**How do I explain Java-Selenium automation project**

Automation testing or Test Automation is a process of automating the manual process to test the application/system under test. Automation testing involves use to a separate testing tool which lets you create test scripts which can be executed repeatedly and doesn’t require any manual intervention.

**Benefits of Automation Testing:-**

1. Supports execution of repeated test cases

2. Aids in testing a large test matrix

3. Enables parallel execution

4. Encourages unattended execution

5. Improves accuracy thereby reducing human-generated errors

6. Saves time and money

When we give any answer the one and only important thing we need to keep in mind is ***It should be brief and to the point***so that should not loose interest from our answer.

So whenever we need to give answer about framework we can divide framework in terms of different components which we are using to build a structure which we called a framework and which we use in our company.

We can start like mentioned below.

1. We are using ***Page Object with Page Factory framework with functional/structural implementation***.  
   *>> Make sure you know what is****functional/structural implementation in this framework.***
2. We have maintained a page class for every page in our application and a page test class to maintain test for that pages. e.g. Product listing page, Add to cart page, Payment page, Invoice generation page.   
   *>> Make sure you know we maintain different page and page test class and all different annotations in page factory*
3. We have maintained **separate package for page and page test** e.g.  
   com.companyName.page1

com.companyName.pageTest1  
*>> Maintaining different packages is always a good practice to follow.*

1. We also have a **base page class** for common functions use by all pages.  
   *>> Make sure you know why we have Base Page class in page object*
2. We also have **library package** to maintain common functions related to Selenium/waits/directory creations etc
3. We are using **JAVA**/Ruby as our **binding language**  
   *>> We use java because it is known to most people when we started automation.*
4. We are using standardized maven project for **build**, **execution** & **dependency management.***>> Make sure you know about a* ***build tool*** *like ant/maven*
5. For handling **data driven cases** we are passing data using java **properties file**/**xls file** /csv file.  
   *>> Make sure you* ***know about libraries*** *like open CSV,JXL/****APACHE POI****/****Java Properties class***
6. For **ordering tests** we are using **testng framework.** *>> Prepare testng questions and make sure you know how to* ***run testng.xml using maven***
7. We are using **log4j library to maintain logging of our project**. We are using all kinds of **logging statements** like INFO,DEBUG,ERROR etc. We have maintained a separate class for it in com.companyName.main package  
   *>> Make sure you know this library usage in java, we can use log4j by mentioning properties of this framework in a xml file or a properties file and putting that file on build path.*
8. We are using **Extent Report for reporting purpose**. It is a third party report and it is easily available at maven central repo.
9. We are using maven postman plugin / JAVA API to send generated extent reports as an attachment to client Distribution list.  
   *>> Make sure you know about this plugin of maven or Java API*
10. We checkin our code into **client repository using** a version controlling tool git bash on windows sytem.

Summary :- We should talk about all major components in our project like Logging, Emailing, Page Objects, Page Factory Annotations, TestNG, Exception Handling, Build tool, Version controlling tool, Data Driven usage etc.

1. Explain the rationale behind choosing a tool/framework, it shouldn't just be, I have chosen coz I know only this tool or framework.
2. How you are **organizing the test cases**- page factory model, page object model, or dimply unorganized as requirement might be very simple.
3. **Test case execution method**- data driven, keyword driven or hybrid- basically how a driver script is written- also if selecting few test cases for execution is possible or every time you need to execute full suite
4. How screenshots or execution proof are captured and stored- how detailed it is.
5. Automated defect logging capabilities- integration to defect management tool, **automatic logging definitions with screenshots.**
6. **Scheduled runs**- whether test cases can be executed in a daily basis or on scheduling using something like **windows scheduler**
7. **Continuous integration**- whether tools like maven or ant is used for CI? How test can be integrated to build or a release and **executed automatically**
8. **Repo and code management**- tools used for script development, repositories like github, mercurial or git lab.

[Alex Siminiuc](https://www.quora.com/profile/Alex-Siminiuc-2), enjoys writing about Selenium test automation and labradors

Let's start with the beginning.

You have a set of functional test cases that are executed manually.

The test cases verify functionality of a website.

Each test case takes the tester to the web page to-be-tested through a few other web pages.

On the page to-be-tested, the test case verifies if an expected result is there or not.

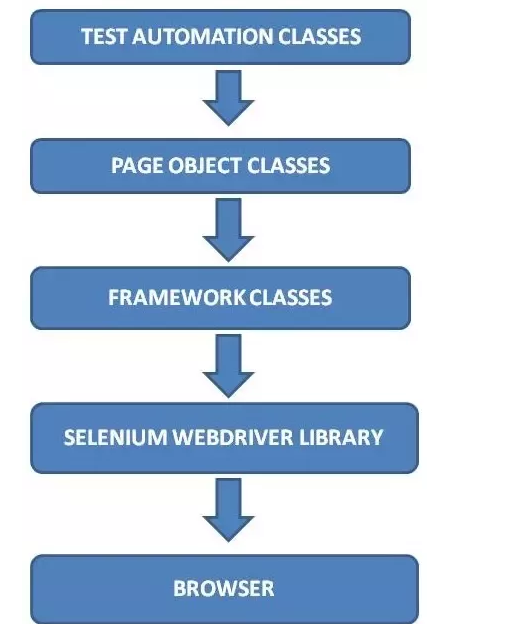
**Your goal is to automate all test cases.**

How do you do it?

First, you create a project using an IDE like Eclipse and add 2 libraries to it:

* **Selenium Web Driver library**; you need it so that the automation code can interact with the site through the browser
* **a**[unit testing library such as TestNG](http://test-able.blogspot.com/2016/03/9-easy-testng-features-you-can-learn.html); this is useful for **creating test automation scripts** that are independent

The project uses a few different layers

.

**Each layer talks only to its previous and next layers.**

The top 3 layers use packages for organizing their classes:

* test classes package
* page object classes package
* framework classes package

**Create a**[**page object**](http://test-able.blogspot.com/2016/10/what-is-a-page-object.html)**class for each web page mentioned in a test case.** The page object class **implements the user interaction with the specific web page**.

**The page object class has fields and methods**.

The page object fields are used for

* the web page title
* the web page url
* the locators of the web page elements
* any other information that is specific to the web page.

The methods are used to

* **provide information about the web page and its elements**
  + page title
  + page url
  + value of a label
  + if an element is enabled or selected
* **do different things on the web page**
  + click elements
  + type text into textboxes
  + select options of list boxes

The page object classes are used by the test scripts.

The page object classes use classes from the project's framework to implement the interaction with the web page.

You create a test script for each test case.

**Each test script uses the @Test annotation to be considered a unit test.**

The test script has 2 parts:

1. navigate to the page that is being tested

2. execute verifications on the tested page

The **test script uses page object classes to navigate to the page** that is being tested.

Once on the to-be-tested page, the test script uses assertions for the verifications it has to do.

The test scripts are grouped in test classes.

**Each test class groups test scripts for a specific purpose.**

A test class includes not only test scripts but also **test fixtures**.

Test fixtures are special methods of the unit testing framework that

1. **prepare the test environment** that the test scripts need
2. **clean the test environment** after the test script runs

Multiple test classes can be organized using groups and suites.

It is a bad idea to duplicate test fixtures in multiple test classes.

The test fixtures can be moved from the test class to a test base class.

Each test class inherits from the test base class to have access to the test fixtures.

The project needs **framework classes** for common purpose functionality such as

* creating a browser driver for any type of browser
* [interacting with the site using explicit waits](http://test-able.blogspot.com/2016/04/beginners-guide-to-explicit-waits.html)
* logging information to a file
* reading settings info from a text file
* taking screenshots on exceptions; since TestNG is the unit testing framework, a listener can be used
* sending an email with the test results
* creating a status report

The **framework classes are used by the page object classes.**

I have written something about the same.. Here I go..   
We need to specify in and out of our Test Automation Framework such as programming **language** used, **Type of framework** used, Test Base Class (Initializing WebDriver, Implicit Waits), How we **separate Element locators and tests** (Page Objects, Page Factory), Utility functions file, Property files, TestNG annotations, How we parameterize tests using Excel files, How we capture error screenshots, Generating reports(Extent Reports), Emailing reports, Version Control System used and Continues Integration Tool used.

**Language:**In our Selenium Project we are using Java language. Even though Selenium supports multiple languages, we are using Java language is just because most of the automation developers have knowledge on Selenium with Java.

**Type of Framework:** In our project, we are **using Data-driven Framework by using Page Object Model design pattern** with Page Factory.

**POM:**As per the Page Object Model, we have maintained a class for every web page. Each web page has a separate class and that class holds the functionality and members of that web page. Separate classes for every individual test.

**Packages:** We have separate packages for *Pages* and *Tests*. All the *web page* related classes come under **Pages** package and all the *tests* related classes come under **Tests** package.

For example, *Home Page* and *Login Page* have a separate classes to store element locators. For the *login test* there would be a separate class which calls the methods from the *Home Page* class and *Login Page* class.

I will explain based on the below mentioned test automation framework structure.

Above screenshot illustrates standardized maven project. As per the above maven project, all the tests are kept in the ‘***src/test/java****‘*and remaining files (such as config.properties, element locators (POM classes), utility files, test data, etc.,) kept under ‘***src/main/java****‘*.

**Test Base Class:**Test Base class (TestBase.java) deals with [Read More…](http://www.softwaretestingmaterial.com/explain-test-automation-framework/)

Already have good answers to this question. But what I find them as more of framework related.

Firstly you can tell the person about the selenium, like what it is?

Secondly, programming languages support, and why you chose Java.

Third, what classes you have created and the purpose of those.

Fourth, you must have used test NG or jUnit, let him know that. Explain in detail why do we need to use it. Discuss and focus on grouping of test cases, parallel execution.

Lastly you can tell the format on which you are getting reports, was any third party tools included? Mailer used?

One more thing, also let him know if you have **used design pattern that is pom**, if yes, why and benefits.

All the best!!

Originally Answered: [How do I explain framework in Selenium WebDriver to a interviewer?](https://www.quora.com/How-do-I-explain-framework-in-Selenium-WebDriver-to-a-interviewer?no_redirect=1)

Answering your framework shows your level of experience in automation, you need to clearly explain the following based on your framework.

1. First you need to answer why you selected this framework
2. How you are maintaining your test suites /cases
3. Passing of test data to your test suite/test cases
4. Object repository
5. Test case execution
6. Library folders
7. Log files generation
8. Test report generation
9. Configuration files
10. Int cross verifies you by asking versions you are using ,we should be ready to answer it.

<artifactId>selenium-java</artifactId>

<version>3.14.0</version> <artifactId>selenium-java</artifactId>

<version>3.14.0</version>

<artifactId>extentreports</artifactId>

<version>2.41.2</version>

<artifactId>testng</artifactId>

<version>6.14.2</version>

Start with the below mentioned topics,

* Project overview
* Advantages of automating your project
* Framework used
* DB validations if any
* Version control tool used
* Project management tool used
* Integration tool used
* Format in which reports will generated

[Apoorv Sharma](https://www.quora.com/profile/Apoorv-Sharma-8), Software engineer, 5+ years exp in automation testing

You can start with explaining about the framework which you have used. You can explain about how you manage your test data and how you generate reports from your scripts. Then the interviewer would probably start asking questions on specifics like how did you handle window pop or multiple frames in selenium etc. So in this way you will get chance to explain your project with more details.

# Data Driven Framework with Apache POI – Excel

Most commercial automated software tools on the market support some sort of data driven testing, which allows you to automatically run a test case multiple times with different input and validation values. As Selenium Webdriver is more an automated testing framework than a ready-to-use tool, you will have to put in some effort to support data driven testing in your automated tests. I usually prefer to use Microsoft Excel as the format for storing my parameters. An additional advantage of using Excel is that you can easily outsource the test data administration to someone other than yourself, someone who might have better knowledge of the test cases that need to be run and the parameters required to execute them.

## Reading data from the Excel

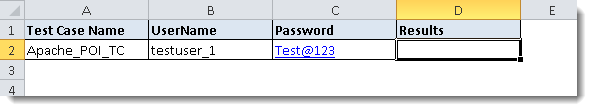
We need a way to open this Excel sheet and read data from it within our Selenium test script. For this purpose, I use the **Apache POI library**, which allows you to read, create and edit Microsoft Office-documents using Java. **The classes and methods we are going to use to read data from Excel sheet are located in the org.apache.poi.hssf.usermodel package.**

## How to do it…

1) [Download JAR files](http://toolsqa.wpengine.com/selenium-webdriver/download-apache-poi/) of Apache POI  and [Add Jars](http://toolsqa.wpengine.com/selenium-webdriver/add-apache-poi-jars/) to your project library. You can download it from [here](http://poi.apache.org/). That’s all about configuration of Apache POI with eclipse. Now you are ready to write your test.

2) Create a ‘[**New Package**](http://toolsqa.wpengine.com/selenium-webdriver/configure-eclipse-with-selenium-webdriver/#package)‘ file and name it as ‘**testData’**, by right click on the Project and select **New** > **Package**. Place all of your test data in this folder (package) whether it is a sql file, excel file or anything.

3) Place a **Excel** file in the above created package location and save it as **TestData.xlsx**. Fill the data in the excel like below image:



4) Add two constant variables (testData package path & Excel file name) in the **Constant** class.

|  |  |
| --- | --- |
|  | package utility;      public class Constant {         public static final String URL = "http://www.store.demoqa.com";         public static final String Username = "testuser\_1";         public static final String Password = "Test@123";         public static final String Path\_TestData = "D://ToolsQA//OnlineStore//src//testData//"         public static final String File\_TestData = "TestData.xlsx"      } |

5) Create a ‘[**New Class**](http://toolsqa.wpengine.com/selenium-webdriver/configure-eclipse-with-selenium-webdriver/#Class)‘ file, by right click on the ‘**utility**‘ Package and select **New** > **Class**and name it as ‘**ExcelUtils**‘**.** First we will write basic read/write methods.

|  |  |
| --- | --- |
| 1  37  98  99  100  101  102  103  104  105  106  107  108  109 | package utility;         import java.io.FileInputStream;               import java.io.FileOutputStream;               import org.apache.poi.xssf.usermodel.XSSFCell;            import org.apache.poi.xssf.usermodel.XSSFRow;            import org.apache.poi.xssf.usermodel.XSSFSheet;            import org.apache.poi.xssf.usermodel.XSSFWorkbook;       public class ExcelUtils {           private static XSSFSheet ExcelWSheet;            private static XSSFWorkbook ExcelWBook;            private static XSSFCell Cell;            private static XSSFRow Row;        //This method is to set the File path and to open the Excel file, Pass Excel Path and Sheetname as Arguments to this method//        public static void setExcelFile(String Path,String SheetName) throws Exception {          try {              // Open the Excel file    FileInputStream ExcelFile = new FileInputStream(Path);    // Access the required test data sheet    ExcelWBook = new XSSFWorkbook(ExcelFile);    ExcelWSheet = ExcelWBook.getSheet(SheetName);    } catch (Exception e){    throw (e);  }  }       //This method is to read the test data from the Excel cell, in this we are passing parameters as Row num and Col num          public static String getCellData(int RowNum, int ColNum) throws Exception{         try{             Cell = ExcelWSheet.getRow(RowNum).getCell(ColNum);              String CellData = Cell.getStringCellValue();              return CellData;              }catch (Exception e){    return"";              }        }        //This method is to write in the Excel cell, Row num and Col num are the parameters        public static void setCellData(String Result,  int RowNum, int ColNum) throws Exception {          try{              Row  = ExcelWSheet.getRow(RowNum);    Cell = Row.getCell(ColNum, Row.RETURN\_BLANK\_AS\_NULL);    if (Cell == null) {    Cell = Row.createCell(ColNum);    Cell.setCellValue(Result);    } else {    Cell.setCellValue(Result);    }             // Constant variables Test Data path and Test Data file name              FileOutputStream fileOut = new FileOutputStream(Constant.Path\_TestData + Constant.File\_TestData);              ExcelWBook.write(fileOut);              fileOut.flush();    fileOut.close();    }catch(Exception e){    throw (e);    }    }    } |

6) Once we are done with writing Excel functions we can go ahead and modify the **SignIn\_Action** module to accept the test data from excel file.

|  |  |
| --- | --- |
|  | package appModules;           import org.openqa.selenium.WebDriver;           import pageObjects.Home\_Page;           import pageObjects.LogIn\_Page;           import utility.ExcelUtils;       // Now this method does not need any arguments       public class SignIn\_Action {    public static void Execute(WebDriver driver) throws Exception{    //This is to get the values from Excel sheet, passing parameters (Row num &amp; Col num)to getCellData method    String sUserName = ExcelUtils.getCellData(1, 1);    String sPassword = ExcelUtils.getCellData(1, 2);    Home\_Page.lnk\_MyAccount(driver).click();    LogIn\_Page.txtbx\_UserName(driver).sendKeys(sUserName);    LogIn\_Page.txtbx\_Password(driver).sendKeys(sPassword);    LogIn\_Page.btn\_LogIn(driver).click();            }   } |

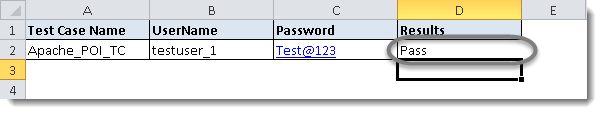
**Note:**In the later chapters we will see how to parameterise the row column as well, as we also have to avoid hard coded values in the scripts. This is just to give you an idea to use Excel and we will move forward step by step towards proper framework.

7) Create a ‘[**New Class**](http://toolsqa.wpengine.com/selenium-webdriver/configure-eclipse-with-selenium-webdriver/#class)‘ and name it as **Apache\_POI\_TC** by right click on the ‘**automationFramework**‘ Package and select **New** > **Class**. In this we will read the values from the Excel sheet to use them as the test data and write the test result in the Excel.

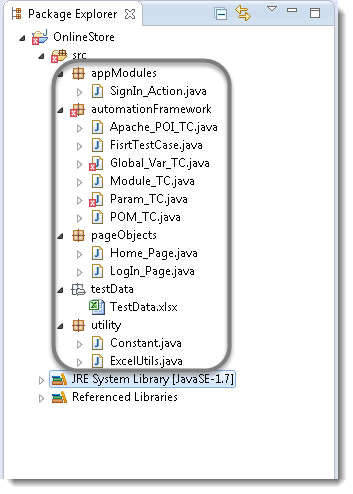
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | package automationFramework;    import java.util.concurrent.TimeUnit;  import org.openqa.selenium.WebDriver;    import org.openqa.selenium.firefox.FirefoxDriver;    import pageObjects.\*;    import utility.Constant;    // Import Package utility.\*    import utility.ExcelUtils;    import appModules.SignIn\_Action;    public class Apache\_POI\_TC {    private static WebDriver driver = null;    public static void main(String[] args) throws Exception {           //This is to open the Excel file. Excel path, file name and the sheet name are parameters to this method           ExcelUtils.setExcelFile(Constant.Path\_TestData + Constant.File\_TestData,"Sheet1");           driver = new FirefoxDriver();           driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);           driver.get(Constant.URL);           SignIn\_Action.Execute(driver);           System.out.println("Login Successfully, now it is the time to Log Off buddy.");           Home\_Page.lnk\_LogOut(driver).click();           driver.quit();           //This is to send the PASS value to the Excel sheet in the result column.           ExcelUtils.setCellData("Pass", 1, 3);    }    } |

Give it a run, see how beautify your script will execute the code.

8) Once it finished open the Excel file and check for the result.



Your Project explorer window will look like this now.



Do not worry about the red cross signs on your test scripts, it is because we have removed arguments from SignIn\_Action class.

**Data Driven framework can be build up by using**[**TestNg Data Provider**](http://toolsqa.wpengine.com/selenium-webdriver/testng-parameters-data-provider/)**.**