

TABLE OF CONTENTS

1. [Overview](#)
2. [Introduction to Falcon UI Framework](#)
3. [Environment Requirements](#)
4. [Setup Falcon UI Project](#)
5. [Understanding Project Structure](#)
6. [Understanding config. properties](#)
7. [Creating and Running Test Scripts](#)
8. Logging

Overview

This document provides instructions for getting started with Falcon UI Framework.

This document covers following topics:

1. [Introduction to Falcon UI Framework](#)
2. [Environment Requirements](#)
3. [Setup Falcon UI Project](#)
4. [Understanding Project Structure](#)
5. [Understanding config. properties](#)

Introduction to Falcon UI Framework:

The Falcon UI Automation Framework is a system of software tools used for running repeatable functional tests against a web application being tested.

This framework can be used for automating most routine processes in web applications automation, such as:

1. Cross Browser Testing by running in Local, Grid and Cloud (Browserstack)
2. Verification of different data types (Primitive and Non-Primitive)
3. Utilities to read and write with different file formats (*.xls, *.xlsx, *.txt, *.csv, *.xml)

Falcon UI Automation Framework uses the following tools and frameworks:

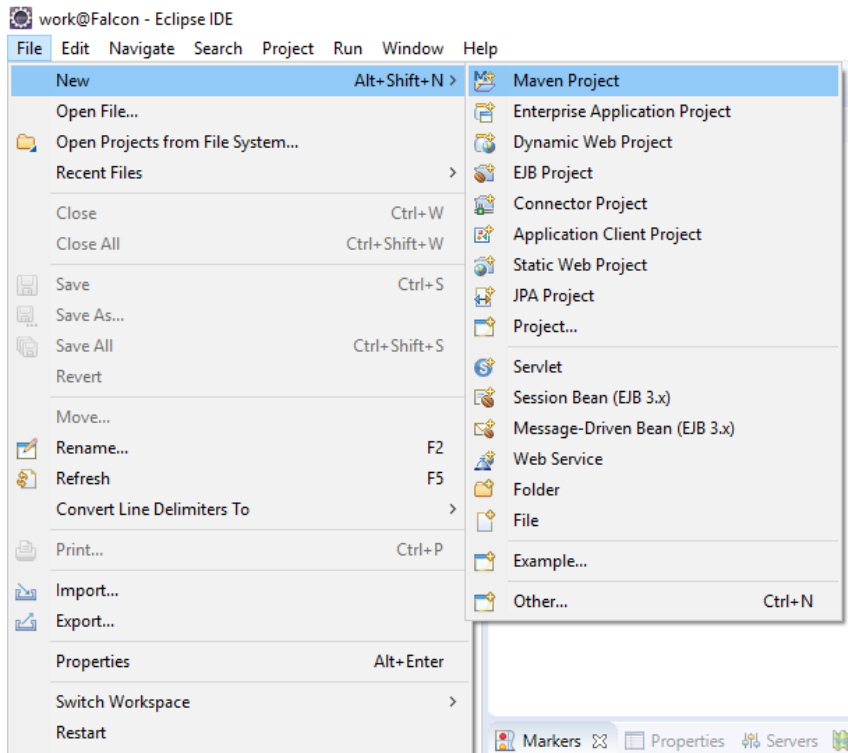
1. Java – A Programming Language
2. TestNG – A Unit Framework
3. Maven – A Build Management Tool
4. Selenium – An Automation Tool

Environment Requirements:

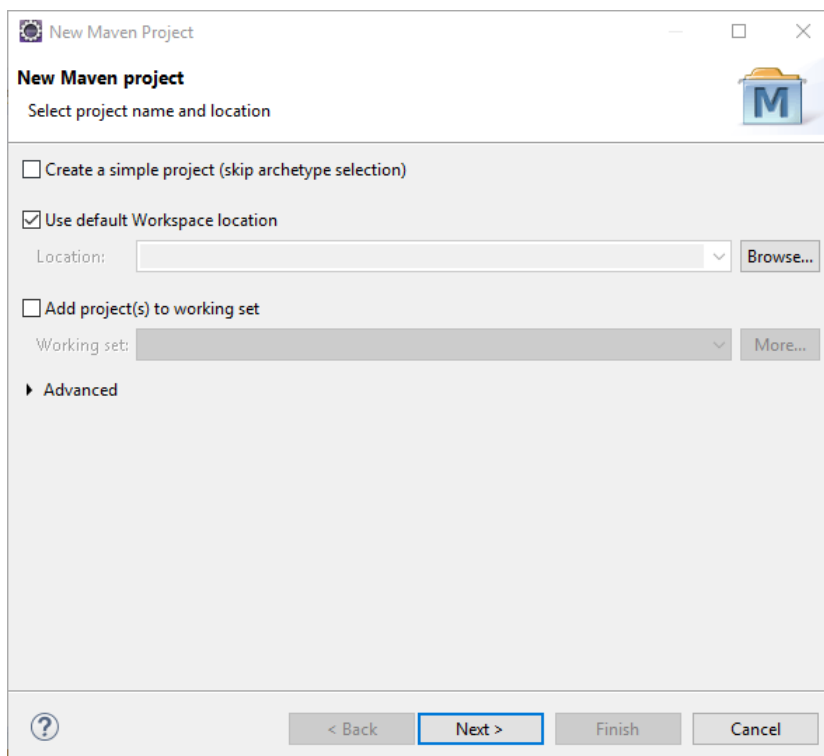
1. Configure Java and Maven
 - a. Follow the steps mentioned in the below link:
<https://www.mkyong.com/maven/how-to-install-maven-in-windows/>
2. Eclipse IDE
 - a. Follow the steps mentioned in the below link:
<http://www.automationtestinghub.com/eclipse-ide-download-and-install/>
3. TestNG
 - a. Follow the steps mentioned in the below link:
<http://toolsqa.com/selenium-webdriver/install-testng/>

Setup Falcon UI Project:

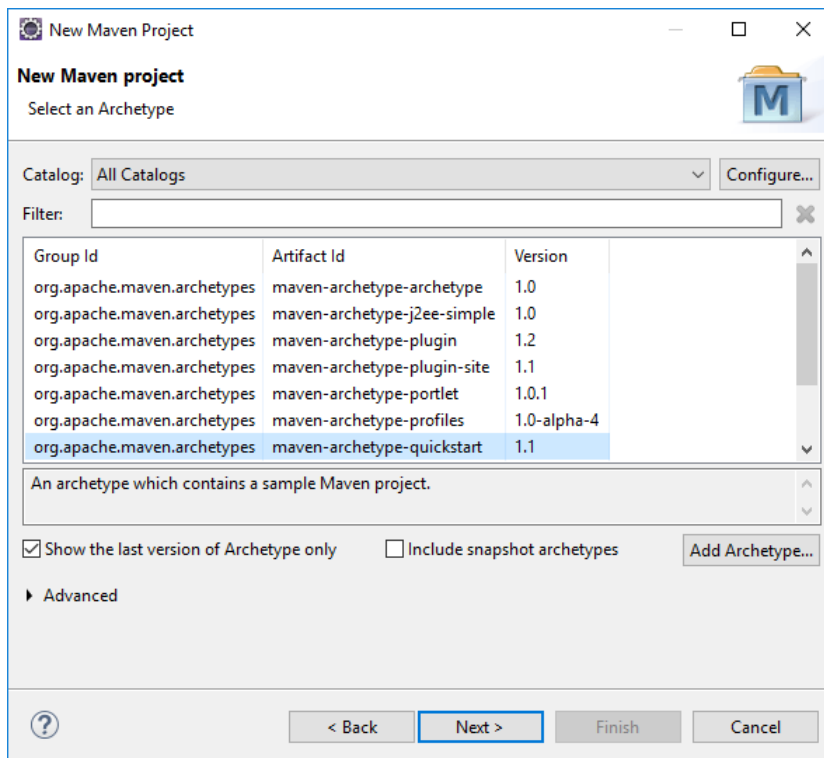
1. Click on **File** Menu and Select **New** Sub Menu and then Click on **Maven Project**



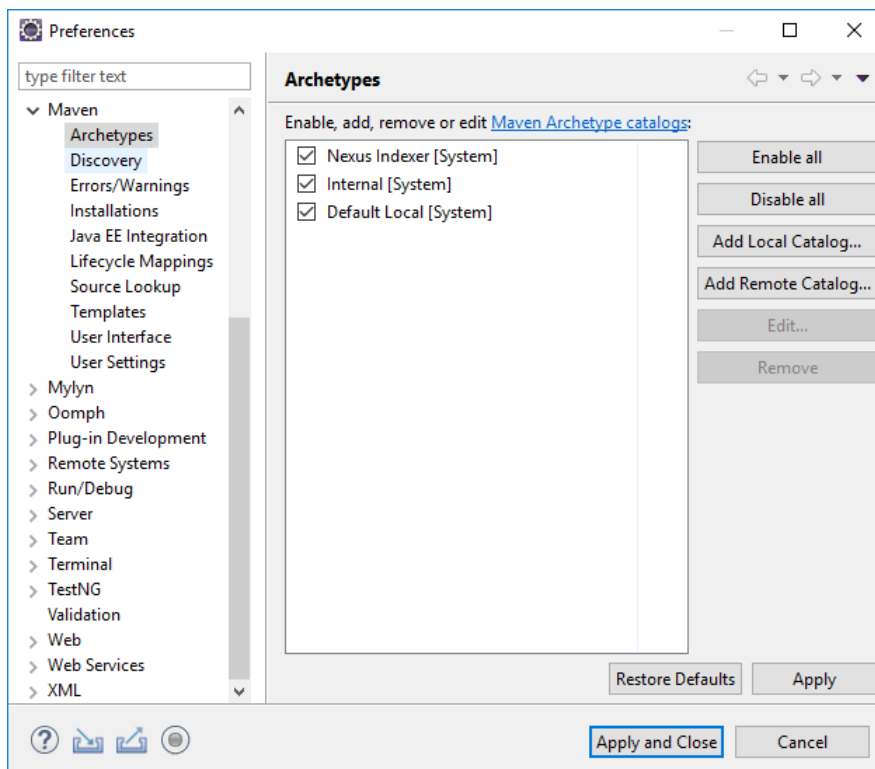
2. Click **Next** under **New Maven Project** window



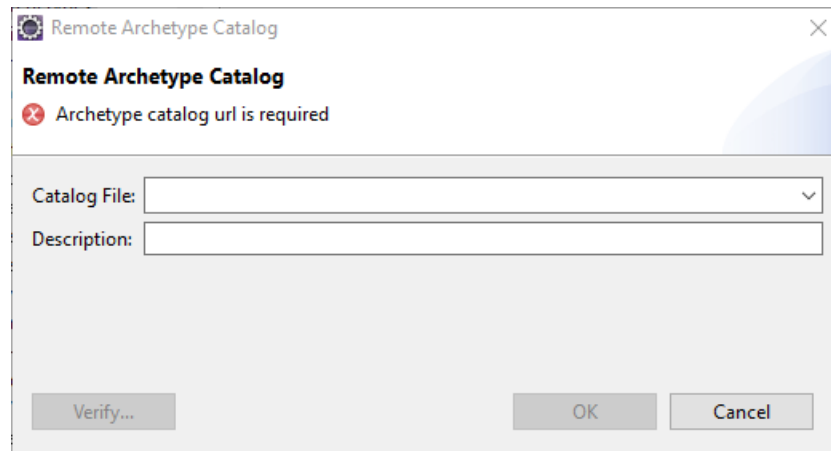
3. Click on **Configure** under **New Maven Project** to add Falcon UI CatLog



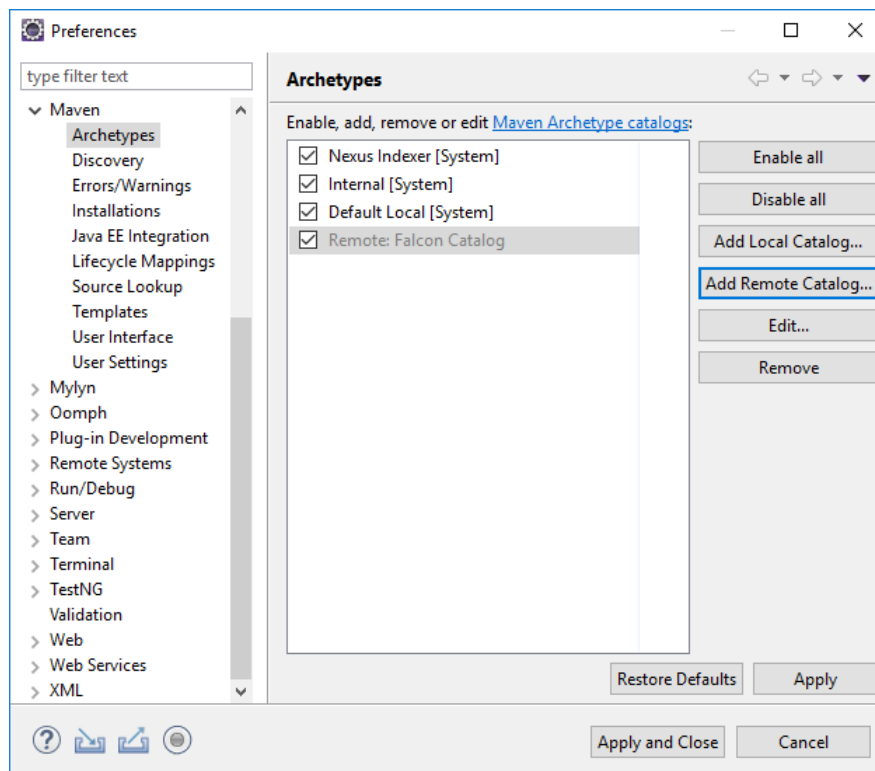
4. Click on **Add Remote CatLog** under **Archetypes** window



5. Enter below details under **Remote Archetype CatLog** window and then Click **OK**
- CatLog File** as
<http://10.10.10.150:8081/nexus/service/local/repositories/falcon/content/archetype-catalog.xml>
 - Description** as any name to identify the CatLog say **Falcon CatLog**



6. Click **Apply and Close** under **Archetypes** window



7. Select **falcon-ui-archetype** as **Artifact Id** and then Click **Next** under **New Maven Project** window

The screenshot shows the 'New Maven Project' dialog box with the title 'New Maven project' and subtitle 'Select an Archetype'. The 'Catalog' is set to 'Falcon Catalog'. A table lists available archetypes, with 'falcon-ui-archetype' selected. The 'Filter' field is empty. The 'Advanced' section is collapsed. The 'Next >' button is highlighted.

Group Id	Artifact Id	Version
com.atmecs.falcon	FalconMobile-archetype	3.0
com.atmecs.falcon	FalconMobileArchetype	2.0
com.atmecs.falcon	falcon-mobile-archetype	1.0
com.atmecs.falcon	falcon-rest-archetype	1.0
com.atmecs.falcon.archetype	falcon-ui-archetype	0.2.0

http://10.10.10.150:8081/nexus/content/repositories/falcon

☒ Show the last version of Archetype only ☐ Include snapshot archetypes Add Archetype...

Advanced

< Back Next > Finish Cancel

8. Enter details as below and Click **Finish** under **New Maven Project** window

The screenshot shows the 'New Maven Project' dialog box with the title 'New Maven project' and subtitle 'Specify Archetype parameters'. The 'Group Id' is 'com.qa.automation', 'Artifact Id' is 'falconUI', 'Version' is '0.0.1-SNAPSHOT', and 'Package' is 'com.qa.automation.falconUI'. The 'Properties available from archetype' table is empty. The 'Advanced' section is collapsed. The 'Finish' button is highlighted.

Group Id: com.qa.automation

Artifact Id: falconUI

Version: 0.0.1-SNAPSHOT

Package: com.qa.automation.falconUI

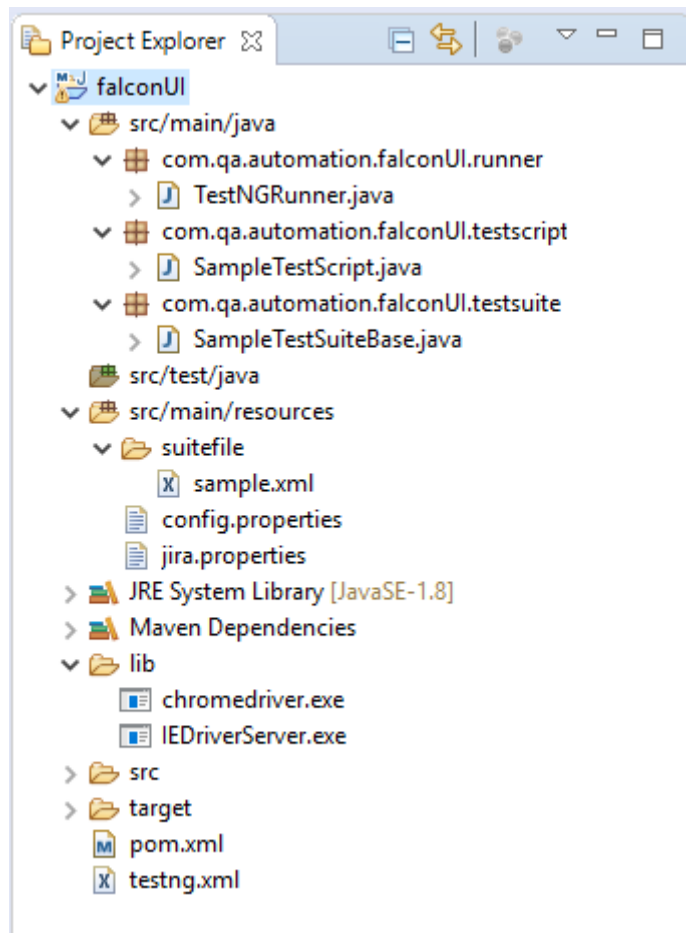
Properties available from archetype:

Name	Value

Advanced

? < Back Next > Finish Cancel

9. Once the Project setup is completed, your Project Structure will be shown as below:



Understanding Project Structure:

In this, we will try to explore the project folder structure.

1. [src/main/java - Application/Library sources](#)
2. [src/main/resources - Application/Library resources](#)
3. [lib - Browser Executables](#)
4. [pom.xml](#)
5. [testing.xml](#)

1. src/main/java - Application/Library sources:

This source folder contains packages like runner, testsuite and testscript.

The main package name follows as **`${Group Id}.${Artifact Id}`**

Earlier, we have created **Artifact Id** as '**falconUI**' and **Group Id** as '**com.qa.automation**'.

So Now, the package name will be **`com.qa.automation.falconUI`**

For any application, if you want to add source code folder like utils, helper etc., you can very well create '**utils/helper**' folder under '**src/main/java/\${packageName}**' and keep all your domain utils/helper under it.

1.1. **`${packageName}/runner/`**

This folder contains '**TestNGRunner.java**' file which has the purpose to set up **listeners, suites to run** to **testng.xml** and **upload TestNG results** (if configured, we will discuss this more detailed in [Understanding config.properties](#))

NOTE: Just keep this file as it is unless you don't have any specific requirements.

1.2. **`${packageName}/testscript/`**

This folder contains your actual test scripts to your application.

1.3. **`${packageName}/testsuite/`**

This folder contains Base Test Suite file for your test scripts.

2. src/main/resources – Application/Library resources

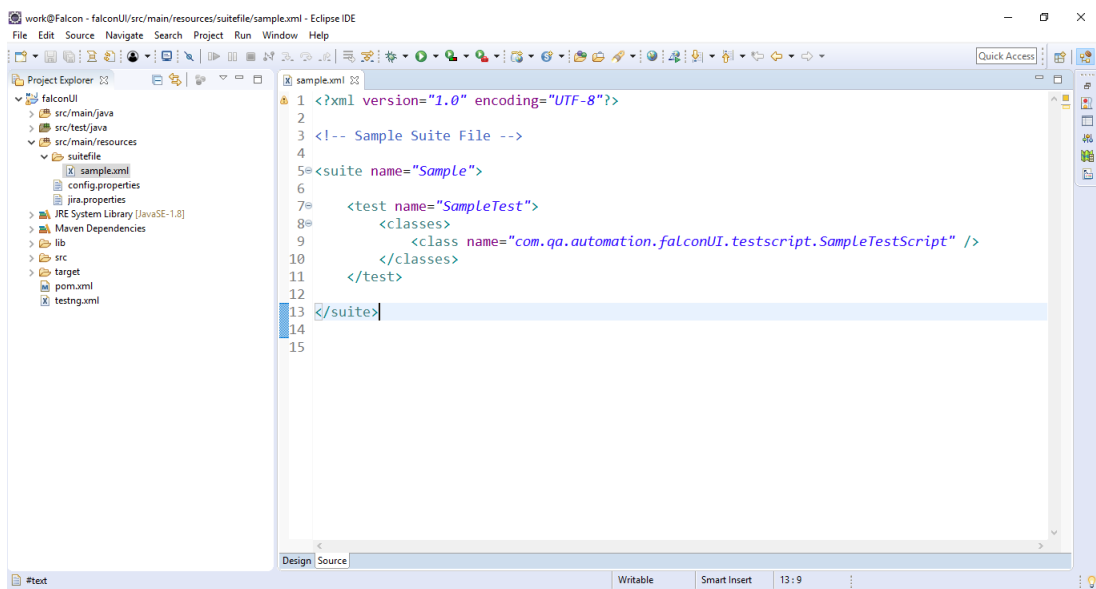
This resources folder contains domain specific resources and test data files.

2.1. **suitefile/**

This folder contains your application specific test suite files where you will mention test classes or packages.

Multiple suite files can be added to **src/main/resources/suitefile/** directory.

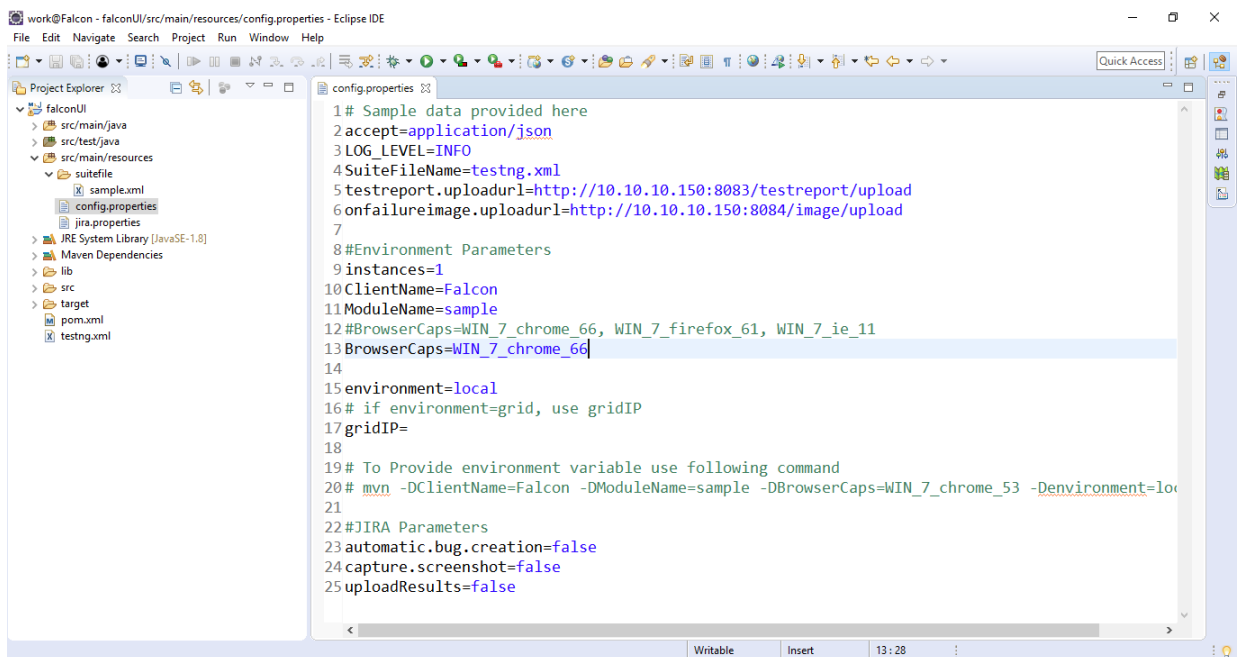
Path of the suite files should be added in **testng.xml**.



2.2. config.properties

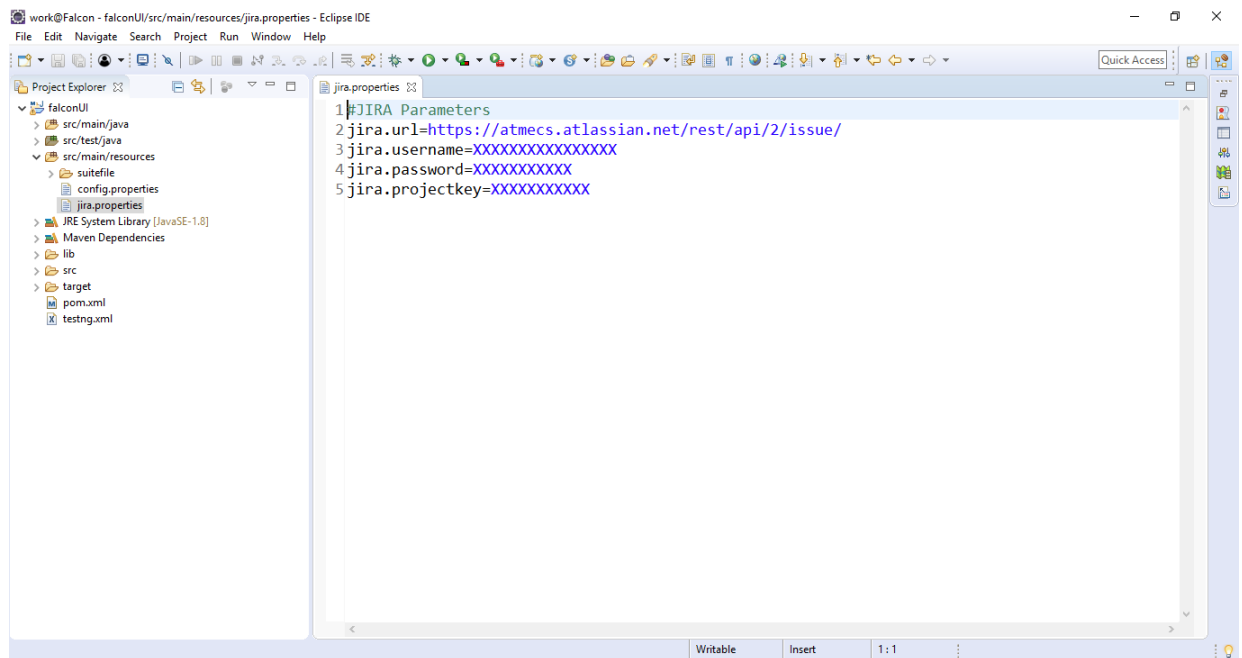
This is the main entry for this project and its purpose is to set up the test environment, browser details and other configurations like bug creation, capturing screenshot and uploading testNG results.

We will discuss this file detailed in [Understanding config.properties](#)



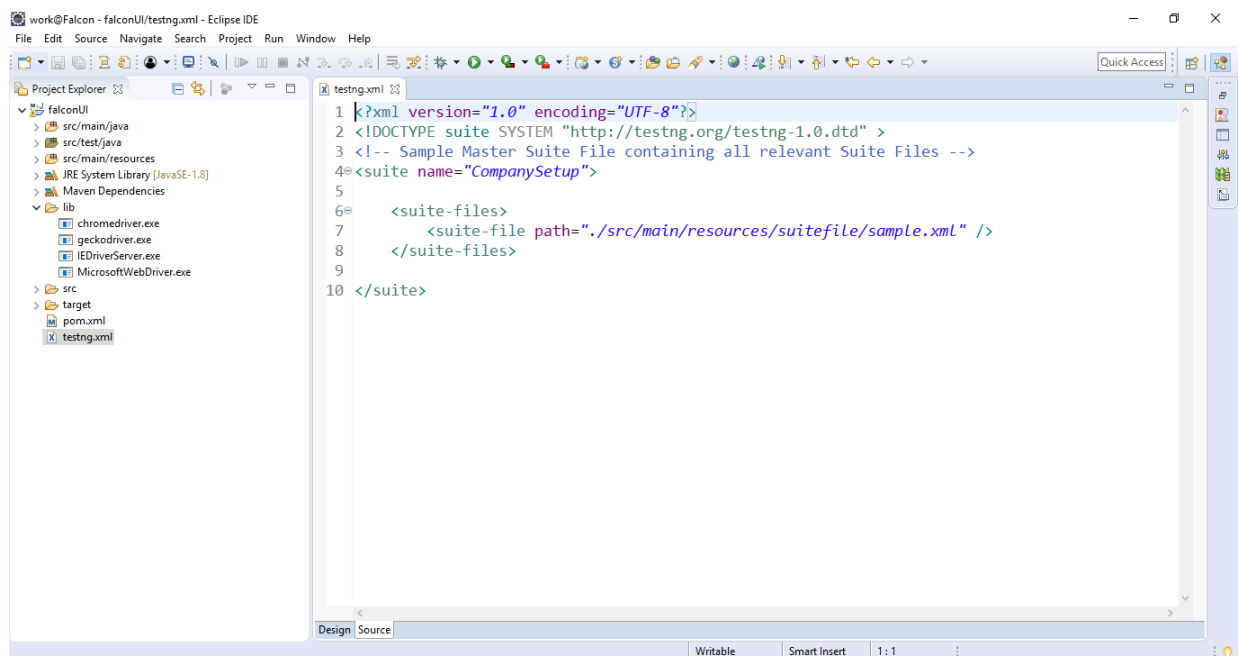
2.3. jira.properties

This is the file where we mention jira credentials for creation of a bug if there is any error or failure.



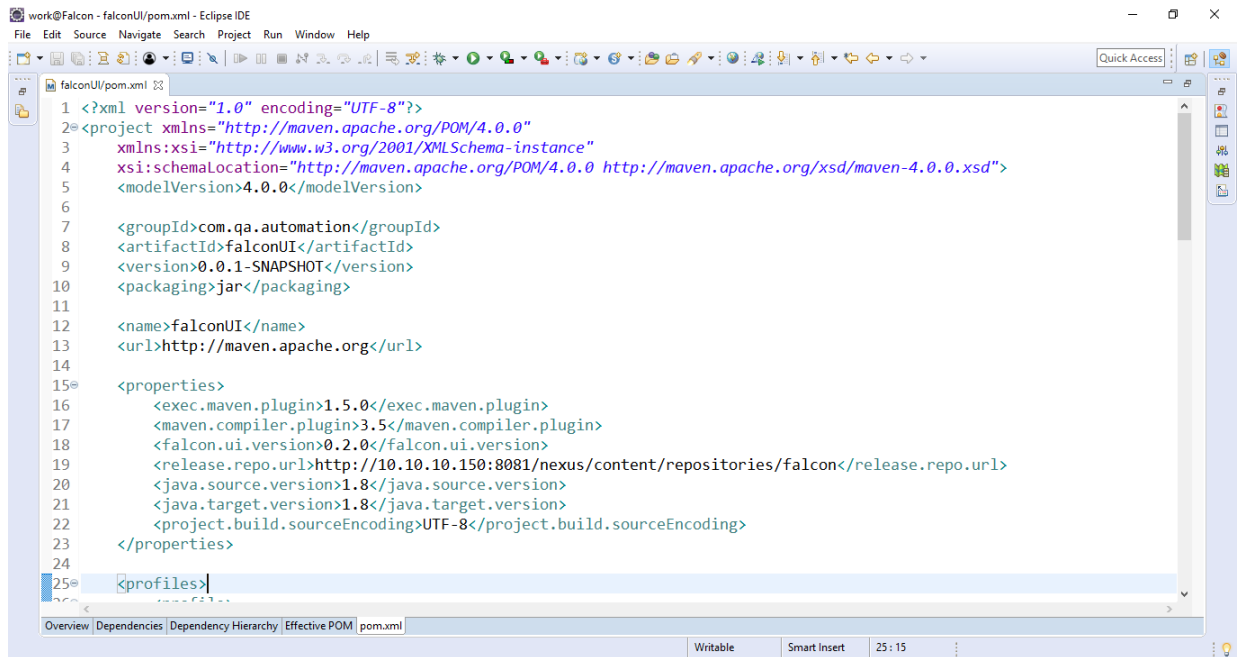
3. lib/

This is the folder which contains all the driver executables like **chromedriver.exe**, **geckodriver.exe**, **IEDriverServer.exe** etc.



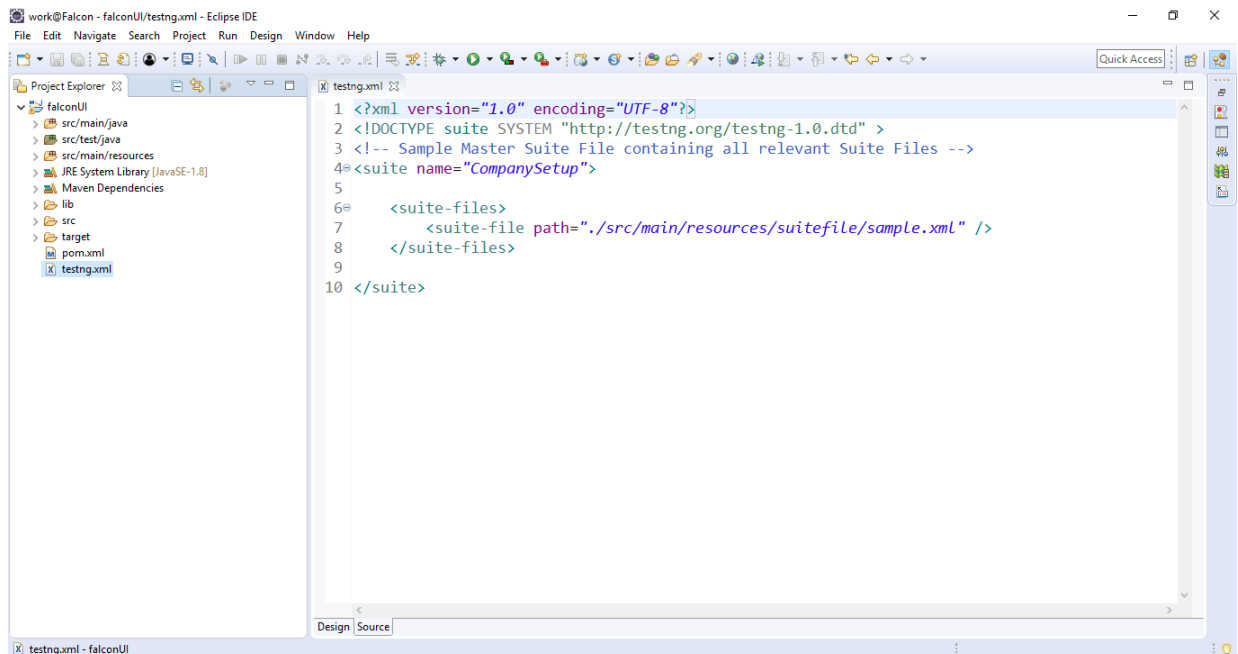
4. pom.xml

This is the main file where the Maven project info is stored and configured to run the test scripts, manage all the falcon and other dependencies.



5. testing.xml

This is the master *.xml file where you will mention all the suite files as below:



Understanding config. properties:

This is the main file which set up the complete test project. Here we will discuss all the main configurations one by one.

1. **SuiteFileName** – This is the property where we store the master *.xml file.
In our case, it is testng.xml
2. **testreport.uploadurl** – This is the property where we store the report dashboard report upload URL. In our case, it is
<http://10.10.10.150:8083/testreport/upload>
3. **onfailureimage.uploadurl** - This is the property where we store the report dashboard image upload URL. In our case, it is
<http://10.10.10.150:8084/image/upload>
4. **instances** – This is the property where we store the number of parallel instances to run.
5. **ClientName** – This is the property where we store the Client Name to identify the Report Dashboard.
6. **ModuleName** – This is the property where we store the list of suite files, comma separated, to execute. Module name is the name of the suite file which are present in ***src/main/resources/suitefile/*** without extension. To add multiple modules(suites) provide multiple ***ModuleName*** values separated by comma (,)
For Example:
ModuleName=sample1,sample2,sampe3
7. **BrowserCaps** – This is the property where we store the test browser details according to instances.
Note: To add multiple browsers provide multiple ***BrowserCaps*** value separated by comma (,)
Supported OS: Win, Mac, Linux
Supported Browsers: firefox (version<=46), chrome, ie, safari
For Example:
WIN_7_chrome_66 (if instances as 1)
WIN_7_chrome_66, WIN_7_firefox_61 (if instances as 2)
8. **environment** – This is the property where we store test environment details like
 - a. local (for local machine run),
 - b. grid (for Selenium Grid Machine run)
 - c. browserstack (for Browser Stack run)

9. **gridIP** – This is the property where we store grid machine details. This property will be picked only when the environment is grid
10. **automatic.bug.creation** – This is the property where we store a Boolean value to create a bug in JIRA.
If True, it will create a JIRA bug according to given properties in jira.properties
If False, it won't create any bug in JIRA even if there is an error or failure.
11. **capture.screenshot** – This is the property where we store a Boolean value to capture a screenshot on test failure.
If True, it will capture a screenshot on test failure
If False, it won't capture any screenshot even if there is an error or failure.
12. **uploadResults** – This is the property where we store a Boolean value to upload the results on completion of execution.

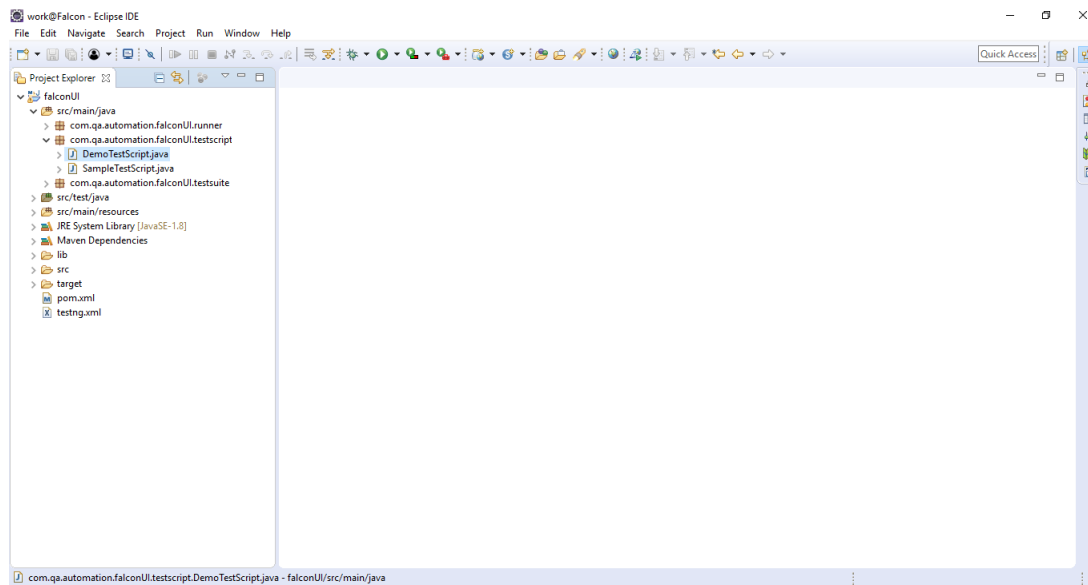
If True, it will upload the testNG results along with screenshot images (if capture.screenshot is true) and error messages if any.
If False, it won't upload the testNG results upon completion of test execution.

Creating and Running Test Scripts:

In this we will look at creating Test Scripts and running them as per test requirement.

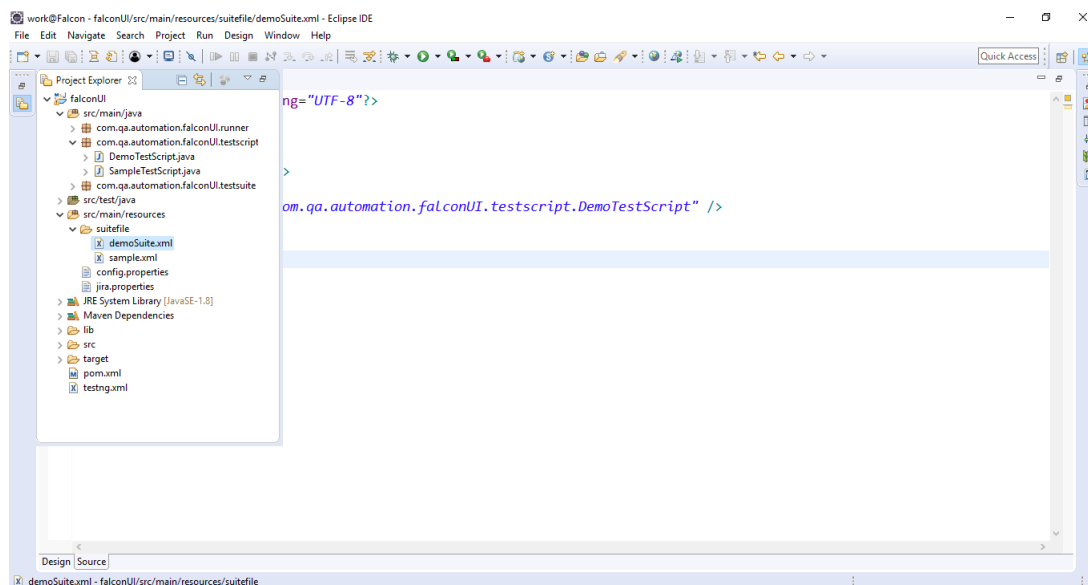
1. Creation of Test Scripts:

Create Your Test Scripts under **testscript** folder as shown in below.



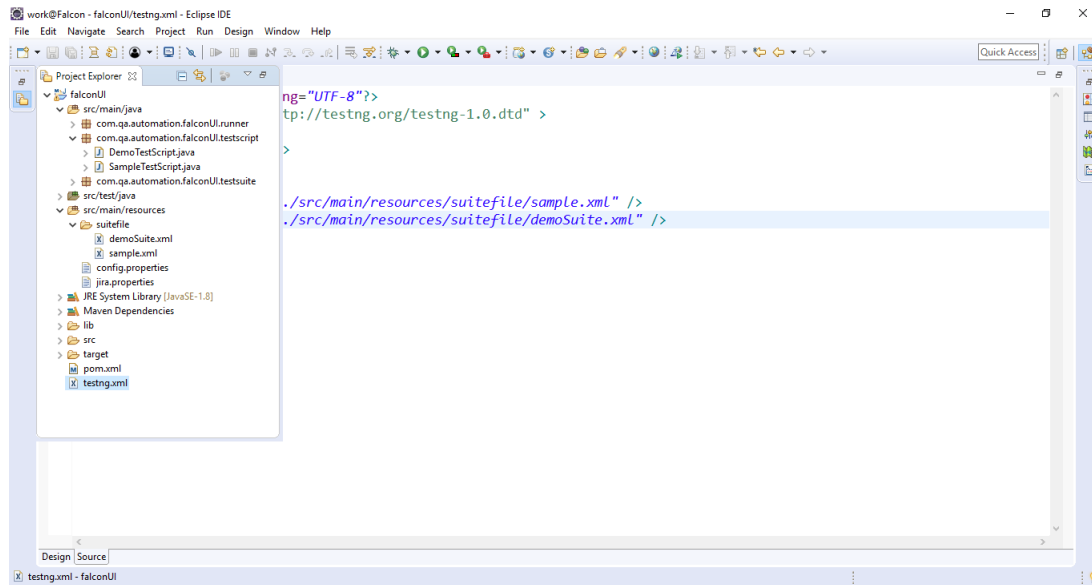
2. Adding Test Scripts to Suite File:

Create a Suite File under **src/main/resources/suitefile** and Add the Created Test Script to it as below:



3. Adding Suite Files to testng.xml:

Add created suite files to **testng.xml** as below:

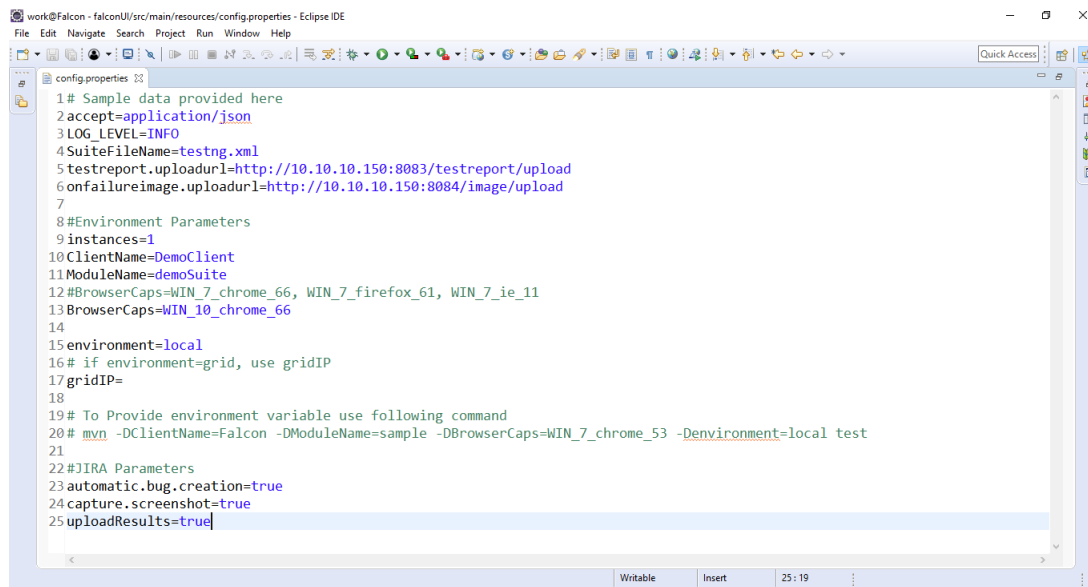


4. Setting up Test Environment in config.properties:

Configure Test Environment under config.properties as below:

Important Configurations:

- a. testreport.uploadurl – Report Dashboard Upload URL (if uploadResults=true)
- b. onfailureimage.uploadurl – Report Dashboard Image Upload URL (if uploadResults=true & capture.screenshot=true)
- c. instances – Number of Parallel Test Runs (1 for Single Run, 2 for Parallel Run)
- d. ClientName – Client Name to identify Reports under Report Dashboard
- e. ModuleName – This is the suite files separated by comma (,) without any extension. For Example: **demoSuite**
- f. BrowserCaps – Browser and OS Details, Syntax will be
OSName_OSVersion_BrowserName_BrowserVersion
We can 've multiple BrowserCaps for Parallel run when instances are more than 1.
For Example:
BrowserCaps=WIN_10_chrome_66 (when instances as 1)
BrowserCaps=WIN_10_chrome_66,WIN_10_ie_11 (when instances as 2)
- g. Environment/gridIp – This will work as below:
 - Local – To Run in the Local Machine
 - Grid – To Run in the Grid Machine (gridIP should be mentioned)
 - Browserstack – To Run in the Cloud Machine (Browser Stack)
- h. automatic.bug.creation – To Create JIRA Issues if there is an error
- i. capture.screenshot – To Capture Screenshot if there is an error
- j. uploadResults – To Upload Results to Report Dashboard



The screenshot shows the Eclipse IDE with the 'config.properties' file open. The file contains configuration settings for a test suite, including sample data, environment parameters, and JIRA parameters. The settings are as follows:

```
1# Sample data provided here
2accept=application/json
3LOG_LEVEL=INFO
4SuiteFileName=testng.xml
5testreport.uploadurl=http://10.10.10.150:8083/testreport/upload
6onfailureimage.uploadurl=http://10.10.10.150:8084/image/upload
7
8#Environment Parameters
9instances=1
10ClientName=DemoClient
11ModuleName=demoSuite
12#BrowserCaps=WIN_7_chrome_66, WIN_7_firefox_61, WIN_7_ie_11
13BrowserCaps=WIN_10_chrome_66
14
15environment=local
16# if environment=grid, use gridIP
17gridIP=
18
19# To Provide environment variable use following command
20# mvn -DClientName=Falcon -DModuleName=sample -DBrowserCaps=WIN_7_chrome_53 -Denvironment=local test
21
22#JIRA Parameters
23automatic.bug.creation=true
24capture.screenshot=true
25uploadResults=true
```

5. Running Test Scripts:

To Run the Project, Right Click on the **Project** and Click on **Run as** and then **Maven Test** as shown in below:

