

Allure Dependency and Annotations

In this lesson, you'll learn about Allure's dependencies and annotations.

We'll cover the following

- Adding dependency
 - Gradle (in build.gradle)
 - Maven (in pom.xml)
- Allure annotations
 - @Step
 - @Attachment

Adding dependency

To use Gradle, we have to add the dependencies below to our project's build file:

Gradle (in `build.gradle`) #

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath "io.qameta.allure:allure-gradle:2.8.1"
    }
}

plugins {
    id 'io.qameta.allure'
}

allure {
    version = '2.13.1'
    autoconfigure = true
    aspectjweaver = true
    clean = true
    allureJavaVersion = "${allure.version}"
    resultsDir = file('test-output/allure-results')
    reportDir = file('test-output/allure-report')
```

```
reportDir = file( test-output/allure-reports )
downloadLink = "https://repo.maven.apache.org/maven2/io/qameta/allure/allur
e-commandline/${allure.version}/allure-commandline-${allure.version}.zip"
}
```

Opening Allure report

To open the Allure report after test execution using Gradle, we can use the following command:

```
./gradlew allureServe
```

Maven (in `pom.xml`) #

```
<properties>
  <aspectj.version>1.9.5</aspectj.version>
</properties>

<dependencies>
  <dependency>
    <groupId>io.qameta.allure</groupId>
    <artifactId>allure-testng</artifactId>
    <version>2.13.1</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>2.22.2</version>
      <configuration>
        <testFailureIgnore>>false</testFailureIgnore>
        <argLine>
          -javaagent:"${settings.localRepository}/org/aspectj/aspect
jweaver/${aspectj.version}/aspectjweaver-${aspectj.version}.jar"
        </argLine>
      </configuration>
    </plugin>
  </plugins>
  <dependencies>
    <dependency>
      <groupId>org.aspectj</groupId>
      <artifactId>aspectjweaver</artifactId>
      <version>${aspectj.version}</version>
    </dependency>
  </dependencies>
</build>
```

```

        </dependencies>
    </plugin>

    <plugin>
        <groupId>io.qameta.allure</groupId>
        <artifactId>allure-maven</artifactId>
        <version>2.10.0</version>
        <configuration>
            <reportVersion>2.13.1</reportVersion>
        </configuration>
    </plugin>
</plugins>
</build>

```

Opening Allure report

To open the Allure report after test execution using Maven, we can use the following command:

```
mvn allure:serve
```

Allure annotations

Allure has few annotations for marking the life cycle of test execution.

- `@Step`
- `@Attachment`

`@Step` #

Any action that constitutes a testing scenario is marked with `@Step`.

```

@Step("send request")
public void sendRequest(String url) {
    ....
}

```

Here, the name of the step will be taken by default if the name parameter is not mentioned in `@Step`. We can also have placeholders in the name parameter. For more information, please follow this [link](#).

This can also be done programmatically, using the code below:

```

import static io.qameta.allure.util.AspectUtils.getName;
import static io.qameta.allure.util.AspectUtils.getParameters;

```

```

import static io.qameta.allure.util.ResultsUtils.getStatus;
import static io.qameta.allure.util.ResultsUtils.getStatusDetails;

import io.qameta.allure.AllureLifecycle;
import io.qameta.allure.Step;
import io.qameta.allure.model.Parameter;
import io.qameta.allure.model.Status;
import io.qameta.allure.model.StepResult;
import io.qameta.allure.Allure;

public void sendRequest() {

    StepResult result = new StepResult().setName("send request");
    Allure.getLifecycle().startStep(UUID.randomUUID().toString(), result);
    try {
        ...
        ...
        getLifecycle().updateStep(s -> s.setStatus(getStatus(e).orElse(Status.
PASSED)));
    } catch (Exception e) {
        getLifecycle().updateStep(s -> s.setStatus(getStatus(e).orElse(Status.
BROKEN)));
        throw e;
    }
}

```

@Attachment #

The method below annotated with **@Attachment** should return either a **String** or **byte[]**. For attaching a file to the report, do as follows:

```

@Attachment(value = "adding log", type = "text/plain")
public String addAttachment() {
    return "hello";
}

```

Alternatively, for doing the same programmatically without using **@Attachment** annotation, do as follows:

```

public void sendRequest() {
    .....
    Allure.getLifecycle().addAttachment("adding log", "text/plain", ".txt", "h
ello");
}

```

Multiple overloaded methods are also available. Please follow this [link](#) for more information.

Additionally, please follow this [link](#) for more comprehensive information about the Allure framework.

That's all about the Allure. In the next lesson, we will go through an example of Allure usage in an API test.