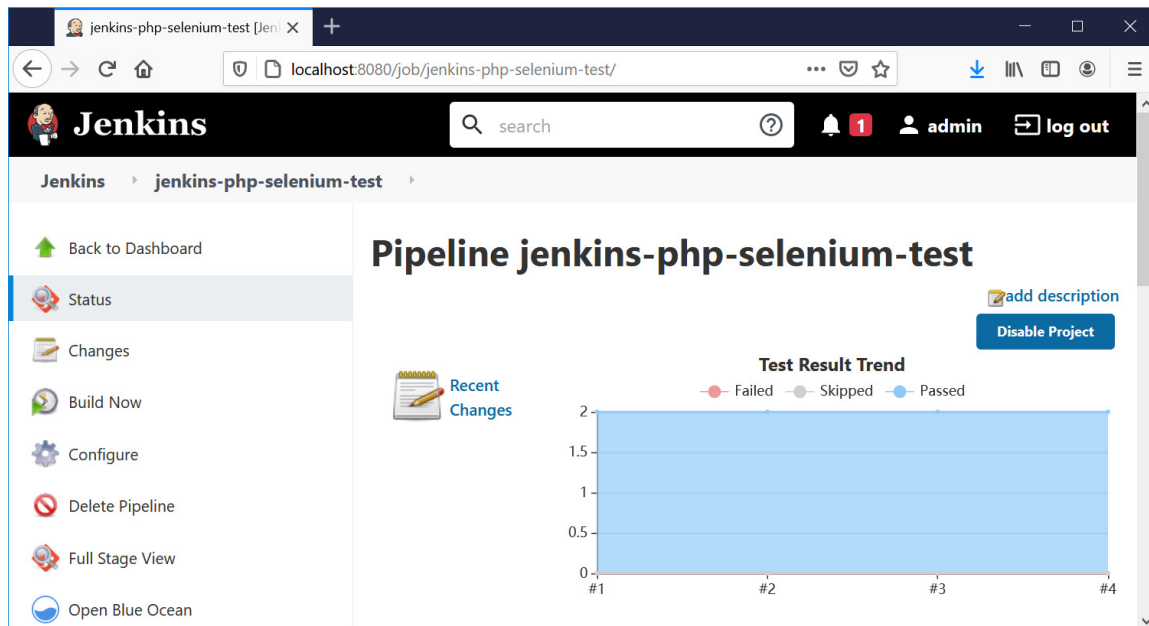


Integrating Jenkins with Automated Integration and UI Testing

Overview



Recall that the Agile testing pyramid recommends not only automated unit testing, but also suitable amount of integration and UI testing although with lesser amount of automation.

You may be wondering how to perform automated integration and UI testing where user interactions are required. The answer lies with the use of headless browser which is a browser without GUI. Specifically, one of the popular tools available is Selenium coupled with WebDriver such as HtmlUnitDriver, FirefoxDriver or ChromeDriver, etc.

To help you, this Lab 7b will guide you on the use of Selenium and HtmlUnit to perform automated integration and UI testing so that you are able to start incorporating it into your team project.

Outcomes

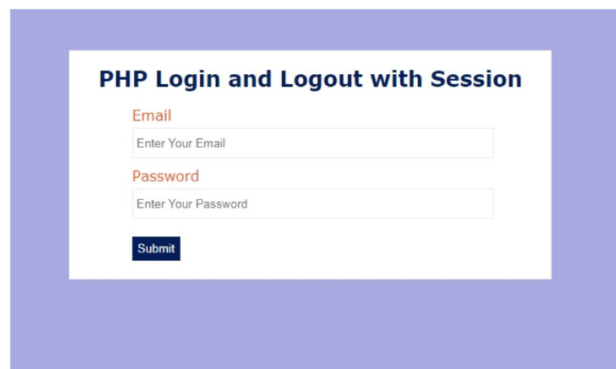
Upon completion of this lab, you should be able to:

- Write test cases for automated integration and UI testing using Selenium with HtmlUnit
- Create Jenkins pipeline to perform automated integration and UI testing on a simple PHP application
- Start incorporating Jenkins Pipeline with automated testing into your team project

1: Pre-requisite

PHP

We'll use the PHP Login and Logout with Session example from <https://www.wdb24.com/session-in-php-example-for-login-logout/> but the original codes seem to have some bugs. So, the codes are corrected, simplified with one pair of hard-coded email and password for one user, and are zipped up inside the provided attached file `jenkins-php-selenium-test.zip`.



Git

Besides the PHP app, the attached file `jenkins-php-selenium-test.zip` also contains a Jenkinsfile and an AppTest.java file with example test cases for automated integration and UI testing using Selenium with HtmlUnit.

Unzip and initialize it as a Git repository using the command:

```
git init
```

Change the access rights of the two accompanied script files so that they can be run by Jenkins:

```
git update-index --chmod=+x jenkins/scripts/deploy.sh
```

```
git update-index --chmod=+x jenkins/scripts/kill.sh
```

Inside the script file `deploy.sh`, update the path `c:\\...\\jenkins-...` as shown below to correspond to the location of your Git repository:

```
docker run -d -p 80:80 --name my-apache-php-app -v c:\\...\\jenkins-php-selenium-test\\src:/var/www/html php:7.2-apache
```

Next, commit with the commands:

```
git add .
```

then

```
git commit -m "Add initial files"
```

2: Integrating Automated Integration and UI Testing into Jenkins Pipeline

Create new Jenkins Pipeline

Create a new Pipeline for the app in Section 1. If you need help, refer to Lab X05.

The Jenkinsfile

Notice that the Jenkins pipeline is written with two stages running in parallel, where the stage('Deploy') is to deploy the PHP server whereas the stage('Headless Browser Test') will perform the integration and UI test when the server is running.

```
pipeline {
  agent none
  stages {
    stage('Integration Test') {
      parallel {
        stage('Deploy') {
          agent any
          steps {
            sh './jenkins/scripts/deploy.sh'
            input message: 'Finished using the web site? (Click "Proceed" to continue)'
            sh './jenkins/scripts/kill.sh'
          }
        }
        stage('Headless Browser Test') {
          agent {
            docker {
              image 'maven:3-alpine'
              args '-v /root/.m2:/root/.m2'
            }
          }
          steps {
            sh 'mvn -B -DskipTests clean package'
            sh 'mvn test'
          }
          post {
            always {
              junit 'target/surefire-reports/*.xml'
            }
          }
        }
      }
    }
  }
}
```

Run Jenkins Pipeline with Integration and UI Test

- a) Before running the Jenkins pipeline, follow the path `/home/src/test/java/mycompany/app` and open using a text editor the `AppTest.java` to change the variable `url` currently pointing to `localhost` to the actual allocated IP address (not `127.0.0.1`) of your laptop (because the PHP app seems not accessible via `localhost` or `127.0.0.1` within docker even though it is running locally).

```
/**
 * Integration UI test for PHP App.
 */
public class AppTest
{
    WebDriver driver;
    WebDriverWait wait;
    String url = "http://localhost";
    String validEmail = "user@example.com";
    String validPassword = "password1234";
    String invalidEmail = "none@example.com";
    String invalidPassword = "password";

    @Before
    public void setUp() {
        driver = new HtmlUnitDriver();
        wait = new WebDriverWait(driver, 10);
    }
}
```

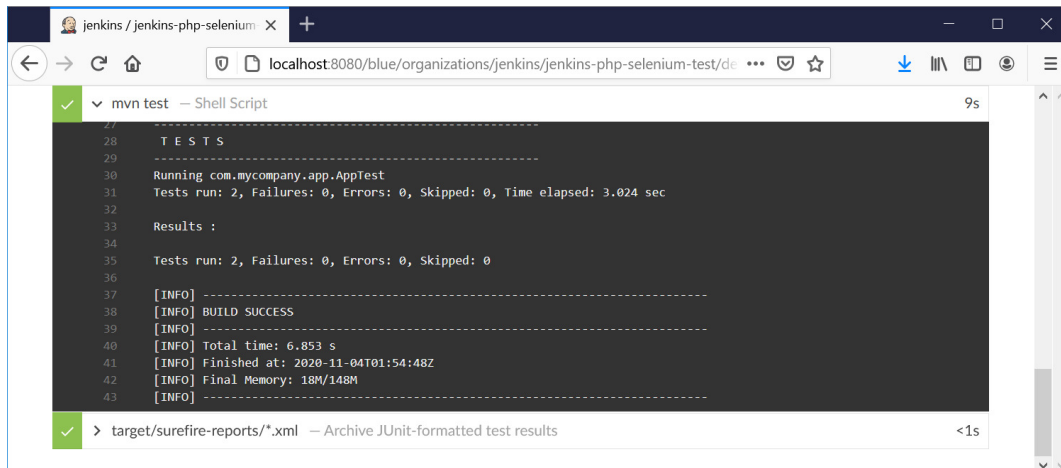
- b) After updating, go through the example codes to have a feel of how to write integration and UI testing using Selenium with HtmlUnit. The url for the available Selenium and HtmlUnit APIs can be found in Section 3 References.
- c) Next, run the Jenkins pipeline where you will notice that the PHP server is being deployed as shown:

The screenshot shows the Jenkins web interface in a browser window. The address bar indicates the URL is `localhost:8080/blue/organizations/jenkins/jenkins-php-selenium-test/de`. The page title is "jenkins-php-selenium-test < 4". The interface shows a pipeline diagram with three stages: "Start", "Integration UI Test", and "End". The "Integration UI Test" stage is expanded, showing a sub-stage "Deploy" which is currently running. Below the diagram, a table lists the steps of the "Integration UI Test / Deploy - 9s" stage:

Step	Duration
✓ > Check out from version control	<1s
✓ > ./jenkins/scripts/deploy.sh — Shell Script	4s
II > Finished using the web site? (Click "Proceed" to continue) — Wait for interactive input	7s

At the top of the expanded stage, there is a "Restart Integration UI Test" button and a download icon.

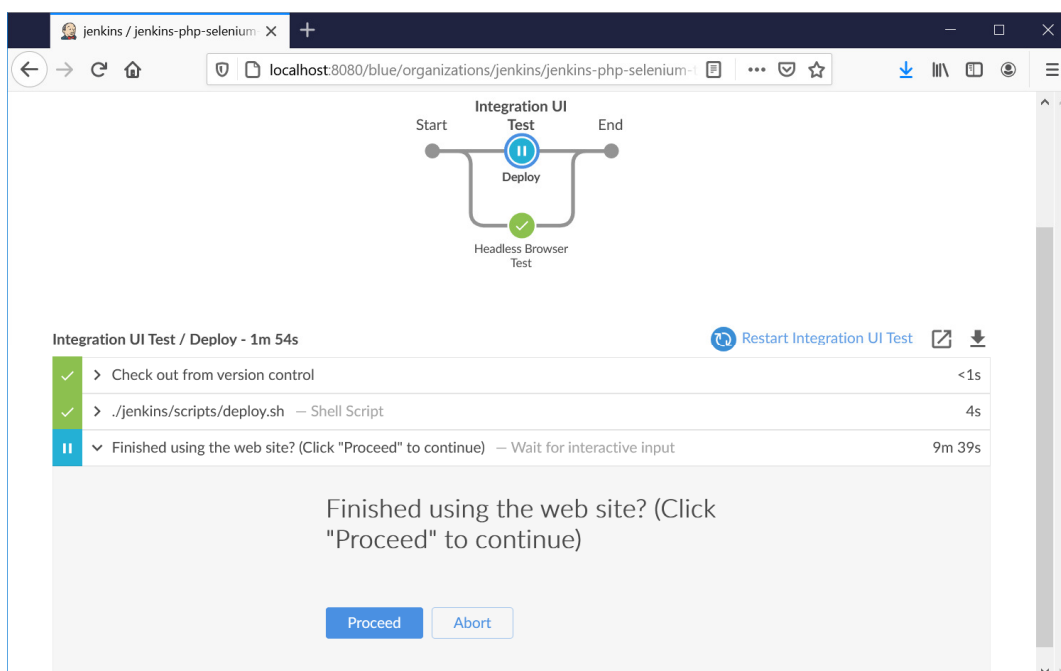
- d) While the test is running, you may wish to open up a web browser to visit the web server at <http://localhost> to have a feel of how the PHP app works.
- e) After a while, the integration and UI test should pass as indicated:



```

jenkins / jenkins-php-selenium
localhost:8080/blue/organizations/jenkins/jenkins-php-selenium-test/de...
✓ mvn test — Shell Script 9s
27
28  T E S T S
29  -----
30  Running com.mycompany.app.AppTest
31  Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.024 sec
32
33  Results :
34
35  Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
36
37  [INFO] -----
38  [INFO] BUILD SUCCESS
39  [INFO] -----
40  [INFO] Total time: 6.853 s
41  [INFO] Finished at: 2020-11-04T01:54:48Z
42  [INFO] Final Memory: 18M/148M
43  [INFO] -----
✓ > target/surefire-reports/*.xml — Archive JUnit-formatted test results <1s
  
```

- f) When done, click the Proceed button to kill the PHP server process as shown:



The screenshot shows the Jenkins Blue Ocean UI for a pipeline named 'Integration UI Test'. The pipeline diagram at the top shows a sequence: Start → Test (with a pause icon) → Deploy → End. Below the diagram, the 'Integration UI Test / Deploy - 1m 54s' stage is expanded, showing three steps: 'Check out from version control' (completed), './jenkins/scripts/deploy.sh' (completed), and 'Finished using the web site? (Click "Proceed" to continue)' (waiting for input). At the bottom, a large gray box contains the text 'Finished using the web site? (Click "Proceed" to continue)' and two buttons: 'Proceed' and 'Abort'.

- g) Exit from the Blue Ocean UI and return to Jenkins classic UI where you will see the Test Result Trend as shown on Page 1 of this hand-out.

Hope that now you have a good understanding of how to implement automated integration and UI testing and is able to start incorporating it into your team project. Enjoy!

3: References

Selenium API <https://www.selenium.dev/selenium/docs/api/java/>

HtmlUnit API <https://www.selenium.dev/htmlunit-driver/index.html?org/opengr/selenium/htmlunit/HtmlUnitDriver.html>

END OF DOCUMENT