Maven

What is Build?

- Build : Compile + Assembly + Create deliverable
- Compile : Convert Source code to machine readable format
- Assembly (Linking) : Grouping all class files
- Deliverable : .war, .jar

Advantages of Build tool

- Automated tasks (Mention all in pom.xml)
- Multiple Tasks at a time
- Quality product
- Minimize bad builds
- Keep history
- Save time - Save money
- Documentation
- Gives set of standards
- Gives define project life cycle (Goals)
- Manage all dependencies
- Uniformity in all projects
- Re-usability

Why separate build team?

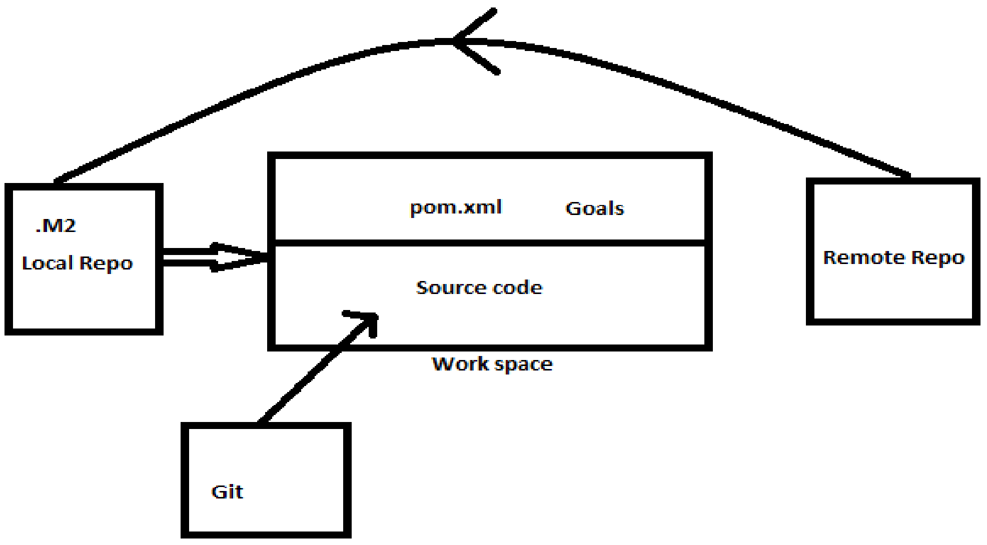
- To match customer's environment
- Build team worry about whole product

SAIDEMY

Build tools

- C, C++ : Make file
- .Net : Visual studio
- Java : Ant, Maven

Architecture of Maven



Architecture of Maven

- Main configuration file is pom.xml
- One project - One workspace - One pom.xml

Sr MY

- Requirements for build:
    1. Source code(Present in workspace)
    2. Compiler(Remote repo - local repo - Workspace)
    3. Dependencies(Remote repo - local repo - Workspace)

Maven Build Life-Cycle

Goals:

1. Generate resources (Dependencies)
2. Compile code
3. Unit test
4. Package (Build)
5. Install (in to local repo & artifactory)
6. Deploy (to servers)
7. Clean (delete all run time files)

eg: mvn install

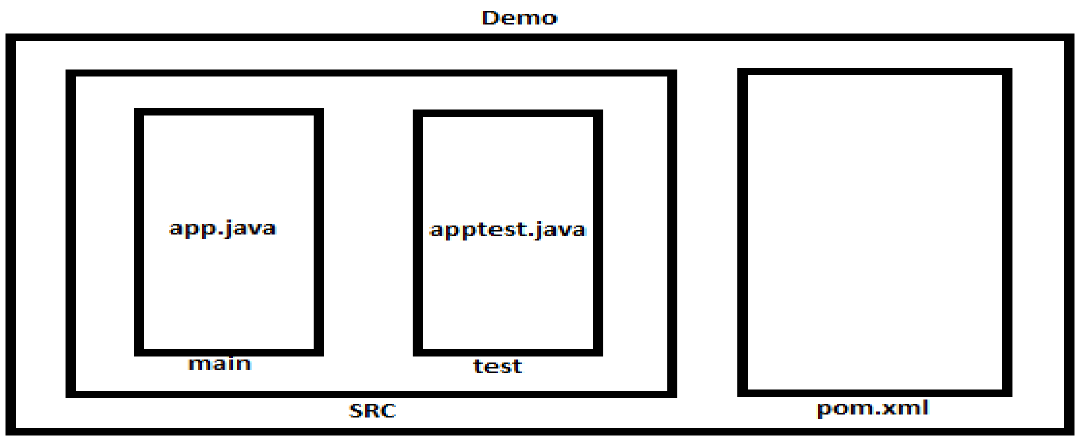
mvn clean package

1-6 -> Default & Sequence order

7 -> Not Default & It won't follow sequence

SAIDEMY

Maven Directory Structure

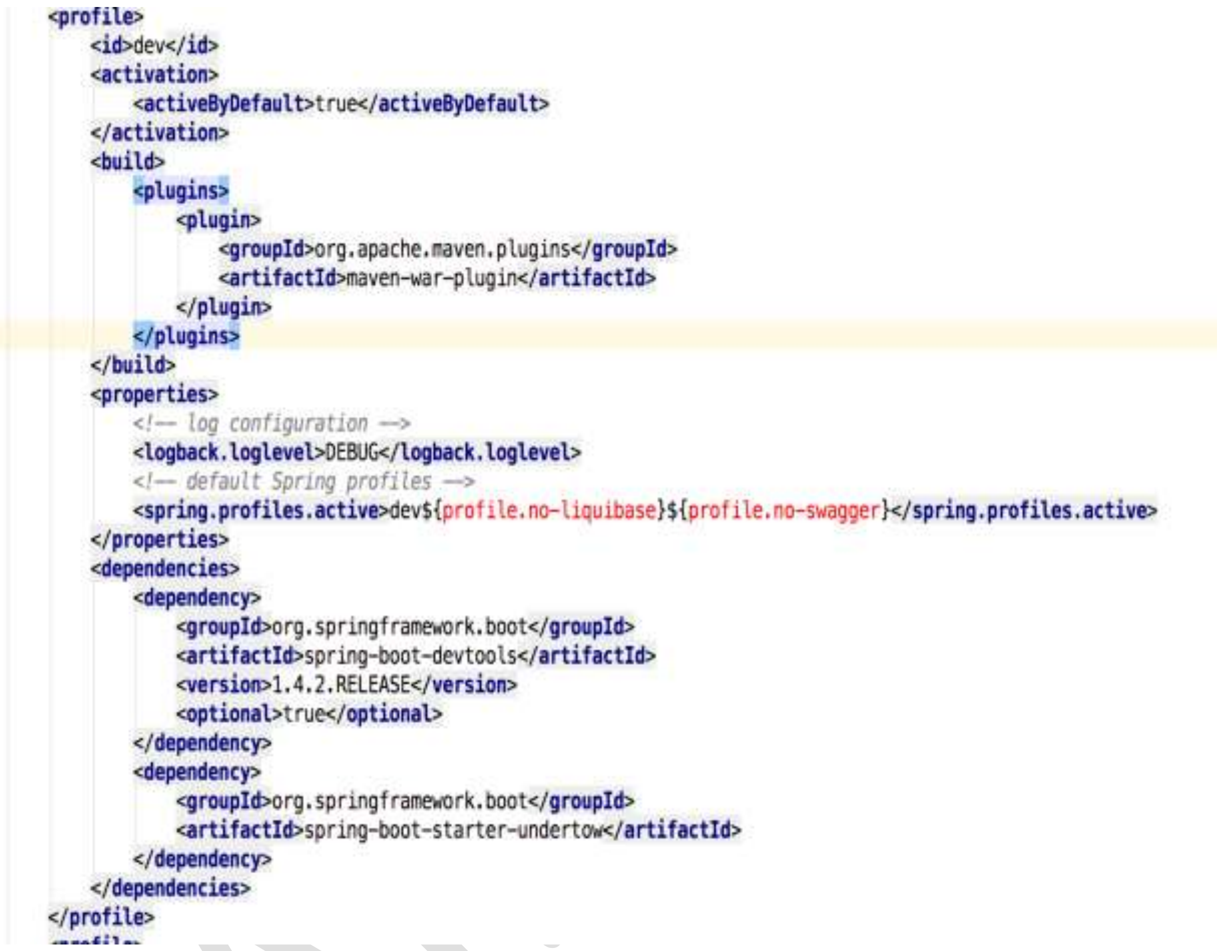


Maven Repositories

- Local (.M2)
- Remote (https://repo1.maven.org/maven2/)

SAIDEN

Pom.xml contains



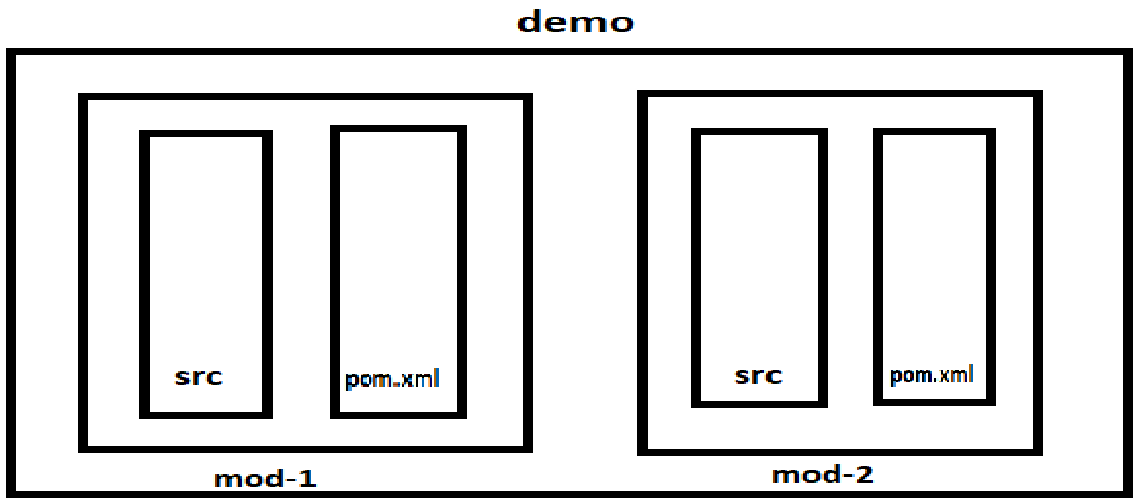
```
<profile>
    <id>dev</id>
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
    <properties>
        <!-- log configuration -->
        <logback.loglevel>DEBUG</logback.loglevel>
        <!-- default Spring profiles -->
        <spring.profiles.active>dev${profile.no-liquibase}${profile.no-swagger}</spring.profiles.active>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <version>1.4.2.RELEASE</version>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-undertow</artifactId>
        </dependency>
    </dependencies>
</profile>
```

- Metadata
- Dependencies
- Kind of project
- Kind of output (.jar, .war)
- Description

Important points

- Maven is all about plug-ins
- Snapshot: Indicates development copy of your project. Not the one which you are going to release.
- eg: 1,0-SNAPSHOT
- If you see version no in place of snapshot, then it means product is ready to give customer.

Multi-module project



Multi-module project

- Simply dividing project into modules
- Each module must have it's own SRC folder & pom.xml so that build will happen separately
- To build all modules with one command, there should be a parent pom.xml file. This calls all child pom.xml files automatically
- In parent pom.xml file, need to mention the child pom.xml order.