**\*\*\*\*\*\*\*\*\*\*\*\*How to Generate Extent Reports in Selenium\*\*\*\*\*\*\*\*\*\*\*\*\*\***

### ****\*\*\*\*\*\*\*\*\*\*What are Extent Reports\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*****

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

Once an automated test script runs successfully, testers need to generate a test execution report. While TestNG does provide a default report, they do not provide the details.

### ****\*\*\*\*\*\*\*\*\*\*Using Extent Reports in Selenium\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*****

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

1. Extent Reports class

2. Extent Test class

🡪The Extent Reports class generates HTML reports based on a path specified by the tester. Based on the Boolean flag, the existing report has to be overwritten or a new report must be generated. ‘True’ is the default value, meaning that all existing data will be overwritten.

🡪The Extent Test class logs test steps onto the previously generated HTML report.

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

* **startTest**: Executes pre conditions of a test case.
* **endTest**: Executes post conditions of a test case.
* **Log:** Logs the status of each test step onto the HTML report being generated
* **Flush:** Erases any previous data on a relevant report and creates a whole new report

\*\*\*\*\*\*\*\*\*\*\*\*A Test Status can be indicated by the following values:

* PASS
* FAIL
* SKIP
* INFO

**Syntax:-**

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 into account two parameters, the first being the test status and the second being the message to be printed onto the generated report.

How to generate Extent Reports (with code samples)

The Log method uses two parameters – 1) test status and 2) the message to be printed on the generated report.

### ****\*\*\*\*\*\*\*\*How to generate Extent Reports****

1. Import the JAR file : **extentreports-java-2.41.2.jar** . After downloading the ZIP file, extract its contents into a folder.
2. Add the JAR files to the project build path with the option **Build Path -> Configure** Build Path.
3. Create a new JAVA class for Extent Reports with the code below.

package com.browserstack.demo;

import org.junit.AfterClass;

import org.junit.BeforeClass;

import org.junit.Test;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import com.relevantcodes.extentreports.ExtentReports;

import com.relevantcodes.extentreports.ExtentTest;

import com.relevantcodes.extentreports.LogStatus;

public class ExtentDemo {

static ExtentTest test;

static ExtentReports report;

@BeforeClass

public static void startTest()

{

report = new ExtentReports(System.getProperty("user.dir")+"ExtentReportResults.html");

test = report.startTest("ExtentDemo");

}

@Test

public void extentReportsDemo()

{

System.setProperty("webdriver.chrome.driver", "D:SubmittalExchange\_TFSQAAutomation3rdpartychromechromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("https://www.google.co.in");

if(driver.getTitle().equals("Google"))

{

test.log(LogStatus.PASS, "Navigated to the specified URL");

}

else

{

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

}

}

@AfterClass

public static void endTest()

{

report.endTest(test);

report.flush();

}

}

Test execution commences with the **startTest** method. It also initializes the Extent Reports object. Any valid user-defined path can be the parameter passed onto the Extent Reports object.

**@Test**: This class automates the following actions:

* Open Chrome browser with the url **https://www.google.com**
* After the page opens, validate page title with the expected
* Log the test case status as PASS/FAIL using the log method explained before

**@AfterClass:** Executes postconditions of the test case – ending the test (using **endTest** method) and flushing the report.

***Note: Don’t forget to use the flush() method, since the report will not be generated otherwise.***

### ****\*\*\*\*\*\*\*\*\*\*\*\*How to capture screenshots in Extent Report?****

By capturing screenshots, testers can better identify what went wrong when the software acted erroneously during a test. Capture screenshots only when a test fails, since they consume a lot of memory.

Try capturing screenshots with the code below

test.log(LogStatus.FAIL,test.addScreenCapture(capture(driver))+ "Test Failed");

public static String capture(WebDriver driver) throws IOException {

File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

File Dest = new File("src/../BStackImages/" + System.currentTimeMillis()

+ ".png");

String errflpath = Dest.getAbsolutePath();

FileUtils.copyFile(scrFile, Dest);

return errflpath;

}

**getScreenShotAs():** Captures screenshot of the current WebDriver instance and stores it in different output forms.

File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

This method returns a file object to be stored onto a file variable. Casting the web driver instance to Take Screenshot is necessary to use the method.

File Dest = new File("src/../BStackImages/" + System.currentTimeMillis()+ ".png");

This statement creates a folder named ‘**BStackImages**’ within the ‘**src**’ folder and stores the file name as the current system time.

String errflpath = Dest.getAbsolutePath();

FileUtils.copyFile(scrFile, Dest);

returnerrflpath;

These statements copy all error images to the destination folder.

Use the log method because it uses the **addScreenCapture** method of **Extent Test** class to get a screenshot and add it to the **Extent Report**.

test.log(LogStatus.FAIL,test.addScreenCapture(capture(driver))+ "Test Failed");

### 

### ****\*\*\*\*\*\*\*\*\*\*Benefits of using Extent Reports:-****

* They can be integrated with TestNG and Junit.
* If required, screenshots can be captured and displayed for each step in a test
* They allow testers to track multiple test case runs in a single test suite
* They show the time needed for test execution
* They can be customized to graphically represent each step in a test.

Try using the code defined above to generate Extent Reports. By doing so, users can offer themselves and their teams, a more extensive and insightful perspective on the success or failure of their [automated Selenium testing](https://www.browserstack.com/automate). By accessing the greater detail provided by Extent Reports, testers can be more effective when it comes to debugging software quickly.