**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Junit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Calculater.java**

public class Calculator{

public static int squared(int x)

{

return x \* x;

}

public static int divide(int divisor)

{

int result=0;

result=100/divisor;

return result;

}

}

**\*\*\*\*\*\*\*\*\*\*CalculatorUnitTest.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import org.junit.Assert;

import org.junit.Test;

import org.junit.runner.RunWith;

import org.junit.runners.Parameterized;

import java.util.Arrays;

import java.util.Collection;

@RunWith(Parameterized.class)

public class CalculatorUnitTest

{

@Parameterized.Parameters

public static Collection data()

{

return Arrays.asList(new Object[][] {{25,4}, {100,1}, {50, 2}, {0, 10},{5,20}});

}

private int input;

private int expected;

public CalculatorUnitTest(int input, int expected)

{

this.input = input;

this.expected = expected;

}

@Test(expected=ArithmeticException.class)

public void testMethod1()

{

System.out.println("Running parameterized tests");

Assert.assertEquals(expected, Calculator.divide(input));

}

@Test

public void testMethod2()

{

System.out.println("Running parameterized tests");

Assert.assertEquals(expected, Calculator.divide(input));

}

@Test

public void testMethod3()

{

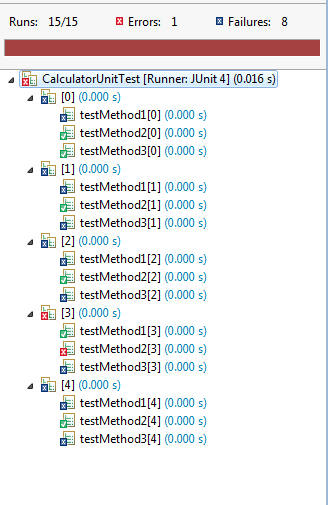
System.out.println("Running parameterized tests");

Assert.assertEquals("Test failed",625, Calculator.squared(input));

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*JUnit Output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Person.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

public class Person

{

private String firstName;

private String lastName;

public Person(String fname,String lname)

{

if(fname == null && lname==null){

throw new IllegalArgumentException("Both Names Cannot be NULL");

}

this.firstName=fname;

this.lastName = lname;

}

public String getFullName()

{

String first=(this.firstName != null)? this.firstName:"?";

String last=(this.lastName != null)? this.lastName:"?";

return first + " " + last;

}

public String getFirstName(){ return this.firstName;}

public String getLastName(){ return this.lastName;}

public static void main(String args[])

{

Person p=new Person("a","b");

System.*out*.println(p.getFirstName());

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestPerson.java\*\*\*\*\*\*\*\*\*\*\*\***

import static org.junit.Assert.\*;

import org.junit.\*;

public class TestPerson

{

@Test

public void GetFullName()

{

System.out.println("from TestPerson1");

Person per = new Person("Robert","King");

assertEquals("Robert King",per.getFullName());

}

@Test

public void NullsInName()

{

System.out.println("from TestPerson1");

Person per1 = new Person("Robert","King");

assertNotNull("full name null", per1.getFullName());

assertNotNull( "First name null",per1.getFirstName());

Person per2 = new Person("Robert","King");

assertEquals("Robert King", per2.getFullName());

}

@Test

public void GetFirstName()

{

Person p = new Person("Robert", "King");

assertEquals(p.getFirstName(), "Robert");

}

@Test

public void testGetLastName()

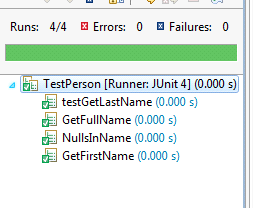
{

Person p = new Person("Robert", "King");

assertEquals(p.getLastName(), "King");

}}

**\*\*\*\*\*\*\*\*\*\*\*TestPerson.java Output\*\*\*\*\*\*\*\*\*\*\***



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestPerson2.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import org.junit.\*;

import org.junit.Test;

import static org.junit.Assert.\*;

public class TestPerson2

{

@Test

public void testGetFullName()

{

System.out.println("from TestPerson2");

Person per = new Person("Robert","King");

assertEquals("Robert King",per.getFullName());

}

@Test (expected=IllegalArgumentException.class)

public void testNullsInName()

{

System.out.println("from TestPerson2 testing exceptions");

Person per1 = new Person(null,null);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestPersonFixture.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import static org.junit.Assert.\*;

import org.junit.Test;

import org.junit.\*;

public class TestPersonFixture

{

@BeforeClass

public static void setBeforeAllTests()

{

System.out.println("One time initialization code before all tests");

}

@AfterClass

public static void doAfterAllTests()

{

System.out.println("One time cleanup code after all tests");

}

@Before

public void setbeforeTests()

{

System.out.println("Here is the initialization code");

}

@After

public void doafterTests()

{

System.out.println("Here is the cleanup code");

}

@Test

public void GetFullName()

{

System.out.println("from TestPerson1");

Person per = new Person("Robert","King");

assertEquals("Robert King",per.getFullName());

}

@Test

public void NullsInName()

{

System.out.println("from TestPerson1");

Person per1 = new Person("Robert","King");

assertNotNull("full name null", per1.getFullName());

assertNotNull( "First name null",per1.getFirstName());

Person per2 = new Person("Robert","King");

assertEquals("Robert King", per2.getFullName());

}

}

………………………………………………………………………………………….

One time initialization code before all tests

Here is the initialization code

from TestPerson1

Here is the cleanup code

Here is the initialization code

from TestPerson1

Here is the cleanup code

One time cleanup code after all tests

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestPersonSuit.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import org.junit.\*;

import org.junit.Test;

import org.junit.AfterClass;

import org.junit.BeforeClass;

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses({ TestPerson.class, TestPerson2.class,TestPersonFixture.class})

public class TestPersonSuite {

@BeforeClass

public static void setUpBeforeClass() throws Exception {

System.out.println("Now running the Test Suite");

}

@AfterClass

public static void tearDownAfterClass() throws Exception {

System.out.println("The Test Suite is completed");

}

}

**…………………………Output…………………………………………….**

Now running the Test Suite

from TestPerson1

from TestPerson1

from TestPerson2

from TestPerson2 testing exceptions

One time initialization code before all tests

Here is the initialization code

from TestPerson1

Here is the cleanup code

Here is the initialization code

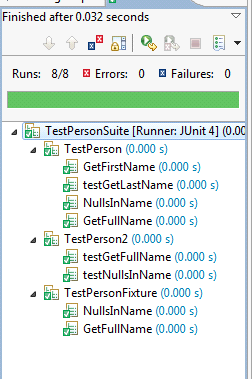
from TestPerson1

Here is the cleanup code

One time cleanup code after all tests

The Test Suite is completed

**…..ScreenShot………………………………………………………………………….**



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Log4J\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*In Project/log4j.properties\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

# Log levels

# Uncomment the following line to enable full loggin for every class

#log4j.rootLogger=trace, stdout, R

log4j.rootLogger=warn, R

#log4j.logger.pkg=trace, stdout, R

# Rolling File Appender

log4j.appender.R=org.apache.log4j.RollingFileAppender

# Path and file name to store the log file.

log4j.appender.R.File=applog.log

log4j.appender.R.MaxFileSize=500KB

# Keep one backup file

log4j.appender.R.MaxBackupIndex=1

# Rolling File Appender layout

log4j.appender.R.layout=org.apache.log4j.PatternLayout

log4j.appender.R.layout.ConversionPattern=%d - %c - %p - %m%n

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*log4j.xml\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

<appender name="appender" class="org.apache.log4j.FileAppender">

<param name="File" value="igatelog.log"/>

<param name="Append" value="true"/>

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%d [%t] %p - %m%n"/>

</layout>

</appender>

<root>

<priority value ="debug"/>

<appender-ref ref="appender"/>

</root>

</log4j:configuration>

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TestLoggerDemo.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import java.io.IOException;

import org.apache.log4j.FileAppender;

import org.apache.log4j.Logger;

import org.apache.log4j.SimpleLayout;

import org.apache.log4j.xml.XMLLayout;

public class TestLoggerDemo

{

public static void main(String[] args)

{

Logger mylogger=Logger.

getLogger("TestLoggerDemo");

//SimpleLayout simplelayout=new SimpleLayout();

XMLLayout xmllayout=new XMLLayout();

try

{

FileAppender fileappender=

new FileAppender(xmllayout,"igate.log",true);

mylogger.addAppender(fileappender);

mylogger.debug("This is Debug Message");

mylogger.info("This is log message ");

mylogger.warn("This is warning message");

mylogger.error("This is error level message");

mylogger.fatal("This is fatal error message");

System.out.println("This is my logging programe");

int numA[]=new int[1];

numA[0]=67;

System.out.println(" numA[0] : ="+numA[1]);;

}

catch (Exception e)

{

System.out.println(" Arrray Exception : "+

e.getMessage());

mylogger.error(e.getMessage());

e.printStackTrace();

} }

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TestLoggerUsingPropertyFile.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import org.apache.log4j.Logger;

import org.apache.log4j.PropertyConfigurator;

public class TestLoggerUsingPropertyFile

{

public static void main(String[] args)

{

PropertyConfigurator.configure("log4j.properties");

Logger mylogger=Logger.getLogger("TestLoggerUsingPropertyFile");

mylogger.debug("This is debug level message ");

mylogger.info("This is info level message ");

mylogger.warn("This is warn level message ");

mylogger.error("This is error level message ");mylogger.debug("This is debug level message ");

mylogger.fatal("This is fatal level message ");

System.out.println("Done........");

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*** **TestLoggerUsingXmlDeml.java\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import org.apache.log4j.Logger;

import org.apache.log4j.xml.DOMConfigurator;

public class TestLoggerUsingXmlDeml {

public static void main(String[] args)

{

DOMConfigurator.configure("log4j.xml");

Logger mylogger=Logger.getLogger("TestLoggerUsingPropertyFile");

mylogger.debug("This is debug level message ");

mylogger.info("This is info level message ");

mylogger.warn("This is warn level message ");

mylogger.error("This is error level message ");mylogger.debug("This is debug level message ");

mylogger.fatal("This is fatal level message ");

System.out.println("Done........");

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Jdbc Connection Test Cases\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.mobilerecharge.test;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Properties;

import org.junit.Test;

//import com.igate.mobile.util.DBConnection;

public class DBConnectiontest

{

@Test

public void getConnection()

{

InputStream propsFile;

Properties tempProp = new Properties();

Connection conn=null;

try {

propsFile = new FileInputStream("resources/jdbc.properties");

tempProp.load(propsFile);

propsFile.close();

//tempProp.list(System.out); ///to print the data in property file

String url=tempProp.getProperty("jdbc.url");

String userName=tempProp.getProperty("jdbc.username");

String password=tempProp.getProperty("jdbc.password");

/\*OracleDataSource dataSource=new OracleDataSource();

dataSource.setURL(url);

dataSource.setUser(userName);

dataSource.setPassword(password);

conn=dataSource.getConnection();

\*/

conn = DriverManager.getConnection(url, userName,password);

}

catch (IOException exception)

{

System.err.println("Enter the proper file"+exception.getMessage());

}

catch (SQLException exception)

{

System.err.println("Check the Login Credentials"+exception.getMessage());

}

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Mock API Testing\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*User.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.capgemini.lesson14.mock;

public class User {

String username, password;

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LoginService.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.capgemini.lesson14.mock;

public interface LoginService {

boolean login(String username, String password);

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LoginServiceImpl.java\*\*\*\*\*\*\*\*\***

package com.capgemini.lesson14.mock;

public class LoginServiceImpl implements LoginService {

private UserDAO userDao;

public void setUserDao(UserDAO userDao){

this.userDao = userDao;

}

public boolean login(String username, String password) {

boolean valid=false;

try{

User user = userDao.loadByUserNameAndPassword(username, password);

if(user != null) valid = true;

}

catch(Exception e){}

return valid;

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*UserDao.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.capgemini.lesson14.mock;

public interface UserDAO {

User loadByUserNameAndPassword(String username, String password);

}

**\*\*\*\*\*\*\*\*\*\*\*\*LoginTest.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.capgemini.lesson14.mock;

import static org.junit.Assert.assertTrue;

import org.easymock.EasyMock;

import org.junit.Before;

import org.junit.Test;

public class LoginTest {

private LoginServiceImpl service;

private UserDAO mockDao;

@Before

public void setup() {

service = new LoginServiceImpl();

mockDao = EasyMock.createMock(UserDAO.class);

service.setUserDao(mockDao);

}

@Test

public void testLogin(){

User user = new User();

String username="testusername";

String password = "testpassword";

/\* expect tells your mock object to expect the method \*/

EasyMock.expect

(mockDao.loadByUserNameAndPassword(username, password))

.andReturn(user);

//tells mock object what to return after this method is called

/\* replay() tells EasyMock : "we are done declaring

\* our expectations. Its time to run what we told you" \*/

EasyMock.replay(mockDao);

/\* assertTrue() does two things:

\* 1. executes the code to be tested

\* 2. Tests that the result is true

\*/

assertTrue(service.login(username, password));

/\* verify() : tells EasyMock to validate that

\* all expected method calls were executed and in the correct order \*/

EasyMock.verify(mockDao);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Thank You\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***