

EASYMOCK - EXCEPTION HANDLING

http://www.tutorialspoint.com/easymock/easymock_exception_handling.htm

Copyright © tutorialspoint.com

EasyMock provides the capability to a mock to throw exceptions, so exception handling can be tested. Take a look at the following code snippet.

```
//add the behavior to throw exception

EasyMock.expect(calc Service.add(10.0,20.0)).and Throw(new Runtime Exception("Add
operation not implemented"));
```

Here we've added an exception clause to a mock object. MathApplication makes use of calcService using its add method and the mock throws a RuntimeException whenever calcService.add method is invoked.

Example

Step 1: Create an interface called CalculatorService to provide mathematical functions

File: CalculatorService.java

```
public interface CalculatorService {
    public double add(double input1, double input2);
    public double subtract(double input1, double input2);
    public double multiply(double input1, double input2);
    public double divide(double input1, double input2);
}
```

Step 2: Create a JAVA class to represent MathApplication

File: MathApplication.java

```
public class MathApplication {
    private CalculatorService calcService;

    public void setCalculatorService(CalculatorService calcService){
        this.calcService = calcService;
    }

    public double add(double input1, double input2){
        return calcService.add(input1, input2);
    }

    public double subtract(double input1, double input2){
        return calcService.subtract(input1, input2);
    }

    public double multiply(double input1, double input2){
        return calcService.multiply(input1, input2);
    }

    public double divide(double input1, double input2){
        return calcService.divide(input1, input2);
    }
}
```

Step 3: Test the MathApplication class

Let's test the MathApplication class, by injecting in it a mock of calculatorService. Mock will be created by EasyMock.

File: MathApplicationTester.java

```

import org.easymock.EasyMock;
import org.easymock.EasyMockRunner;
import org.easymock.Mock;
import org.easymock.TestSubject;

import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;

// @RunWith attaches a runner with the test class to initialize the test data
@RunWith(EasyMockRunner.class)
public class MathApplicationTester {

    // @TestSubject annotation is used to identify class which is going to use the mock
    object
    @TestSubject
    MathApplication mathApplication = new MathApplication();

    // @Mock annotation is used to create the mock object to be injected
    @Mock
    CalculatorService calcService;

    @Test(expected = RuntimeException.class)
    public void testAdd(){

        //add the behavior to throw exception
        EasyMock.expect(calcService.add(10.0, 20.0)).andThrow(new
            RuntimeException("Add operation not implemented"));

        //activate the mock
        EasyMock.replay(calcService);

        //test the add functionality
        Assert.assertEquals(mathApplication.add(10.0, 20.0), 30.0, 0);

        //verify call to calcService is made or not
        EasyMock.verify(calcService);
    }
}

```

Step 4: Execute test cases

Create a java class file named TestRunner in **C:\> EasyMock_WORKSPACE** to execute Test cases .

File: TestRunner.java

```

import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class TestRunner {
    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(MathApplicationTester.class);

        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }

        System.out.println(result.wasSuccessful());
    }
}

```

Step 5: Verify the Result

Compile the classes using **javac** compiler as follows:

```
C:\EasyMock_WORKSPACE>javac MathApplicationTester.java
```

Now run the Test Runner to see the result:

```
C:\EasyMock_WORKSPACE>java TestRunner
```

Verify the output.

```
true
```

```
Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js
```