Database and Routing Homework

Create a Database and Populate it with Data.

Achieve User Interface Reuse (JS code Injects Content on Click of Navigation Link)

CONTENTS

1. Overview

In this homework,

- you create database tables, populate them with data, and run some SQL SELECT statements against your data. You'll put screen captures from MySQL WorkBench into a word document to show what you did. The requirements for the database part of this Homework are provided in a separate document.
- you will add interface reuse to your web site - by adding JavaScript code that implements "routing", meaning that when the user clicks on the "Blog" or "Home" links from your navigation bar, "Blog" or "Home" content gets injected into the content area. This style of web design is called "Single Page Application", because your application is primarily stored in a Single Page with only the content area getting changed based on user input.

2. Database Work

For the database part of this homework, please see a separate document entitled "DB Setup".

3. Routing for User Interface Reuse

This week's homework (in the course web page) provides is a zip file with various folders of sample code. The folders show you how to implement your own simple "Routing Framework" using JavaScript.

- It starts out showing you how you could copy index.html to blog.html, change the content area in blog.html and get a pretty good result where you can click back and forth between index.html and blog.html. The links change as you click and the user interface looks good too. The only problem is maintenance. With this simplistic approach, if you had 100 pages in your site and someone wanted to change the title or the footer or a nav link, a programmer would have to make that same change in 100 files. Not good !
- The next sample (Routing) shows routing where your index.html file becomes a template without any actual content. You then create an array of link/content pairs and set up an event handler that invokes a function whenever the active link changes. Based on the active link, you have code that injects the correct content based on the active link. This is a very simplistic example because "the content" being changed is just a character string, not complicated HTML code as would be more realistic.
- The following example (Routing Encapsulated) is the same but pulling out the JS code into a separate file and making it reusable (consumer/producer style) meaning that the code in the JS file does not reference anything in the HTML page, except what has been passed to it as a parameter.
- The next example (Routing Components) uses "components" whereby you have a separate js file, say home.js that runs whenever the home link is clicked. When home.js runs, it injects more realistic (more complex) content. Then there is an encapsulated version of this code also.
- The final example (Routing Components Encapsulated With DropDownMenus) just shows that there is no problem trying to use drop down menus with the encapsulated routing component code.

4. Web Site Homework Requirements

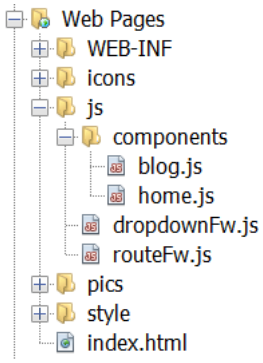
- Your **home and blog links shall work when** clicked, implemented using encapsulated routing components (as shown in the latest code examples mentioned above). This means you'll have a JS file named home.js and another named blog.js that shall be invoked (upon click), injecting the proper content into the content area. When you click on home, you should just see your web site functionality overview enticing users to visit the site. When you click on blog you should see a page that has two blog entries (one for homework 1 and another for homework 2).
- Your site shall continue to meet last week's requirements (drop down menus, images, fixed titleNav and footer, fluid layout, etc.). Check last week's homework if you need to. Improve your layout if you can.
- **Blog content.** Your blog content shall include
    - **In the first blog entry**, describe your web design/development experience and explain what you learned from the first homework (Home Page, last week). Then tell what you found easy about that first HW and what you found hard or confusing.
    - **In the second blog entry**, describe your database experience and link to your word document (that shows your Database/SQL work). To link to the word document, simply place it in your local web root folder, reference the file name in the href attribute of the <a> tag, (test it locally) and then include that file when you transfer the files to the web server for publishing. Tell what was easy and hard/confusing about the Database work.
    - **In the third blog entry**, tell what you learned in the routing/web part of this assignment. Tell what you found easy and hard/confusing.

5. Debugging

While you are not writing a lot of your own JavaScript code in this homework, there is the potential for you to have JavaScript errors. Although you do get context sensitive JavaScript error messages from NetBeans, JavaScript is not compiled, so you don't get compiler errors. You only get runtime errors AND you don't see these runtime errors unless you have pressed F12 in chrome and are looking at the Console tab. Anytime JavaScript has a runtime error, the code just stops executing, so you have to be looking at error messages from the Chrome console.

6. Web Site Homework Requirements

- **Project Organization.** Your project shall be organized as shown below:



- Take advantage of the 3308 Template document that is provided.

7. Submission Requirements

- Make sure your project is organized as shown in the diagram shown in a previous step.
- As usual, test your web site locally (including clicking from home to blog to home, then clicking from blog to the database document) – all links should work properly.
- Then publish, then test all links again.
- Then submit a zip file of your whole web project folder into Canvas.

8. Homework Grading – Sample Deductions

- **-9 for Not Publishing:** We perform functional testing each week based on what you have published. You need to test locally, publish, then test what you published.
- **-9 for Lack of Canvas Submission:** If there is ever any question about a grade, we go by the code that was uploaded into Canvas at the time the assignment was due.
- **Up to -4 for Improperly Designed Database Tables:** We will check that your data model meets all of the requirements listed in the database document (requirements such as data type, PK, FK, null-able, unique). This is very important, since you will lose points in many future homeworks and the project if your data model is not as specified.
- **Up to -2 for Non-Realistic Data:** You were asked to enter realistic data so that your web application looks good when it begins to display data on its pages.
- **Up to -3 for Problems with SELECT Statements** as prescribed in the database document.
- **Up to -3 for Unprofessional / In appropriate Home Page Content:** The "marketing material" from your home page should be of a quality that acceptable by a "real company" that might be paying you to create their web site. The marketing material (what you say your web site will do) must be able to be supported by your choice of layout for your "other" database table.
- **Up to -8 for Lack of Originality of Project and/or Data Model:** As in most homework assignments, points will be deducted if your submission is too similar to the sample(s) provided or to another student in the class.
- **-4 if Routing not implemented:** For example, if a student manages to get their navigation bar to work pretty well by copying and pasting code from an index.html to a blog.html and just linking back and forth from one page to another, without using JavaScript to create a Single Page Application to achieve User Interface code reuse…
- **-1.5 if there is no Database Blog Entry.**
- **-1.5 if there is no Blog Entry about UI Code Reuse (Routing) using JavaScript.**
- **Up to -2 for Look and Feel:** Make sure to meet the requirements for the first homework (Home Page). If you had issues, you should have fixed them by now. For example, you should have drop down menus and nice layouts that do not render slowly (e.g., due to too large image files), should respond well when you narrow and expand the width of the browser, and should meet requirements for your first homework (e.g., all text legible, no text close to visible edges, not overly similar to the sample code).
- **Up to 2 for Syntax Errors:** When we View Source (in Firefox), we should not see any red syntax error messages.