Assignment 5 – MVC
COS318 – FA2015

Due Date: October 15th, 2015
Turn in all files using Moodle

For the fantastic fifth assignment, you will be creating some MVC controllers and Razor views in a wonderfully wizarding world. All of the html rendering will be done on the server-side, so you won't be writing any plain HTML files. This assignment references saving data a few times. Your data storage does not need to be persistent; i.e. it is okay if each time your web program is started the data storage is reset. Expelliarmus!

Your MVC project must contain the following endpoints and associated views:

1. **(40 Points) /spells**
    a. **GET:** Render a list of magic spells currently saved. They should be rendered in an ordered html list, with odd numbered rows being different colors. Each row should contain a link that allows the user to navigate directly to the spell.
    b. **/add:** Accept data that will allow a user to add a new spell to the list. The data either be accepted as JSON data in the body or as form parameters. The field and button to accept new spells should be rendered at the bottom of the /spells endpoint from step a.
    c. **/delete:** Accept a URL or query parameter that will allow a user to delete a spell name.
    d. **/viewSpell:** Add a query parameter or URL parameter that will allow the user to specify a numerical index. If it is a valid index, **render just that spell's name on the page** with its index, but in bigger text than the list from step a. This page should also **include a delete button** that will call the DELETE endpoint from step c, and then redirect the user back to /spells from step a. Lastly this page should **include a 'cancel' button** that returns the user to the /spells page from step a, but doesn't delete the spell.
2. **(40 Points) /potions**
    a. **GET:** Render a list of at least five potion ingredients in html. Make them creative. Give some way for a user to select two ingredients to mix. **Add a 'mix' button** that will send the two selected ingredients to POST /potions.
    b. **POST:** Accept two ingredients as strings as **form data**. If both are not specified, you should redirect to /potions. Otherwise, **mix the two ingredient strings together randomly**, but preserving the order of the letters in each word. For example, if you had 'apple' and 'banana' as ingredients, one possible result returned would be 'apbapnalnea.' The mixed string should then be added to the list of possible ingredients to mix then redirect to /potions.
3. **(20 Points)** Code style, formatting, completeness, and quality.

The Rules

1. No plain HTML files. All HTML from this assignment must be rendered on the server side with Razor pages.
2. No inline CSS styles. All styles must be from separate CSS files. These CSS files must come from the server.

3. Visual Studio projects contain a lot of files. Zip up your entire project directory and submit only a single file to Moodle.

Stretch Levels

If you already have a lot of experience with MVC or server-side code, or if you just like casting spells, try to complete these stretch levels for extra credit. The levels are cumulative, so for example, don't try for silver if you haven't finished bronze. If you try for the stretch levels, make sure to type it in the comments on Moodle so I don't miss it.

Bronze Level (Ron Weasley)

On the /spells endpoint during step 1b and the /potions endpoint from step 2b, you allowed the endpoint to accept values in either URL parameters or query parameters. Update your controllers to accept whichever type you didn't accept before so now you support both. Add a second link on the /spells page. One link should use the query parameter, the other should use the URL parameter.

Silver Level (Hermione Granger)

Add a second button on the /potions page. This button should send the data to the /potions endpoint as URL parameters instead of form data.

Gold Level (Harry Potter)

Make the spell list from step 1 persistent. If you stop and restart the program, the data from the spell list should be restored from its last state. You can choose to store this data in any method you choose, for examples, a SQL or NoSQL database, temporary files, or text files.