Maven

What is Maven

- **At its simplest, Maven is a build tool**
  - It always produces one artifact (component)
  - It helps us manage dependencies
- **It can also be used as a project management tool**
  - It handles versioning and releases
  - Describes what your project is doing or what it produces
  - Can easily produce Javadocs as well as other site information

Who owns it

- Maven is managed by the Apache Software Foundation
- Maven sites are built with Maven
- Open Source

Why do you want to use it

- Repeatable builds
  - We can recreate our build for any environment
- Transitive dependencies
  - Downloading a dependency with also pull other items it needs
- Contains everything it needs for your environment
- Works with a local repo
- Works with your IDE, but also standalone
- The preferred choice for working with build tools like Jenkins or Cruise Control

How To Install Maven ?

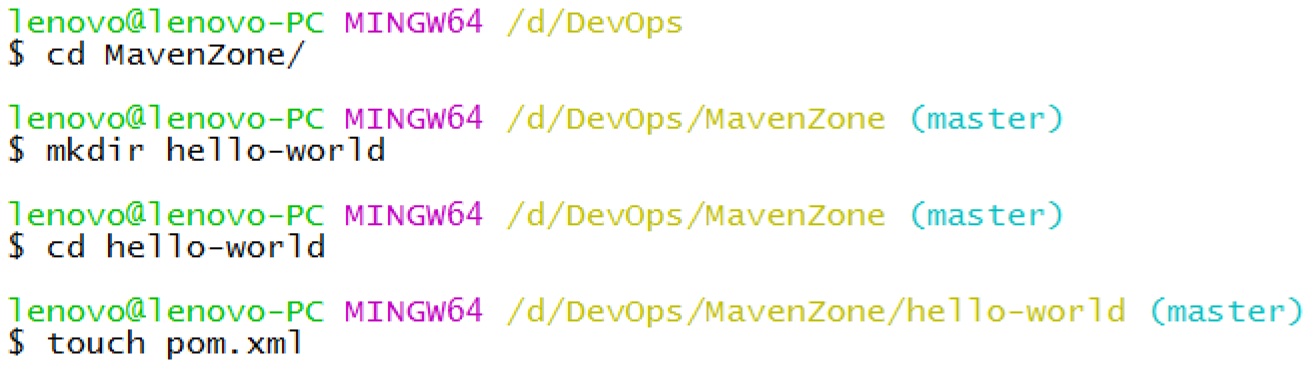
1. https://maven.apache.org/install.html
2. Installing Maven using apt-get "https://www.mkyong.com/maven/how-to-install-maven-in-ubuntu/"
3. Installing maven on Centos / Redhat family https://tecadmin.net/install-apache-maven-on-centos/
4. To install Java follow

https://www.digitalocean.com/community/tutorials/how-to-install-java-on-ubuntu-with-apt-get

https://www.digitalocean.com/community/tutorials/how-to-

Hello World Maven Project

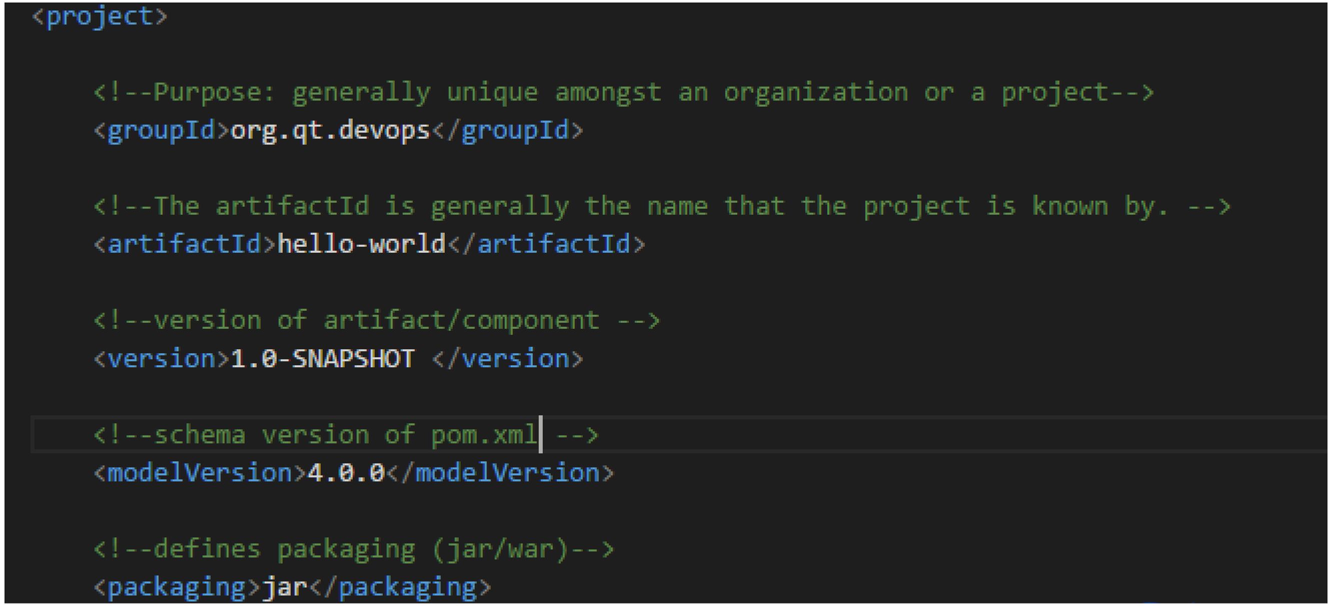
- Create a new folder (Directory) called hello-world
- Navigate with hello-world and create a file called pom.xml



```
lenovo@lenovo-PC MINGW64 /d/DevOps
$ cd MavenZone/

lenovo@lenovo-PC MINGW64 /d/DevOps/MavenZone (master)
$ mkdir hello-world

lenovo@lenovo-PC MINGW64 /d/DevOps/MavenZone (master)
$ cd hello-world

lenovo@lenovo-PC MINGW64 /d/DevOps/MavenZone/hello-world (master)
$ touch pom.xml
```

Hello World Maven project (contd..)

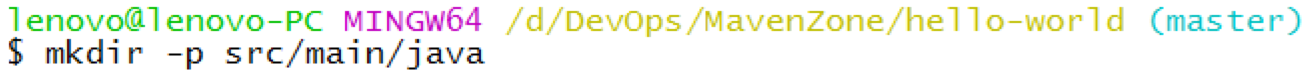
- Edit pom.xml with following contents



```
<project>

    <!--Purpose: generally unique amongst an organization or a project-->
    <groupId>org.qt.devops</groupId>

    <!--The artifactId is generally the name that the project is known by. -->
    <artifactId>hello-world</artifactId>

    <!--version of artifact/component -->
    <version>1.0-SNAPSHOT </version>

    <!--schema version of pom.xml -->
    <modelVersion>4.0.0</modelVersion>

    <!--defines packaging (jar/war)-->
    <packaging>jar</packaging>
```

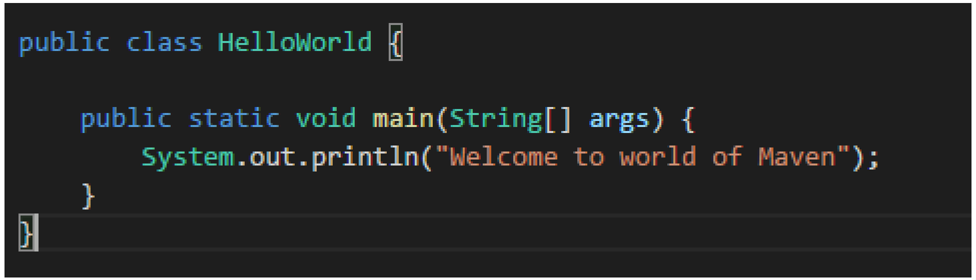
Hello World Maven Project (contd..)

- Create folder src/main/java



```
lenovo@lenovo-PC MINGW64 /d/DevOps/MavenZone/hello-world (master)
$ mkdir -p src/main/java
```

- Create a file HelloWorld.java with following contents



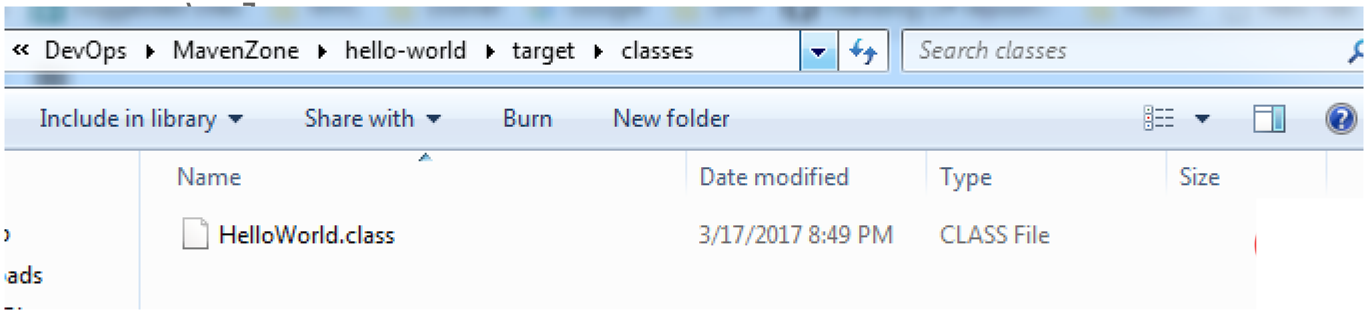
```java
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Welcome to world of Maven");
    }
}
```

- Navigate to the folder containing pom.xml

Hello World Maven Project (contd..)

- Execute **mvn clean** in command line/Terminal. If you are running for first time maven downloads bunch of plugins & it take some time

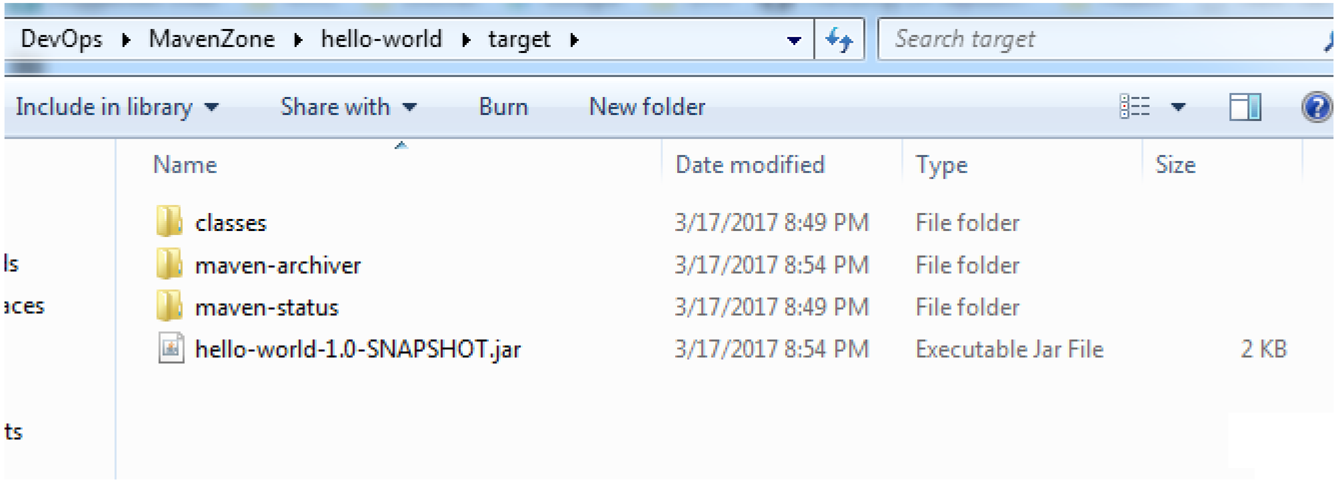
- Execute **mvn compile** in terminal. This will compile the java code and show you compile errors if any. If the compile is success, it will copy the class files to



Name	Date modified	Type	Size
HelloWorld.class	3/17/2017 8:49 PM	CLASS File	

Hello World Maven Project (contd..)

- Execute **mvn package** to perform packaging & create jar file in target folder . The jar files name will be **<artifactid>-<version>.<packaging type>**



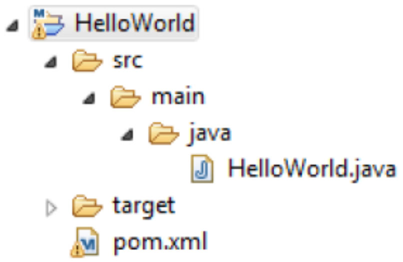
```
Name                          Date modified        Type                  Size
classes                       3/17/2017 8:49 PM    File folder
maven-archiver                3/17/2017 8:54 PM    File folder
maven-status                  3/17/2017 8:49 PM    File folder
hello-world-1.0-SNAPSHOT.jar  3/17/2017 8:54 PM    Executable Jar File   2 KB
```

Outline

- Folder Structure
- POM File Basics
- Basic Commands and Goals
- Dependencies
- Local Repo

src/main/what?

- src/main/java
- target
- pom.xml



```
HelloWorld
  src
    main
      java
        HelloWorld.java
  target
  pom.xml
```

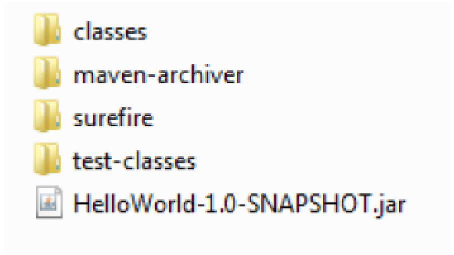
FOLDER STRUCTURE

src/main/java

- **Where we store our Java code**
    - The beginning of our package declaration
        - com.yourcompanyname.division
- **What about other languages**
    - src/main/groovy
- **What about testing**
    - src/test/java

target

- Where everything gets compiled to
- Also where tests get ran from
- Contents in this directory get packaged into a jar, war, ear, etc…


classes
maven-archiver
surefire
test-classes
HelloWorld-1.0-SNAPSHOT.jar

FOLDER STRUCTURE

pom.xml

```xml
<project>

    <!--Purpose: generally unique amongst an organization or a project-->
    <groupId>org.qt.devops</groupId>

    <!--The artifactId is generally the name that the project is known by. -->
    <artifactId>hello-world</artifactId>

    <!--version of artifact/component -->
    <version>1.0-SNAPSHOT </version>

    <!--schema version of pom.xml -->
    <modelVersion>4.0.0</modelVersion>

    <!--defines packaging (jar/war)-->
    <packaging>jar</packaging>

</project>
```

Folder Structure

pom.xml

- **Can be divided into 4 basic parts:**
  - Project Information
    - groupId
    - artifactId
    - version
    - packaging
  - Dependencies
    - Direct dependencies used in our application
  - Build
    - Plugins
    - Directory Structure
  - Repositories
    - Where we download the artifacts from

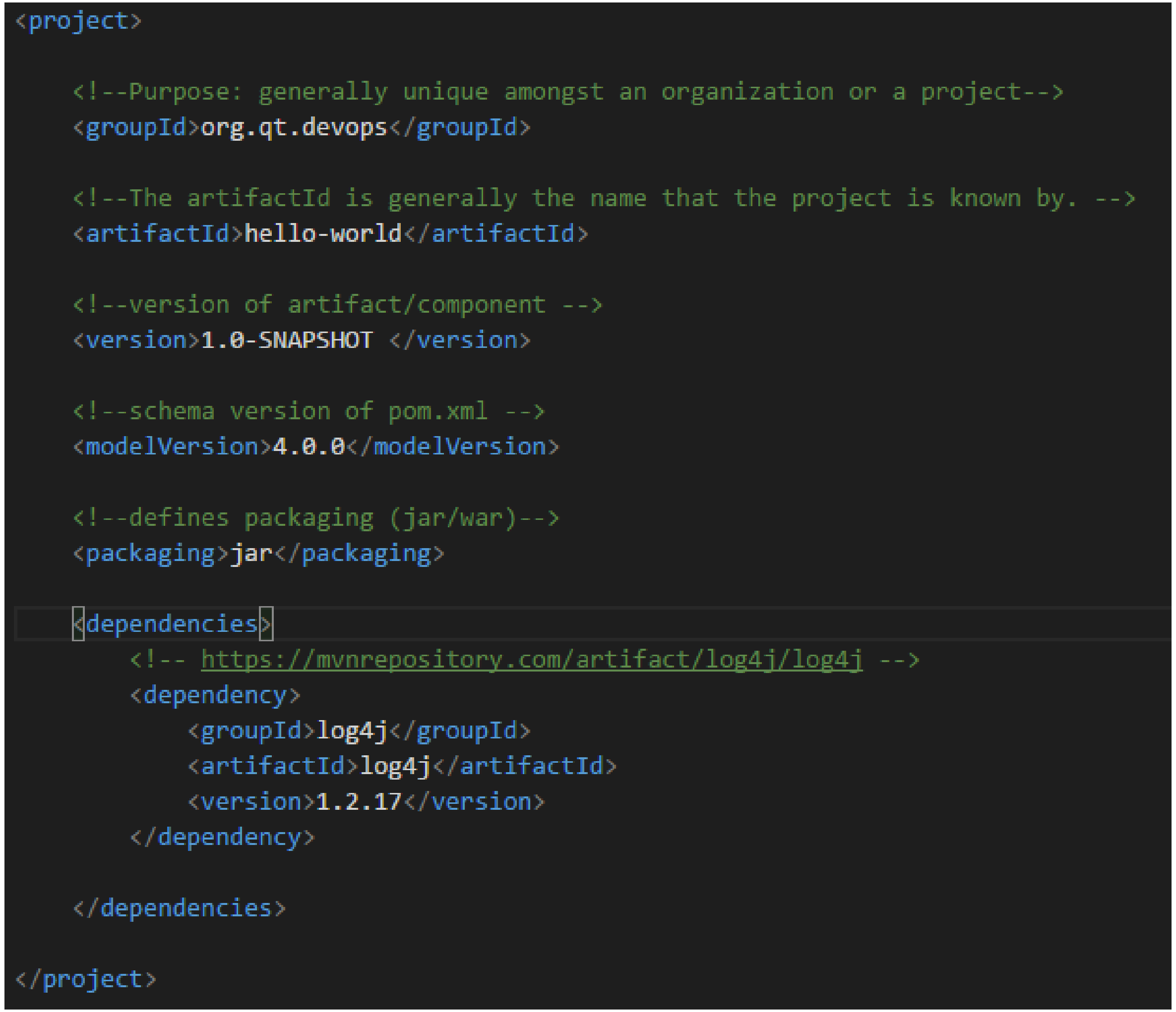
Dependencies

- What we want to use in our application
- Dependencies are imported by their naming convention
  - Often considered the most confusing part of Maven
- We have to know the groupId, artifactId, and the version of what we are looking for
- Added to a dependencies section to our pom file

Dependencies

- **Just list the dependency that we want**
  - Transitive dependencies will be pulled in by Maven
- **Need at a minimum 3 things:**
  - groupId
  - artifactId
  - version

```
<dependencies>
    <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
        <version>2.1</version>
    </dependency>
</dependencies>
```



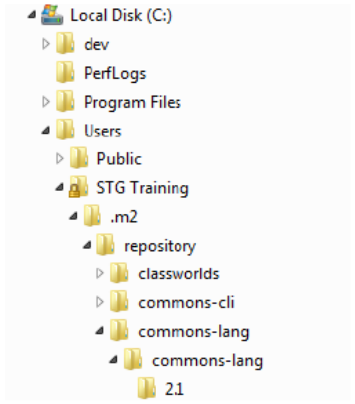
```xml
<project>

    <!--Purpose: generally unique amongst an organization or a project-->
    <groupId>org.qt.devops</groupId>

    <!--The artifactId is generally the name that the project is known by. -->
    <artifactId>hello-world</artifactId>

    <!--version of artifact/component -->
    <version>1.0-SNAPSHOT </version>

    <!--schema version of pom.xml -->
    <modelVersion>4.0.0</modelVersion>

    <!--defines packaging (jar/war)-->
    <packaging>jar</packaging>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/log4j/log4j -->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>

    </dependencies>

</project>
```

Goals

- **clean**
  - Deletes the target directory and any generated resources
- **compile**
  - Compiles all source code, generates any files, copies resources to our classes directory
- **package**
  - Runs the compile command first, runs any tests, packages the app based off of its packaging type
- **install**
  - Runs the package command and then installs it in your local repo
- **deploy**
  - Runs the install command and then deploys it to a corporate repo
  - Often confused with deploying to a web server

Local Repo

- **Where Maven stores everything it downloads**
  - Installs in your home directory\.m2
    - C:\Users\<yourusername>\.m2\repository
- **Stores artifacts using the information that you provided for artifactId, groupId, and version**
  - C:\Users\<yourusername>\.m2\repository\commons-lang\commons-lang\2.1\commons-lang-2.1.jar
- **Avoids duplication by copying it in every project and storing it in your SCM**


Local Disk (C:)
dev
PerfLogs
Program Files
Users
Public
STG Training
.m2
repository
classworlds
commons-cli
commons-lang
commons-lang
2.1