AngularJS Advanced Homework

Overview

In your “AngularJS Intro” homework, you implemented the following functionality (without using AngularJS routing for URL management and UI code reuse):

- index.html: a page that displayed all the records from your “other” database table using Angular’s sortable column heading functionality.
- insert.html: a page (without using any Angular) that inserted a record into your “other” database table using server side validation.

In this lab, you will:

- Add Angular JS routing to what you did last week.
  - Thes means that your project shall have a HTML partial and a JS controller for list and another HTML partial and JS controller for insert.
  - Make sure that all new functionality this week also uses routing.
- Add View functionality, such that each row in your “other” list shall have a View icon.
  - When the user clicks on a View icon, they shall see that “other” record by itself in the content area.
- You DO NOT have to implement update “other” and you DO NOT have to implement delete “other” (as is implemented in this week’s sample code).
  - However, one of the requirements for all labs is that you never have extra/unused code.
  - Make sure that your code has nothing left over from the sample code related to update or delete.
- Your nav bar this week shall consist of the following links:
  - The leftmost link shall be named as you named your “other” database table (not “other”).
  - Then there shall be an insert link (even though your list page may also have an “insert” link on it).
  - Then there shall be a logon link, a logoff link, and a “members only” link – precise functionality shall be defined for this below.
  - Due to time constraints there will be no way for a user to register (insert user record), but since you implemented insert “other” you should know how to do that if you wanted to do this after the class is over.

Database Requirements:

Database requirements are the same as last week, so just be sure you are meeting them.
Project Organization / Design Specifications

All of your work for the Angular Advanced HW shall be stored in a folder named "09_angular_adv".

09_angular_adv/htmlPartials shall contain files with HTML snippets that will be inserted into 09_angular_adv/index.html (based on routing rules), similar to the Angular sample code in "07_routing_for_layout".

Your JavaScript files shall be organized by controller, in 09_angular_adv/js. The index page in 09_angular_adv shall invoke each of these script files (the app.js file first).

One of the js files shall specify routing rules.

09_angular_adv/webAPIs shall store all of your web APIs (JSP pages).

09_angular_adv/index.html shall be the main page (of your Single Page Application) for this homework.

"Source Packages" will contain packages that hold the java classes that are referenced by your Web APIs (JSP pages).

Suggest you copy in and use all the dbUtils classes.

Since you only insert into one table, you should only have one package named model."other" (whatever you named your "other" database table).

The view package shall have two classes: one for your "other" table and one for your user table (code that supports logon).

Right click "Libraries" and add the mysql database driver JAR file and the gson jar file that handles JSON ↔ POJO conversion (POJO: Plain Old Java Object).
A Note About Version Control

- You could copy from 08-angular_into to 09-angular_adv (under “Web Pages”) and then freely make any changes in 09-angular_adv without hurting what you did in 08-angular_intro (since there would be no code reuse between the client side code in these two folders).

- HOWEVER, changes that you make to server side code that is reused between the two homeworks MIGHT cause code in 08-angular_into to stop working properly. SO, BE SURE TO HAVE A GOOD BACKUP of your complete working Angular Intro project. You really should not have any trouble with version control, but if you did, you could (and this is not ideal but) copy your server side APIs (JSP pages) and/or java classes/packages to a new name and have the 09-angular_adv use that (allowing the 08-angular_intro client side code to use the original JSP and java code).

- Remember that you need to have all your source code saved and working properly.
Web API Requirements

This week, you’ll reuse the two Web APIs from last week:

1. `getOtherListAPI.jsp`
2. `insertOtherAPI.jsp`

You’ll need new Web APIs for

3. `getOtherAPI.jsp` (just gets one of your records that matches the specified id).
4. `logonAPI.jsp` shall:
   - Accept a username and password - extract from URL using `request.getParameter()`.
   - If logon was successful, write the username into the JSP session object (which can be checked by the other APIs) and return a `StringData` object (JSON-ized) fully populated with the user’s record (from the user table).
     
     ```java
     session.setAttribute("loggedOnUser", foundUserName); // if credentials are in DB
     ```
   - If logon is unsuccessful, invalidate the JSP session object then return an error message such as "invalid credentials", "database is unavailable" or (while debugging) whatever SQL syntax error message may have been generated by a database exception.
     
     ```java
     session.invalidate(); // if credentials not found in DB
     ```
5. `logoffAPI.jsp` shall require no input. It shall invalidate the JSP session object and then send a confirmation message back to the html page. This is how you invalidate the session in the JSP page:

     ```java
     session.invalidate();
     ```

6. `isLoggedOnAPI.jsp` shall check the JSP session object to be sure the user is logged in before they allowing the user to view some “special message”. This can just be something silly, something to demonstrate the back end refusing to perform work (or show data) to a user who is not logged in.

   ```java
   String loggedOnUserName = (String) session.getAttribute("loggedOnUser");
   ```

As was required last homework, all of your Web APIs that access the database shall provide record (or list) level errors, such as:

- “database is unavailable”,

To be clear, you are being asked to "intercept" the database error and prepend the technically useful database error with a user friendly error such as “Database is unavailable – please try later” or “That name has already been selected by another record, please select another name”.

4
Client Side Requirements

1. **09-angular_adv/index.html**: shall be the "single page" for your triple lab (SPA / single page application). This page shall contain the complete layout of your current home page (top to bottom, no ng-include statements) - except it shall have a placeholder ("ng-view") where Angular Routing will insert content (from a "HTML partial"), similar to the "07_Routing_for_Layout" sample code.

2. The nav bar of triple/index.html shall have these links:
   - **Other**: (but you can’t name the link Other, name it as you named your “other” table in your database) – this page shall list all the data from your “other” database table. Remember that this week you must use routing for this functionality (so you’ll have a routing rule that specifies a list html partial and a list js controller).
   - **Insert**: (allows the user to insert a record into your “other” database table – but remember this week you must use routing, so you’ll need a routing rule that specifies an insert html partial and an insert js controller).
   - **Logon**: (you’ll need a routing rule, a html partial, and a js controller that will call your new logon API).
   - **Logoff**: (you’ll need a routing rule, a html partial, and a js controller that will call your new logoff API).
   - **Members Only**: (you’ll need a routing rule, a html partial, and a js controller that will call your new isLoggedOn API). This API will check if the user is logged on. If so, it can provide a simple message like “welcome member” or if not the message might be “sorry but you must log on to see this page”.
   - **Labs**: this link will get you out of your SPA (Single Page Application) and back to your labs page of your web root folder. This will be a regular link: `<a href="../labs.html">Labs</a>` with no routing rule.

Labs Page / Blog

- Add a blog to your labs page for this week’s homework. Link not only to 09-angular_adv/index.html, but also link directly to each of your Web APIs so that each may be easily tested with URL tampering.

Submission

- As always, publish your code, upload it into Canvas before the deadline, and demonstrate the code to the TA in lab.

  Make sure that you have ONE PROJECT that incorporates all the code from ll the labs – DO NOT have one project per lab! You need to be able to manage version control of all the code that is yours. In the real world version control is much more complicated because multiple programmers are making changes to shared code all the time.

Suggested Approach:

- Please read the suggested approach from last week’s homework (Angular Intro). The approach and tips should be the same. Do not begin with client side code – start with JSP pages (test with URL tampering). Then, once you have fully tested your server side code, start working on your client side code.