Continuous integration Environment
Group 22

BTH

Sai Chand Paruchuri, 9505220096
Neeraj Bala ,20010702T057
Lahari Gaddam, 20000917T083
Marcin Pawlowski, 9503236813

14th of March 2022

Introduction

Our group has worked on implementing a Continuous delivery environment to develop a calculator application. While working on this project we have also used Test-Driven Development.

The Environment

To achieve our goals we had to decide on what tool we will be using. We ended up with the following suite of tools:
For developing the main application we are using Java and using IntelliJ Idea as our IDE.
As a repository for our code, we are using GitHub. To push and pull the code from the repository we employed the use of TortoiseGit. GitHub is commonly used as a repository of code and most of us are already familiar with it. TortoiseGIT offers us an easy-to-use graphical user interface compared to other solutions for using Git which often require the use of PowerShell.
GitHub repository: https://github.com/Laharigaddam/Calculator_St_project
For the automatic building of Java code, we use Maven. We decided on this solution as it's well documented and integrates well with other tools that we are using.
For writing tests we use JUnit. There aren't any other alternatives that are usable and it's well supported and works out of the box with Maven.
For remote building and testing, we have set up a Jenkins server. It is a very robust solution with plugins allowing it to work with any setup. Configuring it is also relatively easy. For us, it works well with Maven and GitHub. Each time a new commit to GitHub is made Jenkins pulls new files from the repository and uses Maven to make a new build and run tests.
This is the extent of our environment at the moment. There are many possibilities to extend its functionality and integrate more tools, for example, tools used for system testing, but we have not managed to implement it for this project.

How our environment works:
First, we use TortoiseGit to get the latest version of the code from the repository. After that using IntelliJ we work on the application's code and if we are implementing a new feature we write new tests first using JUnit. Once the code is ready to be submitted we use TortoiseGit to commit and push the change to the repository. Once the code makes it to GitHub, Jenkins has a hook connected to it that informs it about a new commit. This triggers Jenkins to scan for changes in the repository and pull them. Once that is done Jenkins uses Maven to build the code and execute the tests we have written. After that is done Jenkins flags whether the build is successful or if some tests failed. Jenkins gives us a log of tests that have failed.

Discussion

Setting the environment took us longer than we had initially expected. None of us had any real experience with this kind of work. We had to research and try out some solutions we were considering. Our initial tools haven't survived until the end of the project. We were all trying to work out the solution and how to implement it but we often run into issues for one person and sometimes all of us. We changed from using Eclipse to using IntelliJ as Eclipse proved problematic and overbearing in some aspects, not allowing us to work as easily as we would like. Implementing the whole setup for everyone was also a big challenge. We would often run into different issues and we weren't able to

fully fix all of them. Because of this, we have one Jenkins server setup with one of our group members. Others had troubles with either installing it or having it run as it was giving us odd errors we didn't manage to fix in a timely manner. Nonetheless, we have a working setup. The only issue is that we have to use our one group member as proxy access to Jenkins as we weren't able to set website access to it for everyone. Apart from Jenkins everyone has all the tools setup properly and can use the whole environment by pushing the code to the GitHub repository. We recorded Marcin pushing a code change and Lahari showing the new change being taken from the repository by Jenkins to build and test it.

While developing this project we tried to follow Test-driven development as best we could. We wrote at least unit tests as a start and then we worked on the project properly. One thing we haven't done here properly is record every instance of having the tests before implementing the code but we recorded an example of us implementing a new feature from scratch. We have also included a screenshot of the contributions to the project by our group members. To avoid any confusion, the names in the screenshot correspond to the following people:
Mapl21 - Marcin Pawlowski
Laharigaddam - Lahari Gaddam
Sapa17 - Sai Chand Paruchuri
NeerajBala0207 - Neeraj Bala

Group member experiences

Neeraj Bala

It is my first experience using such environments as Jenkins, IntelliJ, etc. It was fun and I have learned a lot from doing this project. My challenges are setting up the environment and its path. I got errors while I was cloning the project and while committing. But still, it made me happy to complete this with a new concept. I thank my teammates that they really helped a lot by solving my errors. This project helped me to learn new things. The project took me 28 hrs of my work and environment setup is about 15hrs

Lahari Gaddam

The project was interesting and a little tricky since I have tried using many features and tools to build the CI environment, but this setup was set after several trials. Setting Jenkins takes more time when compared to remaining. Through this project, I have learned to build an automated testing CI Environment which helps to expand my knowledge. I even learned how to use tortoiseGit, IntelliJ which is very helpful. I got to know how to write the test cases for an application/program. I thank all my teammates for supporting me to complete the project successfully. I have spent nearly 28 hours to complete the project.

Marcin Pawlowski

Setting up the whole CI environment was a new and rather difficult experience as before I only used it once set up by a different team. I learned a lot from this experience and troubleshooting skills came in very handy. I was sort of overseeing the whole project, I helped with setting up the whole environment, helping and fixing issues that we ran into where I could. I had the most experience programming and took the lead in this department. I was helping others with coding and keeping with TTD to the best of our ability and trying to have our tests follow best practices as much as possible. The project took about 35 hours of my work. With the setup of the environment taking about 20 hours.

Sai Chand Paruchuri

Even though this is my first time using IntelliJ Idea and Jenkins, It felt good getting to know new environments for a change of pace. Most of the challenges I faced so far in this project are setting up the Jenkins server and implementing the integration testing code. Coming to implementation for test cases and mainly focused on adding as many test cases as possible in a way that they won't overlap and are efficient covering the below points

- Check the arithmetic operations are being carried correctly or not - +, -, *, /.
- Verifying BODMAS (Bracket Of Division Multiplication Addition Subtraction) rule in situations where complex operations need to be carried out.
- Verify whether the calculator output matches the expected output or not when using decimal or fractions.
- Verify the input limit and output limit that can be shown on the calculator.
- Integration test cases are implemented in a way that the outputs of single operations are being given as input to different nested operations and checking the expected and actual results matches or not.

So far I spent approximately 33 hours of which I focused mainly on test cases and their optimization, being the contact of the group.

Test case scenarios for JUnit testing

Test Case No.	Description	Test Data	Expected Result	Actual Result	Status (pass/fail)
1	**add _test** To check addition between the input works.	12,9 3,-2 -12,9 -2,-2	21 1 -3 -4	21 1 -3 -4	P P P P
2	**Add_FailTest** To check if addition between two inputs does not work.	21,2 -1,2 21,-2 -1,-2	**Unexpected Result** 40 3 23 -1	23 1 19 -3	P P P P
3	**sub_test** To check if subtraction between two inputs works.	2,1 1,2 1,-1 -500,-499	1 -1 2 -999	1 -1 2 -999	P P P P
4	**Substract_FailTest** To check if subtraction between two inputs does not work.	2,-1 1,-2 1,-1 -500,-499	**Unexpected Result** 1 -1 0 -999	3 3 2 1	P P P P
5	**mult_test** To check if multiplication between two inputs works.	27,3 -1,-3 27,-3 -1,3	81 3 -81 -3	81 3 -81 -3	P P P P
6	**Multiply_FailTest** To check if multiplication between two inputs does not work.	27,3 -1,-3 27,-3 -1,3	**Unexpected Result** -81 -3 81 3	81 3 -81 -3	P P P P
7	**Div_test** To check if division between two inputs works .	81,3 3,-3 -81,3 -3,-3	27 -1 -27 1	27 -1 -27 1	P P P P
8	**Divide_FailTest** To check if subtraction between two inputs does not work.	81,3 3,-3 -81,3 -3,-3	**Unexpected Result** -27 1 27 -1	27 -1 -27 1	P P P P

9	**SqrRT_Test** To check if input square root works.	4 9	|2| |3|	|2| |3|	P P

Test case scenarios for Integration testing

Test Case No.	Description	Test Data	Expected Result	Actual Result	Status (pass/fail)
1	**IT_Add_Sub** Integrating subtraction output as an input into addition	(2,1),2 (2,-1),2 (-2,1),2 (-2,-1),-2	3 5 -1 -3	3 5 -1 -3	P P P P
2	**IT_Sub_Add** Integrating addition output as an input into subtraction	(2,1),2 (2,-1),2 (-2,1),2 (-2,-1),-2	1 -1 -3 -1	1 -1 -3 -1	P P P P
3	**IT_Mul_Add** Integrating addition output as an input into multiplication	(2,1),2 (2,-1),2 (-2,1),2 (-2,-1),-2	6 2 -2 6	6 2 -2 6	P P P P
4	**IT_Mul_Sub** Integrating subtraction output as an input into multiplication	(2,1),2 (2,-1),2 (-2,1),2 (-2,-1),-2	2 6 -6 2	2 6 -6 2	P P P P
5	**IT_Mul_Div** Integrating division output as an input into multiplication	(2,1),2 (2,-1),2 (-2,1),2 (-2,-1),-2	4 -4 -4 -4	4 -4 -4 -4	P P P P
6	**IT_Div_Add** Integrating addition output as an input into division	(2,2),2 (7,-1),2 (-2,10),2 (-2,-10),-2	2 3 4 6	2 3 4 6	P P P P
7	**IT_Div_Sub** Integrating subtraction output as an input into division	(2,4),2 (2,-6),2 (-13,1),2 (-2,-8),-2	-1 4 -7 3	-1 4 -7 3	P P P P
8	**IT_Add_Sub_Mul_Div** Testing all basic operations together.	((((6,2),2),5),2) ((((6,-2),2),-5),2)	3 1	3 1	P P