Сборка и непрерывная интеграция

(Часть 2. Maven)

Владимир Поляков
vladimir.p.polyakov@gmail.com

https://github.com/drxaos-edu/lecture-java-build-ci

Зачем?

- Декларативное описание сборки
- Версионирование
- Репозитории зависимостей

Maven

- Проект описывается POM (Project Object Model)
  - Может содержать подпроекты
  - Соответствует рекомендованной структуре каталогов (src/main/java, src/main/resources, src/test/java, …)
- Фокус на производимые артифакты (Artifact)
- Большой акцент на конфигурацию по-умолчанию

Maven

- Все build-системы делают одно и то же:
  - Компиляция исходного кода
  - Копирование ресурсов
  - Компиляция и запуск тестов
  - Упаковка дистрибутива
  - Публикация дистрибутива
  - Чистка проекта

Build Lifecycle

- Фазы жизненного цикла
  - Validate
  - Generate-sources
  - Compile
  - Test
  - Package
  - Integration-test
  - Verify
  - Install
  - Deploy

- https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

Maven

- Каждая фаза может содержать любое количество целей (“Goals”)
- Конфигурация проекта привязывает определенные цели к фазам (поверх конфигурации по-умолчанию)
- Система плагинов (Mojos), содержащих цели

Установка Maven

http://maven.apache.org

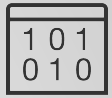
Maven POM

- Название и версия проекта
- Тип артифакта
- Положение исходного кода
- Зависимости
- Плагины
- Профили
  (Альтернативные конфигурации сборки)

Project Name (GAV)

- groupID: Группа проектов (обычно берется java package)
- artfiactId: Имя проекта
- version: Версия проекта (Формат {Major}.{Minor}.{Maintanence}) ‘-SNAPSHOT ‘ для проектов в разработке

Project Name (GAV)




```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example.mvn</groupId>
    <artifactId>maven-example</artifactId>
    <version>1.0</version>
</project>
```

Packaging

- Тип сборки
- Сообщает Maven как собирать проект
- Примеры: pom, jar, war, ear, custom
- По-умолчанию jar

Packaging

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example.mvn</groupId>
    <artifactId>maven-example</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
</project>
```

Наследование

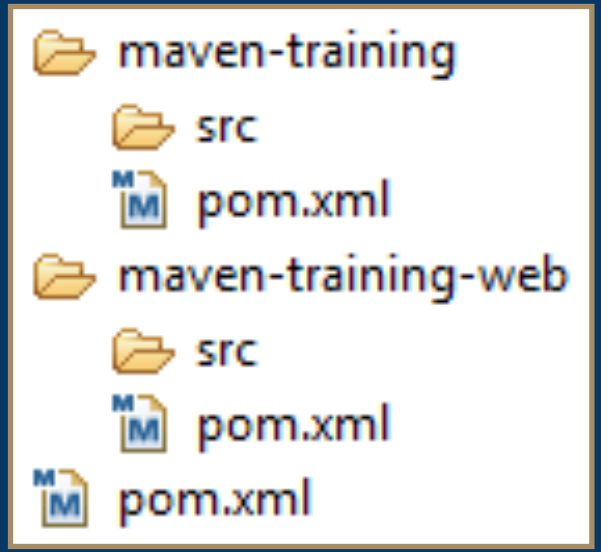
- groupId, version
- Project Config
- Dependencies
- Plugin configuration

Наследование

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
    <parent>
        <artifactId>maven-example-parent</artifactId>
        <groupId>org.example.mvn</groupId>
        <version>1.0</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>
    <artifactId>maven-example</artifactId>
    <packaging>jar</packaging>
</project>
```

Многомодульные проекты

```
<project>
    ...
    <packaging>pom</packaging>
    <modules>
        <module>maven-training</module>
        <module>maven-training-web</module>
    </modules>
</project>
```



Соглашения

- target: Рабочий каталог по-умолчанию
- src: Каталог со всеми исходными кодами
- src/main: Исходный код для первичного артифакта
- src/test: Исходный код для тестирования
- src/main/java: Исходники java
- src/main/webapp: Исходники для web
- src/main/resources: Не компилируемые исходники
- src/test/java: Исходники java тестов

Запуск Maven

```
mvn <GOAL>

mvn install
- generate*, compile, test, package, integration-test, install
mvn clean
- clean
mvn clean compile
- clean, generate*, compile
mvn compile install
- generate*, compile, test, package, integration-test, install
mvn test clean
- generate*, compile, test, clean
```

Зависимости (dependencies)

Maven совершил революцию в управлении зависимостями в Java

- Убрал необходимость копировать библиотеки в систему контроля версий кода
- Создал концепт репозитория зависимостей (и запустил Maven Central)
- Создал концепт транзитивных зависимостей (которые часто включают исходный код и javadoc)

Добавление зависимости

Зависимость состоит из

- GAV
- Scope: compile, test, provided (default=compile)
- Type: jar, pom, war, ear, zip (default=jar)

Добавление зависимости

```
<project>
    ...
    <dependencies>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

Репозитории

- Зависимости скачиваются из репозитория
  - по HTTP
- Кэшируются в локальном репозитории
  - ${user.home}/.m2/repository
- Структура репозитория
  - {groupId}/{artifactId}/{version}/{artifactId}-{version}.jar
  - где groupId ‘.’ заменяется на ‘/’
- Первичный репозиторий Maven Central
  - http://repo1.maven.org/maven2

Добавление репозитория

```
<project>
    <repositories>
        <repository>
            <id>lds-main</id>
            <name>LDS Main Repo</name>
            <url>http://code.lds.org/nexus/content/groups/main-repo</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>
</project>
```

Транзитивные зависимости

ProjectA -> ProjectB -> ProjectC

- Только зависимости в compile и runtime scope
- Можно управлять
  - Exclude
  - Optional

Исключение зависимостей

```
<project>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>3.0.5.RELEASE</version>
            <exclusions>
                <exclusion>
                    <groupId>commons-logging</groupId>
                    <artifactId>commons-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>
</project>
```

Плагины

Расширяют функциональность Maven

Для работы большинства плагинов обычно требуются дополнительные настройки, которые специфичны для конкретного плагина. Настройки задаются в тэгах <configuration>

Подключение плагина

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-checkstyle-plugin</artifactId>
    <version>2.6</version>
    <executions>
        <execution>
            <phase>package</phase>
            <goals>
                <goal>check</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

Подключение плагина

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>tomcat-maven-plugin</artifactId>
    <version>1.1</version>
    <configuration>
        <fork>false</fork>
        <server>test-server</server>
        <url>http://test-server/manager</url>
    </configuration>
</plugin>
```

Пример

http://mvnrepository.com/artifact/com.google.code.gson/gson/2.3.1

http://mojo.codehaus.org/buildnumber-maven-plugin/usage.html

Задача: Декомпрессия

http://mvnrepository.com/artifact/org.apache.commons/commons-compress/1.9

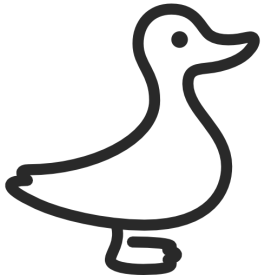
http://mvnrepository.com/artifact/commons-codec/commons-codec/1.10

https://github.com/drxaos-edu/lecture-java-build-ci/blob/master/example3/src/main/java/Encoder.java

https://github.com/drxaos-edu/lecture-java-build-ci/blob/master/example3/pom.xml

```
Enter String: *****
/Td6WFoAAATm1rRGAgAhARYAAAB0L+WjAQBEQXBhY2hlIE1hdmVuIGlzIGEgc29mdHdhcmUgcHJ
vamVjdCBtYW5hZ2VtZW50IGFuZCBjb21wcmVoZW5zaW9uIHRvb2wuAAAAAMXaU7TVffeMAAFdRS
0jJSkftvN9AQAAAAAEWVo=
```

Вопросы



vladimir.p.polyakov@gmail.com
https://github.com/drxaos-edu