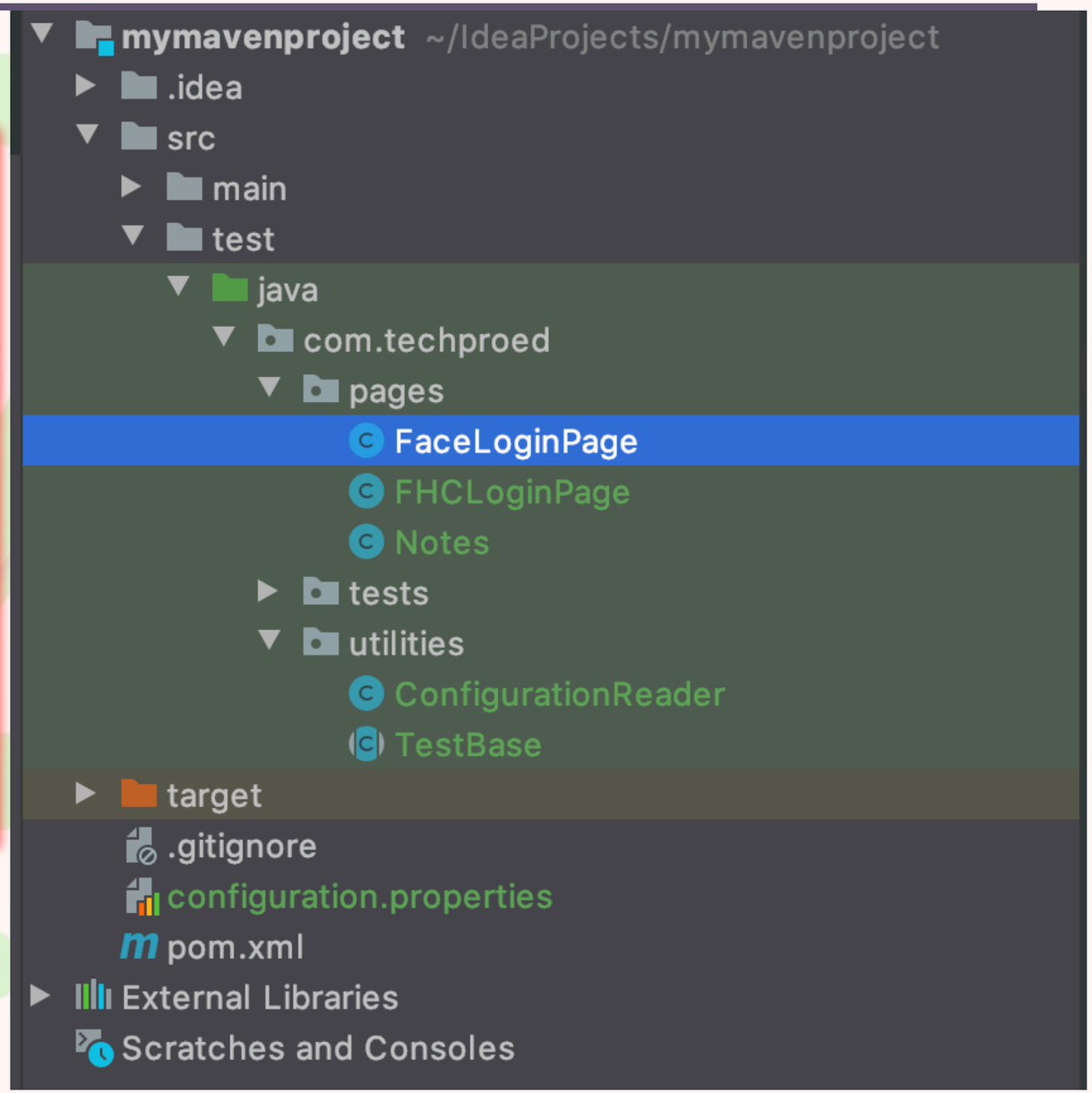
PAGE OBJECT MODEL

- This is a very popular **Framework Design Pattern**.
- When we have a lot of tests in our test suits, it gets more complex to maintain our test cases and codes to have a better framework design that is **maintainable, reusable, faster, understandable.**
- With page object model, we keep page specific elements or methods on **page classes**, and keep them away from the actual **test classes.**
- We will basically create **page classes**, and **test classes** using core Java and Selenium concept to increase the **efficiency** of our framework.



PAGE OBJECT MODEL

- Maintaining Automation framework (part of stlc) may be challenging over the time as we build the test cases and codes.
- When a functionality of an application changes, we have to check our framework t**o fix** the code.
- Page Object Design help us create more independent test cases so it will be easier to **debug(try to understand what is the problem OR step by step execution)** test scripts.
- **Page Object Design is an automation framework design** for testing applications to reduce the challenges.
- Page Object Design make it possible **to reuse the objects, classes, methods, data,** etc. So testers do less work.We just create once and use multiple times.
- A better design makes the test execution time faster.
- Not all companies use page object model design, but everyone knows about it, and it is getting more popular.

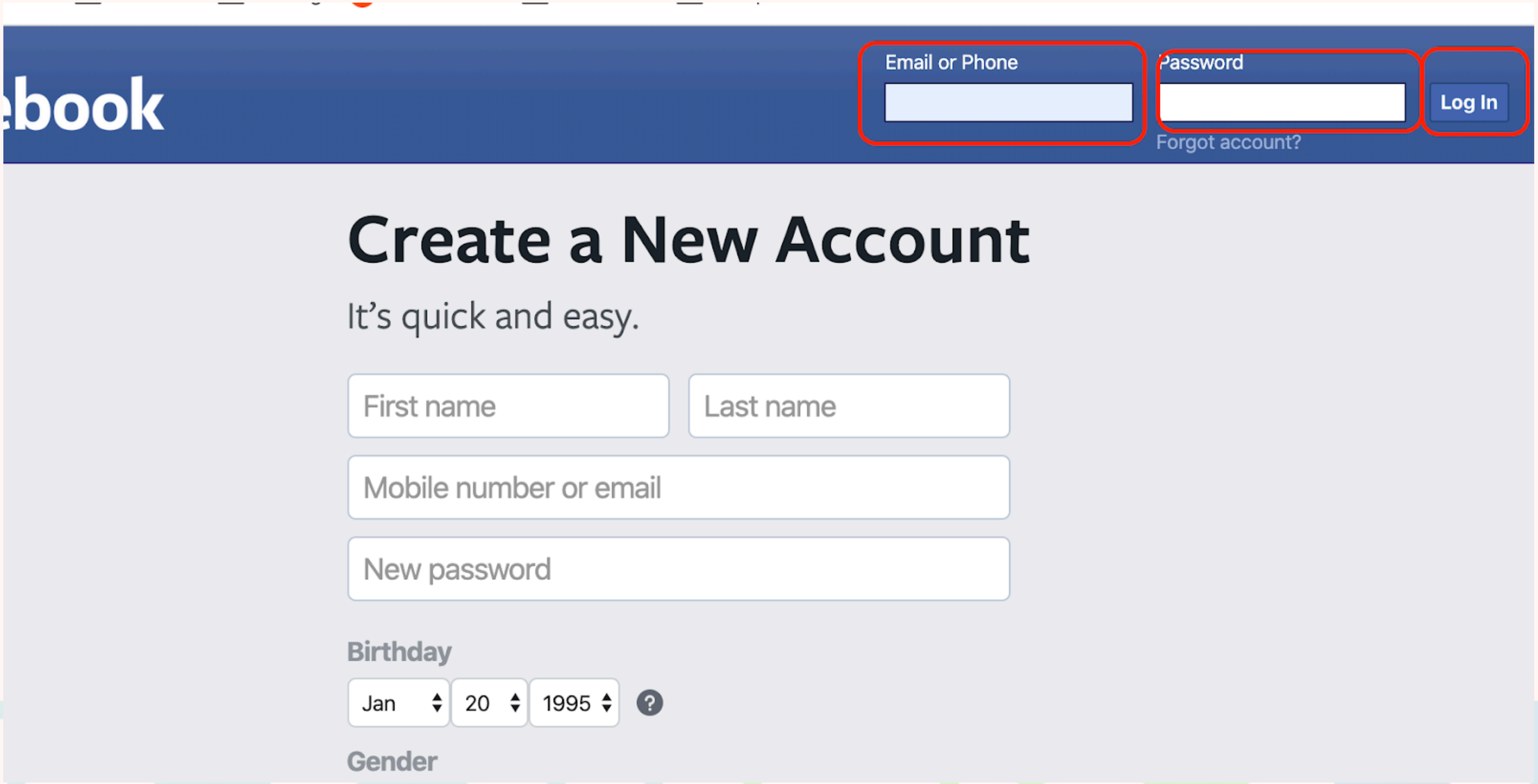
TRADITIONAL WAY TO AUTOMATE

```
public class LogIn_TraditionalWay extends TestBase {

    @Test
    public void LoginTest(){
        driver.get("https://www.facebook.com/");
        driver.findElement(By.id("email")).sendKeys( ...charSequences: "username");
        driver.findElement(By.id("pass")).sendKeys( ...charSequences: "password");
        driver.findElement(By.xpath("locator of the login button")).click();
        Assert.assertTrue(driver.findElement(By.xpath("log in message")).isDisplayed());
    }
}
```

- Are we going to the URL? Is URL hard coded? Yes
- Are we finding the page objects/Webelements in this class? Yes
- Are we doing assertion in this class? Yes

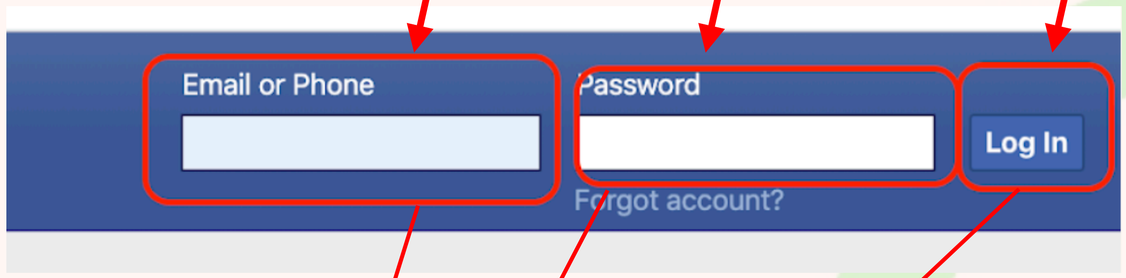
- In Page Page object model, we will not find the page objects in this test class. Because this is a Test Class not a Page class.

IDENTIFY WEB ELEMENT ON THE PAGE



HOW TO CREATE AUTOMATION SCRIPTS IN PAGE OBJECT MODEL

1.PAGE ELEMENTS: EMAIL, PASSWORD, LOGIN BUTTON



- Create the page objects in page classes.
  - We can create major page specific method such as logIn method
- Create Test Classes and use page objects or methods in the test classes

PAGE CLASS

```
public class FaceLoginPage {

    WebDriver driver;
    public FaceLoginPage(WebDriver driver) {//We need th
        this.driver=driver;//to connect the page element
        PageFactory.initElements(driver, page: this);//
    }

    @FindBy(id = "email")
    public WebElement username;

    @FindBy(id = "pass")
    public WebElement password;

    @FindBy(id = "u_0_b")
    public WebElement loginButton;

    public void login(String userid,String pass){
        username.sendKeys( ...charSequences: userid);
        password.sendKeys( ...charSequences: pass);
        loginButton.click();

    }
```

Face Log In Page ELement

Face Log In Page Method

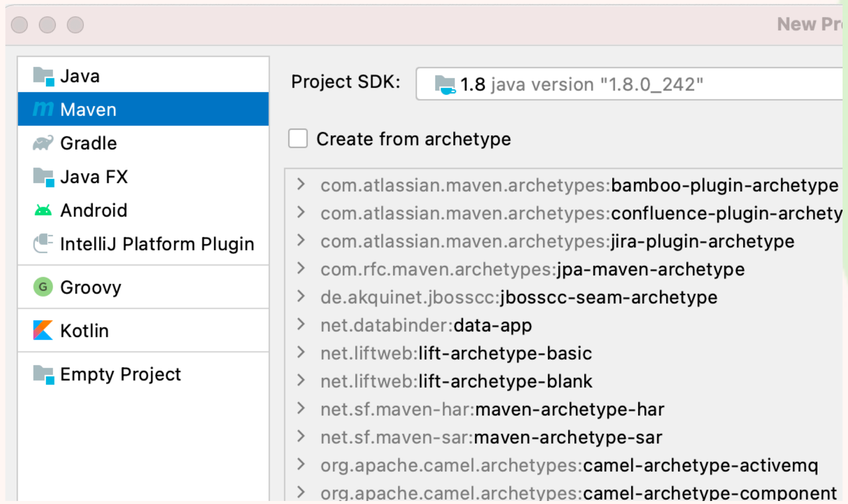
TEST CLASS

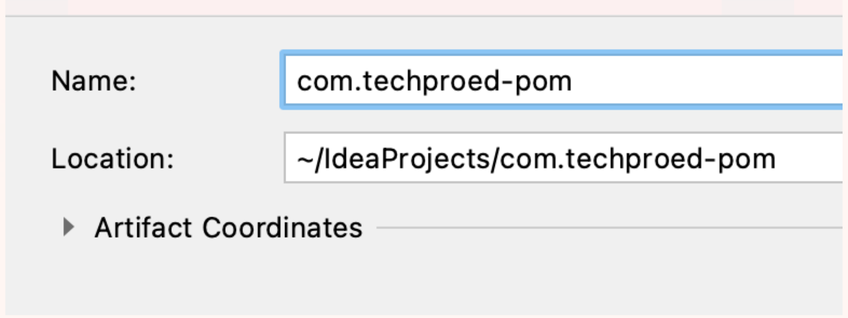
```
public class LogIn_PageObjectModel extends TestBase {

    @Test
    public void loginWithPOM(){
        driver.get("https://www.facebook.com/");
        FaceLoginPage faceLoginPage=new FaceLoginPage(driver);
        faceLoginPage.username.sendKeys( ...charSequences: "user name");
        faceLoginPage.password.sendKeys( ...charSequences: "password");
        faceLoginPage.loginButton.click();
        faceLoginPage.login( userid: "username", pass: "userpassword");

    }
```

HOW TO CREATE A PAGE OBJECT MODEL FRAMEWORK FROM THE SCRATCH?

CREATE PAGE OBJECT MODEL FRAMEWORK AND ADD DEPENDENCIES

- **Create a new project:com.techproed-pom**
- **Add dependencies**

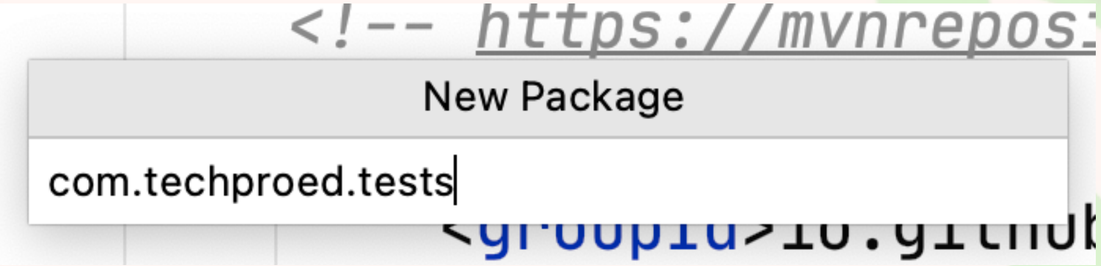


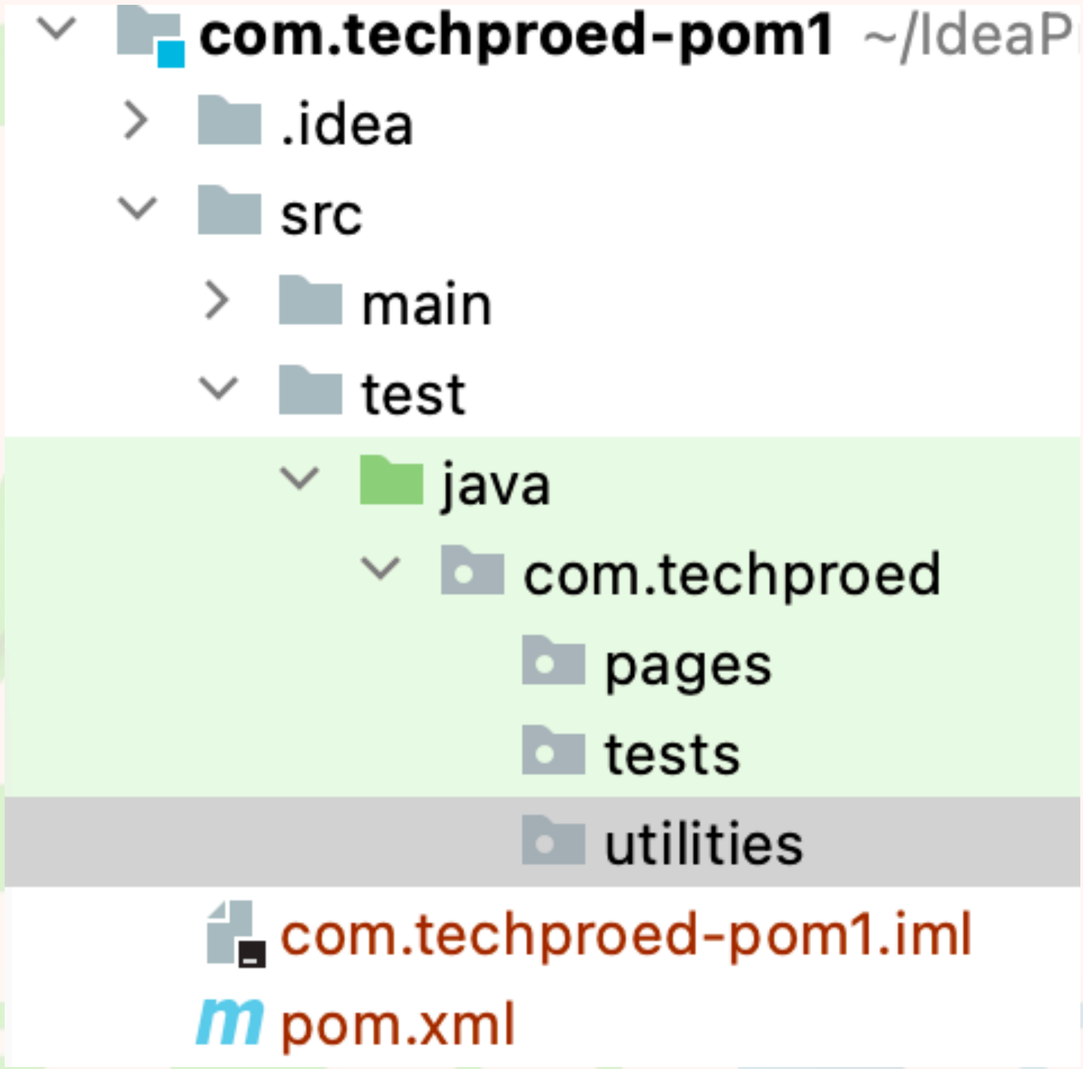



```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-
java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.59</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/
webdrivermanager -->
    <dependency>
        <groupId>io.github.bonigarcia</groupId>
        <artifactId>webdrivermanager</artifactId>
        <version>4.4.3</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.4.0</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

CREATE PACKAGES

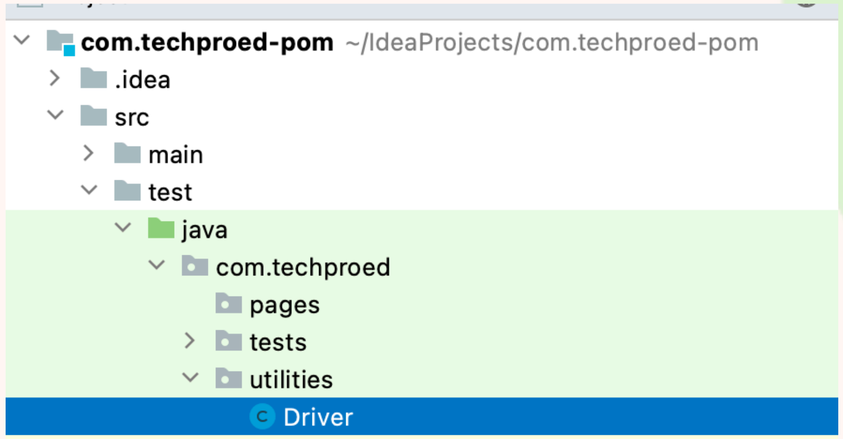
- Create packages
- pages, tests, utilities





CREATE DRIVER CLASS

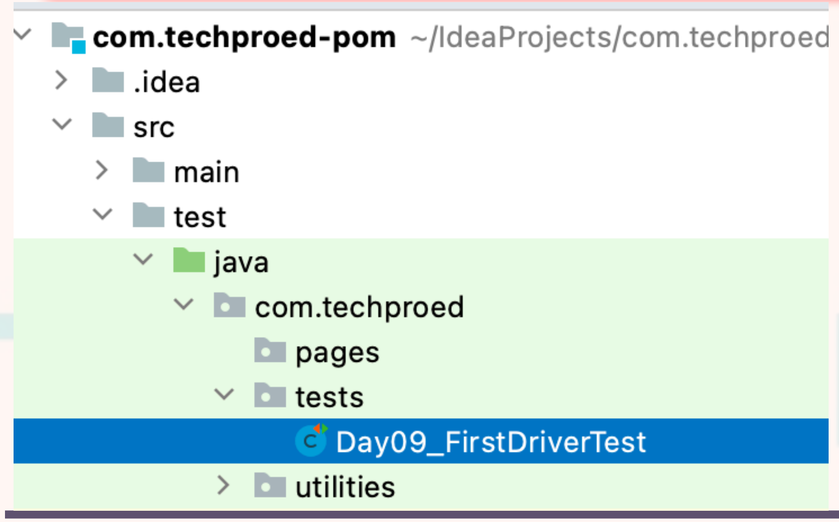
Create Driver Class in utilities package




```java
package com.techproed.utilities;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.concurrent.TimeUnit;

public class Driver {
    //Similar to TestBase, This is a utilities class
    private static WebDriver driver;
    //setup, create, and return the driver instance
    public static WebDriver getDriver(){
        /*
        If driver is not being used, if it is not pointing anywhere, then instantiate
the driver
        We want to use only one driver in the entire framework
         */
        if(driver==null) {
                    WebDriverManager.chromedriver().setup();
                    driver = new ChromeDriver();
        }
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
        driver.manage().window().maximize();
        return driver;
    }//getDriver ends here
    //create closeDriver method to close teh driver
    public static void closeDriver(){
        if (driver!=null) {//if driver is pointing anywhere
            driver.quit();//quit when I call closeDriver
            driver=null;//make the driver null so when we call getDriver, we can open
the driver again
        }
    }
}
```

FIRST DRIVER CLASS

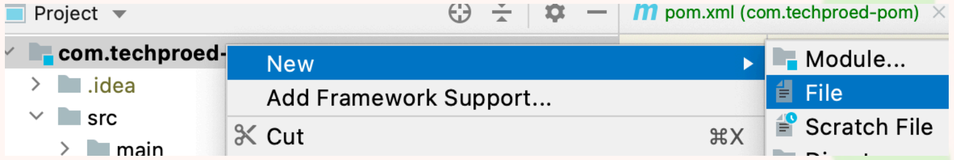
- Create FirstDriverTest class
- Go to amazon page
- Verify the title includes amazon
- Check if Driver class is working

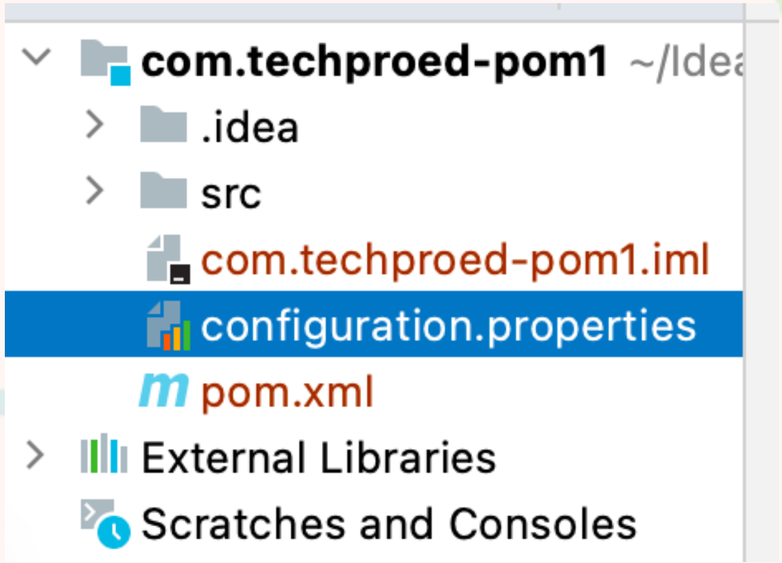



```java
package com.techproed.tests;

import com.techproed.utilities.Driver;
import org.testng.annotations.Test;

public class FirstDriverTest {
    @Test
    public void firstDriverTest(){
        //driver -> Driver.getDriver()
        //driver.get("https://www.amazon.com");
        Driver.getDriver().get("https://www.amazon.com");
//
Driver.getDriver().get(ConfigReader.getProperty("qa_environment"));
    }
}
```

CONFIGURATION PROPERTIES FILE

- Create config properties file on project level:
- configuration.properties
- Add important test data-url,id,...

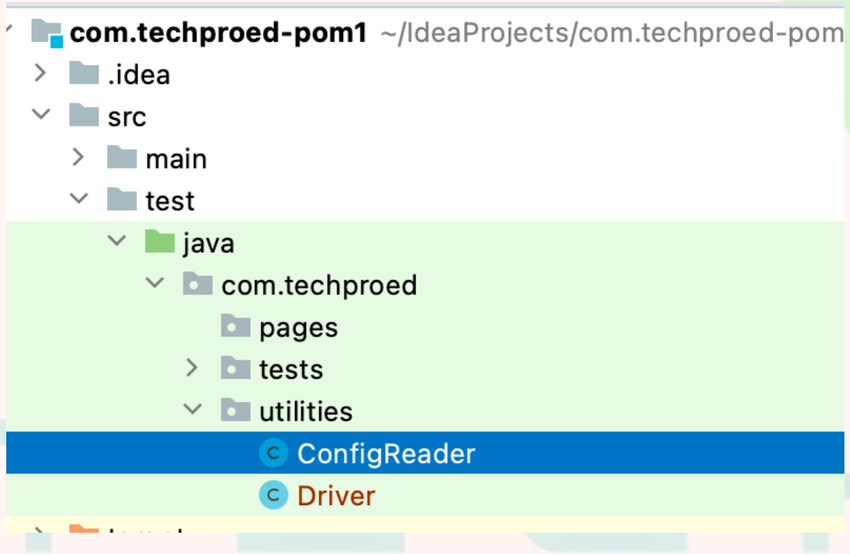





```
#Add major Test Data on this file
qa_environment=https://qa-environment.com
amazon_url=https://www.amazon.com
manager_username=manager123
manager_password=Manager2!!!
browser=chrome
username=john
password=1234
test_email=testtid@gmail.com
test_username=manager
test_phone_number=3442134582
incorrect_username=fakeusername
incorrect_password=fakepass
dblogincreds=asdg!fa
#getProperty(qa_environment)  ->https://qa-environment.com
#getProperty(browser) ->chrome
#getProperty(username) ->john
#driver.get("https://qa-environment.com")  ->
driver.get(getProperty(qa_environment))
#driver.sendKeys("john") ->
driver.sendKeys(getProperty(username))
```

CONFIG READER CLASS

- -Create ConfigReader class in utilities package
- Add important test data-url,id,...




```java
package com.techproed.utilities;

import java.io.FileInputStream;
import java.util.Properties;

public class ConfigReader {
    //This class reads the configuration.properties file
    //Create Properties instance
    private static Properties properties;

    static {
        //path of the configuration file
        String path="configuration.properties";
        try {
            //Opening configuration.properties file using FileInputStream
            FileInputStream fileInputStream = new FileInputStream(path);
            properties = new Properties();
            properties.load(fileInputStream);
            //close the file
            fileInputStream.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    //This method will get the key from properties file,
    //And return the value as String
    //We create this method to read the file
    public static String getProperty(String key){
        String value=properties.getProperty(key);
        return value;
    }

    //TEST IF LOGIC WORKS
//    public static void main(String[] args) {
//        System.out.println(properties.getProperty("qa_environment"));
//    }
}
```

FIRST CONFIG PROPERTIES FILE TEST

- In FirstConfigTest
- Get url from config.properties file
- Get the title from config.properties file

```java
package com.techproed.tests;

import com.techproed.utilities.ConfigReader;
import com.techproed.utilities.Driver;
import org.testng.annotations.Test;

public class Day11_FirstConfigTest {

    @Test
    public void firstConfigTest(){
//        go to app_url
//        Driver.getDriver().get("http://www.carettahotel.com/");
//        ConfigReader.getProperty("app_url")   ===>>>> http://www.carettahotel.com/
        Driver.getDriver().get(ConfigReader.getProperty("app_url"));

//Assert the title equals : Caretta Hotel - Home
        String actualTitle = Driver.getDriver().getTitle();
        String expectedTitle = ConfigReader.getProperty("app_title");
        Assert.assertEquals(actualTitle,expectedTitle);
    }
}
```

FINAL DRIVER CLASS

- We will develop Driver class for all browsers.
- So Before we create driver object we use switch statements to check different browser conditions.
- Put the waits in Driver as we put in TestBase.
- Then close the driver.
- From now on, we don't need to use TestBase class.
- We are using the Driver class
- If you see error, set language level to 7

After Switch, If you see error
Then set language level to 7

```
f(driver==null) {
   //ConfigReader.getProperty("browser") ==>chrome
   switch (ConfigReader.getProperty("browser")) {
       case "chrome":
           WebDriverManage
           driver = new Ch
```

Incompatible types. Found: 'java.lang.String', required: 'byte

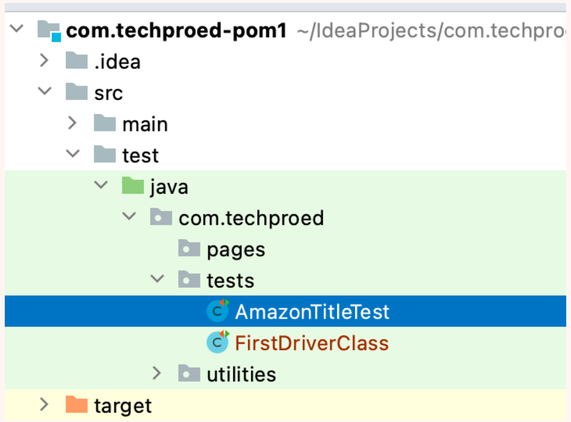
Set language level to 7 - Diamonds, ARM, multi-catch etc. ⌥⇧⏎

com.techproed.utilities.ConfigReader

```java
package com.techproed.utilities;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.firefox.FirefoxDriver;
import java.util.concurrent.TimeUnit;

public class Driver {
    private static WebDriver driver;
    public static WebDriver getDriver(){
        if(driver==null) {
            switch (ConfigReader.getProperty("browser")) {
                case "chrome":
                    WebDriverManager.chromedriver().setup();
                    driver = new ChromeDriver();
                    break;

                case "firefox":
                    WebDriverManager.firefoxdriver().setup();
                    driver = new FirefoxDriver();
                    break;

                case "chrome-headless":
                    WebDriverManager.chromedriver().setup();
                    driver = new ChromeDriver(new ChromeOptions().setHeadless(true));
                    break;
            }
        }
        driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
        driver.manage().window().maximize();
        return driver;
    }//getDriver ends here
    public static void closeDriver(){
        if (driver!=null) {
            driver.quit();
            driver=null;
        }
    }
}
```

FINAL DRIVER AND CONFIG PROPERTIES TEST

- Create a new class: AmazonTitleTest
- When user goes to amazon(get the test data from config file)
- Then verify title includes 'Amazon'




```java
package com.techproed.tests;

import com.techproed.utilities.ConfigReader;
import com.techproed.utilities.Driver;
import org.testng.Assert;
import org.testng.annotations.Test;

public class AmazonTitleTest {
    @Test
    public void amazonTitleTest(){
        Driver.getDriver().get(ConfigReader.getProperty("amazon_url"));
        String amazonTitle=Driver.getDriver().getTitle();
        Assert.assertTrue(amazonTitle.contains("Amazon"));
    }
}
```

FIRST PAGE OBJECT MODEL TEST

PAGE ELEMENTS:

Email
Password
Login button

- http://a.testaddressbook.com/sign_in
- Create the page objects in page classes.
  - We can create major page specific method such as logIn method
- Create Test Classes and use page objects or methods in the test classes

PAGE CLASS

```
package com.techproed.pages;
import com.techproed.utilities.Driver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
public class TestAddressLoginPage {
    public TestAddressLoginPage(){
        PageFactory.initElements(Driver.getDriver(), page: this);
    }
    @FindBy(id = "session_email")
    public WebElement username;

    @FindBy(id = "session_password")
    public WebElement password;

    @FindBy(xpath = "//input[@type='submit']")
    public WebElement signInButton;
}
```

TEST CLASS

```
package com.techproed.tests;

import com.techproed.pages.TestAddressLoginPage;
import com.techproed.utilities.ConfigReader;
import com.techproed.utilities.Driver;
import org.testng.annotations.Test;

public class LoginTest {
    TestAddressLoginPage testAddressLoginPage=new TestAddressLoginPage();
    @Test
    public void loginTest(){
        Driver.getDriver().get("http://a.testaddressbook.com/sign_in");
//      testAddressLoginPage.username.sendKeys("testtechproed@gmail.com");//OR GET USERNAME FROM CONFIG FILE
        testAddressLoginPage.username.sendKeys( ...keysToSend: ConfigReader.getProperty("test_address_username"));
//      testAddressLoginPage.password.sendKeys("Test1234!");//OR GET PASSWORD FROM CONFIG FILE
        testAddressLoginPage.password.sendKeys( ...keysToSend: ConfigReader.getProperty("test_address_password"));
        testAddressLoginPage.signInButton.click();
    }
}
```

SMOKE TEST - POSITIVE TEST

User Story:

- User should be able to login to application with admin profile

Acceptance Criteria:

- Given user tries to login in the application environment using admin profile
- Then verify user should be on default page

Test Case:

- Create a package: smoketest
- Create a class: PositiveTest
- Method: positiveLoginTest
- When user goes to https://qa-environment.resortsline.com
- And click on Log in
- And send the username and password
  - admin
  - Techproed123!