Technical Document for CucumberSample Project

Created by: GangaSravanthi Gollamudi

Date: 28 Jul 2018

Purpose

Automation framework created and shared for web application is light weight and robust. This technical document explains, classes and methods created in framework to verify below scenarios of http://automationpractice.com. This framework not limited to below scenarios and should be extended to automate other test flows with re-usable components and common repositories. By incorporating Behavior Driven Development and Cucumber, understanding would be easier to non-technical or business users.

Scenario 1: Order T-Shirt (and Verify in Order History)
Scenario 2: Update Personal Information (First Name) in My Account

System configuration

This framework created, executed, debugged and committed to GitHub with configuration mentioned below. Recommended configuration should be equivalent or high. This framework will work flawless even with less configuration, however performance may not be at optimum level.

- Processor: Intel I-3 64-bit (x64) processor or higher
- RAM: 4GB or higher
- Operating Systems: Windows 8.1 or later
- Privileges: Administrative rights required during installation only
- Hard Disk Space: 2GB or more

Software Requirements

This framework created, executed, debugged and committed to GitHub with configuration mentioned below. Recommended configuration should be equivalent to framework to work as intended.

- JDK version 1.8.0_181
- Eclipse Kepler
- ChromeDriver 2.41
- Cucumber (feature file plug-in)
  http://cucumber.github.com/cucumber-eclipse/update-site
- Dependencies (details available in **CucumberAutomation/ReadMe)**

Maven

Selenium WebDriver is great for browser automation. But, when using it for testing and building a test framework, it feels underpowered. Integrating Maven with Selenium provides following benefits

- Apache Maven provides support for managing the full lifecycle of a test project.
- Maven is used to define project structure, dependencies, build, and test management.
- Using pom.xml (Maven) you can configure dependencies needed for building testing and running code.
- Maven automatically downloads the necessary files from the repository while building the project

Project Structure:

src/main/java: Java source code for common methods across the application goes here

- ReusableComponents : This is a package which accommodates the class files reused in the scripted testcases
    - CheckBox.java: This class has methods to verify a checkbox and to select a checkbox
        - oCheckBox.isDisplayed(): Verifies if the checkbox is displayed in webpage and returns a Boolean value
        - oCheckBox.isEnabled(): Verifies if the checkbox is enabled in webpage and returns a Boolean value
        - oCheckBox.isSelected(): Verifies if the checkbox is selected in webpage and returns a Boolean value
        - oCheckBox.click(): Selects the chosen checkbox

    - DropDown.java: This is a class file and has methods to choose an option from dropdown by passing the chosen option to "switch case"
        - selectByVisibleText() : command can be used to select a list option from the Drop Down field using its Label Text.
        - selectByValue() : command is used to select the list option using the specified value of the List Option from the multi-selection Box field.
        - selectByIndex() : command is used to select the list option using the specified Index of the List Option from the multi-selection Box field.

- LaunchApplication.java: This is a class file and has the methods to login and take snapshots
  - LaunchApplication() : This is a constuctor to launch the browser and maximize the window
  - login() : This is a public method to open the website and "Sign In"
  - successLogin() : Verifies if the login is successful
  - takeSnapShot() : Takes the snapshots of the webpage

- PropertiesFile.java: This is a File and it holds the methods to read a property file from the destination and get the data from the file
  - PropertiesFile() : This is constructor and has method to load the properties file from the variable of file path in class file
  - getData() : gets the value from the properties file for the given key
  - getLocator() : gets the location of web-element for the specified locator

src/main/resources: contains the properties files and snapshot documents of the class files in src/main/java

- login.properties : contains the <key,value> pairs of variables and web-elements

src/test/java: This has the scripts for test scenarios and test runner

- cucumbersample : This is a package containing class files of test scenarios

  - PlaceOrder.java: This is a class file and has methods to place an order for T-SHIRT in http://automationpractice.com website.
    - Steps covered:
      - → Login into the application
      - → Place an order for T-SHIRT
      - → Proceed to checkout
      - → Verify for the order placed in order history

    - UpdateName.java: This Class file logs into personal information and updates the first name of the user
        - Steps Covered:
            → Login into the application
            → Open personal information page
            → Change the first name
            → Save the data

- runner: This package contains class file to run the cucumber scripts
    - TestRuner.java: This method runs the scripts with "@RunWith(Cucumber.class)" and reads the cucumber option with

```
@CucumberOptions

(features="features",    // path to feature file

glue={"cucumbersample"},  // package name of test scripts

plugin = {"html:target/cucumber-html-report" })  //plugin to generate html reports
```

src/test/resources: contains folders of properties file and snapshots

- Object Repository: This folder contains properties files of "PlaceOrder.java" and "UpdateName.java" contains the <Key,value> pairs of the web element locators
- Snapshots: This folder collects and saves the snapshots taken at run time

Maven Dependencies: Contains the ".jar" filesneede to run the application

Features:

A Feature File is an entry point to the Cucumber tests. This is a file where you will describe your tests in Descriptive language (Like English). It is an essential part of Cucumber, as it serves as an automation test script as well as live documents. A feature file can contain a scenario or can contain many scenarios in a single feature file but it usually contains a list of scenarios. Let's create one such file.

In order for Cucumber to automatically detect the stories (or features, as they're known in Cucumber), you need to make sure that they carry the '.feature' file extension

- **Feature**: Defines what feature you will be testing in the tests below
- **Given**: Tells the pre-condition of the test
- **And**: Defines additional conditions of the test
- **Then**: States the post condition. You can say that it is expected result of the test.

Gherkin:

A language above is called Gherkin and it implements the principles of Business readable domain specific language (BRDSL). Domain specific language gives you the ability to describe your application behavior without getting into details of implementation.

The features folder in the "cucumbersample" project contains the feature files for PlaceOrder.java and UpdateName.java classes. The following plugin is added to make the feature files more user friendly

"http://cucumber.github.com/cucumber-eclipse/update-site"

target:

This build directory contains the html reports generated during the run time

pom.xml:

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Examples for this is the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/test/java; and so on.

Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.

```
<dependencies>

        <dependency>

                <groupId>org.seleniumhq.selenium</groupId>

                <artifactId>selenium-java</artifactId>

                <version>2.53.0</version>

        </dependency>
        <dependency>
              <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>4.11</version>
            </dependency>
            <dependency>
                <groupId>info.cukes</groupId>
                <artifactId>cucumber-junit</artifactId>
                <version>1.2.2</version>
              <scope>test</scope>
            </dependency>
        <dependency>
                <groupId>info.cukes</groupId>
                <artifactId>cucumber-java</artifactId>
              <version>1.2.2</version>
        </dependency>
</dependencies>
```