mirra SM

Testing – Automation Framework Guidelines

Enterprise Technology Group

Version – 1.0.0

Revision History

Date	Version	Comments	Created By
June 3rd 2022	1.0	Initial	Sreenivasulu Boyapati

1. Introduction

This document provides details about QA automation framework and the complete flow of project structure. In addition, it contains all the naming conventions used in a project, Script writing best practices and some other additional references.

All QA automation testers must follow the guidelines defined in this document to avoid any deviations from the standard processes.

2. Naming conventions

2.1 Project name

Project name must follow given naming convention.

Syntax: OrgName.Process.AutionType.ProjectName

Example: Mirra.Automation.UI.CA

2.2 Feature files

Feature file names must begin with upper case letters and the remaining words (if Applicable) should have even upper-case letters followed by ".feature".

Example: Login.feature

2.3 Methods

Method names must begin with lower case letters and the remaining words should have upper case letters i.e., **Camel Casing**.

Example: userRegistrationProcess()

2.4 Variables

Follow same as "Method naming convention".

Examples: name, txtUsername, txtPassword

2.5 Properties file variables

Follow same as "Method naming convention".

Examples: url, username

2.6 Classes

Class names must begin with upper case letters and the remaining words should have even upper-case letters i.e., **Pascal Casing**.

Examples: SeleniumUtils

Notes:

- ✓ Page object class names must end with the word "Page"
  Example: ProfessionalClaimsPage
- ✓ Step Definition class names must end with the word "StepDef"
  Example: ProfessionalClaimsStepDef

2.7 Web elements naming convention

Follow given table based on the type of web element.

Category	UI/Control Type	Prefix	Example
Basic	Button	btn	btnExit
Basic	Check box	chk	chkReadOnly
Basic	Combo box	cbo	cboEnglish
Basic	Common dialog	dlg	dlgFileOpen
Basic	Date picker	dtp	dtpPublished
Basic	Dropdown List / Select tag	ddl	ddlCountry
Basic	Form	frm	frmEntry
Basic	Frame	fra	fraLanguage
Basic	Image	img	imgIcon
Basic	Label	lbl	lblHelpMessage
Basic	Links/Anchor Tags	lnk	lnkForgotPwd
Basic	List box	lst	lstPolicyCodes
Basic	Menu	mnu	mnuFileOpen
Basic	Radio button / group	rdo	rdoGender
Basic	RichTextBox	rtf	rtfReport
Basic	Table	tbl	tblCustomer
Basic	TabStrip	tab	tabOptions
Basic	Text Area	txa	txaDescription
Basic	Text box	txt	txtLastName
Complex	Chevron	chv	chvProtocol
Complex	Data grid	dgd	dgdTitles
Complex	Data list	dbl	dblPublisher
Complex	Directory list box	dir	dirSource
Complex	Drive list box	drv	drvTarget
Complex	File list box	fil	filSource
Complex	Panel/Fieldset	pnl	pnlGroup
Complex	ProgressBar	prg	prgLoadFile
Complex	Slider	sld	sldScale
Complex	Spinner	spn	spnPages
Complex	StatusBar	sta	staDateTime
Complex	Timer	tmr	tmrAlarm
Complex	Toolbar	tlb	tlbActions
Complex	TreeView	tre	treOrganization

3. Script writing best practices

3.1 Locators order

While identifying locators the order must be followed as mentioned here.

ID -------> Name ----------> Class ----------> TagName --------> CSS ---------> Xpath

3.2 Waits

Must use "Explicit Waits" based on wherever they are needed.

Note – Use "Thread.Sleep" if none of the explicit wait methods work.

3.3 Try and catch

Use "try and catch" for any newly created methods with a block of code.

3.4 Test data

DO NOT hard code test data. Read it from "Feature or properties, Excel file or any data resource".

3.5 Complex or scenarios with more steps

Provide detailed commentary on top of the scenario for easy understanding. Provide comments for any step if extra information is needed.

3.6 Test suite keywords

@Smoke, @Sanity and @Regression, @<<FeatureName>>

4. Technologies used

Programming language: Selenium with Java

JDK Version: 18

Framework: BDD Cucumber with Junit

Design Pattern: Page Object Model + Data Driven Testing

Project Type: Maven

Loggers: Log4j2

Reports: HTML, JSON, XML, PDF, and Extent Reports

5. Framework demo

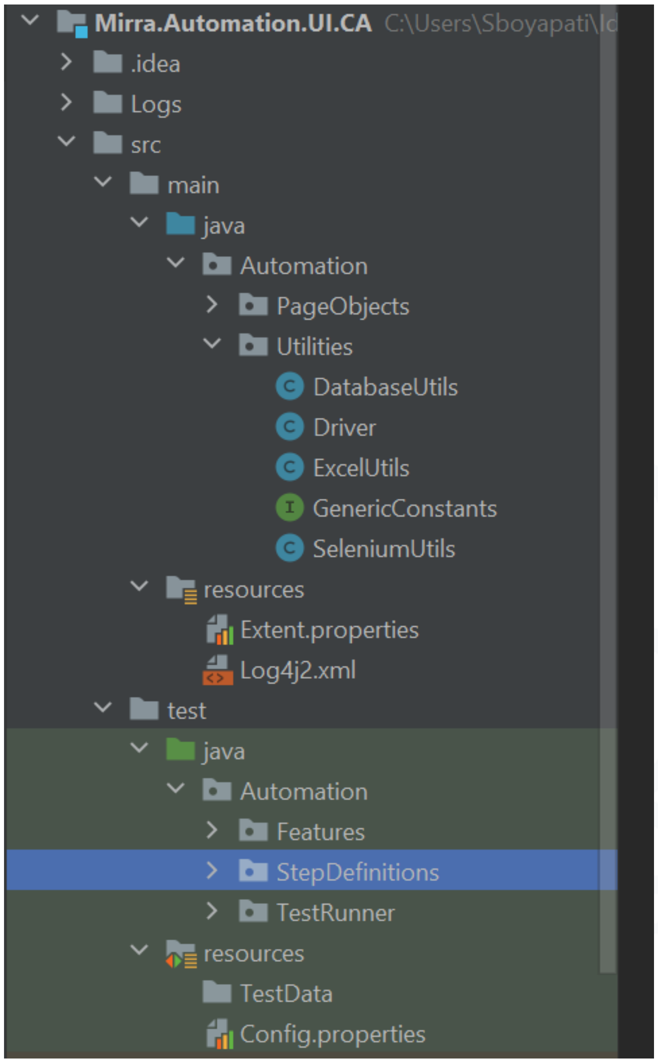
Refer given link for framework demonstration.

Automation framework demo

6. Project structure

6.1 Sample project structure

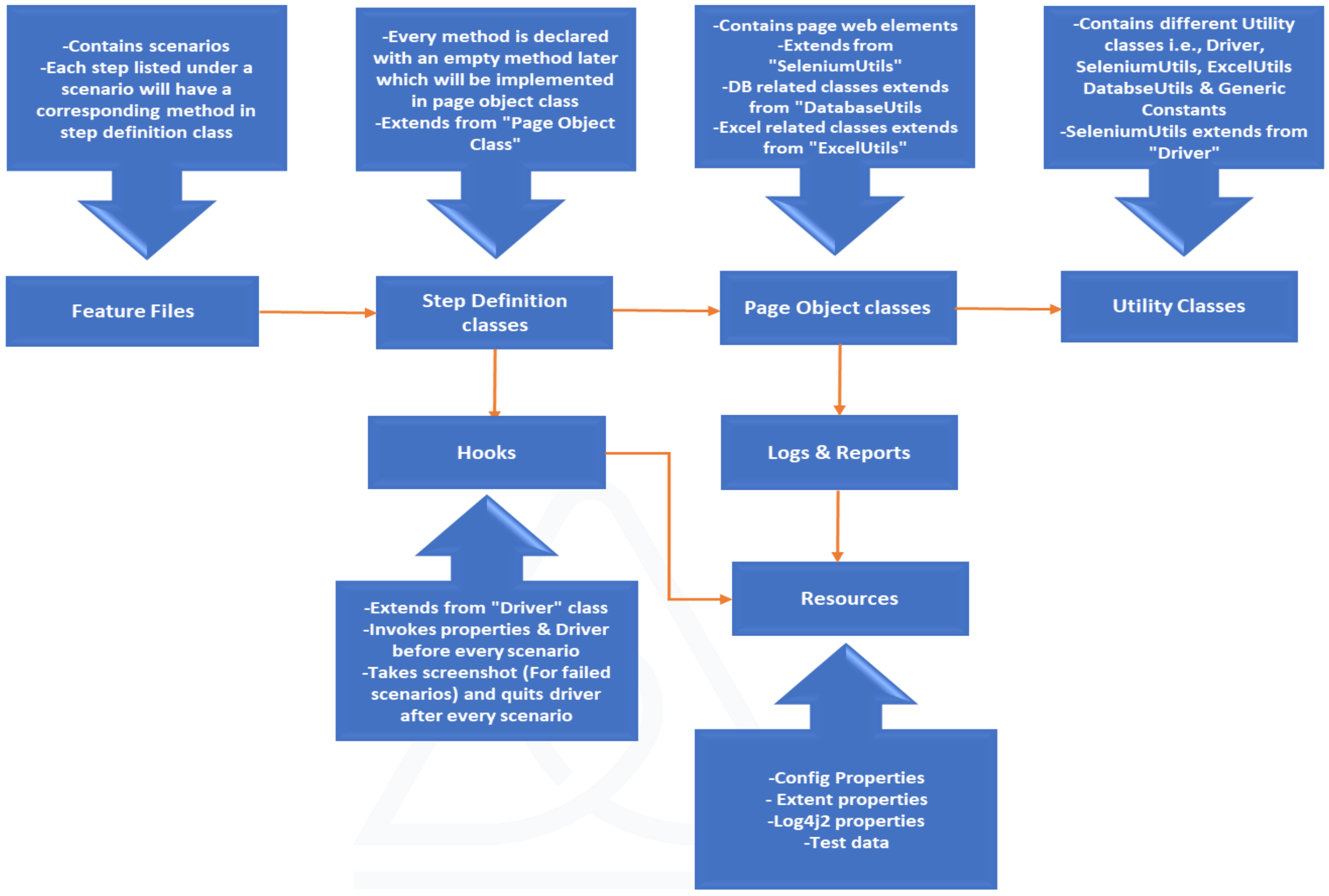
Refer given snapshot to understand the overall folder structure.



```
Mirra.Automation.UI.CA C:\Users\Sboyapati\Id
    .idea
    Logs
    src
        main
            java
                Automation
                    PageObjects
                    Utilities
                        DatabaseUtils
                        Driver
                        ExcelUtils
                        GenericConstants
                        SeleniumUtils
            resources
                Extent.properties
                Log4j2.xml
        test
            java
                Automation
                    Features
                    StepDefinitions
                    TestRunner
            resources
                TestData
                Config.properties
```

6.2 Project structure

Refer given snapshot to understand how the files or classes are linked with each other.


-Contains scenarios
-Each step listed under a scenario will have a corresponding method in step definition class
-Every method is declared with an empty method later which will be implemented in page object class
-Extends from "Page Object Class"
-Contains page web elements
-Extends from "SeleniumUtils"
-DB related classes extends from "DatabaseUtils
-Excel related classes extends from "ExcelUtils"
-Contains different Utility classes i.e., Driver, SeleniumUtils, ExcelUtils DatabseUtils & Generic Constants
-SeleniumUtils extends from "Driver"
Feature Files
Step Definition classes
Page Object classes
Utility Classes
Hooks
Logs & Reports
Resources
-Extends from "Driver" class
-Invokes properties & Driver before every scenario
-Takes screenshot (For failed scenarios) and quits driver after every scenario
-Config Properties
- Extent properties
-Log4j2 properties
-Test data

7. Folder or Package level details

Go through the instructions provided in this section to understand project structure in detail.

7.1 POM.XML

- Contains list of dependencies & Parallel execution related plugin (Surefire) and tags
- **Do not update this file. Must check with lead/Framework SPOC in case of any changes needed.**

7.2 Feature files

- Add scenarios in feature files

7.3 Step definition classes

- Add methods for each step listed under Scenarios in feature files
- Every method in step a definition class will have an empty method declared later which will be implemented in a "Page Object Class"
- Extend "Page Object Classes" to "Step Definition Classes"

7.4 Hooks

- Hooks class is extended from "Driver" class which contains different annotations
- Annotations updated in hooks class are @Before, @AfterStep and @After
- @Before(Order=0) invokes all the properties from "Properties" file before each scenario
- @Before(Order=1) invokes browser driver and returns HomePage URL for each scenario
- @AfterStep() would be executed after every step and captures screen shot for every failed test scenario
- @After() would be executed at the end of every test scenario and closes browser if test case is passed otherwise browser will remain open for reference.
- **Do not update this class. Must check with lead/Framework SPOC in case of any changes needed**

7.5 Page object classes (POC)

- Update POC with all the web elements of a respective web page
- Extend POC from "SeleniumUtils" except "Database and Excel" related page classes
- Database related page classes should be extended from "DatabaseUtils" Whereas "DatabaseUtils" is already extended from "SeleniumUtils"
- Excel related page classes should be extended from "ExcelUtils" Whereas "ExcelUtils" is already extended from "SeleniumUtils"
- Implement all the methods (Write block of code) in "POC" that are declared under "Step Definition Classes"
- Access all the generic methods directly from "SeleniumUtils"
- Write Logs wherever needed

7.6 Utility classes

7.6.1 SeleniumUtils

- "SeleniumUtils" is extended from "driver" class to maintain the same driver across the steps in a scenario
- Contains various methods related to "Windows, Mouse or Keyboard actions, dropdown buttons (Select), Conditions, Element actions, Getters, Locators, Java script Executor, Date & Time and a few generic ones.
- If any generic method needs to be added, then add it under the respective category. If you don't find the category you are looking for create a new one or add them under "Generic methods" section basis where it fits*****
- ***Do not update this class. Must check with lead/Framework SPOC in case of any changes needed**

7.6.2 Driver

- Contains Logger object "log" which can be used in all page object classes to print logs
- Thread local concept has been used to overcome parallel testing issues
- Wherever driver is needed call it with the variable "driver"
- Contains method to load properties and the same method has already been called from "Hooks" class
- Contains properties object "prop" which can be used in page object classes to call properties
- **Do not update this class. Must check with lead/Framework SPOC in case of any changes needed**

7.6.3 ExcelUtils

- "ExcelUtils" is extended from "SeleniumUtils"
- Extend "ExcelUtils" to excel related page object classes
- Both excel and selenium related methods can be directly accessed from the page object classes
- Excel sheet name should be passed as a parameter from properties file
- **Do not update this class. Must check with lead/Framework SPOC in case of any changes needed**

7.6.4 DatabaseUtils

- "DatabaseUtils" is extended from "SeleniumUtils"
- Extend "DatabaseUtils" to database related page object classes
- Both database and selenium related methods can be directly accessed from the page object classes
- When any method is called from "DatabaseUtils" it internally executes **runDBAndGetQueryDetails()** method, establishes connection and returns "resultSet" to the called method to complete the remaining process
- Use **closeConnection()** method to close a connection
- To execute query directly and get "resultset" use **runDBAndGetQueryDetails()** method
- **Do not update this class. Must check with lead/Framework SPOC in case of any changes needed**

8. Resources

- Contains config properties, Extent properties and Log4j2 properties
- Contains test data
- **Do not update Extent and log4j2 properties files. Must check with lead/Framework SPOC in case of any changes needed**

9. Logs

- Logs will be generated under "Logs" folder of the project main directory

10. Reports

- Reports will be generated in "html, JSON, XML & PDF" format
- Refer "Target/ExtentReports" folder to view "ExtentReports HTML & PDF" reports
- Refer "TestReports" of main directory to view reports in all other formats
- Refer "FailedScenarios" file under the same folder to check failed test cases info
- Refer "ThreadReport/index.html" location under the same folder to view threads info

11. Test runner

- Use test runner to run test cases in sequential mode
- Test suits can be handled under the "tags" tab
- Use "FailedTestRunner" to run failed test cases

12. Test cases execution

- Use test runner to run test cases in sequential mode
- Use "Maven Life Cycle" or "command prompt" to run test cases parallelly