SELENIUM PROGRAMMING COOKBOOK

Hot Recipes for Selenium Development

PETR ARSENTEV

Java Code Geeks
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

Selenium Programming Cookbook

Contents

Preface

Selenium is a portable software testing framework for web applications. Selenium provides a record/playback tool for authoring tests without learning a test scripting language (Selenium IDE). It also provides a test domain-specific language (Selenese) to write tests in a number of popular programming languages, including Java, C#, Groovy, Perl, PHP, Python and Ruby.

The tests can then be run against most modern web browsers. Selenium deploys on Windows, Linux, and Macintosh platforms. It is open-source software, released under the Apache 2.0 license, and can be downloaded and used without charge. (Source: https://bit.ly/2bNZYaa)

In this ebook, we provide a compilation of Selenium programming examples that will help you kick-start your own projects. We cover a wide range of topics, from Installation and JUnit integration, to Interview Questions and Standalone Server functionality. With our straightforward tutorials, you will be able to get your own projects up and running in minimum time.

About the Author

Petr Arsentev has over 8 years of experience in java development. He participated in the development a few startup projects, which run successfully. He finished Moscow Power Engineering Institute (National Research University) at 2009.

After he started to work in a local company as java developer and still keeps improving the knowledge about software developments. He focused on JVM languages like Java, Scala and related technologies and frameworks. He has developed the few courses about Java in Russian. He teaches students Java language too. His personal website is https://parsentev.ru/.

Chapter 1

Selenium Installation Example

1.1 Introduction

In this tutorial, we are going to show how you can install the Selenium IDE, Selenium Server and Selenium WebDriver.

Selenium is the tool for automation testing web apps. Selenium consists from IDE, WebDrivers and Server.

The general scenery of testing in Selenium is to record the user activities by Selenium IDE and after that to run this tests cases automatically. It uses two different approaches to execute the tests. One of them uses the native browser API, another uses the injecting JavaScript codes to browsers. The main benefit about Selenium IDE is the ability to execute tests on varied browsers. You can find the list of supported Selenium browsers below:

- Firefox
- IE
- Safari
- Opera
- Google Chrome

Another great thing about Selenium is that you can export the tests cases to your favorite programming languages: Java, Ruby, Python, C# and so on.

1.2 Installation

The installation process is very simple. Firstly, You should go to official web site `seleniumhq.org`.

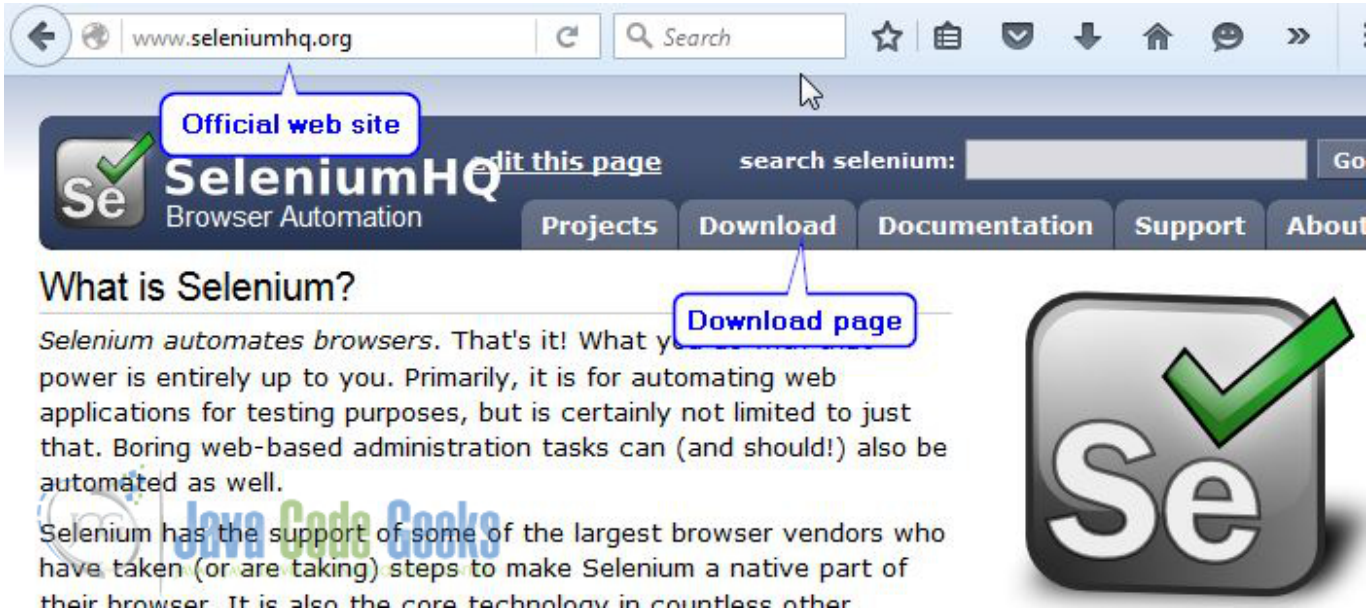


Figure 1.1: Selenium main page

There you can find the download tab, if you click on the tab you will move to download page, as show below. First, you will need Selenium IDE. You need to scroll the page and find the Selenium IDE paragraph. Selenium IDE is the Firefox plugins, so you need to click on this link and follow the next steps. The main important thing is that you can use only Firefox on this process. Selenium IDE addon is not supported by another browsers.

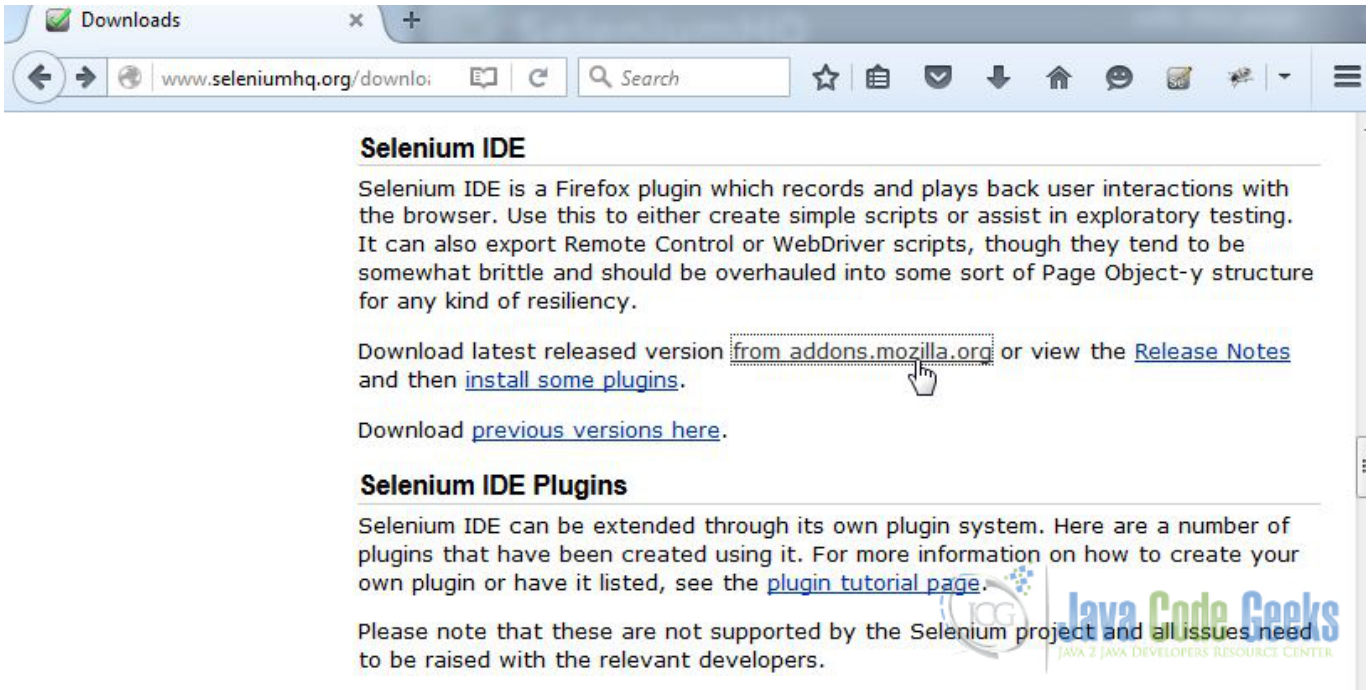


Figure 1.2: Download page

This link redirects you to Firefox addons page. The page offers you to add the Selenium IDE addons. Then you need to click this button.

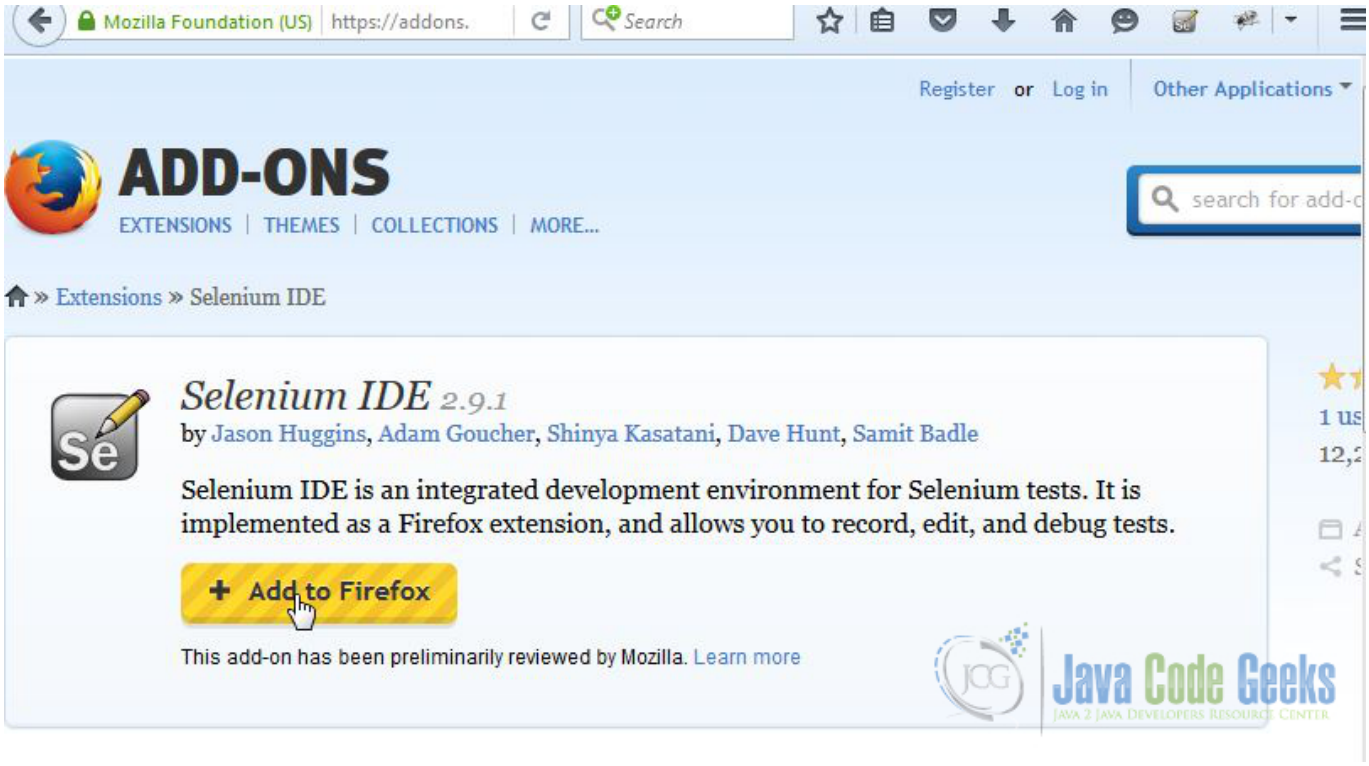


Figure 1.3: Plugins page

After that, Firefox browser offers you to install this addon

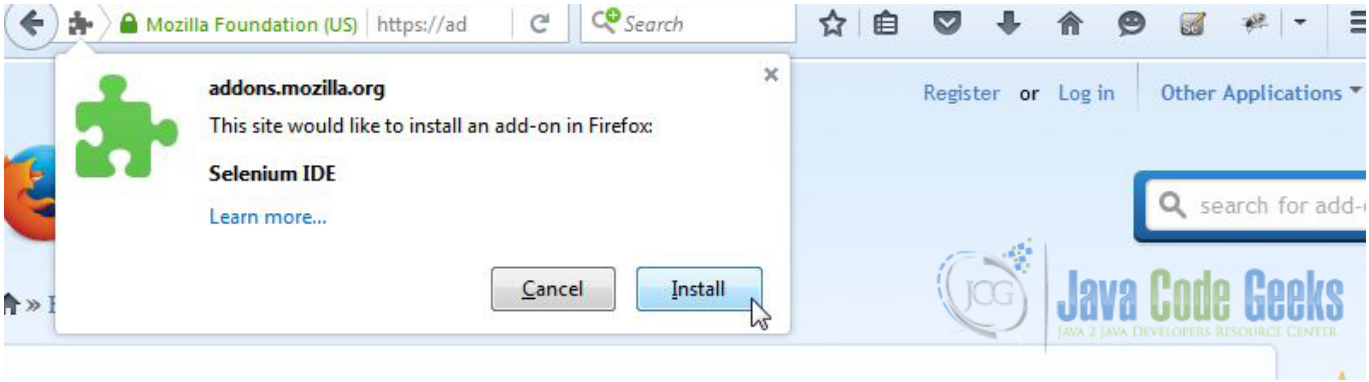


Figure 1.4: Install Addons

and restart the browser.

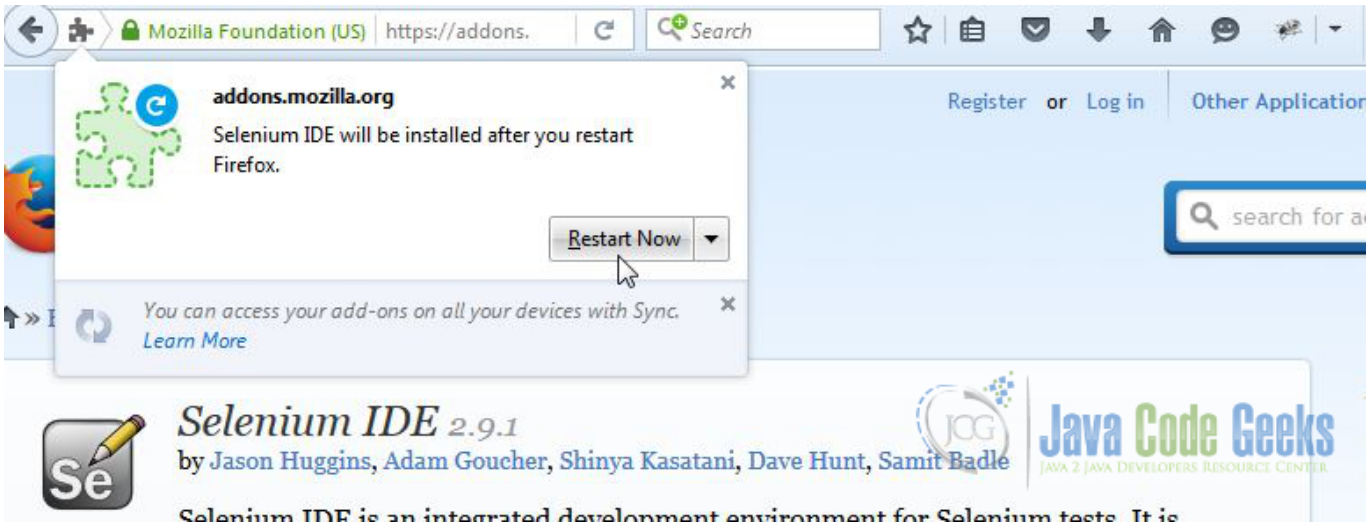


Figure 1.5: Restart

Then you restart to browser, you can find the Selenium IDE button on the right-top corner.

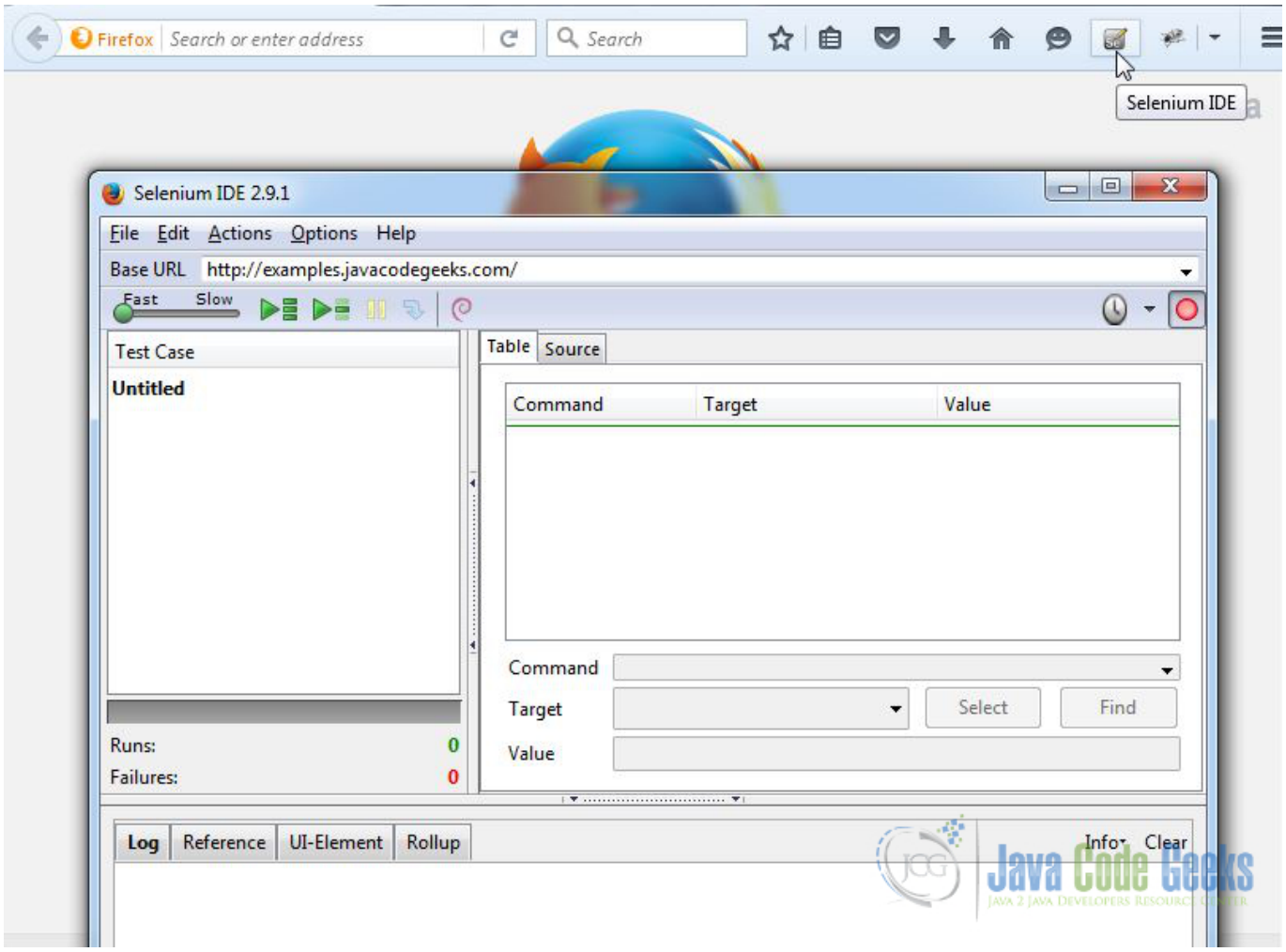


Figure 1.6: Selenium IDE

Now you can record your first test.

Selenium has the special server, which offers to scale your tests. Let's back to Selenium download page and find there the Selenium Server.

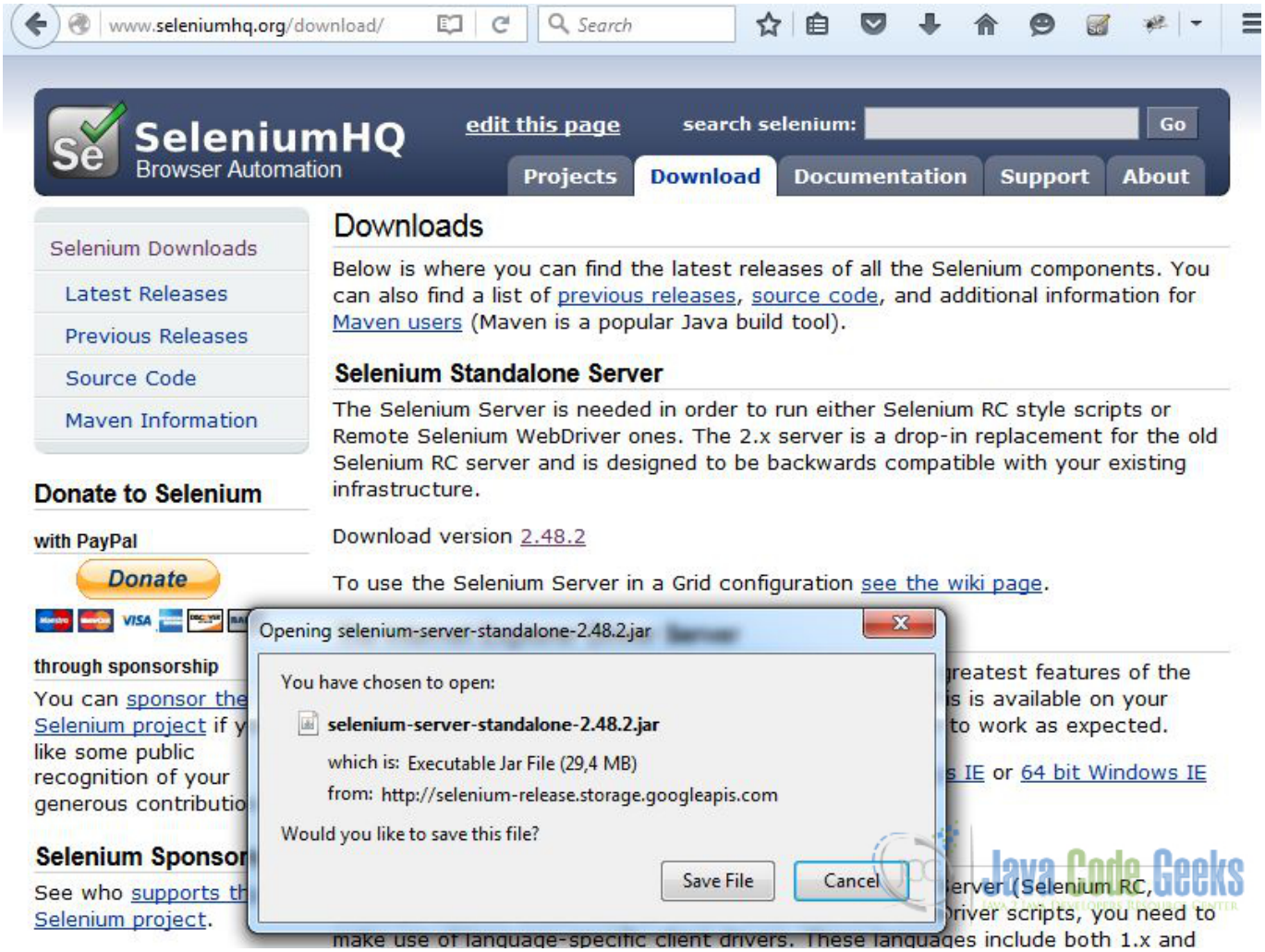
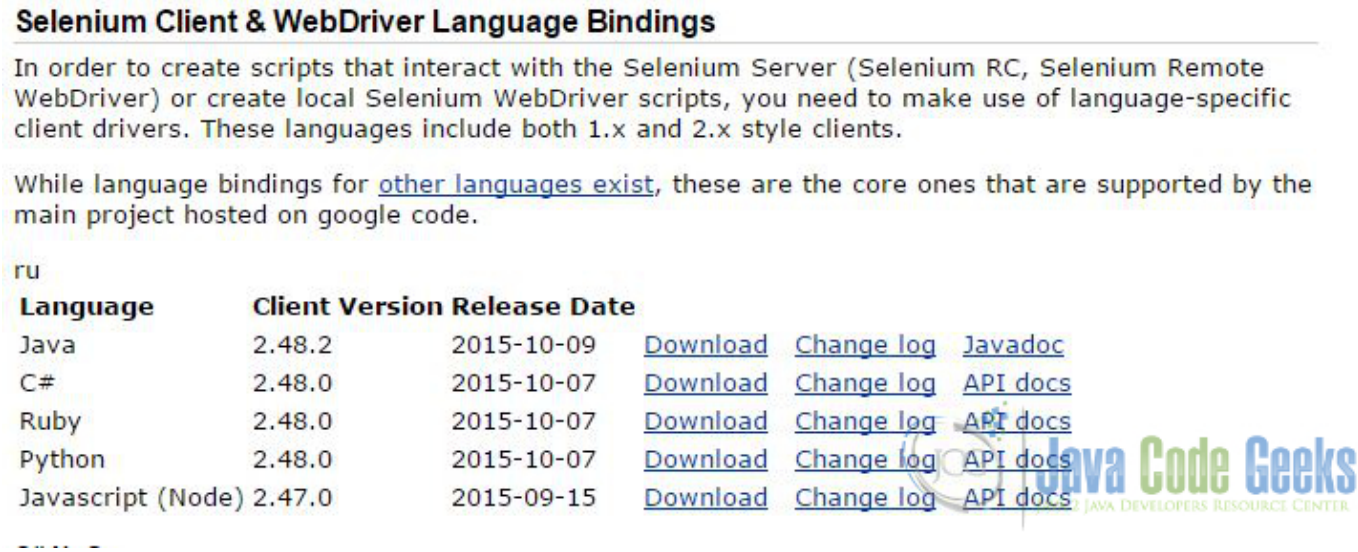


Figure 1.7: Selenium Server

Selenium Server is the JAR file and it does not require the installing. So after you download this JAR, you can run it directly.

How we said before Selenium offers you to use varied browsers for execution the tests. For this reason, we should use the WebDriver. You should back to the download page again.



Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on google code.

ru

Language	Client Version	Release Date			
Java	2.48.2	2015-10-09	Download	Change log	Javadoc
C#	2.48.0	2015-10-07	Download	Change log	API docs
Ruby	2.48.0	2015-10-07	Download	Change log	API docs
Python	2.48.0	2015-10-07	Download	Change log	API docs
Javascript (Node)	2.47.0	2015-09-15	Download	Change log	API docs

Figure 1.8: WebDriver

For example, if you use Java, you need to download the WebDriver for Java programming language and so on. This library has the necessary API for execution tests on varied browsers.

1.3 Conclusion

In this tutorial, we've shown how to install the Selenium tools. As you saw this process is kindly simple. If you want to improve your knowledge about Selenium, you can visit the official web site.

Chapter 2

Selenium IDE Tutorial

2.1 Introduction

In this tutorial, we are going to show how you can test your web app by Selenium IDE. Selenium IDE is the Firefox plugin, which can record the user browser action and run it automatically further. We are going to install the Selenium IDE, explain most useful controlls and panels, record the simple use cases and run it automatically by Selenium IDE, add the schedule for tests, export this test case to programming languages.

Selenium IDE has special commands. Selenium commands emulate user activities. You can emulate any user activities: insert the text, submit the form, navigate in apps, click on link, click on checkbox, select the options in combobox. The main benefic about Selenium IDE is that you don't need to have any experience in programming languages. All you need is install the Selenium plugin, record the use activities and run the tests. It is so user-friendly. Selenium IDE uses the native browser API for testing, so you can use all specters commands, which Firefox browser supports. Let's start to install the Selenium IDE.

2.2 Installation

First of all, we go to the download page and select the last version of Selenium ide there - `https://docs.seleniumhq.org/download/` .Download Selenium IDE Download Selenium IDE

After we downloaded this plugin, Firefox browser offers to install this addons. You need to allow this process. Selenium IDE is installed and we can see the Selenium IDE button on the right-top corner in Firefox.

Figure 2.1: Selenium IDE Button

Then you need to open the Selenium IDE, you can click on this button or select the Selenium IDE in development tools in Firefox, as show below.

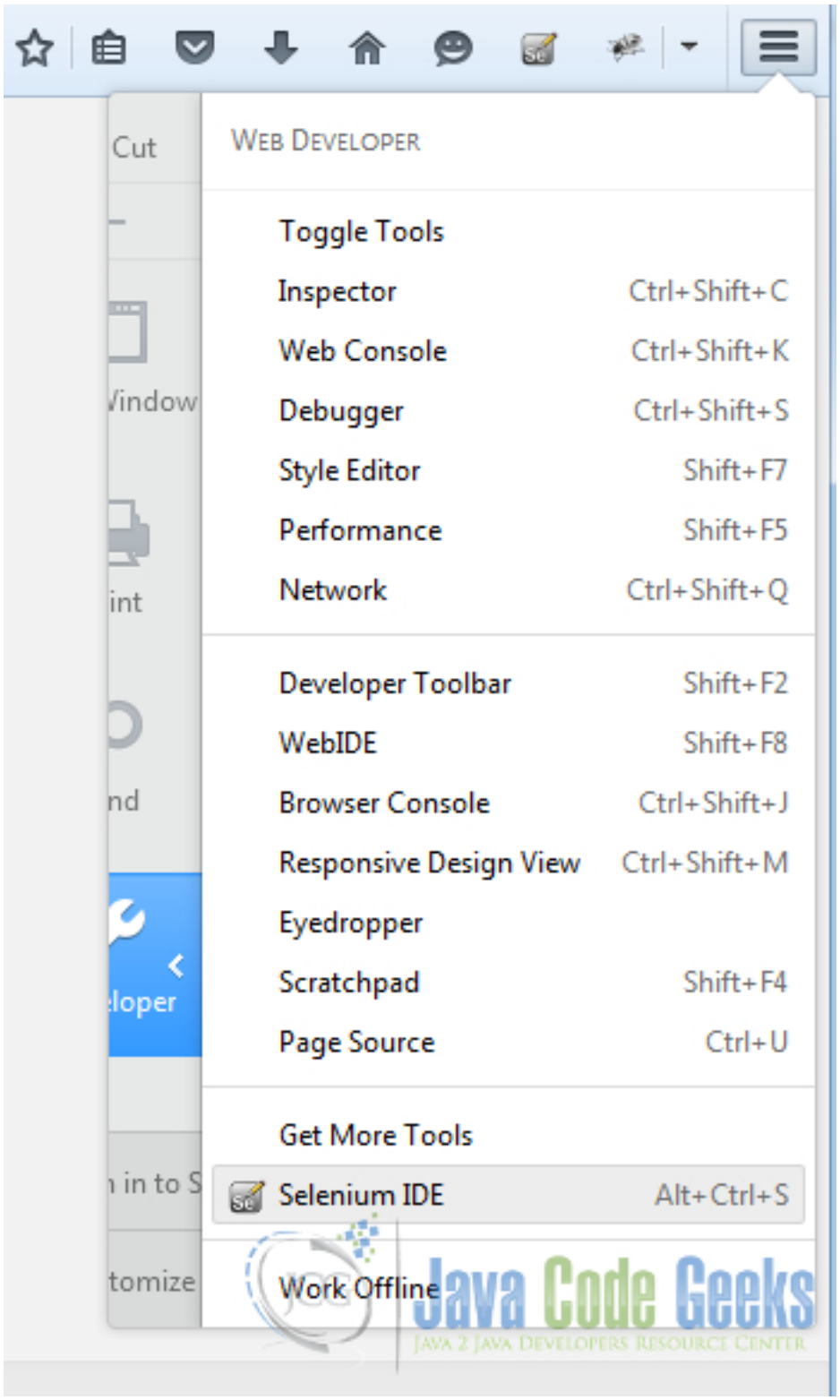


Figure 2.2: Open Selenium IDE plugins

Selenium IDE looks as show below.

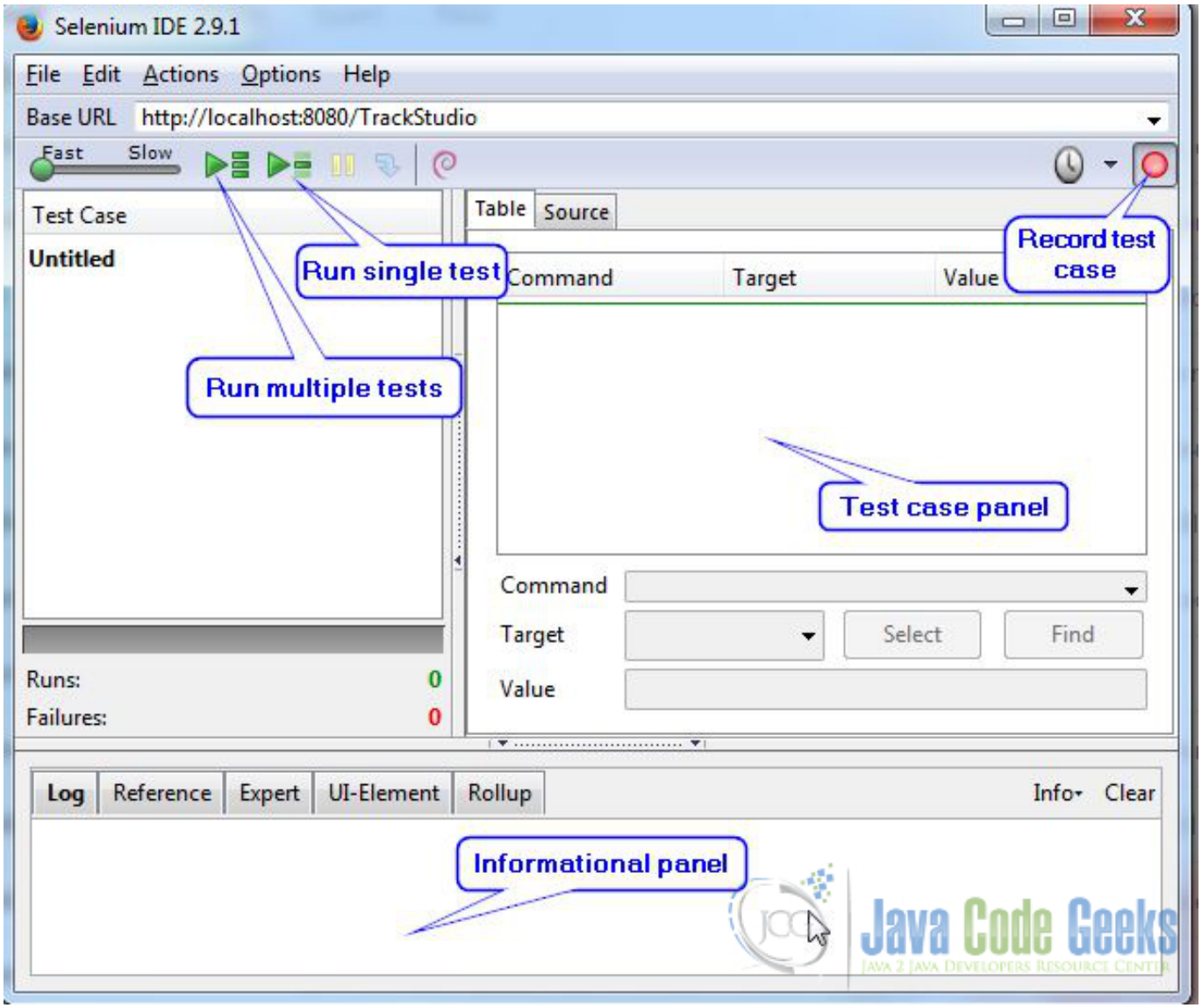


Figure 2.3: Selenium main controls

Let's describe the main useful controll:

- **Record user activities** - it is the toggle button, when it turns on, it records the user activities in the main browser window.
- **Run single test** - it run the single selected test.
- **Run multiple tests** - it run all tests, which it was opened in Selenium IDE.
- **Base url** - this input box defines the webapp url, when we go to this url in main browser window, Selenium IDE startes to record all activities on this window.
- **Test case panel** - this panel contains all user activities, which Selenium IDE records.

Next, we are ready to record our first test case.

2.3 Testing

Let's imagine, that we need to test the searching function in this site - `https://examples.javacodegeeks.com`. First, we should turn the record button on, put the base URL and go to main browser window. We go to this site, find the search input, type the necessary text, submit the form. After that, we get the searching result and now we need to verify this result. We need to select the necessary element and open the right mouse menu, as show below.

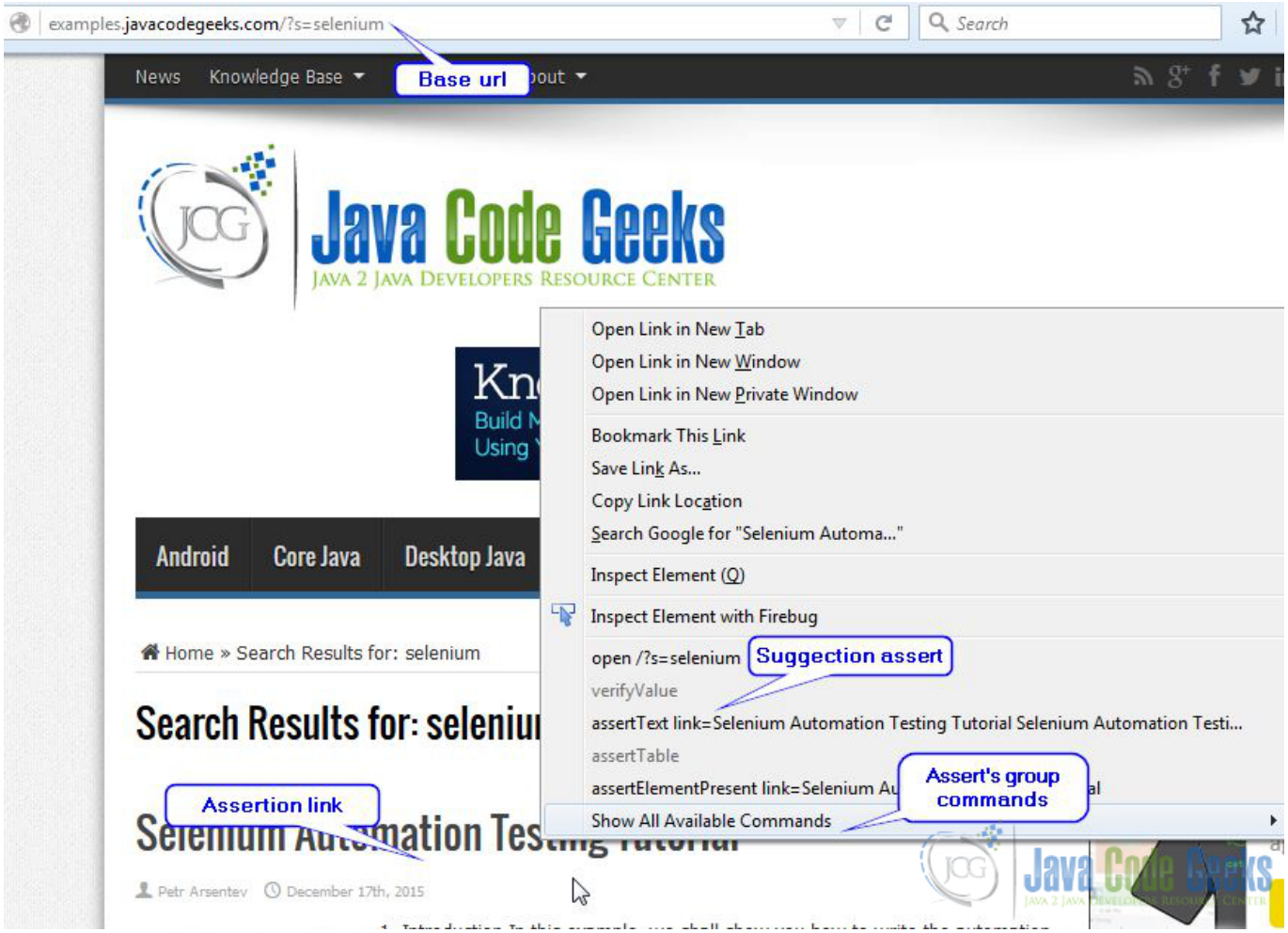


Figure 2.4: User controlls

This menu contains addition Selenium IDE options: list of available commands, assert's commands. We need to select the assert command, if we want to check the result. Now our test case is ready and we can go to Selenium IDE to see how it looks in Selenium code.

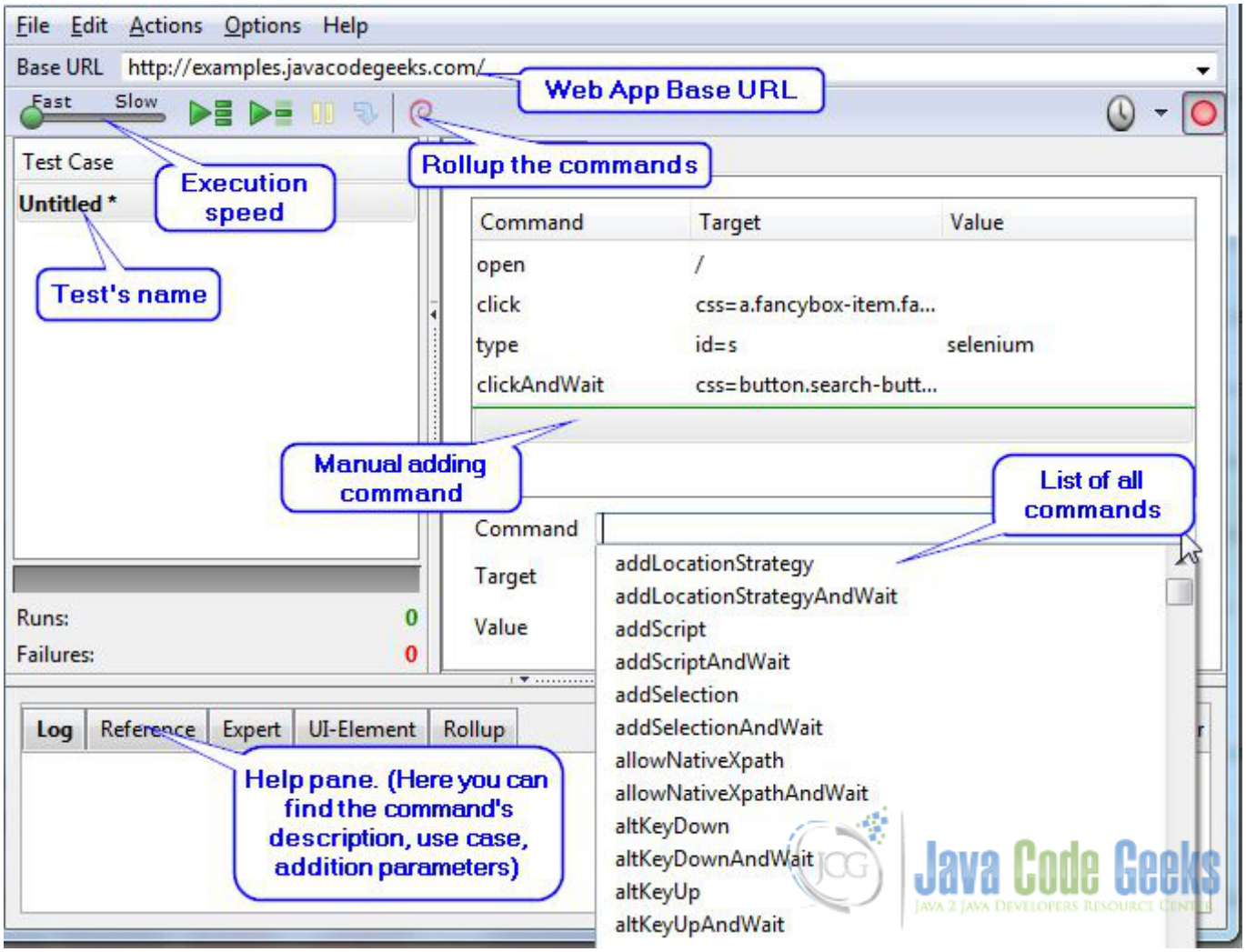


Figure 2.5: Manual adding command

How we can see the test case panel is fulled. This table contains the emulations user activities. Sometimes, we need to add the command manually.

You need to click on empty row in the table and fill the input controls below. Let's run our test and see the result. We click on run button and wait until test finishs.

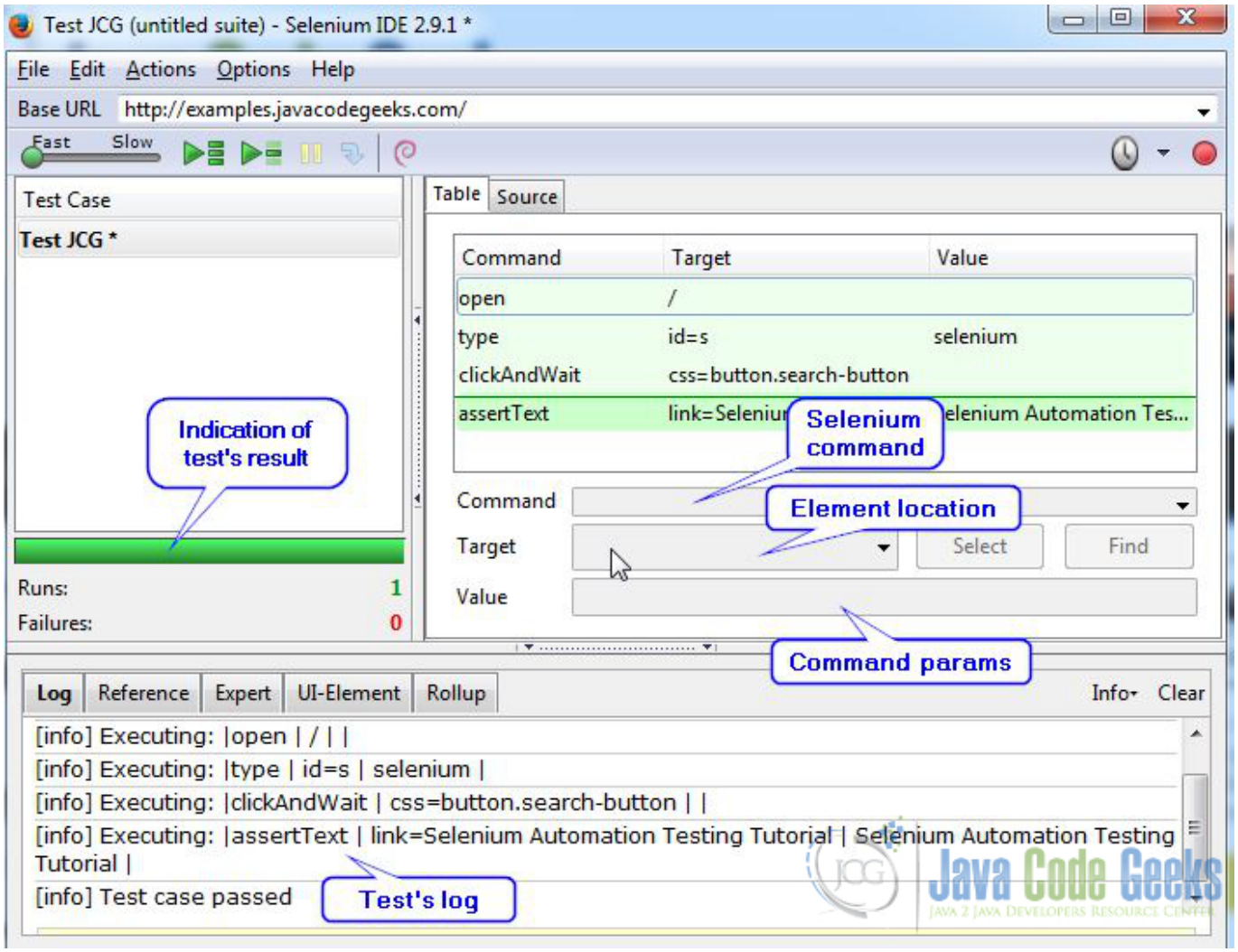


Figure 2.6: Run test

We need to save out test case, that we can use it further.

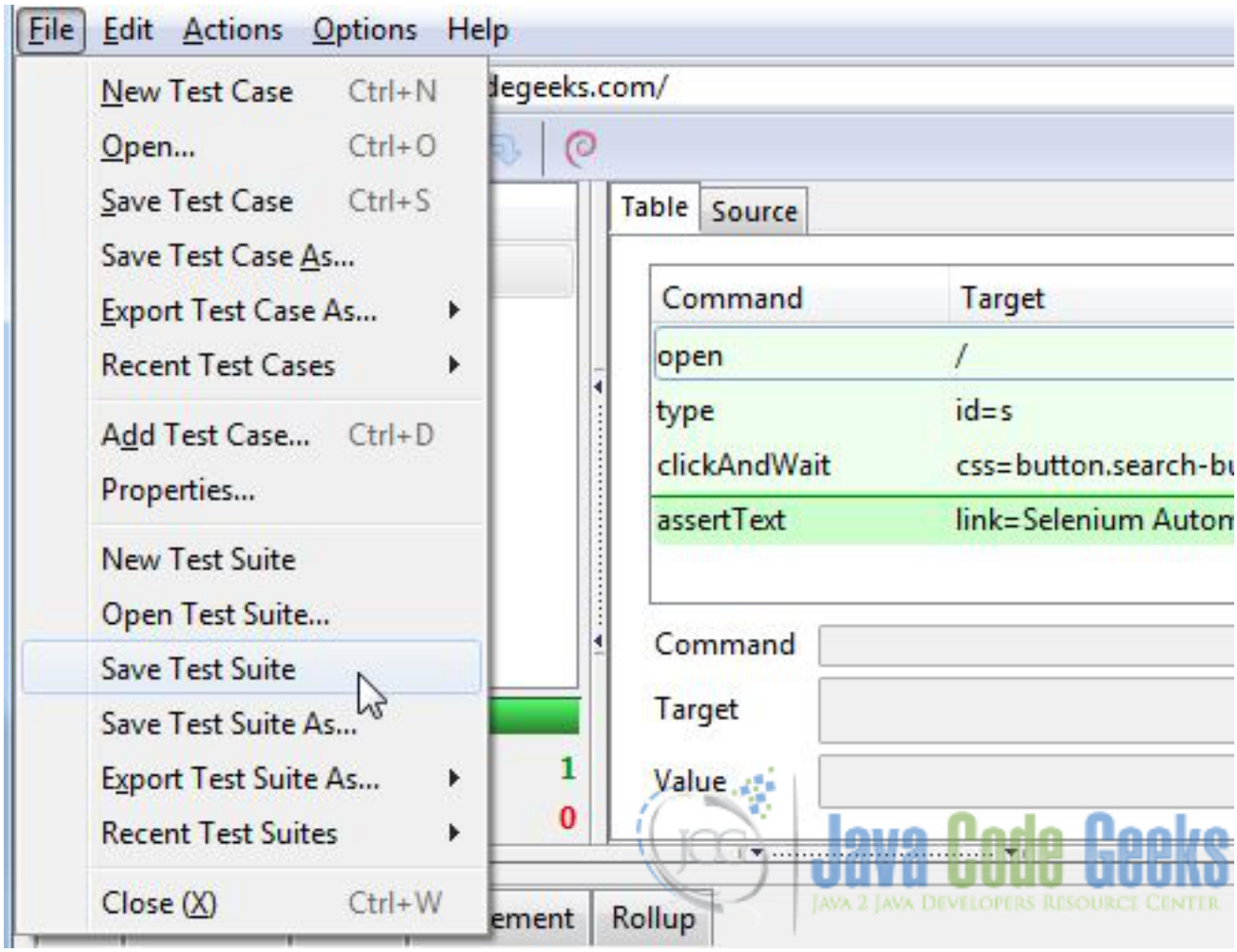


Figure 2.7: Save Test suite

Sometimes, we need to run the test periodically. Selenium IDE supports this functionality too. Go to `Main menu -Options -Schedule tests to run periodically`, as shown below. .Schedule controll Schedule controll

You can make the configuration in open windows. For example, we chose to run our test case every hour. Then you need to turn test schedule on.

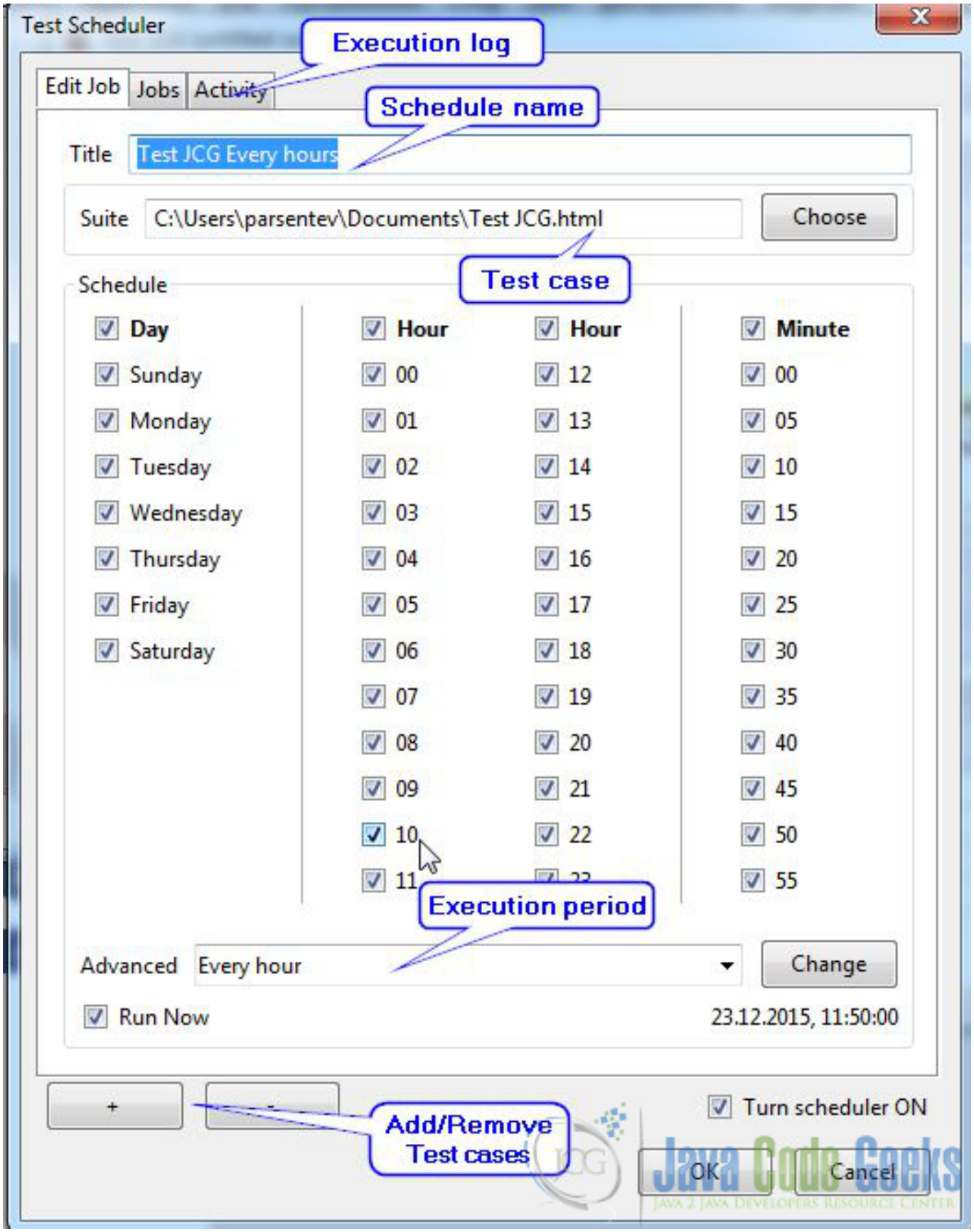


Figure 2.8: Schedule config

Selenium IDE can export your case tests to your favorite programming languages.

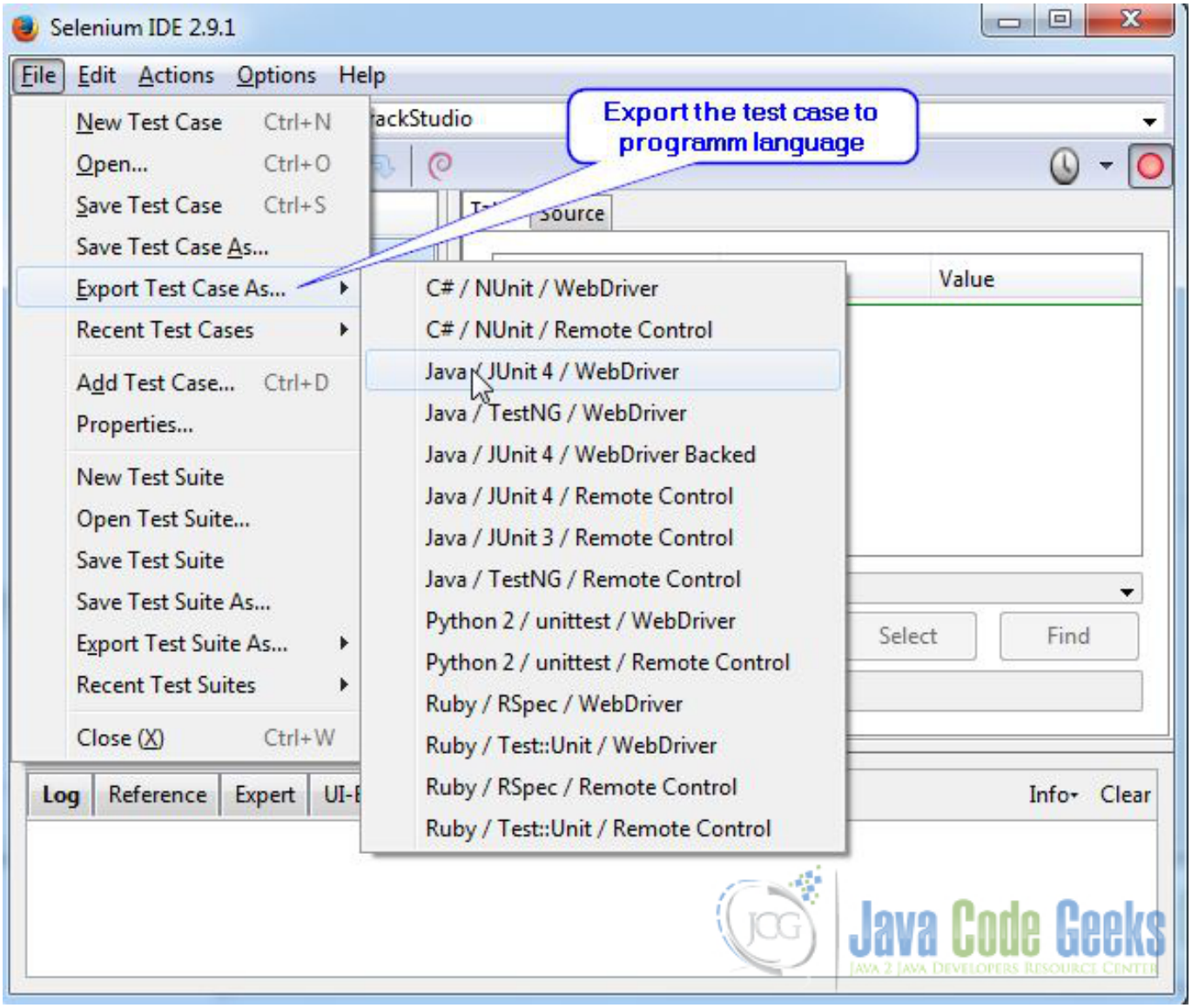


Figure 2.9: Export test cases

2.4 Conclusion

In this article, we show you how you can use Selenium IDE to test your web apps. Selenium IDE is the great tool for building automations tests easily.

- You can find more information about this tool on the official website link.

Chapter 3

Selenium Automation Testing Tutorial

3.1 Introduction

In this example, we shall show you how to write the automation tests by Selenium. Selenium is collections of tools to test the web applications. We are going to cover user's cases for TrackStudio system.

Automation tests has few benefits:

- It is easy for supporting.
- It is faster than manual tests.
- It has possibility to repeat tests.
- It has the lower cost with compare to manual tests.
- It has ability to use in Continue Integration.
- It has ability to get exhausted reports.

Selenium consists from three main parts:

- Selenium IDE - add-ons for Firefox.
- Selenium WebDriver - library, which use native browser API for testing.
- Selenium Server - server for execute tests in different environments.

3.2 Create a tests plan

Before we start to write the tests code, we should write the tests plan. The plan should describe what we should do and what we will expect after those actions. Let's tell few words about TrackStudio system. TrackStudio is issues/tasks tracker in the simple words.

It supports: workflow processing, documents processing, access rules control and so on. It fits great for our demonstration purpose, because it is the web app with rich user interfaces and it has many functional features.

Let's imagine that we want to test all cases for creating the library (books store) configuration in TrackStudio.

Table 3.1: Tests cases

Table 3.1: (continued)

Short explanation	Detail explanation	Expected result
Log in as administrator	1. Go to index page. 2. Fill the login form. 3. Submit the form.	Redirect to main page.
Create the new rule - Library manager.	4. Go to manage user tab 5. Open the rules tab. 6. Click on the create button 7. Fill the form. 8. Submit the form.	The rule should appear in rules list
Create the new rule - Reader.	1. Go to manage user tab 2. Open the rules tab. 3. Click on the create button 4. Fill the form. 5. Submit the form.	The rule should appear in rules list.
Create the new User with Manager rule - Bread.	1. Go to manage user tab 2. Open the users tab. 3. Click on the create button 4. Fill the form. 5. Submit the form.	The new user should appear in users list.
Create the new User with Reader rule - Smith.	1. Go to manage user tab 2. Open the users tab. 3. Click on the create button 4. Fill the form. 5. Submit the form.	The new user should appear in users list.
Create the new workflow - Book	1. Go to manage task tab. 2. Open the workflow tab. 3. Click on the create button. 4. Fill the form. 5. Submit the form.	The new workflow should appear in workflows list.
Create the two workflow states - in, out	1. Go to manage task tab. 2. Open the workflow tab. 3. Open the status tab. 4. Click on the create button. 5. Fill the form. 6. Submit the form.	The new status should appear in workflows status list.
Create the two workflow transaction - out to in, in to out	1. Go to manage task tab. 2. Open the workflow tab. 3. Open the transaction tab. 4. Selected the necessary status and click on the create button.	The new transactions should appear in the workflow.
Create the new category for Book workflow.	1. Go to manage task tab. 2. Open the category tab. 3. Click on the create button. 4. Fill the form. 5. Submit the form.	The new category should appear in the categories list.
Create the new project for library.	1. Go to manage task tab. 2. Click on the new project button. 3. Fill the form. 4. Submit the form.	The new project should appear in the task list.
Add the access for rule Manager and Reader to the library project	1. Go to manage task tab. 2. Open the access rule tab. 3. Selected the rules. 4. Submit the form.	The new access rule should appear in the access rules list.
Login as the Bread user and create the new book.	1. Log out. 2. Log in as Bread. 3. Go to the library project. 4. Click the new book button. 5. Fill the form. 6. Submit the form.	The new book should appear in the book list.
Login as the Smith user and take the new book.	1. Log out. 2. Log in as Smith. 3. Go to the library project. 4. Click the new book link. 5. Click the out operation button. 6. Fill form and submit it.	The book should change the status to out.

Now we can go to write your tests.

3.3 Install the Selenium IDE

First of all we should install the Selenium IDE. You should open the Firefox.Then you should go to official Selenium web site link.

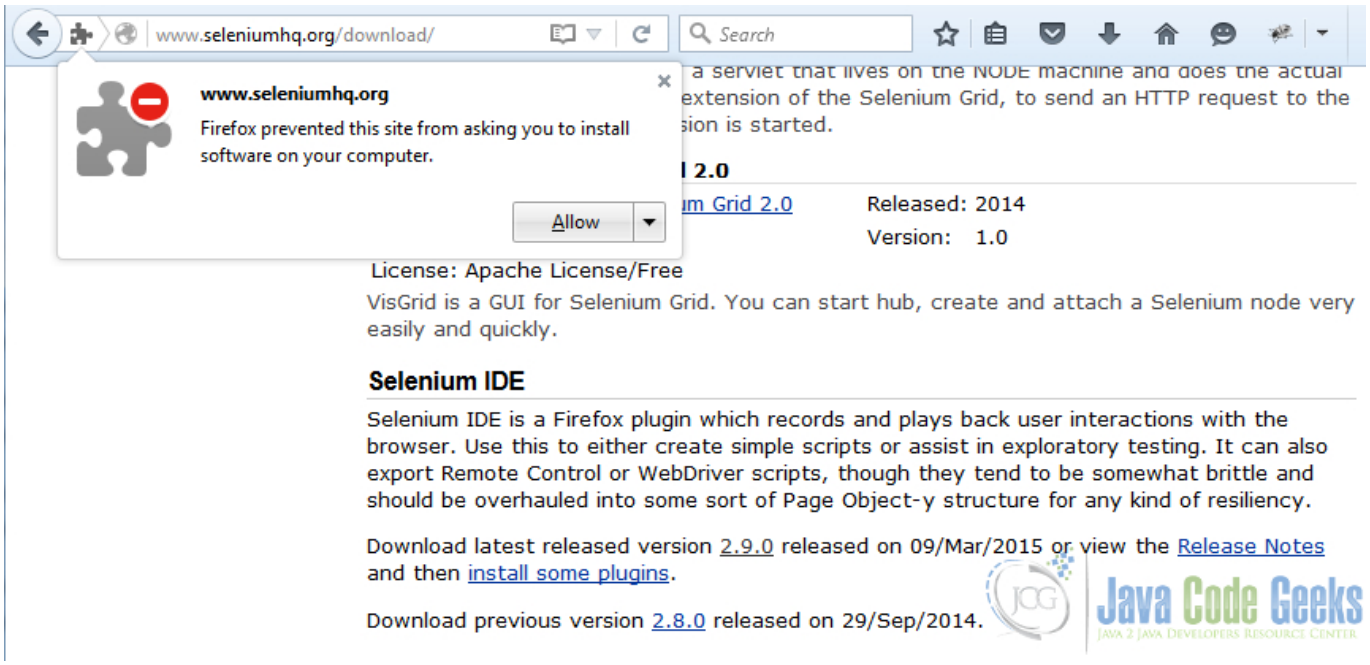


Figure 3.1: Download Selenium IDE

After you restart the firefox browser you can see the button in the right top corner. It is the Selenium IDE.

Figure 3.2: Selenium IDE Button

Then we should click on this button and Selenium IDE opens.

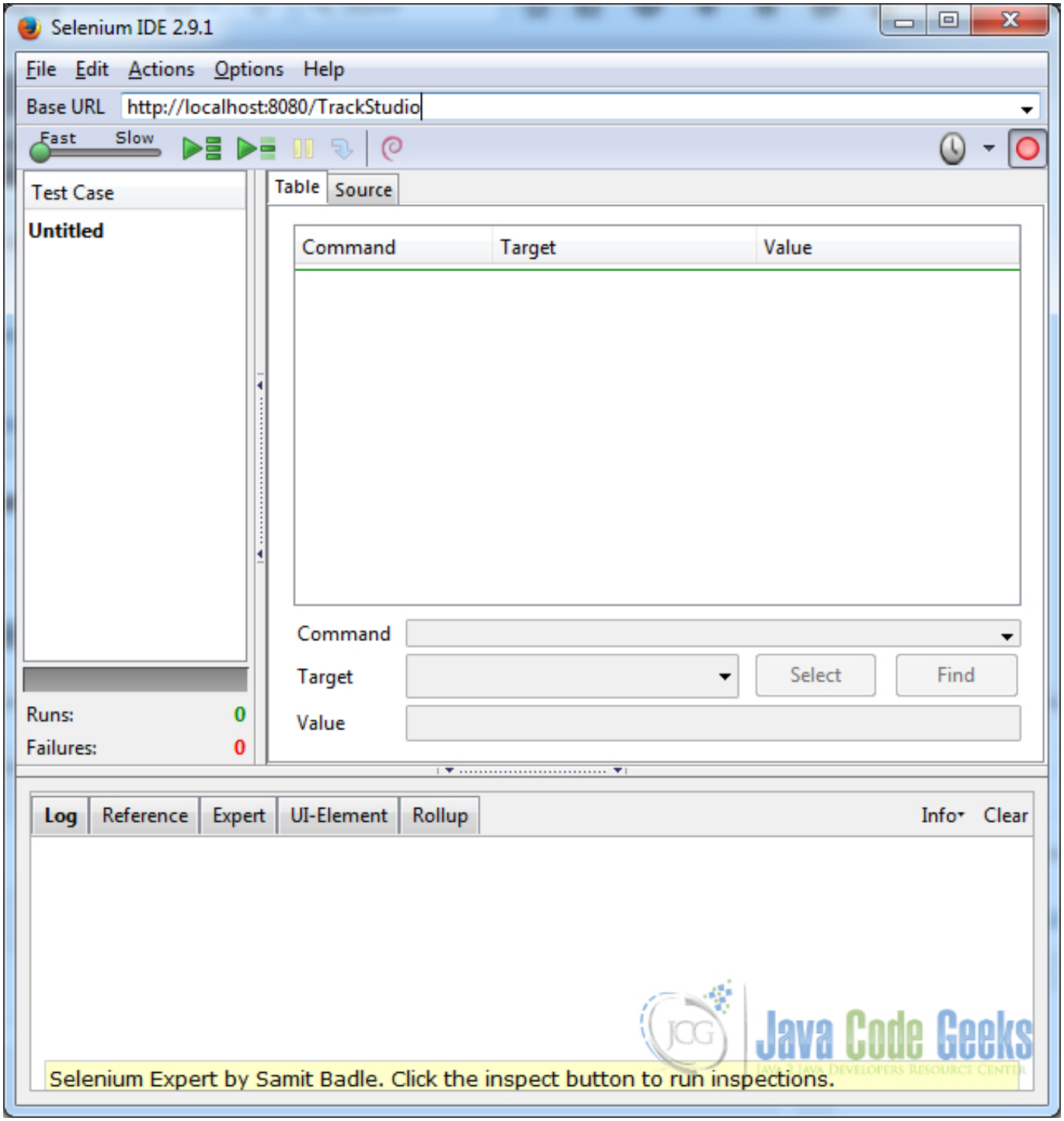


Figure 3.3: Selenium IDE

Now we are ready to record out test cases.

3.4 Record the user activities in Selenium IDE

Firstly, we should run the TrackStudio Server. You should open the FireFox and open the URL `https://localhost:8080/TrackStudio/` It will look similar:

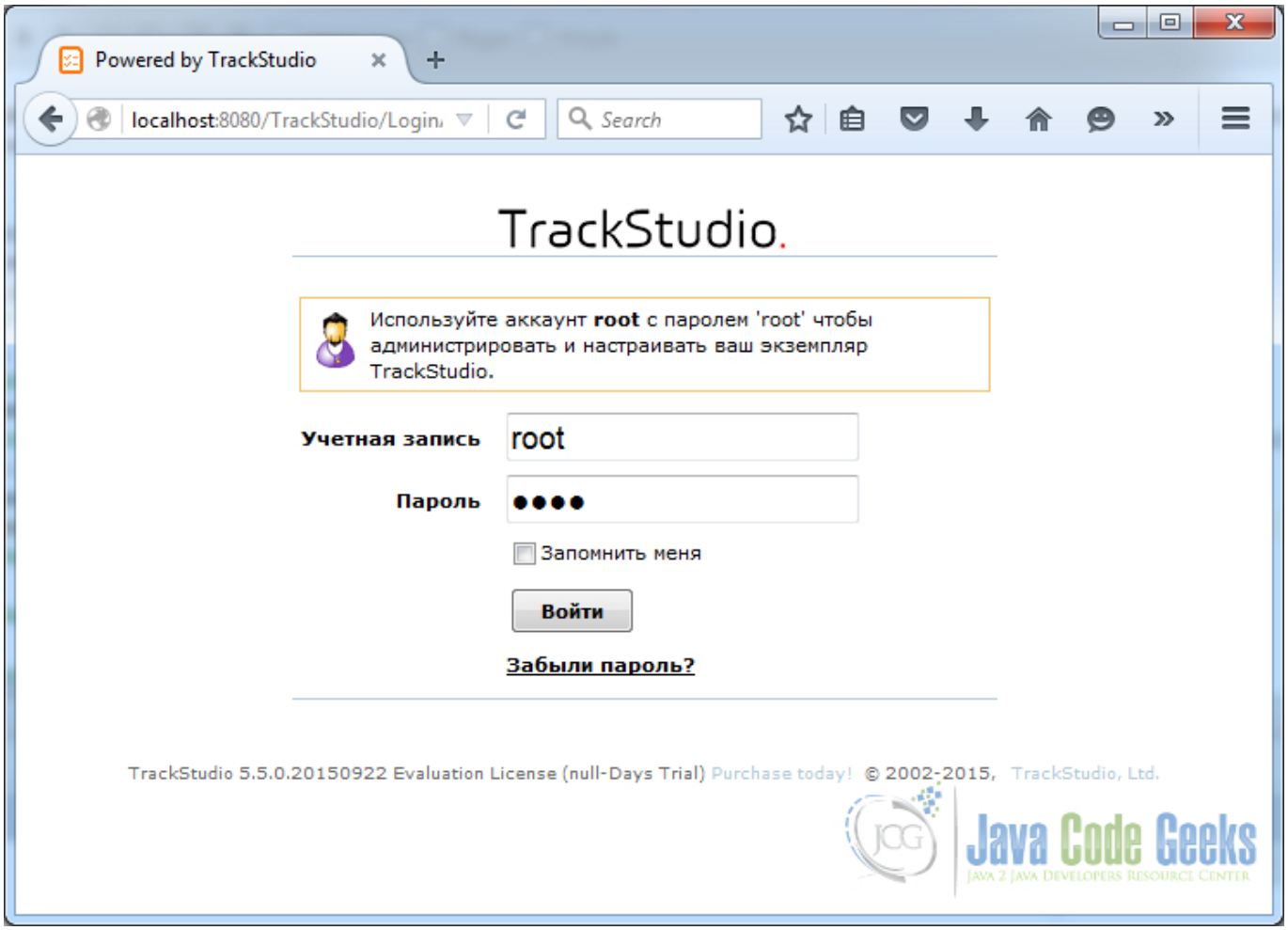


Figure 3.4: TrackStudio Login Page

Now we need to record our user's cases. Open the Selenium IDE. You have to check that Base URL is the same as TrackStudio Login page and the button record turns on.

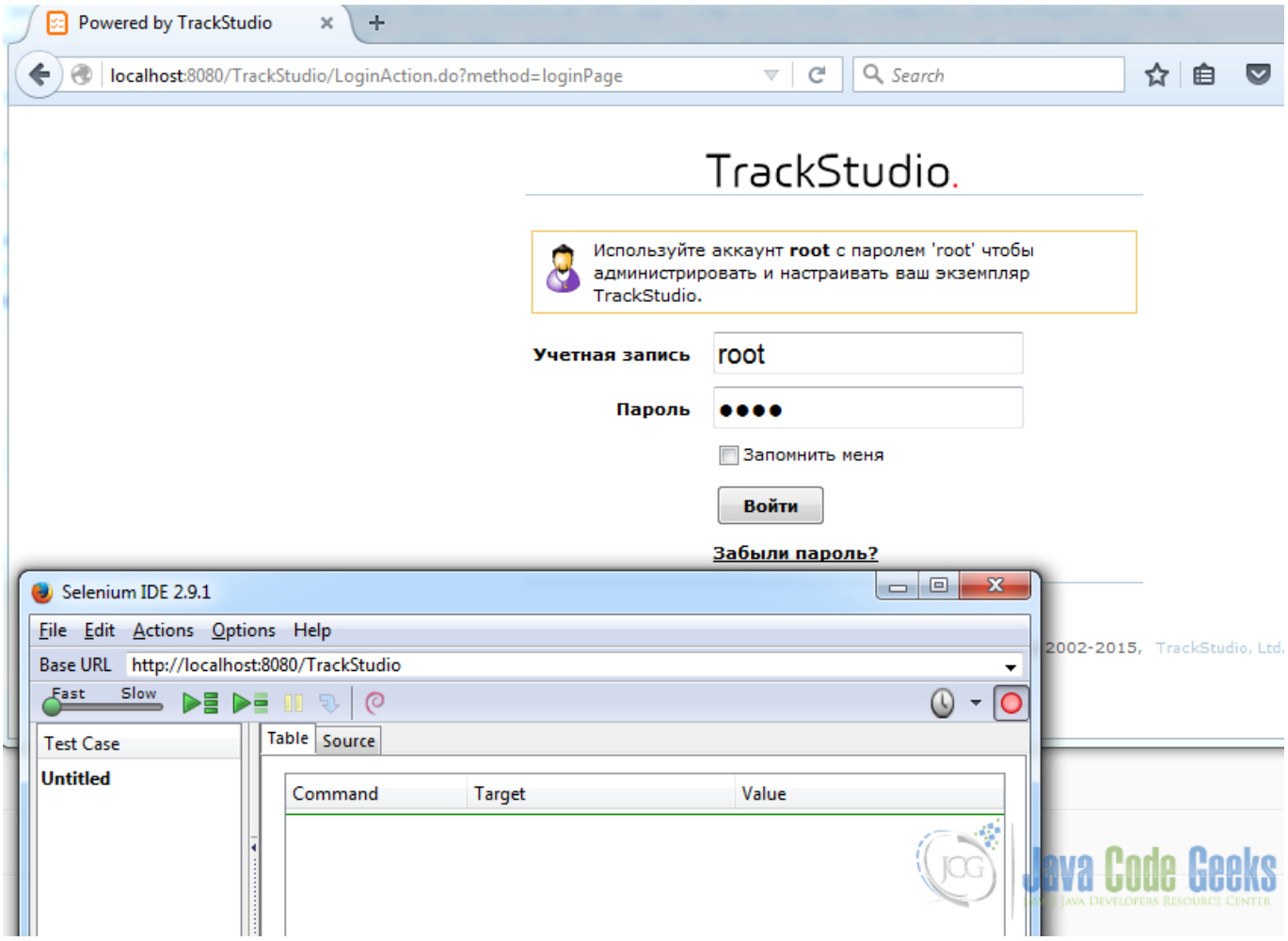


Figure 3.5: Start to record the user activities

Then you start to make the actions in browser as general users: click on link, fill the form and so on. Selenium IDE will records all this activities. After you finished your actions, you can see that Selenium IDE filled the table.

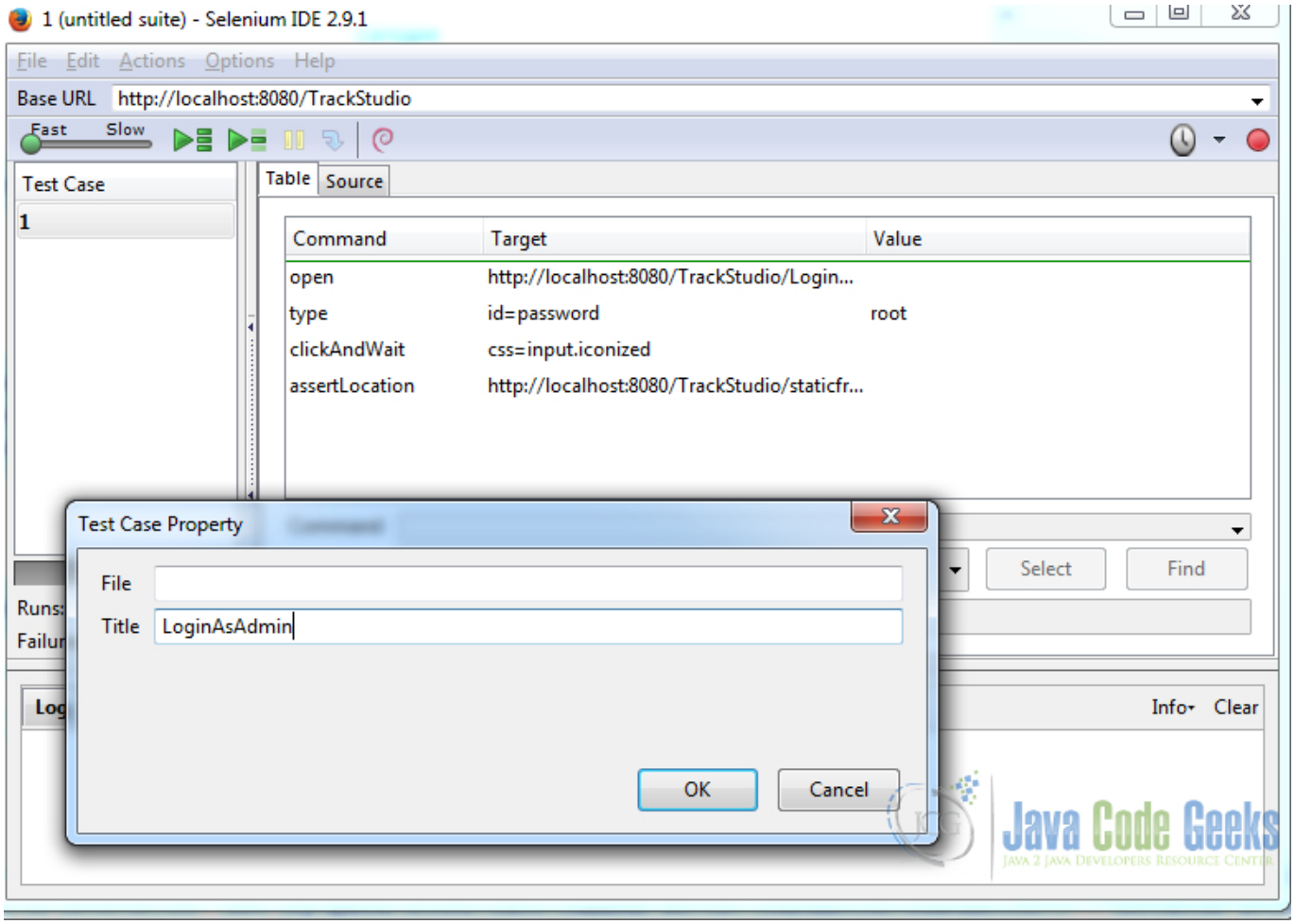


Figure 3.6: Login As Admin

We do other tests similar actions. After that, we should get the list with our tests cases.

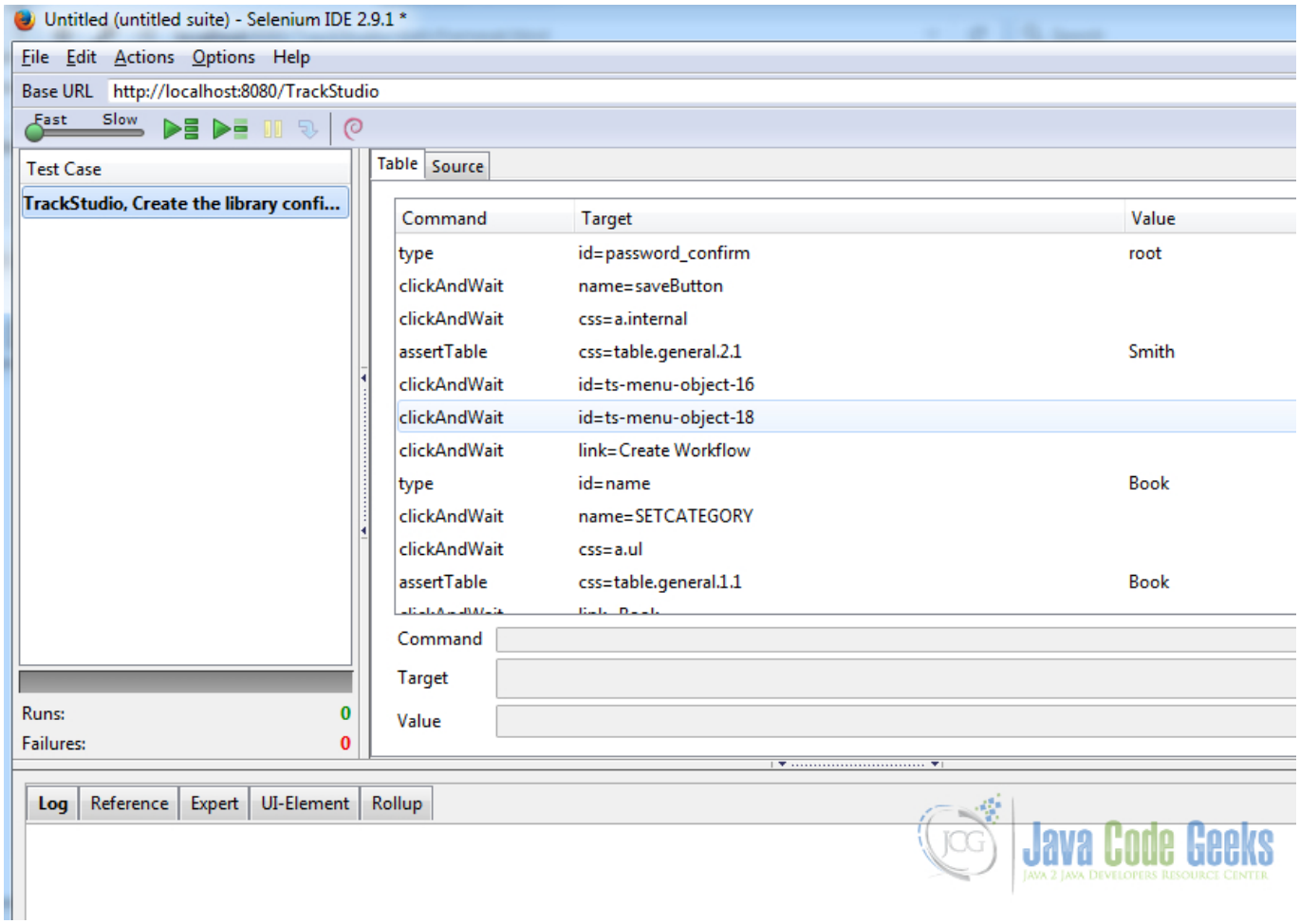


Figure 3.7: TrackStudio Selenium IDE Test Cases

When you need to assert the text in the web page, you should select the necessary element and open the popup menu, like this:

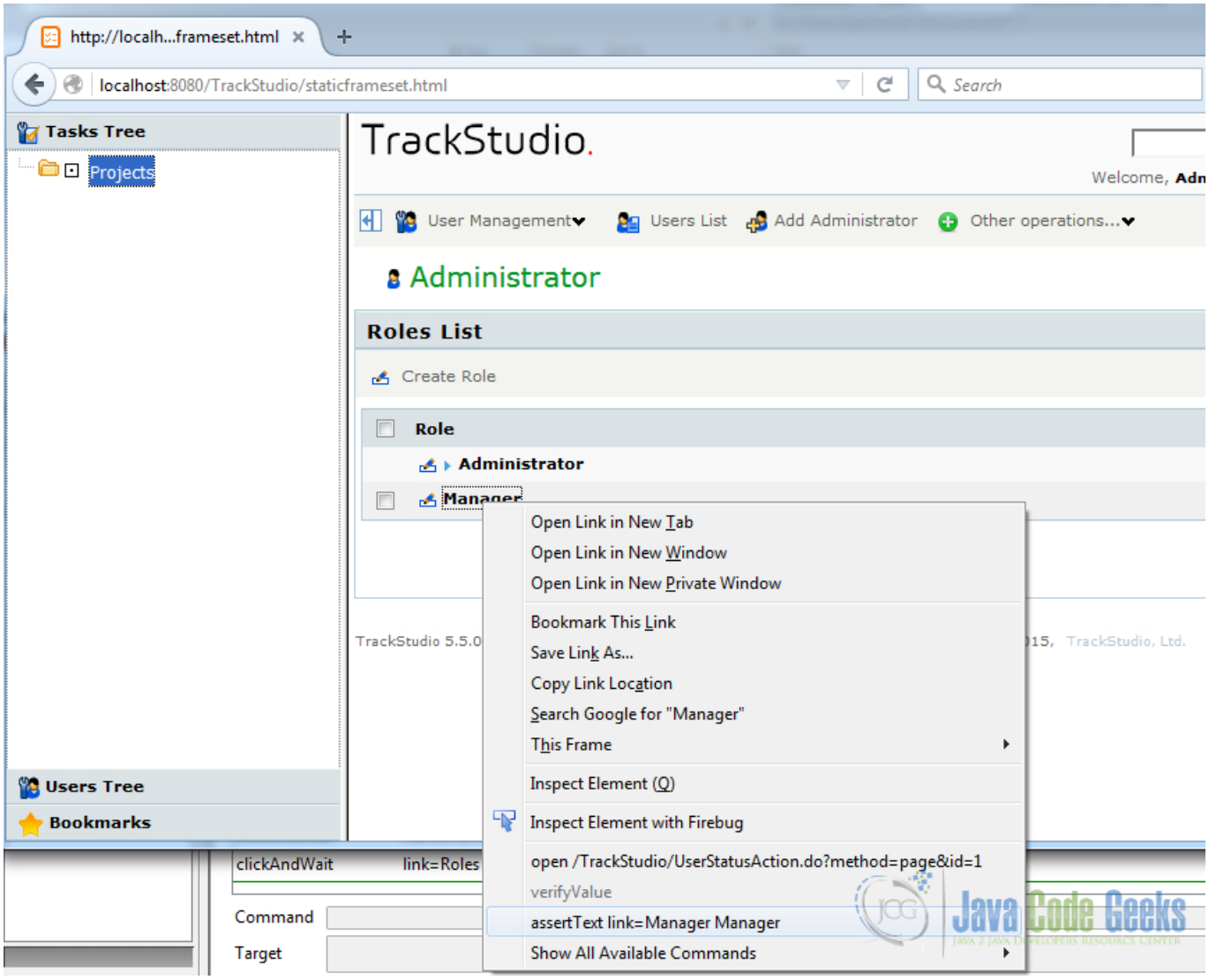


Figure 3.8: Assert Link

Selenium IDE inserts there assertions in the tests code. So now we are ready to export the code to our favorite programming language from Selenium IDE. You should go to `File->Export->Select language`

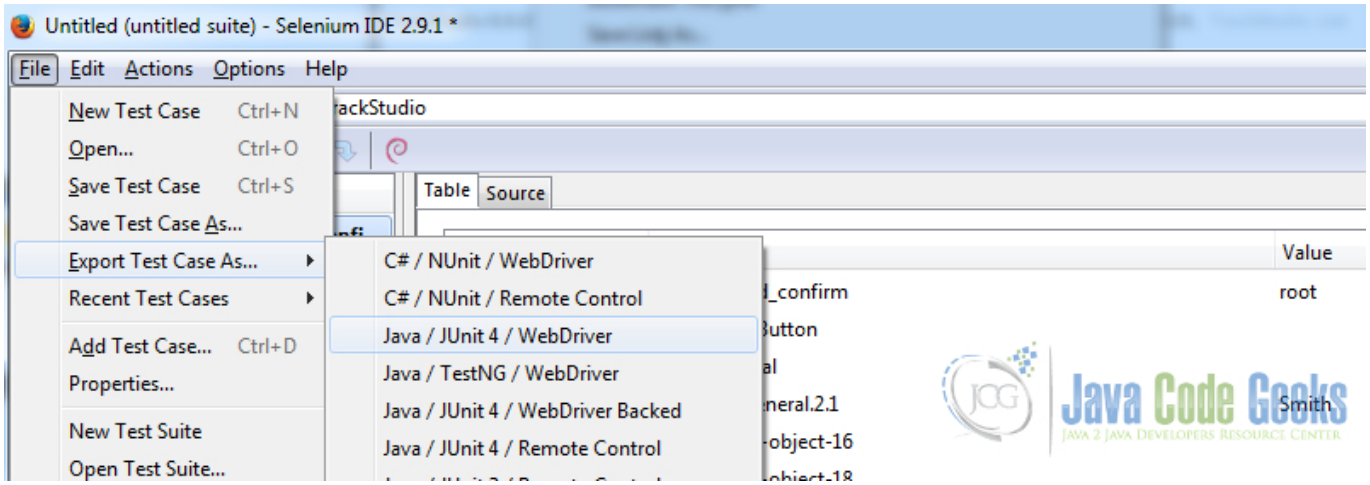


Figure 3.9: Export

3.5 Refactoring exported code

Now we have the source code on your tests cases. Then we should create the maven project with selenium dependency and put there your test code.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="https://maven.apache.org/POM/4.0.0"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/ ↩
             xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>ru</groupId>
    <artifactId>parsentev</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>2.48.2</version>
        </dependency>
    </dependencies>
</project>
```

Now we can open our new project in Eclipse and make the refactoring. How you can see below the exported code does look good. It has duplicates codes.

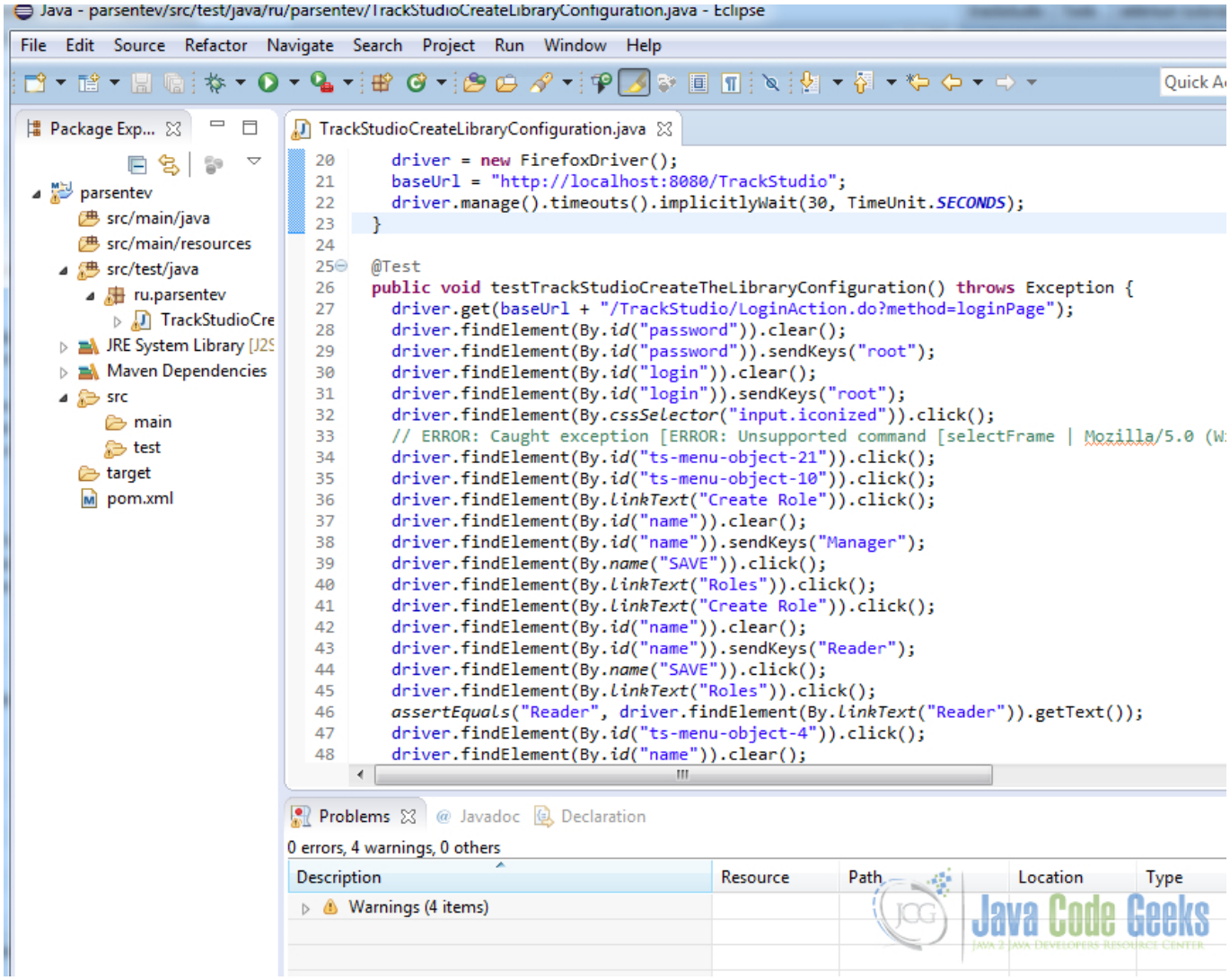


Figure 3.10: Eclipse

First of all, we should extract all use data to specific file. It is the design approach - Map UI.

TrackStudioData.java

```
package ru.parsentev;

/**
 * File contains the trackstudio user data.
 * @author parsentev
 * @since 14.12.2015
 */
public interface TrackStudioData {
        String ROOT_LOGIN = "root";
        String ROOT_PWD = "root";
        String BREAD_LOGIN = "root";
        String BREAD_PWD = "root";
        String SMITH_LOGIN = "root";
        String SMITH_PWD = "root";
        String MANAGER_ROLE = "manager";
        String READER_ROLE = "reader";
        String CATEGORY_NAME = "book";
```

```
        String WORKFLOW_NAME = "book";
        String TASK_NAME = "the catcher in the rye";
        String TR_IN = "in";
        String TR_OUT = "out";
}
```

The next step is to split out main on the few little as we described in table tests ceases.

TrackStudioCreateLibraryConfiguration.java

```
@Test
        public void cases() throws Exception {
                this.loginAsAdmin();
                this.createManagerRole();
                this.createReaderRole();
                this.createUserManager();
                this.createUserReader();
                this.createWorkflow();
                this.createCategory();
                this.createTask();
                this.takeTaskByReader();
        }
```

We have the automation tests, which you can run the follow command `mvn clean test`. You can integrate it to CI easily.

3.6 Download the Maven project

Download

You can download the full source code of this example here: **SeleniumAutomationTests.zip**

Chapter 4

Selenium Interview Questions and Answers

4.1 Introduction

In this example we shall show the most popular interviews questions about Selenium tools and give you exhaustive answers. Questions cover all topics:

- Selenium 1
- Selenium 2
- Selenium IDE
- Selenium Standalone Server

In this article is shown theoretical questions and best practices, which is used by Selenium communities.

4.2 Interview Questions and Answers

What is Selenium?

Selenium is the complex of tools for automation testing the web applications. This tools is developed in Java. You can use different approaches for create the tests: using Selenium IDE, using one of popular programming languages (Java, C#, Perl, Python). Because right now web applications become very interactive it is to difficult to write automation tests for its. Selenium uses two kind of approaches for doing it: directly using browser api (new version), injecting JavaScript (old version).

What modules does Selenium consist?

Selenium has few different modules:

- Selenium IDE - firefox plugin, The plugin has two modes : records user activities, run the tests by user activities.
- Selenium 1 (Selenium RC) - library, which converts the code to JavaScript and injects it to the browser.
- Selenium 2 (Selenium WebDriver) - library, which uses the browser api for testing.
- Selenium Server (Selenium Grid) - server. It is used to run tests in different environments.

What the technology is used for searching elements in Selenium?

Selenium is used DOM for searching elements. The DOM can look like the Tree Structure.

How to find different type of elements in Selenium, explain few different ways?

Let's consider this question by simple HTML example

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Example page for Selenium Test</title>
</head>
<body>
<iframe name="tree" src="tree.html"></iframe>

    <div class="form">
        <form about="/" method="POST">
            User name : <input name="username" type="text">
            <a href="/agreements.html">User agreements</a>
        </form>

</div>
</body>
</html>
```

You can search the elements by different criterias:

- By Element ID `WebElement element =driver.findElement(By.id("general"));`
- By Class name `WebElement element =driver.findElement(By.className("form"));`
- By Tag name `WebElement element =driver.findElement(By.tagName("iframe"));`
- By Name `WebElement element =driver.findElement(By.name("username"));`
- By Link Text `WebElement element =driver.findElement(By.linkText("User agreements"));`
- By Partial Link Text `WebElement element =driver.findElement(By.partialLinkText("agreements "));`
- By XPATH `List inputs =driver.findElements(By.xpath("//input"));`
- Using JavaScript `(WebElement) ((JavascriptExecutor)driver).executeScript("return $('.general')[0]");`

How to fill different type of input elements?

Let's consired another HTML page:

create.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Create new user</title>
</head>
<body>
<form action="/create" method="post">
    User name :<input type="text" name="username">
    Agreement : <input type="checkbox" name="agreement">
    Sex :
    <input type="radio" name="sex" value="male">Male
    <input type="radio" name="sex" value="female">Female
    Country :
    <select name="country">
        <option value="1">Country 1</option>
        <option value="2">Country 1</option>
    </select>
```

```
    Description :
    <textarea name="desc"></textarea>
    <input id="submit" type="button" value="Submit">
</form>
</body>
</html>
```

The main approach looks like : get necessary elements, fill the data.

```
/**
 * Test filling the form with different input elements.
 */
public void fillForm() {
        //fill the user name
        driver.findElement(By.name("username")).sendKeys("Petr");
        //checked agreement
        driver.findElement(By.name("agreement")).click();
        //choose sex
        driver.findElements(By.name("sex")).get(0).click();
        //select country
        WebElement select = driver.findElement(By.tagName("select"));
        select.findElements(By.tagName("option")).get(0).click();
        //fill description
        driver.findElement(By.name("desc")).sendKeys("Petr");
        //push on button
        driver.findElement(By.id("submit")).click();
}
```

How to switch between pupops, frames, windows?

Let's imagine that user opens the two windows `index.html` and `create.html` and we need to switch between them. You can do it by pointing the title of necessary windows. For example : `driver.switchTo().window("Create new user");` We can use this approach to switch frames by its name. For example `driver.switchTo().frame("tree");`

What is Selenium IDE? When is it used?

Selenium IDE is firefox plugin. It has frendly user interface. It is used for developing the automation tests. Selenium community recommend to start to learn Selenium with this tools. The use cases look like to record user activites. IDE will gerenete the test code and after that you can run this tests. First of all you need to install the firefox plugin. It looks like below:

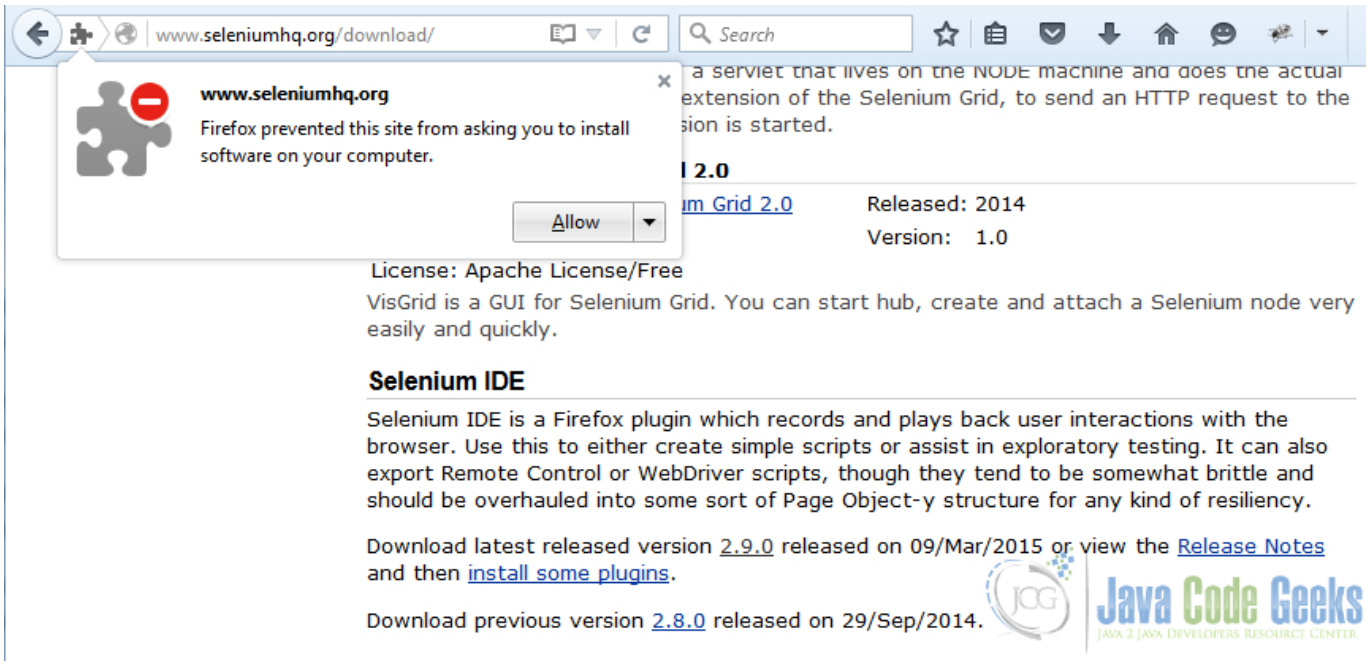


Figure 4.1: Download Selenium IDE

After you get your test cases code you can export it to favotire programming languages : Java, Ruby, Python, C#. It looks like below:

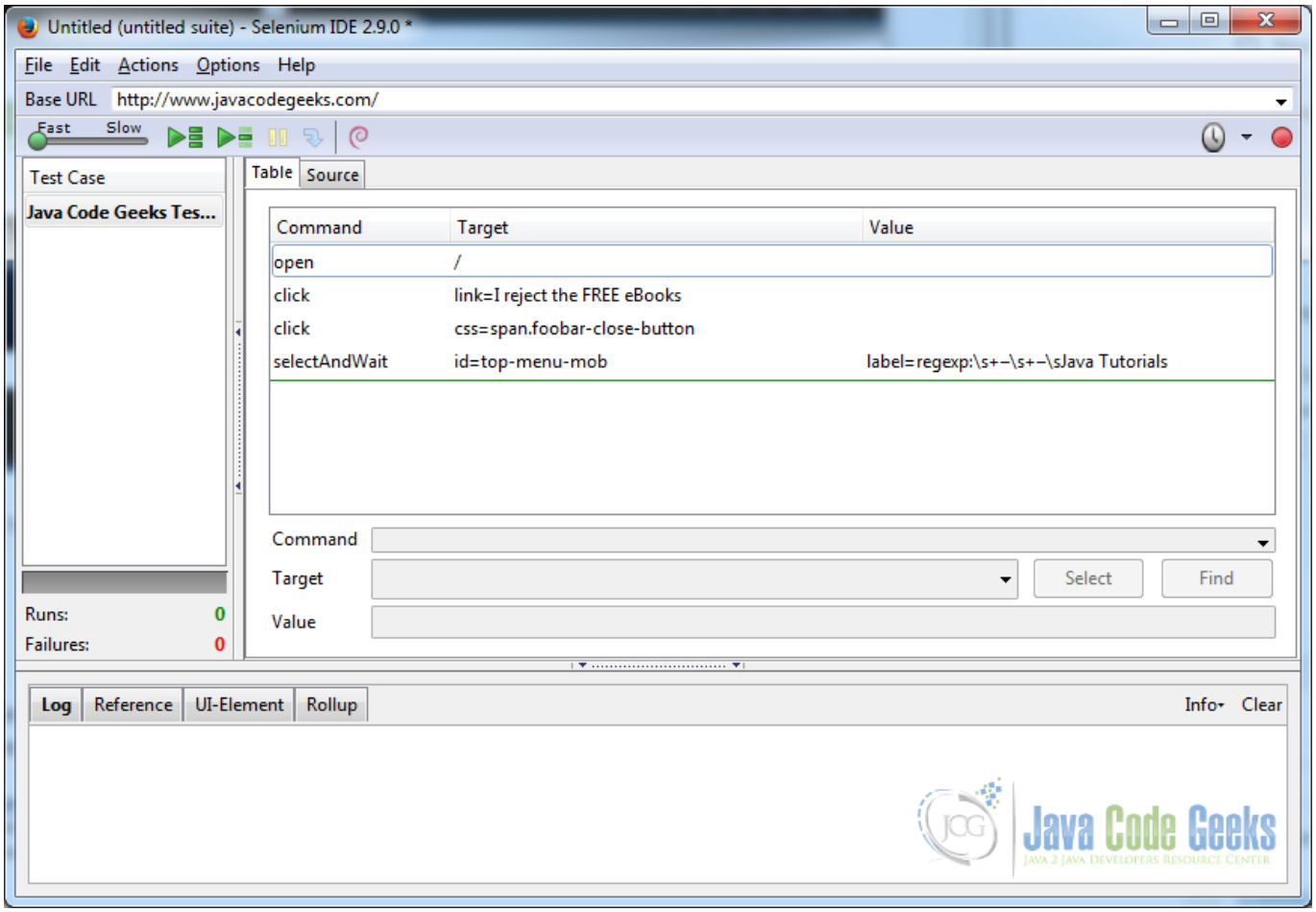


Figure 4.2: Selenium Test IDE

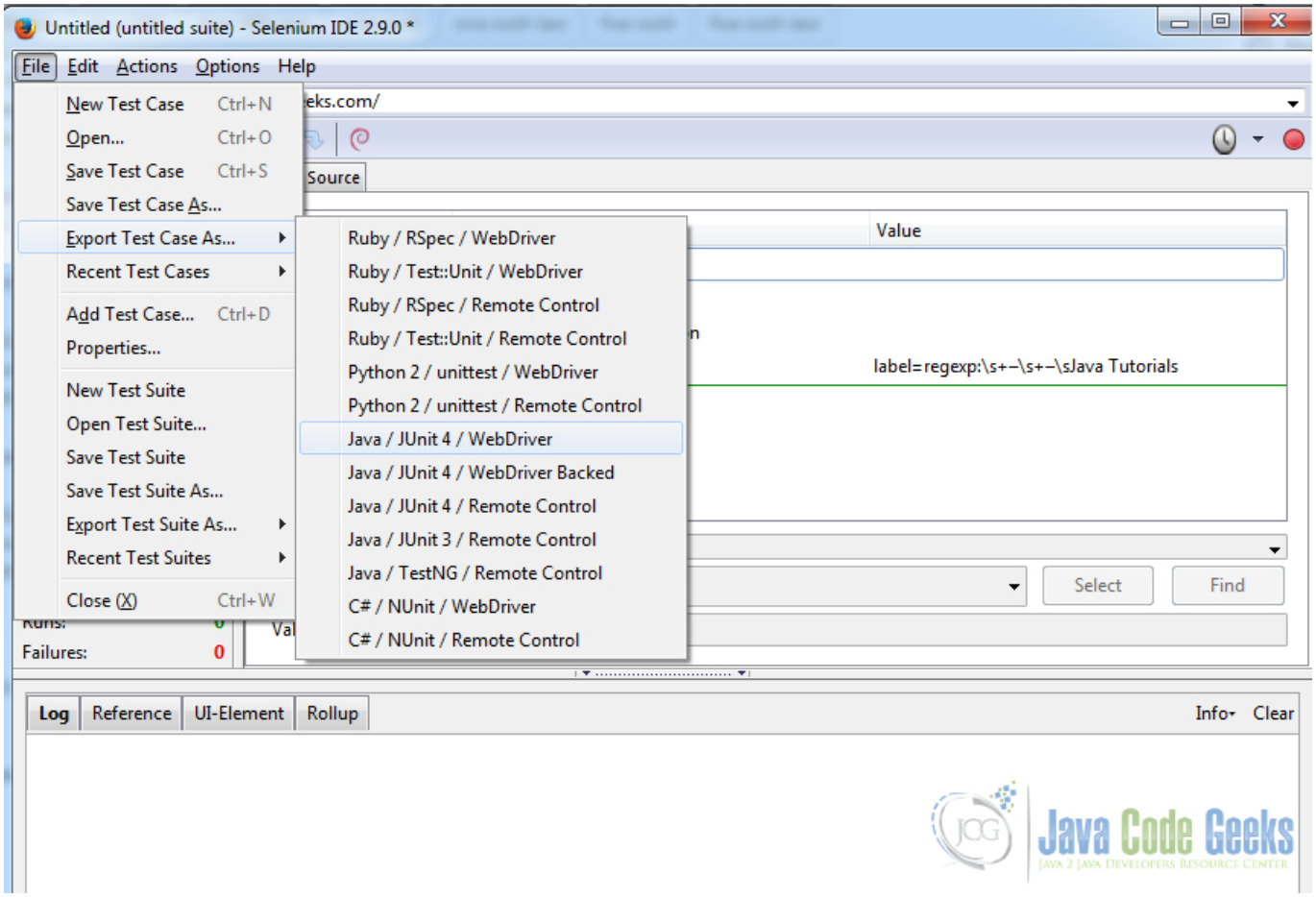


Figure 4.3: Export IDE

After that you get the source code file then you can open in your favorite IDE. The code should look similar :

JavaCodeGeeksTests.java

```
package com.example.tests;

import java.util.regex.Pattern;
import java.util.concurrent.TimeUnit;
import org.junit.*;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class JavaCodeGeeksTests {
  private WebDriver driver;
  private String baseUrl;
  private boolean acceptNextAlert = true;
  private StringBuffer verificationErrors = new StringBuffer();

  @Before
  public void setUp() throws Exception {
    driver = new FirefoxDriver();
    baseUrl = "https://www.javacodegeeks.com/";
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
```

```
  }

  @Test
  public void testJavaCodeGeeksTests() throws Exception {
    driver.get(baseUrl + "/");
    driver.findElement(By.linkText("I reject the FREE eBooks")).click();
    driver.findElement(By.cssSelector("span.foobar-close-button")).click();
    new Select(driver.findElement(By.id("top-menu-mob"))).selectByVisibleText("regexp:\\s ↩
        +-\\s+-\\sJava Tutorials");
  }

  @After
  public void tearDown() throws Exception {
    driver.quit();
    String verificationErrorString = verificationErrors.toString();
    if (!"".equals(verificationErrorString)) {
      fail(verificationErrorString);
    }
  }

  private boolean isElementPresent(By by) {
    try {
      driver.findElement(by);
      return true;
    } catch (NoSuchElementException e) {
      return false;
    }
  }

  private boolean isAlertPresent() {
    try {
      driver.switchTo().alert();
      return true;
    } catch (NoAlertPresentException e) {
      return false;
    }
  }

  private String closeAlertAndGetItsText() {
    try {
      Alert alert = driver.switchTo().alert();
      String alertText = alert.getText();
      if (acceptNextAlert) {
        alert.accept();
      } else {
        alert.dismiss();
      }
      return alertText;
    } finally {
      acceptNextAlert = true;
    }
  }
}
```

What is implicit and explicit wait?

The all web apps load and fetch the data from server. sometimes it could not be faster. for this reason the test case should wait some events which should be happended on page. Selenium provides two kind of method to do this job.

- explicit wait - we don't know how manytimes the event should take so we predict some changes in UI. For example load the new elements : `(new WebDriverWait(driver, 30)).until(ExpectedConditions.presenceOfElementLocated(By.id("comments")));` here we pointed `30` - timeout in milliseconds, if it exceeds the test will be

interuppeted. `ExpectedConditions.presenceOfElementLocated(By.id("comments"))` - expect the show new div block with `id="comments"`

- implicit wait - we predict that event takes definalty times. For example `driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);`

What kind of tests does selemiun support? Selemiun supports variety tests cases:

- Functional tests
- Regressive tests
- Testing Static Content
- Testing Links
- Testing Dynamic Elements
- Ajax Tests

What is UI Map?

This is development approach when we separate tests data and login codes. For example we have some credential to auth in web site. we create the special class where we should store all such data and use it in tests codes. For example:

UserCredential.java

```
/**
 * Credential info.
 * @author parsentev
 * @since 26.11.2015
 */
public class UserCredential {
        public static final String LOGIN = "login";
        public static final String PASSWORD = "password";
}
```

So when you need to change the user credential, you go to this file and change only this code without searching all places where you use it.

What is Page Object Pattern? How is it used in Selenium?

This pattern is used in automation tests overall. The main advantages are hiding the implementation of auth details and splitting the all web apps tests on small undescended pages.

What is DDT and how is it supported in Selenium?

DDT is the data tests. Selenium does not have special mechanism for working with DDT. You can use another libraries for it. For example, read data from database by JDBC, or read data from file and so on.

What is Selenium Grid? Where is it used?

Selenium Grid is distributed servers which offers to run tests in different enviroments.

How to execute the Selenium tests in parallel?

It can be done by Selenium Gird or by using Thread in Java directly.

What kind of problems do you have when using Selenium?

First, it is not so faster to change the automation tests when UI in web app can be changed faster. However, these problems have all automation testing tools. Another problem is appear in dynamic generation IDs. If you use strategy search by ID you should change it to XPATH. One another problem is working with AJAX. Selenium supports AJAX executing but tester should predict how many times the ajax call takes. You can use explicit or implicit wait for solve with problem.

Can Selenium be integrated to CI?

Yes. It can, because all tests can be exported to programming languages and then it can be run independent from Selenium IDE by JUnit, TestNG and similar libraries

Can you use Selenium in commercial product?

Yes, you can. Selenium is distributed by Apache 2.0 license. It means that you can use it in commercial products free without paing any charge.

What kind of disadvantages does Selenium IDE have?

After you recorded the user activates Selenium IDE generates the case tests tables. This tests table does not support the loops, functions, conditions statements, so you cannot use all programming languages constructions.

What is Selenium Remote Control?

It is the test tool for web apps. It uses injecting JavaScript to browser for testing.

What advantages does Selenium have for compare with another web testing tools?

First, it is to start to write the tests, because you can record the user activity directly, Another thing is Selenium supports many programming languages : Java, C#, Python, Ruby.

Can you install the Selenium IDE to another browsers then FireFox?

No. Selenium IDE can be installed only to FireFox.

What kind of browsers can be executed the tests by Selenium?

- Firefox 3.x
- IE 6-8
- Safari 2-4
- Opera 8-10
- Chrome

Can you use Selenium for testing mobiles app?

Yes. if it can be run in mobiles browsers.

What is Selenese?

After you recorded the user activities, Selenium IDE generates the code, this code is written by Selenese.

How can you run the tests in HTTPS?

You need to use Selenium Server and configure the necessary securities environments.

How can you configure the Selenium Grid?

You can do in two ways: 1. adding special key when run the server, 2. using JSON config file.

What is HtmlUnitDriver?

HtmlUnitDriver is web driver which is used HtmlUnit engine. It is the faster web driver in Selenium.

Can you use navigations browser menu in Selenium tests?

Yes. you can. You can handle it with the following code:

```
driver.navigate().back(); driver.navigate().forward(); driver.navigate().refresh(); dri
ver.navigate().to();
```

How can you save the tests result?

You can take the screen shot if tests fail or you can store the testing result to database by JDBC.

Can you run the load tests by Selenium?

Yes, but Selenium is not good choose for such kind of tests. Because it is expense to open the multiple browser at the same time. if you need to test loading, you can use JMeter or Gattling.

4.3 Conclusion

In this acticle we explained most of frequently ask questions in interviews and gived the answers. Of course this is not all. if you want to improve your knowledge about Selemiun go to official web site Selenium Official Documentation.

Chapter 5

Selenium Standalone Server Example

5.1 Introduction

With this example we are going to demonstrate how to use and configure Selenium standalone servers (Selenium Grid). We are going to run the hub server and the two nodes. Each nodes will run the tests in different browsers.

Selenium Grid are the servers which compounds in distributed nodes. It offers you to run your selenium test on separate machines in different kinds environments. This is great opportunities, because your tests can run parallels and use different browsers for testing.

5.2 High level architecture

Below is shown high level architecture. It has the follow flow process. First we pass the tests to hub which send this tests to specific nodes where all tests will be executed.

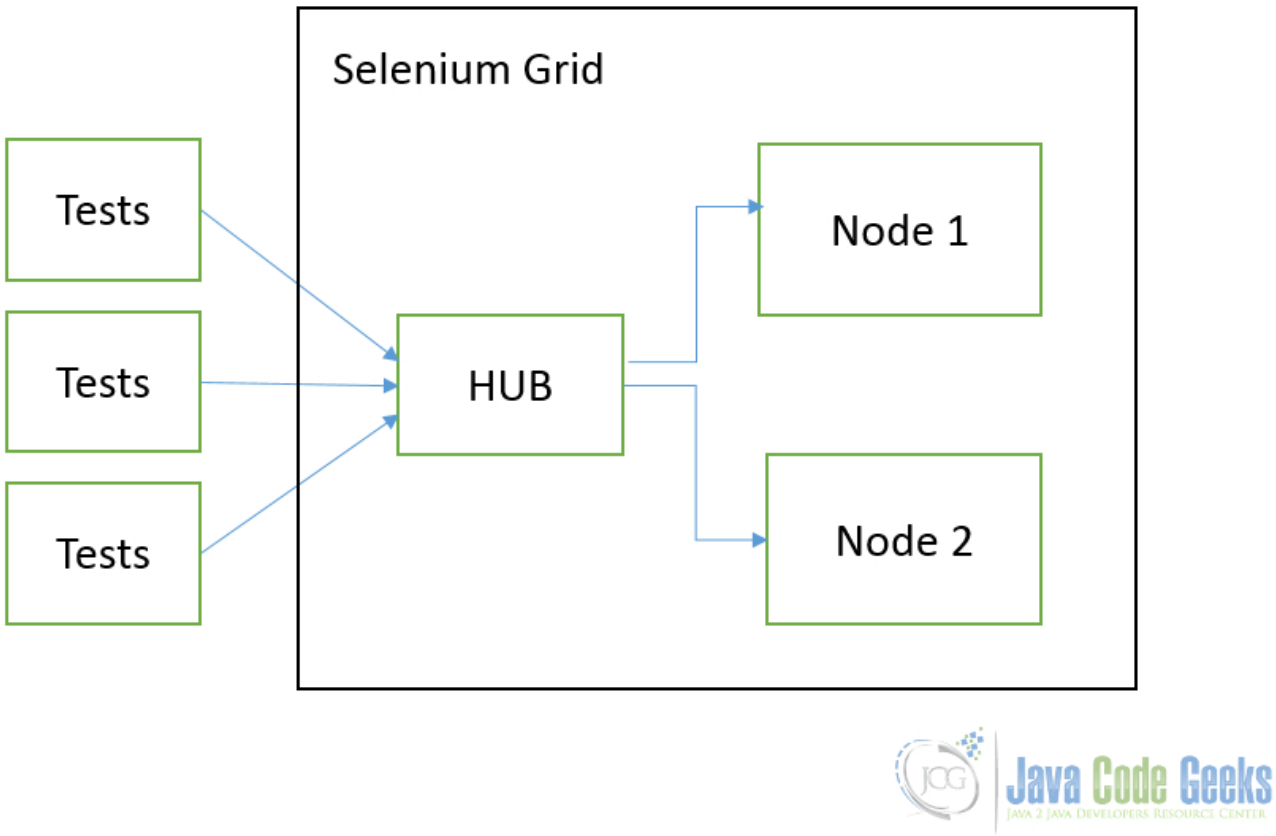


Figure 5.1: Selenium Grid. High level architecture

So all tests and nodes can be located on different machines. Such architecture can be scaled easily.

5.3 Configuration

Before we can start to configure the Selenium Grid we should download the necessary library. Selenium Grid consists only from one JAR file. Go to the official site link and download the Selenium Server JAR file - `selenium-server-standalone-2.48.2.jar`. This jar has the good help information. We should run this jar with the key `-h` that to print help information on screen.

```
java -jar selenium-server-standalone-2.48.2.jar -h
```

This help information has all explanation about supported keys. This keys are used for configuration the instance server. First of all we should run this jar with key `-role hub`. It means the this instance will be the hub server. It will be taken all receiving tests and distributed to the specific nodes server. The hub server is run on `4444` port by default.

```
C:\Tools>java -jar selenium-server-standalone-2.48.2.jar -role hub
10:29:14.270 INFO - Launching Selenium Grid hub
2015-11-19 10:29:15.458:INFO::main: Logging initialized @1362ms
10:29:15.479 INFO - Will listen on 4444
10:29:15.563 INFO - Will listen on 4444
2015-11-19 10:29:15.568:INFO:osjs.Server:main: jetty-9.2.z-SNAPSHOT
2015-11-19 10:29:15.631:INFO:osjsh.ContextHandler:main: Started o.s.j.s.ServletC
ontextHandler@13f88ab{/,null,AVAILABLE}
2015-11-19 10:29:15.770:INFO:osjs.ServerConnector:main: Started ServerConnector@
646db9{HTTP/1.1}{0.0.0.0:4444}
```

```
2015-11-19 10:29:15.771:INFO:osjs.Server:main: Started @1675ms
10:29:15.772 INFO - Nodes should register to https://192.168.0.102:4444/grid/regi
ster/
10:29:15.772 INFO - Selenium Grid hub is up and running
```

The secondary step is to run the node instance. It can be done with key `-role node` as shown below. The same time we should point where is located our hub server by key `-hub https://localhost:4444/grid/register`

```
C:\Tools>java -jar selenium-server-standalone-2.48.2.jar -role node  -hub http:/
/localhost:4444/grid/register
10:31:08.635 INFO - Launching a Selenium Grid node
10:31:09.999 INFO - Java: Oracle Corporation 25.45-b02
10:31:10.000 INFO - OS: Windows 7 6.1 x86
10:31:10.009 INFO - v2.48.0, with Core v2.48.0. Built from revision 41bccdd
10:31:10.089 INFO - Driver class not found: com.opera.core.systems.OperaDriver
10:31:10.090 INFO - Driver provider com.opera.core.systems.OperaDriver is not re
gistered
10:31:10.153 INFO - Selenium Grid node is up and ready to register to the hub
10:31:10.215 INFO - Starting auto registration thread. Will try to register ever
y 5000 ms.
10:31:10.216 INFO - Registering the node to the hub: https://localhost:4444/grid/
register
10:31:10.254 INFO - The node is registered to the hub and ready to use
```

How you can see above the configuration process can be done by adding the keys in command line. But Selenium Server supports another variant of configuration by file configuration in JSON format.

Firstly we should create the file which has the name - `firefox_node.json`. It can have any appropriate name.

```
{
  "capabilities":
      [
        {
          "browserName": "*firefox",
          "maxInstances": 1,
          "seleniumProtocol": "WebDriver"
        }
      ],
  "configuration":
  {
    "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
    "maxSession": 5,
    "port": 6543,
    "host": 127.0.0.1,
    "register": true,
    "registerCycle": 5000,
    "hubPort": 4444,
    "hubHost": 127.0.0.1
  }
}
```

We pointed there that all tests should be run in firefox. Now we can run the new node instance with this configurations. We use `-nodeConfig` that to point which config file to use.

```
C:\Tools>java -jar selenium-server-standalone-2.48.2.jar -role node -nodeConfig
firefox_node.json
11:36:22.804 INFO - Launching a Selenium Grid node
11:36:23.789 INFO - Java: Oracle Corporation 25.45-b02
11:36:23.789 INFO - OS: Windows 7 6.1 x86
11:36:23.798 INFO - v2.48.0, with Core v2.48.0. Built from revision 41bccdd
11:36:23.884 INFO - Driver class not found: com.opera.core.systems.OperaDriver
11:36:23.885 INFO - Driver provider com.opera.core.systems.OperaDriver is not re
```

```
gistered
11:36:23.973 INFO - Selenium Grid node is up and ready to register to the hub
11:36:24.028 INFO - Starting auto registration thread. Will try to register ever
y 5000 ms.
11:36:24.029 INFO - Registering the node to the hub: https://127.0.0.1:4444/grid/
register
11:36:24.041 INFO - The node is registered to the hub and ready to use
```

Sometimes you can need only one instance server. For this reason you should run the selenium server without keys. All tests will be executed in this instance in this case.

Another good opportunities are configure special browsers. For example, below we set the chrome browser environment.

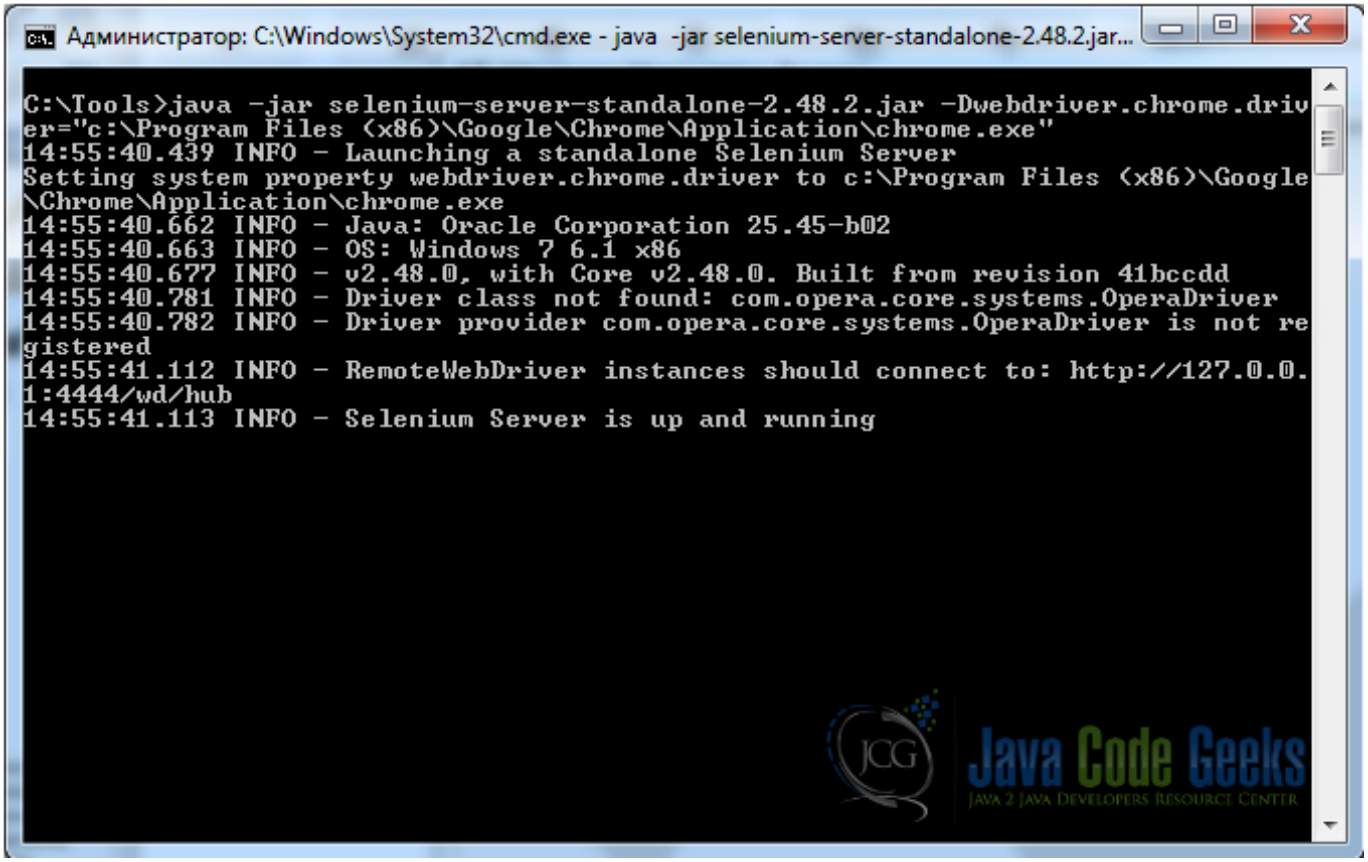


Figure 5.2: Single Selenium Instance. Chrome Config.

We set the properties : `-Dwebdriver.chrome.driver="c:Program Files (x86)GoogleChromeApplicationchrome.exe"`

Right now we have the three instances: one the hub and two nodes and one single server. Let's start to run our tests.

5.4 Run tests

First of all we should create the new maven project.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="https://maven.apache.org/POM/4.0.0"
```

```
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="https://maven.apache.org/POM/4.0.0 https://maven.apache.org/ ↩
             xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>ru</groupId>
        <artifactId>parsentev</artifactId>
        <version>1.0-SNAPSHOT</version>

        <dependencies>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.12</version>
                </dependency>
                <dependency>
                        <groupId>org.seleniumhq.selenium</groupId>
                        <artifactId>selenium-java</artifactId>
                        <version>2.48.2</version>
                </dependency>
        </dependencies>
</project>
```

That we need to add the simple tests.

ruparsentevSeleniumStantaloneServerTest.java

```
package ru.parsentev;

import com.thoughtworks.selenium.DefaultSelenium;
import com.thoughtworks.selenium.Selenium;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

import java.net.MalformedURLException;
import java.net.URL;

import static org.hamcrest.core.Is.is;
import static org.junit.Assert.assertThat;

/**
 * Tests for selenium standalone server.
 * @author parsentev
 * @since 19.11.2015
 */
public class SeleniumStandaloneServerTest {

        @Test
        public void executeFirefoxDriver() throws MalformedURLException {
                this.execute(DesiredCapabilities.firefox());
        }

        @Test
        public void executeChrome() throws MalformedURLException {
                this.execute(DesiredCapabilities.chrome());
        }

        private void execute(final DesiredCapabilities capability) throws  ↩
            MalformedURLException {
```

```
                WebDriver driver = new RemoteWebDriver(
                                new URL("https://localhost:4444/wd/hub"), capability
                );
                driver.get("https://www.javacodegeeks.com/");
                WebElement element = driver.findElement(By.name("s"));
                element.sendKeys("selenuim");
                element.submit();
                assertThat(
                                driver.getTitle(),
                                is("You searched for selenuim | Java Code Geeks")
                );
                driver.quit();
        }
}
```

Now we can run the our test.

```
mvn clean test
```

We can see details information about tests process on the node logs or screens. You should see somethings similar:

```
12:14:25.891 INFO - Executing: [new session: Capabilities [{browserName=firefox,
 version=, platform=ANY}]])
12:14:25.903 INFO - Creating a new session for Capabilities [{browserName=firefo
x, version=, platform=ANY}]
12:14:30.143 INFO - Done: [new session: Capabilities [{browserName=firefox, vers
ion=, platform=ANY}]]
12:14:30.196 INFO - Executing: [get: https://www.javacodegeeks.com/])
12:14:34.283 INFO - Done: [get: https://www.javacodegeeks.com/]
12:14:34.299 INFO - Executing: [find element: By.name: s])
12:14:34.671 INFO - Done: [find element: By.name: s]
12:14:34.689 INFO - Executing: [send keys: 0 [[FirefoxDriver: firefox on WINDOWS
 (2ca50141-8460-4012-bec4-b291e4042f55)] -> name: s], [selenuim]])
12:14:34.774 INFO - Done: [send keys: 0 [[FirefoxDriver: firefox on WINDOWS (2ca
50141-8460-4012-bec4-b291e4042f55)] -> name: s], [selenuim]]
12:14:34.784 INFO - Executing: [submit: 0 [[FirefoxDriver: firefox on WINDOWS (2
ca50141-8460-4012-bec4-b291e4042f55)] -> name: s]])
12:14:39.270 INFO - Done: [submit: 0 [[FirefoxDriver: firefox on WINDOWS (2ca501
41-8460-4012-bec4-b291e4042f55)] -> name: s]]
12:14:39.281 INFO - Executing: [get title])
12:14:39.311 INFO - Done: [get title]
12:14:39.327 INFO - Executing: [delete session: a459baef-2980-4fcc-8093-4ff4eecb
f03f])
12:14:39.806 INFO - Done: [delete session: a459baef-2980-4fcc-8093-4ff4eecbf03f]
```

When the test is received by node it looks appropriate browser in local machine. Then the node opens this browser and starts to do the tests.

It can look like below:

Figure 5.3: Firefox selenium tests

Figure 5.4: Chrome selenium tests

5.5 Download the Code Project

Download

You can download the full source code of this example here: **Selenium**

Chapter 6

Selenium JUnit Example

6.1 Introduction

In this article, we are going to show how you can write automation tests by Selenium and JUnit.

Selenium is tools for building automation tests. Selenium can be used only for testing web applications. When Selenium executes the test, it injects the JavaScript codes to browser or it uses the native browser API. It does not mean that you should write all codes only on JavaScript. Selenium supports all most popular programming languages : Java, C#, Python, Ruby and so.

JUnit is a unit testing framework for the Java programming language. In this example, we will integrate Selenium to this framework. Actually, Selenium IDE has all functionality to write, build and execute automation tests, but if you want to execute tests independently from Selenium IDE you need to use JUnit or another libraries for automation tests.

6.2 Record test cases

The simplest way to get the base test cases code is to record user activities in Selenium IDE. Firstly, you should install Selenium IDE. Actually, Selenium IDE is the Firefox add-ons. After you install this plugin, you can see the Selenium IDE button in the right top corner in Firefox. This plugin is supported only by Firefox.

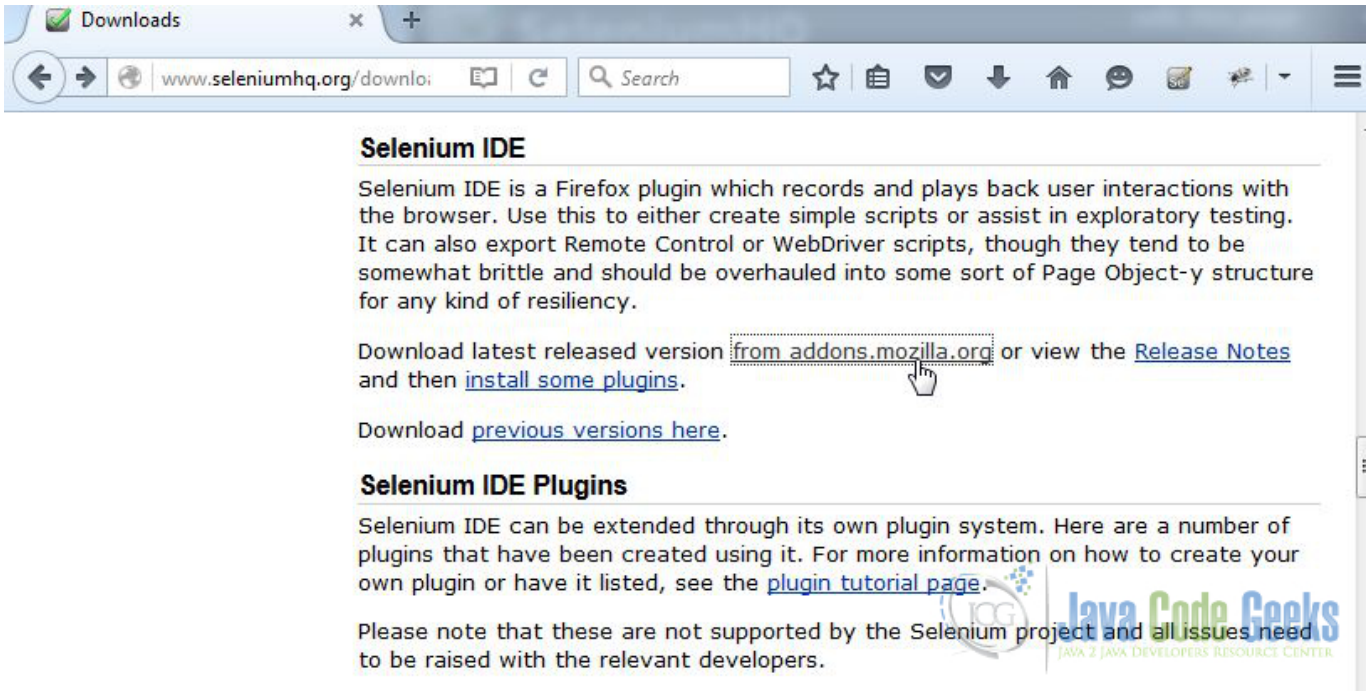


Figure 6.1: Download page for Selenium IDE

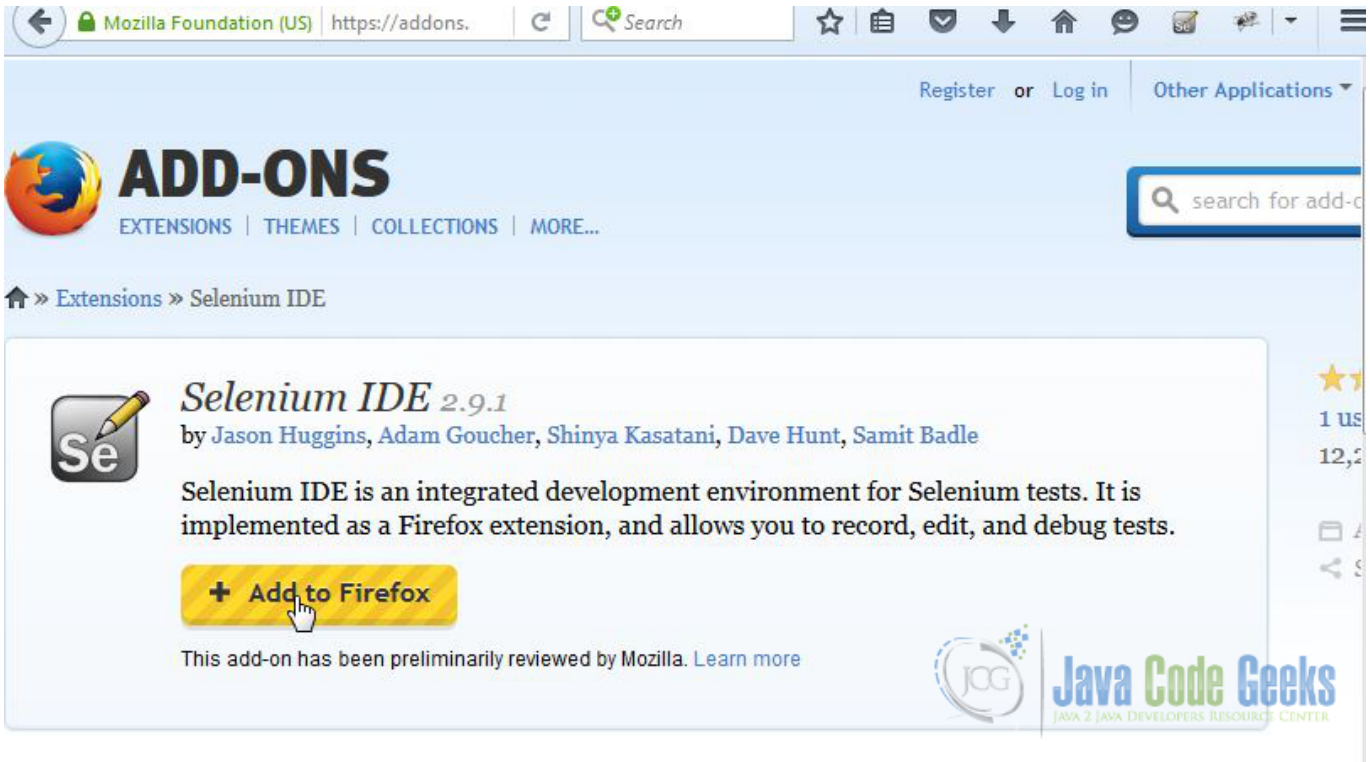


Figure 6.2: Plugins page for Selenium IDE

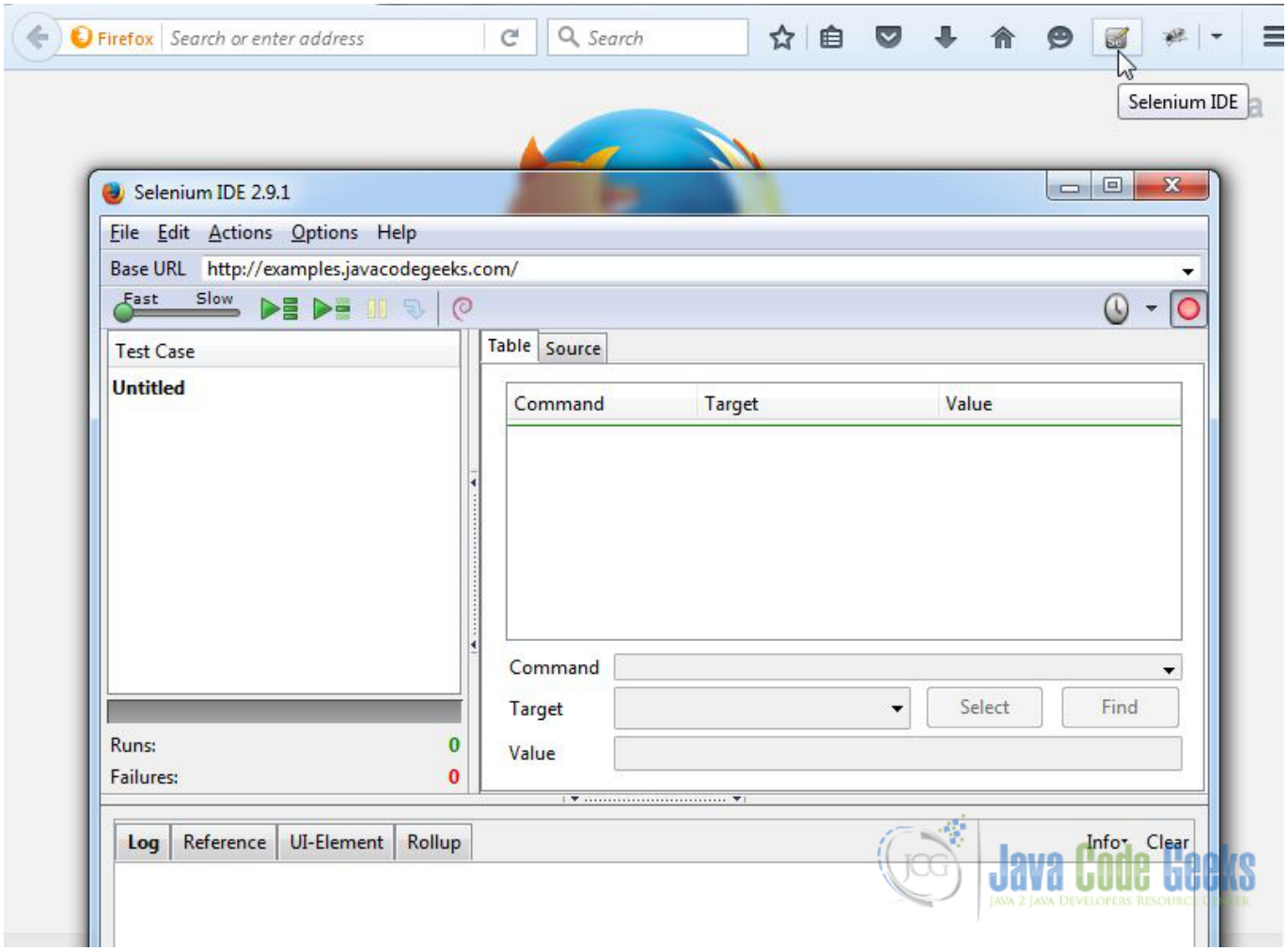


Figure 6.3: Base view of Selenium IDE

Then you need to turn the record button on and start to navigate in the necessary web site. In this case, we want to test the search function in the `https://ebay.com`. EBay has the advanced search functions. It is the good example to show most useful abilities to test web apps by Selenium IDE.

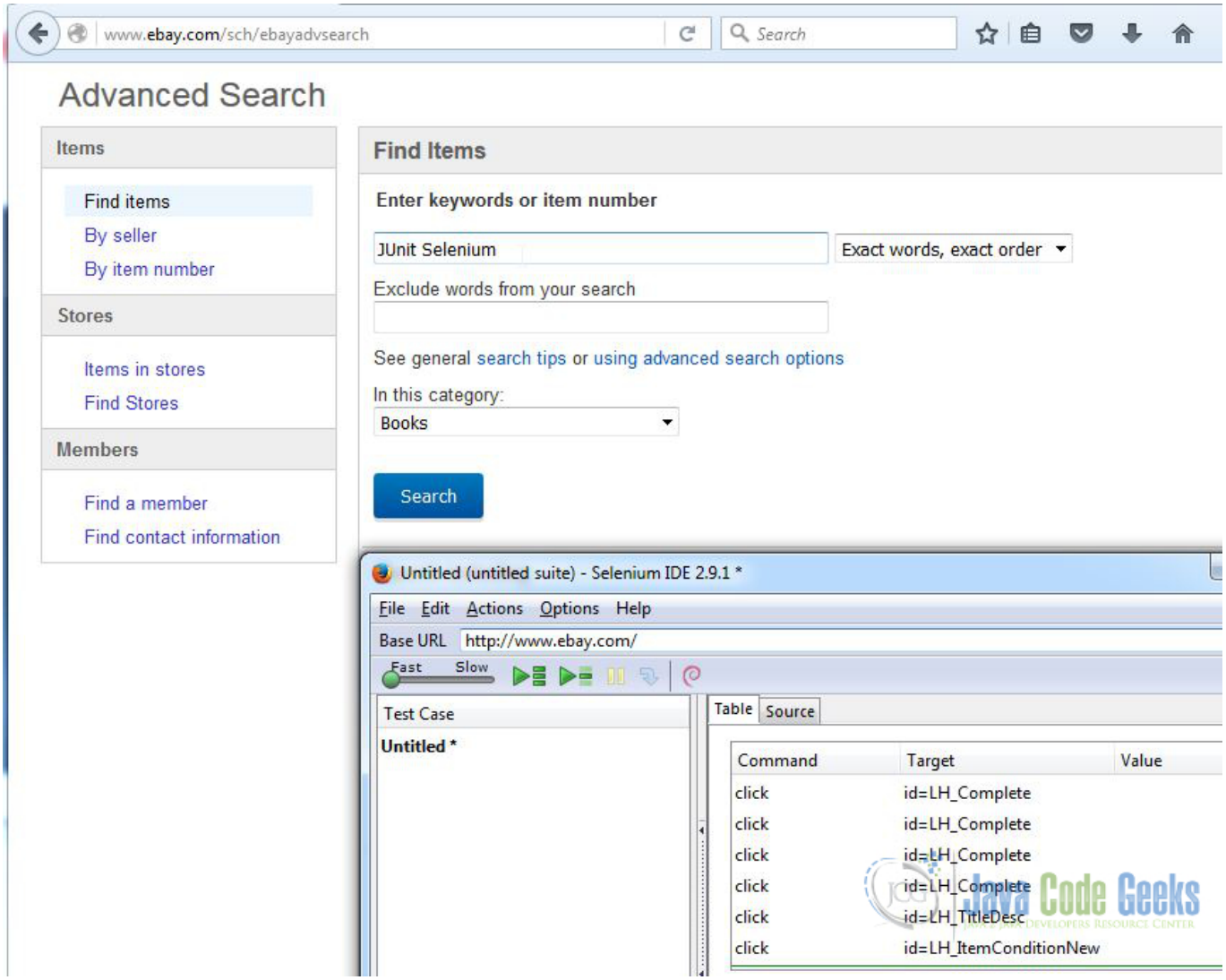


Figure 6.4: Recourd user cases

Now we are ready to export this recorded test cases into your favorite programming . In this example, we use Java. For this reason, we are going to export test cases to Java and ask the Selenium IDE that it generate the necessary structures for JUnit framework too.

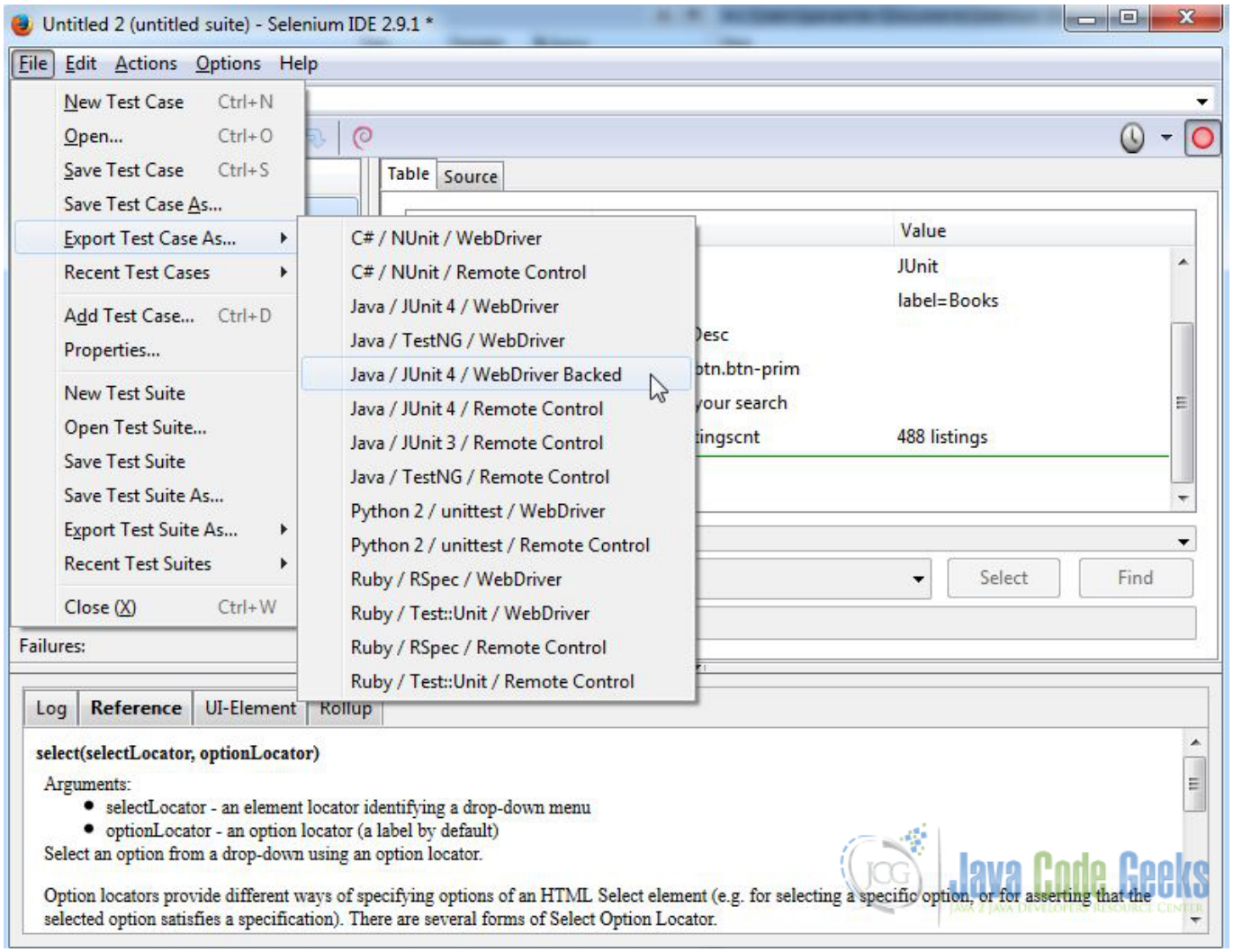


Figure 6.5: Export to Java with JUnit structures

6.3 Integrate to JUnit

The next step is to create the new maven project with JUnit and Selenium dependencies. We will create this new project from default archetype by this follows command

```
mvn -B archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DgroupId=ru. ↩
    parsentev.app -DartifactId=EbayAdvancedSearch
```

```
C:\projects>mvn -B archetype:generate -DarchetypeGroupId=org.apache.maven.archet
ypes -DgroupId=ru.parsentev.app -DartifactId=EbayAdvancedSearch
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:2.3:generate (default-cli) > generate-sources
@ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:2.3:generate (default-cli) < generate-sources
@ standalone-pom <<<
[INFO]
[INFO] --- maven-archetype-plugin:2.3:generate (default-cli) @ standalone-pom --
-
[INFO] Generating project in Batch mode
[INFO] No archetype defined. Using maven-archetype-quickstart (org.apache.maven.
archetypes:maven-archetype-quickstart:1.0)
[INFO] ------------------------------------------------------------------------
---
[INFO] Using following parameters for creating project from Old (1.x) Archetype:
 maven-archetype-quickstart:1.0
[INFO] ------------------------------------------------------------------------
---
[INFO] Parameter: groupId, Value: ru.parsentev.app
[INFO] Parameter: packageName, Value: ru.parsentev.app
[INFO] Parameter: package, Value: ru.parsentev.app
[INFO] Parameter: artifactId, Value: EbayAdvancedSearch
[INFO] Parameter: basedir, Value: C:\projects
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\projects\EbayAdvanced
Search
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5.915 s
[INFO] Finished at: 2016-01-12T11:42:21+03:00
[INFO] Final Memory: 16M/310M
[INFO] ------------------------------------------------------------------------
C:\projects>_
```


Figure 6.6: Build the new project

Now, we can open this project in Eclipse.

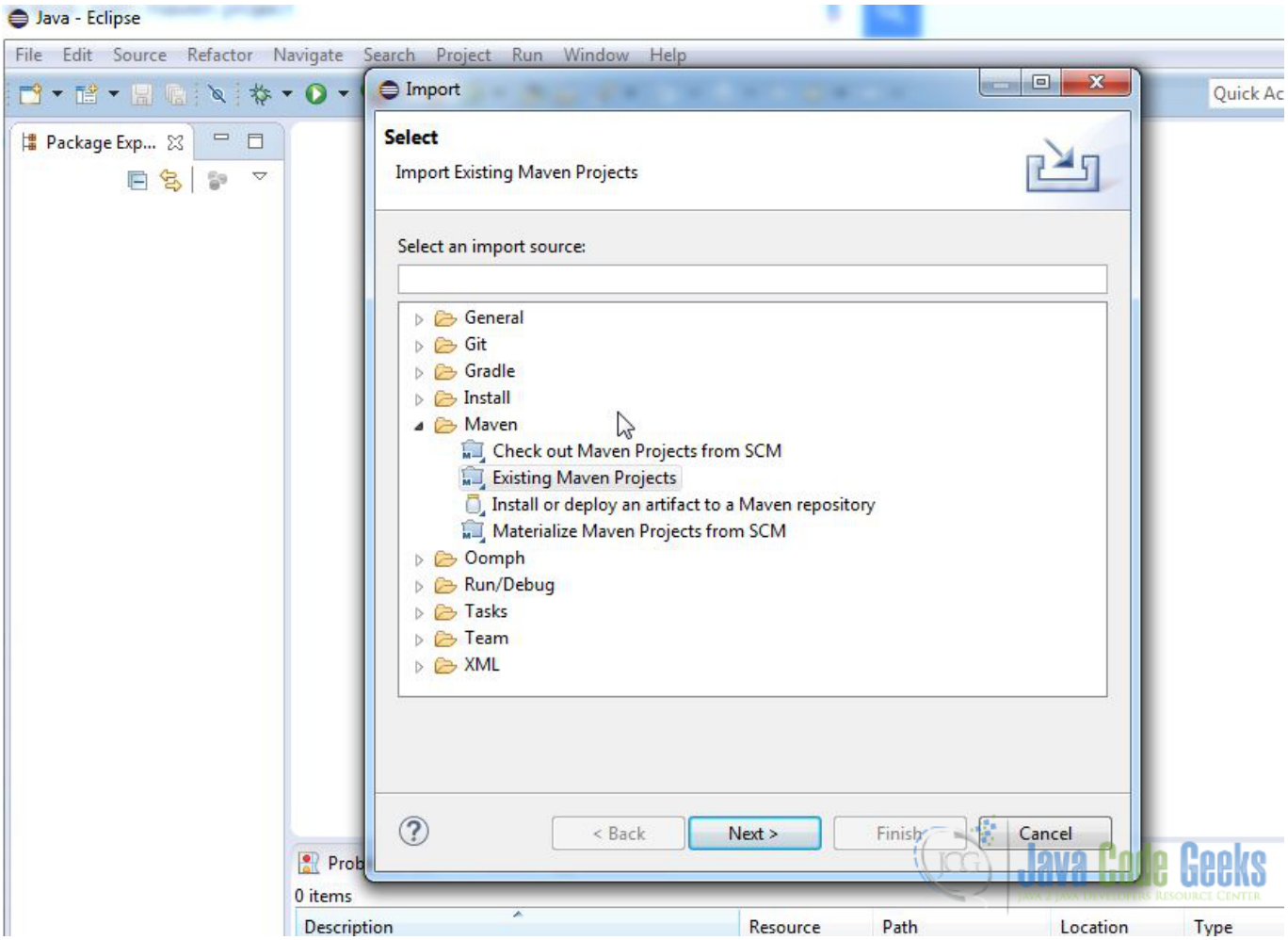


Figure 6.7: Import to Eclipse

Next, we need to move the export Java code from Selenium IDE to the new project. You should put this file to test directory. In my case, it is `srctestjavaruparsentevapp`

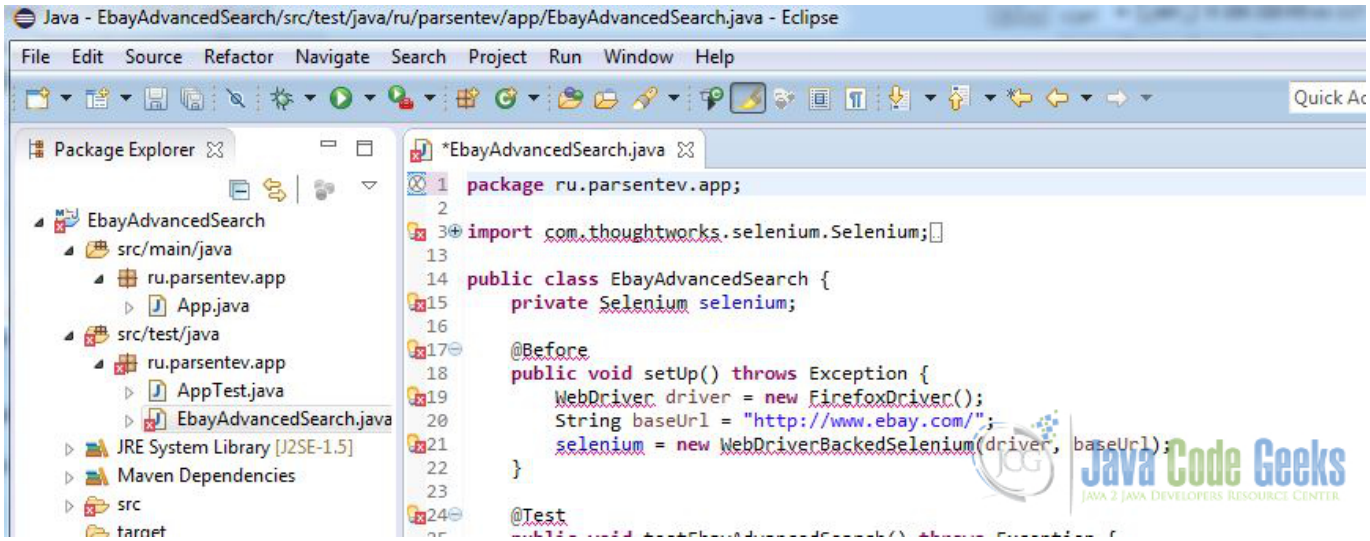


Figure 6.8: Exported Java code

How you can see this code is highlighted by Selenium. it is happened, because we need to add the Selenium library to dependencies block.

pom.xml

```
<project xmlns=&quot;https://maven.apache.org/POM/4.0.0&quot; xmlns:xsi=&quot;https://www. ↩
    w3.org/2001/XMLSchema-instance&quot;
         xsi:schemaLocation=&quot;https://maven.apache.org/POM/4.0.0 https://maven.apache. ↩
             org/maven-v4_0_0.xsd&quot;>
        <modelVersion>4.0.0</modelVersion>
        <groupId>ru.parsentev.app</groupId>
        <artifactId>EbayAdvancedSearch</artifactId>
        <packaging>jar</packaging>
        <version>1.0-SNAPSHOT</version>
        <name>EbayAdvancedSearch</name>
        <url>https://maven.apache.org</url>
        <dependencies>
                <dependency>
                        <groupId>org.seleniumhq.selenium</groupId>
                        <artifactId>selenium-java</artifactId>
                        <version>2.48.2</version>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.12</version>
                        <scope>test</scope>
                </dependency>
        </dependencies>
</project>
```

You can find the full source code of this case below.

EbayAdvancedSearch.java

```
package ru.parsentev.app;

import com.thoughtworks.selenium.Selenium;
import org.openqa.selenium.By;
```

```
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.WebDriver;
import com.thoughtworks.selenium.webdriven.WebDriverBackedSelenium;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;
import java.util.regex.Pattern;
import static org.apache.commons.lang3.StringUtils.join;

public class EbayAdvancedSearch {
        private Selenium selenium;

        @Before
        public void setUp() throws Exception {
                WebDriver driver = new FirefoxDriver();
                String baseUrl = "https://www.ebay.com/";
                selenium = new WebDriverBackedSelenium(driver, baseUrl);
        }

        @Test
        public void testEbayAdvancedSearch() throws Exception {
                selenium.open("/");
                selenium.waitForPageToLoad("30000");
                selenium.click("id=gh-as-a");
                selenium.type("id=_nkw", "JUnit");
                selenium.select("id=e1-1", "value=267");
                selenium.click("id=LH_TitleDesc");
                selenium.click("css=button.btn.btn-prim");
                selenium.waitForPageToLoad("30000");
                for (int second = 0;; second++) {
                        if (second >= 60) fail("timeout");
                        try { if (selenium.isElementPresent("css=span.listingscnt")) break; ↩
                             } catch (Exception e) {}
                        Thread.sleep(1000);
                }

                assertEquals("Classified: 488", selenium.getText("css=span.listingscnt"));
        }

        @After
        public void tearDown() throws Exception {
                selenium.stop();
        }
}
```

Now, we can run this project by this follows command `mvn test`

```
C:\projects\EbayAdvancedSearch>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building EbayAdvancedSearch 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ EbayAdvanc
edSearch ---
[WARNING] Using platform encoding (Cp1251 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\projects\EbayAdvancedSearch\src\ma
in\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ EbayAdvancedSea
rch ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1251, i.e. b
uild is platform dependent!
[INFO] Compiling 1 source file to C:\projects\EbayAdvancedSearch\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ Eb
ayAdvancedSearch ---
[WARNING] Using platform encoding (Cp1251 actually) to copy filtered resources,
i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\projects\EbayAdvancedSearch\src\te
st\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ EbayAdv
ancedSearch ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1251, i.e. b
uild is platform dependent!
[INFO] Compiling 2 source files to C:\projects\EbayAdvancedSearch\target\test-cl
asses
[WARNING] /C:/projects/EbayAdvancedSearch/src/test/java/ru/parsentev/app/EbayAdv
ancedSearch.java: C:\projects\EbayAdvancedSearch\src\test\java\ru\parsentev\app\
EbayAdvancedSearch.java uses or overrides a deprecated API.
[WARNING] /C:/projects/EbayAdvancedSearch/src/test/java/ru/parsentev/app/EbayAdv
ancedSearch.java: Recompile with -Xlint:deprecation for details.
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ EbayAdvancedSearch
 ---
[INFO] Surefire report directory: C:\projects\EbayAdvancedSearch\target\surefire
-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running ru.parsentev.app.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.076 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Figure 6.9: Run tests

6.4 Conclusion

In this article, we have shown how you can integrate Selenium and JUnit frameworks. If you want to get more information about this framework, please vitis official web sites: Selenium and JUnit.

6.5 Download the source code

Download

You can download the full source code of this example here: **Selenium JUnit**

Chapter 7

Selenium Grid Example

7.1 Introduction

In this tutorial, we are going to explain what Selenium Grid is and how you can use it in your project. We are going to install and configure the Selenium Grid, write and execute the tests on it. Selenium Grid is one of tool from Selenium framework. It is the distributed system for execution tests.

It has few benefits:

- You can execute tests in parallel, so it can reduce the execution time.
- You can set particular environments for tests.

Actually. It is the daunting task to reduce the tests executions time, because some of functional tests spent about hour for executions process. Selenium team offers the great solutions for it. It uses the separate nodes, which compound in full distributed system. For this reason, it can be scaled easily.

Below you can see the highly level architecture.

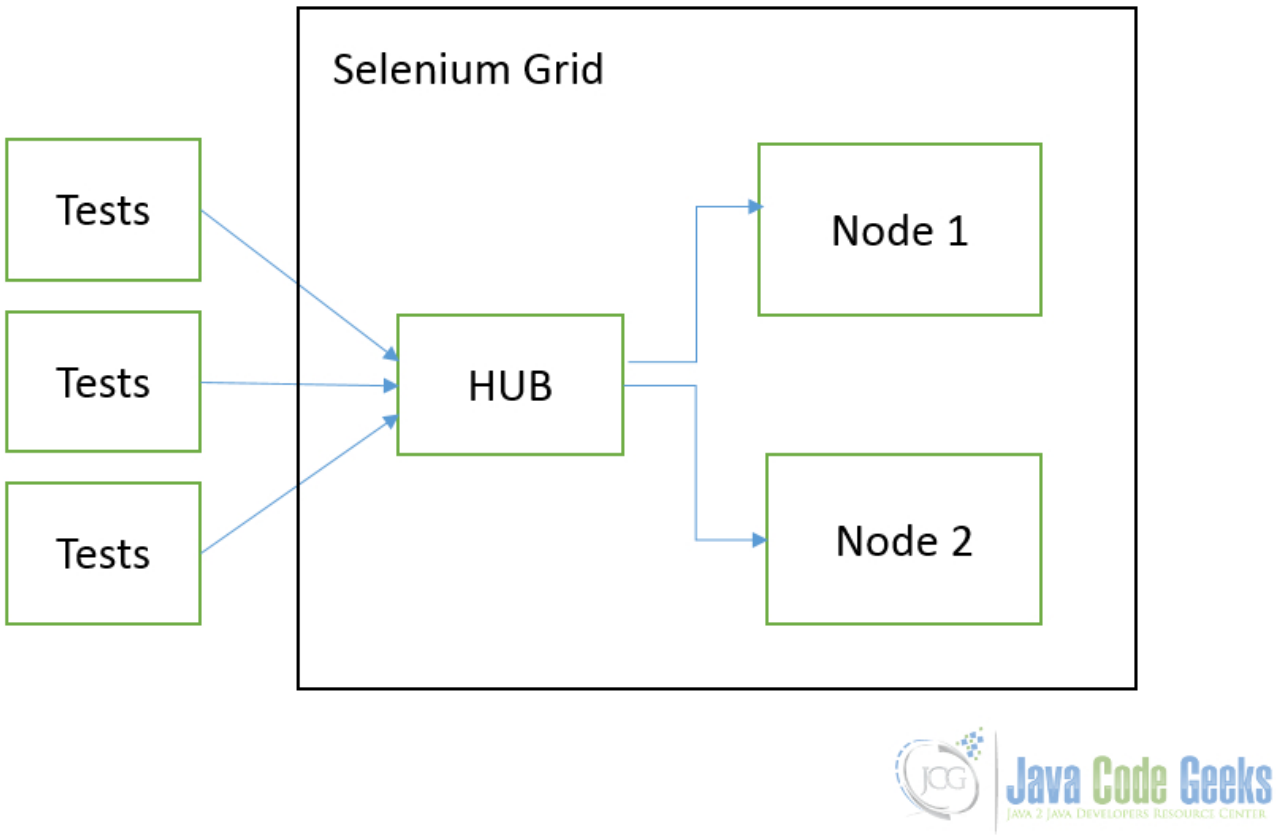


Figure 7.1: Selenium Grid. High level

7.2 Installing Selenium Grid

Firstly, you should download the necessary library. The great thing about Selenium Grid is that you need only one JAR file. You need to go to the official web site `seleniumhq.org/download` and download the Selenium Server. It can be configured to Selenium Grid.

Below you can see the download page:

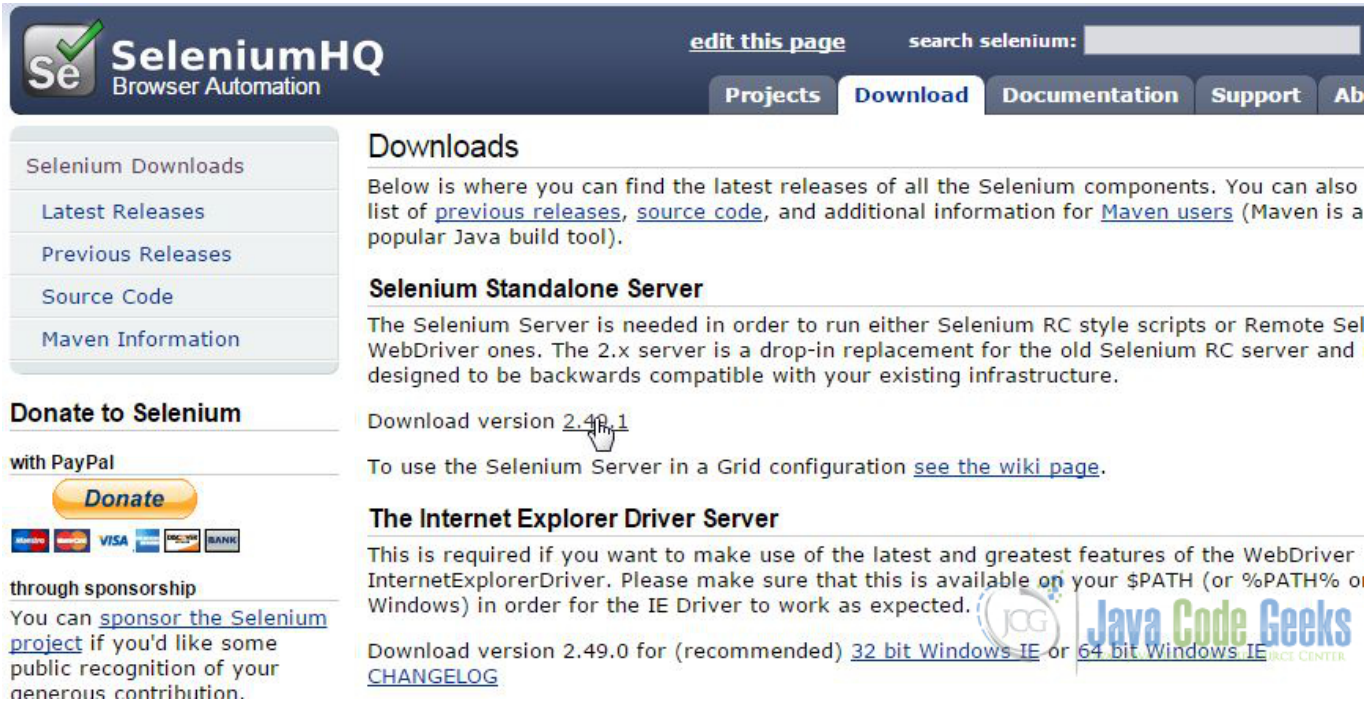


Figure 7.2: Selenium Grid download page

Right now, you can run this jar and use Selenium Grid. You can find the details about it in the next section.

7.3 Usage cases

How we said before, all you need is the Selenium Server JAR. You can run it this key `--help`, that get help information about supported operations.

```
c:\Users\parsentev\Downloads>java -jar selenium-server-standalone-2.49.1.jar -help

Running as a standalone server:

Usage: java -jar selenium-server.jar [-interactive] [options]

  -port : the port number the selenium server should use
    (default 4444)
  -timeout : an integer number of seconds we should allow a
    client to be idle
  -browserTimeout : an integer number of seconds a browser is
    allowed to hang
  -interactive: puts you into interactive mode.  See the tutorial for
    more details
  -singleWindow: puts you into a mode where the test web site
    executes in a frame. This mode should only be selected if the
    application under test does not use frames.
  -profilesLocation: Specifies the directory that holds the profiles
    that java clients can use to start up selenium.  Currently
    supported for Firefox only.
  -forcedBrowserMode : sets the browser mode to a single
    argument (e.g. "*iexplore") for all sessions, no matter what is
    passed to getNewBrowserSession
  -forcedBrowserModeRestOfLine : sets the browser mode to
```

```
    all the remaining tokens on the line (e.g. "*custom
    /some/random/place/iexplore.exe") for all sessions, no matter what
    is passed to getNewBrowserSession
  -userExtensions : indicates a JavaScript file that will be
    loaded into selenium
  -browserSessionReuse: stops re-initialization and spawning of the
    browser between tests
  -avoidProxy: By default, we proxy every browser request; set this
    flag to make the browser use our proxy only for URLs containing
    '/selenium-server'
  -firefoxProfileTemplate : normally, we generate a fresh empty
    Firefox profile every time we launch.  You can specify a directory
    to make us copy your profile directory instead.
  -debug: puts you into debug mode, with more trace information and
    diagnostics on the console
  -browserSideLog: enables logging on the browser side; logging
    messages will be transmitted to the server.  This can affect
    performance.
  -ensureCleanSession: If the browser does not have user profiles,
    make sure every new session has no artifacts from previous
    sessions.  For example, enabling this option will cause all user
    cookies to be archived before launching IE, and restored after IE
    is closed.
  -trustAllSSLCertificates: Forces the Selenium proxy to trust all
    SSL certificates.  This doesn't work in browsers that don't use the
    Selenium proxy.
  -log : writes lots of debug information out to a log
    file and disables logging to console
  -logLongForm: writes information out to console in long format (for
    debugging purpose)
  -htmlSuite    : Run a
    single HTML Selenese (Selenium Core) suite and then exit
    immediately, using the specified browser (e.g. "*firefox") on the
    specified URL (e.g. "https://www.google.com").  You need to specify
    the absolute path to the HTML test suite as well as the path to the
    HTML results file we'll generate.
  -proxyInjectionMode: puts you into proxy injection mode, a mode
    where the selenium server acts as a proxy server for all content
    going to the test application.  Under this mode, multiple domains
    can be visited, and the following additional flags are supported:

    -dontInjectRegex : an optional regular expression that
      proxy injection mode can use to know when to bypss injection
    -userJsInjection : specifies a JavaScript file which will
      then be injected into all pages
    -userContentTransformation  : a regular
      expression which is matched against all test HTML content; the
      second is a string which will replace matches.  These flags can
      be used any number of times.  A simple example of how this could
      be useful: if you add "-userContentTransformation https http"
      then all "https" strings in the HTML of the test application will
      be changed to be "http".

This synopsis lists options available in standalone role only. To
  get help on the options available for other roles run the server with
  -help option and the corresponding -role option value.
```

How you can see in the architecture diagram, firstly, we need to run the HUB nodes. It will takes all receiving tests and route to participated nodes, which has appropriate environment.

You need to run the JAR with follow key `-role hub`

Figure 7.3: Selenium Grid HUB

Now, we need to add the node. Node is used for executing the particular test on special environments. Therefore it means that you can run the node on separate machine. You can do by following command `java -jar selenium-server-standalone-2.49.1.jar -role node -hub https://localhost:4444/grid/register`

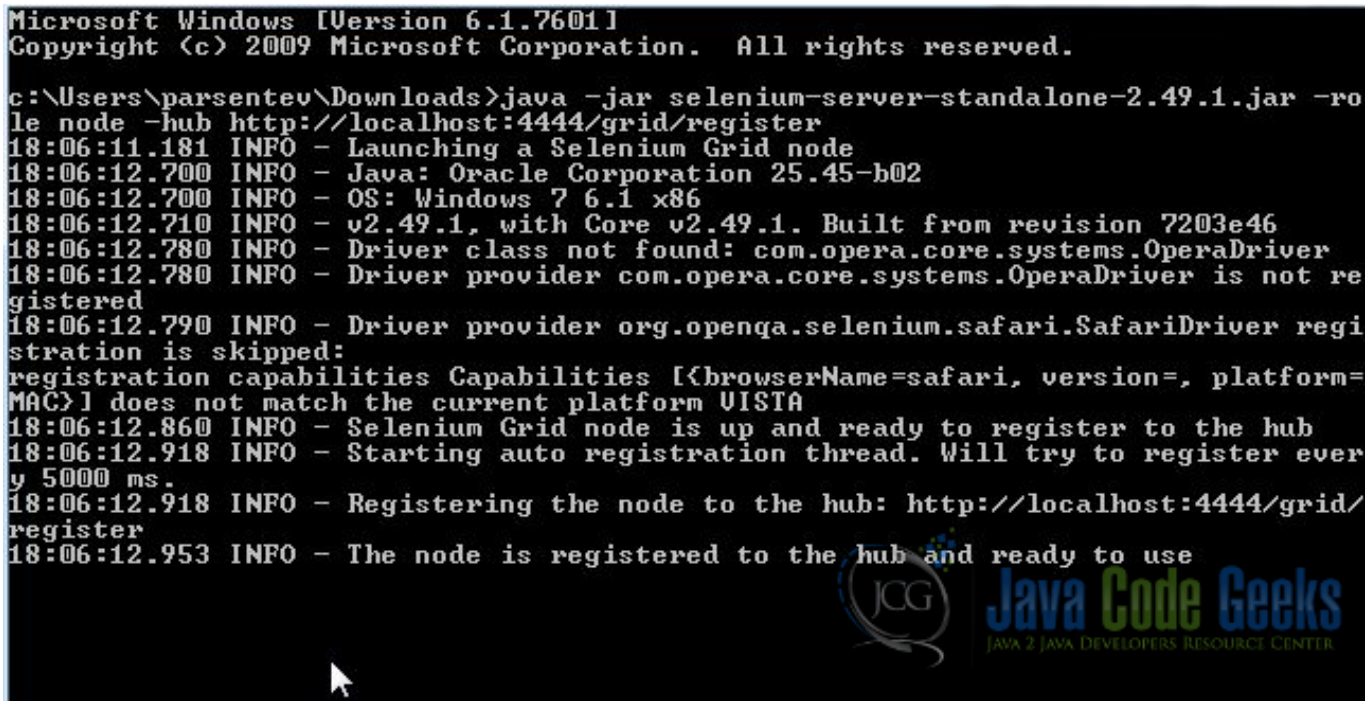


Figure 7.4: Selenium GRID Node

Selenium Grid is run and we can use it.

Let's create the simple maven project in order to demonstrate how it works.

pom.xml

```
<?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?>
<project xmlns=&quot;https://maven.apache.org/POM/4.0.0&quot;
         xmlns:xsi=&quot;https://www.w3.org/2001/XMLSchema-instance&quot;
         xsi:schemaLocation=&quot;https://maven.apache.org/POM/4.0.0 https://maven.apache. ↩
             org/xsd/maven-4.0.0.xsd&quot;>
        <modelVersion>4.0.0</modelVersion>

        <groupId>ru</groupId>
        <artifactId>parsentev</artifactId>
        <version>1.0-SNAPSHOT</version>

        <dependencies>
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.12</version>
                </dependency>
                <dependency>
                        <groupId>org.seleniumhq.selenium</groupId>
                        <artifactId>selenium-java</artifactId>
                        <version>2.48.2</version>
                </dependency>
        </dependencies>
</project>
```

How you can see, we need to add the Selenium library in dependencies. Then, let's create the simple test cases, when we want to test the search function in `https://www.javacodegeeks.com/`

ruparsentevSeleniumStantaloneServerTest.java

```
package ru.parsentev;

import com.thoughtworks.selenium.DefaultSelenium;
import com.thoughtworks.selenium.Selenium;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

import java.net.MalformedURLException;
import java.net.URL;

import static org.hamcrest.core.Is.is;
import static org.junit.Assert.assertThat;

/**
 * Tests for selenium standalone server.
 * @author parsentev
 * @since 19.11.2015
 */
public class SeleniumStandaloneServerTest {

        @Test
        public void executeFirefoxDriver() throws MalformedURLException {
                this.execute(DesiredCapabilities.firefox());
        }

        @Test
        public void executeChrome() throws MalformedURLException {
                this.execute(DesiredCapabilities.chrome());
```

```
        }

        private void execute(final DesiredCapabilities capability) throws  ↩
            MalformedURLException {
                WebDriver driver = new RemoteWebDriver(
                                new URL("https://localhost:4444/wd/hub"), capability
                );
                driver.get("https://www.javacodegeeks.com/");
                WebElement element = driver.findElement(By.name("s"));
                element.sendKeys("selenuim");
                element.submit();
                assertThat(
                                driver.getTitle(),
                                is("You searched for selenuim | Java Code Geeks")
                );
                driver.quit();
        }
}
```

In example above, we pointed that we want to execute the test on Chrome and Firefox browsers by following commands: `DesiredCapabilities.chrome()`.

You can set more specific requirments for execution environments by additional API, which offer the Selenium library:

```
capability.setBrowserName();
capability.setPlatform();
capability.setVersion()
capability.setCapability(,);
```

The same time, you need to configure your nodes for particular environment. Selenium Grid supports two way, how you can do it.

- It is used command-line key. For example, we want that this node execute only tests in Internet Explorer 9. We can configure in like: `-browser browserName=iexplorer,version=9,platform=WINDOWS`
- It is used JSON configuration file.

```
{
  "capabilities":
      [
        {
          "browserName": "*firefox",
          "maxInstances": 1,
          "seleniumProtocol": "WebDriver"
        }
      ],
  "configuration":
  {
    "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
    "maxSession": 5,
    "port": 6543,
    "host": 127.0.0.1,
    "register": true,
    "registerCycle": 5000,
    "hubPort": 4444,
    "hubHost": 127.0.0.1
  }
}
```

7.4 Conclusion

In this article, we explained what Selenium Grid is and shown how to configure, run, execute tests. We could not cover all narrow things about Selenium Grid, so if you want to improve your knowledge about Selenium and particular about Selenium Grid, please, visit the official web site seleniumhq.org

7.5 Download the Maven project

Download

You can download the full source code of this example here: **SeleniumGrid**