JUnit

ADVANCED PROGRAMMING PRACTICES

Agenda

Software testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.

Software Testing involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Benefits of Software Testing

Cost-Effective: Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.

Security: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.

Product quality: Testing ensures a quality product is delivered to customers.

Customer Satisfaction: UI/UX Testing ensures the best user experience.

Product improvement: Time consuming but productive activity.

Automated Testing: Reduce testing time.

Unit testing

Before unit testing, we depend on deploying the entire app and checking if the software works as expected. But that’s not very efficient. And it is manual.

Unit Testing focuses on writing automated tests for individual classes and methods.

This software testing approach is followed by the programmer to test the unit of the program. It helps developers to know whether the individual unit of the code is working properly or not.

The important thing about unit testing is that these tests can be run with continuous integration - as soon as some code changes.

What is JUnit?

JUnit is an open-source Unit Testing Framework for Java programming language.

JUnit has been important in the development of test-driven development.

It is one of a family of unit testing frameworks collectively known as xUnit, that originated with JUnit.

As the name suggests it is used for testing small chunk of code or unit by creating a path, function or a method.

Advantages and uses

- It finds bugs early in the code, which makes our code more reliable.
- JUnit is useful for developers, who work in a test-driven environment.
- Unit testing forces a developer to read code more than writing.
- You develop more readable, reliable and bug-free code which builds confidence during development.
- Best testing framework that can be selected for Efficient testing process
- More application Developer IDEs includes Junit.
- Provides AWT and Spring based graphical test reporting mechanism
- JUnit test Frameworks enables it to test within the application server’s container
- A benchmark for testing in java programming language.

Features of JUnit

- It allows you to write code faster, which increases quality.
- Elegantly simple and less complex.
- Provides annotations to identify test methods and assertion statements for testing expected results.
- It provides test runners and for running the tests.
- Tests can be run automatically.
- Tests can be organized and into test suites.

Junit Framework

Junit is regression testing framework which is used to implement unit testing in Java. This framework also allows quick and easy generation of test data and test cases.

Important Features:

- Fixtures
- Test Suites
- Test Runners
- Junit Classes

Test Fixtures :

Test fixtures ensures that there is a well-known and fixed environment in which tests are run so that results are repeatable.

It includes methods:

setUp() : runs before test invocation. We use an annotation before() in this case.

tearDown() : runs after all test methods. We use annotation after().

Test Suites:

If you want to execute multiple test cases in a specific order, it can be done by combining all the test cases in a single origin. This origin is called as test suites.

It includes annotations:

@runWith & @suiteClasses: used to run the suite test.

```
//JUnit Suite Test
@RunWith(Suite.class)
@Suite.SuiteClasses({
TestJunit1.class ,
            TestJunit2.class
})
public class JunitTestSuite {
}
```

Test Runners:

Test runner is used for executing the test cases.

RunClasses method is used to run one or several test cases.

Return type of this method is the result object that is used to access information about the tests.

```
public class TestRunner {
        public static void main(String[] args) {
                Result result = 
JUnitCore.runClasses(TestJunit.class);
                for (Failure failure : result.getFailures()) {
                        System.out.println(failure.toString());
                }
                System.out.println(result.wasSuccessful());
        }
}
```

JUnit Classes:

JUNit classes are Used in writing and testing JUNits.

Important classes:

Asserts : Contains a set of assert methods.

TestCase : Contains a test case that defines the fixture to run multiple tests.

TestResult : Contains methods to collect the results of executing a test case.

How to write a JUnit Program

Set Up Junit

Install JDK	Install JDK in your machine by going to Java Download page and click on the option for Java Platform (Java Development Kit). https://www.oracle.com/ca-en/java/technologies/javase-downloads.html
Set JDK Path	Set Path of JDK in your system by setting Java Environment and ClassPath variable
Verify Java	Verify JDK in your system by command: java –version
Download & Install IDE	Download & Install Java IDE eg: Eclipse(recommended), Intellij. Link to download Eclipse: https://www.eclipse.org/downloads/packages/release/helios/sr1/eclipse-ide-java-developers
Add JUnit library	Add JUnit library to your project

Example of JUnit Annotations

JUnit Annotations

An annotation is a special form of syntactic metadata that can be added to the java source code for better code readability.

- **@Test :** Test annotation tells you that the public void method can be run as a test case.
- **@before :** Annotating a public void method with @Before causes that method to be run before each test method.
- **@After :** If you allocate external resources in a before method, you need to release them after the test runs. Annotating the method with @After causes that method to be run after the Test method.
- **@BeforeClass :** Annotating a public static void method with @BeforeClass causes it to be run once before any of the test method in the class.
- **@AfterClass :** This will perform the method after all tests are run. This can be used to perform clean-up activities.
- **@ignore :** The ignore annotation is used to ignore the test and that test will not be executed.

Example of JUnit Assert Statement

JUnit Assert Statements

Assert is method used in determining pass or fail status of a test case. In Junit, all the assertations are in the Assert class.

1. void assertEquals(boolean expected, boolean actual): checks that 2 primitives/objects are equal.

2. void assertTrue(Boolean condition) : checks whether condition is true.

3. void assertFalse(Boolean condition) : checks whether condition is false.

4. void assertNotNull (Object object) : checks whether an object isn't null.

5. void assertNull (Object object) : checks whether an object is null.

6. void assertSame (object1, object2) : The assertSame() method tests if two object references point to the same object.

7. void assertNotSame(object1, object2) : The assertNotSame() method tests if two object references do not point to the same object.

8. void assertArrayEquals(expectedArray, resultArray) : The assertArrayEquals() method will test whether two arrays are equal to each other.

THANK YOU