Icon

Description automatically generated

**Test Automation Coding Standards**

version 0.1

25 October 2022

Table of Contents

[1.0 Variables 3](#_Toc117784933)

[2.0 Classes & Packages 4](#_Toc117784934)

[3.0 Methods 5](#_Toc117784935)

[4.0 Miscellaneous 6](#_Toc117784936)

# Variables

The following are the set of standards that need to be followed during automation script development phase when naming and using variables

* The variable names should be in camel case and should have the variable type as well

**Incorrect:**

public WebElement SubmitButton;

public WebElement submitbutton;

public WebElement submit;

public WebElement agencylink;

public WebElement AgencyLink;

public WebElement lnkagency;

String text;

int wt;

**Correct:**

public WebElement submitBtn;

public WebElement agencyLink;

String Text;

int Weight;

* The names should be meaningful and descriptive and should not be random letters. From a third person standpoint oe,os,ob,oh,od means nothing

**Incorrect:**

int i;

Organization o;

String asdgfh;

**Correct:**

int Iterator;

Organization organization; (or) Organization org;

String tempText;

* If the variable name contains multiple word than words are separated with ( \_ ) underscore symbol

**Incorrect:**

public WebElement addagencylink;

**Correct:**

public WebElement add\_Agency\_Link;

* Use single letter identifiers only for the counter in loops
* use UPPER\_CASE for constants - final variables

final int DAYS\_IN\_YEAR = 365;

* Do not use embedded assignments in an attempt to improve run-time performance. This is the job of the compiler.

**Incorrect:**

d = (a = b + c) + r;

**Correct:**

a = b + c;

d = a + r;

* While defining xpaths, do not use static numerical references and instead use contains.

**Incorrect:**

@FindBy(xpath=”//div[@class=’col-tn12 col-xs-6 price’]//p”)

public WebElement priceText;

**Correct:**

@FindBy(xpath=”//div[contains(@class,’price’)]//p”)

public WebElement priceText;

# 2.0 Classes & Packages

The following are the set of standards that need to be followed during automation script development phase when naming and using classes & packages

* Class names start with an upper-case letter.
* Every class should be preceded with a descriptive comment using the "JavaDoc" notational convention. The comment should describe its purpose of the class.

/\*\*

\* Stores the first, middle, and last names for a president.

\*/

class President {

//code...

}

* Packages are all lower case.
* Do not change the class path entries as it is fixed for the project
* All Test Classes should have setup method preceded by @BeforeMethod annotation and

teardown method preceded by @AfterMethod annotation

* All Page Classes should have constructors that instantiates the driver object
* All Test Classes should be extending the BaseTest class and all Page Classes should be extending BasePage class

# 3.0 Methods

The following are the set of standards that need to be followed during automation script development phase when naming and using methods

* Use one blank line to separate each method within a class definition. Use one blank line to separate logical sections of code within a method.
* Every method definition should be preceded with a descriptive comment using the "Javadoc" notational convention.

/\*\*

\* @Method Use :

\* @Method Name :

\* @Input Parameters :

\* @Output Parameters :

\* @Modified By :

\* @Modified Date :

\*\*/

* Include try catch block in every page and test method instead of throws method
* Follow proper indentation. This increases readability and helps in maintainability.
* All else statements where we intend to fail the test case cannot have ExtentTestManager.createInfoTestStepWithScreenshot and should have ExtentTestManager.createAssertTestStepWithScreenshot with fail status
* Test Method name should completely capture the working of the test case and should not be something like login, validate. Etc what are we validating? We re logging in to do what? think from these perspectives and name the test methods.
* Method names start with a lower-case letter and multi-word identifiers are internally capitalized. Do not use hyphens or underscores to separate multi-word identifiers (except for constants, which have all upper-case letters).
* In-line comments should be used to explain complicated sections of code, such as loops. Use the // comment delimiter for in-line comments. Do not comment generally known features of the java language.
* Separate all binary operators, such as "+", "-", "\*", "/", "%", etc., with a single space. The exception is unary operators, such as "++", "--", unary minus "-", etc, which do not need to be separated with a single space.

**Incorrect:**

public static void main(String[] arg){

System.out.println("Hello"+" "+"World");

}

**Correct:**

public static void main(String[] arg) {

System.out.println("Hello" + " " + "World");

}

* Never use @SupressWarnings for any of the page or test methods. Most warnings represent bad programming practices or potential problems that the compiler is not able to resolve
* All the variables that are to be used within the method should be declared first at the top and if the variables are used to fetch the test data, they should be declared and defined at the top section of the method

**Incorrect:**

Public void getTextMethod() {

String username=TestData.getProperty(“username”);

loginPage.enterUserName(username);

String password=TestData.getProperty(“password”);

loginPage.enterPassword(password);

}

**Correct:**

Public void getTextMethod() {

//Test Data section

String username=TestData.getProperty(“username”);

String password=TestData.getProperty(“password”);

loginPage.enterUserName(username);

loginPage.enterPassword(password);

}

# 4.0 Miscellaneous

* Whenever you are creating or adding any new test or page methods or new locator or test data entry, add it at the end of the existing file where the last method or locator of test data row ends. It will help in reducing the merge conflicts in Git.
* Never hard code the test data. For every field that uses externalized data, there should be a field in the test data file
* Do not use implicit wait as it will reduce the performance of the overall automation script execution. Instead use explicit or fluent waits
* Whenever you are committing and pushing your changes to Git do not include the following files. Instead add these files in the .gitignore file
  + /target/
  + /bin/
  + /test-output/\*
  + /test-output/\*.\*
  + /logs/
  + /Screenshots/\*
  + /.DS\_Store
  + /debug.log
  + /.classpath
  + /.settings
  + .idea/
* Never use external Jar dependencies. All the libraries required for the execution of the script should be mentioned in the POM file and should come through it.