**4 Steps to configure extent reports**

1. **Add dependencies in pom.xml🡪** aventstack, relevantcodes extentreports
2. **Add Listener class in Listener package (Standard format)**
3. **Add listener in testing.xml file**
4. **Please refresh your project to see reports**

**package** listener;

**import** java.io.File;

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.Map;

**import** org.testng.IReporter;

**import** org.testng.IResultMap;

**import** org.testng.ISuite;

**import** org.testng.ISuiteResult;

**import** org.testng.ITestContext;

**import** org.testng.ITestResult;

**import** org.testng.xml.XmlSuite;

**import** com.relevantcodes.extentreports.ExtentReports;

**import** com.relevantcodes.extentreports.ExtentTest;

**import** com.relevantcodes.extentreports.LogStatus;

**public** **class** ExtentReporterNG **implements** IReporter {

**private** ExtentReports extent;

**public** **void** generateReport(List<XmlSuite> xmlSuites, List<ISuite> suites,

String outputDirectory) {

String timeStamp= **new** SimpleDateFormat("yyyy.MM.dd.HH.mm.ss").format(**new** Date());

extent = **new** ExtentReports(System.*getProperty*("user.dir")+"/REports/"+

"Velocity"+timeStamp+".html", **true**);

**for** (ISuite suite : suites) {

Map<String, ISuiteResult> result = suite.getResults();

**for** (ISuiteResult r : result.values()) {

ITestContext context = r.getTestContext();

buildTestNodes(context.getPassedTests(), LogStatus.***PASS***);

buildTestNodes(context.getFailedTests(), LogStatus.***FAIL***);

buildTestNodes(context.getSkippedTests(), LogStatus.***SKIP***);

}

}

extent.flush();

extent.close();

}

**private** **void** buildTestNodes(IResultMap tests, LogStatus status) {

ExtentTest test;

**if** (tests.size() > 0) {

**for** (ITestResult result : tests.getAllResults()) {

test = extent.startTest(result.getMethod().getMethodName());

test.setStartedTime(getTime(result.getStartMillis()));

test.setEndedTime(getTime(result.getEndMillis()));

**for** (String group : result.getMethod().getGroups())

test.assignCategory(group);

**if** (result.getThrowable() != **null**) {

test.log(status, result.getThrowable());

} **else** {

test.log(status, "Test " + status.toString().toLowerCase()

+ "ed");

}

extent.endTest(test);

}

}

}

**private** Date getTime(**long** millis) {

Calendar calendar = Calendar.*getInstance*();

calendar.setTimeInMillis(millis);

**return** calendar.getTime();

}

}

<dependency>

<groupId>com.relevantcodes</groupId>

<artifactId>extentreports</artifactId>

<version>2.41.2</version>

</dependency>

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.1.2</version>

</dependency>

**What Are Extent Reports?**

It has been more widely used for report generation than inbuilt reports in various automation testing frameworks because of its enhanced features and customization.

### ****What are Extent Reports?****

Extent Report is an open-source library for generating test reports in automation testing. It can be easily integrated with major testing frameworks like JUnit, TestNG, etc. These reports are HTML documents that shows 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.

**Benefits of using Extent Reports**

* They can be integrated with TestNG and Junit
* 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.
* It is an open-source library.
* It provides a pictorial representation of the test results.
* It can be customized as required.
* It allows users to attach screenshots and logs to the test report for a detailed summary of the tests.
* It can be easily configured with Jenkins, Bamboo, etc.

It is a status chart, which shows no of TC pass, failed result.

It will have multiple options to filter TC according to pass, fail result. When TC pass it will show time execution time. If fails it will show why it is faild, root cause, exceptional logs etc.

It gives overview of how much work done , work in progress, and success. It is better communication tool to share about progress of our automation project with client.