What is Maven

Table of Contents

1 Introduction to Maven

Introduction to Maven

- First, it was used at **Apache’s Jakarta Alexandria Project** in 2001
- What Maven did was to **simplify the build processes**

Introduction to Maven

- As a project management tool, Maven :
  - builds **multiple projects** easily,
  - **publishes documentation** for the projects,
  - accomplishes an **easy deployment**,
  - **helps in collaboration** with development teams.

Introduction to Maven

- Maven can :
  - **manage the versions** of consecutive builds,
  - **compile** source code into binary,
  - **download dependencies**,
  - **run tests**,
  - **package** compiled code
  - **deploy** artifacts

2

Features of Maven

Features of Maven

- **Easy to start** with Maven
- Variety of **options**
- **Same structure** across different projects
- **Easy to integrate** into a developing team
- It has a **powerful dependency management tool**
- **Large repository** of libraries

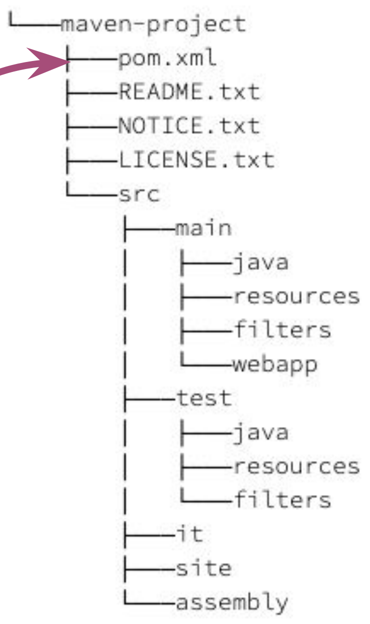
Features of Maven

- **Extra features** with plugins
- **Different outputs** like a **jar**, **ear** or **war**
- Maven can **generate a website**
- Maven can **support the older versions**

3 Directory Structure

Directory Structure

- Project structure **should conform** to
- The most important file is the **pom file**
  - defines project's **config details**



```
└──maven-project
    ├──pom.xml
    ├──README.txt
    ├──NOTICE.txt
    ├──LICENSE.txt
    └──src
        ├──main
        │   ├──java
        │   ├──resources
        │   ├──filters
        │   └──webapp
        ├──test
        │   ├──java
        │   ├──resources
        │   └──filters
        ├──it
        ├──site
        └──assembly
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

POM File

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

1 Introduction to POM File

Introduction to POM File

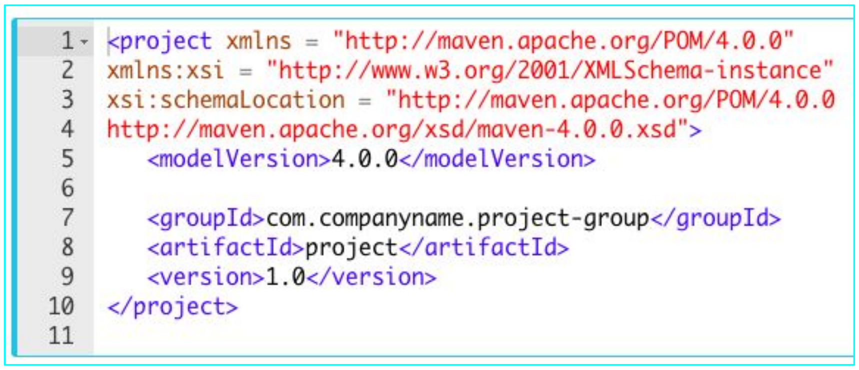
- It is an XML file
- **P**roject **O**bject **M**odel is the **starting point** for a Maven project
- It contains **configurations** about the project
- When a task or goal is executed, **Maven searches for** the **POM file**

Introduction to POM File

- POM defines
  - Project **dependencies**
  - Plugins and **goals** to be executed
  - **Build profiles**
  - Other information like the **project version**, **description**, **developers**, **mailing lists**, and more...

Introduction to POM File

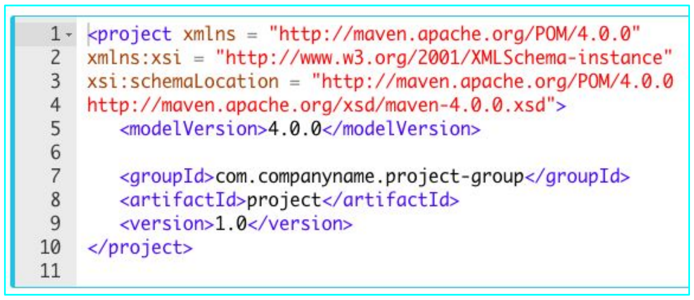
- There **must** be a POM file in every Maven project
- **All POMs need** at least
  - Project tag
  - modelVersion tag
  - **g**roupId tag
  - **a**rtifactId tag
  - **v**ersion (Last three called as **gav** in short)



```
<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">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.companyname.project-group</groupId>
    <artifactId>project</artifactId>
    <version>1.0</version>
</project>

```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Introduction to POM File



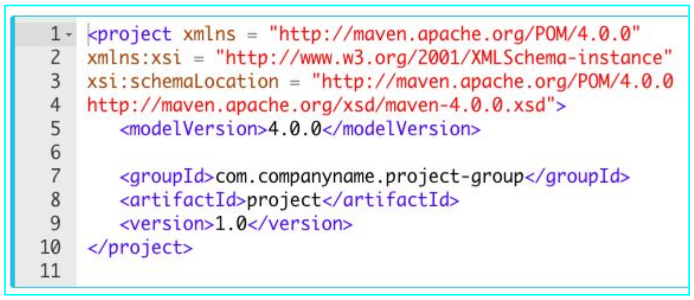
```
<project xmlns = "http://maven.apache.org/POM/4.0.0"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.companyname.project-group</groupId>
    <artifactId>project</artifactId>
    <version>1.0</version>
</project>

```

- **Project tag** is the **root** of the file
- It should reference a basic schema settings such as **apache schema** and **w3.org** specification
- **Model version** describes the **version of Maven**
- **Group Id** is the id of the project's group (Simply it shows the **company** or the **organization** or the **owner of the project**)

CLARUSWAY©
WAY TO REINVENT YOURSELF

Introduction to POM File

- **Group Id** should be long enough to give **uniqueness** to the project
- **Artifact id** is the id for specifying the project under the group



```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">
   <modelVersion>4.0.0</modelVersion>

   <groupId>com.companyname.project-group</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
</project>
```

- It shows the **name of the project** like pet-clinic-server
- **Version** defines the version number of the project

CLARUSWAY©
WAY TO REINVENT YOURSELF

2 Super POM

CLARUSWAY©
WAY TO REINVENT YOURSELF

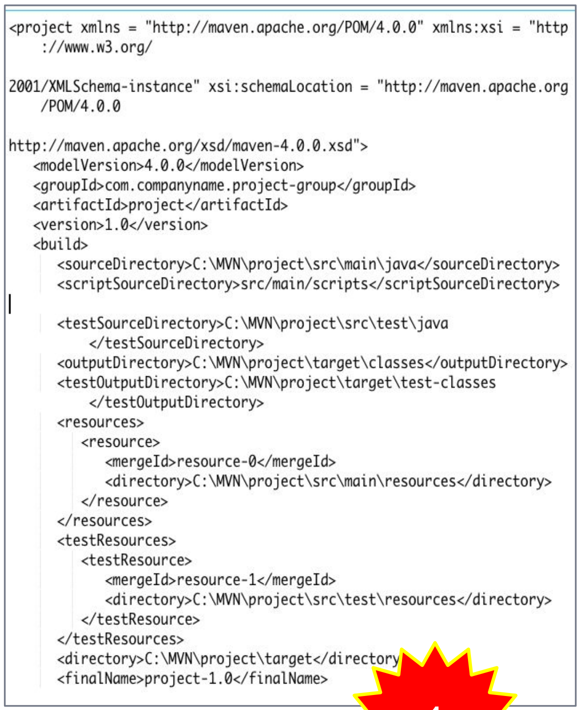
Super POM

- Super POM is Maven's **default POM**
- **All POMs extend** the Super POM **unless explicitly set**
- Super POM and project POM creates the **Effective POM**
- Which is the **overall configuration** file
- Effective POM can be examined by running

  "**mvn help:effective-pom**"

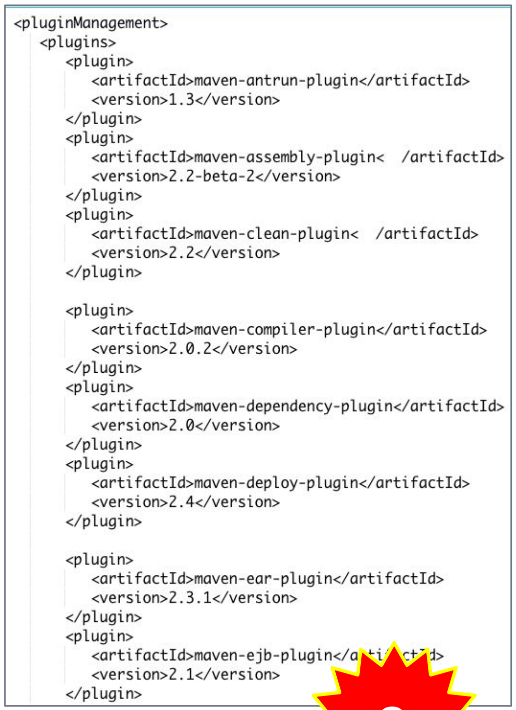
CLARUSWAY©
WAY TO REINVENT YOURSELF

- Effective POM -



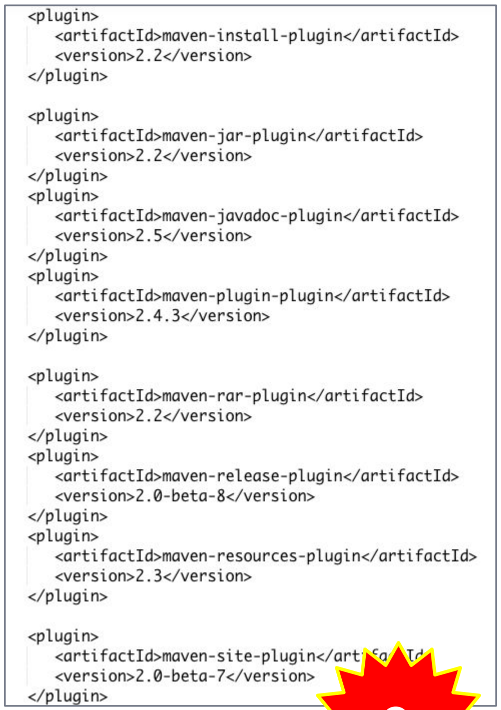
```
<project xmlns = "http://maven.apache.org/POM/4.0.0" xmlns:xsi = "http
    ://www.w3.org/

2001/XMLSchema-instance" xsi:schemaLocation = "http://maven.apache.org
    /POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.companyname.project-group</groupId>
   <artifactId>project</artifactId>
   <version>1.0</version>
   <build>
      <sourceDirectory>C:\MVN\project\src\main\java</sourceDirectory>
      <scriptSourceDirectory>src/main/scripts</scriptSourceDirectory>

      <testSourceDirectory>C:\MVN\project\src\test\java
          </testSourceDirectory>
      <outputDirectory>C:\MVN\project\target\classes</outputDirectory>
      <testOutputDirectory>C:\MVN\project\target\test-classes
          </testOutputDirectory>
      <resources>
         <resource>
            <mergeId>resource-0</mergeId>
            <directory>C:\MVN\project\src\main\resources</directory>
         </resource>
      </resources>
      <testResources>
         <testResource>
            <mergeId>resource-1</mergeId>
            <directory>C:\MVN\project\src\test\resources</directory>
         </testResource>
      </testResources>
      <directory>C:\MVN\project\target</directory
      <finalName>project-1.0</finalName>
```

1



```
<pluginManagement>
   <plugins>
      <plugin>
         <artifactId>maven-antrun-plugin</artifactId>
         <version>1.3</version>
      </plugin>
      <plugin>
         <artifactId>maven-assembly-plugin<  /artifactId>
         <version>2.2-beta-2</version>
      </plugin>
      <plugin>
         <artifactId>maven-clean-plugin<  /artifactId>
         <version>2.2</version>
      </plugin>

      <plugin>
         <artifactId>maven-compiler-plugin</artifactId>
         <version>2.0.2</version>
      </plugin>
      <plugin>
         <artifactId>maven-dependency-plugin</artifactId>
         <version>2.0</version>
      </plugin>
      <plugin>
         <artifactId>maven-deploy-plugin</artifactId>
         <version>2.4</version>
      </plugin>

      <plugin>
         <artifactId>maven-ear-plugin</artifactId>
         <version>2.3.1</version>
      </plugin>
      <plugin>
         <artifactId>maven-ejb-plugin</a[illegible]d>
         <version>2.1</version>
      </plugin>
```

2

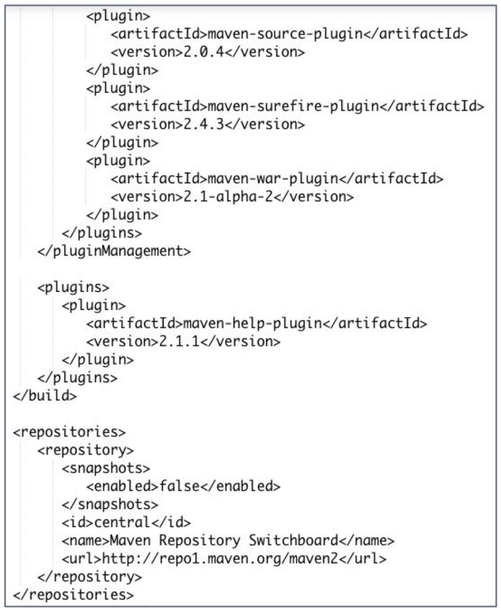


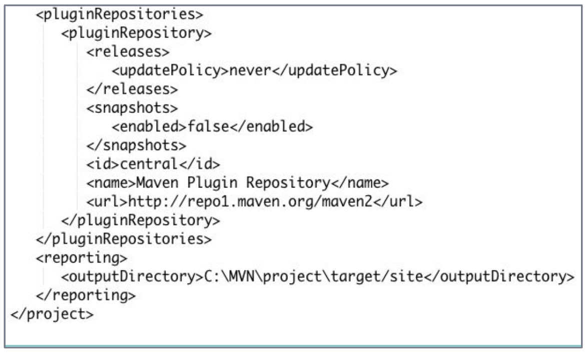
```
<plugin>
   <artifactId>maven-install-plugin</artifactId>
   <version>2.2</version>
</plugin>

<plugin>
   <artifactId>maven-jar-plugin</artifactId>
   <version>2.2</version>
</plugin>
<plugin>
   <artifactId>maven-javadoc-plugin</artifactId>
   <version>2.5</version>
</plugin>
<plugin>
   <artifactId>maven-plugin-plugin</artifactId>
   <version>2.4.3</version>
</plugin>

<plugin>
   <artifactId>maven-rar-plugin</artifactId>
   <version>2.2</version>
</plugin>
<plugin>
   <artifactId>maven-release-plugin</artifactId>
   <version>2.0-beta-8</version>
</plugin>
<plugin>
   <artifactId>maven-resources-plugin</artifactId>
   <version>2.3</version>
</plugin>

<plugin>
   <artifactId>maven-site-plugin</art[illegible]Id>
   <version>2.0-beta-7</version>
</plugin>
```

3

CLARUSWAY©
WAY TO REINVENT YOURSELF

- Effective POM -


```
            <plugin>
               <artifactId>maven-source-plugin</artifactId>
               <version>2.0.4</version>
            </plugin>
            <plugin>
               <artifactId>maven-surefire-plugin</artifactId>
               <version>2.4.3</version>
            </plugin>
            <plugin>
               <artifactId>maven-war-plugin</artifactId>
               <version>2.1-alpha-2</version>
            </plugin>
         </plugins>
      </pluginManagement>

      <plugins>
         <plugin>
            <artifactId>maven-help-plugin</artifactId>
            <version>2.1.1</version>
         </plugin>
      </plugins>
   </build>

   <repositories>
      <repository>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
         <id>central</id>
         <name>Maven Repository Switchboard</name>
         <url>http://repo1.maven.org/maven2</url>
      </repository>
   </repositories>
```


```
   <pluginRepositories>
      <pluginRepository>
         <releases>
            <updatePolicy>never</updatePolicy>
         </releases>
         <snapshots>
            <enabled>false</enabled>
         </snapshots>
         <id>central</id>
         <name>Maven Plugin Repository</name>
         <url>http://repo1.maven.org/maven2</url>
      </pluginRepository>
   </pluginRepositories>
   <reporting>
      <outputDirectory>C:\MVN\project\target/site</outputDirectory>
   </reporting>
</project>
```

4

5

CLARUSWAY©
WAY TO REINVENT YOURSELF

3 Project Inheritance

CLARUSWAY©
WAY TO REINVENT YOURSELF

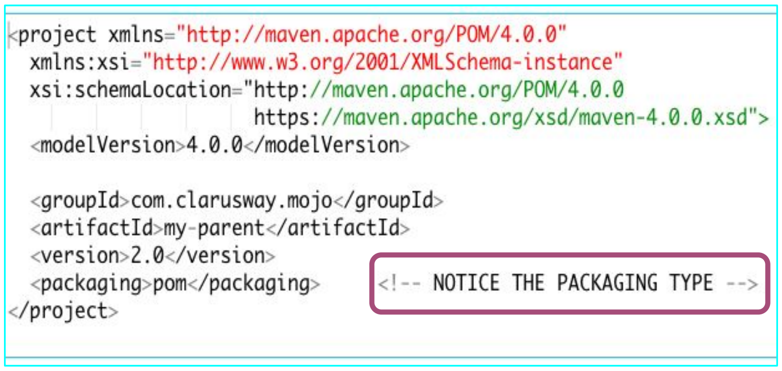
Project Inheritance

- As in the object-oriented programming, POM files **can also be inherited** by other POM files
- Child POM can either **inherit** or **override**
- Parent POM is a **general template**
- **Not every item** in the parent is inherited
- Some elements should be declared specifically
- Like **artifactId**, **name**, and **prerequisites**

Project Inheritance

- Parent POM's **packaging** tag should have the value "**pom**"

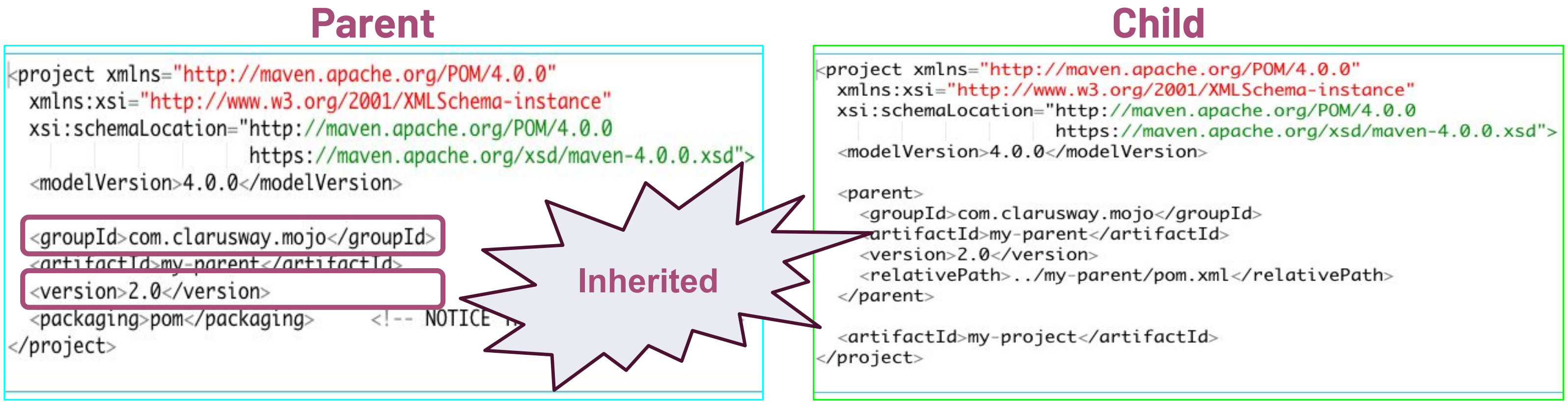
Parent



```
<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
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.clarusway.mojo</groupId>
  <artifactId>my-parent</artifactId>
  <version>2.0</version>
  <packaging>pom</packaging>      <!-- NOTICE THE PACKAGING TYPE -->
</project>
```

Project Inheritance

- Child is related to parent **by specifying the parent element**
- If you want to inherit an element you should remove it



Parent

```
<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
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.clarusway.mojo</groupId>
  <artifactId>my-parent</artifactId>
  <version>2.0</version>
  <packaging>pom</packaging>      <!-- NOTICE T
</project>
```

Child

```
<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
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>com.clarusway.mojo</groupId>
    <artifactId>my-parent</artifactId>
    <version>2.0</version>
    <relativePath>../my-parent/pom.xml</relativePath>
  </parent>

  <artifactId>my-project</artifactId>
</project>
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

4

Project Aggregation

CLARUSWAY©
WAY TO REINVENT YOURSELF

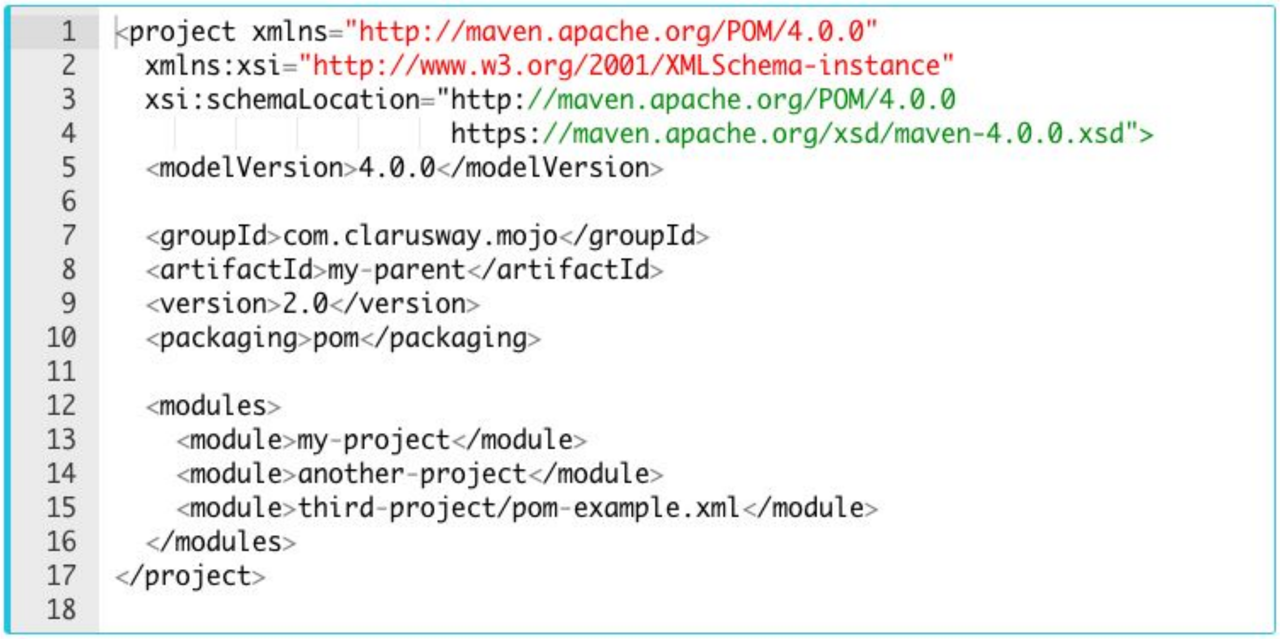
Project Aggregation

- A project **with modules** (children) is called a **multi-module**, or aggregator project
- Modules are projects that a **parent POM file specifies**
- These modules are **built together as a group**
- Aggregator POM should have
  - **packaging tag** with "**pom**"
  - **modules tag** with relative paths to the directories or the POM files of modules

CLARUSWAY©
WAY TO REINVENT YOURSELF

Project Aggregation

- **As in the example :**



```
<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
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.clarusway.mojo</groupId>
  <artifactId>my-parent</artifactId>
  <version>2.0</version>
  <packaging>pom</packaging>

  <modules>
    <module>my-project</module>
    <module>another-project</module>
    <module>third-project/pom-example.xml</module>
  </modules>
</project>

```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Build Lifecycles

Table of Contents

1 Introduction to Build Lifecycles

Introduction to Build Lifecycles

- A Build Lifecycle is **a track** that is comprised of **different number of phases**
- A phase is a **job unit** or a **specific stage** in a lifecycle

Introduction to Build Lifecycles

- There are **three built-in lifecycles :**
  - **default**, **clean**, and **site**
  - Default is the **main lifecycle**
  - Clean is used for **cleaning the project**
  - Site lifecycle is used for building the **project's website**

Introduction to Build Lifecycles

- Each life cycle has a different number of phases
  - Default build lifecycle has **23**
  - Clean lifecycle has **3**
  - Site lifecycle has **4 phases**

Introduction to Build Lifecycles

- **Using Command-Line :**
  - Maven CLI commands generates your outputs
  - For example,
    - “**mvn package**” gives you a “**jar, war or ear …**”
    - “**mvn test**” gives your test code’s results
    - “**mvn clean**” cleans the artifacts of a previous command

2 Clean Lifecycle

Clean Lifecycle

- Clean Lifecycle has **three phases**
  - **pre-clean**, **clean**, and **post-clean**
- These phases are **in sequence**
- When a phase is called (for example "mvn post-clean"), **phases prior to that phase** are also **run**

Clean Lifecycle

- It cleans the project's target directory
- **Pre-clean** phase is used for **any task prior to** the cleanup
- **Post-clean** phase is used for **any task following** the cleanup

3 Default Lifecycle

Default Lifecycle

- Default lifecycle is used **for application build**
- There are **23 phases** in Default Lifecycle
- The most important phases are :
  - **validate:** validates if the project has necessary information
  - **compile:** compiles the source code
  - **test-compile:** compiles the test source code

Default Lifecycle

- The most important phases are :
  - **test:** runs unit tests
  - **package:** packages compiled source code
    - **packaging tag** in POM.xml changes the output

Default Lifecycle

- The most important phases are :
  - **integration-test:** processes and deploys the package if needed to run integration test
  - **install:** installs the package to local repository
  - **deploy:** copies the package to a remote repository

4 Site Lifecycle

Site Lifecycle

- Site lifecycle has **four phases**
  - **pre-site**, **site**, **post-site**, **site-deploy**
- For Site Lifecycle, the **Site Plugin is used**
- The plugin's **main duty** is to **generate a website**

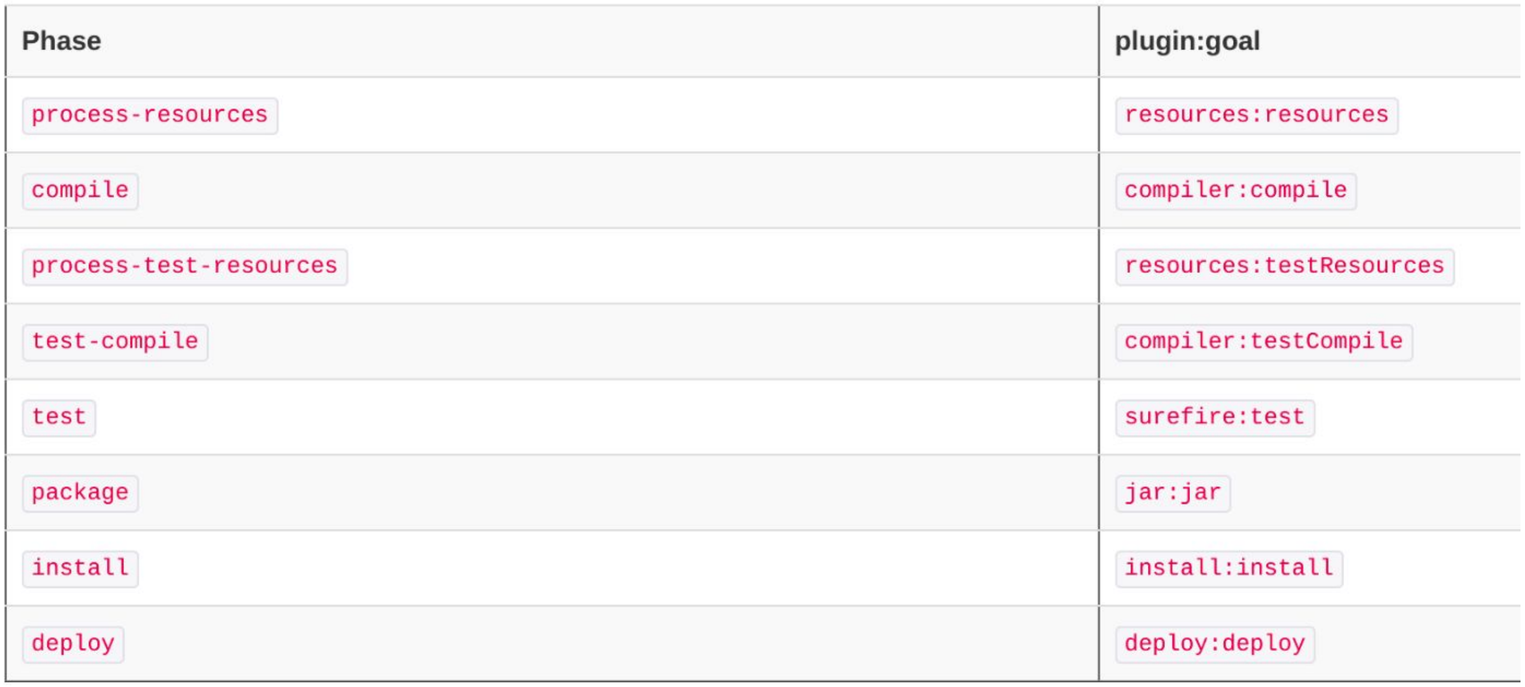
THANKS!

Any questions?

phase-goal-plugin

Maven Build Lifecycle	Phases	Goals	Plugin
	pre-clean		
clean	clean	clean	Clean
	post-clean		

phase-goal-plugin



Phase	plugin:goal
`process-resources`	`resources:resources`
`compile`	`compiler:compile`
`process-test-resources`	`resources:testResources`
`test-compile`	`compiler:testCompile`
`test`	`surefire:test`
`package`	`jar:jar`
`install`	`install:install`
`deploy`	`deploy:deploy`