Module 3 Day 7

MVC Controllers - POST

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

Follow-up on Yesterday

- [Display] Data Annotation for labels
- Select asp-for, asp-items
- Testing Controllers
  - Mock Objects
  - Testing IActionResult
  - Testing the Model

Let’s Code

GET vs. POST

GET	POST
Form data sent in QueryString (URL)	Form data sent in Request Body
Can be bookmarked	Cannot be bookmarked
Visible as clear text in address line and history	Not visible in address line or history
Length of URL limited ~3000 characters	Unlimited Body size
Use when the user may want to execute it again (idempotent). Should not modify data on the server.	When same form data should not be submitted twice (data changes on the server)

- Bottom line:
  - DO NOT use Get when data is to be modified (insert, update, delete)
  - Use Form Post

Post-Redirect-Get Pattern

- If user refreshes page after post, form can be re-submitted
- PRG Prevents “double-posting” of a form due to browser refresh
- Places a Get as the browser’s most recent request
- The pattern
  - A successful Post action returns a Redirect (3xx) to a success page
    - May include an id or other parameter
  - Browser issues the Get Request as demanded
  - Server returns the success page
- If the user refreshes, the success page refreshes

Let’s Code

Handling methods with different actions

- Action method attributes
  - [HttpGet]
    - The action responds to Http GET requests
  - [HttpPost]
    - The action responds to Http POST requests
  - [AcceptVerbs("get", "post")]
- If none of these attributes is specified, all verbs are accepted

Cross-Site Request Forgery

- A malicious site can use a previously created authentication cookie to get work done and do damage
- ASP.Net sends an additional token with each POST request to prevent this type of attack (Anti-forgery token)
- You mark your code to “validate” the token on posts
  - On the controller class or method (action): [ValidateAntiForgeryToken]
  - On the controller class: [AutoValidateAntiforgeryToken]
  - Globally: services.AddMvc(options => options.Filters.Add(new AutoValidateAntiforgeryTokenAttribute()));
- https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery?view=aspnetcore-2.2

Let’s Code