1. Creating jar file and using that jar file in selenium project?

**To export a project to a JAR file**

1.    Start Eclipse and navigate to the project

2.     Right-click on the same project and select **Build path and click configure build path**

3.    Click on libraries and select add external jar files to the project

2. create exception handling class into this project

Your Selenium test should be able to fail, but not because of exceptions that are thrown. If your test is failing from exceptions then quite likely you have no exception handling. By doing this, you don’t have the opportunity to cleanup the WebDriver object at the end of the test.

The tests should be failing under your terms only for example, you should never be getting exceptions like NullPointerException but if you are getting such as ElementNotFoundException, then also it is good idea to catch the exception, stop the further execution and end your test in a Logical way.

**Example 1:** I do not use any Page Object Factory but I use my own Page Object Pattern and I always print error logs and take screenshot on any exception I encounter. Please look at the code below:

public static WebElement btn\_ReportCategory(WebDriver driver) throws Exception{

        try{

            WebElement element = driver.findElement(By.linkText("+ Report Categories"));

        }catch (Exception e){

// Printing logs for my report

            Log.error("Report Category button element is not found.");

// Taking screenshot for defect reporting

Utils.captureScreenShot();

// After doing my work, now i want to stop my test case

            throw(e);

        }

// This will return the Element in case of no Exception

        return element;

    }

3. create Reading data base classes

**1) Make a connection to the Database**

In order to make a connection to the database the syntax is

DriverManager.getConnection(URL, "userid", "password" )

Here,

* Userid is the username configured in the database
* Password of the configured user
* URL is of format jdbc:< dbtype>://ipaddress:portnumber/db\_name"
* <dbtype>- The driver for the database you are trying to connect. To connect to oracle database this value will be "oracle"

For connecting to database with name "emp" in MYSQL URL will bejdbc:mysql://localhost:3036/emp

And the code to create connection looks like

Connection con = DriverManager.getConnection(dbUrl,username,password);

You also need to load the JDBC Driver using the code

Class.forName("com.mysql.jdbc.Driver");

**2) Send Queries to the Database**

Once connection is made, you need to execute queries.

You can use the Statement Object to send queries.

Statement stmt = con.createStatement();

Once the statement object is created use the executeQuery method to execute the SQL queries

stmt.executeQuery(select \* from employee;);

### 3) Process the results

Results from the executed query are stored in the ResultSet Object.

Package htmldriver;

import java.sql.Connection;

import java.sql.Statement;

import java.sql.ResultSet;

import java.sql.DriverManager;

import java.sql.SQLException;

public class SQLConnector {

public static void main(String[] args) throws ClassNotFoundException, SQLException {

//Connection URL Syntax: "jdbc:mysql://ipaddress:portnumber/db\_name"

String dbUrl = "jdbc:mysql://localhost:3036/emp";

//Database Username

String username = "root";

//Database Password

String password = "\*\*\*";

//Query to Execute

String query = "select \* from employee;";

//Load mysql jdbc driver

Class.forName("com.mysql.jdbc.Driver");

//Create Connection to DB

Connection con = DriverManager.getConnection(dbUrl,username,password);

//Create Statement Object

Statement stmt = con.createStatement();

// Execute the SQL Query. Store results in ResultSet

ResultSet rs= stmt.executeQuery(query);

// While Loop to iterate through all data and print results

while (rs.next()){

String myName = rs.getString(1);

String myAge = rs.getString(2);

System. out.println(myName+" "+myAge);

}

// closing DB Connection

con.close();

}

}

**4. TESTNG Framework**

package framework;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

//In TestNG @test methods will execute in alphabetical order

public class TestAnootations

{

@BeforeMethod

//BeforeMethod will execute before every @test method

public void useridGeneration()

{

System.out.println("userid generation");

}

@Test

public void openingBrowser()

{

System.out.println("opening browser");

}

@Test

public void bookingFlight()

{

System.out.println("booking flight");

}

@BeforeTest

//BeforeTest will execute only once and it will execute first

public void deleteCookies()

{

System.out.println("delete the cookies");

}

@AfterTest

//AfterTest will execute only once after completing the test

public void cookiesClose()

{

System.out.println("delete the cookies after finishing");

}

5. JUNIT FRAMEWORK

Basically JUnit is an open source unit testing tool and used to test small/large units of code. To run the JUnit test you don’t have to create class object or define main method. [JUnit](http://junit.org/" \o "JUnit) provide assertion library which is used to evaluate the test result. Annotations of JUnit are used to run the test method. JUnit is also used to run the Automation suite having multiple test cases.

package junitpack;

import org.junit.Test;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class junittest1 {

WebDriver driver = new FirefoxDriver();

@Test

public void test() throws InterruptedException {

driver.manage().window().maximize();

driver.get("http://only-testing-blog.blogspot.in/2013/11/new-test.html");

driver.findElement(By.xpath("//input[@name='fname']")).sendKeys("junittest1 executed");

Thread.sleep(2000);

System.out.print("junittest1 class is executed");

driver.quit();

}

}

}