

# Создание дистрибутива средствами maven

Камнев Георгий  
2020

# Приложение для демонстрации

## Описание

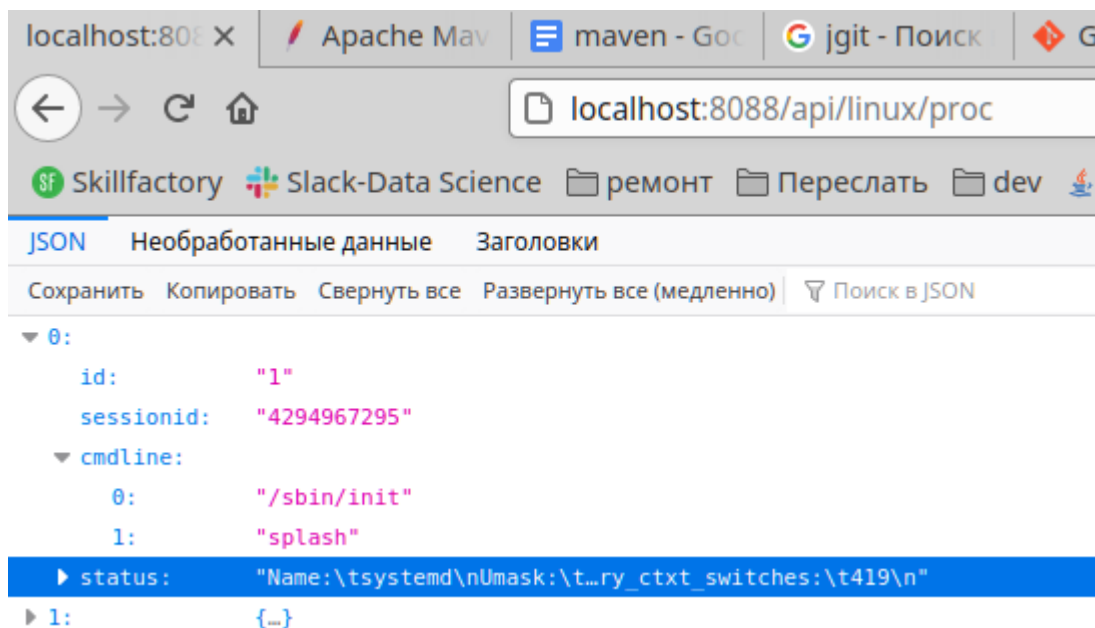
Просмотр процессов ОС

Spring-boot

Java 11

Аутентификация

BASIC



```
localhost:8088 X Apache Mav maven - Gos jgit - Поиск G
localhost:8088/api/linux/proc
Skillfactory Slack-Data Science ремонт Переслать dev
JSON Необработанные данные Заголовки
Сохранить Копировать Свернуть все Развернуть все (медленно) Поиск в JSON
0:
  id: "1"
  sessionid: "4294967295"
  cmdline:
    0: "/sbin/init"
    1: "splash"
  status: "Name:\t systemd\nUmask:\tt...ry_ctxt_switches:\tt419\n"
  1: {...}
```

<https://github.com/gochaorg/samples/tree/master/maven-intro/spring-demo>

# Изначальный Код

## Сгенерировано

<https://start.spring.io/>

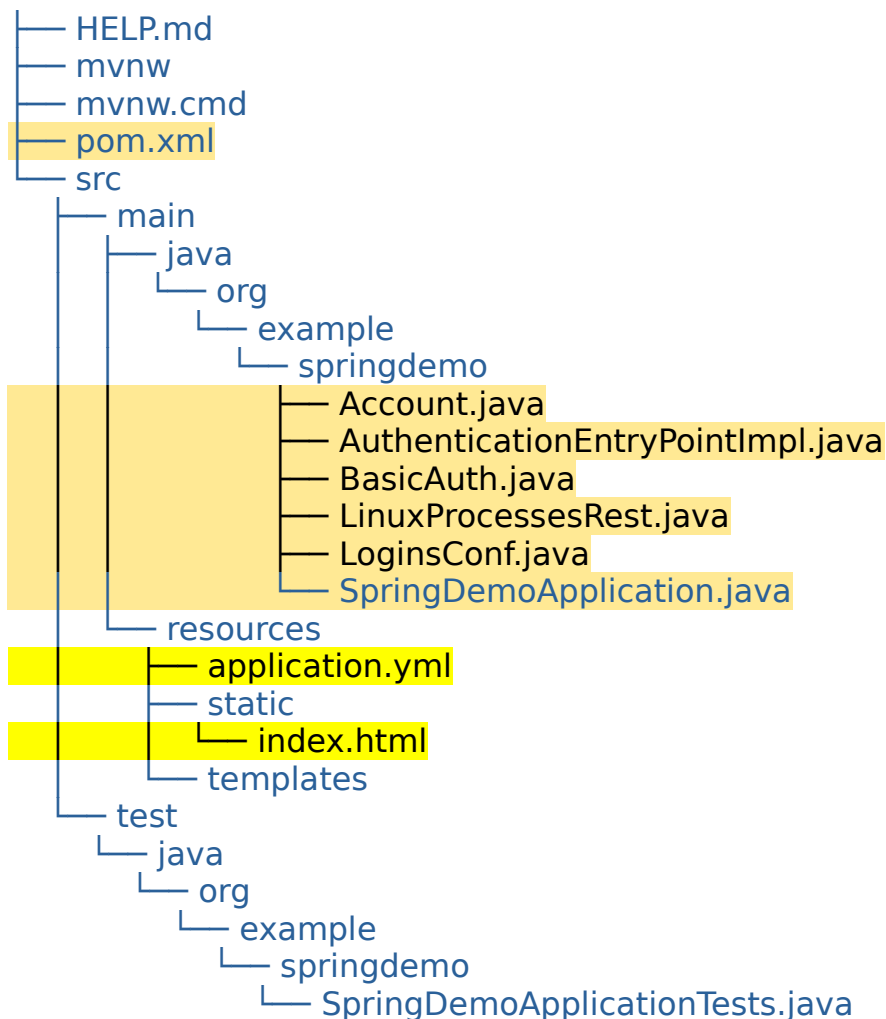
Отмечено синим

## Изменено

Отмечено желтым

## Код

296 строк java



# Изначальный pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.1</version>
    <relativePath/>
  </parent>

  <groupId>org.example</groupId>
  <artifactId>spring-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>spring-demo</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>
```

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

# Изначальный pom.xml

Используем родительский pom,  
Там уже преднастроено

```
<?xml
<project>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.1</version>
    <relativePath/>
  </parent>

  <groupId>org.example</groupId>
  <artifactId>spring-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>spring-demo</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>
```

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

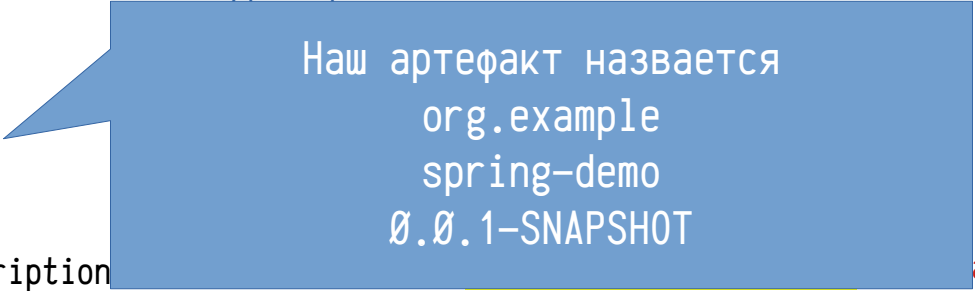
В каталог выше подниматься не надо,  
Искать надо на maven репозитории

# Изначальный pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.1</version>
    <relativePath/>
  </parent>

  <groupId>org.example</groupId>
  <artifactId>spring-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>spring-demo</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>
```

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
```



Наш артефакт называется  
org.example  
spring-demo  
0.0.1-SNAPSHOT

```
</plugin>
</plugins>
</build>
</project>
```

# Изначальный pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.1</version>
    <relativePath/>
  </parent>

  <groupId>org.example</groupId>
  <artifactId>spring-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>spring-demo</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

Используем java 11

# Изначальный pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<parent>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-parent</artifactId>  
  <version>2.4.1</version>  
  <relativePath/>  
</parent>
```

Подключаем другие артефакты  
(jar файлы)

```
<groupId>org.example</groupId>  
<artifactId>spring-demo</artifactId>  
<version>0.0.1-SNAPSHOT</version>  
<name>spring-demo</name>  
<description>Demo project for Spring Boot</description>  
<properties>  
  <java.version>11</java.version>  
</properties>
```

```
<dependencies>  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-web</artifactId>  
  </dependency>  
  <dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-test</artifactId>  
    <scope>test</scope>  
  </dependency>  
</dependencies>  
<build>  
  <plugins>  
    <plugin>  
      <groupId>org.springframework.boot</groupId>  
      <artifactId>spring-boot-maven-plugin</artifactId>  
    </plugin>  
  </plugins>  
</build>  
</project>
```



# Изначальный pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.1</version>
    <relativePath/>
  </parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-docker-demo</artifactId>
  <version>0.0.1</version>
  <name>Spring Demo</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>
```

ЭТОТ ПЛАГИН НАМ ПРИГОДИТСЯ

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

# Чего не хватает для дистрибутива ?

## **Зависимых jar**

Их 40 jar файлов (19Мб) + 1 само приложение (16Кб)

## **Файлов для запуска (sh/bat/exe)**

Пользователю нужна иконка, а ОС exe/bat

## **Пользовательской документации**

Как узнать о установке/настройке ПО

## **API и исходников**

Забота о программистах, которые будут поддерживать

## **Информации о версии**

В какой версии bug ?

## **Все в одном архиве**

Не передавать же кучу файлов, можно какой-то пропустить

# Зависимости - Толстый jar / fat jar

## Pom.xml

Для Spring-boot

Внесем правки в правила сборки

На фазе `package`

Добавим `исполнение`

Добавим `цель`

`Перепаковать` все зависимости и приложение в один jar

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <executions>
        <execution>
          <id>spring-fat-jar</id>
          <phase>package</phase>
          <goals>
            <goal>repackage</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
```

# Зависимости - Толстый jar / fat jar

## В логе

```
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ spring-demo ---  
[INFO] Building jar: /home/user/code/samples/maven-intro/spring-demo/target/spring-demo-0.0.1-SNAPSHOT.jar  
[INFO]  
[INFO] --- spring-boot-maven-plugin:2.4.1:repackage (repackage) @ spring-demo ---  
[INFO] Replacing main artifact with repackaged archive  
[INFO]  
[INFO] --- spring-boot-maven-plugin:2.4.1:repackage (spring-fat-jar) @ spring-demo ---  
[INFO] Replacing main artifact with repackaged archive
```

## На выходе, в target/

```
-rw-rw-r-- 1 user user 19M дек 29 16:17 spring-demo-0.0.1-SNAPSHOT.jar  
-rw-rw-r-- 1 user user 14K дек 29 16:17 spring-demo-0.0.1-SNAPSHOT.jar.original
```

## В jar (unzip -l target/spring-demo-0.0.1-SNAPSHOT.jar)

```
META-INF/                                BOOT-INF/classes/application.yml  
META-INF/MANIFEST.MF                    BOOT-INF/classes/static/index.html  
...                                       BOOT-INF/lib/spring-boot-2.4.1.jar  
org/springframework/boot/loader/Launcher.class  BOOT-INF/lib/spring-boot-autoconfigure-2.4.1.jar  
...  
META-INF/maven/org.example/spring-demo/pom.xml  
META-INF/maven/org.example/spring-demo/pom.properties  
...
```



# appassembler-maven-plugin

## Генерирует скрипты запуска приложения java

Bash, Bat

## Копирует зависимости в отдельный каталог

По умолчанию такая структура

bin/ - скрипты bash/bat

repo/ - каталог с jar

## Может создавать службы Windows / демоны Linux

Используется Java Service Wrapper

# appassembler-maven-plugin

## Создание приложения

### Плагин

#### GAV

org.codehaus.mojo  
Appassembler-maven-plugin  
1.10

### Конфигурация плагина

Фаза — `package`

Цель — `assemble`

Программа (shell скрипты) —  
`springdemo`

Запускаемый класс

`org.example.springdemo.SpringDemoApplication`

Структура каталог с jar — `плоская`

Использовать `*.jar` для формирования classpath

Jar файлы расположить в каталоге `jars`

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>appassembler-maven-plugin</artifactId>
  <version>1.10</version>
  <executions>
    <execution>
      <phase>package</phase>
      <id>app-binary</id>
      <goals>
        <goal>assemble</goal>
      </goals>
      <configuration>
        <programs>
          <program>
            <mainClass>org.example.springdemo.SpringDemoApplication</mainClass>
            <id>springdemo</id>
          </program>
        </programs>
        <repositoryLayout>flat</repositoryLayout>
        <useWildcardClassPath>true</useWildcardClassPath>
        <repositoryName>jars</repositoryName>
      </configuration>
    </execution>
  </executions>
</plugin>
```

# appassembler-maven-plugin

## На выходе в target/

Каталог по умолчанию  
appassembler

appassembler/bin — shell  
скрипты (bat/sh)

appassembler/jars - jar  
файлы со всеми  
зависимостями

```
> tree target/appassembler/  
target/appassembler/  
├── bin  
│   ├── springdemo  
│   └── springdemo.bat  
└── jars  
    ├── cbuffer-1.1-SNAPSHOT.jar  
    ├── ecolls-1.7.jar  
    ├── fs-1.1.jar  
    ├── iofun-1.0.jar  
    ├── jackson-annotations-2.11.3.jar  
    ├── jackson-core-2.11.3.jar  
    ├── jackson-databind-2.11.3.jar  
    ├── jackson-datatype-jdk8-2.11.3.jar  
    ├── jackson-datatype-jsr310-2.11.3.jar  
    └── jackson-module-parameter-names-  
        2.11.3.jar
```



# Документация - maven-site-plugin

Сгенерирует из src/site  
документацию  
Умеет например markdown

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-site-plugin</artifactId>
  <version>3.9.1</version>
  <dependencies>
    <dependency>
      <groupId>org.apache.maven.doxia</groupId>
      <artifactId>doxia-module-markdown</artifactId>
      <version>1.9.1</version>
    </dependency>
  </dependencies>
  <configuration>
    <inputEncoding>UTF-8</inputEncoding>
    <outputEncoding>UTF-8</outputEncoding>
    <generateReports>>false</generateReports>
  </configuration>
  <executions>
    <execution>
      <id>gen-docs</id>
      <phase>package</phase>
      <goals>
        <goal>site</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# Документация - maven-site-plugin

Дополнения при  
работе с markdown

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-site-plugin</artifactId>
  <version>3.9.1</version>
  <dependencies>
    <dependency>
      <groupId>org.apache.maven.doxia</groupId>
      <artifactId>doxia-module-markdown</artifactId>
      <version>1.9.1</version>
    </dependency>
  </dependencies>
  <configuration>
    <inputEncoding>UTF-8</inputEncoding>
    <outputEncoding>UTF-8</outputEncoding>
    <generateReports>>false</generateReports>
  </configuration>
  <executions>
    <execution>
      <id>gen-docs</id>
      <phase>package</phase>
      <goals>
        <goal>site</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# Документация - maven-site-plugin

Кодировки файлов  
Отчет о зависимостях  
и другой полезной инф

в примене выключено (false)

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-site-plugin</artifactId>
  <version>3.9.1</version>
  <dependencies>
    <dependency>
      <groupId>org.apache.maven.doxia</groupId>
      <artifactId>doxia-module-markdown</artifactId>
      <version>1.9.1</version>
    </dependency>
  </dependencies>
  <configuration>
    <inputEncoding>UTF-8</inputEncoding>
    <outputEncoding>UTF-8</outputEncoding>
    <generateReports>>false</generateReports>
  </configuration>
  <executions>
    <execution>
      <id>gen-docs</id>
      <phase>package</phase>
      <goals>
        <goal>site</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# Документация - maven-site-plugin

Генерировать на фазе package  
Цель - site

command line > mvn package

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-site-plugin</artifactId>
  <version>3.9.1</version>
  <dependencies>
    <dependency>
      <groupId>org.apache.maven.doxia</groupId>
      <artifactId>doxia-module-markdown</artifactId>
      <version>1.9.1</version>
    </dependency>
  </dependencies>
  <configuration>
    <inputEncoding>UTF-8</inputEncoding>
    <outputEncoding>UTF-8</outputEncoding>
    <generateReports>>false</generateReports>
  </configuration>
  <executions>
    <execution>
      <id>gen-docs</id>
      <phase>package</phase>
      <goals>
        <goal>site</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# Документация - maven-site-plugin

## Документация

Располагается в каталоге src/site

src/site/**site.xml**

описывает что входит в документацию

Поддерживает различные форматы

Markdown, xhtml, xhtml5, xdoc, fml, apt

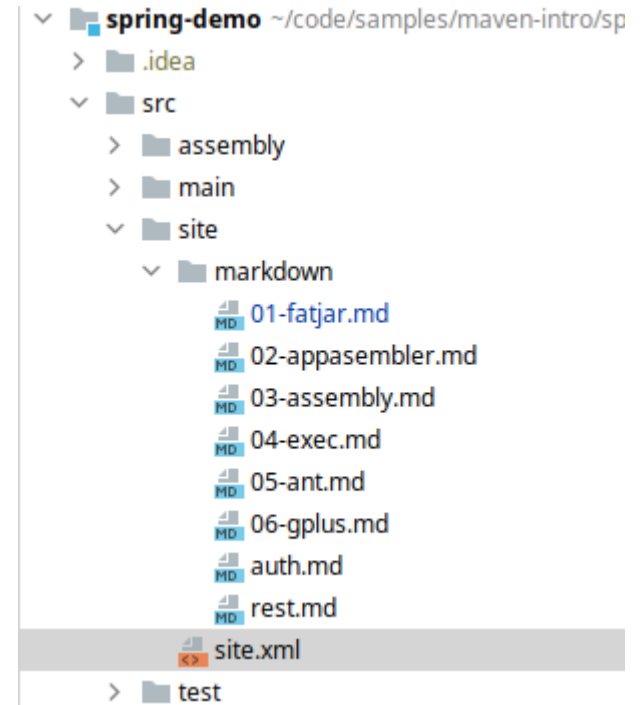
CSS, Images

Располагается в

src/site/resources/css

src/site/resources/images

```
+ - src/  
  +- site/  
    +- resources/  
      +- css/  
        | +- site.css  
        |  
      +- images/  
        +- pic1.jpg
```



# Документация - maven-site-plugin

## Исходные файлы

```
+ - src/  
  +- site/  
    +- apt/  
      | +- index.apt  
      |  
    +- fml/  
      | +- general.fml  
      | +- faq.fml  
      |  
    +- markdown/  
      | +- markup.md  
      |  
    +- xdoc/  
      | +- other.xml  
      |  
    +- xhtml/  
      | +- xhtml-too.xhtml  
      |  
    +- xhtml5/  
      | +- xhtml5-content.xhtml5  
      |  
    +- site.xml
```

mvn site:site



## Результат

```
+ - target/  
  +- site/  
    +- css/  
    |  
    +- images/  
    |  
    +- index.html  
    +- general.html  
    +- faq.html  
    +- other.html  
    +- xhtml-too.html  
    +- xhtml5-content.html
```

# Документация - maven-site-plugin

## Site.xml

В body указываем пункты меню

Можно еще подключить дополнительные заголовки, стили, js, ...

src/site/site.xml

```
<project name="watch server processes">
  <skin>
    <groupId>org.apache.maven.skins</groupId>
    <artifactId>maven-fluido-skin</artifactId>
    <version>1.8</version>
  </skin>
  <body>
    <menu name="Overview">
      <item name="REST api" href="rest.html" />
      <!-- <item name="FAQ" href="faq.html" /> -->
    </menu>
    <menu ref="reports"/>
    <footer><![CDATA[All rights reserved.]]></footer>
  </body>
</project>
```

# Документация - maven-site-plugin

## spring-demo

Last Published: 2020-12-30 | Version: 0.0.1-SNAPSHOT

OVERVIEW

REST api

PROJECT

DOCUMENTATION

Project Information ▾

Dependencies

Dependency Information

Dependency Management

About

Licenses

Plugin

Management

Plugins

Source Code

Management

Summary

Team

## Project Dependencies



### compile

The following is a list of compile dependencies for this project. These dependencies are required to compile and run the application:

GroupId	ArtifactId	Version	Type	Licenses
org.springframework.boot	spring-boot-starter-security 	2.4.1	jar	Apache License, Version 2.0 
org.springframework.boot	spring-boot-starter-web 	2.4.1	jar	Apache License, Version 2.0 
xyz.cofe	ecolls 	1.7	jar	MIT License 
xyz.cofe	fs 	1.1	jar	MIT License 

### test

The following is a list of test dependencies for this project. These dependencies are only required to compile and run unit tests for the application:

GroupId	ArtifactId	Version	Type	Licenses
org.springframework.boot	spring-boot-starter-test 	2.4.1	jar	Apache License, Version 2.0 



# Документация по коду maven-javadoc-plugin

## Javadoc

Документация будет сгенерирована из исходного кода

Результат будет в target/site/apidocs

Плагин настраиваемый (css/templates)

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-javadoc-plugin</artifactId>
  <version>3.2.0</version>
  <configuration>
    <javadocExecutable>${java.home}/bin/javadoc</
javadocExecutable>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>javadoc</goal>
      </goals>
      <phase>package</phase>
    </execution>
  </executions>
</plugin>
```

# Документация по коду maven-javadoc-plugin

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD    DETAIL: FIELD | CONSTR | METHOD

SEARCH:

**Package** org.example.springdemo

**Class Account**

java.lang.Object  
org.example.springdemo.Account

---

```
public class Account
extends java.lang.Object
```

Аккаунт

**Constructor Summary**

**Constructors**

Constructor	Description
Account()	

**Method Summary**

# Исходники — maven-source-plugin

## Исходники

Для opensource — дефакто

Для отладки (debug)

Для своей версии (fork)

## В результате — target/

target/

spring-demo-0.0.1-SNAPSHOT-*sources*.jar

Дополнительный артефакт, с классификатором *sources*

GAV координаты

org.example:spring-demo:0.0.1-SNAPSHOT:jar:sources

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-source-plugin</artifactId>
  <executions>
    <execution>
      <phase>package</phase>
      <goals>
        <goal>jar</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# Сборка — `maven-assembly-plugin`

**Создает `attached` артефакты**

**Работает с многомодульными проектами `maven`**

**Можно создавать манифесты `java`**

**Работает с зависимостями**

**Attached Артефакты**

Являются обычно архивами `zip/jar/...`

Могут содержать различный набор файлов

Можно `deploy` артефактов в `maven`

# maven-assembly-plugin

## Сборка

Сборка обычно описывается в отдельном файле (**descriptor**)

Сборок может быть несколько

Сборки могут быть компонентными

```
<plugin>
  <artifactId>maven-assembly-plugin</artifactId>
  <version>3.3.0</version>
  <configuration>
    <descriptors>
      <descriptor>src/assembly/src.xml</descriptor>
    </descriptors>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>single</goal>
      </goals>
      <phase>package</phase>
    </execution>
  </executions>
</plugin>
```

# Assembly — descriptor

## <id>

Указывает классификатор артефакта

## <format>

zip, dir, tar, war, ...

## <fileSet>

Включение нескольких файлов

## <dependencySets>

Можно в сборку добавить файлы из другого артефакта

<unpack>

И распаковать их

## <...>

Там много опций

```
<assembly>
  <id>dist</id>
  <formats>
    <format>dir</format>
  </formats>
  <fileSets>
    <fileSet>
      <directory>${basedir}/target/appassembler</directory>
      <includes>
        <include>**/*</include>
      </includes>
      <outputDirectory>/</outputDirectory>
    </fileSet>
    <fileSet>
      <directory>${basedir}/target/site</directory>
      <includes>
        <include>**/*</include>
      </includes>
      <outputDirectory>/doc</outputDirectory>
    </fileSet>
  </fileSets>
  <files>
    <file>
      <source>${basedir}/target/git-status</source>
      <outputDirectory>/</outputDirectory>
    </file>
  </files>
</assembly>
```

# Assembly — target/

## target/

Есть каталог со сборкой, хотя можно и zip

В нем есть

bin/ jars/ - с исполняемыми файлами

doc/ - с документацией и api

src/ - с исходниками

git-commit.txt — информация о версии (будет далее)

```
> tree spring-demo-0.0.1-SNAPSHOT-dist/
spring-demo-0.0.1-SNAPSHOT-dist/
├── spring-demo-0.0.1-SNAPSHOT
│   ├── bin
│   │   ├── springdemo
│   │   └── springdemo.bat
│   ├── doc
│   │   ├── summary.html
│   │   └── ....
│   ├── git-commit.txt
│   ├── git-status
│   ├── jars
│   │   ├── cbuffer-1.1-SNAPSHOT.jar
│   │   ├── ecolls-1.7.jar
│   │   ├── fs-1.1.jar
│   │   └── ....
│   └── src
│       └── spring-demo-0.0.1-SNAPSHOT-sources.jar
```

# exec-maven-plugin

## Запуск shell скриптов

Maven много может, но без shell скриптов не обойтись

Например запрос на внешний REST сервис, или есть уже готовые shell скрипты

Скрипты сборки можно расположить рядом с исходниками

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>3.0.0</version>
  <executions>
    <execution>
      <phase>prepare-package</phase>
      <id>generate-git-id</id>
      <goals>
        <goal>exec</goal>
      </goals>
      <configuration>
        <executable>/usr/bin/bash</executable>
        <arguments>
          <argument>${project.basedir}/src/assembly/
build-step.sh</argument>
          <argument>${project.basedir}/target</
argument>
        </arguments>
      </configuration>
    </execution>
  </executions>
</plugin>
```



# exec-maven-plugin

## Запуск shell скриптов

Maven много может, но без shell скриптов не обойтись

Например, для запуска скриптов в shell. Этот файл будет запущен, для Windows Например cmd.exe

Скрипт должен быть расположен рядом с исходниками

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>3.0.0</version>
  <executions>
    <execution>
      <phase>prepare-package</phase>
      <id>generate-git-id</id>
      <goals>
        <goal>exec</goal>
      </goals>
      <configuration>
        <executable>/usr/bin/bash</executable>
        <arguments>
          <argument>${project.basedir}/src/assembly/
build-step.sh</argument>
          <argument>${project.basedir}/target</
argument>
        </arguments>
      </configuration>
    </execution>
  </executions>
</plugin>
```

# exec-maven-plugin

## Запуск shell скриптов

Maven много может, но без shell скриптов не обойтись

Например, для запуска скриптов в сервисе ST shell

Передаем аргументы  
Первый в данном случае –  
Shell скрипт

Скрипт  
расположить рядом с  
исходниками

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>3.0.0</version>
  <executions>
    <execution>
      <phase>prepare-package</phase>
      <id>generate-git-id</id>
      <goals>
        <goal>exec</goal>
      </goals>
      <configuration>
        <executable>/usr/bin/bash</executable>
        <arguments>
          <argument>${project.basedir}/src/assembly/
build-step.sh</argument>
          <argument>${project.basedir}/target</
argument>
        </arguments>
      </configuration>
    </execution>
  </executions>
</plugin>
```

# exec-maven-plugin

## Запуск shell скриптов

Maven много может, но без shell скриптов не обойтись

Например, можно использовать Maven для запуска shell скриптов. Например, можно использовать Maven для запуска shell скриптов.

Скрипты можно располагать рядом с исходниками

А это его аргументы

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>3.0.0</version>
  <executions>
    <execution>
      <phase>prepare-package</phase>
      <id>generate-git-id</id>
      <goals>
        <goal>exec</goal>
      </goals>
      <configuration>
        <executable>/usr/bin/bash</executable>
        <arguments>
          <argument>${project.basedir}/src/assembly/
            generate-step.sh</argument>
          <argument>${project.basedir}/target</
            argument>
        </arguments>
      </configuration>
    </execution>
  </executions>
</plugin>
```

# exec-maven-plugin

## Build-step.sh

Это просто пример скрипта,  
\${1} — это аргумент который  
передали

В вашем случае это может быть  
скрипт на другом языке,  
например python/java/js/...

```
src/assembly/build-step.sh
```

```
#!/bin/bash
```

```
echo "running build step"
```

```
if [ -d $1 ] ;
```

```
then
```

```
    git status 1>${1}/git-status
```

```
fi
```

# maven-antrun-plugin

## Ant

В maven так же  
МОЖНО  
ИСПОЛЬЗОВАТЬ ant

```
<plugin>
  <artifactId>maven-antrun-plugin</artifactId>
  <version>1.7</version>
  <executions>
    <execution>
      <phase>package</phase>
      <configuration>
        <tasks>
          <tar basedir="target/${project.artifactId}-${
project.version}-dist/${project.artifactId}-${project.version}" destfile="target/${
project.artifactId}-${project.version}.tar" />
          <gzip src="target/${project.artifactId}-${
project.version}.tar" destfile="target/${project.artifactId}-${project.version}.tar.gz"
/>
        </tasks>
      </configuration>
      <goals>
        <goal>run</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# gmavenplus-plugin

## Ant

На нем сложно сделать хорошую программу

## Exec-maven-plugin

Зависит от среды, установленного ПО

## Gmavenplus-plugin

Это поддержка скриптов на groovy в среде maven

Не зависимость от ОС / Среды

Можно подключить дополнительные библиотеки (jar) в процесс сборки

# gmavenplus-plugin

```
<plugin>
  <groupId>org.codehaus.gmavenplus</groupId>
  <artifactId>gmavenplus-plugin</artifactId>
  <version>1.12.0</version>
  <executions>
    <execution>
      <id>execute</id>
      <goals>
        <goal>execute</goal>
      </goals>
      <phase>generate-sources</phase>
    </execution>
  </executions>
  <configuration>
    <scripts>
      <script><![CDATA[
        log.info("The settings are " + session.settings)
        log.info("This session's goals are " + session.goals)
        log.info("The local repository is " + session.localRepository)
        assert ant.project.baseDir == project.baseDir
        assert "$project.name" == "${project.name}"
      ]]></script>
      <script>${project.baseDir}/src/assembly/get-gitcommit.groovy</script>
    </scripts>
  </configuration>
</plugin>
```

Подключаем плагин

```
  <dependencies>
    <dependency>
      <groupId>org.codehaus.groovy</groupId>
      <artifactId>groovy-all</artifactId>
      <version>3.0.6</version>
      <type>pom</type>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.eclipse.jgit</groupId>
      <artifactId>org.eclipse.jgit</artifactId>
      <version>5.10.0.202012080955-r</version>
      <scope>runtime</scope>
    </dependency>
  </dependencies>
</plugin>
```

# gmavenplus-plugin

```
<plugin>
  <groupId>org.codehaus.gmavenplus</groupId>
  <artifactId>gmavenplus-plugin</artifactId>
  <version>1.12.0</version>
  <executions>
    <execution>
      <id>execute</id>
      <goals>
        <goal>execute</goal>
      </goals>
      <phase>generate-sources</phase>
    </execution>
  </executions>
  <configuration>
    <scripts>
      <script><![CDATA[
        log.info("The settings are " + session.settings)
        log.info("This session's goals are " + session.goals)
        log.info("The local repository is " + session.localRepository)
        assert ant.project.baseDir == project.baseDir
        assert "$project.name" == "${project.name}"
      ]]></script>
      <script>${project.baseDir}/src/assembly/get-gitcommit.groovy</script>
    </scripts>
  </configuration>
</plugin>
```

Указываем фазу  
На которой plugin стартует

```
<dependencies>
  <dependency>
    <groupId>org.codehaus.groovy</groupId>
    <artifactId>groovy-all</artifactId>
    <version>3.0.6</version>
    <type>pom</type>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.eclipse.jgit</groupId>
    <artifactId>org.eclipse.jgit</artifactId>
    <version>5.10.0.202012080955-r</version>
    <scope>runtime</scope>
  </dependency>
</dependencies>
</plugin>
```



# gmavenplus-plugin

```
<plugin>
  <groupId>org.codehaus.gmavenplus</groupId>
  <artifactId>gmavenplus-plugin</artifactId>
  <version>1.12.0</version>
  <executions>
    <execution>
      <id>execute</id>
      <goals>
        <goal>execute</goal>
      </goals>
      <phase>generate-sources</phase>
    </execution>
  </executions>
  <configuration>
    <scripts>
      <script><![CDATA[
        log.info("The settings are " + session.settings)
        log.info("This session's goals are " + session.goals)
        log.info("The local repository is " + session.localRepository)
        assert ant.project.baseDir == project.baseDir
        assert "$project.name" == "${project.name}"
      ]]></script>
      <script>${project.baseDir}/src/assembly/get-gitcommit.groovy</script>
    </scripts>
  </configuration>
</plugin>
```

Скрипт можно тут же вставить

```
<dependencies>
  <dependency>
    <groupId>org.codehaus.groovy</groupId>
    <artifactId>groovy-all</artifactId>
    <version>3.0.6</version>
    <type>pom</type>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.eclipse.jgit</groupId>
    <artifactId>org.eclipse.jgit</artifactId>
    <version>5.10.0.202012080955-r</version>
    <scope>runtime</scope>
  </dependency>
</dependencies>
</plugin>
```

# gmavenplus-plugin

```
<plugin>
  <groupId>org.codehaus.gmavenplus</groupId>
  <artifactId>gmavenplus-plugin</artifactId>
  <version>1.12.0</version>
  <executions>
    <execution>
      <id>execute</id>
      <goals>
        <goal>execute</goal>
      </goals>
      <phase>generate-sources</phase>
    </execution>
  </executions>
  <configuration>
    <scripts>
      <script><![CDATA[
        log.info("The settings are " + session.settings)
        log.info("This session's goals are " + session.goals)
        log.info("The local repository is " + session.localRepository)
        assert ant.project.baseDir == project.baseDir
        assert "$project.name" == "${project.name}"
      ]]></script>
      <script>${project.baseDir}/src/assembly/get-gitcommit.groovy</script>
    </scripts>
  </configuration>
```

```
<dependencies>
  <dependency>
    <groupId>org.codehaus.groovy</groupId>
    <artifactId>groovy-all</artifactId>
    <version>3.0.6</version>
    <type>pom</type>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.eclipse.jgit</groupId>
    <artifactId>org.eclipse.jgit</artifactId>
    <version>5.10.0.202012080955-r</version>
    <scope>runtime</scope>
  </dependency>
</dependencies>
</plugin>
```

Либо вынести скрипт  
в отдельный файл

# gmavenplus-plugin

```
<plugin>
  <groupId>org.codehaus.gmavenplus</groupId>
  <artifactId>gmavenplus</artifactId>
  <version>1.0</version>
  <executions>
    <execution>
      <id>eclipse-jgit</id>
      <goals>
        <goal>eclipse-jgit</goal>
      </goals>
      <phase>generate-sources</phase>
    </execution>
  </executions>
  <configuration>
    <scripts>
      <script><![CDATA[
        log.info("The settings are " + session.settings)
        log.info("This session's goals are " + session.goals)
        log.info("The local repository is " + session.localRepository)
        assert ant.project.baseDir == project.baseDir
        assert "$project.name" == "${project.name}"
      ]]></script>
      <script>${project.baseDir}/src/assembly/get-gitcommit.groovy</script>
    </scripts>
  </configuration>
</plugin>
```

Указываем версию groovy  
И область runtime,  
чтоб другие модули не использовали  
groovy в неявном виде (транзитивно)

```
<dependencies>
  <dependency>
    <groupId>org.codehaus.groovy</groupId>
    <artifactId>groovy-all</artifactId>
    <version>3.0.6</version>
    <type>pom</type>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.eclipse.jgit</groupId>
    <artifactId>org.eclipse.jgit</artifactId>
    <version>5.10.0.202012080955-r</version>
    <scope>runtime</scope>
  </dependency>
</dependencies>
</plugin>
```

# gmavenplus-plugin

```
<plugin>
  <groupId>org.codehaus.gmavenplus</groupId>
  <artifactId>gmavenplus-plugin</artifactId>
  <version>1.12.0</version>
  <executions>
    <execution>
      <id>
      <goals>
      <configuration>
    </execution>
  </executions>
  <configuration>
    <scripts>
      <script><![CDATA[
        log.info("The settings are " + session.settings)
        log.info("This session's goals are " + session.goals)
        log.info("The local repository is " + session.localRepository)
        assert ant.project.baseDir == project.baseDir
        assert "$project.name" == "${project.name}"
      ]]></script>
      <script>${project.baseDir}/src/assembly/get-gitcommit.groovy</script>
    </scripts>
  </configuration>
</plugin>
```

Добавляем для скрипта  
Дополнительные jar  
Например для работы с git

```
<dependencies>
  <dependency>
    <groupId>org.codehaus.groovy</groupId>
    <artifactId>groovy-all</artifactId>
    <version>3.0.6</version>
    <type>pom</type>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.eclipse.jgit</groupId>
    <artifactId>org.eclipse.jgit</artifactId>
    <version>5.10.0.202012080955-r</version>
    <scope>runtime</scope>
  </dependency>
</dependencies>
</plugin>
```

# gmavenplus-plugin

## get-gitcommit.groovy

Это скрипт для примера

Переменная `project`  
Это объект `MavenProject`  
Он предоставляется  
при запуске `mvn`

Нашли каталог `.git`  
В дальнейшем  
ведем работу с ним

src/assembly/get-gitcommit.groovy

```
import org.eclipse.jgit.lib.*

try {

    // find .git dir
    File dir = new File(project.basedir.toString()).
        getAbsoluteFile().getCanonicalFile()
    File gitdir = null
    while (true) {
        if (dir.name == '.git') ...
    }

    if (gitdir == null) {
        println "git dir not found"
        return;
    }

    // read git info
    def repo = new RepositoryBuilder().setGitDir(gitdir).build()
```

# gmavenplus-plugin

## get-gitcommit.groovy

Это скрипт для примера

src/assembly/get-gitcommit.groovy

```
// read git info
def repo = new RepositoryBuilder().setGitDir(gitdir).build()

// write commit id, if branch found
if (repo.fullBranch) {
    println "branch $repo.fullBranch"
    def ref = repo.findRef(repo.fullBranch)
    if (ref != null) {
        def cmtId = new StringWriter()
        ref.objectId.copyTo(cmtId)
        println "commit $cmtId"

        def commitTxt = new File(project.basedir, 'target/git-
commit.txt')
        commitTxt.setText("""|branch $repo.fullBranch
|commit $cmtId""").stripMargin()
    }
}
```

Записываем в файл  
target/git-commit.txt  
Текущую ветку git  
И идентификатор commit