# Introduction to JUnit Framework and Its Usage in Selenium Script

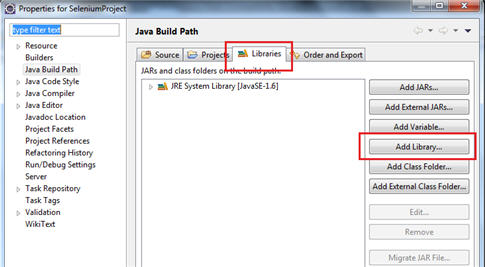
Basically JUnit is an open source unit testing tool and used to test small/large units of code. To run the JUnit test you don’t have to create class object or define main method. [JUnit](http://junit.org/" \o "JUnit) provide assertion library which is used to evaluate the test result. Annotations of JUnit are used to run the test method. JUnit is also used to run the Automation suite having multiple test cases.

**Adding JUnit library in Java project**

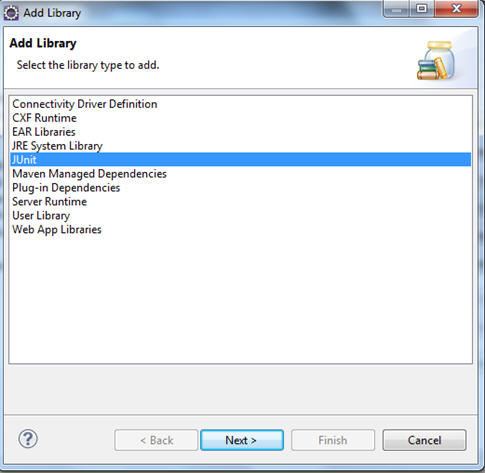
First we will learn how to add JUnit library in your Java project:

**Step #1:** Right click on Java project->Build Path->Configure Build path

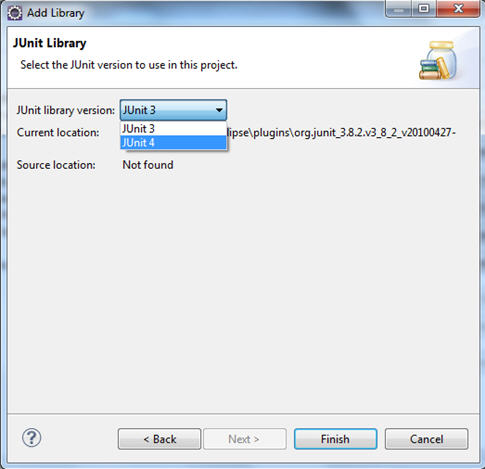
**Step #2:** Click Libraries->Add Library

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Junit-framework-Selenium-script.jpg)

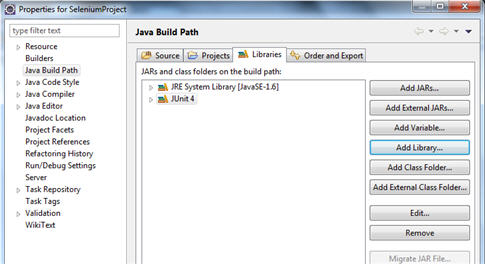
**Step #3:** Click on Junit.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Junit-framework-Selenium-script-2.jpg)

**Step #4:** Select Junit4->Finish

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Junit-framework-Selenium-script-3.jpg)

**Step #5:** Click OK.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Junit-framework-Selenium-script-4.jpg)

There are many frameworks like Data Driven Framework, Keyword Driven Framework, and Hybrid Framework which use Junit tool as test runner and which will help to start the batch execution and reporting.

**JUnit Annotations Used in Selenium scripts**

There are many annotations available in Junit. Here we have described few annotations which are used very frequently in Selenium scripts and framework.

**#1. @Test**

**@Test** annotation is used to run a Junit test.

**Example**:

|  |  |
| --- | --- |
| 1 | @Test |
| 2 | public void junitTest() | |

|  |  |
| --- | --- |
| 3 | { |
| 4 | System.out.println("Running Junit test"); | |

|  |  |  |
| --- | --- | --- |
| 5 | Assert.assertEquals(1,1); | |
| 6 | } |

**How to Run a Junit test:**

Navigate to run ->Run as Junit test

**#2. @Before:**

**@Before** annotation is used to run any specific test before each test.

|  |  |  |
| --- | --- | --- |
| 1 | public class Junttest { | |
| 2 | @Before |

|  |  |
| --- | --- |
| 3 | public void beforeTest(){ |
| 4 | System.out.println("Running before test"); | |

|  |  |  |
| --- | --- | --- |
| 5 | } | |
| 6 |  |

|  |  |
| --- | --- |
| 7 | @Test |
| 8 | public void junitTest(){ | |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | System.out.println("Running Junit test"); | | |
| 10 | | } |

|  |  |
| --- | --- |
| 11 | } |

**Output:**  
Running before test  
Running Junit test

Example of before annotation using two junit test method.

|  |  |  |
| --- | --- | --- |
| 1 | public class Junttest { | |
| 2 | @Before |

|  |  |
| --- | --- |
| 3 | public void beforeTest(){ |
| 4 | System.out.println("Running before test"); | |

|  |  |  |
| --- | --- | --- |
| 5 | } | |
| 6 |  |

|  |  |
| --- | --- |
| 7 | @Test |
| 8 | public void junitTest(){ | |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | System.out.println("Running Junit test"); | | |
| 10 | | } |

|  |  |
| --- | --- |
| 11 |  |
| 12 | @Test | |

|  |  |
| --- | --- |
| 13 | public void secondJunitTest(){ |
| 14 | System.out.println("Running second Junit test"); | |

|  |  |
| --- | --- |
| 15 | } |
| 16 | } |

**Output:**  
Running before test  
Running Junit test  
Running before test  
Running second Junit test

Before running junitTest method beforeTest method will run. Similarly before running secondJuntiTest again beforeTest method will run and produces output like above.

**#3. @BeforeClass**

This method executes once before running all test. The method has to be a static method. Initialization of properties files, databases etc are done in beforeClass method.

------------

|  |  |  |
| --- | --- | --- |
| 1 | public class Junttest { | |
| 2 | @BeforeClass |

|  |  |
| --- | --- |
| 3 | public static void beforeClassTest(){ |
| 4 | System.out.println("Executed before class method"); | |

|  |  |  |
| --- | --- | --- |
| 5 | } | |
| 6 |  |

|  |  |
| --- | --- |
| 7 | @Test |
| 8 | public void junitTest(){ | |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | System.out.println("Running Junit test"); | | |
| 10 | | } |

|  |  |
| --- | --- |
| 11 |  |
| 12 | @Test | |

|  |  |
| --- | --- |
| 13 | public void secondJunitTest(){ |
| 14 | System.out.println("Running second Junit test"); | |

|  |  |
| --- | --- |
| 15 | } |
| 16 | } |

**Output:**  
Executed before class method  
Running Junit test  
Running second Junit test

**#4. @After**

This method executes after each test.

|  |  |  |
| --- | --- | --- |
| 1 | public class Junttest { | |
| 2 | @Test |

|  |  |
| --- | --- |
| 3 | public void junitTest(){ |
| 4 | System.out.println("Running Junit test"); | |

|  |  |  |
| --- | --- | --- |
| 5 | } | |
| 6 |  |

|  |  |
| --- | --- |
| 7 | @After |
| 8 | public void afterTest(){ | |

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | System.out.println("Running after method"); | | |
| 10 | | } |

|  |  |
| --- | --- |
| 11 | } |

**Output:**  
Running Junit test  
Running after method

**#5. @AfterClass**

Like @BeforeClass, @AfterClass executes once after executing all test methods. Like @BeforeClass method, @AfterClass method has to be a static method.

|  |  |  |
| --- | --- | --- |
| 1 | public class Junttest { | |
| 2 |  |

|  |  |
| --- | --- |
| 3 | @Test |
| 4 | public void junitTest(){ | |

|  |  |  |
| --- | --- | --- |
| 5 | System.out.println("Running Junit test"); | |
| 6 | } |

|  |  |
| --- | --- |
| 7 |  |
| 8 | @Test | |

|  |  |  |
| --- | --- | --- |
| 9 | public void secondJunitTest(){ | |
| 10 | | System.out.println("Running second Junit test"); | |

|  |  |  |
| --- | --- | --- |
| 11 | } | |
| 12 |  |

|  |  |
| --- | --- |
| 13 | @AfterClass |
| 14 | Public static void afterClassTest(){ | |

|  |  |  |
| --- | --- | --- |
| 15 | System.out.println("Running afterclass method"); | |
| 16 | } |

|  |  |
| --- | --- |
| 17 | } |

**Output:**  
Running Junit test  
Running second Junit test  
Running afterclass method

Junit assertions are used to validate certain condition and stops execution of program if the conditions are not satisfied.

**#6. Parameterized Junit class:**

Parameterized class is used to run same scenario with multiple dataset.

Below is the example to pass multiple parameters in a Junit test.

@Parameters annotation tag is used to pass multiple data. Here, we have taken 2\*2 dimensional array and the data can be visualized like below:

[Junit framework Selenium script 5](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Junit-framework-Selenium-script-5.jpg)

|  |  |  |
| --- | --- | --- |
| 1 | @RunWith(Parameterized.class) | |
| 2 | public class Junttest { |

|  |  |  |
| --- | --- | --- |
| 3 | public String name; | |
| 4 | public int age; |

|  |  |  |
| --- | --- | --- |
| 5 | public Junttest(String name,int age){ | |
| 6 | this.name=name; |

|  |  |  |
| --- | --- | --- |
| 7 | this.age=age; | |
| 8 | } |

|  |  |  |
| --- | --- | --- |
| 9 |  | |
| 10 | | @Test | |

|  |  |
| --- | --- |
| 11 | public void testMethod(){ |
| 12 | System.out.println("Name is: "+name +" and age is: "+age); | |

|  |  |  |
| --- | --- | --- |
| 13 | } | |
| 14 |  |

|  |  |
| --- | --- |
| 15 | @Parameters |
| 16 | public static Collection<Object[]> parameter(){ | |

|  |  |  |
| --- | --- | --- |
| 17 | Object[][] pData=new Object[2][2]; | |
| 18 | pData[0][0]="Tom"; |

|  |  |
| --- | --- |
| 19 | pData[0][1]=30; |
| 20 | pData[1][0]="Harry"; | |

|  |  |
| --- | --- |
| 21 | pData[1][1]=40; |
| 22 | return Arrays.asList(pData); | |

|  |  |
| --- | --- |
| 23 | } |
| 24 | } |

**JUnit Assertions**

**JUnit assertEquals**: This checks if the two values are equal and assertion fails if both values are not equal.

This compares Boolean, int, String, float, long, char etc.

**Syntax**:  
*Assert.assertEqual(“excepted value”, ”actual value”);*

**Example**:  
*Assert.assertEqual(“ABC”,”ABC”);* //Both the strings are equal and assertion will pass.  
*Assert.assertEqual(“ABC”,”DEF”);* //Assertion will fail as both the strings are not equal.  
*Assert.assertEqual(“Strings are not equal”, “ABC”,”DEF”);* //message will be thrown if equal condition is not satisfied.

**Below is the example of use of JUnit assertion in selenium:**

|  |  |
| --- | --- |
| 1 | String username=driver.findElement(By.id(“username”)).getText(); |
| 2 | String password=driver.findElement(By.id(“password”)).getText(); |

|  |  |
| --- | --- |
| 3 | Assert.assertEqual(“Mismatch in both the string”, username, password); |

In above example assertion will fail as both the strings are not equal. One is text of username field and other is the text of password field.

**JUnit assertTrue**: Returns true if the condition is true and assertion fails if the condition is false.  
*Assert.assertTrue(“message”, condition);*  
*Assert.assertTrue(“Both the strings are not equal”, (“HelloWorld”).equals(“HelloWorld”));*

Here assertion will pass as both the strings match. It will print message if the assertion fails.

**JUnit assertFalse**: Returns true if the condition is false and assertion fails if the condition is true.  
*Assert.assertFalse(“message”, condition);*  
*Assert.assertFalse(“Both the strings are equal”, (“Hello”).equals(“HelloWorld”));*

There will not be any assertion error as the condition is false.

**Conclusion:**

Most of the programmers use Junit as it is easy and does not take much effort to test. A simple green or red bar will show the actual result of the test. Junit makes life easy as it has its own set of libraries and annotations. Here we have also described commonly used annotations used with selenium scripts and framework.

More detail about framework and use of Junit annotations will be discussed in upcoming tutorial which is dedicated exclusively for framework design using Junit. This tutorial will help us in designing the framework using Junit.