



Framework

Class 1

Agenda

What is Framework

What Maven and why do we need it?

What is Maven Repository? Types of Maven Repo.

What is pom.xml file? Adding dependencies in pom.xml

What is Page Object Model and why do we need it?

How to implement POM ?

What is Framework?

A testing framework is a set of guidelines or rules used for creating and designing test cases. These guidelines could include coding standards, test-data handling methods, object repositories, processes for storing test results, or information on how to access external resources.

Framework is a collection of concepts you know in an organised manner with some proper rules, validation, proper exception and error handling which can be easily reuse, manage and scale.

In a framework we create set of multiple reusable components like
methods to launch browser, methods for clicking, sending keys, selecting, switching between frames etc.

We can create methods for reading writing from external data sources, validation, reporting etc and can place it in such a way that you and other projects can also use it. It saves efforts, time and of course money of company.

Why Framework?

Framework defines the organization's way of doing things – a 'Single Standard'. Following this standard would result in the project team achieving:

Script-less representation of Automated tests:

The testing framework should offer point-and-click interface for accessing and interacting with the application components under test—as opposed to presenting line after line of scripting.

Testers should be able to visualize each step of the business scenario, view and edit test cases intuitively. This will shorten the learning curve for testers and help QA teams meet deadlines.

Data Driven tests: Excel A key benefit of automating functional testing is the ability to test large volumes of data on the system quickly. But you must be able to manipulate the data sets, perform calculations, and quickly create hundreds of test iterations and permutations with minimal effort.

Concise Reporting: The framework must automatically generate reports of the test run and show the results in an easy-to-read format. The reports should provide specifics about where application failures occurred and what test data was used.

Why Framework?

- Reports must present application screenshots for every step to highlight any discrepancies and provide detailed explanations of each checkpoint pass and failure. Reports must also be easily shared across the entire QA and development teams.
- **Standard Scripting and Team Consistency:** Modular Scripting standard should be maintained across the framework library creation, which includes business components, system communications, data check points, loggers, reporters etc. Project team should follow the defined scripting standards. Published standards across the project team pre-empt the effort involved in duplicate coding, which prevent individuals from following their own coding standards.
- **Implement and Maximize Re-Usability:** Documents Establish the developed libraries across the organization/project team/product team, i.e. publish the library and provide access rights. Utilities/components shared across the team. Usage of available libraries. Minimized effort for repeated regression cycle.

Characteristics of a framework

- Framework should be developed in such a way that any body can plug and use it. They should be able to start scripting with minimal or no setup.
- It must not be project dependent.
- It should be reusable and easy to manage and maintain.
- Flows should be clear.
- It should be simple so that an average automation tester could understand and use it.
- It should not contain scripting codes.
- It should be easily extendable i.e. enhancing.
- It should be platform independent.

Benefits of Test Automation Framework

Utilizing a framework for automated testing will increase a team's test speed and efficiency, improve test accuracy, and will reduce test maintenance costs as well as lower risks. They are essential to an efficient automated testing process for a few key reasons:

- Improved test efficiency
- Lower maintenance costs
- Minimal manual intervention
- Maximum test coverage
- Reusability of code

What is Maven?

- Maven is build automation and management tool developed by Apache Software Foundation. It was initially released on 13 July 2004.
- It allows the developer to create projects using **Project Object Model and plugins**.
- It helps to build **projects, dependency, and documentation**
- Maven allows the developer to automate the process of the creation of the initial folder structure, performing the compilation and testing and the packaging and deployment of the final product. It cuts down the good number of steps in build process and it makes it one step process to do a build which makes programmer's life easier

Why To Use Maven In Selenium Project?

- Maven uses **POM.xml** configuration file which kept all project configuration related Information.
- For selenium, we need to provide selenium webdriver version related information in POM.xml file and then it will download all required jar files automatically and store it **in local repository called m2**.
- Later on if we want to change version of selenium webdriver then we **need to modify version** in POM.xml file only. Then maven will download new version jar files automatically and store in local repository.
- If we upgrade any dependencies version in POM.xml file, first maven will check if that version of jar files are available or not in local repository. If available then fine else It will **download them from maven central repository**.

Main objectives of Maven

- It makes project build process easy.
- It provides easy and uniform build system.
- It manages project dependencies by downloading required dependencies jar files automatically from Maven central repositories which also allows users to reuse same jars across multiple projects.
- Provides guidelines for better project management practices.
- It allows to build project using project object model (POM).
- It provides quality project document information.
- Allows execution of Tests from Command Line

What is a Maven Repository?

Maven Repository is a directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

There are three types of maven repository:

- **local** - is a folder location on your machine. It gets created when we run any maven command for the first time. Maven local repository keeps project's all dependencies (library jars, plugin jars etc.). Maven local repository by default get created by Maven in %USER_HOME% directory.
- **central** - is repository provided by Maven community. It contains a large number of commonly used libraries. This server downloads all the latests jar files from the remote maven repositories.
- **remote** - is a repository that manages by the companies who own the software, but they will expose the maven software to the maven.
 - Selenium jars are available at openqa servers
 - TestNG, POI jars are available at apache server

How Maven works?

- Maven is dependency management tool i.e to execute the frameworks, we need the jar files. Maven helps user to keep the up to date jar file on the framework.
- During runtime maven will check the version of the jar files present in the local system and compares with pom.xml file. If the latest version is available online then it downloads them automatically and replaces the local version and then executes the framework.
- If there is no internet connection or if the latest version is not available, then maven executes the framework with the jar files available in the local system.
- Maven can helps us to **minimize** project and build management **time and efforts**.

Create maven project in Eclipse

MAVEN helps us in creating the project structure, managing and downloading the dependencies.

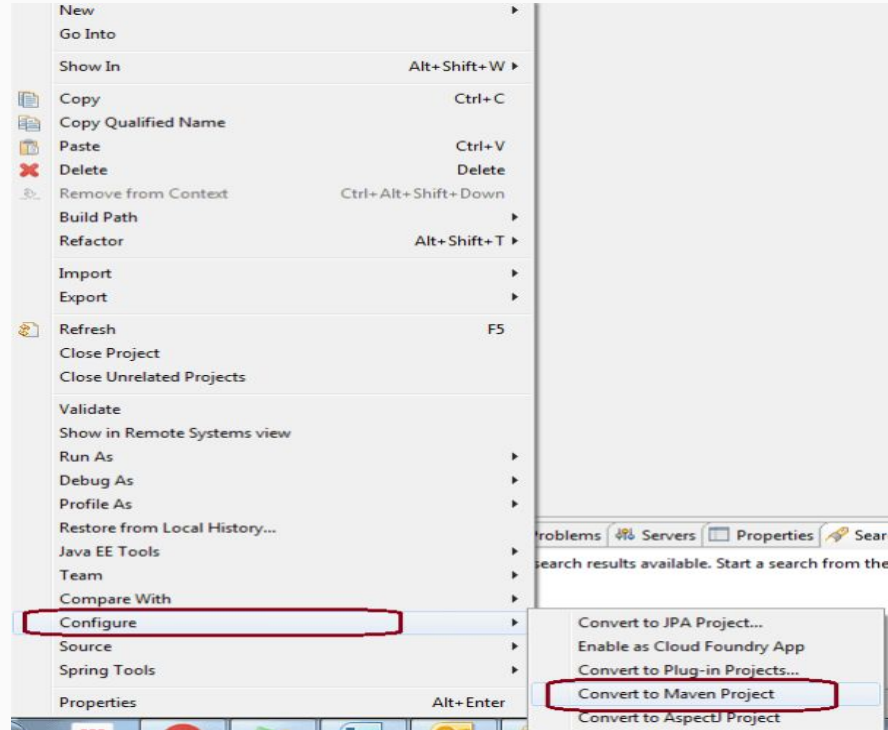
We need to define the required dependencies in pom.xml. By default maven adds few dependencies specific to project structures and that will be based on the archetype that we choose.

After creating Maven project if we observe, we will see pom.xml file created.

Now we can start adding the dependencies which ever we require for our project like Selenium jars, Testng jars etc. in **pom.xml** file

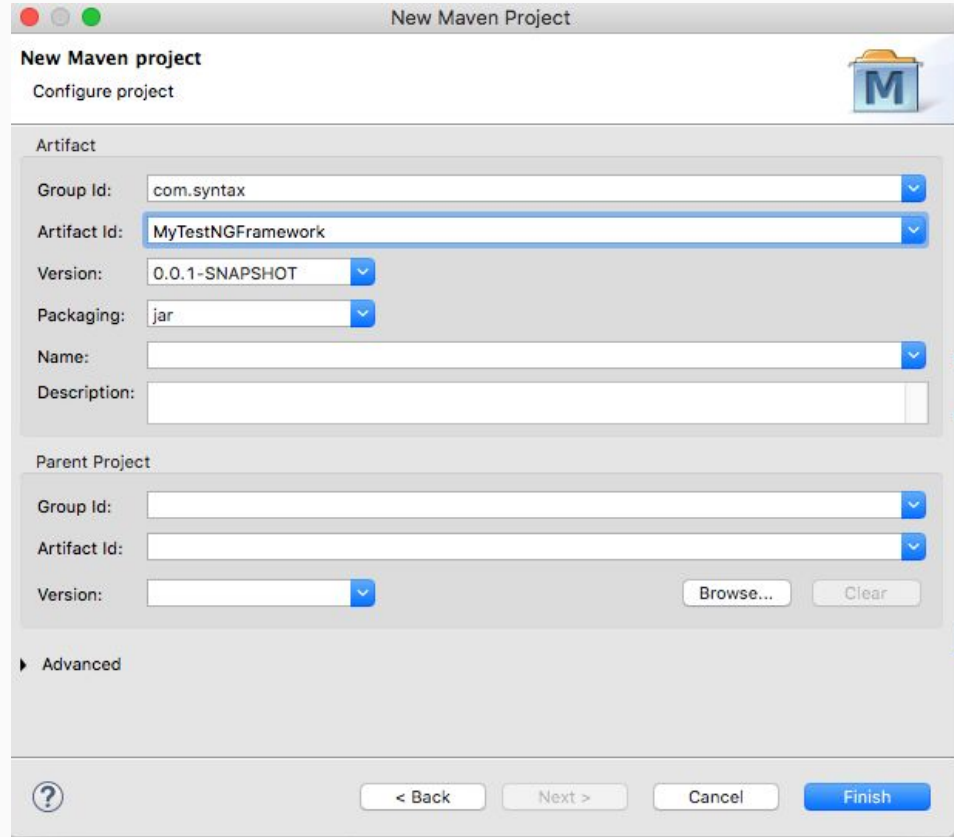
Convert Java project to Maven project

Once the maven plugin is installed in eclipse then our task is so simple Select the project which needs to be converted into a maven project and right click on it and click on **Configure** and you will see “**Convert to Maven Project**” click on it.



Maven Example

- In eclipse, click on **File** menu → **New** → **Project** → **Maven** → **Maven Project**. → **Select CheckBox** to create simple project → **Next** → Now write the group Id, artifact Id → **finish**.



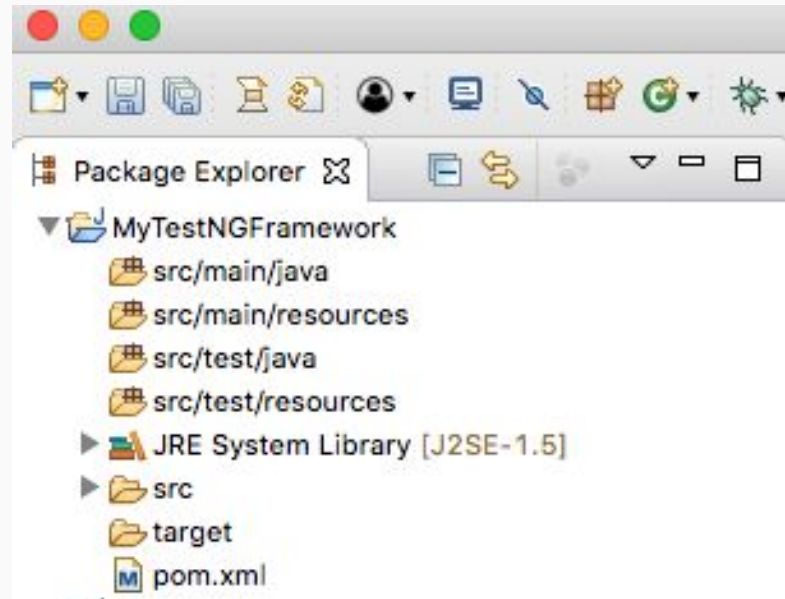
The screenshot shows the 'New Maven Project' dialog box in Eclipse. The title bar says 'New Maven Project'. Below the title bar, there's a section 'New Maven project' with a sub-label 'Configure project' and a Maven logo icon. The dialog is divided into several sections:

- Artifact**: This section contains five fields:
 - Group Id:** A text field with the value 'com.syntax' and a dropdown arrow.
 - Artifact Id:** A text field with the value 'MyTestNGFramework' and a dropdown arrow.
 - Version:** A text field with the value '0.0.1-SNAPSHOT' and a dropdown arrow.
 - Packaging:** A text field with the value 'jar' and a dropdown arrow.
 - Name:** An empty text field with a dropdown arrow.
 - Description:** An empty text field.
- Parent Project**: This section contains three fields:
 - Group Id:** An empty text field with a dropdown arrow.
 - Artifact Id:** An empty text field with a dropdown arrow.
 - Version:** An empty text field with a dropdown arrow.
 - Buttons: 'Browse...' and 'Clear'.
- Advanced**: A section with a collapsed arrow icon and the label 'Advanced'.

At the bottom of the dialog, there are four buttons: a help icon (question mark in a circle), '< Back', 'Next >', 'Cancel', and 'Finish'.

Maven Example

Now you will see a maven project with complete directory structure. All the files will be created automatically such as Hello Java file, pom.xml file, test case file etc. The directory structure of the maven project is shown in the below figure.



What is POM.xml

POM is an acronym for **Project Object Model**.

pom.xml which is the core of any project. This is the configuration file where all required information are kept.

The pom.xml file contains **information of project and configuration information for the maven** to build the project such as **dependencies, build directory, source directory, test source directory, plugin, goals etc.**

POM also contains the goals and plugins. While executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal.

Elements of maven pom.xml file

Element	Description
project	It is the root element of pom.xml file.
model	It is the sub element of project. It specifies the modelVersion. It should be set to 4.0.0.
groupId	<p>It is the sub element of project. It specifies the id for the project group.</p> <p>Generally groupId refers to domain id. For best practices company name is used as groupId. It identifies the project uniquely</p>
artifactId	<p>It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.</p>
version	It is the sub element of project. It specifies the version of the artifact under given group.
dependencies	defines dependencies for this project.

POM.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.javatpoint.application1</groupId>

    <artifactId>my-app</artifactId>

    <version>1</version>

</project>
```

How to add dependency in Maven using Eclipse

In order to add the dependent jar for our project using Maven, we can add it through `<dependencies></dependencies>` tag in our **POM.xml** like below.

```
<dependencies>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-annotations</artifactId>
    <version>2.6.1</version>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.6.1</version>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.6.1</version>
  </dependency>
</dependencies>
```

What is POM?

Page Object model is a design pattern for enhancing test maintenance and reducing code duplication.

Page Object model is an object design pattern in Selenium, where web pages are represented as classes, and the various elements on the page are defined as variables on the class. All possible user interactions can then be implemented as methods on the class.

Page Object Model is a design pattern to create Object Repository for web UI elements. For each web page in the application there should be corresponding page class. This Page class will find the WebElements of that web page and also contains Page methods which perform operations on those WebElements.

Advantages of Page Object Model

- Code reusability – We could achieve code reusability by writing the code once and use it in different tests.
- Code maintainability – There is a clean separation between test code and page specific code such as locators and layout which becomes very easy to maintain code. Code changes only on Page Object Classes when a UI change occurs. It enhances test maintenance and reduces code duplication.
- Object Repository – Each page will be defined as a java class. All the fields in the page will be defined in an interface as members. The class will then implement the interface.
- Readability – Improves readability due to clean separation between test code and page specific code

Design and Implementation of POM using Selenium

How to Design:

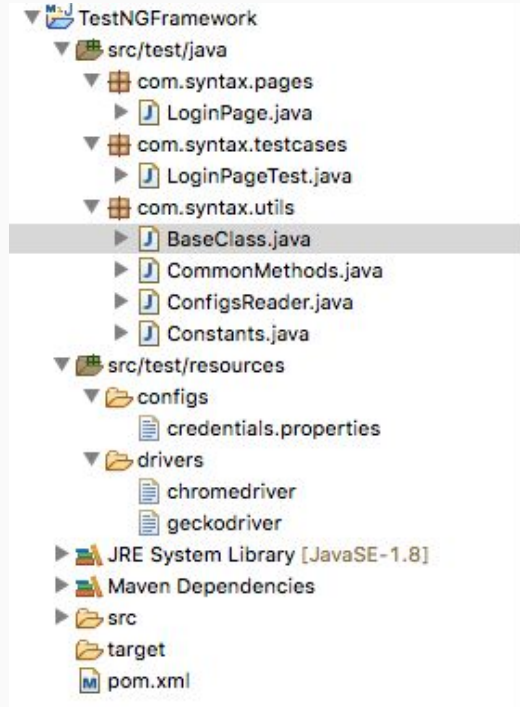
- For each Page in the application we will be creating a separate Java Class.
- Each class is referred to as a PageObjects and returns other PageObjects to facilitate the flow between pages.
- Page Object class is responsible to find the WebElements of that page and also hold methods which perform operations on those WebElements.
- We will divide our Framework structure into three parts.
- Page classes, Test Classes and Utility classes.

How to implement :

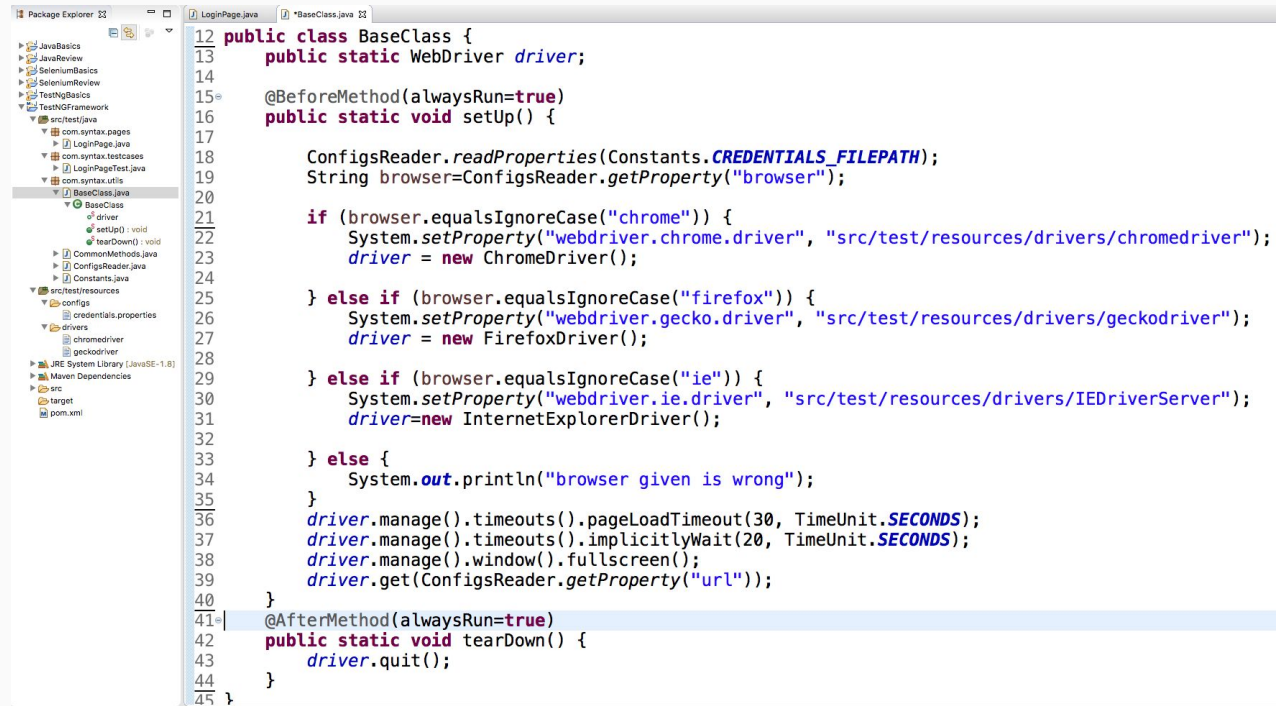
1. Page Object model without PageFactory
1. Page Object Model with PageFactory

Design and Implementation of POM using Selenium

In Pages package, we design the Page classes for every single page available in Website. In page classes, we define the Locators for the elements present on the particular web page and actions what we can take on these Web Elements.



Page Object model without PageFactory



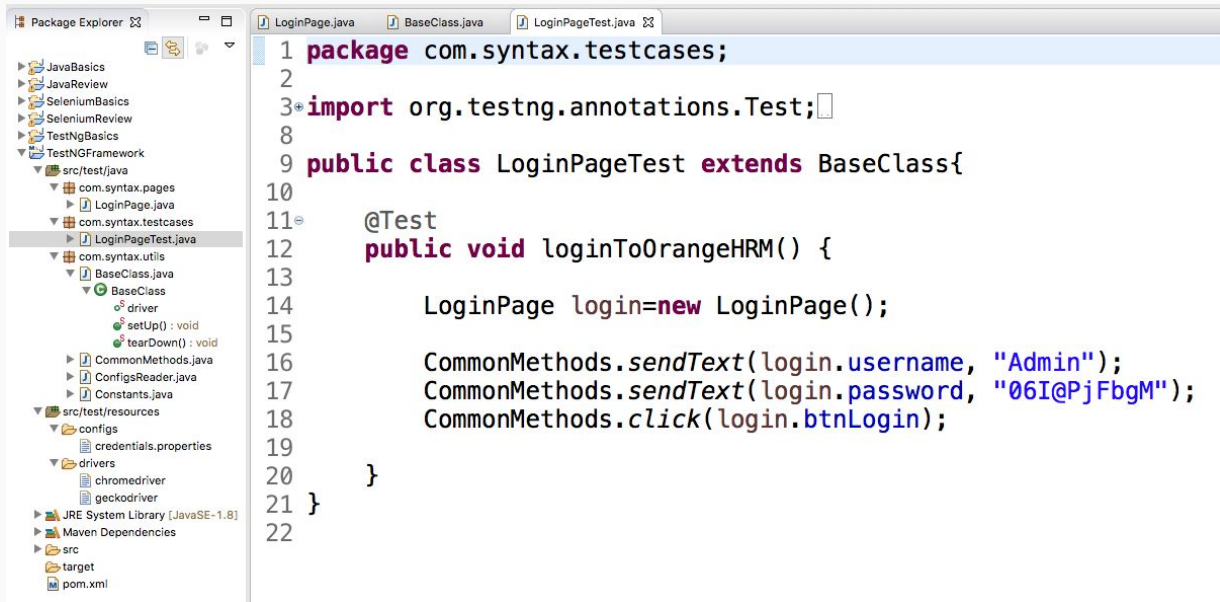
- In utility package, we declare and implement the project utilities like logging, configuration, connectivity, reporting implementation.
- In our example, for the time, we have created BaseClass that will initialize WebDriver
- Every page class will inherit this BasePage.Class class.

Page Object model without PageFactory

```
LoginPage.java
1 package com.syntax.pages;
2
3 import org.openqa.selenium.By;
4 import org.openqa.selenium.WebElement;
5
6 import com.syntax.utils.BaseClass;
7
8 public class LoginPage extends BaseClass {
9
10     public WebElement username = driver.findElement(By.id("txtUsername"));
11     public WebElement password = driver.findElement(By.id("txtPassword"));
12     public WebElement btnLogin = driver.findElement(By.id("btnLogin"));
13
14 }
15
```

- In pages package we created LoginPage class that is being inherited from BaseClass.
- In LoginPage class, we have declared the Locators for the WebElement Username, Password, Login button and Logo and also define the actions, what we can perform on these elements like input text and click the button.

Page Object model without PageFactory



- The second thing we do is create the Separate Test Class for each page class or each functionality.
- In our example, we are creating the LoginTest.class .
- In this Test Class, we create the Test scripts.
- In our example, we are creating the Login test. We created the object of **LoginPage class** and called elements of LoginPage class in our Test Class to create the test script.

Page Factory in Selenium POM Framework

Page Factory is an inbuilt page object model concept for Selenium WebDriver but it is very optimized.

Here as well we follow the concept of separation of Page Object repository and Test methods. Additionally with the help of **PageFactory** class we use annotations **@FindBy** to find WebElement.

We use **initElements** method to initialize web elements.

The **PageFactory** Class is an extension of the Page Object design pattern. It is used to **initialize** the elements of the Page Object or **instantiate** the Page Objects itself.

PageFactory is used to initialize elements of a Page class without having to use 'FindElement' or 'FindElements'.

Annotations can be used to supply descriptive names of target objects to improve code readability.

@FindBy

In PageFactory, we use the **@FindBy** annotations to store the WebElements.

The **@FindBy** annotation supports all the other locators strategies that we use:
id, name, className, css, xpath, tagName, linkText
and partialLinkText

```
@FindBy(name="txtUsername")  
private WebElement user_name;
```

PageFactory.init Elements

We should initialize page objects using `initElements()` method from `PageFactory` Class as below, Once we call `initElements()` method, all elements will get initialized.

`PageFactory.initElements()` static method takes the driver instance of the given class and the class type, and returns a Page Object with its fields fully initialized

`PageFactory.initElements(driver, this);`
or
***`PageFactory.InitElements(WebDriver,
PageObject);`***

InitElements

This Instantiate an Instance /Elements of the given class.

This method will attempt to instantiate the class given to it, preferably using a constructor which takes a WebDriver instance as its only argument or falling back on a no-arg constructor.

An exception will be thrown if the class cannot be instantiated.

WebDriver – The driver that will be used to look up the elements

PageObjects – A class which will be initialised

Returns: An instantiated instance of the class with WebElement and List<WebElement> fields

Page Factory in Selenium POM Framework

```
public class LoginPage extends BaseClass {  
  
    @FindBy(id = "divLogo")  
    public WebElement logo;  
  
    @FindBy(id = "txtUsername")  
    public WebElement userName;  
  
    @FindBy(id = "txtPassword")  
    public WebElement password;  
  
    @FindBy(id = "btnLogin")  
    public WebElement loginBtn;  
  
    public LoginPage() {  
        PageFactory.initElements(driver, this);  
    }  
}
```


Page Factory in Selenium POM Framework

```
public class LoginPageTest extends BaseClass{

    HomePage home;
    LoginPage login;

    @Test
    public void userLogin() {
        login=new LoginPage();
        CommonMethods.sendText(login.userName, ConfigsReader.getProperty("userName"));
        CommonMethods.sendText(login.password, ConfigsReader.getProperty("password"));
        CommonMethods.click(login.loginBtn);
        home=new HomePage();
        String actualName=home.accountName.getText();
        Assert.assertEquals(actualName, "Admin");
    }
}
```