Ekta Dosad

Automation of end-to-end user e-commerce transaction flow
GitHub URL: https://github.com/ektadosad/Walmart-Transaction_flow.git

Reasoning behind technical choice- Page Factory Design Pattern:
Disadvantage of Selenium test cases is that it leads to an unmaintainable project because of the delicacy (duplicated usage of locators). We have to walk through the whole test code to adjust locators if it is changed, which is time a time consuming process. A better approach to script maintenance can be achieved by separating the abstraction of test objects and test scripts.

I chose Page Factory design pattern for this project because it provides us an ability to reuse the Page Factory Web elements across different classes/methods while reducing multiple findElement calls. Moreover, this approach helped me in reducing code-duplicity from my functional test cases and enabled me to write modular test cases in much optimized manner to conduct testing of various web elements.
In a nut-shell, because of following advantages over other approaches, I chose Page Factory Design Pattern:

- Readability of scripts.
- Reduce or Eliminate duplication.
- Easy to Maintain.
- Re-usability of code to implement more tests.
- Can be used in any kind of framework such as Data Driven, Keyword or Modular Driven.

Technology:

- **Testing tool:** Selenium Web Driver
- **Programming language:** Java

Framework:

- TestNG
- ReportNG

Design Pattern:

- Page Factory Design Pattern

Coverage:

- Functional Tests
- Appearance/User Experience Testing

Others:

- Eclipse IDE, TestNG Eclipse plug-in, ReportNG
- JDK 1.7 or higher

This approach covers all the 5 scenarios given in the requirement document. I might have used data driven or keyword driven approach to run the test case multiple times for different users by reading log in details and item keywords from excel.

Instructions for running the code:

- Open Eclipse IDE.
- From the main menu bar, select File -> The Import wizard opens.

Ekta Dosad

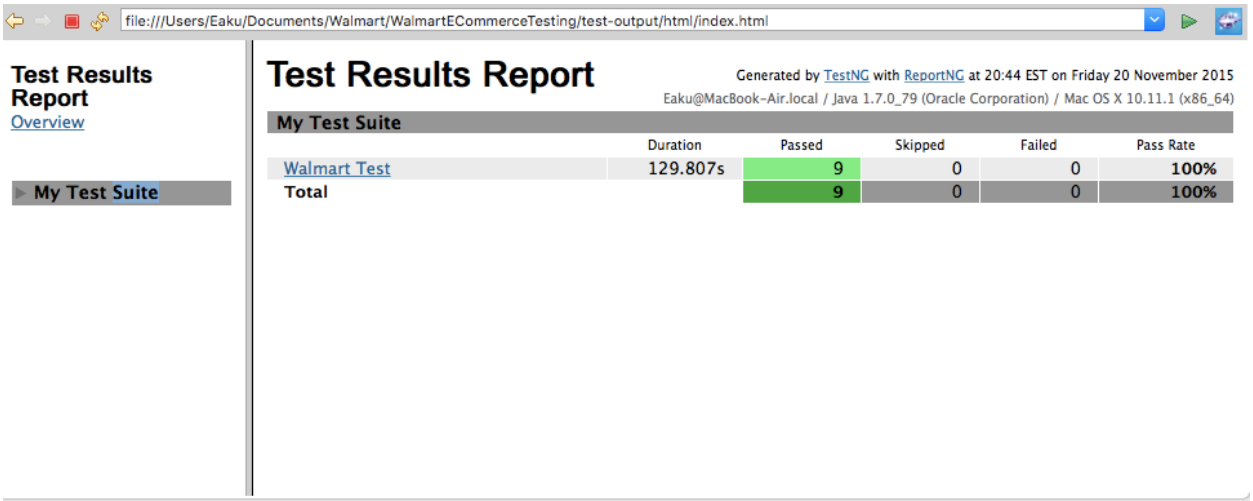
- Select General > Existing Project into Workspace and click Next.
- Choose either Select root directory or Select archive file and click the associated Browse to locate the directory or file containing the project. Select project **WalmartECommerceTesting.**
- Click Finish to start the import.
- Select project properties.
- Add all the Jar Files from JAR folder to the project's build path.
- Select TestNG from properties and Disable default TestNG listeners.
- Open src folder and run testing.xml as TestNG Suit.
- Open index.html (under test output) in browser to see test report.

Screenshot of HTML report through ReportNG:

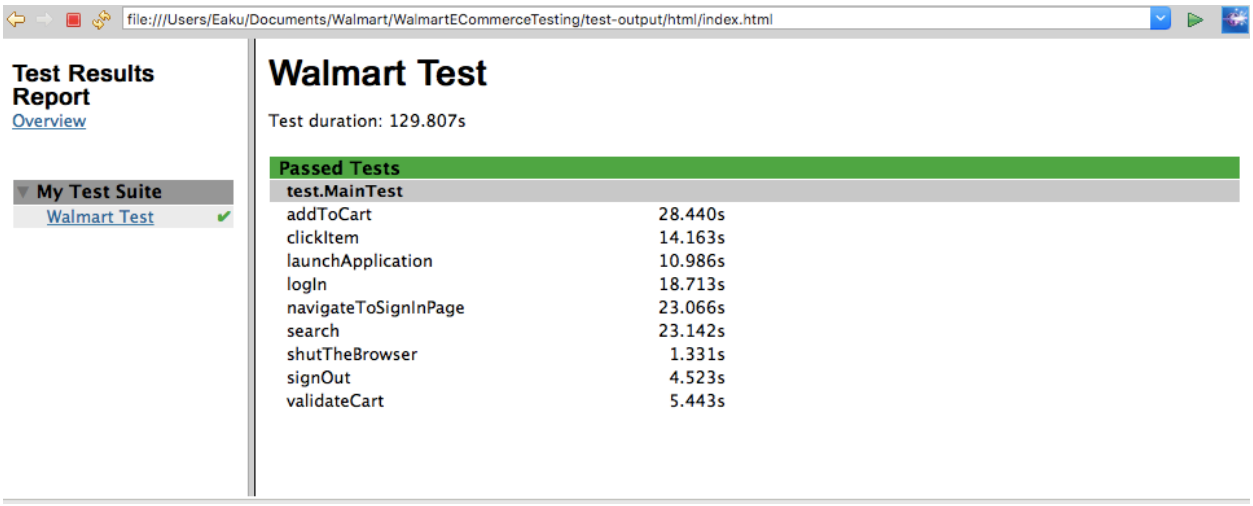
JAR files:

1. **Selenium-2.46.0 (Add all the JAR files inside and outside the "libs" folder)**
2. **ReportNG-1.1.4.jar**
3. **velocity-dep-1.4.jar**
4. **guice-3.0.jar**
5. **Download TestNG from - https://marketplace.eclipse.org/content/testng-eclipse**

Overview



Ekta Dosad



Note: Regarding requirements that are not listed in the homework assignment:

- Negative scenarios: I noticed that few items have option of selecting size and color. If the item is not available in particular or size, then either we can search for different item or can fail the test as "the item is out of stock". So, even after implement size and color selection option, there is a possibility that the test will fail (if item is out of stock).

  Right now, I am choosing items at random. As mentioned by the Hiring Manager. These scenarios are being ignored for now and only items that have quantity are added (as quantity web element is present for all the items). I have implemented selection of quantity (no. of items you want to add in your cart). The requirement of color and size selection can be implemented in the same way if needed.
- As mentioned by the hiring manager, I am randomly selecting an item from the result set (by selecting location of random items). This kind of selection can not be linked to a particular item requirement (As we don't know what item will be present at that location). Due to frequent changes in the site, locator of elements can change and selenium web driver can not locate the element if its locator has changed.

I could have written better tests if above requirements were mentioned in the homework assignment.

- **Issue with the site:** The site takes a lot of time to load, which leads to timeout and causes the test to fail with "Unable to find element with id…" error.

Solution: I have implemented explicit wait and have given 35 sec wait. But if the site takes more than 35 sec and the test will fail with "Unable to locate element" …timeout error.