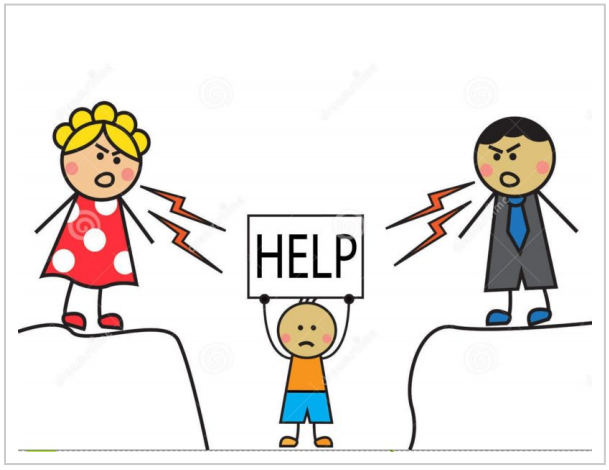
← Назад Вперед →

Многомодульные проекты maven

Опубликовано **Октябрь 10, 2017** автор **EasyJava** — **Нет комментариев** ↓



Со временем все программные проекты разрастаются. То, что начиналось как довольно жирный Hello World, весьма скоро обзаводится отдельным фронтендом, парочкой batch процессов, тремя видами RPC и общим кодом доступа к данным. И вот, в какой-то момент времени, возникает желание распилить этого монстра на неколько раздельных maven проектов, которые будут существовать независимо друг от друга.

Однако на пути к светлому многоартефактному будущему имеются некоторые препятствия — артефакты имеют зависимости друг от друга, требуют использования одной и той же версии какой-то библиотеки, должны собираться все вместе и так далее. К счастью, в

maven есть механизм для автоматического решения этих проблем — многомодульные проекты.

Многомодульный проект проще всего представить себе как дерево — у него есть общий корень, который ничего не делает, а лишь описывает общие параметры, и листья, которые наследуют эти общие параметры. Листья могут иметь свои листья и так далее, пока память не кончится 🙂

Родительский модуль

Родительский модуль состоит из одного лишь *pom* файла, в котором описаны его дочерние модули:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
    <modelVersion>4.0.0</modelVersion>
    <groupId>ru.easyjava.maven</groupId>
    <artifactId>multi</artifactId>
    <packaging>pom</packaging>
    <version>1</version>
    <name>multi</name>
    <url>http://maven.apache.org</url>

    <modules>
        <module>targets</module>
        <module>greeter</module>
    </modules>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.12</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

```
24
25      <build>
26          <plugins>
27              <plugin>
28                  <groupId>org.apache.maven.plugins</groupId>
29                  <artifactId>maven-compiler-plugin</artifactId>
30                  <version>3.3</version>
31                  <configuration>
32                      <source>1.8</source>
33                      <target>1.8</target>
34                      <encoding>UTF-8</encoding>
35                  </configuration>
36              </plugin>
37          </plugins>
38      </build>
39  </project>
```

В родительском *pom* есть две важные вещи — он должен определять `<packaging/>` как *pom* и должен перечислять дочерние модули. Всё остальное — как у обычного проекта: зависимости, плагины и так далее. Все настройки сборки, определённые в родительстком модуле, будут автоматически наследоваться дочерними.

Дочерний модуль

Дочерний модуль так же обязательно имеет свой *pom* файл и может как иметь код, так и быть родительским модулем для своих подмодулей. В моём случае это будет модуль с кодом.

```
XHTML
1  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
2    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maver
3    <modelVersion>4.0.0</modelVersion>
4      <parent>
5          <groupId>ru.easyjava.maven</groupId>
6          <artifactId>multi</artifactId>
7          <version>1</version>
8      </parent>
```

```
  <artifactId>targets</artifactId>
  <packaging>jar</packaging>

</project>
```

Дочерний модуль должен обязательно ссылаться на родительский модуль с помощью элемента `<parent>` Всё остальное — как у обычного maven проекта. В моём модуле будет один простой класс и тест к нему:

```java
public class GreetingTarget
{
    public static String getTarget() { return "Modules"; }
}
```

```java
public class GreetingTargetTest {
    @Test
    public void testGreet() {
        assertThat(GreetingTarget.getTarget(), is("Modules"));
    }
}
```

Обратите внимание, что в коде класса теста используется JUnit, зависимость от которого объявляется в родительском модуле.

Зависимости между дочерними модулями

Иметь многомодульный проект из только одного модуля — грустно. Поэтому давайте напишем класс, который будет использовать уже существующие классы и поместим этот класс в новый модуль:

```java
public class Greeter
{
    public static void main(String[] args) {
```

```
        System.out.println(String.format("Hello, %s!", GreetingTarget.getTarget()));
    }
}
```

Итак, теперь у нас есть два модуля и один зависит от другого. Такую зависимость можно описать в pom файле второго модуля:

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>ru.easyjava.maven</groupId>
        <artifactId>multi</artifactId>
        <version>1</version>
    </parent>
    <artifactId>greeter</artifactId>
    <packaging>jar</packaging>

    <dependencies>
        <dependency>
            <groupId>ru.easyjava.maven</groupId>
            <artifactId>targets</artifactId>
            <version>1</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <configuration>
                    <archive>
                        <manifest>
                            <mainClass>ru.easyjava.maven.Greeter</mainClass>
                        </manifest>
                    </archive>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</descriptorRef>
                    </descriptorRefs>
                </configuration>
                <executions>
                    <execution>
```

```
36                          <id>make-assembly</id>
37                          <phase>package</phase>
38                          <goals>
39                              <goal>single</goal>
40                          </goals>
41                      </execution>
42                  </executions>
43              </plugin>
44          </plugins>
45      </build>
46  </project>
```

Теперь maven знает, что перед сборкой второго модуля надо собрать первый и результаты сборки сделать доступными второму. Maven так же будет сам следить за изменениями в первом модуле и пересобирать второй по мере надобности. Стоит отметить, что если maven не может выявить зависимости между модулями, то он их будет собирать в том порядке, в котором они перечислены в элементе `<modules/>`.

Так второй модуль зависит от первого, это надо учесть и при сборке финального jar файла, поэтому я добавляю плагин *maven-assembly-plugin,* тем самым переопределяя унаследованные от родительского модуля настройки. Запуск готового приложения подтверждает корректность сборки:

Shell

```
1  MultiModule\greeter\target>java -jar greeter-1-jar-with-dependencies.jar
2  Hello, Modules!
```

Код примера доступен на github

Эта запись была размещена в **Apache Maven** и помечено как **maven** по **EasyJava** (**постоянная ссылка**).

Рекомендованное чтение

Архетипы Maven

Самый простой и удобный способ создания нового проекта в Apache maven, это создание его из…

Создание Maven проекта

Apache Maven это система сборки проектов. Поэтому сразу после установки Apache Maven создадим первый проект,…

Управление зависимостями в Maven

Apache Maven - отличная штука, для управления сторонними зависимостями в вашем проекте. Достаточно сказать ему,…

Подписаться

Ваш e-mail адрес

Подписаться на обновления

Метки

h2 hibernate spring unit-test java jpa jdbc
spring-data-access hello-world junit spring-jdbc database-pooling sql
hikaricp stream-api PostgreSQL easymock maven lombok
spring-data-jpa functional spring-core spring-boot lambda c3po quicktip
di ioc spring-webmvc collections exceptions integration-test mock spring-orm
spring-tx multitenancy dbcp derby hsqldb mysql hamcrest orm ehcache
hql jpql db2 jetty jcache servlet mssql archetype querydsl builder
transaction jodatime groovy xml foreach multi-threading rdbms gradle guava
spliterator

Search