The automated testing of a website can be built by creating behavior-driven development Cucumber feature files and associated unit tests (JUnit) that reference Java page objects representing the web application using the Selenium WebDriver and a headless environment such as PhantomJS Driver, and then porting the solution into a continuous integration environment (Jenkins).

Below are the Maven dependencies to execute an automated testing solution using Cucumber, Selenium WebDriver, PhatomJS Driver, JUnit, Spring, and Log4j.

|  |  |
| --- | --- |
| Description | Maven Dependency |
| Cucumber | <dependency>  <groupId>info.cukes</groupId>  <artifactId>cucumber-java</artifactId>  <version>1.2.4</version>  </dependency>  <dependency>  <groupId>info.cukes</groupId>  <artifactId>cucumber-picocontainer</artifactId>  <version>1.2.4</version>  </dependency>  <dependency>  <groupId>info.cukes</groupId>  <artifactId>cucumber-junit</artifactId>  <version>1.2.4</version>  </dependency> |
| Selenium WebDriver | <dependency> <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>2.53.0</version>  </dependency>  <dependency> <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-server</artifactId>  <version>2.44.0</version>  </dependency>  <dependency> <groupId>org.seleniumhq.selenium</groupId>  <artifactId>htmlunit-driver</artifactId>  <version>2.21</version>  </dependency> |
| PhatomJSDriver (Ghost Driver) | <dependency>  <groupId>com.github.detro</groupId> <artifactId>phantomjsdriver</artifactId>  <version>1.2.0</version>  </dependency> |
| JUnit | <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  </dependency> |
| Spring | <dependency> <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.3.3.RELEASE</version>  </dependency> |
| Log4j | <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-api</artifactId>  <version>2.7</version>  </dependency>  <dependency>  <groupId>org.apache.logging.log4j</groupId>  <artifactId>log4j-core</artifactId>  <version>2.7</version>  </dependency> |

The steps to install PhantomJS below are to be executed on the server where Jenkins is hosted.

|  |  |
| --- | --- |
| Install PhantomJS | As Referenced from <https://gist.github.com/julionc/7476620>.  Version: 2.1.1  Platform: x86\_64  First, install or update to the latest system software.  sudo apt-get update  sudo apt-get install build-essential chrpath libssl-dev libxft-dev  Install these packages needed by PhantomJS to work correctly.  sudo apt-get install libfreetype6 libfreetype6-dev  sudo apt-get install libfontconfig1 libfontconfig1-dev  Get it from the [PhantomJS website](http://phantomjs.org/).  cd ~  export PHANTOM\_JS="phantomjs-2.1.1-linux-x86\_64"  wget https://bitbucket.org/ariya/phantomjs/downloads/$PHANTOM\_JS.tar.bz2 sudo tar xvjf $PHANTOM\_JS.tar.bz2  Once downloaded, move Phantomjs folder to /usr/local/share/ and create a symlink:  sudo mv $PHANTOM\_JS /usr/local/share  sudo ln -sf /usr/local/share/$PHANTOM\_JS/bin/phantomjs /usr/local/bin  Now, It should have PhantomJS properly on your system.  phantomjs –version |

Cucumber uses feature files to specify use cases that describe a specific function of the software being tested. The feature files use the Gherkin language to define the test cases. The Gherkin syntax is designed to be non-technical and human readable in effort to promote business driven development practices across an entire development team. A feature file is separated into three parts: Features, Scenarios, and Steps. The Features describe specific function of the software being tested. Each feature is made of a collection of scenarios and each Scenario is defined by a sequence of Steps. Using java annotations, the Cucumber Steps associate with JUnit tests.

Below is the defined feature file for this application: automationPracticeForm.feature.

Feature: To test the ToolsQA Automation Practice Form page.

Background:

Given On the ToolsQA Automation Practice Form page.

Then Verify on the Automation Practice Form page.

|Field |Value |

|Page Url |http://toolsqa.com/automation-practice-form/ |

@toolsQAAutomationPracticeForm

Scenario: Test the ToolsQA Automation Practice Form page.

When Select the Partial Link Test.

Then Verify the Partial Link Test url.

|Field | Value |

|Partial Link Test | Partial Link Test |

|Partial Link Url | http://toolsqa.wpengine.com/automation-practice-form/|

|Navigation Back Url | http://toolsqa.com/automation-practice-form/ |

And Select the Link Test.

Then Verify Link Test url.

|Field | Value |

|Link Test | Link Test |

|Link Url | http://toolsqa.wpengine.com/automation-practice-table/|

|Navigation Back Url| http://toolsqa.com/automation-practice-form/ |

And Enter a First Name and Last Name.

Then Verify First Name and Last Name.

|Field | Value |

|First Name | Gregory |

|Last Name | Dombchik |

And Select the "Male" Sex radio button.

Then Verify Sex radio button.

And Select the "3" Years of Experience radio button.

Then Verify Years of Experience radio button.

And Enter the date "10/24/2016" in the Date field.

Then Verify Date field.

And Select the Profession checkboxes.

|Field | Value |

|Manual Tester | Manual Tester |

|Automation Tester | Automation Tester |

Then Verify the selected Profession checkboxes.

And Select the "Selenium Automation Hybrid Framework" link.

Then Verify the Selenium Automation Hybrid Framework link url.

|Field | Value |

|Url | http://toolsqa.wpengine.com/wp-content/uploads/2014/04/OnlineStore.zip |

And Select the "Australia" option from the Continents dropdown.

Then Verify the Continents dropdown.

And Select the Selenium Commands multiselect dropdown.

|Field | Value |

|Navigation Commands | Navigation Commands |

|Switch Commands | Switch Commands |

|Wait Commands | Wait Commands |

Then Verify the Selenium Commands multiselect dropdown.

And Select the Button.

Then Verify the query string.

|Field | Value |

|query string | <http://toolsqa.com/automation-practice->form/?firstname=Gregory&lastname=Dombchik&sex=Male&exp=3&profession=Manual+Tester&profession=Automation+Tester&continents=Australia&selenium\_commands=Navigation+Commands&selenium\_commands=Switch+Commands&selenium\_commands=Wait+Commands&submit=|

And Verify text labels.

|Field | Value |

|Text1 | Text1 |

|Text2 | Text2 |

Selenium WebDriver (Selenium 2.0) makes calls to a web browser’s native support for automation using a browser-specific browser driver. Selenium documentation lists the available [Selenium Drivers](http://www.seleniumhq.org/docs/03_webdriver.jsp#selenium-webdriver-s-drivers). Firefox Driver, ChromeDriver, and HtmlUnit Driver are an example of a few Selenium Drivers. The example project uses the PhantomJS Driver. The PhantomJS Driver utilizes Ghost Driver, a Remote WebDriver that uses PhantomJS as back-end (github.com/detro/ghostdriver). The HtmlUnit Driver has been installed, but is not currently be used. HtmlUnit is a java-based implementation of a WebBrowser without a GUI. I found the HtmlUnit Driver did not consistently represent the target WebBrowser, especially in regards to emulating JavaScript events, and I switched to the PhantomJS Driver for a similar headless environment.

Below are highlights of some functionality as demonstrated in the [ToolsQA](https://github.com/gdombchik/ToolsQA/) GitHub repository. The Cucumber feature files are located in the src/test/resources source folder. The application.properties file is located in the src/main/resources source folder. The JUnit files can be located in the com.cucumber package.

|  |  |  |
| --- | --- | --- |
| Functionality | Description | Java Files |
| CucumberOptions | CucumberOptions specifies JUnit configuration items such as location of Cucumber feature files, Cucumber reporting, and Cucumber test suites. | RunnerTest.java |
| Selenium WebDriver | Specify the Selenium WebDriver implementation. The example uses PhantomJSDriver and returns the WebDriver interface. | AbstractPageStepDefinition.java  WebDriverList.java |
| Properties File | The main functionality is to specify the PhantomJS binary package located on my machine versus the location on the AWS Ubuntu server. | AppConfig.java  FileService.java  FileServiceImpl.java |
| Page Object Design Pattern | As specified in the Selenium documentation:  “Page Object is a Design Pattern which has become popular in test automation for enhancing test maintenance and reducing code duplication.  The Page Object Design Pattern provides the following advantages  1. There is a clean separation between test code and page specific code such as locators (or their use if you’re using a UI Map) and layout.  2. There is a single repository for the services or operations offered by the page rather than having these services scattered throughout the tests.” | *Cucumber Feature File:*  automationPracticeForm.feature  *Test File:*  AutomationPracticeForm.java  *Page Objects:*  AbstractPage.java  LandingPage.java  AutomationPracticeFormPage.java  *Web Driver Utility:*  WebDriverUtils.java |

In reviewing how to run the Java tests using a continuous integration (ci) tool such as Jenkins, an option is to use Maven to run the Selenium tests using a headless environment using tools such as PhantomJS.

The steps to install Jenkins, Java, Maven, and Git on AWS E2 Ubuntu server: <https://gist.github.com/jsuwo/9038610>.

The Jenkins project setup for a Java project in GitHub. This example Jenkins has been installed on an AWS E2 Ubuntu server.

|  |  |
| --- | --- |
| Description | Command |
| *AWS E2 Ubuntu server* | |
| Switch to the Jenkins user. | $ sudo su – jenkins |
| Create public private key pair. | $ cd .ssh  $ ssh-keygen -t dsa |
| Cat the public key and copy the results. | $ cat ToolsQA.pub |
| *GitHub* | |
| Add and paste deploy key in GitHub project. | <https://github.com/gdombchik/ToolsQA/settings/keys> |
| *Manage Jenkins – Configure System* | |
| Jenkins Location | Scroll to the “Jenkins Location” section.  Jenkins URL:  <http://ec2-52-25-64-69.us-west-2.compute.amazonaws.com/>  System Admin e-mail address:  [greg@gregorydombchik.com](mailto:greg@gregorydombchik.com) |
| E-mail Notification | Scroll to the “E-mail Notification” section.  SMTP server:  mail.gregorydombchik.com |
| *Jenkins Server – Global Tool Configuration* | |
| Maven Installation | Scroll to the “Maven” section.  Select “Maven installation…” button.  Name:  Apache Maven 3.0.5  MAVEN\_HOME:  /usr/share/maven/ |
| JDK Installation | Scroll to the “JDK” section.  Select “JDK installation…” button.  Name:  Oracle JDK 1.7  JAVA\_HOME:  /usr/lib/jvm/java-7-oracle/ |
| *Jenkins Server – New Project* | |
| Add A Jenkins Project for a Java project. | Select New Item.  Enter an item name.  Select Freestyle project.  Select Ok. |
| Specify the URL of the remote GitHub repository. | Scroll to the “Source Code Management” section.  Select the “Git” option.  Repository URL:  <https://github.com/gdombchik/ToolsQA.git> |
| Update the Poll SCM. | Scroll to the “Build Triggers” section.  Select the “Poll SCM” option.  Enter the following in the “Schedule” text box:  H \*/3 \* \* \* |
| Update the Build. | Scroll to the “Build” section.  Select from the “Add build step” and select the “Invoke top-level Maven targets” option.  Maven Version:  (Default)  Goals:  clean install  POM:  pom.xml |
| Update E-mail Notification. | Scroll to the Post-build Actions.  Select from the “Add post-build action” and select the “E-mail Notification” option.  Enter email address in the “Recipients” text box:  [greg@gregorydombchik.com](mailto:greg@gregorydombchik.com) |
| *GitHub* | |
| Add Jenkins GitHub Plugin you can automatically trigger build jobs when  pushes are made to GitHub. | <https://github.com/gdombchik/ToolsQA/settings/installations>  Select from the “Add service” and select the “Jenkin’s (Git plugin)” option. (NOT GITHUB PLUGIN. This service does not appear to work).  Enter the following in the “Jenkins url” text box:  <http://ec2-52-25-64-69.us-west-2.compute.amazonaws.com/> |

