CT 310 Web Development

Computer Science Department (http://www.cs.colostate.edu)

Spring 2017

CT 310Project 3: Ingredients Federation

Project #3

Ingredients Federation

For Milestone due dates see schedule below.

See Addendum for updates.

Continuation: A Joint Federation of Ingredients For You Sites

This is a Paired Programming Project. It is the final in a three part semester project working toward a class federation of ingredient websites. New Teams will be assigned through Canvas by Friday 4/14. You'll build upon what you developed for Project 1 and 2. As mentioned in prior Projects, both of the team partners first need to review, discuss and agree upon what parts of their Project could be carried forward to this phase. In Project 3, you will be enhancing the functionality of your Ingredients For You site. The objective of this project is to have the class as whole create a Federated Ingredients Site(s). Each site will share the information about it's available ingredients with the Federation. Most of this new functionality will be developed in JavaScript utilizing JSON and jQuery.

Master Federation List and Common API alements

The CT310 account will provide a master list of all sites in the Federation in JSON (Array) format. The master list URL is accessible at `https://www.cs.colostate.edu/~ct310/yr2017sp/more_assignments/project03masterlist.php.` The JSON response is formatted as shown in the following example:

```
[{"team":"0","nameShort":"Yammy","nameLong":"Ingredients are Yammy","baseURL":"http:\/\/w
ww.cs.colostate.edu\/~yammy\/project3\/"}]
```

This example has a singel fake sight, once teams supply this information, the page will return an array of sites.

As will be discussed further below, there are four pages that you must supply with your site.

- `ajax_status.php`
- `ajax_listing.php`
- `ajax_ingredient.php`
- `ajax_ingrimage.php`

These files must reside at the URL provided in your baseURL folder in order to standardize the AJAX calls across the federation. The functionality of these four pages is discussed below, both from the standpoint of what the AJAX service must provide and also how you are expected to utilize that functionality when extending your Ingredients Website.

AJAX Status Page

The page `ajax_status.php` provides a mechanism for indicating if your site is open for business. This will be useful to all of you as you write code that must decide when to attempt to display ingredients from another site in the list of federated sites.

Your `ajax_status.php` page must return a JSON object. More precisely, if you want to signal your site is ready for business, it should return: `{"status":"open"}`
and if you are not ready to have other sites querying your site then instead return: `{"status":"closed"}` . Please implment this functionality,returning the closed option ASAP, since the entire class will start relying on each others pages.

Displaying Federation Status

Your site also should provide a page `fedr_status.php` that provides immediate visual feedback regarding the status of every site in the Federation Master List. Exactly how you format this display is up to you, provided you work into your display the following:

- The short name of each site.
- The long name of each site.
- A prominent color encoded status indicator for each site. There are three possible colors

There are actually three possible states and with each a color:

1. *Green:* Site responded as open when asked through their ajax_status.php page.
2. *Red:* Site responded as closed when asked through their ajax_status.php page.
3. *Yellow:* Site has not responded to an AJAX call to their ajax_status.php page.

Notice that this page is not something a Customer would generally want to see, but it will be handy as a visual indicator of what other sites are up and running: something you will want to know as you start pulling ingredient content from these other sites.

AJAX Listing Page

Your site must provide a page, `ajax_listing.php` , through which any other site in the federation may obtain a listing of available ingredients. The results should be returned using JSON. Here is an example:

```
[{"name":"Carrot","short":"What's Up Doc","unit":"bunch","cost":"1.99"},{"name":"Kale","s
hort":"Cabbage on the Wildside","unit":"lbs","cost":"2.69"}]
```

The `name` field must be unique to your site and it will be used in other AJAX calls to specify a specific ingredient your site sells. The `short` description is whatevery you like so long as the text does not exceed *50* characters. The `unit` field clarifies for customers what the our buying when they select a quantity to purchase. Finally, `cost` is a dollar amount per unit specified in the format shown: i.e. dollars then a colon then cents.

Displaying All Available Ingredients

Currently you have a page that displays all ingredients for sale from you site. For Project 3 you must now display all ingredients from all sites that are open as defined above by status. You have a lot of latitude when it comes to how exactly you want to display this information visually to customer. That said, here are some guidelines and

suggestions. You may want to not include thumbnail images in this master listing becuase it will become long. However you display each ingredient, perhpas as a unique row in a table, there must be a hyperlink to a page hosted on your site that displays information aobut that specific ingredient. More will be said about that page later.

You also face some choices in terms of timing and sorting. For example, you could insert directly into the table each time you get a response from another site. Alternatively, you could collect responses in and intermediate JavaScript structer (array of ingredient objects perhaps) and then only redo the table every couple of seconds or when you've detected that all open sites have provided you their ingreidents listing information. You should also think about how you want to group ingredients. Finally, pagination is a nice touch, but it is NOT required for displaying ingredients in this assignment: a single long presentation on one page is fine.

The AJAX Ingredients Page

Your `ajax_ingredient.php` page will return additional information on a specific ingredient appropriate for the generation of a display page highlighting the ingredient. Note this page will be used with an ingredient specifying on the URL. So, continuing in the spirit of caring about kale:

```
ajax_ingredient.php?ing="kale"
```

This AJAX call returns a JSON object very similar to a single entry from the `ajax_listing.php` page with two notable exceptions. It will include two additional fields in order to supply a long description, `desc` , and an availability status, `time` . So, for example:

```
{"name":"Kale","short":"Cabbage on the Wildside","unit":"lbs","cost":"2.69","time":"Ships
 today","desc":"A very popular member of the Cabbage family sometimes conisdered more ele
mental or primitive relative to other forms of cabbage. As to health benefits, consider t
his: \"Lutein and zeaxanthin, nutrients that give kale its deep, dark green coloring and
protect against macular degeneration and cataracts Minerals including phosphorus, potassi
um, calcium, and zinc.\" according to WebMD."}
```

As you write the code to generate these AJAX responses pay attention to escape characters in the formatting of strings, particularly the long description. Observe above that `json_encode` added and escape character in front of each of the internal quotations marks. Finally, for your own sake and the other sites depending upon you, make sure that information in the common fields returned by `ajax_ingredient.php` matches that returned by `ajax_listing.php` for the same ingredient.

The AJAX Ingredient Image Page

Your `ajax_ingrimage.php` page will stream the contents of an image of a single named ingredient that may then in turn be displayed. The details of how to send and image through an AJAX called will be presented in class. Note that like the `ajax_ingredient.php` page, the name of the ingredient is passed along in the URL. Indeed, the ingredient specification should be communicated in exactly the same manner.

Some Suggestions

Your approach in this assignment should be to build the AJAX API functionality first. That includes obviously testing it an making sure it is functioning as desired.

Do NOT treat your site as special when displaying the summary listing of federation ingredients nor when displaying specific ingredients. This advice is key for two reasons. First, it means you 'eat your own cooking' - to borrow a phrase. Second, and more important, it reduces the total amount of code you need to write. For example, think about the page you use to display a single ingredient. In Project 2, in all likelihood it took one GET (URL passed) argument specifying the unique name of the ingredient. To build Project 3, in essence you add one more argument, the short name of the site where the ingredient is hosted. Then, simply use the standard AJAX API to populate the page decription. If your AJAX calls are functioning properly, this will work as well for your site as any other.

As you build up support for the Federation, do not break functionality. In other words, everything your Projec 2 site (s) could do this new site can do. That includes the ability to reset passwords, shop and accumulate purchases in a shopping cart, etc. Now, one simplfication, you need NOT communicate information back to other sites when a customer adds ingredients from their site into your shopping cart. This lack of communication back is unrealisitc, but it would take us into a level of complexity beyond the scope of this assignment. Your shopping carts should simply record the short name of the sites hosting the ingredients being purchased.

Schedule

Because Project 3 has teams utilizing functionality being implemented by other teams a staggered set of deadlines/milestones is being established. These are summarized next.

Tuesday April 18th by Midnight
Groups submit through email to ct310@cs.colostate.edu their entry line for the master listing. Specifically their `team` , `nameShort` , `nameLong` and `baseURL` . Keep in mind the base URL is going to be associated with one of your CS Department accounts and websites. You can, and are encouraged, to place project3 in a subfolder of your `public_html` root.
Thursday April 20th by Midnight
Please have your `ajax_status.php` page in place and at least returning a properly formatted 'closed' message.
Thursday April 27th by Midnight
Dry run testing of Project 3 Team AJAX API. The GTA will test and record which teams have all four AJAX files in place and responding to queries correctly.
Friday May 5th Project 3 is Due
Code development freezes. The Projects are due and will be graded. Grading will be through live sites, although project teams will be asked to submit tar files of their site code as well. **Please host it on your CS account and provide the URL to your website as a comment during the submission.**

Project Teams

As with the previous projects, project teams are assigned at random. Every effort will be made to avoid pair students who have not previously worked together on a project.

Addendum

Beware of Dragons! Ok, not Dragons, but a nasty scoping issue that may show up as you update the display of site status information . Hopefully you have studied Example 1 (../aplay/lec26/ex01/lec26ex01.php) from Lecture 26 and if you followed the pattern of creating a seperate function to carryout your on-page-updates, you are blissfully unaware of the Dragon lurking nearby. However, if you thought it might be a good idea to avoid the extra function call, you will be very wrong and likely Dragon food. A symptom of the problem is your updates get loaded into the last of the page elements you want to update. We recommend you go and look at the newly written Example 2 (../aplay/lec26/ex02/lec26ex02.php) from Lecture 26; it generates the error for your inspection and thus you can avoid being trapped in a similar mistake. (4/29/17 Ross)

Please everyone note that for streaming your ingredient image, read your image data using get_file_content in php, create base64 encoded string of it and echo it. You can use your assignment8 solution to fetch image. (4/27/2017 Nikhil)

You will not be able to display or add ingredient comments on ingredients hosted by other sites. Moreover, to simplify your work on Project 3, you may discard all commenting capabilities if you find that makes your development work go more smoothly. (Ross 4/26/17)

This is important. In your `ajax_status.php` page you MUST include the following in your file at the very top of your file:

```
2 header ( 'Content-Type: text/json' );
3 header ( "Access-Control-Allow-Origin: *" );
```

If you mistakenly omit this header information then the automatic JSON parse at the receiving end will not work. Worse, as you use other site status queries you will not know whether or not to explicitly decode the JSON. This also holds for your other three `ajax_...` files. Lastly, please remember to use `json_encode` to format that information that you are sending back using `echo` . (4/24/17 Ross and Nikhil)

There is a new code example (../aplay/lec27/ex02/lec27ex02.php) as part of Lecture 27 (../aplay/lec27). This example demonstrates some useful techniques for testing whether a collection of ajax calls are successful. In particular, it demonstrates how to handle a file not found (404) error. The example also demonstrates how to accumulate status information in a JavaScript array. Please download and experiment with this example, including reading the comments in the JavaScript file, before you get very far along in handling the construction of your page for displaying the status of the Federated Sites. (Ross 4/21/17)

The baseURL provided by the Master List of sites follows these rules. First, it is the path to the root folder containing the site. It does not include any specific file name. For example, no `index.php` or `home.php` is included on the path. There is a trailing slash on the path so that it may be easily concatentated to the front of a file name to generate a complete file URL. Last but not least, you *must* provide an `index.php` file in your root that presents a pleasing homepage for your site. Because all sites will follow this rule, your site may correctly create links to the homepages of other sites. (Ross 4/20/17)