How to generate extent reports in Selenium

By Mithilesh Singh

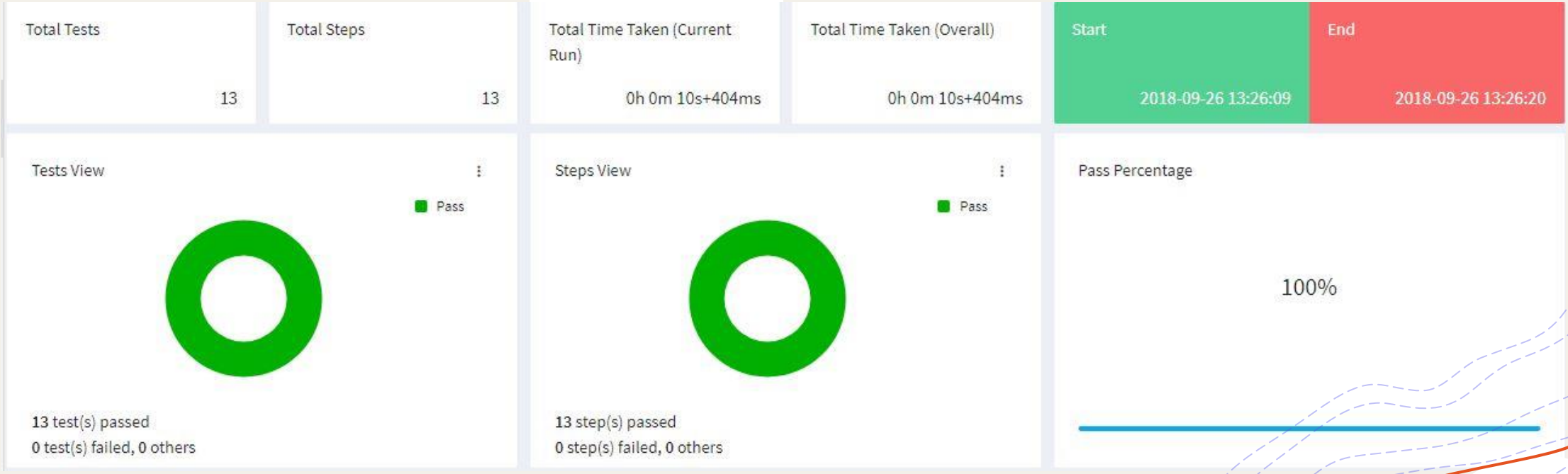
Introduction

In this article, we see one of the most popular and widely used Selenium Reporting Tools (Extent Reports). Selenium Results can be seen using different Selenium Reporting tools. Some of the Selenium WebDriver Reporting tools are as follows.

1. Selenium TestNG Report Generation.
2. Selenium Extent Reports.

Extent Reports

Extent Reports is an open-source reporting library used in selenium test automation. Extent reports become the first choice of Selenium Automation Testers, **even though Selenium comes with inbuilt reports using frameworks like JUnit and TestNG.**



Pre-requisites to Generate Extent Reports:

- Java should be installed – Install and setup Java
- TestNG should be installed – Install TestNG
- Extent Report Jars (Version Latest) – Download
- extent-config.xml – It allows to configure HTML Report

Steps to generate report

- Firstly, create a TestNG project in eclipse
- Now download extent library files from the following link: https://jar-download.com/artifacts/com.aventstack/extentreports/5.0.6
- Add the downloaded library files to your project.
  Steps:

  Right click on your project.

  Select Build Path.

  Click on Configure Build Path.

  Click on Libraries and select **Add External JARs**.

  Select the **jar** file from the required folder.

  Click on Apply and Ok.
- Create a java class and add the following code to it

```
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.apache.commons.io.FileUtils;
import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.ITestResult;
import org.testng.annotations.*;
import com.aventstack.extentreports.ExtentReports;
import com.aventstack.extentreports.ExtentTest;
import com.aventstack.extentreports.Status;
import com.aventstack.extentreports.reporter.ExtentHtmlReporter;
import com.aventstack.extentreports.reporter.configuration.Theme;

public class NopCommerceTest {
public WebDriver driver;
public ExtentHtmlReporter htmlReporter;
public ExtentReports extent;
public ExtentTest test;

@BeforeTest
public void setExtent() {
 // specify location of the report
 htmlReporter = new ExtentHtmlReporter(System.getProperty("user.dir")  + "/test-output/myReport.html");
 htmlReporter.config().setDocumentTitle("Automation  Report"); // Tile of report
 htmlReporter.config().setReportName("Functional  Testing"); // Name of the report
 htmlReporter.config().setTheme(Theme.DARK);
 extent = new ExtentReports();
 extent.attachReporter(htmlReporter);
 // Passing General information
 extent.setSystemInfo("Host  name", "localhost");
 extent.setSystemInfo("Environemnt",  "QA");
 extent.setSystemInfo("user",  "Mithilesh");
}
```

```
@AfterTest
public void endReport(){
 extent.flush();
}
@BeforeMethod
public void setup(){
 System.setProperty("webdriver.chrome.driver","C://Drivers/chromedriver_win32/chromedriver.exe");
 driver = new ChromeDriver();
 driver.manage().window().maximize();
 driver.get("http://demo.nopcommerce.com/");
}
//Test1
@Test
public void noCommerceTitleTest(){
 test = extent.createTest("noCommerceTitleTest");
 String title = driver.getTitle();
 System.out.println(title);
 Assert.assertEquals(title,"eCommerce demo store");
}
//Test2
@Test
public void noCommerceLogoTest(){
 test = extent.createTest("noCommerceLogoTest");
 boolean b = driver.findElement(By.xpath("//img[@alt='nopCommerce demo store']")).isDisplayed();
 Assert.assertTrue(b);
}
//Test3
@Test
public void noCommerceLoginTest(){
 test = extent.createTest("noCommerceLoginTest");
 test.createNode("Login with Valid input");
 Assert.assertTrue(true);
 test.createNode("Login with In-valid input");
 Assert.assertTrue(true);
}
```

```
@AfterMethod
public void tearDown(ITestResult result) throws IOException {
 if (result.getStatus() == ITestResult.FAILURE) {
  test.log(Status.FAIL, "TEST CASE FAILED IS " + result.getName()); // to add name in extent report
  test.log(Status.FAIL, "TEST CASE FAILED IS " + result.getThrowable()); // to add error/exception in extent report
  String screenshotPath = NopCommerceTest.getScreenshot(driver, result.getName());
  test.addScreenCaptureFromPath(screenshotPath);// adding screen shot
 } else if (result.getStatus() == ITestResult.SKIP) {
  test.log(Status.SKIP, "Test Case SKIPPED IS " + result.getName());
 }
 else if (result.getStatus() == ITestResult.SUCCESS) {
  test.log(Status.PASS, "Test Case PASSED IS " + result.getName());
 }
 driver.quit();
}
//how to take screen shot
public static String getScreenshot(WebDriver driver, String screenshotName) throws IOException {
 String dateName = new SimpleDateFormat("yyyyMMddhhmmss").format(new Date());
 TakesScreenshot ts = (TakesScreenshot) driver;
 File source = ts.getScreenshotAs(OutputType.FILE);

 // after execution, you could see a folder "FailedTestsScreenshots" under src folder
 String destination = System.getProperty("user.dir") + "/Screenshots/" + screenshotName + dateName + ".png";
 File finalDestination = new File(destination);
 FileUtils.copyFile(source, finalDestination);
 return destination;
}
}
```

How to take screenshot

Important points

Extent reports library provides following classes:

ExtentHtmlReporter : It helps for look and feel of report - report name, report generation directory, doc title and theme of the report

```
htmlReporter= new
ExtentHtmlReporter(System.getProperty("user.dir")+"/test-
output/myReport.html");
```

+ **Note**: System.getProperty("user.dir")--> it will give the current directory location.

```
htmlReporter.config().setDocumentTitle("<mention document title>");

htmlReporter.config().setReportName("<name of the report>");

htmlReporter.config().setTheme(Theme.DARK);
```

ExtentReports : We can add Author name, browser name, OS name in the report using this class object.

+ extent.attachReporter(htmlReporter);
+ extent.setSystemInfo("Hostname","Localhost");
+ extent.setSystemInfo("OS","Windows10");
+ extent.setSystemInfo("Tester Name","TestersZone");
+ extent.setSystemInfo("Browser","Chrome");
+ **ExtentTest** :we create the variable with the reference of this class and intantiate it with the help of ExtentReports object

ExtentTest test:

+ test= extent.createTest("<any name>");---> this line of code will register the test method or test case in the report.
+ It also use to update the logs of the test execution(can be observed in previous snippet of code).

Configure extent xml file with java class

Use below line of code to load the xml file.
extent.loadConfig(new File(System.getProperty("user.dir")+"\\extent-config.xml"));

+ **Note:**
+ 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.

extent-config.xml

this file we will get while expanding downloaded extent reports jar. We can use this file in framework as it is.

```
<?xml version="1.0" encoding="UTF-8"?><extentreports>
 <configuration>
    <!-- report theme -->
 <!-- standard, dark -->
   <theme>standard</theme>
    <!-- document encoding -->
  <!-- defaults to UTF-8 -->
    <encoding>UTF-8</encoding>
    <!-- protocol for script and stylesheets -->
<!-- defaults to https -->
<protocol>https</protocol>
    <!-- title of the document -->
   <documentTitle>ExtentReports 2.0</documentTitle>
    <!-- report name - displayed at top-nav-->
<reportName></reportName>

 <!-- report headline - displayed at top-nav, after reportHeadline
-->
<reportHeadline>Automation Report</reportHeadline>
```


```
<!-- global date format override -->
   <!-- defaults to yyyy-MM-dd -->
   <dateFormat>yyyy-MM-dd</dateFormat>
    <!-- global time format override -->
    <!-- defaults to HH:mm:ss -->
<timeFormat>HH:mm:ss</timeFormat>
<!-- custom javascript -->
<scripts>
<![CDATA[
  $(document).ready(function() {
        });
  ]]>
  </scripts>
   <!-- custom styles -->
  <styles>
<![CDATA[
 ]]>
 </styles>
 </configuration>
</extentreports>
```

Till last slide I have explained how we can use extent report in our framework. But we can also use testNG listeners to add more details in the report.
So let's see listener's class with extent report concept.

Steps:

- We have so many interfaces which can be used, here we will use ITestResult interface. There is TestListenerAdapter class which implements ITestResult interface so we can easily extends TestListenerAdapter class and override the method.

Step 1: Create a Class Listeners extends TestListenerAdapter class.

Step 2: Create a Test Case LoginTest.java

Step 3: Create listener.xml file to run test case

Step 4: Listener.java

Step 1: Create a Class Listeners extends TestListenerAdapter class.

```
+ import org.testng.ITestResult;
  import org.testng.TestListenerAdapter;

  public class Listeners extends TestListenerAdapter {
   public void onTestStart(ITestResult tr) {
   System.out.println("test  is started");
   }
   public void onTestSuccess(ITestResult tr) {
   System.out.println("  test is passed");
   }
   public void onTestFailure(ITestResult  tr) {
   System.out.println("  test is failed");
   }
   public void onTestSkipped(ITestResult  tr) {
   System.out.println("  test is skipped");
   }
  }
```

Step 2: Create a Test Case LoginTest.java

```
import org.testng.Assert;
import org.testng.annotations.Test;
public class LoginTest extends BaseClass{
 @Test
 void setup()
 {
 Assert.fail();
 }
@Test
 void loginByEmail()
 {
 Assert.assertTrue(true);
 }
@Test(dependsOnMethods={"setup"})
 void loginByFacebook()
 {
 Assert.assertTrue(true);
 }
}
```

Base Class

```
import java.io.File;
import java.io.IOException;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.testng.Assert;
import org.testng.ITestResult;
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.Test;
import org.apache.commons.io.FileUtils;
public class LoginTest {
WebDriver driver;
@BeforeTest
      <launch the browser>
   @AfterMethod
public void captureScreen(ITestResult result) throws IOException{
 if(result.getStatus()==ITestResult.FAILURE) {
TakesScreenshot ts=(TakesScreenshot)driver;
File source=ts.getScreenshotAs(OutputType.FILE);  // capture screenshot file
File target=new File(System.getProperty("user.dir")+"/Screenshots/"+result.getName()+".png");
FileUtils.copyFile(source,target);
System.out.println("screenshot captured");
} }
@AfterTest
void closeBrowser(){
driver.quit();
} }
```

Step 3: Create listener.xml file to run test case

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name="TestNGListeners">

    <listeners>
        <listener class-name="reporting.Listeners" />
    </listeners>


    <test name="logintest">
        <classes>

            <class name="reporting.LoginTest">
            </class>

        </classes>

    </test>
</suite>
```

Listener.java

```
import java.io.IOException;
import org.testng.ITestContext;
import org.testng.ITestResult;
import org.testng.TestListenerAdapter;
import com.aventstack.extentreports.ExtentReports;
import com.aventstack.extentreports.ExtentTest;
import com.aventstack.extentreports.Status;
import com.aventstack.extentreports.markuputils.ExtentColor;
import com.aventstack.extentreports.markuputils.MarkupHelper;
import com.aventstack.extentreports.reporter.ExtentHtmlReporter;
import com.aventstack.extentreports.reporter.configuration.ChartLocation;
import com.aventstack.extentreports.reporter.configuration.Theme;

public class Listeners extends TestListenerAdapter
{
public ExtentHtmlReporter htmlReporter;
public ExtentReports extent;
public ExtentTest logger;
public void onStart(ITestContext testContext)
{
htmlReporter=new ExtentHtmlReporter(System.getProperty("user.dir")+ "/test-output/myReport.html");//specify location of the
report
htmlReporter.loadXMLConfig(System.getProperty("user.dir")+ "/extent-config.xml");
extent=new ExtentReports();
extent.attachReporter(htmlReporter);
extent.setSystemInfo("Host name","localhost");
extent.setSystemInfo("Environemnt","QA");
extent.setSystemInfo("user","pavan");
htmlReporter.config().setDocumentTitle("Automation Report"); // Tile of report
htmlReporter.config().setReportName("Functional Testing"); // name of the report
htmlReporter.config().setTestViewChartLocation(ChartLocation.TOP); //location of the chart
htmlReporter.config().setTheme(Theme.STANDARD);
}
```

```
public void onTestSuccess(ITestResult tr)
{
logger=extent.createTest(tr.getName());//create new entry in the report
logger.log(Status.PASS,MarkupHelper.createLabel(tr.getName(),ExtentColor.GREEN));//send the passed information to
the report with GREEN color highlighted
}
public void onTestFailure(ITestResult tr)
{
logger=extent.createTest(tr.getName());//create new entry in the report
logger.log(Status.FAIL,MarkupHelper.createLabel(tr.getName(),ExtentColor.RED));//send the passed information to the
report with GREEN color highlighted
String screenshotPath=System.getProperty("user.dir")+"\\Screenshots\\"+tr.getName()+".png";
try {
logger.fail("Screenshot is below:" + logger.addScreenCaptureFromPath(screenshotPath));
} catch (IOException e) {
e.printStackTrace();
}
}
public void onTestSkipped(ITestResult tr)
{
logger=extent.createTest(tr.getName());//create new entry in th report
logger.log(Status.SKIP,MarkupHelper.createLabel(tr.getName(),ExtentColor.ORANGE));
}
public void onFinish(ITestContext testContext)
{
extent.flush();
}
}
```

Project Connectivity

- We have to create a listener class and extends TestListenerAdapter class.
- If we have multiple Test Class based on the different web pages. We can create one base class having testNG annotations and extends that class in every test class.
- Listeners class method will be called automatically, we can create one xml file and add listener's class name there and also can add test class in same xml file. Now our xml, listener's class and test class are connected together.
- Extent.xml already loaded in listener's class.

  In this way we can connect all the components of the framework and generate good extent report.

Advantages of Extent Reports:

- It can be easily integrated with frameworks like JUnit, NUnit, & TestNG
- It displays the time taken for test case execution
- Extent reports are more customizable than others
- Extent API can produce more interactive reports, a dashboard view, graphical view, capture screenshots at every test step, and emailable reports


Testers Zone
Thank you
Mithilesh Singh