Selenium acts on webelements with the help of their properties such ID, name, XPath, etc. Unlike QTP which has an inbuilt object repository(OR), Selenium has no inbuilt ORs.

Hence we need to build an OR which should also be maintainable and accessible on demand. Page Object Model (POM) is a popular design pattern to create an Object Repository in which each one of those webelements properties are created using a class file.

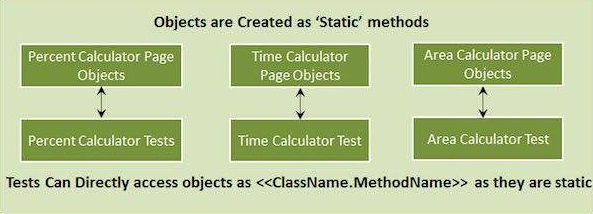
Advantages

* POM is an implementation where test objects and functions are separated from each other, thereby keeping the code clean.
* The objects are kept independent of test scripts. An object can be accessed by one or more test scripts, hence POM helps us to create objects once and use them multiple times.
* Since objects are created once, it is easy to access as well as update a particular property of an object.

POM Flow Diagram

Objects are created for each one of the pages and methods are developed exclusively to access to those objects. Let us use http://calculator.net for understanding the same.

There are various calculators associated with it and each one of those objects in a particular page is created in a separate class file as static methods and they all are accessed through the 'tests' class file in which a static method would be accessing the objects.



Example

Let us understand it by implementing POM for percent calculator test.

**Step 1** − Create a simple class (page\_objects\_perc\_calc.java) file within a package and create methods for each one of those object identifiers as shown below.

package PageObject;

import org.openqa.selenium.\*;

public class PageObjectsPercCalc {

private static WebElement element = null;

// Math Calc Link

public static webElement lnk\_math\_calc(WebDriver driver) {

element = driver.findElement(By.xpath(".//\*[@id = 'menu']/div[3]/a"));

return element;

}

//Percentage Calc Link

public static webElement lnk\_percent\_calc(WebDriver driver) {

element = driver.findElement(By.xpath(".//\*[@id = 'menu']/div[4]/div[3]/a"));

return element;

}

//Number 1 Text Box

public static webElement txt\_num\_1(WebDriver driver) {

element = driver.findElement(By.id("cpar1"));

return element;

}

//Number 2 Text Box

public static webElement txt\_num\_2(WebDriver driver) {

element = driver.findElement(By.id("cpar2"));

return element;

}

//Calculate Button

public static webElement btn\_calc(WebDriver driver) {

element =

driver.findElement(By.xpath(".//\*[@id = 'content']/table/tbody/tr/td[2]/input"));

return element;

}

// Result

public static webElement web\_result(WebDriver driver) {

element =

driver.findElement(By.xpath(".//\*[@id = 'content']/p[2]/span/font/b"));

return element;

}

}

**Step 2** − Create a class with main and import the package and create methods for each one of those object identifiers as shown below.

package PageObject;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class PercentCalculator {

private static WebDriver driver = null;

public static void main(String[] args) {

PageObjectsPercCalc page\_objects\_perc\_calc=**null**;

driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("http://www.calculator.net");

//Use page Object library now

page\_objects\_perc\_calc.lnk\_math\_calc(driver).click();

page\_objects\_perc\_calc.lnk\_percent\_calc(driver).click();

page\_objects\_perc\_calc.txt\_num\_1(driver).clear();

page\_objects\_perc\_calc.txt\_num\_1(driver).sendKeys("10");

page\_objects\_perc\_calc.txt\_num\_2(driver).clear();

page\_objects\_perc\_calc.txt\_num\_2(driver).sendKeys("50");

page\_objects\_perc\_calc.btn\_calc(driver).click();

String result = page\_objects\_perc\_calc.web\_result(driver).getText();

if(result.equals("5")) {

System.out.println(" The Result is Pass");

} else {

System.out.println(" The Result is Fail");

}

driver.close();

}

}

Output

The test is executed and the result is printed in the console. Given below is the snapshot of the same.