Consultant Tracker Testing Policy

Edited By

Sibekezelo Mamba 16095414
Johan de Waal 16155140
Stephen Munro 16024479
Hulisani Mudimeli 16073364
Ngonidzashe Mujuru 16285256
Tatenda Mafunga 16094965

University of Pretoria
2018

Contents

1 Introduction

Jenkins CI (Continuous Integration) improves development process by automatically building and testing code modifications which in turn provides instant response or feedback on the status of the modification.

Jenkins CI is linked with our GitHub repository that contains our project source code. Jenkins was chosen because it makes use of continuous integration which is a development practice that joins small code alterations or modifications regularly instead of merging a large change when development cycle is completed.

2 Overview

Jenkins CI use its own system to perform tests and also has access to the localhost of the machine running the automated tests (Runs locally). The automation of executions does not depend on the speed or performance of an individual's machine. Results of the tests are sent via email and shown in the Jenkins job status section.

3 Variances

Multiple conditions and events were detected during the test. This includes the condition of having to delete something that never existed. Jenkins helped integrate the test steps to avoid that risky condition from happening. The test cases below will elaborate further on how this was solved.

Most test cases were successful only when the tomcat server and the database server where running, otherwise, it would produce an error because of failure to communicate with the database.

4 Assessment

Java was used for back-end by the implementation of servlets that exchanges data with the front-end of our system. Individual units of source code of the system, associated with usage methods, data control and operating procedures will be tested using Unit Testing to ensure that the code works as it is intended to.

Our test executed by Jenkins makes use of an Http Url Connection and then uses the Url connection that has been established by tomcat as Odata Service.

Testing Framework: Jenkins Continuous Integration
Module Tested: Java Test Files that in turn tests the servlets
Test Cases:

a) Creating Projects
a) Deleting Projects
b) Adding Consultants
c) Removing Consultants
d) Adding Tasks
a) Removing Tasks
a) Posting Feedback
a) Viewing Feedback

5 Results

The following results shows the executed test cases' output and shows that all test cases have passed the test. All test cases used separate java files to test the individual parts of the system. The computer connects tomcat, the database and jenkins workspace, to separate servers and then jenkins simply use the odata service that has been established by tomcat and the database to return results.

5.1 Test Case 1: Creation and deletion of project

A project is created using our system that runs on tomcat. We use this platform to test if indeed a project has been created successfully created or not. After the execution of test case one, the database is populated with an extra project which symbolizes success. This test case was successful only when the tomcat server and the database server where running.

5.2 Test Case 2: Adding and Removing Consultants to a project

A new consultant is added using our system that runs on tomcat server. We use this platform to test if indeed a consultant has been succesfully added or not. Test Case 2 requires Test Case 1 to be successful so that consultants can be added to a certain Project. Upon execution of test case 2, a certain consultant was added to it. This test case was successful only when the tomcat server and the database server where running.

```
projectsTest:
     [echo] Test Case 1 executing...
     [java]
     [java] Sending 'GET' request to URL : http://localhost:8080/Consultant-Tracker/emplist.svc/Projects?$format=json
     [java] Response Code : 200
     [java]
     [java] There are 6 Projects read from the database, which include:
     [java]
     [java] 1. Project_OnSite: false| Project_Description: Building an online store for Dougs Store| Project_Deadline:
/Date(1526248800000)/| Project_ID: 1| Project_Deleted: false| Client_ID: 1| Project_Name: Dougs Online Store
     [java]
     [java] 2. Project_OnSite: true| Project_Description: awdawda| Project_Deadline: /Date(1526940000000)/| Project_ID: 2|
Project_Deleted: true| Client_ID: 2| Project_Name: awdad
     [java]
     [java] 3. Project_OnSite: false| Project_Description: An software system that is able to monitor financial data and report on
suspicious activity.| Project_Deadline: /Date(1537912800000)/| Project_ID: 3| Project_Deleted: false| Client_ID: 2| Project_Name:
Fraud Detector
     [java]
     [java] 4. Project_OnSite: true| Project_Description: Random project for bob| Project_Deadline: /Date(1526940000000)/|
Project_ID: 1008| Project_Deleted: false| Client_ID: 2| Project_Name: Bobs Project
     [java]
     [java] 5. Project_OnSite: true| Project_Description: awdaw| Project_Deadline: /Date(1526421600000)/| Project_ID: 1009|
Project_Deleted: true| Client_ID: 2| Project_Name: awda
     [java]
     [java] 6. Project_OnSite: true| Project_Description: Keep track of stock levels of products for the business.| Project_Deadline:
/Date(1526421600000)/| Project_ID: 1010| Project_Deleted: false| Client_ID: 2| Project_Name: Tatenda's StockTracker
     [echo] Test Case 1 finished...

BUILD SUCCESSFUL
Total time: 2 seconds
[Consultant-Tracker] $ cmd.exe /C '""C:\Program Files\apache-ant-1.10.3\bin\ant.bat"' consultantsTest && exit %%ERRORLEVEL%%"
Buildfile: C:\Program Files (x86)\Jenkins\workspace\Consultant-Tracker\build.xml
```

Figure 1: Creation and Deletion Project

```
consultantsTest:
     [echo] Test Case 2 executing...
     [java]
     [java] Sending 'GET' request to URL : http://localhost:8080/Consultant-Tracker/emplist.svc/Consultants?$format=json
     [java] Response Code : 200
     [java]
     [java] There are 5 Consultants read from the database, which include:
     [java]
     [java] 1. Consultant_Surname: Peters| Consultant_email: JP@CodeDynamic.com| Consultant_Admin: 0| Consultant_Cell: 0835631245|
Consultant_ID: 1| Consultant_Name: James
     [java]
     [java] 2. Consultant_Surname: Van Wyk| Consultant_email: Bob@CodeDynamic.com| Consultant_Admin: 0| Consultant_Cell: 0648547845|
Consultant_ID: 2| Consultant_Name: Bob
     [java]
     [java] 3. Consultant_Surname: Fredrickson| Consultant_email: DylanF@CodeDynamic.com| Consultant_Admin: 0| Consultant_Cell:
0795486352| Consultant_ID: 3| Consultant_Name: Dylan
     [java]
     [java] 4. Consultant_Surname: Douglas| Consultant_email: Fred@CodeDynamic.com| Consultant_Admin: 1| Consultant_Cell: 0769493806|
Consultant_ID: 1007| Consultant_Name: Fred
     [java]
     [java] 5. Consultant_Surname: Less| Consultant_email: Doug@CodeDynamic.com| Consultant_Admin: 0| Consultant_Cell: 07694953687|
Consultant_ID: 1051| Consultant_Name: Doug
     [echo] Test Case 2 finished...

BUILD SUCCESSFUL
Total time: 1 second
[Consultant-Tracker] $ cmd.exe /C '""C:\Program Files\apache-ant-1.10.3\bin\ant.bat"' tasksTest && exit %%ERRORLEVEL%%"
Buildfile: C:\Program Files (x86)\Jenkins\workspace\Consultant-Tracker\build.xml
```

Figure 2: Addition and Removal of Consultants to Project

5.3 Test Case 3: Adding and Removing tasks

Certain tasks are assigned to a group of consultants. We use the system to integrate this test. Test Case 3 requires Test Case 2 to be successful so that a specific

consultant can assigned to a task. Upon execution test case 3, consultants had been assigned to different tasks and could be removed. This test case was successful only when the tomcat server and the database server where running.

```
tasksTest:
     [echo] Test Case 3 executing...
     [java]
     [java] Sending 'GET' request to URL : http://localhost:8080/Consultant-Tracker/emplist.svc/Tasks?$format=json
     [java] Response Code : 200
     [java]
     [java] There are 4 Tasks read from the database, which include:
     [java]
     [java] 1. Description: jhguh| Due_Date: /Date(1525557600000)/| Task_ID: 1| Name: Project Planning
     [java]
     [java] 2. Description: do the things| Due_Date: /Date(1526508000000)/| Task_ID: 2| Name: Build Conceptual Design
     [java]
     [java] 3. Description: null| Due_Date: /Date(1529791200000)/| Task_ID: 3| Name: Plan Project
     [java]
     [java] 4. Description: null| Due_Date: /Date(1530309600000)/| Task_ID: 4| Name: Build Prototype
     [echo] Test Case 3 finished...

BUILD SUCCESSFUL
Total time: 0 seconds
[Consultant-Tracker] $ cmd.exe /C '""C:\Program Files\apache-ant-1.10.3\bin\ant.bat"' feedbackTest && exit %%ERRORLEVEL%%"
Buildfile: C:\Program Files (x86)\Jenkins\workspace\Consultant-Tracker\build.xml
```

Figure 3: Assigning of tasks to Project

5.4 Test Case 4: Posting and Viewing of Feedback

Test Case 4 requires Test Case 3 to be successful so that through tasks, administrators and consultants can post to one another. This test case was successful only when the tomcat server and the database server where running.

```
feedbackTest:
     [echo] Test Case 4 executing...
     [java]
     [java] Sending 'GET' request to URL : http://localhost:8080/Consultant-Tracker/emplist.svc/Feedbacks?$format=json
     [java] Response Code : 200
     [java]
     [java] There are 3 Feedbacks read from the database, which include:
     [java]
     [java] 1. Feedback_ID: 1| Message: The scope needs to be clerified with the client regarding some ambiguity we have encountered.
     [java]
     [java] 2. Feedback_ID: 2| Message: Rough plans are complete, however we need to find a solution to a minor difficulty
     [java]
     [java] 3. Feedback_ID: 10| Message:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
     [echo] Test Case 4 finished...

BUILD SUCCESSFUL
Total time: 1 second
Finished: SUCCESS
```

Figure 4: Feedback

6 Evaluation

All four test cases for Consultant Tracking went through extensive automated testing by the help of the testing tool, Jenkins CI. After the execution of all this four test cases, errors would have occurred only when trying to remove a consultant that was never created and when trying to delete a project that never existed. But our automation process was programmed to follow the no erroneous steps such that no consultant or project is removed without their existence. Therefore, no errors where produced by our automated testing

7 Summary of Activities

The automated testing execution only took, 12 seconds to run all the configurations that have been specified in the Project Configuration section.

Figure 5: Testing in progress

8 Conventional Quality Metrics

- Requirements Unambiguity - This was carried out to ensure that the entire team was certain of the requirements of the team. Each team member read the system requirements, then explained what they understood from them. The number of requirements that the team members were able to give the same interpretations for (Ns) was compared to the total number of requirements (Nr).

- Fan-In - Each function was checked to see the number of functions and or other servlets that called it. The highest fan-in number is 4 (refresh data). This is relatively low and means that the project does not have a single point of failure.

- Fan-Out - Each function was checked to see the number of functions and or other servlets that it called. The highest fan-out number is 4 (made from refresh data). This is relatively low and is a strong indication that the project modules have low coupling.

- Lines of Code - Each functions lines of code were noted. The maximum number is 56. Modules of code should have a minimum number of lines possible so that they are easier to test, maintain and understand.