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 Test Case Example for Web Application

Posted by: Vinod Kumar Kashyap in junit May 2nd, 2017

In this tutorial, we shall show users the usage of JUnit Example Web Application. We will see how we can test our web applications and what are the technologies that we need to work with.

Table Of Contents

1. Introduction

In this example we will try to find a solution of testing our web applications with the help of JUnit.

2. Technology Stack

In this example we will e using following technology stack:

- **Java 1.8**– We will be using the Java 1.8 for this example to work.
- **JUnit 4.12**– Testing framework
- **Maven**– Build and dependency tool. You can visit **here** for more details
- **JWebUnit**– Framework used with JUnit to test the web application. We will go into details in our example below.
- **Eclipse** – IDE for coding

3. What is JUnit?

There are several technologies out there which are used to test the applications. JUnit is one of them. It is very famous framework which is used by the Java developers for unit testing their applications. It provides lots of flexibility to test the application from a developers point of view.

You can get a detail tutorials about JUnit written by me **here**. Also you can follow the **link** to get all Junit related tutorials on Java Code Geeks.

Currently, latest stable version of JUnit is 4.x and 5.x is coming most probably in Q1 of 2017. JUnit contains many annotations that are used while creating test cases.

- **@BeforeClass**: It is used to write code that we want to run before all test cases.
- **@Before**: It will run before every test case.
- **@Test**: This is actual test case.
- **@After**: It will run after every test case.
- **@AfterClass**: It is used to write code that we want to run after all test cases.

For the sake of simplicity of the example, we are using the Maven so that you don't need to include the jar yourself. Maven is dependency management tool for Java. The jar and its dependencies would be automatically pulled by Maven.

I would recommend readers to read the **Mastering Unit Testing Using Mockito and JUnit** and **JUnit in Action** for the deeper knowledge about the JUnit.

4. Ways to test Web Applications

Since in last all tutorials I have written, I had used the core java applications to test with JUnit. But we can also test the web applications with the help of JUnit. There are numerous frameworks available which in collaboration with JUnit helps in testing. You can use any one of them to test your web applications. Some of them are:

- **JWebUnit**
- **HttpUnit**
- **Selenium**

But in this example we will stick to the JWebUnit.

5. JWebUnit Introduction

JWebUnit is a Java based testing framework for web applications. **JWebUnit provides a high-level Java API for navigating a web application** combined with a set of assertions to verify the application's correctness. This includes navigation via links, form entry and submission, validation of table contents, and other typical business web application features.

Here is the architecture of JWebUnit.

Figure 1: JWebUnit Architecture

It wraps existing testing frameworks such as **HtmlUnit** and **Selenium** with a unified, simple testing interface to allow you to quickly test the correctness of your web applications.

You can read **Selenium Web Driver Practical Guide** about the usage of selenium web drivers to use for testing.

6. Project Setup

Tip
You may skip project creation and jump directly to the **beginning of the example** below.
Open Eclipse. **File -> New -> Maven Project**. First Screen will open. Go with defaults and click on **Next** button.

Figure 2: JUnit Web Testing Setup 1

On this screen select **maven-archetype-webapp** from options and click on **Next** button.

Figure 3: JUnit Web Testing Setup 2

On this final screen, fill in the details as shown and click on the **Finish** button.

Figure 4: JUnit Web Testing Setup 3

Here we are ready with the blank maven web project. But before starting code, you need to create a folder under **src/main** and name it **java**. By default maven creates a **src/main/resorces** folder.

6.1 JWebUnit Installation

To continue using our example for testing web application we need to add the JUnitWeb jar file to out classpath. This can be achieved either by deploying directly the jar file or using the Maven.
Since we are using Maven for our example we will be using the

```
pom.xml
```

for the dependency of the JUnitWeb jar.

Copy the below code and paste it onto the pom.xml file under the

```
dependences
```

tag.

pom.xml

```
<dependency>
    <groupId>net.sourceforge.jwebunit</groupId>
    <artifactId>jwebunit-htmlunit-plugin</artifactId>
    <version>3.3</version>
    <scope>test</scope>
</dependency>
```

7. JUnit Example Web Application

Here is the final structure of our example.

Figure 5: JUnit Web Testing Structure

Let's start by adding some code to the

```
pom.xml
```

file.

pom.xml

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
<properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

Now we are ready to go with the coding of our example. But before starting the coding first we need to update the project so that it will use Java 8 for compilation and running. Simply **right click on the project name -> Maven -> Update Project**