Tutorial 5 – JavaScript jQuery Rest Services GUI

The purpose of this tutorial is to introduce how to implement a CRUD (Create, Read, Update, and Data) Web GUI for backend Rest services using JavaScript and jQuery.

JavaScript is an interpreted object-oriented language. Web browsers provide integrated JavaScript interpreters to bring JavaScript code to life creating a dynamic programming environment for web-based user interface development. JQuery is a specialized JavaScript library that adds extra functionality for web programming and in recent years jQuery has become the most popular JavaScript library used in web development.

Many resources abound online with examples for JavaScript, jQuery, and HTML programming.

Here are a few links:

JavaScript:
https://www.w3schools.com/jsref/
https://msdn.microsoft.com/en-us/library/yek4tbz0(v=vs.94).aspx

HTML:
https://www.w3schools.com/TAgs/

jQuery:
http://api.jquery.com/
http://api.jquery.com/jquery.ajax/
https://www.w3schools.com/jquery/
https://oscarotero.com/jquery/
http://www.tutorialspark.com/jquery/jQuery_Ajax_Methods_Reference.php

1. Download and Configure the latest sample_maven_web_app

First, delete or rename your existing sample_maven_web_app git repository by going to github.com using a web browser.

Next, navigate to the sample_maven_web_app repository and fork the latest version into your account to replace the deleted version. In the upper-right had portion of the screen click on the ***"Fork"*** Icon.

Fork:
https://github.com/wjlloyd/sample_maven_web_app

Now, from the command line, create a new directory, change into this directory (cd {new_dir}, and clone your forked code:

```
git clone git@github.com:{your-user-name}/sample_maven_web_app.git
```

From this directory, log into your heroku account:

```
heroku login
```

Then create a new heroku project for this source tree:

Note: you may want to purge old heroku projects since free accounts are limited to (4) projects.

```
heroku create
```

Next, add the postgresql add on to this project:

```
heroku addons:create heroku-postgresql:hobby-dev
```

Now you will want to create tables for this project.
The tables are described in the file: src/sql/create_database.sql.

Running psql on the command line requires installation of the postgresql-client package on Ubuntu: **`sudo apt-get install postgresql-client`**

Alternatively, create tables can be created using the PGAdmin GUI and loading the SQL file.

To create tables using the command line:

```
heroku pg:psql
```

Then:

```
\i src/sql/create_database.sql
```

Next, check that the tables have been created:

To see the list of tables:

```
\dt
```

And the structure of the tables:

```
\d+ public.users
```

```
\d+ public.messages
```

2. Build the project, and test locally

Now, using Netbeans, perform a "Clean and Build" of the project source.

From the command line, launch a local instance of the application using the script file called localrun.sh.

From the sample_maven_web_app project source directory:

```
./localrun.sh
```

This will launch the application locally using your local Apache Tomcat server. It will take several seconds to initialize. The application will be hosted at localhost:8080.

Once launched, you should be able access the Users's GUI at the following URL:

http://localhost:8080/

Next, using the GUI test the CREATE, READ, UPDATE, and DELETE functionality using your browser.

3. Deploy the project, and test the remote deployment

Now, deploy the application to the cloud:

```
git push heroku master
```

Try visiting the web page for your heroku app.
This will be the base URL of your Heroku application.
To check what this is use the following command:

```
heroku apps:info
```

See the "Web URL" parameter.

Now, try out the CREATE, READ, UPDATE, and DELETE operations provided by the Web GUI hosted on Heroku. The application should automatically connect to the same database.

4. Create a new Messages GUI based on the User example. Provide Read functionality

(7 points) Inspect the index.html User GUI example. Now copy the User example and refactor it for use as a Messages GUI. Begin by implementing Read (get) functionality. Update the backend services so they return JSON instead of text as in UserService.java.

Use your messages additions from tutorial 4, or download the extract the following tar gzip archive you're your "sample_maven_web_app" directory:

Download here:
http://faculty.washington.edu/wlloyd/courses/tcss360/examples/messages.tar.gz

From your "sample_maven_web_app" directory:

```
tar xzf messages.tar.gz
```

5. Add support for Creating messages based on the User example

(3 points) Provide support for creating new messages. When new messages are created, the list of messages should be updated.

BONUS EXERCISE: Complete the Web GUI by adding support for UPDATE and DELETE

As an added challenge, complete the Web GUI by adding UPDATE and DELETE functionality.

Receiving Credit for This Tutorial

To receive credit for this tutorial, have the instructor review your working Messages GUI in class, or during office hours, or by appointment.