

## Полезные Maven-плагины

JSP & Servlets  
2 уровень, 5 лекция

ОТКРЫТА

### Свой maven-репозиторий на GitHub

Разработчики могут загружать свою библиотеку в GitHub, для этого у него есть специальный плагин **site-maven-plugin**. Давай рассмотрим пример его использования:

```
<project>
  <properties>
    <github.global.server>github</github.global.server>
    <github.maven-plugin>0.9</github.maven-plugin>
  </properties>

  <distributionManagement>
    <repository>
      <id>internal.repo</id>
      <name>Temporary Staging Repository</name>
      <url>file://${project.build.directory}/mvn-repo</url>
    </repository>
  </distributionManagement>

  <build>
    <plugins>
      <plugin>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.8.1</version>
        <configuration>
          <altDeploymentRepository>
            internal.repo::default::file://${project.build.directory}/mvn-repo
          </altDeploymentRepository>
        </configuration>
      </plugin>
      <plugin>
        <groupId>com.github.github</groupId>
        <artifactId>site-maven-plugin</artifactId>
        <version>${github.maven-plugin}</version>
        <configuration>
          <message>Maven artifacts for ${project.version}</message>
          <noJekyll>true</noJekyll>
          <outputDirectory>${project.build.directory}/mvn-repo</outputDirectory>
          <branch>refs/heads/mvn-repo</branch>
          <includes>*/*</includes>
          <repositoryName>SuperLibrary</repositoryName>
          <repositoryOwner>javarushu-student</repositoryOwner>
        </configuration>
        <executions>
          <execution>
            <goals>
```

```
        <goal>site</goal>
      </goals>
      <phase>deploy</phase>
    </execution>
  </executions>
</plugin>
</plugins>
</build>

</project>
```

Разберемся, что тут написано.

Синим цветом выделено создание временного локального репозитория. Технически это просто папка, но нам нужно, чтобы Maven-рассматривал ее как отдельный репозиторий.

Красным цветом мы выделили запуск плагина **maven-deploy-plugin**, где указали, что собранную библиотеку нужно класть именно в этот временный репозиторий.

И, наконец, зеленым цветом выделен плагин **site-maven-plugin**, который должен взять все файлы из репозитория и закомитить их на GitHub. Тут понадобятся некоторые пояснения. Все параметры делятся на две группы: что заливаем и куда заливаем.

Что заливаем:

- outputDirectory** — директория, где брать файлы для коммита
- includes** — задает маску файлов для коммита

Куда заливаем:

- repositoryOwner** — имя владельца репозитория на GitHub
- repositoryName** — имя репозитория
- branch** — задает ветку репозитория на GitHub, в которую комитить
- message** — сообщение, которое будет добавлено при коммите

Так же нужно указать логин и пароль к своему репозиторию в Maven **setting.xml**:

```
<settings>
  <servers>
    <server>
      <id>github</id>
      <username>[username]</username>
      <password>[password]</password>
    </server>
  </servers>
</settings>
```

Чтобы подключить (использовать) библиотеку из GitHub-репозитория в другой проект, нужно указать этот репозиторий в своем **pom.xml**:

```
<repositories>
  <repository>
    <id>[name-project]-mvn-repo</id>
    <url>https://raw.github.com/[username]/[name-project]/mvn-repo/</url>
    <snapshots>
      <enabled>true</enabled>
      <updatePolicy>always</updatePolicy>
    </snapshots>
  </repository>
</repositories>
```

После этого Maven будет понимать, откуда брать библиотеку.

- **[name-project]** — это имя проекта, в нашем случае SuperLibrary
- **[username]** — это логин на GitHub, в примере это javarush-user

## Запаковываем сборку в Docker образ

Мы живем в новое время, когда проекты в результате сборки могу класться в Maven-репозиторий, а могут и в docker storage.

Чтобы подружить Maven и Docker, нам понадобится плагин **docker-maven-plugin**. Ничего сложного:

```
<build>
  <plugins>
    <plugin>
      <groupId>com.spotify</groupId>
      <artifactId>docker-maven-plugin</artifactId>
      <version>0.4.10</version>
      <configuration>
        <dockerDirectory>${project.basedir}</dockerDirectory>
        <imageName>javarush/${project.artifactId}</imageName>
      </configuration>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>build</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

Синим выделен момент, где мы добавили goal bulid в package-фазу сборки. Его можно вызвать с помощью команды **mvn docker:build**.

Тег **dockerDirectory** задает папку, где находится Dockerfile. А имя образа задается с помощью тега **imageName**.

Если проект упакован в jar-файл, то docker-файл будет выглядеть примерно так:

```
FROM java:11
EXPOSE 8080
ADD /target/demo.jar demo.jar
ENTRYPOINT ["java","-jar","demo.jar"]
```

Если же ты упаковываешь web-приложение, то может понадобится добавить Tomcat:

```
FROM tomcat8
ADD sample.war ${CATALINA_HOME}/webapps/ROOT.war
CMD ${CATALINA_HOME}/bin/catalina.sh run
```

Если тебе интересно, как подружить Maven, Docker и SpringBoot приложения, то для тебя есть [хорошая статья](#).

−

+16

+

Комментарии (3)

популярные

новые

старые

JavaCoder

Введите текст комментария

Роберт Тахненко

Java Developer в СберТех

6 июля, 10:49

...

"Если проект упакован в jar-файл, то dickeг-файл будет выглядеть примерно так:"

Исправьте, плиз, на docker-файл)

Ответить

−

+8

+

Андрей Пазюк

Уровень 70, Винница, Украина

14 июля, 17:30

...

[Роберт](#), не нужно

Ответить

−

+16

+



Mentor-02

Backend Developer в JavaRush

MENTOR

31 октября, 18:42

...

спасибо, поправили

Ответить

−

0

+

ОБУЧЕНИЕ

[Курсы программирования](#)

[Курс Java](#)

[Помощь по задачам](#)

[Подписки](#)

[Задачи-игры](#)

СООБЩЕСТВО

[Пользователи](#)

[Статьи](#)

[Форум](#)

[Чат](#)

[Истории успеха](#)

[Активности](#)

КОМПАНИЯ

[О нас](#)

[Контакты](#)

[Отзывы](#)

[FAQ](#)

[Поддержка](#)

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

 Русский

▼

СКАЧИВАЙТЕ НАШИ ПРИЛОЖЕНИЯ

 ДОСТУПНО В  
Google Play

 Загрузите в  
App Store

