Sel Webdriver interacts with browser using json protocol api

1. Page Object Model:

Layer1 : Super class with webdriver initialization, get , maximize other common apis for each page  
        Layer2 : each page objects and apis extends layer1  
        Layer3 : each page test class extends layer1  
        Layer4 : env and config reader  
        Layer5 : excel reader  
        Layer6 : common util like screenshot sendmail etc  
        Layer7 : report generator

Layer1 :

**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.apache.log4j.Logger;

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

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

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

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

**import** org.openqa.selenium.support.events.EventFiringWebDriver;

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

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

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

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

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

**public** **static** EventFiringWebDriver *e\_driver*;

**public** **static** WebEventListener *eventListener*;

**public** TestBase(){

**try** {

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

FileInputStream ip = **new** FileInputStream(System.*getProperty*("user.dir")+ "/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", "/Users/naveenkhunteta/Downloads/chromedriver");

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

}

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

System.*setProperty*("webdriver.gecko.driver", "/Users/naveenkhunteta/Documents/SeleniumServer/geckodriver");

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

}

*e\_driver* = **new** EventFiringWebDriver(*driver*);

// Now create object of EventListerHandler to register it with EventFiringWebDriver

*eventListener* = **new** WebEventListener();

*e\_driver*.register(*eventListener*);

*driver* = *e\_driver*;

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

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

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

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

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

}

}

package com.crm.qa.util;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PURPOSE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- This class implements the WebDriverEventListener, which is included under events.

The purpose of implementing this interface is to override all the methods and define certain useful Log statements

which would be displayed/logged as the application under test is being run.

Do not call any of these methods, instead these methods will be invoked automatically

as an when the action done (click, findBy etc).

\*/

import java.io.IOException;

import org.openqa.selenium.By;

import org.openqa.selenium.OutputType;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.events.WebDriverEventListener;

import com.crm.qa.base.TestBase;

public class WebEventListener extends TestBase implements WebDriverEventListener {

public void beforeNavigateTo(String url, WebDriver driver) {

System.out.println("Before navigating to: '" + url + "'");

}

public void afterNavigateTo(String url, WebDriver driver) {

System.out.println("Navigated to:'" + url + "'");

}

public void beforeChangeValueOf(WebElement element, WebDriver driver) {

System.out.println("Value of the:" + element.toString() + " before any changes made");

}

public void afterChangeValueOf(WebElement element, WebDriver driver) {

System.out.println("Element value changed to: " + element.toString());

}

public void beforeClickOn(WebElement element, WebDriver driver) {

System.out.println("Trying to click on: " + element.toString());

}

public void afterClickOn(WebElement element, WebDriver driver) {

System.out.println("Clicked on: " + element.toString());

}

public void beforeNavigateBack(WebDriver driver) {

System.out.println("Navigating back to previous page");

}

public void afterNavigateBack(WebDriver driver) {

System.out.println("Navigated back to previous page");

}

public void beforeNavigateForward(WebDriver driver) {

System.out.println("Navigating forward to next page");

}

public void afterNavigateForward(WebDriver driver) {

System.out.println("Navigated forward to next page");

}

public void onException(Throwable error, WebDriver driver) {

System.out.println("Exception occured: " + error);

try {

TestUtil.takeScreenshotAtEndOfTest();

} catch (IOException e) {

e.printStackTrace();

}

}

public void beforeFindBy(By by, WebElement element, WebDriver driver) {

System.out.println("Trying to find Element By : " + by.toString());

}

public void afterFindBy(By by, WebElement element, WebDriver driver) {

System.out.println("Found Element By : " + by.toString());

}

/\*

\* non overridden methods of WebListener class

\*/

public void beforeScript(String script, WebDriver driver) {

}

public void afterScript(String script, WebDriver driver) {

}

public void beforeAlertAccept(WebDriver driver) {

// TODO Auto-generated method stub

}

public void afterAlertAccept(WebDriver driver) {

// TODO Auto-generated method stub

}

public void afterAlertDismiss(WebDriver driver) {

// TODO Auto-generated method stub

}

public void beforeAlertDismiss(WebDriver driver) {

// TODO Auto-generated method stub

}

public void beforeNavigateRefresh(WebDriver driver) {

// TODO Auto-generated method stub

}

public void afterNavigateRefresh(WebDriver driver) {

// TODO Auto-generated method stub

}

public void beforeChangeValueOf(WebElement element, WebDriver driver, CharSequence[] keysToSend) {

// TODO Auto-generated method stub

}

public void afterChangeValueOf(WebElement element, WebDriver driver, CharSequence[] keysToSend) {

// TODO Auto-generated method stub

}

public <X> void afterGetScreenshotAs(OutputType<X> arg0, X arg1) {

// TODO Auto-generated method stub

}

public void afterGetText(WebElement arg0, WebDriver arg1, String arg2) {

// TODO Auto-generated method stub

}

public void afterSwitchToWindow(String arg0, WebDriver arg1) {

// TODO Auto-generated method stub

}

public <X> void beforeGetScreenshotAs(OutputType<X> arg0) {

// TODO Auto-generated method stub

}

public void beforeGetText(WebElement arg0, WebDriver arg1) {

// TODO Auto-generated method stub

}

public void beforeSwitchToWindow(String arg0, WebDriver arg1) {

// TODO Auto-generated method stub

}

}

Layer2 : Page Factory

package com.crm.qa.pages;

import org.openqa.selenium.By;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

import org.openqa.selenium.support.PageFactory;

import org.openqa.selenium.support.ui.Select;

import com.crm.qa.base.TestBase;

public class ContactsPage extends TestBase {

@FindBy(xpath = "//td[contains(text(),'Contacts')]")

WebElement contactsLabel;

@FindBy(id="first\_name")

WebElement firstName;

@FindBy(id="surname")

WebElement lastName;

@FindBy(name="client\_lookup")

WebElement company;

@FindBy(xpath = "//input[@type='submit' and @value='Save']")

WebElement saveBtn;

// Initializing the Page Objects:

public ContactsPage() {

PageFactory.initElements(driver, this);

}

public boolean verifyContactsLabel(){

return contactsLabel.isDisplayed();

}

public void selectContactsByName(String name){

driver.findElement(By.xpath("//a[text()='"+name+"']//parent::td[@class='datalistrow']"

+ "//preceding-sibling::td[@class='datalistrow']//input[@name='contact\_id']")).click();

}

public void createNewContact(String title, String ftName, String ltName, String comp){

Select select = new Select(driver.findElement(By.name("title")));

select.selectByVisibleText(title);

firstName.sendKeys(ftName);

lastName.sendKeys(ltName);

company.sendKeys(comp);

saveBtn.click();

}

}

# API Examples

1. Alert

Driver.switchTo().Alert()

Alert.getText(), Alert.accept(), Allert.dismiss(),alert.sendkey(),alert.send

1. Element

Element.isDisplayed(), element.isSelected(), element.isEnabled()

1. Actions -> new Actions(driver)

clickAndHold, moveToElement, release, build, perform,keyUP, keydown

1. WebdriverWait wait = new WebdriverWait(driver, time in miliseconds)

Wait.ignoring(Exception).until(ExpectedConditions.elementToBeClickable..)

Elemnt.click()

1. Select -> new Select(webelement)

select : should have select tag

Select.selectByVisibleText()

Select.selectByIndex()

Select.selectByValue()

Deselect, isMultiple()

1. By

Xpath, id, name, linkText, partialLinkText,cssSelector, className, tagName

1. Navigate

to, back, forward, refresh

1. TakeScreenShot

(TakeScreenShot) driver.getScreenshotAs(Output.File)

1. Listeners

WebDriverEventListener -> Class implements WebDriverEventListener (all before and after api call logs capture and other action can be taken place here)

EventFiringWebDriver

* + Initialize EventFiringWebDriver with webdriver object
  + Register EventFiringWebDriver with class which implemented WebDriverEventListener in first step

1. Exceptions
2. import org.openqa.selenium.NoSuchSessionException; // when no browser available(after close/quit call)
3. import org.openqa.selenium.ElementNotInteractableException; //element is presented in the DOM but it is impossible to interact with such element
4. import org.openqa.selenium.ElementNotSelectableException; //element is presented in the DOM but you can be able to selec
5. import org.openqa.selenium.ImeActivationFailedException; //when IME engine activation has failed.
6. import org.openqa.selenium.ElementNotVisibleException; //existing element in DOM has a feature set as hidden
7. import org.openqa.selenium.ImeNotAvailableException; //when IME support is unavailable.
8. import org.openqa.selenium.InvalidArgumentException; //argument does not belong to the expected type.
9. import org.openqa.selenium.InvalidCookieDomainException; //try to add a cookie under a different domain rather than the current URL.
10. import org.openqa.selenium.InvalidElementStateException; //if a command cannot be finished as the element is invalid.
11. import org.openqa.selenium.InvalidSelectorException;
12. import org.openqa.selenium.JavascriptException; //when executing JavaScript supplied by the user.
13. import org.openqa.selenium.NoAlertPresentException; //when you switch to no presented alert
14. import org.openqa.selenium.NoSuchCookieException; // if there is no cookie matching
15. import org.openqa.selenium.NoSuchElementException; //if an element could not be found.
16. import org.openqa.selenium.NoSuchFrameException; // if frame target to be switch does not exist
17. import org.openqa.selenium.NoSuchWindowException; //if window target to be switch does not exist.
18. import org.openqa.selenium.NotFoundException; //is subclass of WebDriverException. It happens when an element on the DOM does not exis
19. import org.openqa.selenium.ScriptTimeoutException; //when executeAsyncScript takes more time than the given time limit to return the value.
20. import org.openqa.selenium.SessionNotCreatedException; //A new session could not be successfully created
21. import org.openqa.selenium.StaleElementReferenceException; //if the web element is detached from the current DOM.
22. import org.openqa.selenium.TimeoutException; //when there is not enough time for a command to be completed.
23. import org.openqa.selenium.UnableToSetCookieException; //if a driver is unable to set a cookie
24. import org.openqa.selenium.UnhandledAlertException; //when there is an alert, but WebDriver is unable to perform Alert operation.
25. import org.openqa.selenium.UnsupportedCommandException; // when remote WebDriver does not send valid command as expected.
26. import org.openqa.selenium.WebDriverException; //when the WebDriver is performing the action right after you close the browser.
27. import org.openqa.selenium.safari.ConnectionClosedException; //disconnection in the driver.
28. import org.openqa.selenium.remote.ErrorHandler.UnknownServerException; //server returns an error without a stack trace
29. import org.openqa.selenium.interactions.InvalidCoordinatesException; //if the coordinates offered to an interacting operation are not valid.
30. import org.openqa.selenium.interactions.MoveTargetOutOfBoundsException; //if the target provided to the ActionChains move() methodology is not valid.
31. import org.openqa.selenium.remote.JsonException; //when you afford to get the session capabilities where the session is not created.
32. import org.openqa.selenium.remote.ScreenshotException; //It is impossible to capture a screen
33. import org.openqa.selenium.remote.UnreachableBrowserException; //if the browser is unable to be opened or have crashed because of some reasons.
34. import org.openqa.selenium.firefox.UnableToCreateProfileException; //sometimes a new version of Selenium driverserver or browser may not support the profiles
35. import org.openqa.selenium.support.ui.UnexpectedTagNameException; //if a support class did not get a web element as expected.
36. Driver exe set at environment variable and not req to set in code

<http://toolsqa.com/selenium-webdriver/running-tests-in-chrome-browser/>

1. Upload

Should have attribute type=”File” and use sendkeys to upload

1. sendKeys

To type character in upper case : Sendkeys(keys.shift, "value")

1. Get and navigate

The difference between these two methods comes not from their behavior, but from the behavior in the way the application works and how browser deal with it.

navigate().to() navigates to the page by changing the URL like doing forward/backward navigation.

Whereas, get() refreshes the page to changing the URL.

So, in cases where application domain changes, both the method behaves similarly. That is, page is refreshed in both the cases. But, in single-page applications, while navigate().to() do not refreshes the page, get() do.

Moreover, this is the reason browser history is getting lost when get() is used due to application being refreshed.

**driver.get() :** It's used to go to the particular website , But it doesn't maintain the browser History and cookies

**driver.navigate() :** it's used to go to the particular website , but it maintains the browser history and cookies, so we can use forward and backward button to navigate between the pages during the coding of Testcase

1. Proxy

//with url

/\*In case your authentication server requires username with domain like “domainuser” you need to add double slash ‘\‘ to the Url.

http://localdomain\user:password@example.com\*/

WebDriver driver1 = **new** FirefoxDriver();

driver1.get("http://UserName:Password@Example.com");

//Step to craete profile manually

//http://toolsqa.com/selenium-webdriver/custom-firefox-profile/

ProfilesIni profileini = **new** ProfilesIni();

FirefoxProfile myprofile = profileini.getProfile("profileToolsQA");

driver1 = **new** FirefoxDriver(myprofile);

String proxy = "localhost:7777";

Proxy p =**new** Proxy();

p.setHttpProxy(proxy);

p.setFtpProxy(proxy);

p.setSslProxy(proxy);

DesiredCapabilities cap = **new** DesiredCapabilities();

cap.setCapability(CapabilityType.***PROXY***, p);

driver1 = **new** FirefoxDriver(cap);

FirefoxProfile profile = **new** FirefoxProfile();

profile.setPreference("network.proxy.type", 1);

profile.setPreference("network.proxy.http", "proxy.domain.example.com");

profile.setPreference("network.proxy.http\_port", 8080);

profile.setPreference("network.proxy.ssl", "proxy.domain.example.com");

profile.setPreference("network.proxy.ssl\_port", 8080);

driver1 = **new** FirefoxDriver(profile);

FirefoxProfile profile1 = **new** FirefoxProfile();

profile1.setPreference("network.proxy.type", ProxyType.***AUTODETECT***.ordinal());

WebDriver driver = **new** FirefoxDriver(profile1);

/\*0 - Direct connection (or) no proxy.

1 - Manual proxy configuration

2 - Proxy auto-configuration (PAC).

4 - Auto-detect proxy settings.

5 - Use system proxy settings.\*/

//For PAC based urls

Proxy pacproxy = **new** Proxy();

pacproxy.setProxyType(Proxy.ProxyType.***PAC***);

pacproxy.setProxyAutoconfigUrl("http://some-server/staging.pac");

DesiredCapabilities capabilities = **new** DesiredCapabilities();

capabilities.setCapability(CapabilityType.***PROXY***, proxy);

//In case if you have an autoconfig URL -

FirefoxProfile firefoxProfile = **new** FirefoxProfile();

firefoxProfile.setPreference("network.proxy.type", 2);

firefoxProfile.setPreference("network.proxy.autoconfig\_url", "http://www.etc.com/wpad.dat");

firefoxProfile.setPreference("network.proxy.no\_proxies\_on", "localhost");

//set the WebDriver to Auto Detect Proxy

Proxy autodetectProxy = **new** Proxy();

autodetectProxy.setAutodetect(**true**);

cap = **new** DesiredCapabilities();

cap.setCapability(CapabilityType.***PROXY***, proxy);

//For Chrome

ChromeOptions opt = **new** ChromeOptions();

opt.addArguments("--proxy-server=http://user:password@yourProxyServer.com:8080");

WebDriver driver2 = **new** ChromeDriver(opt);

DesiredCapabilities capabilities1 = DesiredCapabilities.*chrome*();

capabilities1.setCapability("chrome.switches", Arrays.*asList*("--proxy-server=http://user:password@proxy.com:8080"));

WebDriver driver3 = **new** ChromeDriver(capabilities1);

opt.addArguments("--no-proxy-server");

Edureka.com/testing-with-selenium-webdriver

Udemy.com

Naveen automation labs

* Page refresh in many ways  
               Get , navigate refresh , navigate to url
* Findelements returns 0 list if elent nt found  
  Findelement returns exception
* Handle hidden elements  
               Use javascript executor
* No : captcha images ,bar code readings, window app ,bit map comp ,pdf comp, image validation,third party calenders, webservices
* Play on chrome -> inspect->CTRL+F, <https://www.freecrm.com/> play on Calendars for xpath attributes, text(), contains , parent::, preceding-sibling:: , descendant::, forward-sibling, on freecrm site, //\*[@id='firstname'] -> \* any tag with id attribute