IQ

IST_Quality

Private group

Not following Next steps

6 members

New Send by email Page detai Draft saved 11/29/2018 Edit Publish

Test framework utilities and usage

Test Framework Utilities and their Usage

Test Framework

Framework-utils provides the set of utilities which can be used directly into the test project.

Import the dependency in the maven pom or gradle build file to use the framework-utils within the test project

Usage:

Gradle:

compile 'com.esri:framework-utils:1.0.0'

Maven:

```
<dependency>

 <groupId>com.esri</groupId>

 <artifactId>framework-utils</artifactId>

 <version>1.0.0</version>

</dependency>
```

Config and properties file reader

The config properties can be read using the framework utilities classes. Config.java implements on how properties are read within the project.

Usage. Config.getConfigProperty("<property_name>") Or
Config.getConfigProperty(ConfigProperty.PROPERTY) -> ConfigProperty is enum

Tests may need to be executed on different environments which require us to pass the environment related properties in run time. This is achieved by reading the configurations in certain order.

· **Enum**: default properties written in an enum in the framework code. This is required so that we can always run test cases using default local framework related properties. These properties are used within the framework utils only.

· **test.properties** file - the test application related properties will be listed in this file AND can also overwrite the properties in the enum. Generally we need this in our local environments while working/testing on different environment.

· **System properties** - system properties can be set during run time via command line or jenkins job configurations, set these properties based on the environment where the jobs are running. Suppose test application is using "url" is one property to connect to dev environment an d we want to run the test in another environment then we can override the test.properties property using system property.

Reporting

The test reports are created using extent reporting. Custom report listener classes

are created and are added as Listeners in the testng.xml file.

The reports are customized to:

- Print the test name using the xml parameters in case parameters are passed via xml file or use the test description from the CSV input file.
- Each test case lists the various assertion steps performed in the test case. Each step has the status indicator, description and details. The details and descriptions are customizable based on the values passed in the assertion method from test case. The details are passed as the part of the assert command.
- Print the stack trace if the test fails. The TestNg will catch the exceptions and will be printed in the reports. Please note that we should handle the exceptions appropriately. the exception should be thrown if it occurs and there is no point of executing the test case further as it will fail.
- The tests are categorized in groups as well.
- The report provides the test execution stats.

Usage:

Add listener entry in the xml file

```
<listeners>
    <listener class-name=uot;com.esri.qa.reporting.ExtentReporter" />
   <listener class-name=uot;com.esri.qa.reporting.ReportListener" />
</listeners>
```

In the test class the reporting can be performed using.

ReportLogger reportLogger =ew ReportLogger();

reportLogger.assertEquals(status, 200, "Validate response status is 200" + status);

All TestNg provided assertions methods are supported

Extended Report Generation

The main scope of this article is to show Extent Reports. This is the most popular and widely used Selenium Reporting tool in the current market. Let's see how to generate extent reports in Selenium WebDriver.

Pre-requisites to Generate Extent Reports:

1. Java should be installed (Link to Install and setup Java)
2. TestNG should be installed – Install TestNG
3. Extent Report Jars (Version 2.41.2) – Download
4. extent-config.xml – It allows to configure HTML Report

Steps To Generate Extent Reports:

1. Firstly, create a TestNG project in eclipse
2. Now download extent library files from the following link: http://extentreports.relevantcodes.com/
3. Add the downloaded library files to your project
4. Create a java class say 'ExtentReportsClass' and add following code to it

Code Explanation:

i. Imported two classes ***ExtentReports*** and ***ExtentTest***.

***ExtentReports*:** By using this class we set the path where our reports need to generate.

***ExtentTest*:** By using this class we could generate the logs in the report.

ii. Took three methods with @Test annotation such as *passTest*, *failTest* and *skipTest* and a method *startTest*with @BeforeTest annotation and another method *endReport* with @AfterMethod annotation

Here my intention is to generate a report with all the three types of results such as Pass, Fail and Skip.

Add Screenshots of a failed Test Cases in Extent Reports

iii. Used object of ***ExtentReports*** class (i.e., *extent*) in the *startReport* method which was assigned to @BeforeTest annotation to generate the HTML report in the required path

iv. Used object of ***ExtentTest*** class (i.e., *logger*) in the remaining methods to write logs in the report.

v. Used ***ITestResult*** class in the @AfterMethod to describes the result of a test.

Given clear explanation in the comments section with in the program itself. Please go through it to understand the flow.

```
package com.esri.qa.ui;



import java.io.IOException;
import java.net.UnknownHostException;



import org.apache.commons.lang3.StringUtils;
import org.openqa.selenium.Proxy;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.ITestContext;
import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.DataProvider;



import com.applitools.eyes.TestResults;
import com.esri.qa.adobe.analytics.ESRIAdobeAnalyticsNetworkTraffic;
import com.esri.qa.config.Config;
import com.esri.qa.config.Config.ConfigProperty;
import com.esri.qa.reporting.CBTManager;
import com.esri.qa.reporting.Log;
import com.esri.qa.reporting.ReportLogger;
import com.esri.qa.testng.TestNGUtilities;



/**
 * This Class will be extended in each of the test class to initialize webDriver
 * and open the URL The TestNG annotation is before and after methods.
 *
 * @author Deenesh
 * @updated Al
 */
public class BaseTestUi {
    private ApplitoolsCapabilities p_AppliTools;
    private String p_AppName;
    private String fileName;
    private String startRow;
    private String endRow;
    private String bVersion;
    private Browser browser;
    private String buildTag =onfig.getConfigProperty(ConfigProperty.BUILD_TAG);
    private boolean cbtEnabled
=oolean.parseBoolean(Config.getConfigProperty(ConfigProperty.CBT_ENABLED));
    private boolean quitBrowser
=oolean.parseBoolean(Config.getConfigProperty(ConfigProperty.QUIT_BROWSER));
    private boolean isScreenshotEnabled
=oolean.parseBoolean(Config.getConfigProperty(ConfigProperty.SCREENSHOT_ENABLED));
    private boolean networkAnalysisEnabled
=oolean.parseBoolean(Config.getConfigProperty("network.analysis.enabled"));
    protected WebDriver driver;
    protected ReportLogger reportLogger;
    protected String url;
    protected ESRIAdobeAnalyticsNetworkTraffic network;
    protected Proxy proxy;
    protected String bType;
    protected String platform;
    protected String dimension;



    public BaseTestUi(String fileName, String startRow, String endRow,
            String bType, String bVersion, String platform, String url, String dimension) {
        this.fileName =ileName;
        this.startRow =tartRow;
        this.endRow =ndRow;
        this.bType =Type;
        this.bVersion =Version;
        this.platform =latform;
        this.url =rl;
        this.dimension =imension;
    }



    /**
     * This method is invoked at the beginning of each test class and sets up
     * the browser instance.
     *
     * @param context
     * @throws UnknownHostException
     */
    @BeforeClass(alwaysRun =rue)
    public void initializeBrowser() throws UnknownHostException {
        Log.info("Initializing before class objects");
        p_AppName =his.getClass().getName().substring(this.getClass().getName().lastIndexOf('.') + 1);
        //create proxy server to capture HAR data. **not stable**
        if (networkAnalysisEnabled) {
            network =ew ESRIAdobeAnalyticsNetworkTraffic();
            proxy =etwork.getHTTPPRoxy();
        }
        browser =ew Browser(bType, bVersion, platform, p_AppName, dimension, proxy);
        driver =rowser.getDriver();
    }



    /**
     * This method is invoked at the beginning of each test and sets up the
     * report logger instance
     *
     */
    @BeforeMethod(alwaysRun =rue)
    public void beforeMethodSetup(ITestResult tr) {

        //Set the attributes to be printed in the HTML report and create the report object.
        tr.setAttribute("BrowserType", bType);
        tr.setAttribute("BrowserVersion",((RemoteWebDriver) driver).getCapabilities().getVersion());
        tr.setAttribute("Platform", System.getProperty("os.name"));
        tr.setAttribute("BuildEnv", Config.getConfigProperty("test.environment"));
        tr.setAttribute("driver", driver);
        reportLogger =ew ReportLogger(driver, tr, isScreenshotEnabled);



        //open the url if it is passed from xml file.
        if (!StringUtils.isBlank(url)) {
            PageObjects page =ew PageObjects(driver);
            page.openURL(url);
        }



        //Initialize applitools
        p_AppliTools =ew ApplitoolsCapabilities("AEMpre001","4545");
    }



    @AfterMethod(alwaysRun =rue)
    public void afterMethodCleanup(ITestContext tr){



         //Close Applitools if not closed
        if(p_AppliTools !=ull){
            Log.error("Applitools not closed in the test method");
            TestResults appliResults =_AppliTools.getTestResults();
            p_AppliTools =ull;
            Log.info("RESULT: " + appliResults.getStatus());
        }
    }



    /**
     * This method get the results of applitools
     */
    public void closeTest(){



         //Close Applitools
        TestResults appliResults =_AppliTools.getTestResults();



        if(appliResults !=ull){
            reportLogger.info("APPLITOOLS RESULT STATUS: " + appliResults.getStatus());
            if (!appliResults.isNew()) {
            reportLogger.infoLink("Applitools Result Link",appliResults.getUrl());
            reportLogger.assertTrue(appliResults.isPassed(), "Validate Image Comparison");
            }
        }
        else{
            Log.info("Applitools not enabled for this execution");
        }
        p_AppliTools=ll;
        reportLogger.assertAll();
    }



    /**
     * After every Test method the tear down method is called to close the
     * session.
     *
     * @param result
     * @param context
     */
    @AfterClass(alwaysRun =rue)
    public void tearDown(ITestContext context) {
        if (cbtEnabled) {
            CBTManager.setScore(((RemoteWebDriver) driver).getSessionId().toString(), context);
        }
        if (quitBrowser) {
            browser.quitBrowser();
        }
        if (networkAnalysisEnabled) network.stopServer();
    }
    public void AppliCompareImage(AppliCompareData AData) {
        p_AppliTools.appliCompareImage(AData);
    }



    @DataProvider(name =csv")
    public Object[][] createData() throws IOException {
        return TestNGUtilities.parseCsvToMap(fileName, Integer.parseInt(startRow),
Integer.parseInt(endRow));
    }

    @DataProvider(name =csvParallel", parallel=ue)
    public Object[][] createParallelData() throws IOException {
        return TestNGUtilities.parseCsvToMap(fileName, Integer.parseInt(startRow),
Integer.parseInt(endRow));
    }



}
```

extent-config.xml:

By using this external XML file (extent-config.xml), we could change the details such as Report Theme (either standard or dark), Report Title, Document Title etc.,

We use extent object and use loadConfig() method to load this XML file.

```
<suite name=uite name" preserve-order=rue"
  parallel=ests" thread-count=">

    <test name=est name'>
        <parameter name=ileName"
            value=sv file location" />
        <parameter name=tartRow" value=" />
        <parameter name=ndRow" value=" />
        <parameter name=Type" value=hrome" />
        <classes>
            <class
                name=lass.name" />
        </classes>
    </test>

<listeners>
        <listener class-name=om.esri.qa.reporting.ExtentReporter" />
        <listener class-name=om.esri.qa.reporting.ReportListener" />
    </listeners>
</suite>
```

Console Output:

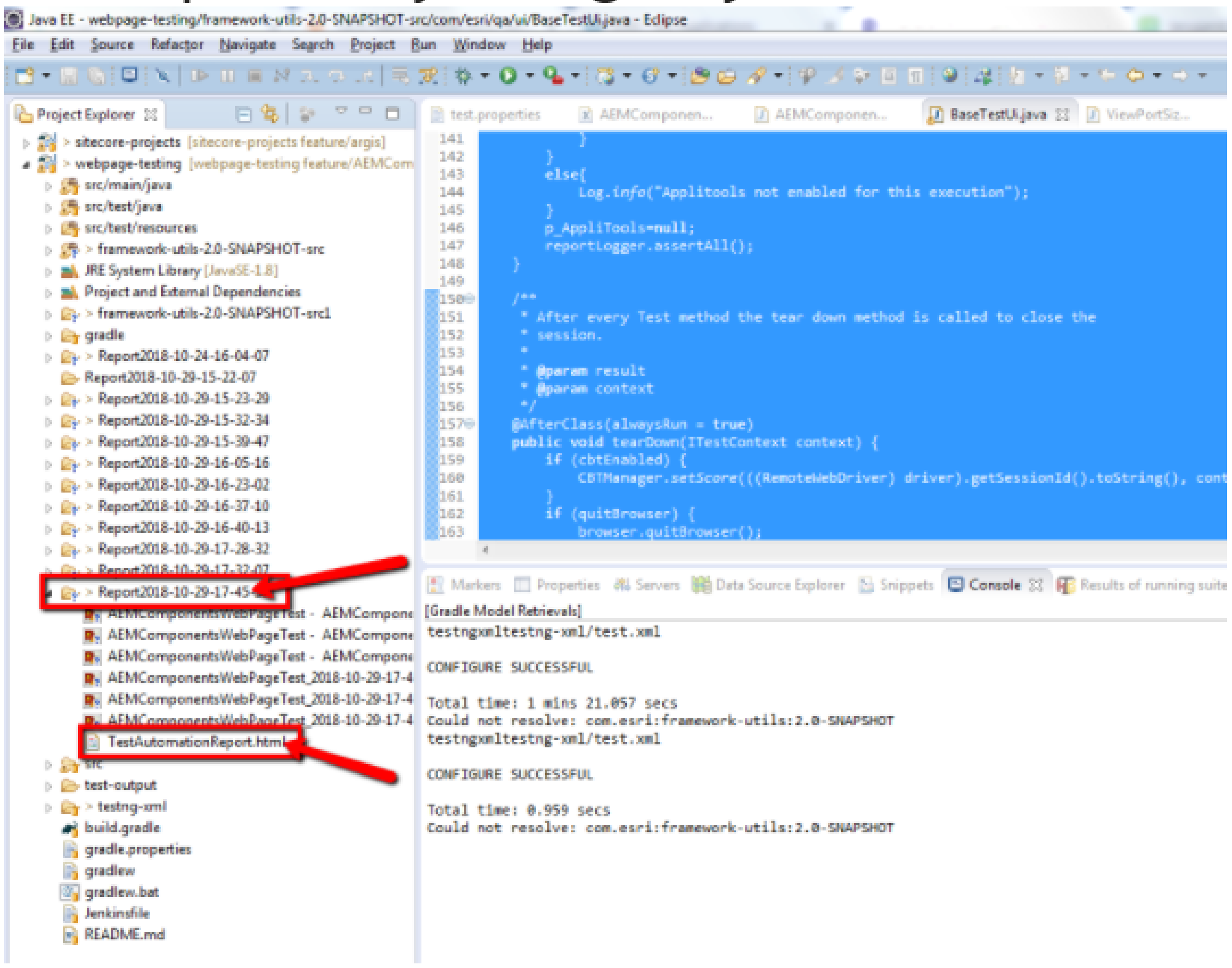
================/span>

Default suite

Total tests run: 3, Failures: 1, Skips: 1

================/span>

Refresh the project after execution of above BaseTestUi.java file. You could find an Report folder with name with date and time as suffix file locate the .html file and open it by using any browser.



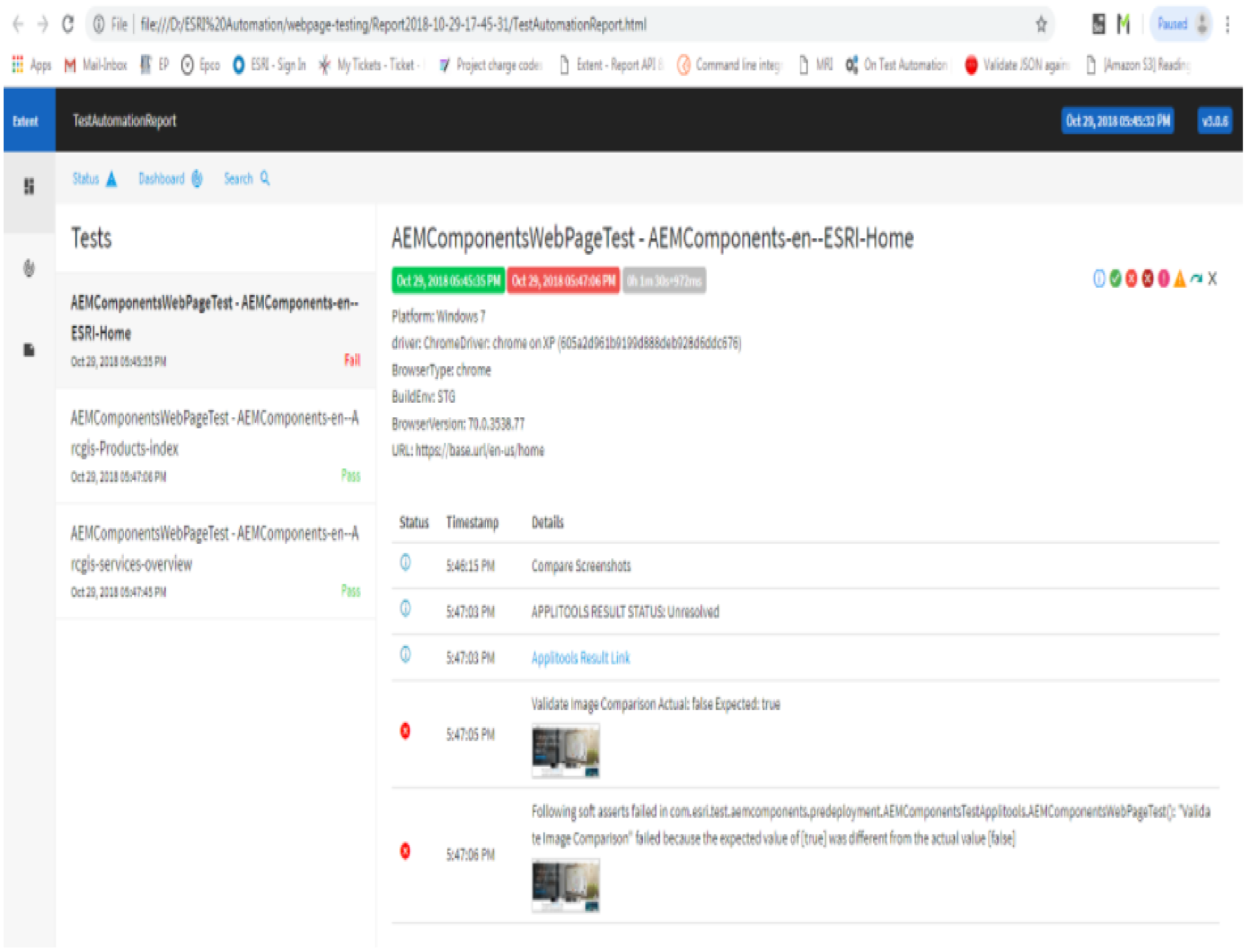
Java EE - webpage-testing/framework-utils-2.0-SNAPSHOT-src/com/esri/qa/ui/BaseTestUi.java - Eclipse

```
sitecore-projects [sitecore-projects feature/argis]
webpage-testing [webpage-testing feature/AEMCom
    src/main/java
    src/test/java
    src/test/resources
    framework-utils-2.0-SNAPSHOT-src
    JRE System Library [JavaSE-1.8]
    Project and External Dependencies
    framework-utils-2.0-SNAPSHOT-src1
    gradle
    Report2018-10-24-16-04-07
    Report2018-10-29-15-22-07
    Report2018-10-29-15-23-29
    Report2018-10-29-15-32-34
    Report2018-10-29-15-39-47
    Report2018-10-29-16-05-16
    Report2018-10-29-16-23-02
    Report2018-10-29-16-37-10
    Report2018-10-29-16-40-13
    Report2018-10-29-17-28-32
    Report2018-10-29-17-32-07
    Report2018-10-29-17-45
        AEMComponentsWebPageTest - AEMCompone
        AEMComponentsWebPageTest - AEMCompone
        AEMComponentsWebPageTest - AEMCompone
        AEMComponentsWebPageTest_2018-10-29-17-4
        AEMComponentsWebPageTest_2018-10-29-17-4
        AEMComponentsWebPageTest_2018-10-29-17-4
        TestAutomationReport.html
    src
    test-output
    testng-xml
    build.gradle
    gradle.properties
    gradlew
    gradlew.bat
    Jenkinsfile
    README.md
```

```java
            }
        }
        else{
            Log.info("Applitools not enabled for this execution");
        }
        p_AppliTools=null;
        reportLogger.assertAll();
    }

    /**
     * After every Test method the tear down method is called to close the
     * session.
     *
     * @param result
     * @param context
     */
    @AfterClass(alwaysRun = true)
    public void tearDown(ITestContext context) {
        if (cbtEnabled) {
            CBTManager.setScore(((RemoteWebDriver) driver).getSessionId().toString(), cont
        }
        if (quitBrowser) {
            browser.quitBrowser();
```

```
[Gradle Model Retrievals]
testngxmltestng-xml/test.xml

CONFIGURE SUCCESSFUL

Total time: 1 mins 21.057 secs
Could not resolve: com.esri:framework-utils:2.0-SNAPSHOT
testngxmltestng-xml/test.xml

CONFIGURE SUCCESSFUL

Total time: 0.959 secs
Could not resolve: com.esri:framework-utils:2.0-SNAPSHOT
```

You could see beautiful high rich HTML reports as shown below.


file:///D:/ESRI%20Automation/webpage-testing/Report2018-10-29-17-45-31/TestAutomationReport.html
Extent
TestAutomationReport
Oct 29, 2018 05:45:32 PM
v3.0.6
Status
Dashboard
Search

Tests

AEMComponentsWebPageTest - AEMComponents-en--ESRI-Home
Oct 29, 2018 05:45:35 PM Fail

AEMComponentsWebPageTest - AEMComponents-en--Arcgis-Products-index
Oct 29, 2018 05:47:06 PM Pass

AEMComponentsWebPageTest - AEMComponents-en--Arcgis-services-overview
Oct 29, 2018 05:47:45 PM Pass

AEMComponentsWebPageTest - AEMComponents-en--ESRI-Home

Oct 29, 2018 05:45:35 PM | Oct 29, 2018 05:47:06 PM | 0h 1m 30s+972ms

Platform: Windows 7
driver: ChromeDriver: chrome on XP (605a2d961b9199d888deb928d6ddc676)
BrowserType: chrome
BuildEnv: STG
BrowserVersion: 70.0.3538.77
URL: https://base.url/en-us/home

Status	Timestamp	Details
ⓘ	5:46:15 PM	Compare Screenshots
ⓘ	5:47:03 PM	APPLITOOLS RESULT STATUS: Unresolved
ⓘ	5:47:03 PM	Applitools Result Link
✖	5:47:05 PM	Validate Image Comparison Actual: false Expected: true
✖	5:47:06 PM	Following soft asserts failed in com.esri.test.aemcomponents.predeployment.AEMComponentsTestApplitools.AEMComponentsWebPageTest(): "Validate Image Comparison" failed because the expected value of [true] was different from the actual value [false]

Applitools Screenshot Reporting

Applitools used to test the functionality of your app through the UI, this allowed you to automatically test the look & feel and user experience of your app. That is, verifying that each UI element in each page appears in the right color, shape, position and size, and that it does not overlap or hide other UI elements.

Add the Applitools Capabilities in Framework sample :

```
package com.esri.qa.ui;



import com.applitools.eyes.BatchInfo;
import com.applitools.eyes.TestResults;
import com.applitools.eyes.selenium.Eyes;
import com.applitools.eyes.selenium.StitchMode;
import com.esri.qa.config.Config;
import com.esri.qa.reporting.Log;
```

```
public class ApplitoolsCapabilities {



    private Eyes p_Eyes;
    private BatchInfo p_BatchInfo;
    private Boolean p_AppliEnabled =alse;
    private Boolean p_EyesClosed =alse;



    public ApplitoolsCapabilities(String BatchName, String BatchID){



        if(BatchID.trim().toLowerCase().length() > 1)
        {
            p_BatchInfo =ew BatchInfo(BatchName);
            p_BatchInfo.setId( BatchID );
            p_AppliEnabled =rue;
        }
    }



    private void initializeEyes(AppliCompareData AData) {



        // Initialize the eyes SDK and set your private API key.
        p_Eyes =ew Eyes();
        p_Eyes.setApiKey("SeHi101102VZBGOT1HiwBcr7FEx1Ykg1OMuThhVrWuZFWmI110");
        p_Eyes.setBatch(p_BatchInfo);
        p_Eyes.setStitchMode(StitchMode.CSS);
        p_Eyes.setHideScrollbars(false);
        try {
            // Start the test and set the browser's viewport size to 1080x800.
            p_Eyes.open(AData.Driver, Config.getConfigProperty(Config.ConfigProperty.TEST_APP_NAME),
AData.TestName,AData.ViewPort);
        } catch (Exception e) {
            Log.error("Unable to initialize eyes", e);
        }



        return;
    }



    public void appliCompareImage(AppliCompareData AData) {
        if(!p_AppliEnabled)
        {
            Log.info("Applitools not enabled, provide a valid batch id");
            return;
        }



        //initialize eyes object
        if (p_Eyes =null || !p_Eyes.getIsOpen()) {
            initializeEyes(AData);
        }



        //Set Full Page Screenshot
        p_Eyes.setForceFullPageScreenshot(AData.FullScreenShot);



        //Set Comparison Level
        p_Eyes.setMatchLevel(AData.Compare);




        // Visual checkpoint
        p_Eyes.checkWindow(AData.StepName);
    }



    public TestResults getTestResults() {



        if(!p_AppliEnabled || p_Eyes =null)
        {
            Log.info("Applitools not enabled, provide a valid batch id or do an actual image comparision");
            return null;
        }



        TestResults results =_Eyes.close(false);
        p_Eyes.abortIfNotClosed();
        p_EyesClosed =rue;
        return results;
    }



    public void closeAppliTools()
    {
        if(!p_AppliEnabled)
        {
            Log.info("Applitools not enabled, provide a valid batch id");
            return;
        }



        if(!p_EyesClosed)
        {
            TestResults results =_Eyes.close(false);
            p_Eyes.abortIfNotClosed();
            p_EyesClosed =rue;
            Log.info("Test results:" + results) ;
        }
    }
}
```

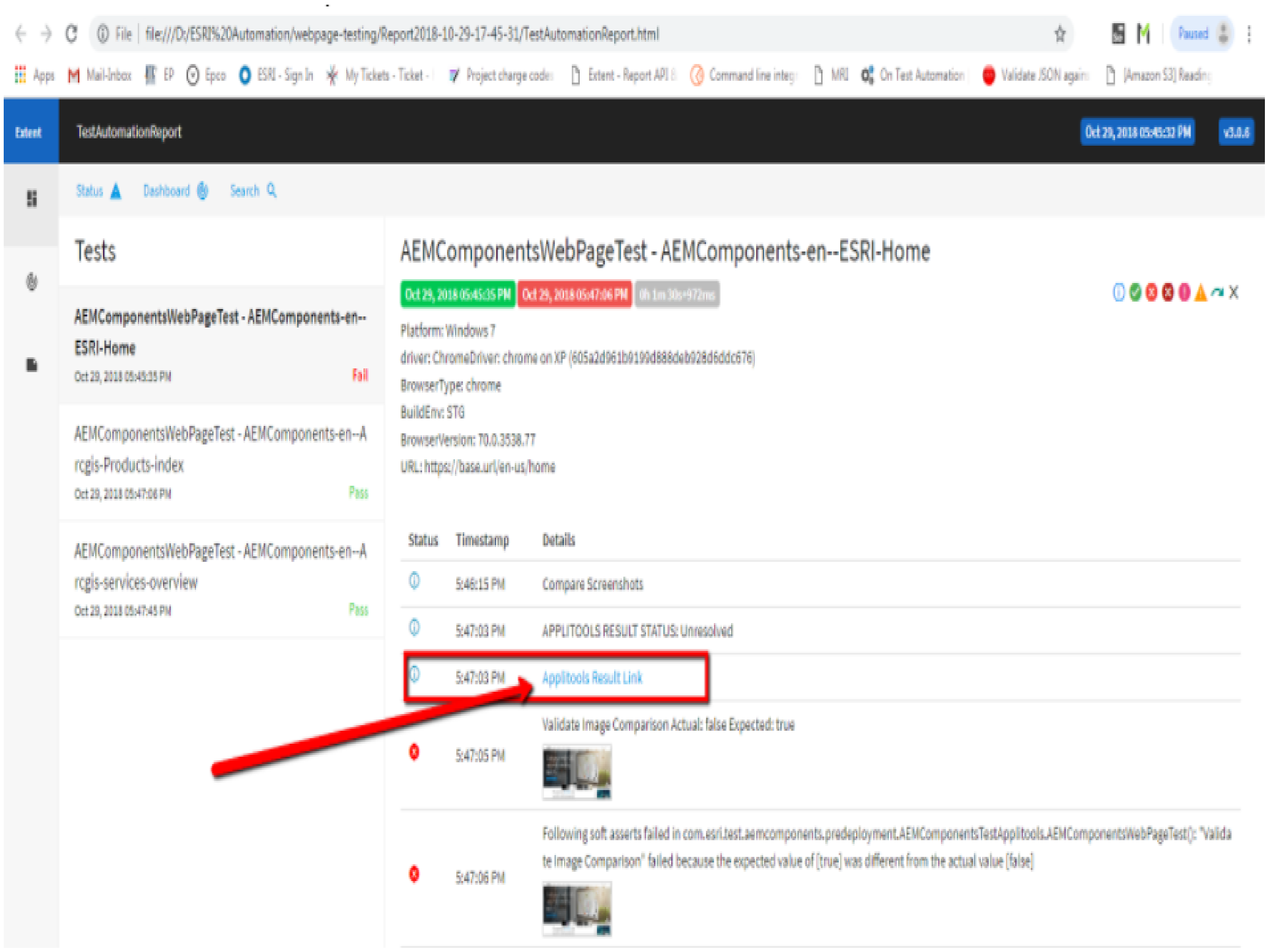
Add Viewport size java class

```
package com.esri.qa.ui;



import com.applitools.eyes.RectangleSize;



public class ViewPortSize {



    public static RectangleSize MAX =ew RectangleSize(1600, 758);
    public static RectangleSize IPHONE7 =ew RectangleSize(375, 667);
    public static RectangleSize IPHONE7PLUS =ew RectangleSize(414, 736);
    public static RectangleSize SAMSUNG =ew RectangleSize(480, 853);
    public static RectangleSize NEXUS =ew RectangleSize(411, 731);
    public static RectangleSize TABLET =ew RectangleSize(768, 1024);
    public static RectangleSize DESKTOP =ew RectangleSize(768, 1024);

 }
```

Sample Java code the Applitools test in Project:

```
package com.esri.test.aemcomponents.predeployment;



import java.util.Map;
import org.testng.annotations.Optional;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;



import com.esri.qa.config.Env;
import com.esri.qa.reporting.Log;
import com.esri.qa.ui.AppliCompareData;
import com.esri.qa.ui.BaseTestUi;
import com.esri.qa.ui.ComparisonLevel;
import com.esri.qa.ui.ViewPortSize;
import com.esri.sm.aemcomponents.predeployment.AEMComponentsPredeploymentPageFeatures;



/**
 * This test class to validate the Web Page
 *
 * @author Rizwan
 */
public class AEMComponentsTestForArcgisProOverview extends BaseTestUi {



    private AppliCompareData p_AData;
    private AEMComponentsPredeploymentPageFeatures
aemComponentsPredeploymentPageFeatures;
    private String headerCount =Header";
    private String casestudy =Casestudy";



    @Parameters({ "fileName", "startRow", "endRow", "bType", "bVersion", "platform", "url", "dimension"
})
    public AEMComponentsTestForArcgisProOverview(@Optional String fileName, @Optional String
startRow,
            @Optional String endRow, @Optional String bType, @Optional String bVersion, @Optional
String platform,
            @Optional String url, @Optional String dimension) {
        super(fileName, startRow, endRow, bType, bVersion, platform, url, dimension);
    }



    /**
     * Test to validate the AEM Components
     *
     * @param inputDatamap
     */
    @Test(dataProvider =csv", priority =)
    public void AEMComponentsWebPageTest(Map<String, String> inputDataMap) {



        // initialize objects
        aemComponentsPredeploymentPageFeatures =ew
AEMComponentsPredeploymentPageFeatures(driver, inputDataMap);



        // Navigate to base url
        aemComponentsPredeploymentPageFeatures
                .openURL(inputDataMap.get("url").replaceAll("base.url", Env.get("AEMCOMPONENTS")));
        aemComponentsPredeploymentPageFeatures.acceptCookies();
        reportLogger.info("Compare Screenshots");
        p_AData =ew AppliCompareData(driver, inputDataMap.get("TestCaseName"),
                String.format("%s - %s", Env.get("AEMCOMPONENTS"),
inputDataMap.get("TestCaseName")),
                ComparisonLevel.CONTENT, true, ViewPortSize.MAX);
        AppliCompareImage(p_AData);

}
```

Once we executed the Java Code the Applitools Report link will be added in the Extended Report.



Tests

AEMComponentsWebPageTest - AEMComponents-en--ESRI-Home
Oct 29, 2018 05:45:35 PM Fail

AEMComponentsWebPageTest - AEMComponents-en--Arcgis-Products-index
Oct 29, 2018 05:47:06 PM Pass

AEMComponentsWebPageTest - AEMComponents-en--Arcgis-services-overview
Oct 29, 2018 05:47:45 PM Pass

AEMComponentsWebPageTest - AEMComponents-en--ESRI-Home

Oct 29, 2018 05:45:35 PM Oct 29, 2018 05:47:06 PM 0h 1m 30s+972ms

Platform: Windows 7
driver: ChromeDriver: chrome on XP (605a2d961b9199d888deb928d6ddc676)
BrowserType: chrome
BuildEnv: STG
BrowserVersion: 70.0.3538.77
URL: https://base.url/en-us/home

Status	Timestamp	Details
ⓘ	5:46:15 PM	Compare Screenshots
ⓘ	5:47:03 PM	APPLITOOLS RESULT STATUS: Unresolved
ⓘ	5:47:03 PM	Applitools Result Link
⊗	5:47:05 PM	Validate Image Comparison Actual: false Expected: true
⊗	5:47:06 PM	Following soft asserts failed in com.esri.test.aemcomponents.predeployment.AEMComponentsTestApplitools.AEMComponentsWebPageTest(): "Validate Image Comparison" failed because the expected value of [true] was different from the actual value [false]

Logging

log messages can be recorded/printed in the test classes using the Log.class file. Just use the static info, debug, error or warn methods to log in the calling code. It can also log the throwable exceptions.

Usage

Log.info("message") or Log.info("message", throwable)

TestNg Utilities

the TestNg utilities are provided to create data provider objects. refer a test case using CSV file for the usage.

UI Utilities:

UI utilities using the page object model to interact with browser and drivers are also created. We have configurations to run the test cases on any browser or platform. The tests can be run locally or on third party Cross Browser Testing platform. The configuration to switch between environments is provided in the test.properties file. More details to follow

Exception Handling:

Exception handling is very important aspect of our application and framework.

Exceptions should be handled in a way that the test output report prints the exception encountered in the test case. The framework is designed in such a way that the output report will show the exception with its stack trace if the test case is written following the proposed strategy as below:

- Use Try-Catch is the code which is risky and can throw exceptions.
- If an exception occurs, there are two ways of handling depending upon the severity.

1) In the catch block re-throw the exception or custom exception if there is no point of executing the rest of the case. This will be the scenario in most cases. E.g.

catch (Exception ex) {

throw new CustomExceptionClass ("message", ex);

2) if we can proceed with the test case then the exception can be simply logged and move forward.

- The thrown exception will be logged in the report