Decorative sidebar

Version number and date

Title and subtitlePresenter, company name and address

Carina Automation Framework

Overview

|  |  |
| --- | --- |
| Practice: | Banking Automation |
| Document Owner: | Tech Team |
| Version: | 1.0 |
| Suggested users: | Automation/Solution Architect |
| Author (s) | Archana Tikole |
| Reviewer (s) | Kaveri Keer /Rupesh Sawant |
| Practice: |  |

**Table of Contents**

[1](#_heading=h.1fob9te) INTRODUCTION 4

[1.1](#_heading=h.3znysh7) Objective 4

[1.2](#_heading=h.2et92p0) Summary 4

[2](#_heading=h.tyjcwt) HIGH LEVEL DESIGN FEATURES 5

[3](#_heading=h.3dy6vkm) ENVIRONMENT- HARDWARE AND SOFTWARE REQUIREMENT 6

[3.1](#_heading=h.1t3h5sf) System Configuration 6

[3.2](#_heading=h.4d34og8) Applications and Jar files 6

[4](#_heading=h.2s8eyo1) ARCHITECTURE 7

[4.1](#_heading=h.17dp8vu) General Architecture of Carina Framework 7

[4.2](#_heading=h.26in1rg) Implementation of Page Objects 9

[4.3](#_heading=h.lnxbz9) Implementation of UI Components 10

[5](#_heading=h.35nkun2) PROCEDURE OF INSTALLING THE FRAMEWORK 12

[6](#_heading=h.1ksv4uv) FOLDER STRUCTURE AND COMPONENT 13

[6.1](#_heading=h.44sinio) Sample project structure for WEB 13

[6.2](#_heading=h.2jxsxqh) Common Class 13

[7](#_heading=h.z337ya) TEST DATA 15

[8](#_heading=h.3j2qqm3) LIBRARY OF UTILITY FUNCTIONS 16

[9](#_heading=h.1y810tw) PROCEDURE FOR DEVELOPING AUTOMATION SCRIPTS 17

[10](#_heading=h.4i7ojhp) PROCEDURE FOR CONFIGURING TEST ENVIRONMENT 18

[11](#_heading=h.2xcytpi) PROCEDURE FOR EXECUTION OF AUTOMATION SCRIPT 19

[11.1](#_heading=h.1ci93xb) Setup before execution 19

[11.2](#_heading=h.3whwml4) Execution 19

[12](#_heading=h.2bn6wsx) REPORTING AND LOGS 20

[13](#_heading=h.qsh70q) INTEGRATION SCRIPTS IN DEVOPS 21

[14](#_heading=h.3as4poj) IMPEDIMENTS 22

# INTRODUCTION

## Objective

Functional Test Automation, much as it is essential for achieving higher Test coverage and improved Test execution productivity on any project has its own challenges.

In a scenario where the inputs/requirements for any Test Automation project are varied, the need of the hour is to develop a “Test Automation Framework” that can accommodate these varying inputs/requirements and additionally provide the following benefits

* A consistent architecture for the automated solution
* A consistent approach to develop the automated solution specific to each Test Automation project.
* Improved Test script development productivity
* Reusable Utility libraries/functions that an be consistently used across all projects supported by the tool

## Summary

To enable user to understand and use of framework, this document support the way the framework support the development, execution and maintenance of the automation script.

This document will provide the information related to tool, framework components.

# HIGH LEVEL DESIGN FEATURES

The framework developed using Carina has the following features-

Carina is a Java-based test automation framework that unites all testing layers: Mobile applications (web, native, hybrid), WEB applications, REST services, Databases.

* To be able to create automated tests quickly by using appropriate abstraction layers
* Meaningful logging and reporting structure
* Easily maintainable and extendable
* Simple enough for testers to write automated tests

# ENVIRONMENT- HARDWARE AND SOFTWARE REQUIREMENT

## System Configuration

* 8 GB RAM
* i3 Processor
* Windows (64 bit)

## Applications and Jar files

* Eclipse (2019-12)
* Link - <https://www.eclipse.org/downloads/packages/release/2019-12/r/eclipse-ide-enterprise-java-developers>
* Selenium Stand-alone Jar
* <https://selenium-release.storage.googleapis.com/3.141/selenium-server-standalone-3.141.59.jar>
* JAVA JDK 1.8.0\_251
* <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html#license-lightbox>
* Apache Maven
* <http://apachemirror.wuchna.com/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.zip>
* Carina Framework (Version 6.4.35)
* TestNG(Eclipse Add-in)
* In eclipse-> Help-> Marketplace -> Search TestNG and install
* IEDriver compatible with Internet Explorer 11
* <https://seleniumrelease.storage.googleapis.com/3.150/IEDriverServer_x64_3.150.1.zip>

# ARCHITECTURE

## General Architecture of Carina Framework

Carina test project is structured as a standard Maven project:

carina-demo

|-- pom.xml

|-- src/test/java

|-- src/test/resources

|-- api

|-- testng\_suites

|-- xls

|-- src/main/java

|-- src/main/resources

|-- L10N

* **src/test/java** - contains test classes organized using TestNG annotations



* **src/test/resources** - contains TestNG xml files, API templates and XLS/CSV data providers etc



* **src/main/resources** - contains L10N resources, configuration properties files and MyBatis profiles if needed



All properties may be retrieved in test using R class:

R.CONFIG.get("browser")

All project configuration properties are located in **\_config.properties** file.



* **Screenshots**
* For screenshots in Reports folder keep “gallery-lib” folder in Reports folder. **config.properties file setting**:
* In global configuration
  + Set URL - to expected url
  + Set browser to ‘iexplore’ for internet explorer browser
* In ‘Report configuration’
  + Set property ‘custom\_project\_report\_directory’ - report path where you want to save reports.
  + Set property ‘custom\_screenshot’ – true if you want to take screenshots.

## Implementation of Page Objects

In general, Page Object contains locators of the elements situated on the page and some business logic that may be reused by different tests:

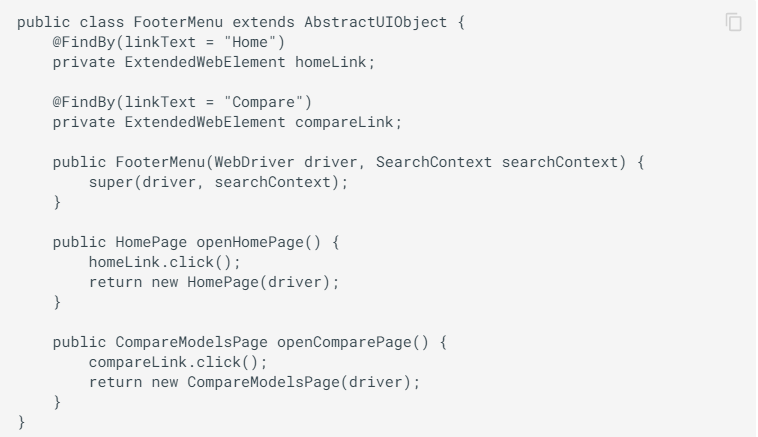


**Important:**

* Page should extend **com.qaprosoft.carina.core.gui.AbstractPage**
* Use **com.qaprosoft.carina.core.foundation.webdriver.decorator.ExtendedWebElement** instead of Selenium WebElement
* Locate Page Object classes in src/main/java

## Implementation of UI Components

In some cases, it is useful to implement UI Objects that may be reused between multiple Page Objects. For instance, a top menu or footer may be shared between multiple pages:



And then you can use this in Page Object:



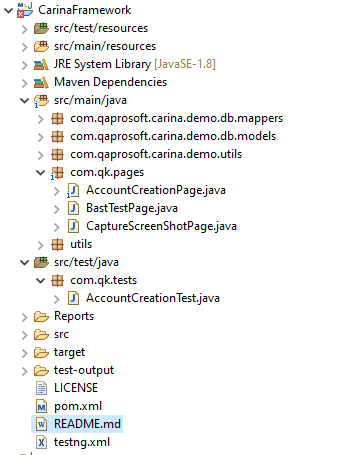
# PROCEDURE OF INSTALLING THE FRAMEWORK

Download the framework zip file and unzip to use.

# FOLDER STRUCTURE AND COMPONENT

* There are two packages and each module is divided into Pages and Tests.
* Page files contain entire business logics and Test files contain all verifications and are executables.
* For some pages there is only one Test file as there is dependency of data.

## Sample project struture for WEB



## Common Class

1. **BaseTestPage**

This page contains common functions like

* + Login Application
  + Logout Application
  + Accept certificate
  + Switch and close windows

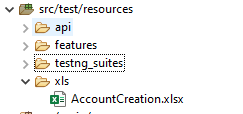
1. **CaptureScreenShotsPage**

* This page contains methods to capture screenshots and save on specific location.

# TEST DATA

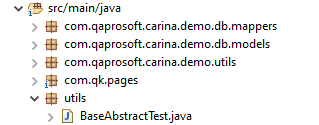
To add test data in framework

src/test/resourece -> xls folder -> Add testdata file in excel format.



# LIBRARY OF UTILITY FUNCTIONS

Added BaseAbstractTest.java as utility with updated code for Reports.



# PROCEDURE FOR DEVELOPING AUTOMATION SCRIPTS

First download the Framework in local system and then start adding pages in page object format.

1. Create all pages within src/main/page -> com.qk.pages package
2. Then add testdata excel in src/test/resources -> xls folder
3. After that create test pages in src/main/test -> com.qk.test package

# PROCEDURE FOR CONFIGURING TEST ENVIRONMENT

To configure different properties of framework for WEB use \_config.properties file.



# PROCEDURE FOR EXECUTION OF AUTOMATION SCRIPT

## Setup before execution

* Before starting execution from qps-hub -> start selenium server.
* Open ‘web.xml’ file from “src/test/resources/testng\_suites”
* Set up test file name which is to be executed in following format
  + <test name = “Test”>
  + <classes>
    - <class name = “com.qk.tests.AccountCreationTest” />
  + </classes>
  + </test>
* All test files are already setup in web.xml file.
* Tests which don’t want to execute can be commented using <!--Test -->

## Execution

* Right click on web.xml file and select Run as -> TestNG Suite
* Another way to execute specific file
  + Select test file from ‘com.qk.tests’ -> right click file -> Run as -> TestNG Test

# REPORTING AND LOGS

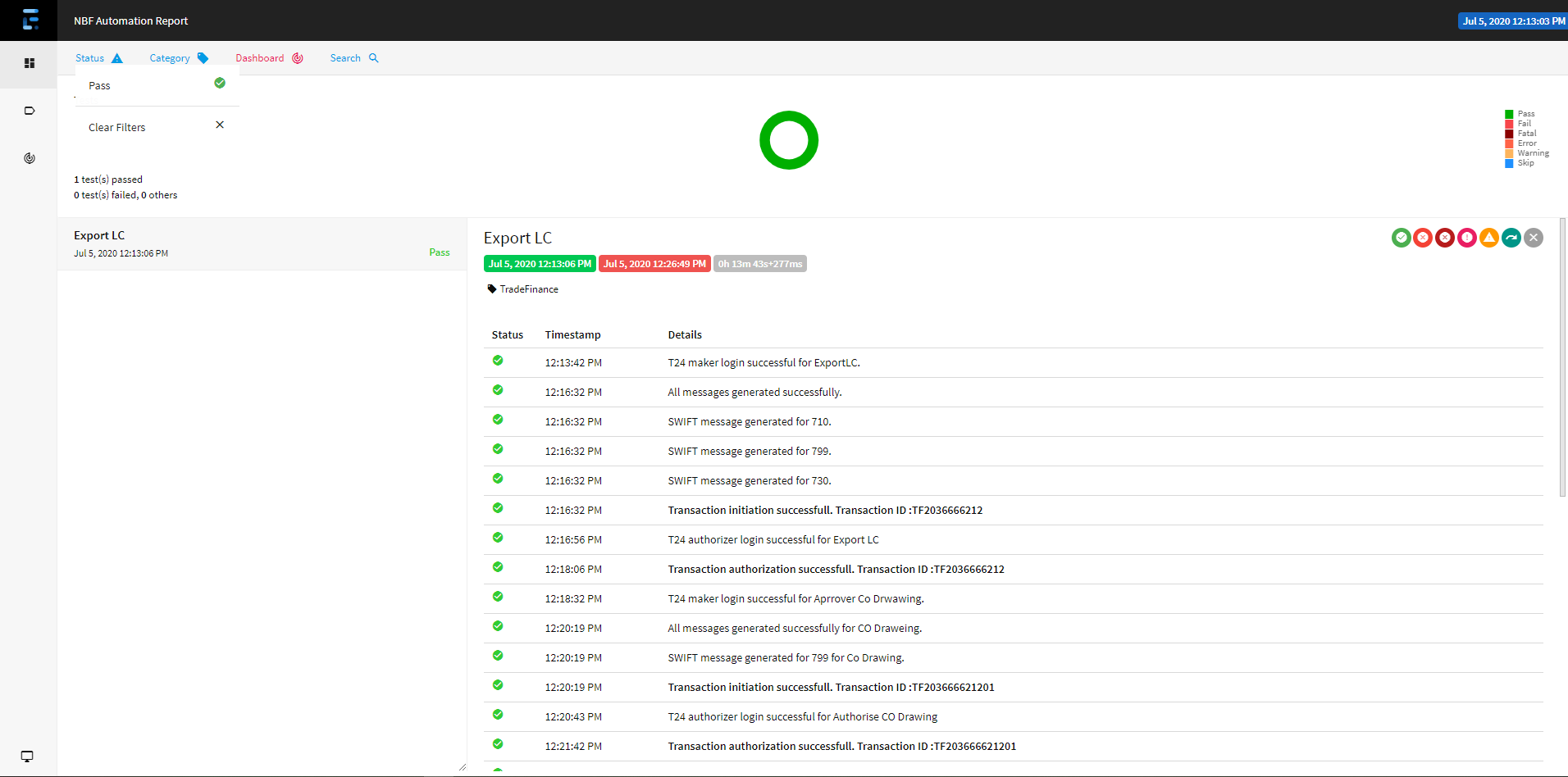
Set the path for Report folder in \_config.properties file against

custom\_project\_report\_directory

For example: custom\_project\_report\_directory=D:\\QKCarinaFramework\CarinaFramework\\Reports

All Reports will be stored in the ‘Reports’ folder with the name mentioned in Test file appended with current date and time.

Here is screenshot of sample report



# INTEGRATION SCRIPTS IN DEVOPS

# IMPEDIMENTS