APRIL 2, 2014

ASSIGNMENT 4 REPORT

IAT 352

CASSANDRA DE GIT
301166220
cdegit@sfu.ca

Project Overview

{ Tutor } is a tool to help connect people who want to learn programming and those who want to teach it. Contributors post Lessons; blog post style content designed to teach some programming Topic. Learners are able to view these Lessons, and follow Contributors or Topics. By grouping content into specific Topics that users can search, { Tutor } makes it easier for Learners to find content they are interested in.

Unlike the suggested project, most of { Tutor }'s focus is on the posts themselves rather than the users creating them, as well as the organizing system for the posts, called Topics. This shaped a great deal of the functionality and design of the system.

Implementation

Users

{ Tutor } has two types of users: Contributors and Learners. These two users use the site for different purposes, and have different activities and features that are available to them.

Contributors

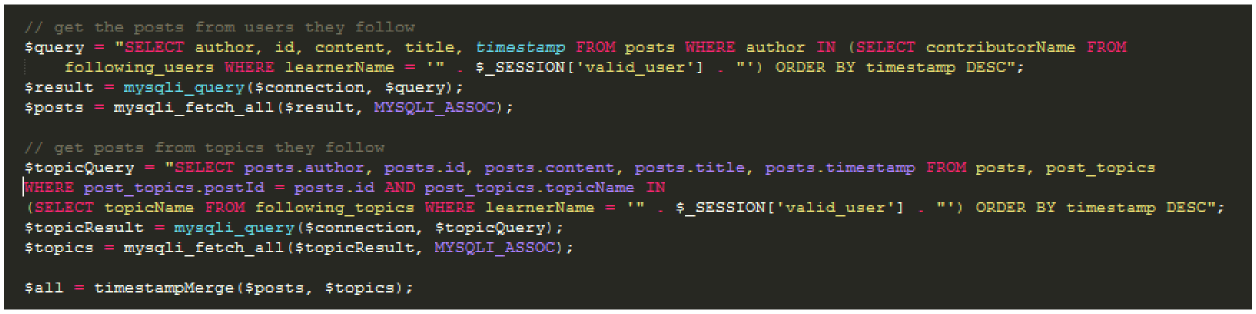
Contributors are the content producers of { Tutor }. Contributors write posts, add specific topics to them, and have public profiles. These profiles are used to help others learn more about the Contributor; the Contributor is able to configure them to display a description, their most recent Twitter posts and post recent Flickr photos.

Learners

While Contributors create content, Learners consume it. Learners read posts on the site, and follow Contributors and Topics. As Leaners don't require any information in the database that isn't already needed for Contributors, they are stored in the same table, but with null or default values for the unneeded fields.

Dashboard

The Dashboard is a page where all recent posts and tweets by followed Contributors or in followed Topics are aggregated. The Dashboard is only available to Learners, and allows them to collect and track content they care about.



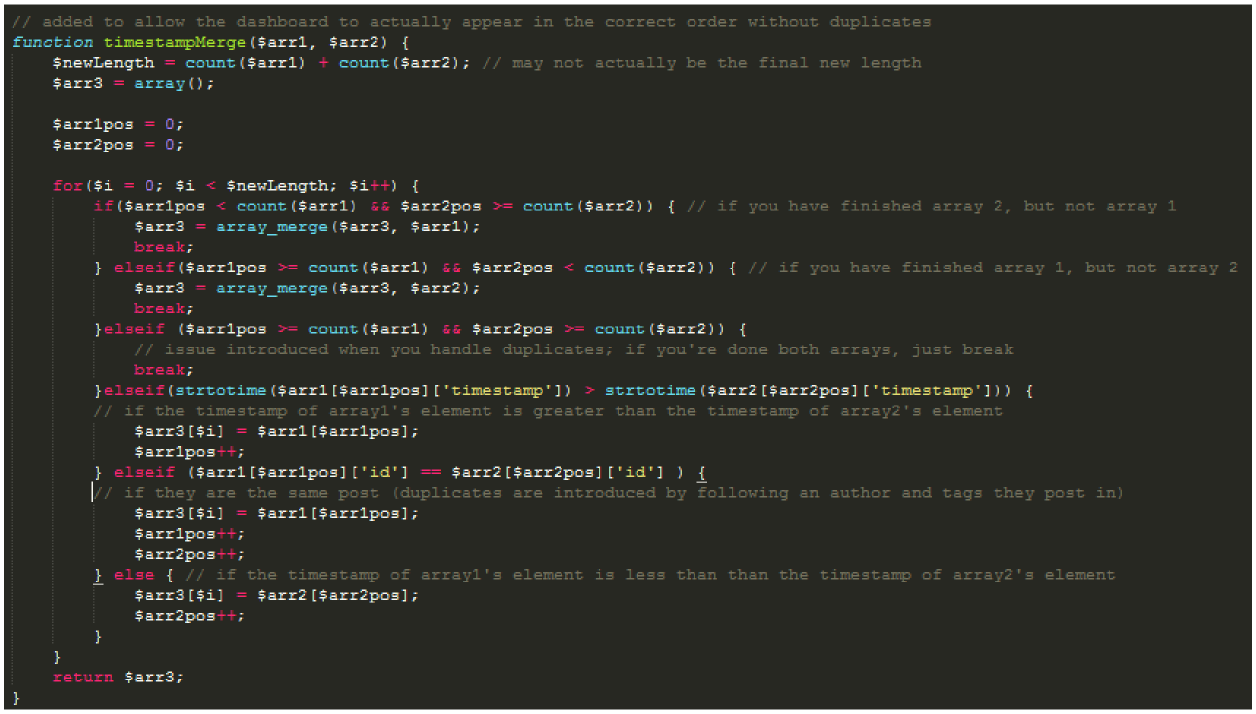
```
// get the posts from users they follow
$query = "SELECT author, id, content, title, timestamp FROM posts WHERE author IN (SELECT contributorName FROM
    following_users WHERE learnerName = '" . $_SESSION['valid_user'] . "') ORDER BY timestamp DESC";
$result = mysqli_query($connection, $query);
$posts = mysqli_fetch_all($result, MYSQLI_ASSOC);

// get posts from topics they follow
$topicQuery = "SELECT posts.author, posts.id, posts.content, posts.title, posts.timestamp FROM posts, post_topics
WHERE post_topics.postId = posts.id AND post_topics.topicName IN
(SELECT topicName FROM following_topics WHERE learnerName = '" . $_SESSION['valid_user'] . "') ORDER BY timestamp DESC";
$topicResult = mysqli_query($connection, $topicQuery);
$topics = mysqli_fetch_all($topicResult, MYSQLI_ASSOC);

$all = timestampMerge($posts, $topics);
```

Figure 1: The queries used to get the posts from the followed users and topics. From dashboard.php

Posts by followed contributors are fetched with a different request than posts in followed topics, so they results must be combined before they can be displayed. When combining the posts, we also want to ensure that we display them in order – to do this, a custom merge function, derived from the merge step of mergesort, is used. This function goes through both arrays, comparing the timestamps of each of the elements. The element with the most recent timestamp is added to the sorted result array, and the index used to iterate through the array of the chosen element is incremented. This function also removes any duplicate posts, which occur when the Learner follows multiple sources where the same post exists – a post tagged with multiple topics will occur in both topics, or the Learner following both the user and the Topic a post is added to.



```php
// added to allow the dashboard to actually appear in the correct order without duplicates
function timestampMerge($arr1, $arr2) {
    $newLength = count($arr1) + count($arr2); // may not actually be the final new length
    $arr3 = array();

    $arr1pos = 0;
    $arr2pos = 0;

    for($i = 0; $i < $newLength; $i++) {
        if($arr1pos < count($arr1) && $arr2pos >= count($arr2)) { // if you have finished array 2, but not array 1
            $arr3 = array_merge($arr3, $arr1);
            break;
        } elseif($arr1pos >= count($arr1) && $arr2pos < count($arr2)) { // if you have finished array 1, but not array 2
            $arr3 = array_merge($arr3, $arr2);
            break;
        }elseif ($arr1pos >= count($arr1) && $arr2pos >= count($arr2)) {
            // issue introduced when you handle duplicates; if you're done both arrays, just break
            break;
        }elseif(strtotime($arr1[$arr1pos]['timestamp']) > strtotime($arr2[$arr2pos]['timestamp'])) {
        // if the timestamp of array1's element is greater than the timestamp of array2's element
            $arr3[$i] = $arr1[$arr1pos];
            $arr1pos++;
        } elseif ($arr1[$arr1pos]['id'] == $arr2[$arr2pos]['id'] ) {
        // if they are the same post (duplicates are introduced by following an author and tags they post in)
            $arr3[$i] = $arr1[$arr1pos];
            $arr1pos++;
            $arr2pos++;
        } else { // if the timestamp of array1's element is less than than the timestamp of array2's element
            $arr3[$i] = $arr2[$arr2pos];
            $arr2pos++;
        }
    }
    return $arr3;
}
```

Figure 2: Merges the given arrays based on the timestamps of their elements. From dashboard.php

Settings

Users have some control over their experience of the website. If users do not want to see Tweets or Flickr photos, they can change their settings to disable these features. Settings are stored in their own table in the database to make them easily extensible – each setting is a column in the table, and records are associated with the user by using their name as the foreign key. Having the value of a setting equal to 1 enables that feature, while having it set to 0 disables that feature. Contributor settings for displaying tweets or Flickr photos are also stored here.

Posts

Posts are the main content of { Tutor }.

Post Display

Posts are displayed throughout the site in two main ways – preview and full view. Preview view displays all of the post's info – with content shortened to just an excerpt - in a condensed form. This allows multiple posts to be displayed on the same page. This form is used throughout the site, on the Lessons page and the profiles of Contributors. The full view is used on the page of the post itself. This displays all the same information, but with full content and a wider display, making the post easier to read.

Post Creation and Editing

Post creation and editing are quite simple; both use a simple form for the user to provide information about the post such as its title, content and tags. The Create Post form is available from the sidebar for

Contributors, and an Edit Post button is visible to the creator of a particular post. Creating a post adds the post entry to the database, while editing the post updates that entry.

Topics

As the focus of this site is content, users should be able to easily find content relevant to their interests. Topics are used for this purpose. Posts can be tagged with one or more Topics - all programming or computer science topics possibly relevant to learners – to reflect the content within that post. These posts are then displayed within the page for that Topic. Topic pages are accessible from the Lessons page. Tweets can also be tagged with Topics, and will also be displayed on the Topic's page. Finally, Learners can follow Topics to have all posts and tweets in a Topic automatically added to their dashboard, allowing them to keep track of all new information posting in that area.

Social Media Integration

Contributors are about to connect their twitter or Flickr accounts to share more information on { Tutor }. This can be used to share programming related content, or just to personalize the Contributor's profile page.

Twitter

Because users could potentially use tweets to convey information about programming topics, tweets are treated fairly similarly to posts on the site. Tweets appear in two places on the site: a Contributor's profile page, and the Topic page of any Topic they are tagged with. If a contributor has provided their Twitter account and enabled tweets, their most recent 5 tweets will automatically be displayed and cached on the site. If a tweet contains the text " #tutortweet " then it is displayed on the Lessons and Topics pages as well as the profile. To add a tweet to a topic, the Contributor just needs to include the name of a topic as a tag, such as " #lua ". While on a Contributor's profile, new tweets are retrieved by AJAX every 30 seconds.

Flickr

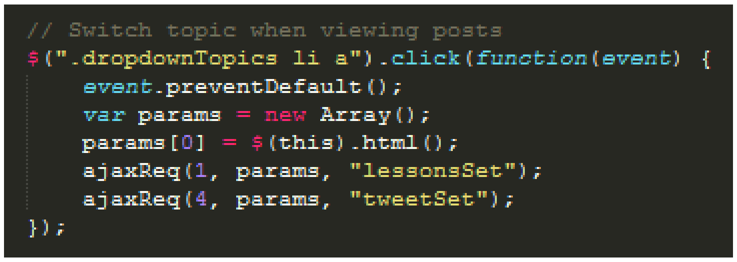
It is less likely that users are going to use Flickr photos to convey programming related information or lessons, so Flickr photos are not as deeply integrated with the rest of the site. They only appear on the pages of Contributors who have connected their Flickr accounts and have Flickr photos enabled in their settings. Their most recent 9 photos are automatically displayed and cached, with the cache expiring after 1 day. The Flickr API is quite slow, so this is done to minimize the number of requests we make to it.

AJAX

This iteration of the project introduced AJAX to update content for users without having to refresh the page. I used AJAX to improve 3 features: Topic navigation, search and tweet fetching.

Posts and Topics

When browsing lessons, users may choose to view only posts tagged with specific topics. In previous versions, the user would be taken to a new page where the posts with the selected tag were displayed. Now, the posts for the selected tag are loaded using AJAX, then replace the existing page content. A click event handler was added to the Topics links, which prevents the default behavior of going to that page, and passes the text of the selected element to the AJAX controller. The controller sends a request to ajaxPosts.php, which gets all of the posts for the selected topic, as well as if the user is currently following that topic or not. The AJAX controller then displays these results, replacing the displayed posts from the previously selected Topic.



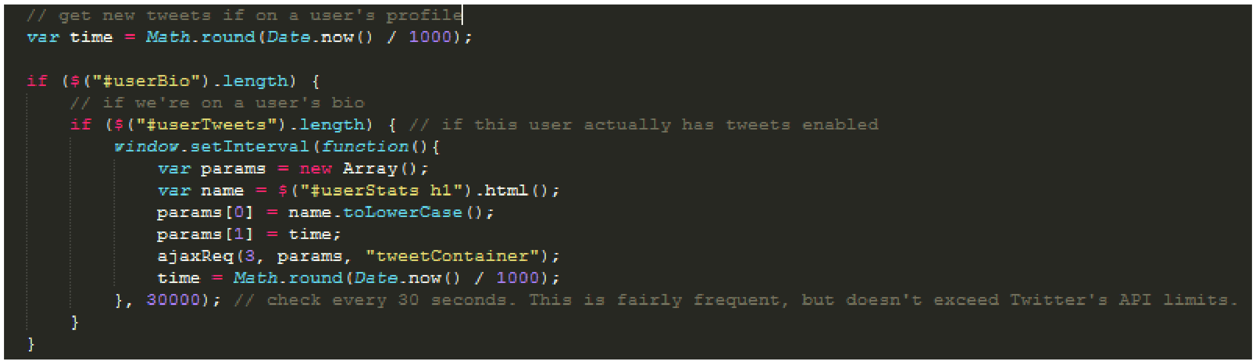
```javascript
// Switch topic when viewing posts
$(".dropdownTopics li a").click(function(event) {
    event.preventDefault();
    var params = new Array();
    params[0] = $(this).html();
    ajaxReq(1, params, "lessonsSet");
    ajaxReq(4, params, "tweetSet");
});
```

Figure 3: Click handler for topic menu. From script.js

Another AJAX request to get the tweets for the selected topic is also sent, and the resulting tweets replace those currently displayed.

Tweets

Unlike the other features that use AJAX, getting new tweets is not triggered by the user. To ensure that the tweets displayed on a user's profile are always their most recent, AJAX is used to connect to the Twitter API and fetch any new tweets every 30 seconds. By using the fetchTweets function in displayTwitter.php, any new tweets are automatically added to the database. Then ajaxTweets.php gets the newest 5 tweets from the database. If the timestamp on the tweet, indicating when it was added to the database, is more recent than the timestamp passed to ajaxTweets.php, that tweet is added to the XML returned. This ensures that we only get new tweets back, which allows for nice JQuery effects on the Javascript side. The tweet is added to the top of the current tweets, and a JQuery effect is used to slide it in.



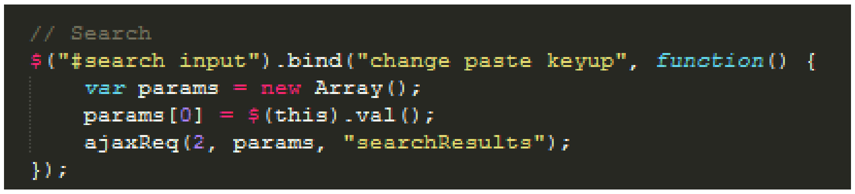
```javascript
// get new tweets if on a user's profile
var time = Math.round(Date.now() / 1000);

if ($("#userBio").length) {
    // if we're on a user's bio
    if ($("#userTweets").length) { // if this user actually has tweets enabled
        window.setInterval(function(){
            var params = new Array();
            var name = $("#userStats h1").html();
            params[0] = name.toLowerCase();
            params[1] = time;
            ajaxReq(3, params, "tweetContainer");
            time = Math.round(Date.now() / 1000);
        }, 30000); // check every 30 seconds. This is fairly frequent, but doesn't exceed Twitter's API limits.
    }
}
```

Figure 4: If on a user's profile, fetch tweets every 30 seconds. From script.js

Search

AJAX is used to allow the user to quickly and easily search for posts on the site, without interrupting their current task or moving them from their current page. A small search bar is displayed at the top every page, within the header. The change, paste and keyup events for this search bar are all bound to the same function handler, which passes the AJAX controller the current value of the search bar whenever the user enters text. The AJAX controller executes the actual AJAX request, sending the user's search query to search.php to be processed. This file executes a simple database query where it looks for posts with titles that contain the user's search query. The title, author and id of these posts are then returned, and the handler displays these search results.



```
// Search
$("#search input").bind("change paste keyup", function() {
    var params = new Array();
    params[0] = $(this).val();
    ajaxReq(2, params, "searchResults");
});
```

Figure 5: Binds the change, paste and keyup events on the search bar to call the search AJAX request. From script.js

Summary of Current Functionality

New additions are bolded

- Users can register on the site as either Contributors or Learners
- Users can log in to the site
- Contributors are able to create posts and tag them with predefined topics
- Contributors are able to edit their own posts
- Contributors can edit their profile (currently just their description)
- Both user types and non-logged in users are about to browse the site
    - All posts can be viewed by selecting the Lessons link
    - All posts from a specific Topic can be viewed
    - The Topic navigation menu on the Lessons page will update whenever new topics or categories are added (this has to be done directly in PHPmyAdmin at the moment)
    - All users can be viewed, either sorted alphabetically or by the topics they write about
        - A user's topics are determined by the topics selected for their posts
    - Individual users' profiles can be viewed, and displays all of their posts
    - Individual posts can be viewed
- Learners are about to follow users and topics
    - Posts either from these users or in these topics will appear on their dashboard
    - Learners are able to unfollow users and topics
    - **Dashboard content now appears in absolute order based on timestamp – previously, posts from followed users were displayed, then topics were displayed**
- Users are able to log out
- Contributors are able to connect their Twitter account through the Edit Profile page
- Contributors are able to connect their Flickr account through the Edit Profile page
- If the user has Twitter enabled, their 5 most recent tweets are displayed
    - The Twitter icon provides a link to their Twitter profile
- If the user had Flickr enabled, their 9 most recent images are displayed
    - A link to their Flickr page is also provided
- If a Tweet is tagged #tutortweet, it is integrated with the rest of the site in a post-like manner
    - Tweets are displayed on the Lessons page
    - Tweets with tags containing topic names, such as #Python, are displayed in that tag
    - Tweets that are not tagged will only appear on the user's page
- Both contributors and visitors can disable or enable Tweets and Flickr photos
    - If these are disabled, Tweets and Flickr photos will not be visible to them while browsing the site
- Contributors can disable or enable posting of their own Tweets or photos
- Tweets and Flickr photos are cached. Currently, the Twitter cache expires after a minute, and the Flickr cache expires after a day
    - This is done because Flickr is quite slow, and users are unlikely to post to Flickr as frequently as they do Twitter
    - When the user's cached Flickr photos exceed 9, the max amount shown on their profile, all older photos are deleted from the database
    - When the user's cached Tweets exceed 5, all older tweets are deleted from the database

    - An error in how Flickr caching was handled has been corrected – the timestamps of a user’s photos are updated to the current time after checking if they have expired. This prevents over checking when no photos have been added for over a day.
- Tweets made by followed users will appear in Learner’s dashboards
- Switching between topics is now done using AJAX. Old posts will fade out and posts from the new topic will fade in
    - Tweets within a particular tag are also shown via AJAX
- Users are able to search the site. The search returns all posts with a title that contains the query string the user entered. This is done using AJAX and a small overlay under the search bar, so users are able to search without affecting their current activities on the site
- On a user’s profile, that user’s tweets are checked for updates every 30 seconds. If new tweets occur, they slide in to join the existing tweets on the page

Future Changes

If I were to continue development of this project, there are quite a few things I would change.

There are many issues with naming that occurred throughout the project, with multiple names being used for the same item. Lessons and posts both refer to the same entity, as do tags and topics. Early in development, these were used interchangeably, which created many issues later on. Future work would have to involve fixing these inconsistencies to improve project structure.

Additionally, I would employ a more strict Model View Controller architecture. Currently, I have one main controller file which calls the other files as needed. This worked quite well and helped keep my code fairly clean. However, I didn't have much separation between processing of data and the creation of the views, and pulling these apart would improve the modularity and readability of my system.

Additionally, there are changes I intended to implement during development but never got around to. These included adding the ability to upload profile pictures and pictures for posts, as well as pagination.