|  |
| --- |
| QA Consulting |
| Build a development environment |
|  |

|  |
| --- |
|  |

Contents

[Introduction 3](#_Toc503098760)

[Tooling 3](#_Toc503098761)

[Installing Java 3](#_Toc503098762)

[Installing Maven 3](#_Toc503098763)

[Installing Git 3](#_Toc503098764)

[Installing Eclipse 4](#_Toc503098765)

[Installing Server 4](#_Toc503098766)

[Installing Postman 5](#_Toc503098767)

[Extra activities 5](#_Toc503098768)

[Encapsulating a new feature 5](#_Toc503098769)

# Introduction

This is a team exercise for everyone to build a development environment in preparation for the JavaEE module. You have an agreed amount of time to get everyone to the same stage and have your individual development environments ready, using the following instructions.

# Tooling

All tooling for this exercise can be found @ C:\LocalInstall.

There are various tools that can aid building a development environment. For example, RapidEE. As a developer, feel free to use any of this tooling during the task.

# Installing Java

1. Execute the Java Installer. If successful, you should get Java execution messages when running the command ‘java’ from any shell.

# Installing Maven

1. In the following folder C:\Users\Admin create a new folder to store external applications in. Unzip Maven into this new folder.
2. Add maven to the system path and also create system variables for Java and Maven. Hint (JAVA\_HOME & M2\_HOME)
3. If successful, you should get Maven ‘BUILD FAILURE’ message when running the command ‘mvn’ from any shell.

# Installing Git

1. Execute the Git installer and accept all default values. If successful, you should get git execution messages when running the command ‘git’ from any shell.
2. Fork the following repo into your own Github profile: <https://github.com/scrappy1987/sampleapp>
3. Clone this repository onto an appropriate place within your C: drive. (Hint this should be in the admin user folder)

# Installing Eclipse

1. In your new external applications folder unzip eclipse and execute this program. Import the sample application project into eclipse as a maven project.
2. Build the application artefact by running the project as ‘Maven install’. If an error is encountered then solve this problem. If successful, you should get a ‘Build Success’ message. In the target directory a new file should also be created called ‘sampleapp.war’ (Hint open this file with 7zip to observe it’s content)

# Installing Server

1. In your new external applications folder unzip wildfly-10.1.0.Final.zip.
2. Inside the wildfly folder there is several import directories. The key ones we will be working inside are wildfly-10.1.0.Final\bin (binary directory) and wildfly-10.1.0.Final\standalone\deployments (deployments directory).
3. Into the deployments directory add the ‘sampleapp.war’ file created in the installing eclipse stage. In the binary directory execute the batch script called ‘standalone.bat’
4. You have now deployed the sample application to a real application server. If this has been successful you should be able to access this application and interact with it by visiting the following URL: <http://localhost:8080/sampleapp>

# Installing Postman

1. Execute the Postman exe file from the Misc Folder.
2. Later we will be working a lot with REST. Using the Postman client access the following URL: <http://localhost:8080/sampleapp/rest/json>. If successful, you should get the following JSON object returned.

{

“result”: “Hello World!”

}

# Extra activities

The following are extra activities and should only be completed if you’ve managed to complete all the above sections as a team.

# Encapsulating a new feature

In the following exercise we will mimic creating a new feature within the source code for the sample application.

1. Within the GitHub web client create a new branch in the fork of the sampleapp repository called ‘feature1-branch’.
2. Within eclipse pull this new branch into the IDE and switch to this branch as a new local branch.
3. In the following file ‘HelloWorld.java’ edit the source code so that your name is produced instead of world for both the json and xml method.
4. Rebuild the war by executing ‘Maven Install’
5. Delete the old war out of the deployments folder and add the new rebuilt war into this directory. Access the following URL: <http://localhost:8080/sampleapp/rest/json> should now produce the following result:

{

“result”: “John”

}

1. If you have got the expected result push this change to the remote github branch. The new change should now be in the sampleapp branch. If this is correct, merge this branch with master using a pull request (PR) and enter within the comment section “Change to edit from hello world to hello [user’s name]”. Finally merge this branch with Master and delete the old branch.