Введение в систему сборки Maven

Камнев Георгий
2020

Системы сборки

Цели сборки

компиляция исходного кода (*.java → *.class)

сборка бинарного кода (*.class → *.jar/exe)

выполнение тестов

развёртывание программы (deploy) в целевой среде

написание сопроводительной документации

Предшественники (make)

Появление — 1976

Оперирует

- Цели
- Реквизиты
- Наборы команд
- Есть переменные
- Поддерживает
  - инкрементальные сборки
  - Параллельное исполнение

```
make [ -f make-файл ] [ цель ] ...
```

```
# Simple makefile for compiling a
program from
# two C source files.

.KEEP_STATE

functions: main.o data.o
        cc -O -o functions main.o
data.o
main.o: main.c
        cc -O -c main.c
data.o: data.c
        cc -O -c data.c
clean:
        rm functions main.o data.o
```

Предшественники - make

Появление — 1976

Оперирует

- Цели
- Реквизиты
- Наборы команд
- Есть переменные
- Поддерживает
  - инкрементальные сборки
  - Параллельное исполнение

```
make [ -f make-файл ] [ цель ] ...
```

```
# Simple makefile for compiling a program
from
# two C source files.

.KEEP_STATE

functions: main.o data.o
        cc -O -o functions main.o data.o
main.o: main.c
        cc -O -c main.c
data.o: data.c
        cc -O -c data.c
clean:
        rm functions main.o data.o
```

Предшественники - Apache Ant

Появление

19 July 2000

Возможности

Цели (target)

Зависимые цели (depends)

Команды

Плагины

```
<?xml version=''1.0''?>
<project name=''HelloWorld'' default=''run''>
    <target name=''compile''>
        <mkdir dir=''build/classes''/>
        <javac destdir=''build/classes''
includeantruntime=''false''>
            <src path=''src''/>
        </javac>
    </target>
    <target name=''run'' depends=''compile''>
        <java classname=''HelloWorld''
classpath=''build/classes''/>
    </target>
    <target name=''clean''>
        <delete dir=''build''/>
    </target>
</project>
```

Предшественники - Apache Ant

Появление

19 July 2000

Возможности

Цели (target)

Зависимые цели (depends)

Команды

Плагины

```
<?xml version="1.0"?>
<project name="HelloWorld" default="run">
    <target name="compile">
        <mkdir dir="build/classes"/>
        <javac destdir="build/classes"
includeantruntime="false">
            <src path="src"/>
        </javac>
    </target>
    <target name="run" depends="compile">
        <java classname="HelloWorld"
classpath="build/classes"/>
    </target>
    <target name="clean">
        <delete dir="build"/>
    </target>
</project>
```

Предшественники - Apache Ivy

Появление

2004 г.

Функции

менеджер зависимостей,

поддержкой транзитивных зависимостей

Отличие от maven

Apache Maven – это **не только менеджер зависимостей**, он также осуществляет управление всем проектом и его сборку,

тогда как **Apache Ivy** - **только** инструмент **управления зависимостями**

Maven - это

Стандартизация жизненного цикла

Менеджер зависимостей

Расширяемость

Независимость от ОС

Документированность

Средство UNIT тестирования

Артефакты

Установка maven

Нужно

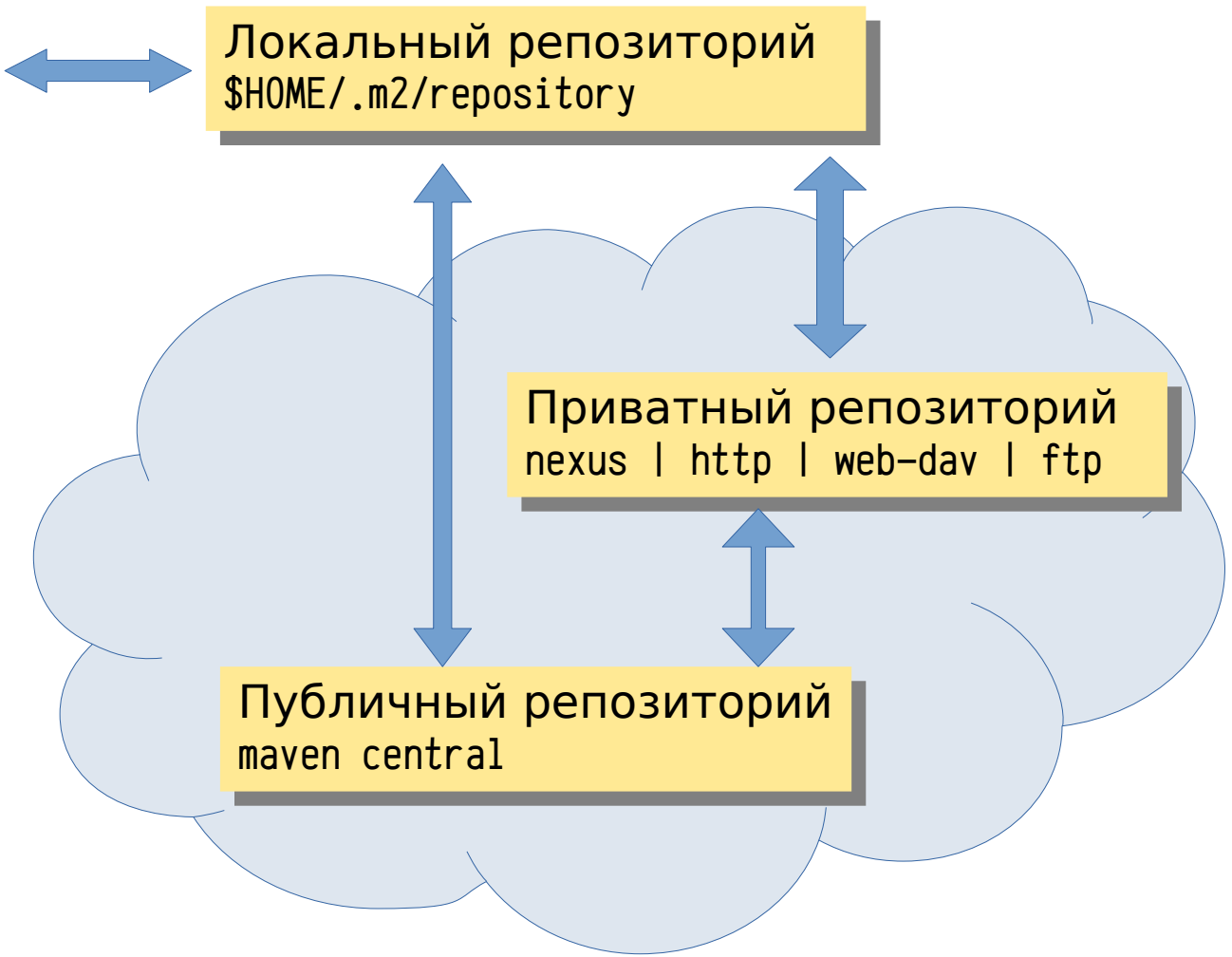
- JDK
- ОС с поддержкой JAVA

Последовательность

- Скачать и распаковать
- Прописать переменные ОС
  - JAVA_HOME
  - PATH

Структура проекта

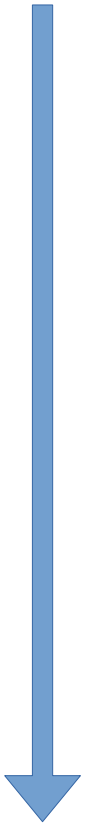
```
Project
    pom.xml
    src/
        main/
            java/
            resources/
        site/
            markdown/
                intro.md
        test/
            java/
            resources/
    target/
        classes/
        project-1.0.jar
```



Основные фазы сборки проекта

Phase

- compile
  - скомпилировать исходный код проекта
- test
  - Тестирование скомпилированного исходный кода.
- package
  - Упаковать скомпилированный код в распространяемый формат, например JAR.
- install
  - Установить пакет в локальный репозиторий.
- deploy
  - Копирует пакет в удаленный репозиторий для совместного использования.
- clear
  - Очистить каталог сборки
- site
  - Генерировать документацию



compile
*.java → *.class

test
run junit

package
*.class → *.jar

install
jar → $HOME/.m2

deploy
jar → http://

Создание проекта из архетипа 0

```
mvn \
archetype:generate \
-DarchetypeGroupId=org.apache.maven.archetypes \
-DarchetypeArtifactId=maven-archetype-quickstart \
-DarchetypeVersion=1.4
```

Архетип — это некая стандартная компоновка файлов и каталогов в проектах различного рода (веб, swing-проекты и прочие)

Архетип — определен это артефакт maven определенного типа

Создание проекта из архетипа 1

```
> mvn \
> archetype:generate \
> -DarchetypeGroupId=org.apache.maven.archetypes \
> -DarchetypeArtifactId=maven-archetype-quickstart \
> -DarchetypeVersion=1.4
WARNING: An illegal reflective access operation has occurre
WARNING: Illegal reflective access by com.google.inject.int
) to method java.lang.ClassLoader.defineClass(java.lang.Str
WARNING: Please consider reporting this to the maintainers of com.google.inject.internal.cglib.
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access oper
WARNING: All illegal access operations will be denied in a future release
[WARNING]
[WARNING] Some problems were encountered while building the effective settings
[WARNING] Unrecognised tag: 'build' (position: START_TAG seen ...</activation>\n      <build>..
ngs.xml, line 41, column 14
[WARNING]
[INFO] Scanning for projects...
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-a
l
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugins/maven-ar
 (949 B at 986 B/s)
```

Красным — Введенная команда
Оранжевый — сообщение java 11+, плохие настройки maven.
Желтый — сообщение, скачен пакет из central, по указанному url

Создание проекта из архетипа 2

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quick
rchetype-quickstart-1.4.pom (1.6 kB at 12 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/ap
hetype-bundles-1.4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apa
etype-bundles-1.4.pom (4.5 kB at 34 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/ap
archetype-quickstart-1.4.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apa
rchetype-quickstart-1.4.jar (7.1 kB at 54 kB/s)
Define value for property 'groupId': org.example
Define value for property 'artifactId': mvn-01-intro
Define value for property 'version' 1.0-SNAPSHOT: :
Define value for property 'package' org.example: :
Confirm properties configuration:
groupId: org.example
artifactId: mvn-01-intro
version: 1.0-SNAPSHOT
package: org.example
 Y: : Y
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: maven-archetype-quickstart:1.4
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: org.example
[INFO] Parameter: artifactId, Value: mvn-01-intro
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
```

Красным — ввод параметров генерируемого проекта

Желтым — подтверждение

Создание проекта из архетипа 3

Красным — успешно
Желтым — сгененированные файлы

```
[INFO] Project created from Archetype in dir: /home/user
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:07 min
[INFO] Finished at: 2020-12-28T03:59:18+05:00
[INFO] ------------------------------------------------------------------------
user@user-Modern-14-A10RB:03:59:18:~/code/samples/maven-intro:
> cd mvn-01-intro/
user@user-Modern-14-A10RB:04:15:47:~/code/samples/maven-intro/mvn-01-intro:
> find .

./pom.xml
./src
./src/main
./src/main/java
./src/main/java/org
./src/main/java/org/example
./src/main/java/org/example/App.java
./src/test
./src/test/java
./src/test/java/org
./src/test/java/org/example
./src/test/java/org/example/AppTest.java
user@user-Modern-14-A10RB:04:15:51:~/code/samples/maven-intro/mvn-01-intro:
```

Pom.xml

Координаты сборки

GroupId, artifactId, version

Свойства

Properties

Зависимости

Dependency

Плагины сборки

plugin

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>mvn-01-intro</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>mvn-01-intro</name>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

Pom.xml

Координаты сборки

GroupId, artifactId, version

Свойства

Properties

Зависимости

Dependency

Плагины сборки

plugin

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>mvn-01-intro</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>mvn-01-intro</name>
  <properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <pluginManagement>
      <plugins>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

Effective pom

POM **(Project Object Model)**

расшифровывается как «объектная модель проекта».

POM

Это дерево объектов, где каждый узел описывает аспекты сборки проекта

POM

Можно наследовать и переопределять модель

Просмотр эффективного POM

```
mvn help:effective-pom
```

```
> mvn help:effective-pom 1>pom-eff.xml 2>/dev/null && ll -h
итого 36K
drwxrwxr-x 4 user user 4,0K дек 28 12:03 ./
drwxrwxr-x 3 user user 4,0K дек 28 03:59 ../
drwxrwxr-x 4 user user 4,0K дек 28 11:22 .idea/
-rw-rw-r-- 1 user user  943 дек 28 11:22 mvn-01-intro.iml
-rw-rw-r-- 1 user user  11K дек 28 12:08 pom-eff.xml
-rw-rw-r-- 1 user user 2,6K дек 28 03:59 pom.xml
drwxrwxr-x 4 user user 4,0K дек 28 03:59 src/
```

GAV координаты

Основная цель maven

Создание артефактов

Артефакт

Файл с определенным идентификатором

Идентификатор

Group — Указывает на команду разработки, обычно имя домена (com.mycompany.app)

Artifact — Имя артефакта

Version — Версия артефакта

Опционально

Тип — (jar/war/pom/rar/...)

Классификатор - sources/javadoc

```
<groupId>org.example</groupId>
<artifactId>mvn-01-intro</artifactId>
<version>1.0-SNAPSHOT</version>

...
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
```

Зависимости

Добавляют

- Для проекта
  - Для основной области
  - Для тестов
- Для plugin-ов

Зависимости

- Указывают на артефакты
- Транзитивные

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.11</version>
  <scope>test</scope>
</dependency>
```

Области зависимостей

compile

Зависимости компиляции доступны во всех путях к классам проекта. Кроме того, эти зависимости распространяются на зависимые проекты.

provided

Это очень похоже на *compile*, но указывает на то, что вы ожидаете, что JDK или контейнер предоставят зависимость во время выполнения (например Servlet API). Зависимость с этой областью видимости добавляется к пути к классам, используемому для компиляции и тестирования, но не к пути к классам среды выполнения. Не транзитивно.

runtime

что зависимость не требуется для компиляции, а предназначена для выполнения.

test

зависимость не требуется для нормального использования приложения и доступна только на этапах компиляции и выполнения теста. Эта область не является переходной.

- system
- import

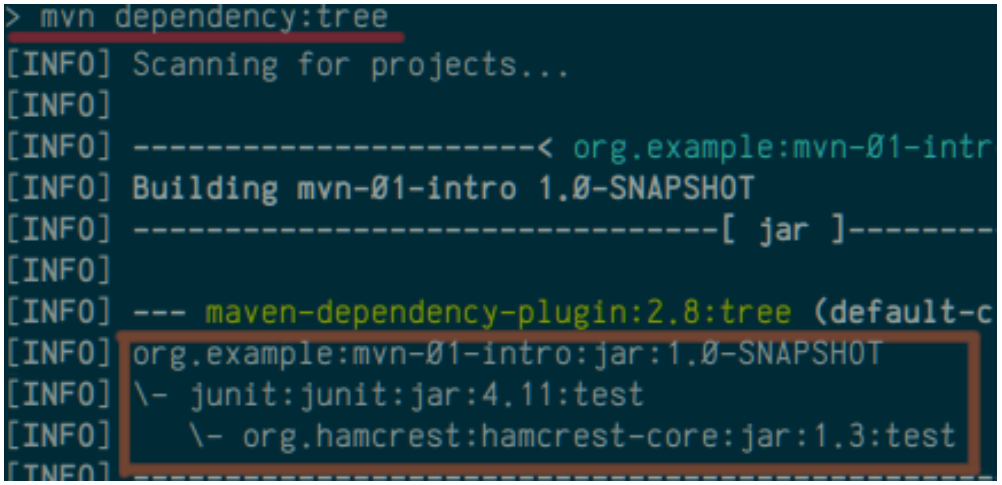
Транзитивные зависимости

Пример

- Проект mvn-01-intro ссылается на junit
  - mvn-01-intro → junit
- Junit ссылается на hamcrest-core
  - Junit → hamcrest-core
- Следовательно mvn-01-intro ссылается на hamcrest-core
  - mvn-01-intro → junit → hamcrest-core

Просмотр зависимостей

```
mvn dependency:tree
```



Сборка проекта 0

Сборка проекта

`mvn фаза`

mvn install

Включает в себя фазы

- compile
- test
- package

На каждой фазе будут запущены плагины

- compile
  - maven-resources-plugin:3.0.2:resources
  - maven-compiler-plugin:3.8.0:compile
- test
  - maven-resources-plugin:3.0.2:testResources
  - maven-compiler-plugin:3.8.0:testCompile
  - maven-surefire-plugin:2.22.1:test
- package
  - maven-jar-plugin:3.0.2:jar
- install
  - maven-install-plugin:2.5.2:install

Сборка проекта 1

```
> mvn install
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.google.inject.internal.cglib.core.$Ref
lectUtils$1 (file:/usr/share/maven/lib/guice.jar) to method java.lang.ClassLoade
r.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of com.google.inject.
internal.cglib.core.$ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< org.example:mvn-01-intro >----------------------
[INFO] Building mvn-01-intro 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ mvn-01-i
ntro ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/user/code/samples/maven-intro/m
vn-01-intro/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ mvn-01-intro
---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/user/code/samples/maven-intro/mvn-01-int
ro/target/classes
```

```
> tree
.
├── mvn-01-intro.iml
├── pom-eff.xml
├── pom.xml
└── src
    ├── main
    │   └── java
    │       └── org
    │           └── example
    │               └── App.java
    └── test
        └── java
            └── org
                └── example
                    └── AppTest.java
```

`mvn install`
: Запуск сборки

`maven-resources-plugin`
: Копирование ресурсов

`maven-compiler-plugin`
: Компиляция исходников

Сборка проекта 2

Цель — `testResources`

Копирование тестовых ресурсов

`testCompile`

Компилирование тестов

`test`

Запуск тестов

```
ro/target/classes
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @
mvn-01-intro ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/user/code/samples/maven-intro/m
vn-01-intro/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ mvn-0
1-intro ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /home/user/code/samples/maven-intro/mvn-01-int
ro/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ mvn-01-intro ---
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/
```

Сборка проекта 3

```
urefire/surefire-junit4/2.22.1/surefire-junit4-2.22.1.jar (85 kB at 311 kB/s)
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.example.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.039 s -
 in org.example.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ mvn-01-intro ---
[INFO] Building jar: /home/user/code/samples/maven-intro/mvn-01-intro/target/mvn
-01-intro-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ mvn-01-intro -
--
[INFO] Installing /home/user/code/samples/maven-intro/mvn-01-intro/target/mvn-01
-intro-1.0-SNAPSHOT.jar to /home/user/.m2/repository/org/example/mvn-01-intro/1.
0-SNAPSHOT/mvn-01-intro-1.0-SNAPSHOT.jar
[INFO] Installing /home/user/code/samples/maven-intro/mvn-01-intro/pom.xml to /h
ome/user/.m2/repository/org/example/mvn-01-intro/1.0-SNAPSHOT/mvn-01-intro-1.0-S
NAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

`TEST` — Запуск тестов
Запущен 1 тест, 0 ошибок, 0 пропущено

`jar` — сборка jar файла

`install` — установка jar файла в локальный репозиторий

Installing ...mvn-01-intro-1.0-SNAPSHOT.jar to … .m2/repository
Installing ...pom.xml to … .m2/repository

Сборка проекта 4

target/

Содержит результат сборки

`Classes/ test-classes/`

Скомпилированные исходники и тесты

`surfire-reports/`

Результаты тестов

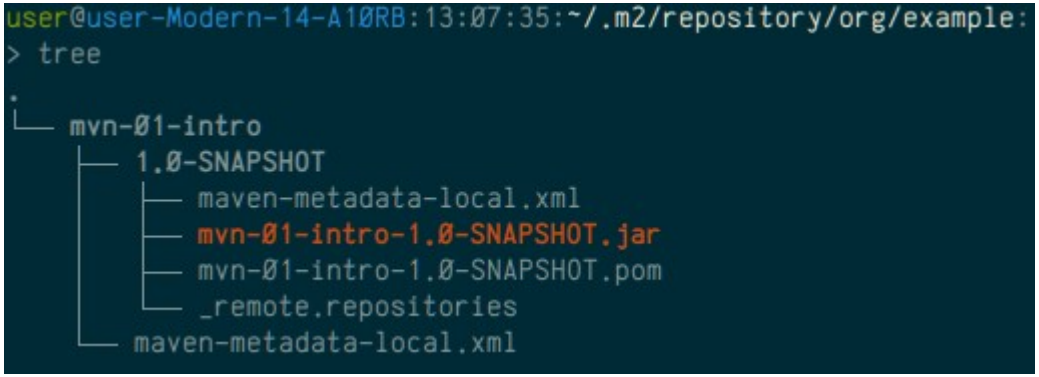
`artifact-version.jar`

Упакованные, исполняемые, бинарные файлы

Сборка проекта 5 — локальный репозиторий

В локальном репозитории

- Расположение по умолчанию $HOME/.m2/repository
- Артефакт располагается относительно groupId (точки заменяются на слеш /)
- Для разных версий свой каталог
- Обязательно для каждого артефакта наличие pom файла



goals

Для каждой фазы

```
maven-install-plugin:2.5.2:install (default-install)
```

Есть свой плагин

Плагин может предоставлять несколько целей (goals), которые запускаются в той или иной фазе сборки

maven-install

Название плагина

2.5.2

Версия плагина

install

Название цели (goals)

default-install

Название цели,

Однотипных целей может быть несколько, у каждой должен быть уникальный идентификатор

Properties

Properties

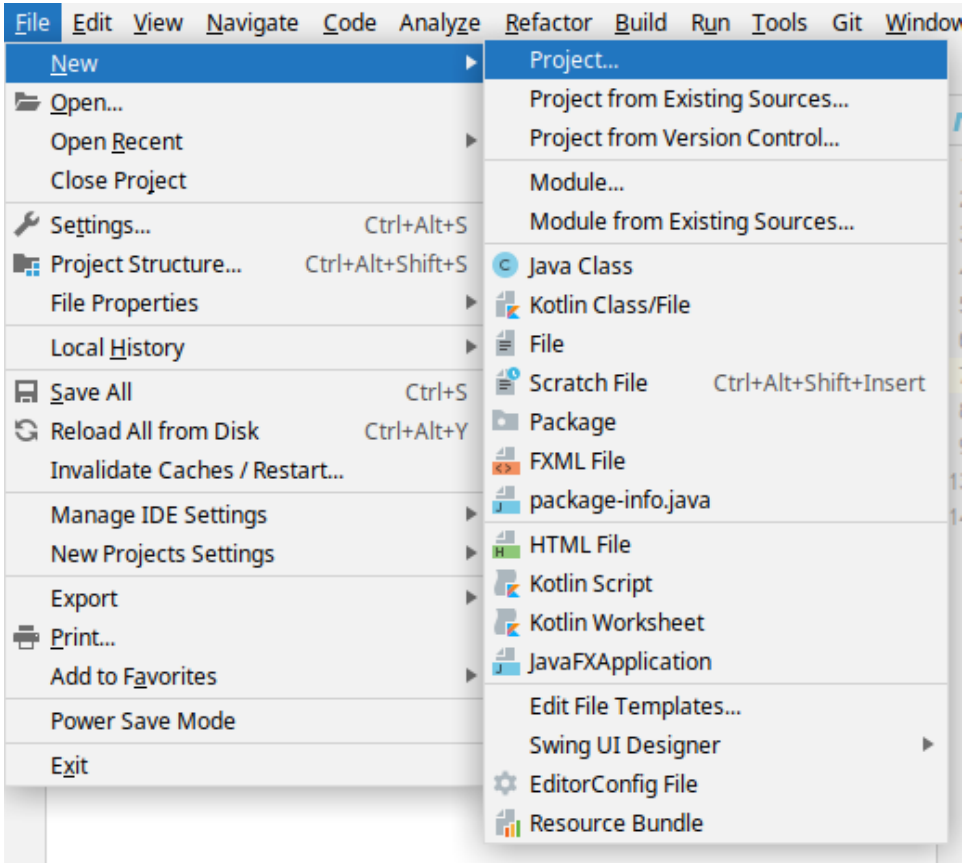
- Это текстовые переменные
- Переменные могут наследоваться от родительского проекта
- Переменные могут передавать параметры
  - Плагинам
  - В исходный код/ресурсы
  - …
- Есть предопределенные переменные, можно создавать свои
  - env.X — Указывает на переменную окружения X, например: **${env.PATH}**
  - project.x — Указывает на параметры проекта, например **${project.version}**

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.7</maven.compiler.source>
  <maven.compiler.target>1.7</maven.compiler.target>
  <maven-clean-plugin.version>3.1.0</maven-clean-plugin.version>
</properties>

<dependencies...>

<build>
  <pluginManagement><!-- lock down plugins versions to avoid using Maven
    <plugins>
      <!-- clean lifecycle, see https://maven.apache.org/ref/current/mav
      <plugin>
        <artifactId>maven-clean-plugin</artifactId>
        <version>${maven-clean-plugin.version}</version>
```

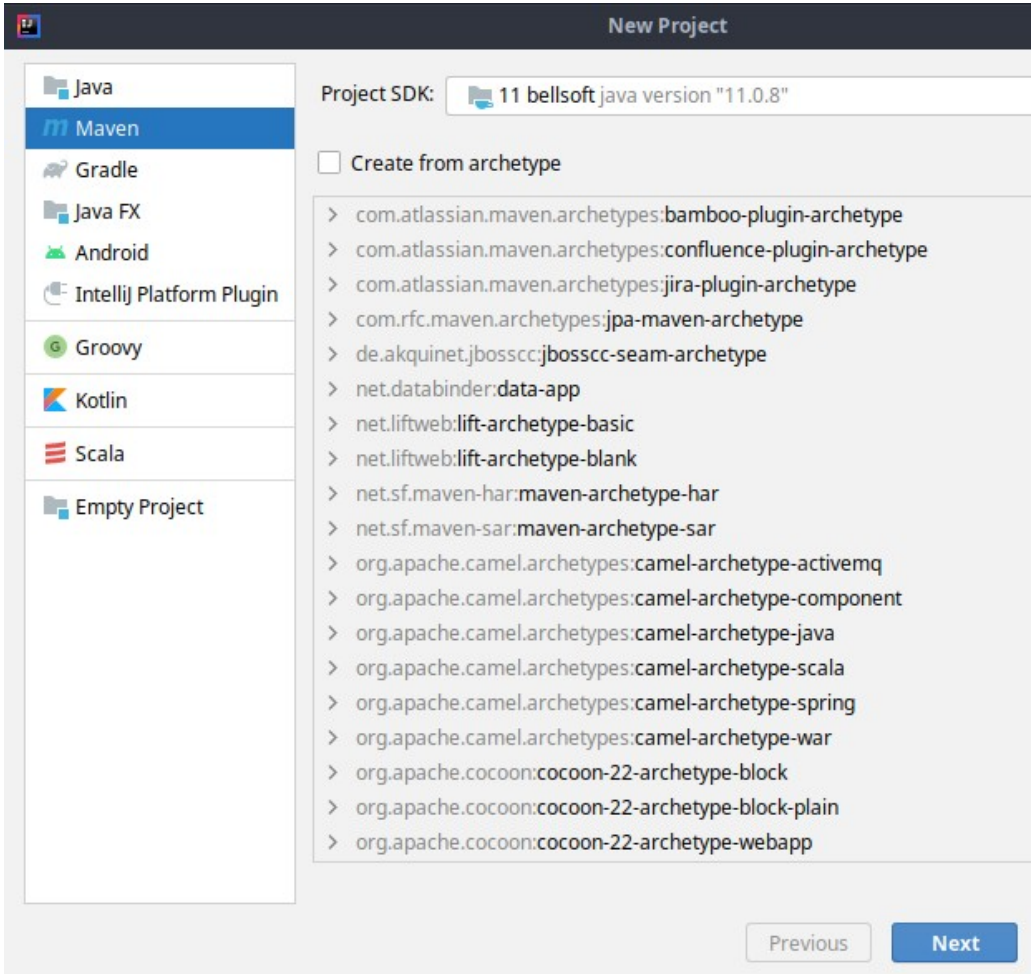
Maven idea 0



Меню

File → New → Project

Maven idea 1



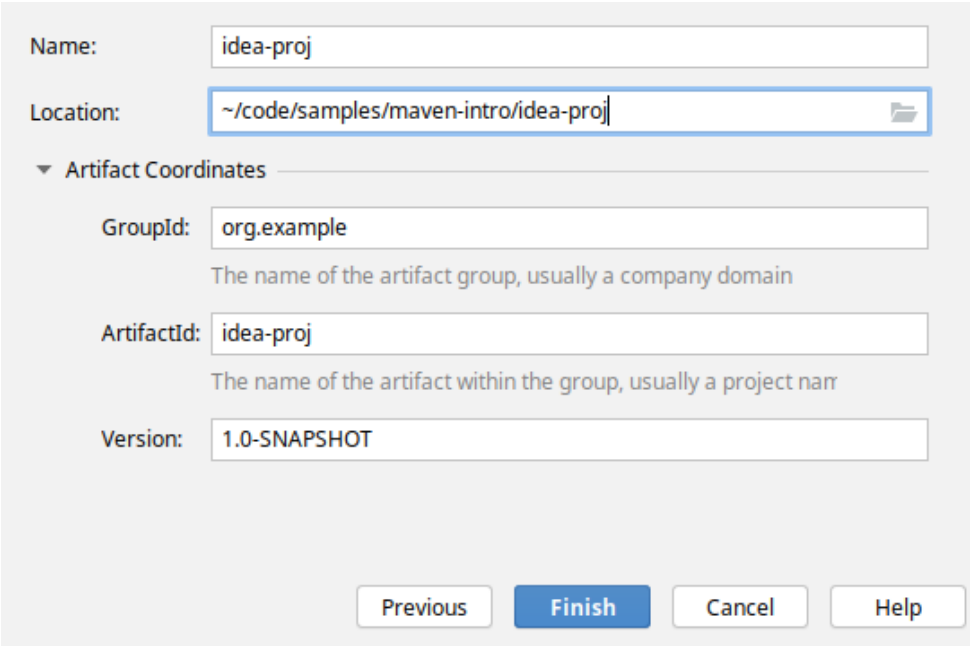
New Project

Тип проекта → Maven

Указываем JDK

Можно создать из архетипа

Maven idea 2



New Project /2

Указываем имя проекта

расположение проекта

GAV координаты

Maven idea 3

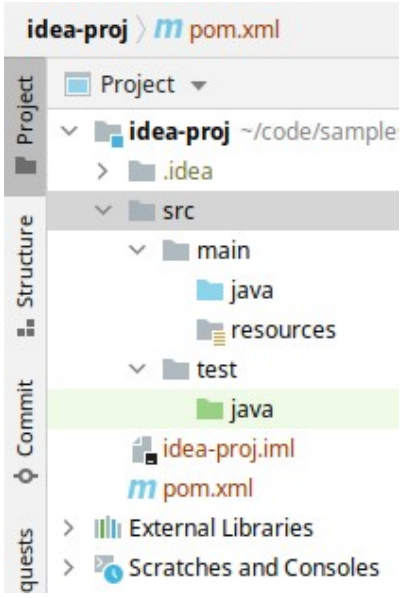
Вкладка Project

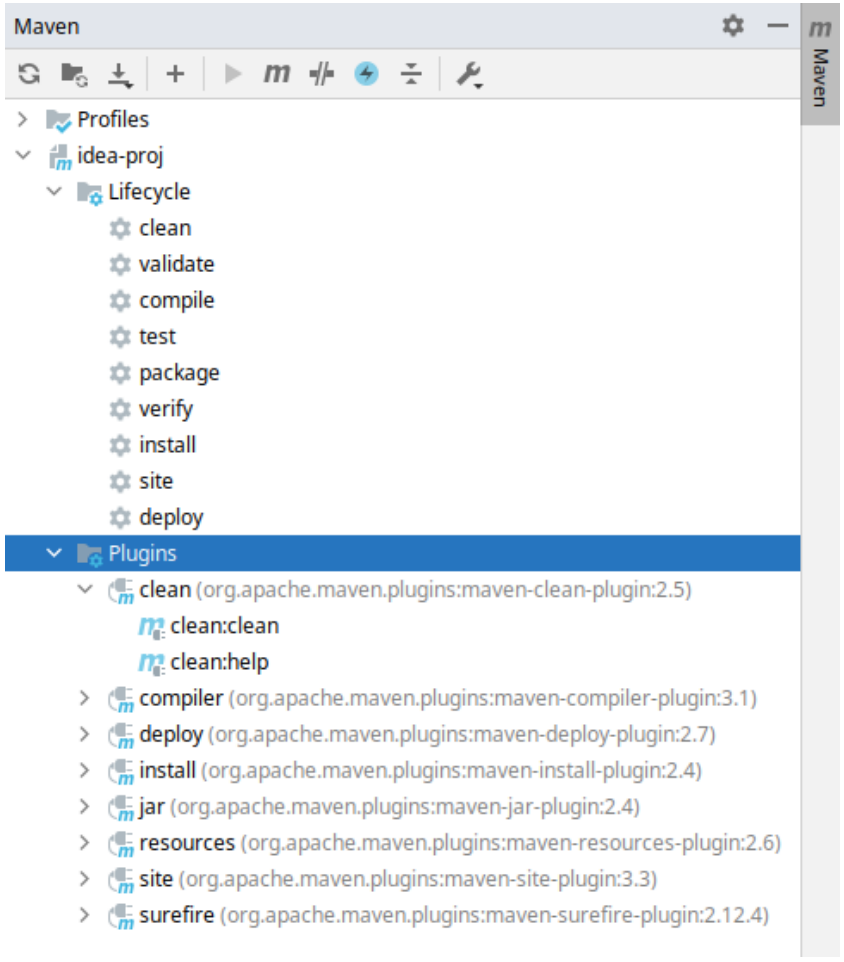
Содержит список файлов

Вкладка Maven

Содержит

- Профили сборки
- Модули сборки (idea-proj)
- Список фаз
- Список плагинов и их целей





mvn --help

usage:

```
mvn [options] [<goal(s)>]
[<phase(s)>]
```

Options:

-X,--debug

Produce execution debug output

-T,--threads <arg>

Thread count, for instance 2.0C

where C is core multiplied

-s,--settings <arg>

Alternate path for the user settings file

-o,--offline

Work offline

-U,--update-snapshots

Forces a check for missing releases and updated snapshots on remote repositories

-f,--file <arg>

Force the use of an alternate POM file (or directory with pom.xml)