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 Hamcrest Example

Posted by: Vinod Kumar Kashyap in junit March 9th, 2017

In this example we shall show users the usage of Hamcrest. Through JUnit Hamcrest Example we will show users what is hamcrest, where it is used, why it is used, when it is used and how to use it on your applications. If you are a regular user of my JUnit series then you are already familiar with the JUnit.

If you want to see more example of JUnit, please visit my series page.

We will start by getting a little bit information about the hamcrest. It has a very good integration with JUnit and both provides a good framework for testing.

1. Introduction

Hamcrest is a open source framework matcher library used in various language to match expression for your test cases. You can visit github page if you want to explore the code of it.
Hamcrest has a very rich library of methods to fulfill our needs. It is used with different testing frameworks like JUnit and jMock.

Hamcrest is typically viewed as a third generation matcher framework.

- **First Generation**: It typically uses

```
assert(some statement)
```

. In this case tests are not easily readable.
- **Second Generation**: It uses the special methods such as

for test. But this approach creates a lots of assert methods.

- **Third Generation**: It uses

```
assertThat()
```

method for test. It is more flexible and covers most of the scenarios. The benefit is that you still get fluent error messages when the assertion fails, but now you have greater extensibility.

In our example we will use

```
assertThat()
```

for all our tests.

2. Technologies Used

- Java
- JUnit 4.12 – Testing framework
- Hamcrest 1.3 – Used for matchers
- Eclipse – IDE for code
- Maven – dependency management tool

3. Project SetUp

Tip
You may skip project creation and jump directly to the **beginning of the example** below.
Open Eclipse. Select

```
File -> New -> Maven Project
```

. Fill in the following details.

Figure 1: JUnit Hamcrest Example Setup 1

Figure 2: JUnit Hamcrest Example Setup 2

This will create an empty maven project.

4. JUnit Hamcrest Example

Start by writing following lines to the

`pom.xml`

pom.xml

```
    <dependencies>
        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.hamcrest/hamcrest-library -->
        <dependency>
            <groupId>org.hamcrest</groupId>
            <artifactId>hamcrest-library</artifactId>
            <version>1.3</version>
        </dependency>
    </dependencies>
```

If you simply write JUnit in

`pom.xml`

, it will fetch

`hamcrest-core`

jar with it. But we have also included

`hamcrest-library`

to run our test cases.

Employee.java

```
01 package junithamcrest;
02
03 import java.util.List;
04
05 public class Employee {
06
07     private Long empId;
08     private String empName;
09     private String gender;
10     private List awards;
11
12     public Employee(Long empId, String empName, String gender, List awards) {
13         super();
14         this.empId = empId;
15         this.empName = empName;
16         this.gender = gender;
17         this.awards = awards;
18     }
19
20     public Long getEmpId() {
21         return empId;
22     }
23
24     public void setEmpId(Long empId) {
25         this.empId = empId;
26     }
27
28     public String getEmpName() {
29         return empName;
30     }
31
32     public void setEmpName(String empName) {
33         this.empName = empName;
34     }
35
36     public String getGender() {
37         return gender;
38     }
39
40     public void setGender(String gender) {
41         this.gender = gender;
42     }
43
44     public List getAwards() {
45         return awards;
46     }
47
48     public void setAwards(List awards) {
49         this.awards = awards;
50     }
51 }
```

Now create a test class, so that we can test above class. This class will test all hamcrest matchers. We have tried to cover most common but users are advised to see other matchers if they want to dig more deeper.

JUnitHamcrestTestClass.java

```
001 package junithamcrest;
002
003 import static org.hamcrest.CoreMatchers.anyOf;
004 import static org.hamcrest.CoreMatchers.equalTo;
005 import static org.hamcrest.CoreMatchers.everyItem;
006 import static org.hamcrest.Matchers.allOf;
007 import static org.hamcrest.Matchers.containsString;
008 import static org.hamcrest.Matchers.endsWith;
009 import static org.hamcrest.Matchers.equalToIgnoringCase;
010 import static org.hamcrest.Matchers.hasProperty;
011 import static org.hamcrest.Matchers.hasSize;
012 import static org.hamcrest.Matchers.instanceOf;
013 import static org.hamcrest.Matchers.is;
014 import static org.hamcrest.Matchers.isA;
015 import static org.hamcrest.Matchers.isIn;
016 import static org.hamcrest.Matchers.notNullValue;
017 import static org.hamcrest.Matchers.startsWith;
018 import static org.hamcrest.Matchers.stringContainsInOrder;
019 import static org.hamcrest.Matchers.emptyCollectionOf;
020 import static org.junit.Assert.assertThat;
021
022 import java.util.ArrayList;
023 import java.util.Arrays;
024 import java.util.List;
025
026 import org.junit.BeforeClass;
027 import org.junit.Test;
028
029 public class JUnitHamcrestTestClass {
030
031     // Creating instances
032     private static Employee empA;
033     private static Employee empB;
034     private static List strList = Arrays.asList("Apple", "Apricot", "August");
035
036     @BeforeClass
037     public static void init() {
```

```
042     }
043
044     /**
045      * This method will test functionality of <code>is</code> matcher.
046      */
047     @Test
048     public void isTest() {
049         // checks that empA is an object of Employee class
050         assertThat(empA, isA(Employee.class));
051
052         // below are 3 versions of "is" method. All are same
053         assertThat(2, equalTo(2));
054         assertThat(2, is(equalTo(2)));
055         assertThat(2, is(2));
056     }
057
058     /**
059      * This method will test functionality of <code>beans</code> matcher.
060      */
061     @Test
062     public void beansTest() {
063         // checks that object contains the property
064         assertThat(empA, hasProperty("empName"));
065
066         // checks that the object contains the property with a value
067         assertThat(empA, hasProperty("empName", equalTo("Vinod Kumar Kashyap")));
068     }
069
070     /**
071      * This method will test functionality of <code>collections</code> matcher.
072      */
073     @Test
074     public void collectionsTest() {
075         // checks that object is of checked size
076         assertThat(empA.getAwards(), hasSize(2));
077
078         // checks a collection for the element present
079         assertThat("Best Team", isIn(empA.getAwards()));
080
081         // checks for empty collection
082         assertThat(new ArrayList(), emptyCollectionOf(String.class));
083     }
084
085     /**
086      * This method will test functionality of <code>String</code> matcher.
087      */
088     @Test
089     public void stringTest() {
090         assertThat(empA.getEmpName(), containsString("Kumar"));
091         assertThat(empA.getEmpName(), endsWith("Kashyap"));
092         assertThat(empB.getEmpName(), startsWith("Asmi"));
093
094         // checks by ignoring case
095         assertThat(empA.getEmpName(), equalToIgnoringCase("vinod KUMAR Kashyap"));
096
097         // checks that the elements are occurring in the same order
098         assertThat(empA.getEmpName(), stringContainsInOrder(Arrays.asList("Vinod", "Kashyap")));
099     }
100
101     /**
102      * Other common matchers
103      */
104     @Test
105     public void otherCommonTest() {
106         // all of the conditions should be met to pass the case
107         assertThat(empB.getGender(), allOf(startsWith("F"), containsString("ale")));
108
109         // any of the conditions should be met to pass the case
110         assertThat(empB.getEmpName(), anyOf(startsWith("Dhwani"), endsWith("yap")));
111
112         // checks that value is not null
113         assertThat(empA, is(notNullValue()));
114
115         // checks that object id instance of a Class
116         assertThat(empA.getEmpId(), instanceOf(Long.class));
117
118         // checks every item in  list
119         assertThat(strList, everyItem(startsWith("A")));
120     }
121 }
```

Now we will start with explanation part by part of the example. Most of the matchers are self explanatory.

4.1 Is Matcher

This is one of the most common matcher used.

```
01 @Test
02 public void isTest() {
03     // checks that empA is an object of Employee class
04     assertThat(empA, isA(Employee.class));
05
06     // below are 3 versions of "is" method. All are same
07     assertThat(2, equalTo(2));
08     assertThat(2, is(equalTo(2)));
09     assertThat(2, is(2));
```

```
is(T value)
```

is overloaded to return

```
is(equalTo(value))
```

.

4.2 Beans Matchers

These matchers are used to check out beans.

```
@Test
public void beansTest() {
    // checks that object contains the property
    assertThat(empA, hasProperty("empName"));

    // checks that the object contains the property with a value
    assertThat(empA, hasProperty("empName", equalTo("Vinod Kumar Kashyap")));
}
```

If you see here, we are testing that our class has a property associated with it or not.

4.3 Collections Matchers

These matchers works with the collections. JUnit and Hamcrest provides various ways to test collections.

```
@Test
public void collectionsTest() {
    // checks that object is of checked size
    assertThat(empA.getAwards(), hasSize(2));

    // checks a collection for the element present
    assertThat("Best Team", isIn(empA.getAwards()));

    // checks for empty collection
    assertThat(new ArrayList(), emptyCollectionOf(String.class));
}
```

4.4 String Matchers

These matchers helps to work with Strings.

```
@Test
public void stringTest() {
    assertThat(empA.getEmpName(), containsString("Kumar"));
    assertThat(empA.getEmpName(), endsWith("Kashyap"));
    assertThat(empB.getEmpName(), startsWith("Asmi"));

    // checks by ignoring case
    assertThat(empA.getEmpName(), equalToIgnoringCase("vinod KUMAR Kashyap"));

    // checks that the elements are occurring in the same order
    assertThat(empA.getEmpName(), stringContainsInOrder(Arrays.asList("Vinod", "Kashyap")));
}
```

4.5 Other Common Matchers

There are many different matchers used. Here we are using some of the common matchers. It is also possible to chain matchers, via the

```
anyOf()
```

of

```
allOf()
```

method. See below for details.

```
@Test
public void otherCommonTest() {
    // all of the conditions should be met to pass the case
    assertThat(empB.getGender(), allOf(startsWith("F"), containsString("ale")));

    // any of the conditions should be met to pass the case
    assertThat(empB.getEmpName(), anyOf(startsWith("Dhwani"), endsWith("yap")));

    // checks that value is not null
    assertThat(empA, is(notNullValue()));

    // checks that object id instance of a Class
    assertThat(empA.getEmpId(), instanceOf(Long.class));

    // checks every item in  list
    assertThat(strList, everyItem(startsWith("A")));
}
```

matcher. It defines that all of the condition inside should match to pass the test.
Line no 7 uses

```
anyOf()
```

matcher, which tells us that if any of the condition is matched, test case will pass. As you see that in our scenario first condition is not matched but second one does. This case passes with flying colors. Because any one of the conditions is true.
Line no 16 scans every item in a list and matches the condition. If condition is matched it will pass the test.

5. Output

For running the example simply right click on class and

```
Run As -> Junit Test
```

. You will see the following output in JUnit console.

Figure 3: JUnit Hamcrest Example Output

5. Conclusion

JUnit Hamcrest Example focus mostly on the usage of the JUnit Hamcrest matchers. You have learned what is hamcrest, why we should use it, where it should be used, how to use it. This example shows the usage of a simple example which covers most if the matchers.

6. Download the Eclipse project

This is JUnit Hamcrest Example.

Download
You can download the full source code of this example here: **JUnitHamcrest.zip**

Tagged with: HAMCREST