TCSS 360: Software Development and Quality Assurance Techniques
Winter 2017
http://faculty.washington.edu/wlloyd/courses/tcss360

Institute of Technology
University of Washington – Tacoma

Tutorial 1 – Java Web Services

The main purpose of this tutorial is to install and configure a working Java-based development environment for REST-based webservices projects. It is acceptable to work in pairs for this assignment if you do not have access to a suitable laptop. It is preferred if you work individually so that you gain experience and also configure your development environment.

Prequistires for this tutorial include having already created a GitHub account and a Heroku account.

I recommend installing Linux either on a Virtual Machine, or on your development laptop/computer to facilitate working with free open source development tools more easily.

To install a virtual machine, use Oracle Virtual Box, and Ubuntu 16.04:
Oracle Virtual Box: https://www.virtualbox.org/
Ubuntu 16.04: https://www.ubuntu.com/download/desktop
Down the .iso file, and create a new Linux/Ubuntu machine using the .iso file to install the operating system.

It is entirely acceptable to stick with a Windows based development environment, but TCSS 360 presents an opportunity to learn the Linux stack.

Once you've chosen an operating system, go ahead and install and configure the following tools:

1. **Install a Java IDE**

   We will need an IDE which supports integrated Maven builds. Maven is required to deploy web applications to Heroku. Maven helps to manage all of the software dependencies and helps to define how a web application is built through the use of a pom.xml file.

   I have created a sample Maven project in Oracle Netbeans. Netbeans is the IDE supported by Oracle (formally Sun Microsystems) whom is the curator of Java.

   You could optionally try opening this maven project in Eclipse. This link describes a possible technique to import the project. I have not yet gone down this path. If you are familiar with Eclipse projects and Maven projects this could be an alternative:
   http://stackoverflow.com/questions/21535023/how-to-get-your-netbeans-project-into-eclipse

   Otherwise, since the example is a Netbeans projects, it will be easier to download and Install Netbeans 8.2 to natively open the project.

Download the Java EE version:
https://netbeans.org/downloads/

On Linux, this will download an .sh script file which is best run as root

For the installation, please unselect GlassFish Server, and select Apache Tomcat.

2. Get Project Sources from GitHub, Setup your environment to interact with GitHub

Download a github client that enables you to interact with repositories on github.
If you are using linux, this can be the github packages for Ubuntu, CentOS, Debian, etc.

On Windows/Mac, GitHub provides GitHub Desktop to provide git command-line support:
https://desktop.github.com/

On Windows, this requires installation of msys2, which provides a “linux” like command line environment:
http://msys2.github.io/

Using a web browser, go to this URL:
https://github.com/wjlloyd/sample_maven_web_app

In the upper-right had portion of the screen click on the ***“Fork”*** Icon.
If you are not already signed into github, this will ask you to login first.
This will fork the source code repository and create a local copy of the code in your github account.

If you don't already have an ssh-key, go ahead a create one using ssh-keygen:
`$ssh-keygen`

Please note where this places your id_rsa.pub file.

Next on the command-line add this id_rsa key to the ssh agent so it will provide the key to git hub. Use the command:

`$ssh-add id_rsa`

Copy the contents of the id_rsa.pub file, and add this to github.
Log into github on your browser.

On the upper-right hand corner, select the drop down menu next to your user icon:

Select Settings.
On the left-hand side, select SSH and GPG keys.

Click the button to add a “New SSH Key”

Provide a name for the key. Recommendation is to use something to identify the computer or your rsa key (if you use the same key on multiple machines).

Paste the id_rsa.pub text into the “Key” text box, and click “Add SSH Key”.

Now using the command-line, use should be able to clone the repository on your local machine:

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

You should be able to inspect the “sample_maven_web_app” project files which have been downloaded from your github account.

3. Open project in Netbeans

Try right-clicking on the “sample_maven_web_app” and select “Properties”.
Under “Build” on the left hand menu, click on “Compile”.
Be sure that the Java Platform is “JDK 1.8”.

4. Build and test the project locally

Right-click on “sample_maven_web_app” and click “Clean and Build”

This will begin build and compile the project. Since this is a Maven build, the project dependencies will be downloaded automatically the first time a build is made. They are cached so this step only happens once.

If the build is successful, you should be able to the click on the “Run” icon to run the web app. Running the application performs a local installation onto Apache Tomcat.

5. Deploy to HerokuDeploy to Heroku

You will need to install the heroku command-line environment on Windows/Linux/OS X.

Visit this page for instructions:
https://devcenter.heroku.com/articles/heroku-cli

Once you have the heroku environment installed, to deploy your web service application to heroku:

First log into your heroku account from the command-line:

```
$heroku login
```

Next if you don't already have a project in heroku, then create one:

$heroku create

Finally, go ahead and deploy to your newly created project on heroku.com.
This will take your Java web application archive, and publish it so there is a public endpoint on heroku's PaaS Cloud:
$sudo git push heroku master

Publishing your application to heroku may take awhile. Performance is better on higher speed networks.

Once the application is published to Heroku, heroku will provide a URL for the endpoint, bold and underlined below:

```
remote: -----> Compressing...
remote:        Done: 72.3M
remote: -----> Launching...
remote:        Released v3
remote:        https://enigmatic-retreat-67260.herokuapp.com/
deployed to Heroku

remote:
remote: Verifying deploy.... done.
To https://git.heroku.com/enigmatic-retreat-67260.git
 * [new branch]      master -> master
```

6. Testing Your Service Endpoint on Heroku

Using this endpoint, find the script file at:
{github-root-dir}\sample_maven_web_app\src\scripts\post_test.sh

This is a script to test the POST generic service.

The test uses CURL to act as a lightweight client to POST a JSON object to your service endpoint.

Edit this script.

Update the first line where “ENDPOINT” is defined.

Use the new URL reported by Heroku as the end point.
After the URL, append “/rest/generic”.
This provides the specific path to the endpoint of your service.

Now run this script.
If you're in Linux, and curl is installed, it will just run !

If you're in Windows, and you've installed msys2, then open an msys2 terminal.
Find this in the start menu.

The root DIR of msys2 is probably something like “C:\msys64”.

You will need to copy the script and json object to this directory.

Copy these files to C:\msys64 using your favorite method:
post_test.sh
fred.json

Optionally you could create a new directory under C:\msys64 for test scripts.

Once you msys2 terminal has access to the post_test.sh, go ahead and run this script. This script will send a POST request to your newly published web service running on Heroku!

7. Receiving Credit For This Tutorial

To receive credit for this tutorial, have one person in your group submit the URL of your endpoint in Heroku.

Other group members must submit the name of the person who has submitted the URL.