<https://www.youtube.com/watch?v=P9ZWOWm7i0k>

Architecture of Page Object Model (POM) Design With Selenium - Part -1

PART 2

<https://www.youtube.com/watch?v=LxJzeiTQGoE>

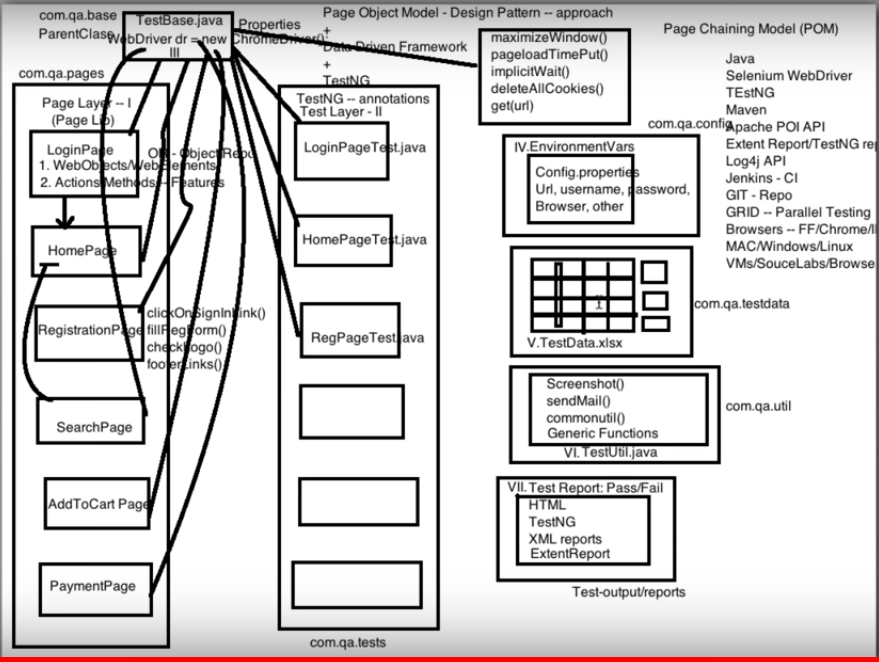
POM is not a framework, it is a design pattern, an automation approach.

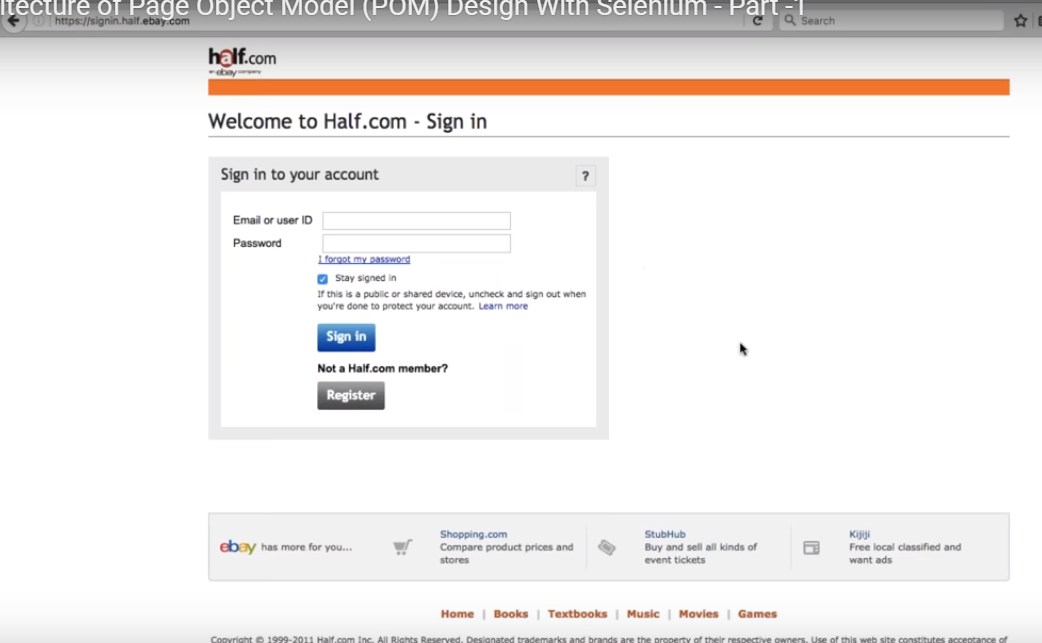
Example we have several pages like login, home, registration, transaction, search , etc.

In POM, for each and every page, we have to create a separate Java class. Suppose we have 100 pages in our application, we have to create 100 classes.

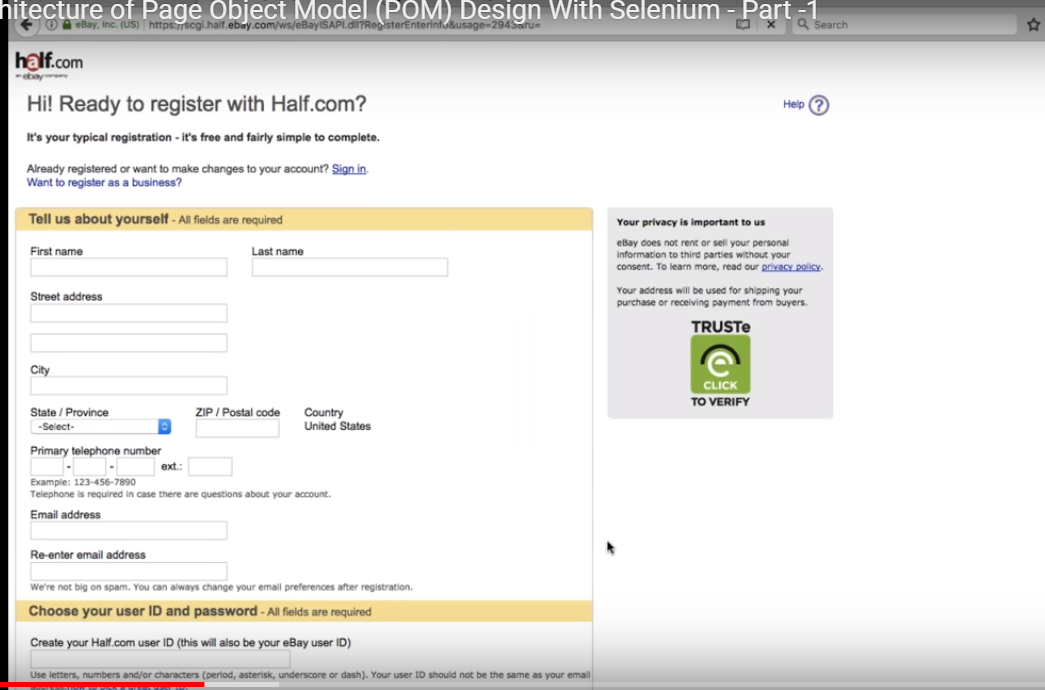
1). Say for example for Login page, we have to identify and define WebElements/WebObjects like uid, pwd, login button, forgot password link, registration link etc.

2). We have to define actions or methods for features for that particular page. Example, login feature, click on forgot password, click on registration link, etc.





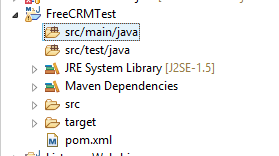
On clicking the register button, below page is displayed



<https://www.youtube.com/watch?v=LxJzeiTQGoE>

Part POM by Naveen Hybrid framework, combination of DDT + POM

By creating Maven-Java project, you will have 2 source folders; src/main/java and src/test/java



We need to setup the pom.xml file. We need to add all the dependencies in this file. Like testNG, apache POI API, selenium webdriver APIs, extent report APIs. No need to download respective jar files.

1). Lets add the dependency for Selenium

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

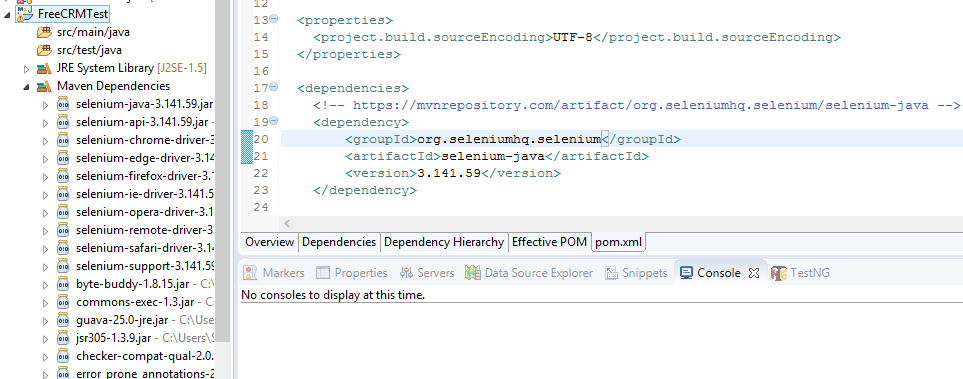
<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

When you add the selenium-java dependency and click on Save button, you will see that all the related jar files automatically got added as below under Maven Dependencies.



========================================

2). Similarly search the dependency for testng

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

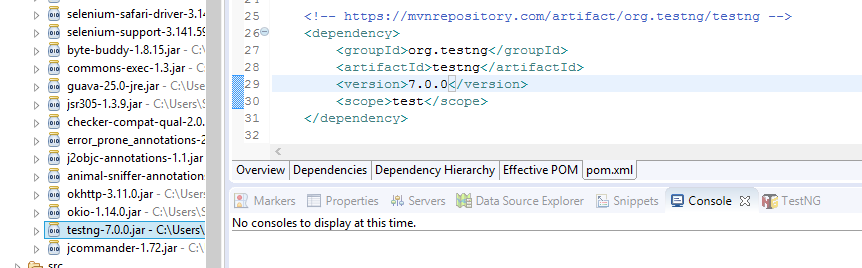
<artifactId>testng</artifactId>

<version>7.0.0</version>

<scope>compile</scope>

</dependency>

Our scope is compile and not test, so change the scope to compile

Now we can see under MavenDependencies folder testng jar file got added as below

Under this url <https://mvnrepository.com/artifact/org.apache.poi>

U will get all the apache poi dependencies

2). Similarly search the dependency for Apache POI API

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>4.1.1</version>

</dependency>

--------------------------------

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>4.1.1</version>

</dependency>

========================

Similarly add all the required Apache poi dependencies as below

<dependencies>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.141.59</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.0.0</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>4.1.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>4.1.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml-schemas -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml-schemas</artifactId>

<version>4.1.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-scratchpad -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-scratchpad</artifactId>

<version>4.1.1</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/ooxml-schemas -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>ooxml-schemas</artifactId>

<version>1.4</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.poi/openxml4j -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>openxml4j</artifactId>

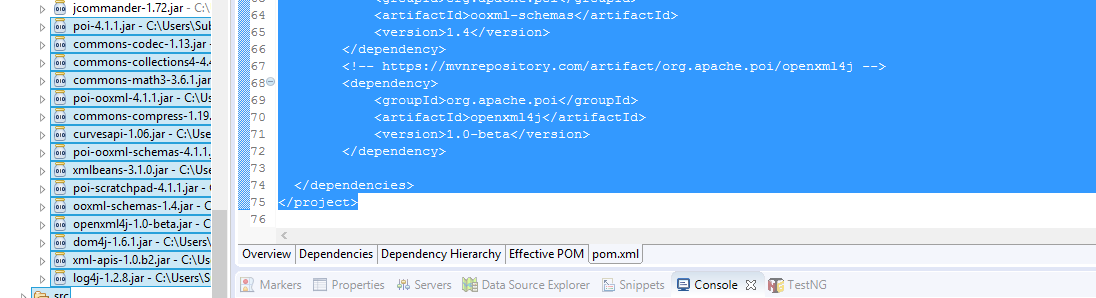
<version>1.0-beta</version>

</dependency>

</dependencies>

</project>

Once you Save it all the dependencies are downloaded and get added as below

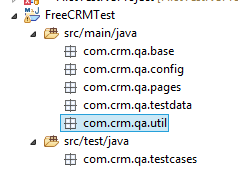


Under the src/main/java ; u need to add or create all your utilities, page classes, libraries etc .

Under src/test/java, u need to create only testcases which r written under testng

https://www.youtube.com/watch?v=LxJzeiTQGoE

13:53

Only the testcases package is created seperately

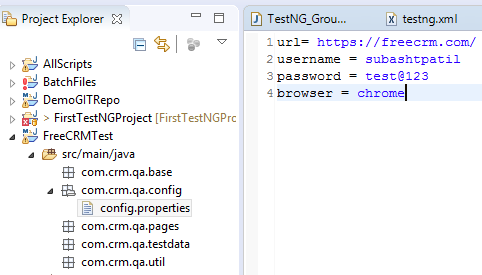
This is the application used <https://freecrm.com/index.html>

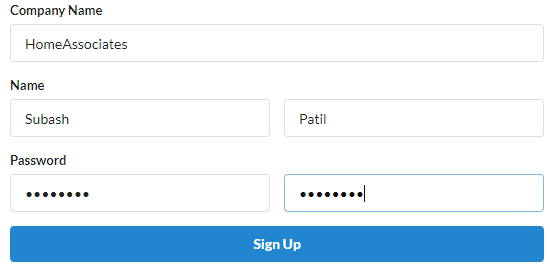
Create a config.properties file under com.crm.qa.config

We use this config.properties file only for global environment variables not for testdata

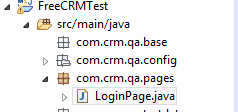
Username = subashtpatil@gmail.com

Password = Test@123





1). Now create a LoginPage class as below



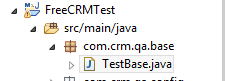
2). Now create a class for the SignUp page

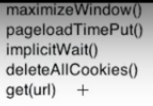
3). Create the HomePage class

4). Create the ContactsPage class

5). Create the DealsPage class.

Now will create a Base class which is parent class of all the classes inside the base package



In the base class we will perform steps like  so we will inherit all the above methods from base class into all page classes

Create a testBase() constructor as below in the base class

**package** com.crm.qa.base;

**import** java.io.FileInputStream;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.util.Properties;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.openqa.selenium.firefox.FirefoxDriver;

**import** com.crm.qa.util.TestUtil;

**public** **class** TestBase {

**public** **static** WebDriver *driver*;

**public** **static** Properties *prop*;

**public** TestBase() {

**try** {

*prop* = **new** Properties();

FileInputStream ip= **new** FileInputStream("E:\\MyWorkspace\_Completed\\FreeCRMTest\\src\\main\\java\\com\\crm\\qa\\config\\config.properties");

*prop*.load(ip);

}**catch** (FileNotFoundException e) {

e.printStackTrace();

}**catch** (IOException e) {

e.printStackTrace();

}

}

**public** **static** **void** initialization() {

String browserName= *prop*.getProperty("browser");

**if** (browserName.equals("chrome")) {

System.*setProperty*("webdriver.chrome.driver","E:\\MyWorkspace\_Completed\\FreeCRMTest\\Software\\Old\_chromedriver.exe" ); // Old\_chromedriver.exe

//System.setProperty("webdriver.chrome.driver",".//FreeCRMTest//Software//Old\_chromedriver.exe" );

*driver*= **new** ChromeDriver();

}**else** **if** (browserName.equals("FF")) {

System.*setProperty*("webdriver.gecko.driver","E:\\MyWorkspace\_Completed\\FreeCRMTest\\Software\\geckodriver.exe" );

*driver*= **new** FirefoxDriver();

}

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

*driver*.manage().deleteAllCookies();

*driver*.manage().timeouts().pageLoadTimeout(TestUtil.*PAGE\_LOAD\_TIMEOUT*, TimeUnit.***SECONDS***);

//driver.manage().timeouts().pageLoadTimeout(60, TimeUnit.SECONDS);

//driver.manage().timeouts().implicitlyWait((prop.getProperty("IMPLICIT\_WAIT")), TimeUnit.SECONDS);

*driver*.manage().timeouts().implicitlyWait(TestUtil.*IMPLICIT\_WAIT*, TimeUnit.***SECONDS***);

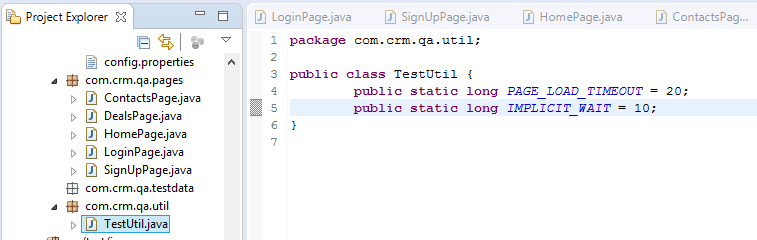
//if u want to navigate to the origin of this code TestUtil.IMPLICIT\_WAIT use ctrl+mouseclick

*driver*.get(*prop*.getProperty("url"));

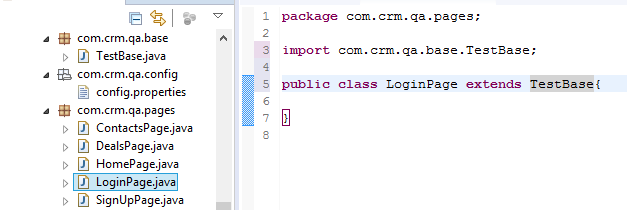
}

}

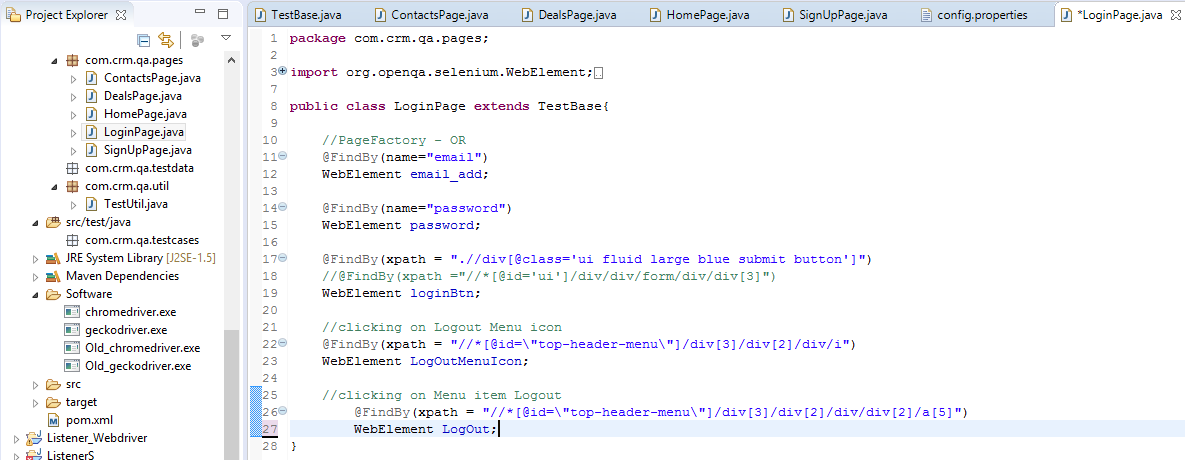
Under the com.crm.qa.util, create a TestUtil class as below;

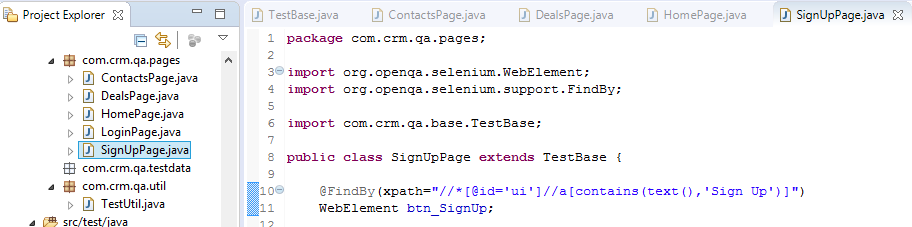


Now you need to inherit the TestBase(parent) class into all the child page classes using the keyword extends as below;



Now we have to define the PageFactory or OR for each of the pages as below for Login page





How will you initialize your PageFactory objects?

We need to create a constructor in the LoginPage as below

We initialize elements of the PageFactory using initElements



public LoginPage(WebDriver rdriver) {   
 *driver* = rdriver;  
 *//PageFactory.initElements(driver, this);  
 //wait = new WebDriverWait(driver, 60);  
  
  
 //this.signinPage = new SigninPageLocators();* AjaxElementLocatorFactory factory = new AjaxElementLocatorFactory(*driver*,10);  
 PageFactory.*initElements*(factory, this);  
}

Code explanation;

1). driver is coming from the TestBase class.

2). How to initialize ur PageFactory; using the method ;

PageFactory.initElements (Where initElements means initialize elements with the driver)

3). And **this** means all the variables in the current class like username, pwd, LoginBtn, etc will be initialized with this driver. This means it is pointing to the current class object.

After creating the PageFactory we then

