Structure

Bryan Hansen

twitter: bh5k

http://www.linkedin.com/in/hansenbryan

pluralsight

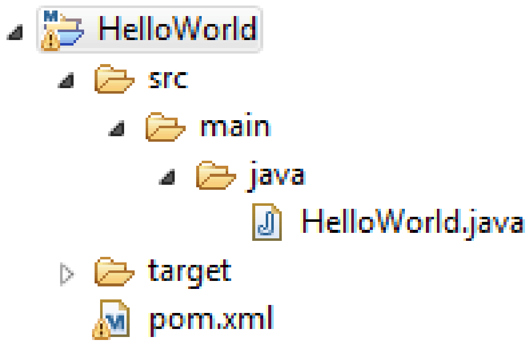
hardcore developer training

Outline

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

src/main/what?

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

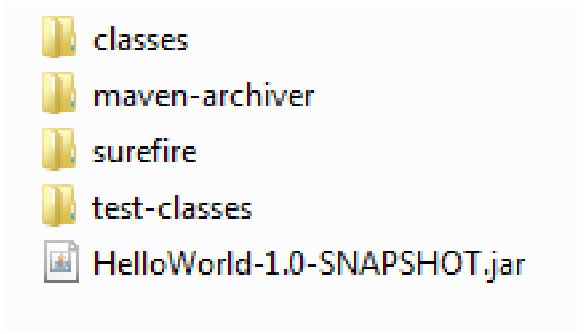


src/main/java

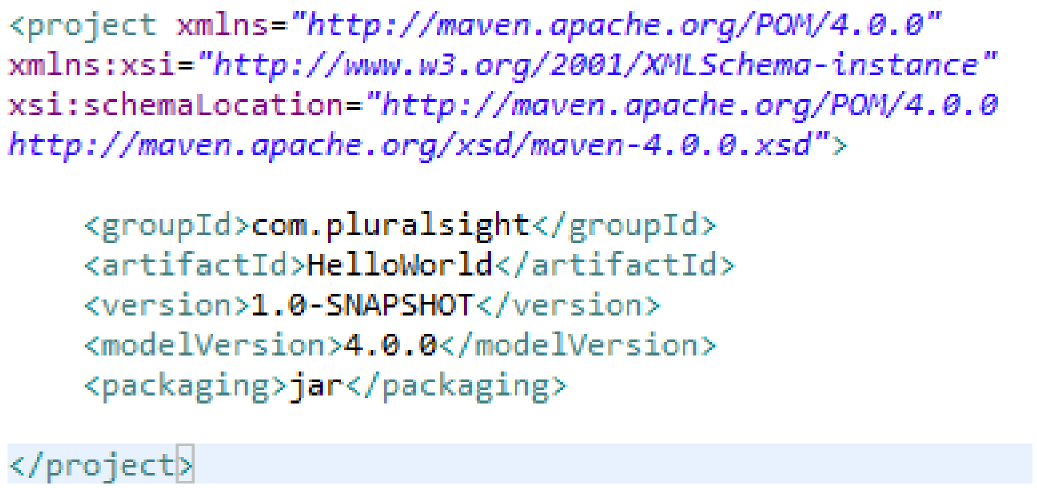
- **Where we store our Java code**
  - The beginning of our package declaration
    - com.yourcompanyname.division
- **What about other languages**
  - src/main/groovy
- **What about testing**
  - src/test/java

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

pom.xml



```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <groupId>com.pluralsight</groupId>
    <artifactId>HelloWorld</artifactId>
    <version>1.0-SNAPSHOT</version>
    <modelVersion>4.0.0</modelVersion>
    <packaging>jar</packaging>

</project>
```

pom.xml

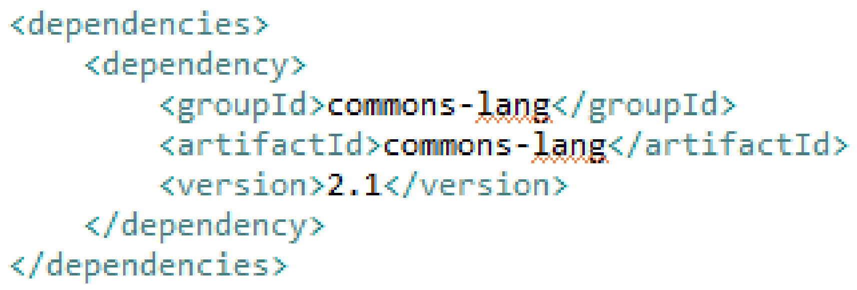
- **Can be divided into 4 basic parts:**
  - Project Information
    - groupId
    - artifactId
    - version
    - packaging
  - Dependencies
    - Direct dependencies used in our application
  - Build
    - Plugins
    - Directory Structure
  - Repositories
    - Where we download the artifacts from

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**

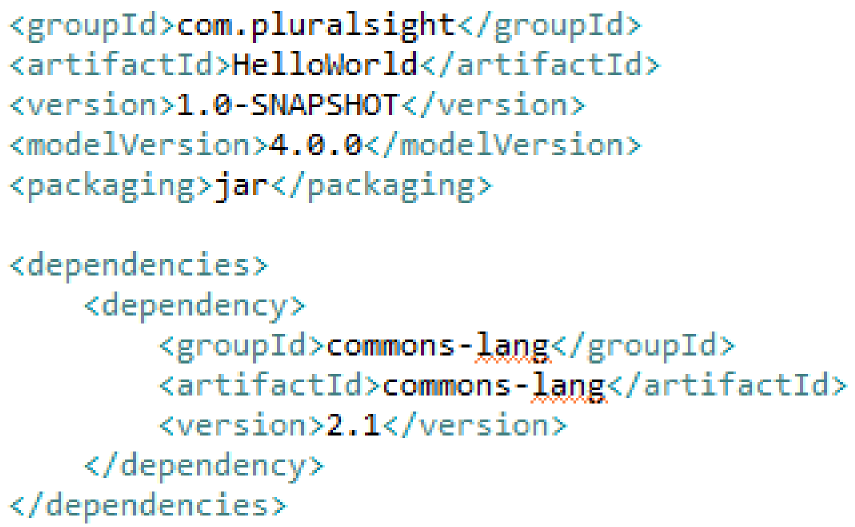
Dependencies

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



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

pom.xml with our new dependency



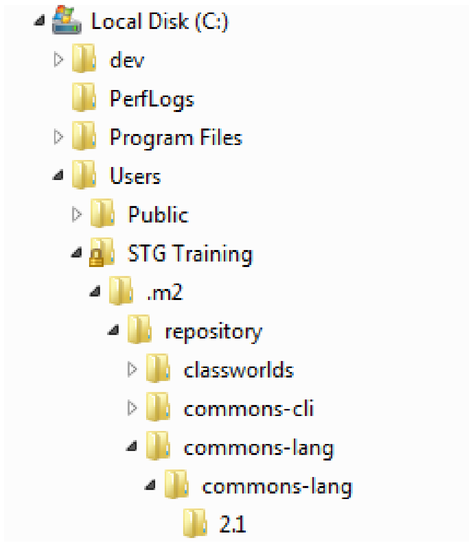
```xml
<groupId>com.pluralsight</groupId>
<artifactId>HelloWorld</artifactId>
<version>1.0-SNAPSHOT</version>
<modelVersion>4.0.0</modelVersion>
<packaging>jar</packaging>

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

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
```

Defaults

- **These are all the defaults that Maven has, but how do we override them?**
  - The build section!
  - Let’s Demo the options for structure changes.

Summary

- **Source code goes in src/main/java**
- **Everything is compiled to our target directory**
- **The pom has 4 major parts**
- **Introduction of goals**
  - You can chain goals
- **Basic example of a dependency**
- **Where things are stored in your local repo**
- **How we can override the default behavior**