

## What is an Object Repository?

An object repository is a common storage location for all objects. In Selenium WebDriver context, objects would typically be the locators used to uniquely identify web elements.

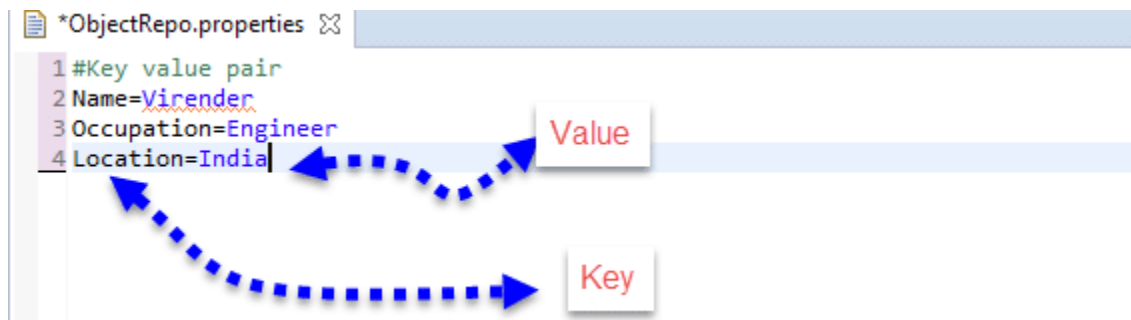
The major advantage of using object repository is the segregation of objects from test cases. If the locator value of one web element changes, only the object repository needs to be changed rather than making changes in all test cases in which the locator has been used. Maintaining an object repository increases the modularity of framework implementation.

## What is an Object repository?

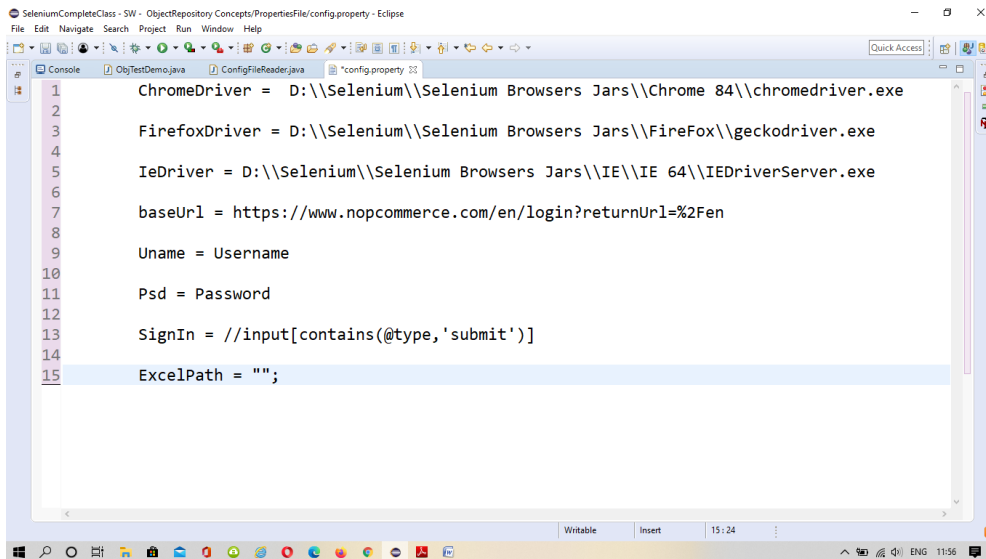
An **Object Repository** is a map between UI element and its locator. Which can also be written as an Object Map between UI element and the way to find it. In *Selenium WebDriver's* context it means a Mapping between *WebElement* and the corresponding *locator type and value*. Lets understand it in more details by looking at a typical WebDriver code which finds an element.

## Object repository in a Properties file

Test page that I have used in this tutorial is “[www.easycalculation.com](http://www.easycalculation.com)” on this page we have three element *First and Last name* field and a link element with “*Partial Link Text*” as the Link text. Lets try to create a repository for these three elements. A property file stores information in a **Key-Value pair**. *Key value pair* is represented by two string values separated by the equal to sign. A typical property file looks like this:



In our case we have to store three values instead of two. **Element Name, Locator type** and the **Locator value**. We will simply separate two values with ‘:’ and use it. Here is a sample object repository file that we will create and will name it **ObjectRepo.properties**.

The screenshot shows the Eclipse IDE interface. The main editor window displays a file named 'config.properties' with the following content:

```
1  ChromeDriver = D:\\Selenium\\Selenium Browsers Jars\\Chrome 84\\chromedriver.exe
2
3  FirefoxDriver = D:\\Selenium\\Selenium Browsers Jars\\FireFox\\geckodriver.exe
4
5  IeDriver = D:\\Selenium\\Selenium Browsers Jars\\IE\\IE 64\\IEDriverServer.exe
6
7  baseUrl = https://www.nopcommerce.com/en/login?returnUrl=%2Fen
8
9  Uname = Username
10
11  Pwd = Password
12
13  SignIn = //input[contains(@type,'submit')]
14
15  ExcelPath = "";
```

Lines 1 through 15 are highlighted in green. The IDE's status bar at the bottom shows 'Writable', 'Insert', and the time '15:24'. The taskbar at the very bottom shows various application icons and the system clock '11:56'.

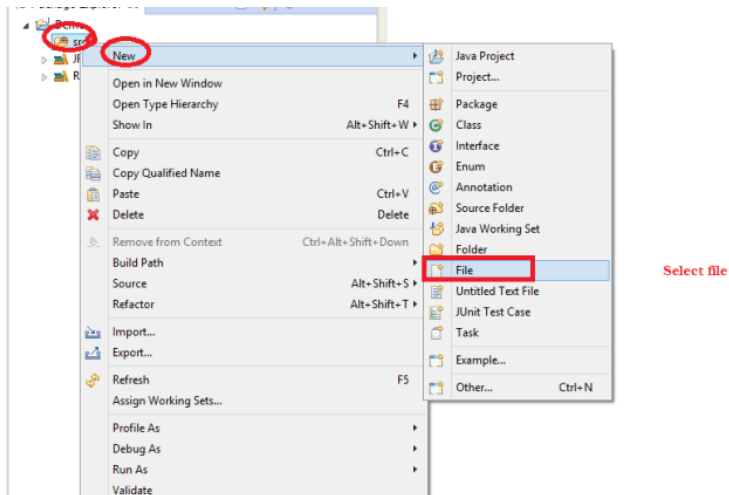
**Note:** In a properties file a comment can be added by using the **# keyword**. As shown above the part that is displayed in green is basically a comment and will not be read by the properties file reader.

## Reading Object Repository properties file

Java has support for the **.properties** files. One can easily read a properties file in three simple steps

1. Create a **FileInputStream** object on the **.properties** file
2. Creating a **Properties** object over the File input stream created in step 1
3. Simply read the **Key-Values** by using the **getProperty("Property name")**; method on **Properties** class.

## Complete Guide to Set Object Repository in Selenium Webdriver



Whenever you talk about repository by the name itself you can think that it is kind of storage. Object repository is the collection of object and object here is locators. Object Repository in Selenium Webdriver is quite easy to design so let's get started.

*Note- Before we start I also want to highlight that people also use the same concept to store all kind of configuration details as well.*

*he example you can keep URL, browser details, some path, port details and so on, in that case, this will behave as the configuration file.*

Here locators mean WebElement id, name, CSS, XPath, class name, LinkText, PartialLinkText, TagName etc.

## Object Repository in Selenium Webdriver

To understand the importance of Object repository, we will take a real-time scenario.

The scenario is you have 100 test cases and in all test cases, you have login scenario. If your application changes now and some locators changes like id changes from email to login-email then you have to modify all the test cases and you have to change the id.

This process does not make any sense and this will be a very tedious task, so to overcome with this we will move all locators in a separate file and we will link all test cases to this file. In case any changes happen in our locators then we will simply change in that file, not all our test cases.

This will increase the modularity of test cases and I will strongly suggest that you should use Object Repository in Selenium Automation.

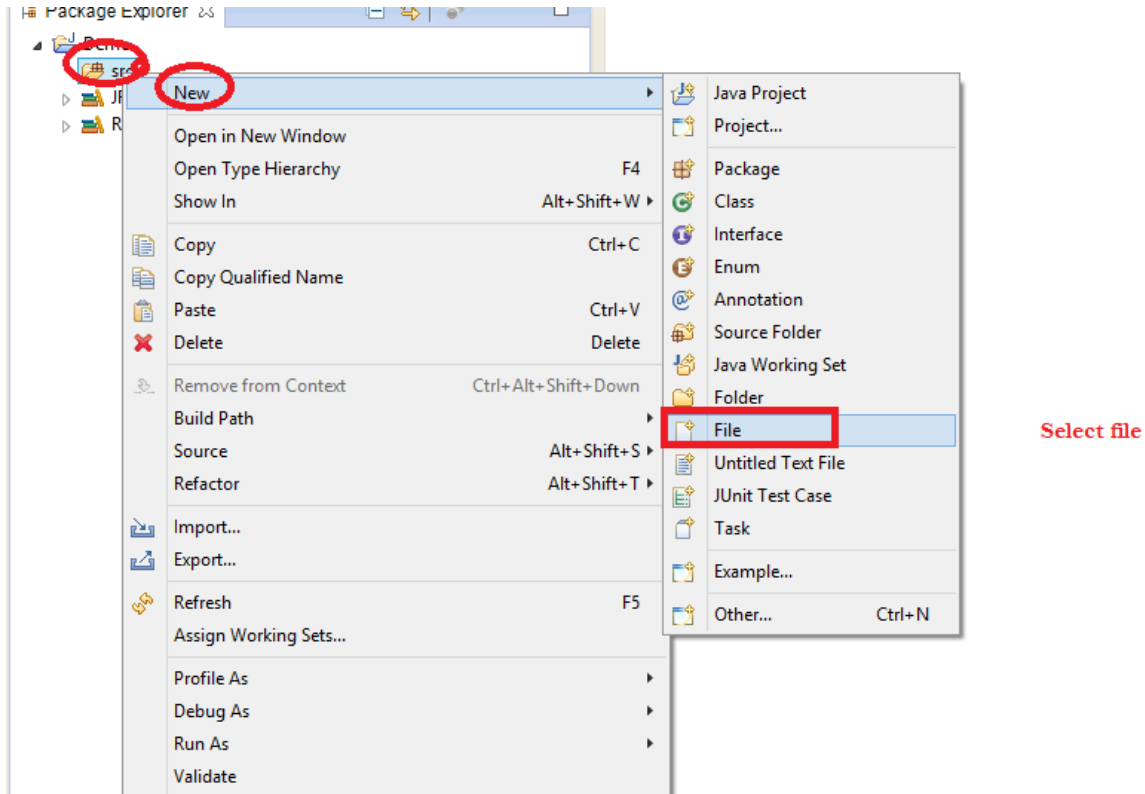
*Note- Many companies also use Page Object Model to store all locators which also make the test in a readable format. Depends on your current framework style you can adopt any of these. Personally, I use Page Object Model using PageFactory which make my test more readable and in terms of performance as well, I felt POM is faster.*

## Object Repository in Selenium Webdriver- Selenium Tutorial

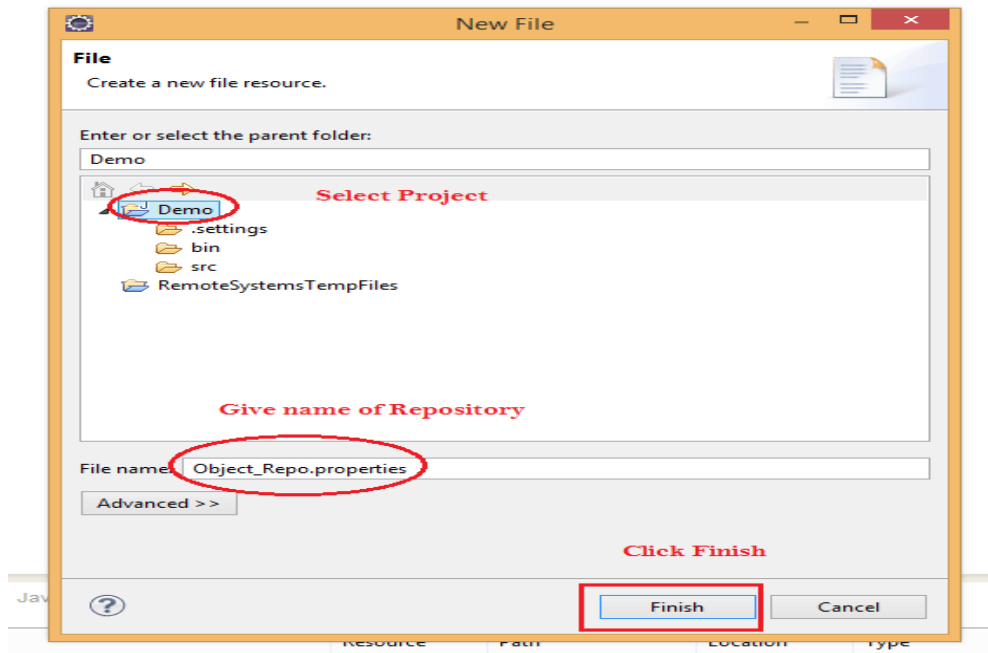
## Precondition

- 1- Project should be created and Selenium jars should be added- [Click here to configure .](#)
- 2- Now create a new file and give file name as Object\_Repo (depends on you)with extension .properties

For this right click on project,> create a new file > Specify name



Select folder and specify name & extension

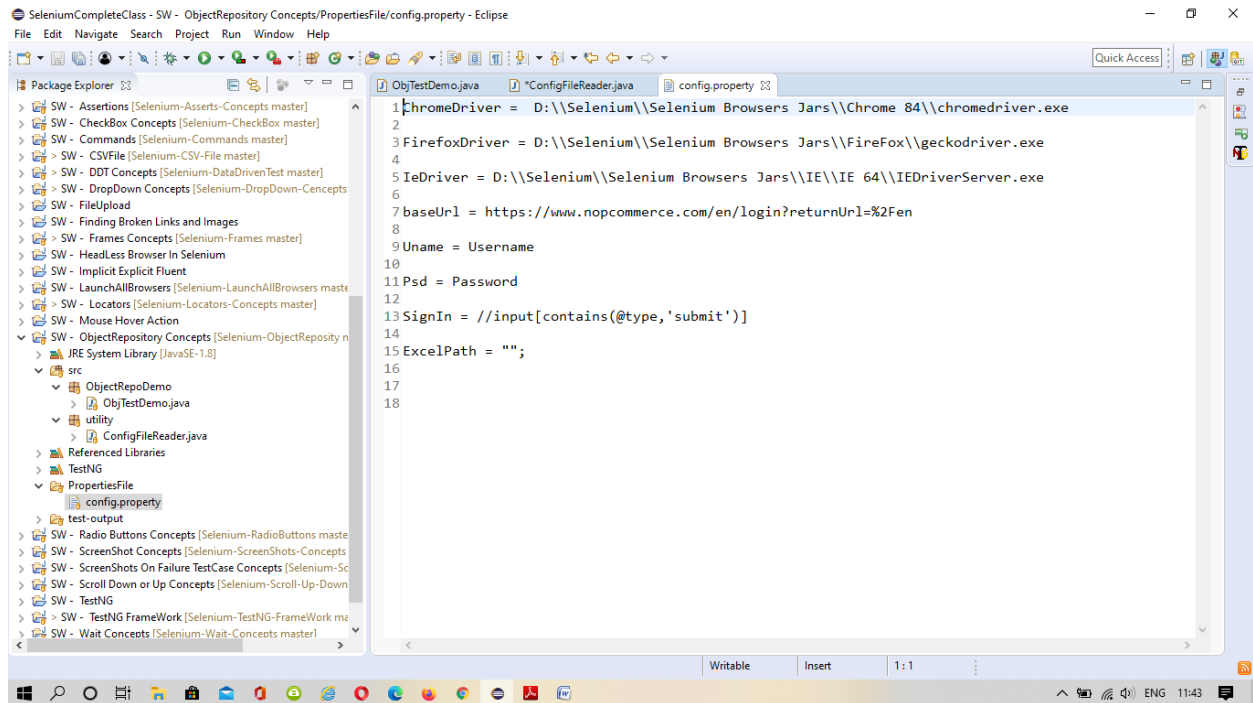


3- Now file is created > Double click on file > File will open in Edit mode

Object repository works on KEY and Value pair so we will specify Keys based on our project and in values we will give locator value.

In below example, I have taken XPath for username, password and login button for the facebook login page.

Key you can write based on your project standard, but value should be correct which is coming from the application



4- Write Selenium script and use the same in the script

## Object Repository in Selenium Webdriver- Selenium Tutorial

package ObjectRepoDemo;

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.Test;
```

```
import utility.ConfigFileReader;
```

```
public class ObjTestDemo {
```

```
    @Test
```

```
    public void ConfigRead() throws Exception {
```

```
        File src = new File("./PropertiesFile/config.property");
        FileInputStream fis= new FileInputStream(src);
        Properties pro = new Properties(); pro.load(fis);
        String chromepath = pro.getProperty("ChromeDriver");
        System.out.println("ChromeBrowser Path is.... " + chromepath);
```

```
        ConfigFileReader config = new ConfigFileReader();
```

```
        System.setProperty("webdriver.chrome.driver", config.getChromePath());
        WebDriver driver = new ChromeDriver();
        driver.get(config.getAppURL());
        driver.manage().window().maximize();
```

```

driver.findElement(By.id(config.UName())).sendKeys(raghavendra.bn@gmail.com);
driver.findElement(By.id(config.PWord())).sendKeys("raghubn@123");
driver.findElement(By.xpath(config.SignButton())).click();

Thread.sleep(5000);
driver.quit();

    }
}

```

**Create Utility package so that you can make use of it.**

```
package utility;
```

```

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

```

```

public class ConfigFileReader {

    Properties pro;

    public ConfigFileReader() {
        try {
            File src = new File("./PropertiesFile/config.property");
            FileInputStream fis = new FileInputStream(src);
            pro = new Properties();
            pro.load(fis);
        } catch (Exception e) {
            System.out.println("Exception is...." + e.getMessage());
        }
    }

    public String getChromePath() {
        String path = pro.getProperty("ChromeDriver");
        return path;
    }

    public String getAppURL() {
        String Url = pro.getProperty("baseUrl"); // Or return pro.getProperty(getAppURL())
        return Url;
    }

    public String UName() {
        String uname = pro.getProperty("Uname");
        return uname;
    }

    public String PWord() {
        String Pswd = pro.getProperty("Psd");
        return Pswd;
    }
}

```

```
public String SignButton() {  
    String button = pro.getProperty("SignIn");  
    return button;  
}  
}
```