♥ Selenium and Web Driver Interview Questions and Answers

Posted on May 16, 2015 by Arulkumaran Kumaraswamipillai

Q1. What is Selenium?
A1. Selenium is a suite of tools for browser automation (i.e. automated Web testing). It is composed of

- **Selenium IDE**: A tool for recording and playing back. This is a Fire-fox plugin.
- **WebDriver** and **RC** provide the APIs for a variety of languages like Java, .NET, PHP, etc. A Java example is shown below. The WebDriver and RC work with most browsers.
- **Grid** transparently distribute your tests on multiple machines so that you can run your tests in parallel, cutting down the time required for running in-browser test suites.

Q2. How would you go about using selenium for your web testing?
A2. The example below shows the steps involved in testing a

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

open all | close all
- Ice Breaker Interview
- Core Java Interview C
- JEE Interview Q&A (3
- Pressed for time? Jav
- SQL, XML, UML, JSC
- Hadoop & BigData Int
- Java Architecture Inte
- Scala Interview Q&As
- Spring, Hibernate, & M
- Testing & Profiling/Sa
  - Automation Testing
    - ♥ Selenium and
  - Code Coverage (2)
  - Code Quality (2)
  - jvisualvm profiling (
  - Performance Testir
  - Unit Testing Q&A (2
- Other Interview Q&A f
- ▶ Free Java Interviev

simple login page with selenium + Web Driver. The pages have HTML snippets as shown below

Login page:

```
...

<form method="POST" action="some/url">

    <input type="text" name="username"/>

    <input type="text" name="password" />

</form>


...


```

Login response page

```
...
<table>
     <tr>
    <td>John</td><
  /tr>
</table>
...

```

STEP 1: Configure your Maven to include the required jar file selenium-java-x.x.x.jar.

```
<properties>
 <junit.version>4.4</junit.version>
 <selenium.version>2.7.0</selenium.version>
 <slf4j.version>1.5.2</slf4j.version>
</properties>

<dependencies>
 <dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>${selenium.version}</version>
 </dependency>
 <dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>${junit.version}</version>
 </dependency>
 <dependency>
```

```
19    <groupId>org.slf4j</groupId>
20    <artifactId>slf4j-log4j12</artifactId>
21    <version>${slf4j.version}</version>
22   </dependency>
23 </dependencies>
24
```

The selenium-java-x.x.x.jar will transitively bring in the other relevant driver jars for different browsers.

STEP 2: It is a best practice to have separate page objects as these page objects can be shared by multiple JUnit test cases. If a particular page element changes, you will have to change it only in one place, which is page object, and not in all JUnit test cases where a paricular element is used. The JUnit test cases will depend on the page objects and should not refer to the page elements directly.

```
1  package myapp.systemtest;
2
3  import org.openqa.selenium.WebDriver;
4
5  public class SystemTestPage {
6
7      protected WebDriver driver;
8
9      protected SystemTestPage(WebDriver driver) {
10         this.driver = driver;
11     }
12 }
13
```

You can have your page objects extend the above generic SystemTestPage class.

```

```

```
1  package com.systemtest.page;
2
3  import org.openqa.selenium.By;
4  import org.openqa.selenium.WebDriver;
5  import org.openqa.selenium.WebElement;
6  import org.openqa.selenium.support.CacheLookup;
7  import org.openqa.selenium.support.FindBy;
8
9  import myapp.systemtest.SystemTestPage;
10
11
12 public class LoginPage extends SystemTestPage {
13
14     @FindBy(name = "username")
15     @CacheLookup
16     private WebElement userNameInput;
17
18  @FindBy(name = "password")
19     @CacheLookup
20     private WebElement passwordInput;
21
22     @FindBy(xpath ="//input[@type=\"submit\"]")
23     @CacheLookup
24     private WebElement submitButton;
25
26     public LoginPage(WebDriver driver){
27         super(driver);
28     }
29
30     public void login(String userName, String pa
31         userNameInput.sendKeys(userName);
32   userNameInput.sendKeys(password);
33         submitButton.submit();
34     }
35
36     public String getUserName(String userName) {
37         WebElement element = driver.findElement(
38         return element.getText();
39     }
40 }
41
42
```

STEP 3: Write the JUnit class using the page objects defined above and assert the results. These JUnit test cases can be run to test your web pages.

```
1  package com.unittests;
2
3  import junit.framework.Assert;
4
5  import org.junit.After;
6  import org.junit.Before;
7  import org.junit.Test;
8  import org.openqa.selenium.WebDriver;
9  import org.openqa.selenium.firefox.FirefoxDriver
10 import org.openqa.selenium.support.PageFactory;
11 import org.openqa.selenium.support.ui.WebDriverW
12
13 import com.systemtest.page.LoginPage;
```

```
14
15 public class LoginTest {
16
17     private static final String BASE_URL = "http
18     private static final String USER_NAME = "Joh
19  private static final String PASSWORD = "aa44"
20
21     private WebDriver driver;
22     private WebDriverWait wait;
23     private LoginPage loginPage; // the page obj
24
25     @Before
26     public void setUp() {
27         driver = new FirefoxDriver();
28         wait = new WebDriverWait(driver, 5);
29         driver.get(BASE_URL);
30         loginPage = PageFactory.initElements(dri
31     }
32
33     @Test
34     public void testLogin() throws Exception {
35         loginPage.login(USER_NAME, PASSWORD);
36         Assert.assertEquals(MAC, loginPage.getUs
37     }
38
39     @After
40     public void tearDown() {
41         driver.quit();
42     }
43 }
44
45
```

Q3. What are selenium locators? What tools do you use to locate them?
A3. Selenium Locators are the way of finding the HTML element on the page to perform a Selenium action on. The example above has a line asshown below to extract the username element from the Login response page. This uses an XPath expression to locate the element.

```
1 public String getUserName(String userName) {
2     WebElement element = driver.findElement(By.xp
3     return element.getText();
4 }
5
```

The XPath expression will be something like //td[[text()=John] which looks for a td element with text value “John”.

The annotation in the above example is also a locator by name as shown below

```
1 @FindBy(name = "username")
2 @CacheLookup
3 private WebElement userNameInput;
4
```

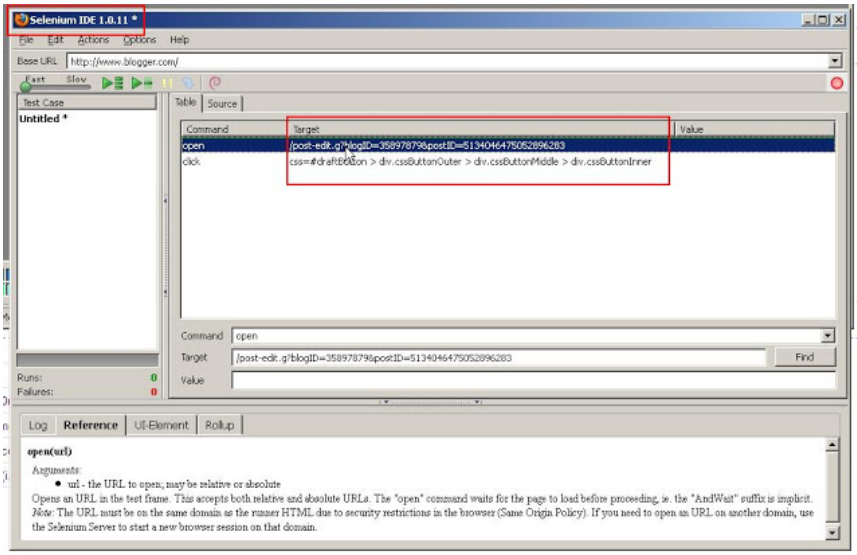
This will match the HTML snippet

```
1 <input type="text" name="username">
2
```

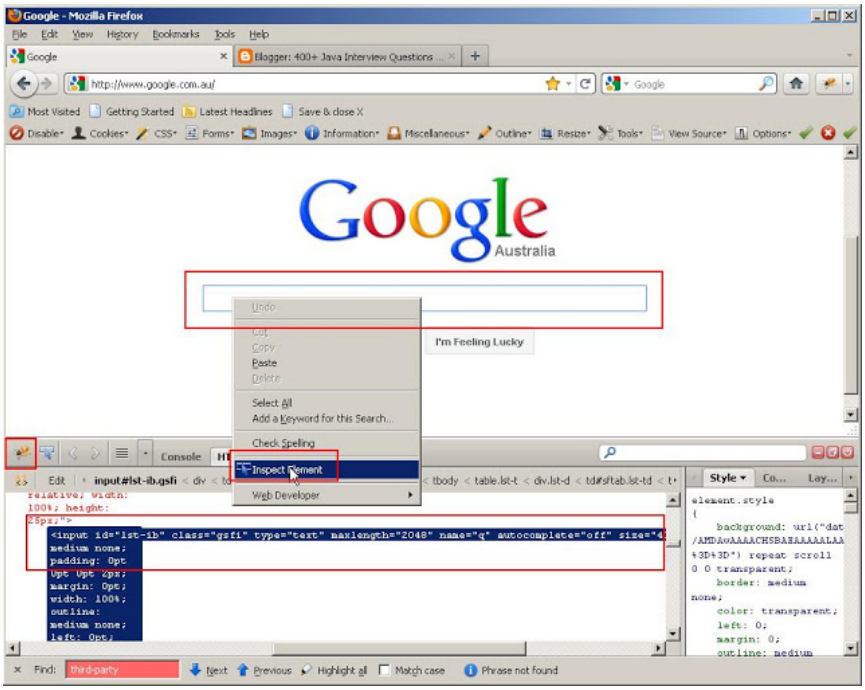
You could also find by tagName, id, css, etc.

There are handy tools to identify the HTML elements or locators.

- Selenium IDE, which is a Firefox plugin useful in identifying the locators, debugging, etc.



- The Fire-bug plugin, which allows you to inspect the elements by right clicking on the page and then selecting “Inspect Element”. Google chrome provides a similar functionality to inspect element out of the box.



Q4. In your experience, what are some of the challenges with Selenium?

A4. In general, badly written test cases whether junit tests or web tests, the biggest complaint is about writing test cases that are not maintainable. Unmaintainable automated test cases can take more time than manual tests. So, it is important to write quality test cases by clearly separting the page objects from the test cases as demonstrated in the Q&A above. The use of the locators need to be carefully thought through. For example, some frameworks like JSF dynamically generate HTML element IDs. So, if IDs are used in your tests, then the test cases may fail if the IDs have changed. The solution to this problem is to use XPath to find the relevant HTML elements. The ClickAndWait action will not work for AJAX calls, and use "waitForElement" instead.

Popular Member Posts

♦ 11 Spring boot interview questions & answers

905 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

816 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

427 views

18 Java scenarios based interview Questions and Answers

408 views

♦ 7 Java debugging interview questions & answers

324 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

311 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

304 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

♦ Object equals Vs == and pass by reference Vs value

234 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ ♥ Beginner array coding problems in Java

jBehave and jUnit for BDD ›

Posted in Automation Testing Q&A

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category ▼

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

↑