1.16 Listeners - Context, iTest, iTestResults

This section will guide you to:

* Use Listeners in Selenium

**Development Environment:**

* Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
* JavaDevelopment Kit Version 8
* Selenium Standalone Server Version 3.141.59

This guide has only three sub-sections, namely:

1.16.1 Using iTestListener

1.16.2 Running the code

1.16.3 Pushing the code to your GitHub repositories

**Step 1.16.1:** Using iTestListener:

* Listener is an interface that modifies the TestNG behavior.
* Listener listens to the event defined in the Selenium and behaves accordingly.
* It is used in Selenium by implementing Listeners Interface.
* iTestListener is used in Selenium to generate logs or customize the TestNG reports.
* iTestListener has the following methods:

1. **onStart:** onStart method is called when any Test starts.
2. **onTestSuccess:** onTestSuccess method is called on the success of any Test.
3. **onTestFailure:** onTestFailure method is called on the failure of any test.
4. **onTestSkipped:** onTestSkipped method is called when any test gets skipped.
5. **onTestFailedButWithinSuccessPercentage:** onTestFailedButWithinSuccessPercentage method is called each time Test fails but within the success percentage.
6. **onFinish:** onFinish method is called after all the tests are executed.

* Open Eclipse and create a Project.
* Create a Listener class that will implement iTestListener.
* The code in Eclipse will look like:

**package** test.testing;

**import** org.testng.ITestContext;

**import** org.testng.ITestListener;

**import** org.testng.ITestResult;

**public** **class** ListenersTest **implements** ITestListener {

**public** void onFinish(ITestContext **Result**) {

**System**.out.println(**Result**.getName()+"case finished");

}

**public** void onStart(ITestContext **Result**) {

// TODO Auto-generated method stub

}

**public** void onTestFailedButWithinSuccessPercentage(ITestResult **Result**) {

// TODO Auto-generated method stub

}

**public** void onTestFailure(ITestResult **Result**) {

// TODO Auto-generated method stub

**System**.out.println("The name of the testcase failed is :"+**Result**.getName());

}

**public** void onTestSkipped(ITestResult **Result**) {

// TODO Auto-generated method stub

**System**.out.println("The name of the testcase Skipped is :"+**Result**.getName());

}

**public** void onTestStart(ITestResult **Result**) {

// TODO Auto-generated method stub

**System**.out.println(**Result**.getName()+" test case started");

}

**public** void onTestSuccess(ITestResult **Result**) {

// TODO Auto-generated method stub

**System**.out.println("The name of the testcase passed is :"+**Result**.getName());

}

}

* Create a Java class and implement a Listener class in this.
* The code in Eclipse will look like:

**package** test.testing;

**import** org.openqa.selenium.By;

**import** org.testng.Assert;

**import** org.testng.annotations.Listeners;

**import** org.testng.annotations.Test;

**@Listeners(test.testing.ListenersTest.class)**

**public** **class** Home **extends** Baseclass {

**@Test**

**public** void clickOnCategory()

{

driver.findElement(By.xpath("//div[@id='navigation']/a[1]")).click();

**System**.out.println("Clicked on links");

}

}

**Step 1.16.2:** Running the code

* Run the code through Eclipse.

**Step 1.16.3:** Pushing the code to your GitHub repositories

Open your command prompt and navigate to the folder where you have created your files.

cd <folder path>

Initialize your repository using the following command:

git init

Add all the files to your git repository using the following command:

git add . 

Commit the changes using the following command:

git commit . -m “Changes have been committed.”

Push the files to the folder you initially created using the following command:

git push -u origin master