# Kantar Framework (Rainbow):

The Framework is having the below layers

* BaseClass
* Object Repository
* Components
* TestScripts
* Log4J
* TestReports
* Java Library

BaseClass: It contains all the reusable methods. Which can be used to automate the scenarios. The below the reusable methods in BaseClass.

public void setbrowser(String browser)

public static void launchapp(String urlfromproperties)

public static void click(WebElement elem)

public static void click(By locator)

public static void set(WebElement elem, String data)

public static void select(WebElement elem, String data)

public static void switchframe(WebElement elem)

public static void SwitchToDefaultFrame()

public static void switchwindow(int index)

public void flush() throws Exception

public static void wait(int timer, WebElement elem)

public static boolean IsElementPresent(WebElement elem)

public int totalitemsdropdownlist(WebElement elem)

public static boolean ischeckboxcheckedbbydefault(WebElement elem)

public static void clickCheckBox(WebElement elem, boolean paramBoolean)

public void clickRadioButton(WebElement elem)

public static void clickImage(WebElement elem)

public static void verifyElementIsEnabled(WebElement elem, boolean paramBoolean)

public static boolean isAlertPresent()

public static void handleConfirmation(String paramString) throws AutomationException

public static boolean verifyPopupMessage(String paramString) throws AutomationException

public static String getPopupMessage()

public static void switchBackToOriginalBrowser() throws AutomationException

public static void sleep(float paramFloat)

public static int getTableRowCount(String tableid) throws AutomationException

public static int getColHeaderNumber(String tableid, String colExpct)

public static int getRowNumber(String tableid, String colExpct,String rowExpct) throws

public static void pressEnterKey()

public static void swtichToChildTab()

public static void VerifyText(WebElement elem, String paramString2)

public static String getToolTipText(WebElement elem,String paramString1)

public static void verifyListItems(WebElement elem)

public static void verifyValuesInDropdown(WebElement elem, String[] value)

public static By getLocators(String paramString1, String paramString2)

public static String defaultdropdownselecteditem(WebElement elem) {

public String alldropdownlistvalues(WebElement elem) {

public String getdate(int period, String format) {

public static String getattributevalue(WebElement elem, String requiredattribute)

public static void alertaction(String action) {

public static String text(WebElement elem) {

public int totallinnks(WebElement elem) {

public static void capturesnapshot(String imagename) throws IOException

public static String getabsoluteimagepath(String imagename)

public static boolean verifyElementExist(WebElement elem)

public static void VerifyTableElement(String[] value1,String[] value2)

public static void Mousehover(WebElement elem)

public static void selectListItem(WebElement elem, String paramString)

public static void waitForObj(long ms)

public static void switchToBrowser(String paramString)

public static void handleAlert()

public void AutomationException(String paramString)

public void switchbacktodefaultframe()

@BeforeSuite

public void setUpSuiteDetails(ITestContext ctx)

@BeforeMethod

public void captureDesc(Method method){

@AfterSuite

public void afterSuite()

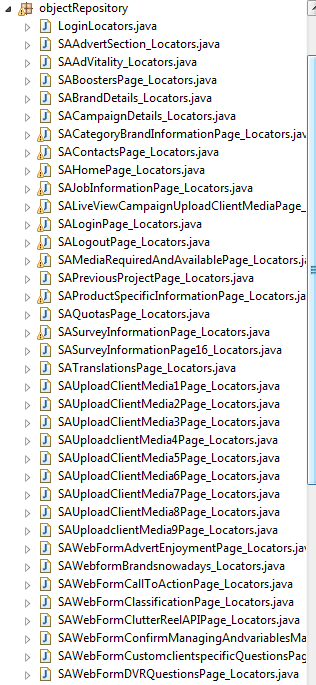
@BeforeClass

public void BeforeClass(String str,String browserName) throws Exception

public void destroy() throws Exception

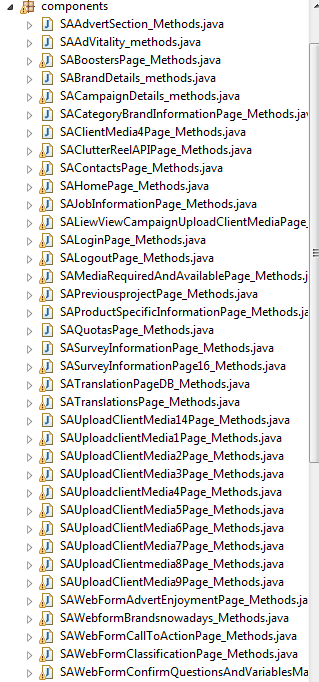
## Object Repository:

We can implement the page wise locator in this layer by using POM (page Object Model). Below are the pages which contains locators



## Components:

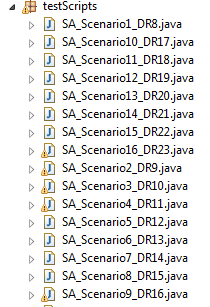
It contains the methods for the scenario. The methods will be implemented by using BaseClass reusable methods and Object Repository



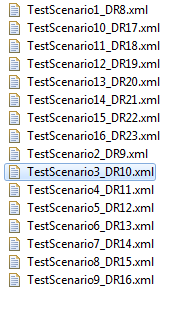
## TestScripts:

It contains the test cases (test methods). The test Scripts will be implemented by using the methods which are available in the components.

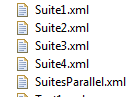
I have implemented 16 test scripts for the 16 scenarios



To execute the scripts, implemented the xml files to run individual test script as below,



To run multiple test scenarios as a suite, I have implement below xml files



## Log4J:

Will capture each and every action in the log file, in the form of message. It will help to analyse the failures

## TestReports:

Once the execution is completed the framework will provide the results in PDF format along with screenshots

The PDF report contains SummaryReport, Individual test script report and screenshots

## Java Library:

It contains all the supported library file for the framework.

