1INSURER Selenium Guide

Instruction guide for Selenium Automation.

Prepared by: James Pinchen

Confidential Document

October 2017

REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VERSION | STATUS | DATE | BY | SUMMARY |
| 0.1 | Draft |  | M.Tookey | Draft |

SIGN OFF

|  |  |  |  |
| --- | --- | --- | --- |
| VERSION | DATE | BY | SIGNATURE |
|  |  |  |  |

CONTENTS

[1. Architecture design 3](#_Toc495658353)

[1.1.1. Overview 3](#_Toc495658354)

[1.1.2. Abstraction Design 4](#_Toc495658355)

[1.1.2.1. Area Break Down 4](#_Toc495658356)

[1.1.3. Page Objects 4](#_Toc495658357)

[1.1.3.1. Object Identification Hierarchy 4](#_Toc495658358)

[1.1.3.2. Page Factory 4](#_Toc495658359)

[1.1.3.3. Standard Object Repository 5](#_Toc495658360)

[1.1.3.4. Object Repository name conventions 5](#_Toc495658361)

[1.2. Pages 6](#_Toc495658362)

[1.2.1.1. Layout 6](#_Toc495658363)

[1.2.1.2. The page naming. 6](#_Toc495658364)

[1.2.1.3. Methods 6](#_Toc495658365)

[1.2.1.4. Constructors 7](#_Toc495658366)

[1.2.1.5. Variable coding standards 7](#_Toc495658367)

[1.2.1.6. Naming Standards for Constants 7](#_Toc495658368)

[1.3. Resources 7](#_Toc495658369)

[1.4. Misc 8](#_Toc495658370)

[1.5. Code hints. 8](#_Toc495658371)

[1.5.1. XPATH for identical buttons: 8](#_Toc495658372)

[1.5.2. Wait until 8](#_Toc495658373)

[1.5.3. Dealing with popup boxes 8](#_Toc495658374)

# Architecture design

### Overview

* The 1insurer Selenium mission statement is when in doubt “keep it simple”. The architecture will evolve over time to adapt to the nuances of the 1i product suite, however, where possible simple solutions should be utilised.
* The scripts are to be designed so that they can be run over multiple machines. (Build, Dev Local Machines, Sales environment).
* Each regression script JUNIT needs to be designed to be run in a heads down standalone manor. There should be no chaining of scripts feeding data into additional scripts. This approach follows best practice design regarding automation and allows for parallel script deployment.
* Individual script run length should not exceed 15 minutes. Combining this approach with a parallel deployment will allow for an efficient automated environment validation.
* Data mining can be used to parametrise scripts, also setup data can also be inserted via SQL if required.
* Where possible random data should be used to drive fields. If the tester is going to automate the name field then there is more value in using a random length field up to the max field length rather than hardcoding in the name “Kevin”.
* All controls should be parametrised either locally or from the regression layer. For example, selecting a dropdown list of title “Mr, Mrs, Master” should never be hardcoded as an option on the page method.
* Spend time making the objects flexible rather than inflexible with a desire to refactor later.
* The scripts will be architected for an enterprise sized application following best practices of Selenium automation. For example, EXCEL is not an application that should be used to drive an enterprise level automation framework, as it is not reliable enough for deployment on a server.
* 1 insurer will be following a Page Object Model using Page Factory.
* The code will be written in JAVA currently JAVA 8.
* Our scripts will support running over FireFox Chrome & IE.

### Abstraction Design



#### Area Break Down

* Regression: (End to End processes designed by the product owner). No coded business logic.
* Feature Regression: (Specific regression tests for work done within the scrum). No coded business logic.
* Pages: This is the business logic layer.
* Objects: This contains the properties of the object for identification.

### Page Objects

#### Object Identification Hierarchy

Objects should be identified in the following preference:

1. ID
2. Name
3. CSS
4. Link Text
5. XPATH (the XPATH search takes half a second longer than ID)

#### Page Factory

* Most objects will be declared using Selenium’s page factory function.
* Page factory uses Caching to speed up the run time.
* Only static objects should be cached with the @CacheLookup tag.

***@CacheLookup***

***@FindBy***(how = *How*.***ID***, using = "SearchType")

**public** **static** WebElement *btnSearchType*;

* The object should follow standard camel case notation with the object type prefixed e.g. btn for button.

#### Standard Object Repository

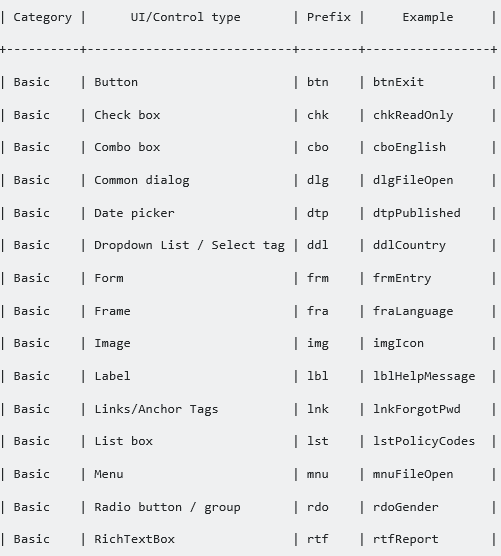
In the case where the Page Factory cannot determine the object then a standard approach can be taken:

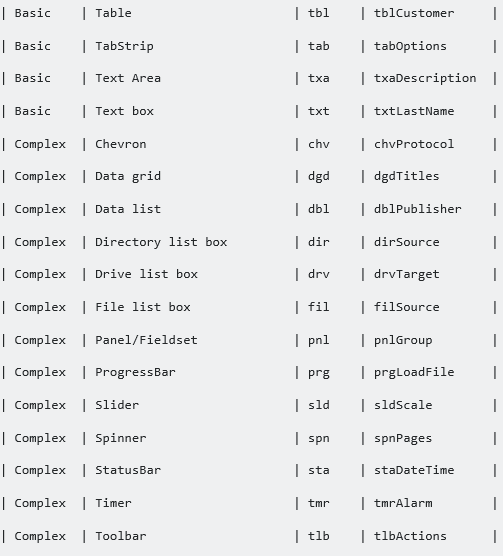
* By ddlTitle = By.id ("ClientAdd\_NameTitle\_1a");

The object names should follow the same format as the page factory.

This should only be used as a last resort if all other page factory identification approaches have failed.

#### Object Repository name conventions





## Pages

#### Layout

Pages are split into four folders:

* client
* claims
* policy
* common
* vendor

#### The page naming.

* This should reflect the html name and be preceded with folder e.g. “PolicyClientSearch”

#### Methods

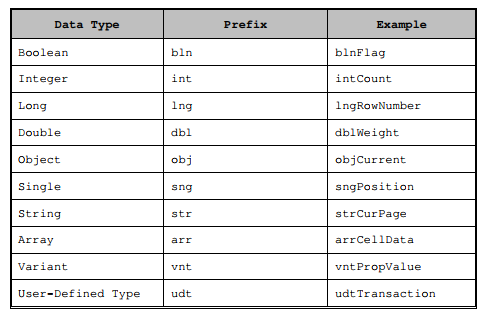
* The methods should be named in a camel Case notation with a verb as the first word. “selectTitle” or “typeFirstName”
* Methods should have a comment explaining the purpose of the method. However, in the example of a method called selectTitle that just contains one line of code then there is no need for a comment.

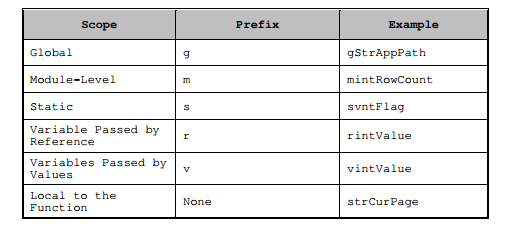
#### Constructors

* These should appear as the first method on the page.
* The method name should be the page name not in camel case.

#### Variable coding standards

* Variables in pages should conform to the standard naming convention.





#### Naming Standards for Constants

The constant names should be initial capped with underscores between words as shown in the following example.

Example: gstrApplication\_Path

## Regression.

* Regression scripts will contain no code logic they will just refer to pages.
* Each page load with have a selenium waiter call before it to make sure the page has fully loaded.
* Each page will have a start and stop stopwatch function.
* The regression script should start with loading a new browser and end with the browser loading.
* All regression scripts will be standalone. (There will be no chaining of data.)

## Resources

* This should contain XML files holding any data that needs to be set.

## Misc

* This should contain methods with generic functions.
  + XML Data reader.
  + SQL Data reader
  + String functions like a random string builder.

## Code formatting.

* 1insurers standard JAVA coding format should be adhered too and can be done automatically by holding down “control alt f”.
* Control S should be used to save the page as this will remove any unused imports.

## Code hints.

### XPATH for identical buttons:

Use the index value:

driver.findElement (By.*xpath* ("(//\*[@id='btnSubmitActions'])[1]")).click ();

driver.findElement (By.*xpath* ("(//\*[@id='btnSubmitActions'])[2]")).click ();

driver.findElement (By.*xpath* ("(//\*[@id='btnSubmitActions'])[3]")).click ();

### Wait until

WebDriverWait wait= new WebDriverWait(this.driver,20 );

wait.until(ExpectedConditions.visibilityOf(OR\_XXX.XXX));

### Dealing with popup boxes

**try** {

            //**WebDriverWait** **wait** = **new** WebDriverWait(*driver*, 2);

            //wait.until(**ExpectedConditions**.*alertIsPresent*());

            Alert **alert** = *driver*.switchTo().alert();

            alert.accept();

        } **catch** (**Exception** **e**) {

            //exception handling

        }