

Maven-плагины

JSP & Servlets
1 уровень, 5 лекция

ОТКРЫТА

6.1 Знакомство с плагинами

Стандартные жизненные циклы можно дополнить функционалом с помощью Maven-плагинов. Плагины позволяют вставлять в стандартный цикл новые шаги (например, распределение на сервер приложений) или расширять существующие шаги.

Плагины в Maven не являются чем-то экстраординарным, наоборот, это самая обычная и часто встречающаяся вещь. Ведь если вы хотите задать какие-нибудь нюансы сборки вашего проекта, то вам нужно указать нужную информацию в pom.xml. И единственный способ это сделать – написать “плагин”.

Так как плагины являются такими же артефактами, как и зависимости, то они описываются практически так же. Вместо раздела dependencies – plugins, вместо dependency – plugin, вместо repositories – pluginRepositories, repository – pluginRepository.

Пример:

```
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-checkstyle-plugin</artifactId>
    <version>2.6</version>
  </plugin>
</plugins>
```

Объявление плагина в pom.xml позволяет зафиксировать версию плагина, а также задать ему необходимые параметры, определить различные конфигурационные параметры и привязать к фазам.

Иными словами, Maven запускает определенные плагины, которые выполняют всю работу. То есть, если мы хотим научить Maven особенным сборкам проекта, **то необходимо добавить в pom.xml указание на запуск нужного плагина в нужную фазу и с нужными параметрами.**

Количество доступных плагинов очень велико, есть разнообразные плагины, позволяющие непосредственно из maven запускать web-приложение для тестирования его в браузере, генерировать ресурсы и тому подобное. Главной задачей разработчика в этой ситуации является **найти и применить наиболее подходящий набор плагинов.**

6.2 Жизненный цикл и плагины

Очень часто плагин используется для того, чтобы во время выполнения определенной фазы запустить какую-нибудь консольную утилиту. Более того, мы можем запустить даже обычный Java-класс (у которого есть метод main, конечно).

Пример:

```
<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>exec-maven-plugin</artifactId>
  <version>1.2.1</version>
  <executions>
    <execution>
      <goals>
```

```
        <goal>java</goal>
    </goals>
</execution>
</executions>
<configuration>
    <mainClass>com.example.Main</mainClass>
    <arguments>
        <argument>first-argument</argument>
        <argument>second-argument</argument>
    </arguments>
</configuration>
</plugin>
```

Обычно плагины можно очень гибко настраивать. Все официальные плагины от разработчиков Maven очень хорошо документированы на официальном сайте Maven. Например, для **maven-compiler-plugin** на странице Apache Maven Project можно увидеть перечень всех переменных, управляющих плагином. Информация по плагину доступна [по ссылке](#)

Еще важная информация. Разные плагины вызываются Maven'ом на разных стадиях жизненного цикла. Так проект, описывающей десктопное Java-приложение на swing, имеет стадии жизненного цикла отличные от тех, что характерны для разработки web-приложения (war).

Или, например, когда выполняется команда “mvn test”, инициируется целый набор шагов в жизненном цикле проекта: “process-resources”, “compile”, “process-classes”, “process-test-resources”, “test-compile”, “test”. Упоминания этих фаз вы можете видеть в выводимых Maven-ом сообщениях:

```
[INFO] Scanning for projects...
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources)    @ javarush ---
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile)      @ javarush
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ javarush ---
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ javarush ---
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test)          @ javarush ---
[INFO] Surefire report directory:      t:\ projects\javarush\target\surefire-reports
```

6.3 Цели в Maven – goals

В Maven есть еще такое понятие как цель (goal). goal – это как бы цель запуска Maven’а. Основные цели совпадают с основными фазами:

- **validate;**
- **compile;**
- **test;**
- **package;**
- **verify;**
- **install;**
- **deploy.**

В каждой фазе жизненного цикла проекта вызывается определенный плагин (jar-библиотека), который включает некоторое количество целей (goal)

Например, плагин «maven-compiler-plugin» содержит две цели: compiler:compile для компиляции основного исходного кода проекта и compiler:testCompile для компиляции тестов. Формально, список фаз можно изменять, хотя необходимость в этом бывает редко.

Если тебе нужно выполнить какие-нибудь нестандартные действия в определенной фазе, то всего лишь нужно добавить соответствующий плагин в pom.xml

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>имя-плагина</artifactId>
  <executions>
    <execution>
      <id>customTask</id>
      <phase>generate-sources</phase>
      <goals>
        <goal>pluginGoal</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Самое важное в данном случае – это определить для плагина наименование фазы “execution/phase”, в которую нужно встроить вызов цели плагина “goal”. Например, тебе нужно сгенерировать Java-код на основе xml. Тогда тебе нужна фаза “generate-sources”, она располагается перед вызовом фазы compile и идеально подходит для генерирования части исходных кодов проекта.

+25

Комментарии (6)

популярные

новые

старые

JavaCoder

Введите текст комментария

MineJavaAcc

Уровень 37, Москва, Russian Federation

26 сентября, 22:36

...

У меня дежавю с run level в Linux-подобных системах)

Ответить

0

OB11TO

Backend Developer в Мечтах

16 июня, 18:39

...

Чтобы легче было понять, что такое gloal (цели), просто представляйте обычный java class (Потому-что это так и есть)

Ответить

+4

Винеѣох

Уровень 41, Москва, Россия

EXPERT

11 июня, 02:22

...

Понятия не имею, зачем и каким образом может понадобиться генерировать Java-код из XML, для чего это нужно и почему бы не написать нужный код сразу. И где без этого не обойтись. Надеюсь, когда-нибудь просветление придёт...

Ответить

+5

Dmitry Vezhnin

CEO в JavaRush

13 июня, 10:01

...

Обычно это связано с API в корпоративных системах. Пересылаемые объекты описываются с помощью xsd-схемы, и по этой схеме можно сгенерировать Java-классы. Даже [IDEA умеет генерировать Java классы на основе xsd из коробки.](#)

Ответить

+4

Jh-007

Уровень 47

21 июня, 18:06

...

Есть такой язык - UML для визуального моделирования, он основан на xml. Можно нарисовать диаграмму классов и перегнуть её в код. Была программа - Rational Rose. Я когда учился нам рассказывали. Видимо сейчас это заглохло.

Ответить

+2

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

