ABOUT VINOD KUMAR KASHYAP

Vinod is Sun Certified and love to work in Java and related technologies. Having more than 10 years of experience, he had developed software's including technologies like Java, Hibernate, Struts, Spring, HTML 5, jQuery, CSS, Web Services, MongoDB, AngularJS. He is also a JUG Leader of Chandigarh Java User Group.

JUnit Testcase Example

Posted by: Vinod Kumar Kashyap in junit April 6th, 2017

In this example, we shall show users how to use JUnit Testcase. JUnit Testcase example will follow you to the scenarios of how we can use and test our methods with the help of the JUnit Testcase.

As a regular reader, you are already familiar with the JUnit and its usage. If you are new to it you are recommend to visit JUnit Series on Java Code Geeks.

Before this article, I have shown many different uses of JUnit and its many different properties. But in this example, we will show something different which we haven't done in previous articles.

1. Introduction

JUnit provides a class known as

```
TestCase
```

. This class will help us to run all our test cases. In previous articles, we have used the

```
@Test
```

annotation on all our test cases. But, here we will not use any annotation and see how we can test the methods with the help of the

```
TestCase
```

```
TestCase
```

class and then proceed further for testing. Let's start creating a project.

2. Technologies Used

Following technologies are used in this example.

- **Java 8**: Language for example
- **JUnit 4.12**: Testing Framework
- **Maven**: Dependency and build tool
- **Eclipse**: IDE for coding

3. Project Setup

Tip
You may skip project creation and jump directly to the **beginning of the example** below.
We will be using the Eclipse and Maven for our project. Click **File -> New -> Maven Project** On the first screen simply check the check mark corresponding to the "Create a simple project" and click on **Next** button.

Figure 1: JUnit Testcase Example Setup 1

Fill in all the details and click on **Finish** button.

JUnit Testcase Example Setup 2

Now our blank project is ready.

4. JUnit Testcase Example

We will be doing 2 things here. First configure the JUnit by adding dependency in the

```
pom.xml
```

and secondly, creating java classes for testing.

4.1 xml configuration

Firstly, put the following code in the

```
pom.xml
```

.

pom.xml

```xml
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
    </dependency>
</dependencies>

<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

Here we are simply telling maven to pull dependency of

```
JUnit 4.12
```

and use

for compile.

4.2 Java classes

Next step is to create a model class which will help in testing.

ListTest.java

```java
package junittestcase;

import java.util.ArrayList;
import java.util.List;

public class ListTest {

    private List lstFruits = new ArrayList();

    public void add(String fruit) {
        lstFruits.add(fruit);
    }

    public void remove(String fruit) {
        lstFruits.remove(fruit);
    }

    public int size() {
        return lstFruits.size();
    }

    public void removeAll(){
        lstFruits.clear();
    }
}
```

Now we are ready to write our main class that will actually run our test cases. We need to follow some points so that we run our test cases:

- We need to extend the
  ```
  TestCase
  ```
  class of JUnit

-
  ```
  setUp()
  ```
  and
  ```
  tearDown()
  ```
  methods, which are optional and runs before and after every test case.
- Every test case shoulds be public and name should begin with ***test***. They should not take any argument.
- Test case should not return any value.

MyTestCase.java

```java
01 package junittestcase;
02 
03 import junit.framework.TestCase;
04 
05 public class MyTestCase extends TestCase {
06 
07     protected ListTest lstTest = new ListTest();
08 
09     protected void setUp() {
10         lstTest.add("Apple");
11         lstTest.add("Orange");
12         lstTest.add("Grapes");
13     }
14 
15     public void testSize() {
16         assertEquals("Checking size of List", 3, lstTest.size());
17     }
18 
19     public void testAdd() {
20         lstTest.add("Banana");
21         assertEquals("Adding 1 more fruit to list", 4, lstTest.size());
22     }
23 
24     public void testRemove() {
25         lstTest.remove("Orange");
26         assertEquals("Removing 1 fruit from list", 2, lstTest.size());
27     }
28 
29     protected void tearDown() {
30         lstTest.removeAll();
31     }
32 }
```

Let's analyze all the details line by line of this class.
At line no 5, we have extended the

class.
Line no 9, specifies a

```
setUp()
```

method that will run before any test method.
Line 15,19,24 are our test cases.
Line no 29 specifies the

```
tearDown()
```

method that will be called after each test case.

It is to be noted that,

```
setUp()
```

and

```
tearDown()
```

are not mandatory at all. We have used here only to illustrate the usage of them.

Output

Figure 3: JUnit Testcase Example Output

5. Conclusion

We have learned a way to test our test cases with the help of the

```
TestCase
```

class of JUnit. Now a days we are using annotations instead of the

```
TestCase
```

class. But through JUnit Testcase example we have learned a new way to test.

6. Download the Eclipse Project

This is a JUnit Testcase Example.

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