**Serenity BDD:**

* Open source library that aims to make the idea of living documentation a reality.
* Helps to write cleaner and ore maintainable automated acceptance and regression tests.
* Provides strong support for automated web test using selenium and non-web tests.
* Can work with BDD tools like Cucumber or JBehave or simple use Junit.
* Can also integrate with requirements stored in an external source such as JIR or other test cases management tools.
* Serenity BDD is also commonly used for automated Regression Tests. Whereas BDD Acceptance tests are defined early on in the piece, before development starts.

1. Serenity report.
2. Aggregated Serenity report.

Aggregated report can be organized by using features, stories, steps, scenarios/test

Every story can contain one or more scenarios.

Each scenario consists of one or more steps and some example,

default number of examples is 1.

**Building Serenity projects in maven:**

***Serenity BDD integrates*** with **Maven** via ***Serenity-maven-plugin***

***1.<!--Core Serenity dependency -->***

***<dependency>***

***<groupId>net.serenity-bdd</groupId>***

***<artifactId>core</artifactId>***

***<version>${serenity.version}</version>***

***</dependency>***

***2.<!-- JUnit Serenity dependency -->***

***<dependency>***

***<groupId>net.serenity-bdd</groupId>***

***<artifactId>serenity-junit</artifactId>***

***<version>${serenity.version}</version>***

***</dependency>***

***3.<!-- Maven Failsafe plugin -->***

***<plugin>***

***<artifactId>maven-failsafe-plugin</artifactId>***

***<version>2.18</version>***

***<configuration>***

***<includes>***

***4.<!--include only tests in the junit directory -->***

***<include>\*\*/features/\*\*/When\*.java</include>***

***</includes>***

***8.<!--pass the webdriver.driver system property to the tests -->***

***<systemProperties>***

***<webdriver.driver>${webdriver.driver}</webdriver.driver>***

***<surefire.rerunFailingTestsCount>${surefire.rerunFailingTestsCount}</surefire.rerunFailingTestsCount>***

***<surefire.rerunFailingTestsCount>${surefire.rerunFailingTestsCount}</surefire.rerunFailingTestsCount>***

***</systemProperties>***

***</configuration>***

***</plugin>***

***5.<!--The Serenity Maven plugin -->***

***<plugin>***

***<groupId>net.serenity-bdd.maven.plugins</groupId>***

***<artifactId>serenity-maven-plugin</artifactId>***

***<version>${serenity.maven.version}</version>***

***<dependencies>***

***<dependency>***

***<groupId>net.serenity-bdd</groupId>***

***<artifactId>core</artifactId>***

***<version>${serenity.version}</version>***

***</dependency>***

***</dependencies>***

***<executions>***

***<execution>***

***<id>serenity-reports</id>***

***6.<!--generate the aggregate reports during the post-integration test phase -->***

***<phase>post-integration-test</phase>***

***<goals>***

***7.<!-- call the aggregate goal to generate them -->***

***<goal>aggregate</goal>***

***</goals>***

***</execution>***

***</executions>***

***</plugin>***

* First, you need to add the Serenity BDD dependencies to your project.
* You will typically add core and another dependency that correpsonds to the testing library you are using (JUnit in this example). Other supported testing libraries include JBehave and Cucumber.
* You typically want the Serenity tests to run as integration tests (that is, during the integration-test phase of the Maven build) rather than as unit tests.
* You also want the build not to immediately fail when a test fails, but to continue until it has generated the Serenity aggregate reports before failing at the end of the build. To do this, we use the maven-failsafe-plugin (3).
* This plugin runs your integration test in the integration-test phase without immediately failing the build when a test fails. Build failure is triggered later in the lifecycle, during the verify phase. Also it is good idea turn off failing build if some test was failed - just to allow maven execute all tests.

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>xxxx</version>

<configuration>

<testFailureIgnore>true</testFailureIgnore>

</configuration>

</plugin>

* Normal JUnit tests run from Maven need to start or end with Test. But for acceptance tests, a more flexible strategy is better, as it makes it easier to name test cases after scenarios or stories. In the pom.xml file shown above, we configure the maven-failsafe-plugin to run all of the tests in the junit directory, regardless of how they are named (4).
* Next, you need to add and configure the serenity-maven-plugin.Pt. 5 A useful technique is to bind the aggregate goal plugin to the post-integration-test phase. Pt.6 and Pt.7 This way, to run the tests and to generate the reports, you would run the following:

mvn verify

* This will run the tests and generate an aggregate report in the target/site/serenity directory.
* Like the surefire plugin the maven-failsafe-plugin starts a new JVM instance to run the tests. For this reason, if you need to pass system parameters to the tests (for example, the webdriver.driver property shown here), you need to use the <systemProperties> section

Tip: It is possible to use a Junit Run Configuration to run a Serenity Testrunner. This will not generate an aggregate report. If you find your index.html file is missing, check that you are using a Maven build Run Configuration with goal verify to run your test and get the aggregate report.