Test Automation & Advanced Selenium

Lesson 6: Web Driver Test with Xunit

Lesson Objectives

- Introduction to Xunit and Junit
- Junit Annotations
- Assertions/Verifications with Junit or TestNG
- Web Driver Test cases with Junit or TestNG
- Test Suite

Copyright © Capgemini 2015. All Rights Reserved

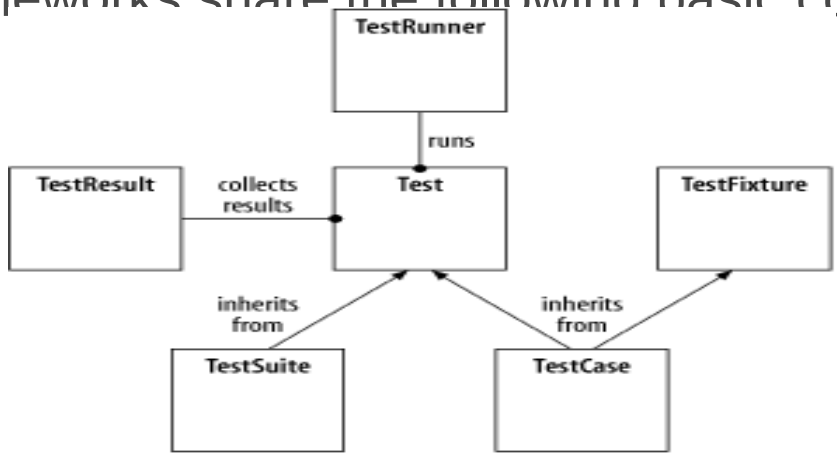
Web Driver Test with Xunit

- Is something not missing? We do Testing, where are Test Cases , Test Suite, Verifications, Results , Reports….
- Because, Selenium web driver is automation API not testing API
- So, combine Selenium automation with testing frameworks, like Xunit

Copyright © Capgemini 2015. All Rights Reserved

Introduction To XUNIT

- Xunit is the collective name for several unit testing frameworks that derive their structure and functionality from Smalltalk's SUnit.
- The names of many of these frameworks are a variation on "SUnit", usually substituting the "S" for the first letter (or letters) in the name of their intended language ("JUnit" for Java, "RUnit" for R etc.).
- These frameworks and their common architecture are collectively known as "Xunit".
- All Xunit frameworks share the following basic component architecture:



Copyright © Capgemini 2015. All Rights Reserved

Introduction to JUnit

- JUnit is a unit testing framework for the Java programming language.
- Important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as Xunit .
- JUnit is linked as a JAR at compile-time. The framework resides under package ”junit.framework ” for JUnit 3.8 and earlier, and under package ”org.junit” for JUnit 4 and later.

Copyright © Capgemini 2015. All Rights Reserved

Junit – Annotations

@Test:

- The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case. To run the method, JUnit first constructs a fresh instance of the class then invokes the annotated method. Any exceptions thrown by the test will be reported by JUnit as a failure. If no exceptions are thrown, the test is assumed to have succeeded.

```
1  public class MyTestClass {
2      @Test
3      public void myTestMethod() {
4          /**
5           * Use Assert methods to call your methods to be tested.
6           * A simple test to check whether the given list is empty or not.
7           */
8          org.junit.Assert.assertTrue( new ArrayList().isEmpty() );
9      }
10 }
```


Copyright © Capgemini 2015. All Rights Reserved

Junit – Annotations (Cont.)

@Before:

- When writing tests, it is common to find that several tests need similar objects created before they can run. Annotating a public void method with @Before causes that method to be run before the Test method. The @Before methods of super classes will be run before those of the current class

```
1  public class MyTestClass {
2      List<String> testList;
3      @Before
4      public void initialize() {
5          testList = new ArrayList<String>();
6      }
7      @Test
8      public void myTestMethod() {
9          /**
10          * Use Assert methods to call your methods to be tested.
11          * A simple test to check whether the given list is empty or not.
12          */
13         org.junit.Assert.assertTrue( testList.isEmpty() );
14     }
15 }
```


Copyright © Capgemini 2015. All Rights Reserved

Junit – Annotations (Cont.)

@After:

- If you allocate external resources in a Before method you need to release them after the test runs. Annotating a public void method with @After causes that method to be run after the Test method. All @After methods are guaranteed to run even if a Before or Test method throws an exception. The @After methods declared in super classes will be run after those of the current class.

```
public class MyTestClass {
    OutputStream stream;
    @Before
    public void initialize() {
        /**
         * Open OutputStream, and use this stream for tests.
         */
        stream = new FileOutputStream(...);
    }
    @Test
    public void myTestMethod() {
        /**
         * Now use OutputStream object to perform tests
         */
        ...
        ...
    }
    @After
    public void closeOutputStream() {
        /**
         * Close output stream here
         */
        try{
          if(stream != null) stream.close();
        } catch(Exception ex){
        }
    }
}
```


Copyright © Capgemini 2015. All Rights Reserved

Junit – Annotations (Cont.)

@BeforeClass:

- Annotating a public static void no-arg method with @BeforeClass causes it to be run once before any of the test methods in the class. The @BeforeClass methods of superclasses will be run before those the current class.
- The annotations @BeforeClass and @Before are same in functionality. The only difference is the method annotated with @BeforeClass will be called once per test class based, and the method annotated with @Before will be called once per test based.

```
public class MyTestClass {
    @BeforeClass
    public void initGlobalResources() {
        /**
         * This method will be called only once per test class.
         */
    }
    @Before
    public void initializeResources() {
        /**
         * This method will be called before calling every test.
         */
    }
    @Test
    public void myTestMethod1() {
        /**
         * initializeResources() method will be called before calling this method
         */
    }
    @Test
    public void myTestMethod2() {
        /**
         * initializeResources() method will be called before calling this method
         */
    }
}
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved

Junit – Annotations (Cont.)

@AfterClass:

- If you allocate expensive external resources in a BeforeClass method you need to release them after all the tests in the class have run. Annotating a public static void method with @AfterClass causes that method to be run after all the tests in the class have been run. All @AfterClass methods are guaranteed to run even if a BeforeClass method throws an exception. The @AfterClass methods declared in superclasses will be run after those of the current class.

```
public class MyTestClass {
    @BeforeClass
    public void initGlobalResources() {
        /**
         * This method will be called only once per test class. It will be called
         * before executing test.
         */
    }
    @Test
    public void myTestMethod1() {
        // write your test code here...
        ...
        ...
    }
    @AfterClass
    public void closeGlobalResources() {
        /**
         * This method will be called only once per test class. It will be called
         * after executing test.
         */
    }
}
```


Copyright © Capgemini 2015. All Rights Reserved

Junit – Annotations (Cont.)

@Ignore:

- Sometimes you want to temporarily disable a test or a group of tests. Methods annotated with Test that are also annotated with @Ignore will not be executed as tests. Also, you can annotate a class containing test methods with @Ignore and none of the containing tests will be executed. Native JUnit 4 test runners should report the number of ignored tests along with the number of tests that ran and

```
1  public class MyTestClass {
2      @Ignore
3      @Test
4      public void myTestMethod() {
5          /**
6           * This test will be ignored.
7           */
8          org.junit.Assert.assertTrue( new ArrayList().isEmpty() );
9      }
10 }
```


Copyright © Capgemini 2015. All Rights Reserved

Junit - Assertion

- JUnit provides overloaded assertion methods for all primitive types and Objects and arrays.
- The parameter order is expected value followed by actual value
- Some of the important methods of Assert class are:
  - void assertEquals(boolean expected, boolean actual)
    Check that two primitives/Objects are equal
  - void assertTrue(boolean expected, boolean actual)
    Check that a condition is true
  - void assertFalse(boolean condition)
    Check that a condition is false
  - void assertNotNull(Object object)
    Check that an object isn't null
  - void assertNull(Object object)
    Check that an object is null

Copyright © Capgemini 2015. All Rights Reserved

Junit – Assertion (Cont.)

- void assertSame(boolean condition)
  The assertSame() methods tests if two object references point to the same object

- void assertNotSame(boolean condition)
  The assertNotSame() methods tests if two object references not point to the same object

- void assertArrayEquals(expectedArray, resultArray);
  The assertArrayEquals() method will test whether two arrays are equal to each other

Copyright © Capgemini 2015. All Rights Reserved

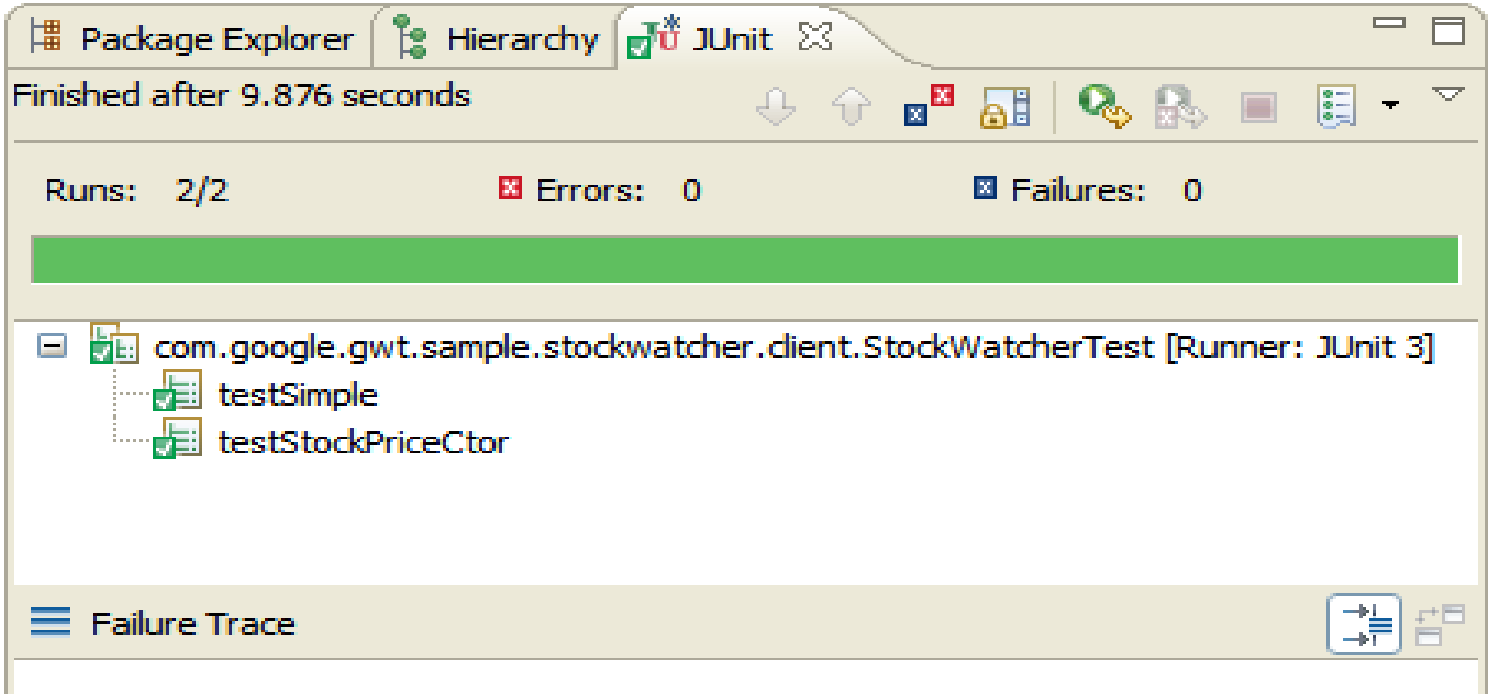
Junit – Assertion Example

```
1  public class TestAssertions {
2      @Test
3      public void testAssertions() {
4          //test data
5          String str1 = new String ("abc");
6          String str2 = new String ("abc");
7          String str3 = null;
8          String str4 = "abc";
9          String str5 = "abc";
10         int val1 = 5;
11         int val2 = 6;
12         String[] expectedArray = {"one", "two", "three"};
13         String[] resultArray =  {"one", "two", "three"};
14         //Check that two objects are equal
15         assertEquals(str1, str2);
16         //Check that a condition is true
17         assertTrue (val1 < val2);
18         //Check that a condition is false
19         assertFalse(val1 > val2);
20         //Check that an object isn't null
21         assertNotNull(str1);
22         //Check that an object is null
23         assertNull(str3);
24         //Check if two object references point to the same object
25         assertSame(str4,str5);
26         //Check if two object references not point to the same object
27         assertNotSame(str1,str3);
28         //Check whether two arrays are equal to each other.
29         assertArrayEquals(expectedArray, resultArray);
30     }
31 }
```


Copyright © Capgemini 2015. All Rights Reserved

Junit – Reports

- Junit report collects individual XML files
- Merge the individual XML files generated by the JUnit task and eventually apply a stylesheet on the resulting merged document to provide a browsable report of the test cases results



Copyright © Capgemini 2015. All Rights Reserved

Web Driver Test cases with TestNG

- Open source Java testing framework, not limited to unit tests
- Designed to be better than JUnit, especially when testing integrated classes
- Supports parameterized tests out-of-the-box (in much more convenient way than JUnit does)
- Facilitates running multi-threaded tests
- Allows to express dependencies between test methods
- Integrates very well with the build tools: Ant, Maven and Gradle
- Supported by all major IDEs
- Can be used with different JVM languages (e.g. Java, Groovy, Scala) and cooperates with many quality and testing tools (e.g. code coverage tools, mocking libraries, matchers libraries)
- Some popular solutions - e.g. Spring Framework - provide means to facilitate testing with TestNG

Capgemini CONSULTING · TECHNOLOGY · OUTSOURCING
Copyright © Capgemini 2015. All Rights Reserved

Web Driver Test cases with TestNG (Cont.)

Writing a test is typically a three-step process:

- Write the business logic of your test and insert TestNG annotations in your code.
- Add the information about your test (e.g. the class name, the groups you wish to run, etc...) in a testng.xml file or in build.xml.
- Run TestNG

```
import org.testng.annotations.Test;
import static org.testng.Assert.assertEquals;

public class TestNGSimpleTest {
        @Test
        public void testAdd() {
                String str = "TestNG is working fine";
                assertEquals("TestNG is working fine", str);
        }
}
```

Test Case Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Suite1">
  <test name="test1">
    <classes>
      <class name="TestNGSimpleTest"/>
    </classes>
  </test>
</suite>
```

testng.xml File

Copyright © Capgemini 2015. All Rights Reserved

TestNG Annotations

Annotation	Description
@Test	The annotation notifies the system that the method annotated as @Test is a test method
@BeforeSuite	The annotation notifies the system that the method annotated as @BeforeSuite must be executed before executing the tests in the entire suite
@AfterSuite	The annotation notifies the system that the method annotated as @AfterSuite must be executed after executing the tests in the entire suite
@BeforeTest	The annotation notifies the system that the method annotated as @BeforeTest must be executed before executing any test method within the same test class

Copyright © Capgemini 2015. All Rights Reserved

TestNG Annotations (Cont.)

@AfterTest	The annotation notifies the system that the method annotated as @AfterTest must be executed after executing any test method within the same test class
@BeforeClass	The annotation notifies the system that the method annotated as @BeforeClass must be executed before executing the first test method within the same test class
@AfterClass	The annotation notifies the system that the method annotated as @AfterClass must be executed after executing the last test method within the same test class
@BeforeMethod	The annotation notifies the system that the method annotated as @BeforeMethod must be executed before executing any and every test method within the same test class

Copyright © Capgemini 2015. All Rights Reserved

TestNG Annotations (Cont.)

@AfterMethod	The annotation notifies the system that the method annotated as @AfterMethod must be executed after executing any and every test method within the same test class
@BeforeGroups	The annotation notifies the system that the method annotated as @BeforeGroups is a configuration method that enlists a group and that must be executed before executing the first test method of the group
@AfterGroups	The annotation notifies the system that the method annotated as @AfterGroups is a configuration method that enlists a group and that must be executed after executing the last test method of the group

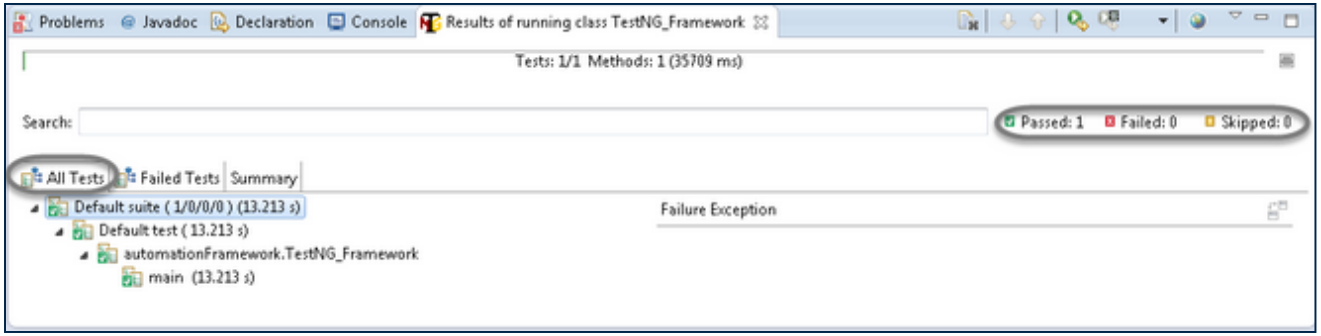
Copyright © Capgemini 2015. All Rights Reserved

TestNG Result

- TestNG result is displayed into two windows as shown below:
  - Console Window
  - Testing Result Window

a

b



Copyright © Capgemini 2015. All Rights Reserved

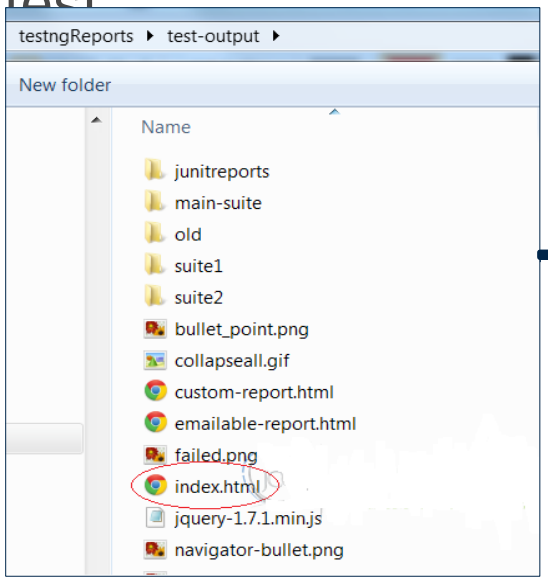
TestNG Reports

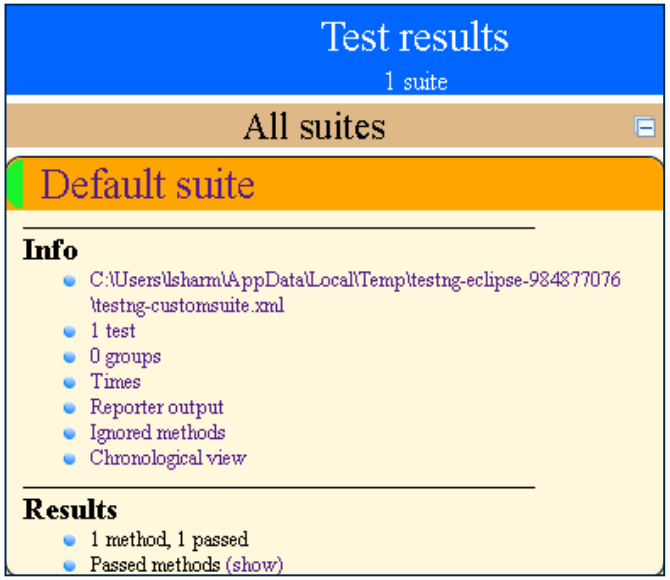
- Generates a different type of report for test execution
- Whenever TestNG is run, HTML and XML reports are generated by default in the directory
- For implementing a reporting class, the class has to implement an org.testng.IReporter interface
- Has its own reporter objects which are called when whole suite run ends
- Object containing the information of the whole test run is passed on to the report implementations
- Default implementations are:
  - Main
  - Failed Reporter
  - XML Reporter
  - EmailableReporter2
  - JUnitReport Reporter
  - SuiteHTML Reporter

Copyright © Capgemini 2015. All Rights Reserved

TestNG Reports

- In Main report layout ,test-output directory contains HTML reports like an index.html file, that is the entry point to the TestNG HTML report.
- The top-level report gives us a list of all the suites that were just run, along with an individual and compound total for each passed, failed, and skipped test





Copyright © Capgemini 2015. All Rights Reserved

Test Suite(JUnit)

- Test suite means bundle a few unit test cases and run it together.
- In JUnit, both @RunWith and @Suite annotation are used to run the suite test.

Example of Test Suite in JUnit:

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({
  TestFeatureLogin.class,
  TestFeatureLogout.class,
  TestFeatureNavigate.class,
  TestFeatureUpdate.class
})

public class FeatureTestSuite {
  // the class remains empty,
  // used only as a holder for the above annotations
}
```


Copyright © Capgemini 2015. All Rights Reserved

Test Suite(TestNG)

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="example suite 1" verbose="1" >
  <test name="Regression suite 1" >
    <classes>
      <class name="com.first.example.demoOne"/>
      <class name="com.first.example.demoTwo"/>
      <class name="com.second.example.demoThree"/>
    </classes>
 </test>
</suite>
```

In the above xml

class name has been specified as “com.first.example.demoOne” and “com.first.example.demoOne” which are in “com.first.example” package

Class name demoThree is in package “com.second.example.”

Copyright © Capgemini 2015. All Rights Reserved

Summary

- In this lesson, you have learnt
  - In this lesson, you have understood that Xunit is the latest technology for unit testing.
  - JUnit is an open source framework which is used for writing & running tests.
  - Junit Provides Annotation to identify the test methods, Assertions for testing expected results and also provides Test runners for running tests.
  - You have also understood how to execute Web Driver with Junit ,Testing and Test Suite.
  - Test suite enables you to execute the bundle of unit test cases at a time .
  - The only drawback of Xunit is:
    - Lack of documentation- Compared to MSTest and NUnit, xUnit.NET lacks documentation

Copyright © Capgemini 2015. All Rights Reserved

Review Question

- Question 1
  - Select the Annotation which is NOT part of JUnit Annotations
  - @After
  - @After or Before
  - @Before
  - @AfterClass

- Question 2: True/False
  - The Selenium web driver is automation API not testing AP

- Question 3: Fill in the Blanks
  - The assertSame() methods tests if two object references point to the _____ .

Copyright © Capgemini 2015. All Rights Reserved