements there?  
      A: Because of the statelessness of HTTP, and because this is data that the user doesn’t need, but the app does.
  + Sessions enabled
    - For the cart, session state is stored in the server’s memory
    - Required more packages, specified in project.json
    - Wrote extension methods for serializing / deserializing JSON
  + CartController
    - Cart objects are stored and retrieved using session state
    - Model binding is used to associate form fields with parameter names  
      *See Ch. 26, Model Binding*
    - RedirectToAction methods
  + CartIndex view model
    - Has a ReturnURL
  + Cart displayed in Default.cshtml
    - Uses a cart icon from <http://fontawesome.io>

Reading Guide

Ch. 10, SportsStore: Completing the Cart

* Services (like the one used to inject Fake repositories into controller methods)
  + We won’t focus on this
* Remove button on Cart summary—this is a form with a button and uses tag helpers
* Submitting Orders
  + New model
    - Uses validation attributes, Ch. 27: Model Validation
  + Checkout.cshtml form
    - Uses regular HTML Form tag (asp-action and post attributes)

Tag Helpers

Look at the Cities example in Ch. 23

Example TagHelper class definition:

public class ButtonTagHelper : TagHelper

{

public string BsButtonColor { get; set; }

public override void Process(TagHelperContext context, TagHelperOutput output)

{

output.Attributes.SetAttribute("class", $"btn btn-{BsButtonColor}");

}

}

TagHelperContext: Provides information about the HTML element. It has these properties:

* AllAtributes: read-only dictionary of attributes applied to the element
* Items: a dictionary used for coordinating between TagHelpers
* UniqueId: the element’s identifier

output.Attributes.SetAttribute: You can set any HTML element’s attribute using a C# version of it’s name: bs-button-color, becomes BSButtonColor.