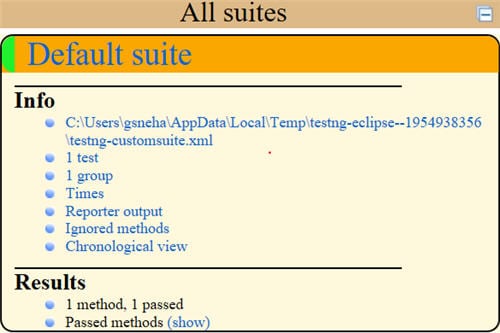
Selenium provides inbuilt reports using frameworks such as JUnit and TestNG.

Although the built-in reports provide information on the steps that are executed as part of the test case, they need more customization to be shared with all the major project stakeholders.

**Below is the snapshot of the built-in report provided by TestNG framework:**



**What are Extent Reports?**

Once an automated test script runs successfully, testers need to generate a test execution report. While TestNG testing framework does provide a default report, they do not provide the detail and easy readability of Extent Reports in Selenium WebDriver.

Extent Reports is a logger-style API written for Java and .NET environments which allows creating HTML reports from tests.

ExtentReports is an open-source reporting library useful for test automation. It can also be easily integrated with major testing frameworks like JUnit, NUnit, TestNG, etc. These reports are HTML rich documents that depict results as pie charts. They also allow the generation of custom logs(stepwise report generation), snapshots, and other customized details.

**Advantages of using Extent Reports**

* Customizable HTML report with stepwise and pie chart representation.
* Displays the time taken for test case execution within the report.
* If required, screenshots can be taken and displayed for each step in a test.
* Allows easy tracking of multiple test case runs in a single test suite.
* Easy to integrate with numerous frameworks, especially TestNG and JUnit.

**Other Reporting tools in Market:**

Different types of other reporting Tools like XSLT Reporting, Allure Reporting mentioned below: [Based on project practice these can also be used, just remember the name as of now]

XSLT Report: <http://total-qa.com/generate-xslt-report-ant-build/>

Allure Report: <http://total-qa.com/advanced-selenium/allure-reporting/>

**Using Extent Reports in Selenium**

Extent Reports in [Selenium](https://www.browserstack.com/selenium) contain two major, frequently used classes:

* ExtentReports class
* ExtentTest class

**Syntax**

ExtentReports reports = new ExtentReports("Path of directory to store the resultant HTML file", true/false);

ExtentTest test = reports.startTest("TestName");

**ExtentReports** class generates HTML reports based on a user-specified path. The Boolean flag notes if the existing report needs to be overwritten or a completely new report needs to be generated. ‘True’ is the default value, which indicates that all existing data will be overwritten.

**ExtentTest** class is required in order to log test steps onto the previously generated HTML report.

Both these classes can be used with the frequently used built-in methods:

* startTest
* endTest
* Log
* flush

**startTest** and **endTest** methods are utilized to execute preconditions and post conditions of a test case. The log method is utilized to log the status of each test step onto the resultant HTML report. The flush method is used to erase any previous data on a relevant report and create a whole new report.

Test Status can be denoted by any of the following values:

* PASS
* FAIL
* SKIP
* INFO

**Syntax**

reports.startTest("TestName");

reports.endTest();

test.log(LogStatus.PASS,"Test Passed");

test.log(LogStatus.FAIL,"Test Failed");

test.log(LogStatus.SKIP,"Test Skipped");

test.log(LogStatus.INFO,"Test Info");

The Log method takes in two parameters, the first being the test status and the second being the message to be printed onto the generated report.

We can provide below information to the report using XML file:

1. Report Title
2. Report Name
3. Report Headline
4. Date Format
5. Time Format

Apart from the above we can provide **HostName**, **Environment Information** and **User Name** etc…

<http://automationtesting.in/generating-extent-reports-java/>

**How to generate Extent Reports – steps:**

**Step #1:**

Extent Reports can be directly used in selenium WebDriver by importing the JAR file – extentreports-java-version.jar

Once the ZIP file is downloaded, extract the contents of the ZIP file into a folder.

<https://mvnrepository.com/artifact/com.aventstack/extentreports/3.1.5>

**Step #2:**

Add the jar files present in the ZIP file to the project build path using the option **Build Path -> Configure Build Path.**

If Maven project add the respective dependency in POM.xml file

**Step #3:**

**Create a new JAVA class with the below code for Extent Reports.**

Example 1: How to have different log status in Extent Report

**package** com.google.tests;

**import** java.io.IOException;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.testng.Assert;

**import** org.testng.annotations.Test;

**import** com.aventstack.extentreports.ExtentReports;

**import** com.aventstack.extentreports.ExtentTest;

**import** com.aventstack.extentreports.MediaEntityBuilder;

**import** com.aventstack.extentreports.Status;

**import** com.aventstack.extentreports.reporter.ExtentHtmlReporter;

**public** **class** ExtentReportsEx {

WebDriver driver;

@Test

**public** **void** verifyHomePageTitle() **throws** IOException

{

ExtentHtmlReporter htmlReporter = **new** ExtentHtmlReporter("extent.html");

// create ExtentReports and attach reporter(s)

ExtentReports extent = **new** ExtentReports();

extent.attachReporter(htmlReporter);

// creates a toggle for the given test, adds all log events under it

ExtentTest test = extent.createTest("verifyHomePageTitle", "Checking the Title");

System.*setProperty*("webdriver.chrome.driver", "G:\\Selenium RS 2019\\chromedriver\_win32\\chromedriver.exe");

driver = **new** ChromeDriver();

// log(Status, details)

test.log(Status.***INFO***, "Chrome Browser Launched Successfully");

driver.get("http://total-qa.com");

test.log(Status.***INFO***,"Navigated to URL");

String actual = driver.getTitle();

test.log(Status.***INFO***, "Actual Title returned :: " + actual);

String expected = "Total-QA - Future of Software Testing";

test.log(Status.***INFO***, "Expected Title returned:: "+ expected);

Assert.*assertEquals*(actual,expected);

// log with snapshot

test.pass("details", MediaEntityBuilder.*createScreenCaptureFromPath*("screenshot.png").build());

// test with snapshot

test.addScreenCaptureFromPath("screenshot.png");

// calling flush writes everything to the log file

extent.flush();

}

}

**Output:**

[file:///G:/Selenium/eclipse-workspace/FirstDemo/extent.html#!](file:///G:\Selenium\eclipse-workspace\FirstDemo\extent.html#!) Use your extent file in the browser

PASSED: verifyHomePageTitle

===============================================

Default test

Tests run: 1, Failures: 0, Skips: 0

===============================================

**Example 2: How to attach screenshot in Extent Report**

**package** com.google.tests;

**import** java.io.IOException;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.testng.Assert;

**import** org.testng.ITestResult;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.Test;

**import** com.aventstack.extentreports.ExtentReports;

**import** com.aventstack.extentreports.ExtentTest;

**import** com.aventstack.extentreports.MediaEntityBuilder;

**import** com.aventstack.extentreports.reporter.ExtentHtmlReporter;

**public** **class** ExtentReportDemo2 {

// Create global variable which will be used in all method

ExtentReports extent;

ExtentTest logger;

WebDriver driver;

// This code will run before executing any testcase

@BeforeMethod

**public** **void** setup()

{

ExtentHtmlReporter reporter=**new** ExtentHtmlReporter("./Reports/automation2.html");

extent = **new** ExtentReports();

extent.attachReporter(reporter);

logger=extent.createTest(“loginTest”,"LoginTest");

}

// Actual Test which will start the application and verify the title

@Test

**public** **void** loginTest() **throws** IOException

{

System.*setProperty*("webdriver.chrome.driver", "G:\\Selenium RS 2019\\chromedriver\_win32\\chromedriver.exe");

driver=**new** ChromeDriver();

driver.get("http://www.google.com");

System.***out***.println("title is "+driver.getTitle());

Assert.*assertTrue*(driver.getTitle().contains("selenium"));

}

// This will run after testcase and it will capture screenshot and add in report

@AfterMethod

**public** **void** tearDown(ITestResult result) **throws** IOException

{

**if**(result.getStatus()==ITestResult.***FAILURE***)

{

String temp=Utility.*getScreenshot*(driver);

logger.fail(result.getThrowable().getMessage(), MediaEntityBuilder.*createScreenCaptureFromPath*(temp).build());

}

extent.flush();

driver.quit();

}

}

**package** com.google.tests;

**import** java.io.File;

**import** java.io.IOException;

**import** org.apache.commons.io.FileUtils;

**import** org.openqa.selenium.OutputType;

**import** org.openqa.selenium.TakesScreenshot;

**import** org.openqa.selenium.WebDriver;

**public** **class** Utility {

**public** **static** String getScreenshot(WebDriver driver)

{

TakesScreenshot ts=(TakesScreenshot) driver;

File src=ts.getScreenshotAs(OutputType.***FILE***);

String path=System.*getProperty*("user.dir")+"/TCscreenshot/"+System.*currentTimeMillis*()+".png";

File destination=**new** File(path);

**try**

{

FileUtils.*copyFile*(src, destination);

} **catch** (IOException e)

{

System.***out***.println("Capture Failed "+e.getMessage());

}

**return** path;

}

}

**Output:**

[file:///G:/Selenium/eclipse-workspace/FirstDemo/Reports/automation.html#!](file:///G:\Selenium\eclipse-workspace\FirstDemo\Reports\automation.html#!) Use your extent file in the browser

title is Google

===============================================

Mavendemo

Total tests run: 1, Passes: 0, Failures: 1, Skips: 0

===============================================

**Important points to be remember:**

1. ExtentReports will create the report file.
2. ExtentTest will log the information in the report.
3. **startTest()** method of ExtentReports class is the starting point of the test and it will return the ExtentTest object.
4. We need to capture that object into ExtentTest object.
5. Used this reference to log the information into the report.
6. ExtentReports object will be used to add the report information like Title, Header and Theme etc..
7. And the above configuration need to passed from the external XML file using **loadConfig()** method of ExtentReports class. It will take the XML file path as argument.
8. **endTest()** method of ExtentReports will stop capturing information about the test log.
9. **flush()** method of ExtentReports wil push/write everything to the document.
10. **close()** method of ExtentReports will clear/close all resource of the ExtentReports object.

**Reference links:**

<http://total-qa.com/extent-reports-selenium-webdriver/>

<https://stackoverflow.com/questions/49931846/how-can-i-have-the-soft-asserts-that-i-put-in-my-test-script-be-reported-into-th>

<https://www.techbeamers.com/generate-reports-selenium-webdriver/>

<http://learn-automation.com/extent-report-with-selenium-webdriver/>

<https://www.seleniumeasy.com/selenium-tutorials/creating-extent-reports-in-selenium-example>

<https://www.browserstack.com/guide/extent-reports-in-selenium>

<http://automationtesting.in/generating-extent-reports-java/>

<https://mvnrepository.com/artifact/com.aventstack/extentreports>

<http://learn-automation.com/how-to-capture-screenshot-for-failed-test-cases-in-selenium-webdriver/>