Pom.xml

POM stands for project object model. It's the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used to build the project. It downloads required libraries easily using POM XML tags.

Dependencies are the libraries you would use in your application. If you are not using Maven, You have to download these jar files manually from the internet and add it to your project libraries. But with maven, You can just updated pom.xml with the dependency you want and, Maven will do the downloading and adding it to your project.

To avoid the discrepancy between team members we will use pom.xml file.

When there is no Maven, it needs to add all the library JAR files one by one to the project. But when there is Pom.xml there is no need to add library JAR files one by one.

It will add all libraries automatically.

It contains dependencies to download libraries in it.

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi</artifactId>

<version>3.17</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.14.3</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

</dependency>

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.1.5</version>

</dependency>

**groupId** will identify your **project** uniquely across all **projects**, so we need to enforce a naming schema. It has to follow the package name rules, what means that has to be at least as a domain name you control, and you can create as many subgroups as you want. ... **artifactId** is the name of the jar without version.

GROUP ID:

usually a reversed domain name, like com.example.foo