5 декабря 2009 в 19:52

Apache Maven — основы

JAVA*

После публикации топика о Maven в комментариях возникли вопросы о том, как начать с ним работать, с чего начать, как составлять файлы pom.xml, откуда брать плагины и т.п. Данный топик будет своего рода getting started или f.a.q.

Терминология

Как в любой системе, в Maven, есть свой набор терминов и понятий.

Вся структура проекта описывается в файле pom.xml (POM – Project Object Model), который должен находиться в корневой папке проекта. Ключевым понятием Maven является **артефакт** — это, по сути, любая библиотека, хранящаяся в репозитории. Это может быть какая-то зависимость или плагин.

Зависимости — это те библиотеки, которые непосредственно используются в вашем проекте для компиляции кода или его тестирования.

Плагины же используются самим Maven'ом при сборке проекта или для каких-то других целей (деплоймент, создание файлов проекта для Eclipse и др.).

В самом начале работы с Maven, пользователь непременно столкнется с таким понятием как архетип. **Архетип** — это некая стандартная компоновка файлов и каталогов в проектах различного рода (веб, swing-проекты и прочие). Другими словами, Maven знает, как обычно строятся проекты и в соответствии с архетипом создает структуру каталогов.

Как правило, название артефакта состоит из названия группы, собственного названия и версии. К примеру Spring будет иметь вот такое название в среде Maven: org.springframework.spring:2.5.5. Последний домен означает всегда artifactId, все, что перед ним – groupId – хорошо это запомните!

На жизненном цикле останавливаться не буду, так как он хорошо описан в вышеобозначенной статье. А теперь перейдем к практике.

Установка Maven

Последнюю версию всегда можно скачать на странице загрузки на официальном сайте. Просто распаковываем архив в любую директорию. Далее необходимо создать переменную в Path, в которой необходимо указать путь к Maven. Заходим в Win + Pause – Дополнительно – Переменные среды – в верхнем окошке нажимаем Создать, вводим имя M2_HOME и значение допустим “C:\apache-maven-2.2.1”. Далее там же создаем еще одну переменную M2 со значением %M2_HOME%\bin. Так же убеждаемся, что есть переменная JAVA_HOME с путем к JDK. Ее значение должно быть примерно таким «c:\Program Files\Java\jdk1.6.0_10\». И наконец в том же окошке

создаем/модифицируем переменную Path, в нее необходимо просто написать %M2%, чтобы наша папочка с исполняемым файлом Maven была видна из командной строки. Теперь необходимо проверить работоспособность нашей установки. Для этого заходим в командную строку и вводим команду

```
mvn –version
```

Должна появиться информация о версиях Maven, jre и операционной системе, что-то вроде:

```
Maven version: 2.2.1
Java version: 1.6.0_10
OS name: "windows 2003" version: "5.2" arch: "x86" Family: "windows"
```

Maven создаст вам локальный репозиторий в вашей личной папке, например в каталоге C:\Documents and Settings\username\.m2\repository

Все, Maven готов к работе, можно приступать к созданию приложения.

Создание приложения из архетипа

На сайте Maven перечислены наиболее популярные архетипы для приложений, но вы можете легко создать свой или найти более специфичный например здесь.

Итак, допустим нас интересует веб-приложение – находим подходящий архетип, называется он maven-archetype-webapp. В командной строке, в необходимом каталоге выполняем команду Maven:

```
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=my-webapp
   -DarchetypeArtifactId=maven-archetype-webapp
```

Теперь мы можем лицезреть довольно наглядную структуру каталогов с говорящими названиями java – здесь будут ваши классы, webapp – здесь размещаются странички веб-приложения, resources – различного рода ресурсы в classpath (файлы конфигурации, например), test – юнит-тесты, соответственно и т.п.

Сборка проекта

Здесь все просто – выполняем команду

```
mvn package
```

или

```
mvn install
```

в корневом каталоге приложения, там, где находится файл pom.xml. Первая команда скомпилирует ваш проект и поместит его в папку target, а вторая еще и положит его к вам в локальный репозиторий.

Есть полезная функция, наподобие конвеера, то есть можно написать

```
mvn clean install
```

и Maven сначала очистит папку target проекта, потом соберет его и положит в репозиторий.

Минимальный набор действий для работы Maven мы изучили, теперь переходим к кастомизации и добавлению зависимостей проекта.

Зависимости и репозитории

Как правило, большинство популярных библиотек находятся в центральном репозитории, поэтому их можно прописывать сразу в раздел dependencies вашего pom-файла. Например чтобы подключить Spring Framework необходимо определить следующую зависимость:

```
<dependencies>
   ...
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring</artifactId>
      <version>2.5.5</version>
   </dependency>
</dependencies>
```

Версию хотя и можно не указывать и тогда Maven возьмет последний вариант, но я вам лично советую это делать, потому как у нас неоднократно бывали случаи, что проект просто в один момент переставал собираться или начинал падать в совершенно неожиданных и хорошо оттестированных местах, хотя вроде бы никто ничего не менял.

Специфические вещи обычно не находятся в центральном репозитории и тогда вам придется указать репозиторий производителя вручную. Для примера добавим зависимость JSF-фреймворка ajax-компонентов JBoss RichFaces.

С зависимостями все просто:

```
<dependencies>
   <dependency>
      <groupId>org.richfaces.ui</groupId>
      <artifactId>richfaces-ui</artifactId>
      <version>3.3.1.GA</version>
   </dependency>
</dependencies>
```

А вот репозиторий JBoss теперь необходимо прописать ручками либо в файле settings.xml, который лежит в корне вашего локального репозитория, либо в самом файле pom.xml, в зависимости от того, нужен ли данный репозиторий во всех проектах, либо в каком-то одном конкретном, соответственно:

```
<!-- JBoss RichFaces Repository -->
<repositories>
   <repository>
      <releases>
         <enabled>true</enabled>
      </releases>
      <snapshots>
         <enabled>false</enabled>
         <updatePolicy>never</updatePolicy>
      </snapshots>
      <id>repository.jboss.com</id>
      <name>Jboss Repository for Maven</name>
      <url>
         http://repository.jboss.com/maven2/
      </url>
      <layout>default</layout>
   </repository>
</repositories>
```

Как правило на сайтах крупных проектов пишут всю информацию, необходимую для встраивания их библиотеки в проект на основе Maven, но бывают случаи, когда

артефакт приходится искать очень и очень долго. Здесь нам очень сильно может помочь MVNrepository.com — он вам всегда подскажет где может находиться искомая библиотечка. Но если уж и там не нашлось, то из собственного опыта могу посоветовать гуглить «<*название библиотеки*> pom.xml». Бывает так, что проекты уже давно закрыты и в репозитории не положены потому что разработчики уже не заботятся об их распространении. Тогда остается один единственный способ – добавить файл в репозиторий вручную командой:

```
mvn install:install-file
  -Dfile=<path-to-file>
  -DgroupId=<group-id>
  -DartifactId=<artifact-id>
  -Dversion=<version>
  -Dpackaging=<packaging>
```

Последний параметр чаще всего имеет значение jar.

Плагины

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

Плагинами Maven делает все, даже непосредственно то, для чего он затевался – сборку проекта, только этот плагин необязательно указывать в свойствах проекта, если вы не хотите добавить какие-то фичи.

Посмотрим как настроить плагин для создания проекта для Eclipse с использованием WTP ver. 2.0. В раздел plugins нашего pom.xml прописываем следующий плагин:

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-eclipse-plugin</artifactId>
   <configuration>
      <wtpversion>2.0</wtpversion>
   </configuration>
</plugin>
```

Теперь идем опять таки в командную строку и выполняем команду

```
mvn eclipse:eclipse
```

Ждем пока Maven найдет все библиотеки в репозитории или скачает их и вуаля –

теперь наш Maven-проект можно открыть как проект eclipse. При этом библиотеки никуда не копируются как при классическом подходе, а остаются в репозитории и Eclipse делает на них ссылку через свои переменные.

Единого списка всех плагинов естественно не существует, на официальном сайте только есть поддерживаемые плагины непосредственно разработчиками Maven. Однако хотелось бы отметить, что названия плагинов довольно прямолинейны и сделав поиск по ключевым словам «maven tomcat plugin» вы скорее всего обнаружите первой ссылкой плагин для деплоймента проекта в Tomcat.

Собственный репозиторий

К сожалению сам не имею большого опыта настройки репозитория, но могу посоветовать как наиболее простой и распространенный Nexus. За дополнительной информацией следует обратиться на сайт данного проекта.

Однако нельзя оставить без внимания и достойных конкурентов в лице Artifactory и Archiva.

Заключение

Очень надеюсь, что цель данной статьи достигнута – минимальных знаний по Maven должно хватить на работу с не очень большими проектами. Для более серьезных целей очень советую детально изучить maven-compiler-plugin и maven-resource-plugin – они напрямую отвечают за конечную компоновку проекта. Я считаю, что самым главным моментом в обучении Maven является понимание идеологии – Maven описывает конечную структуру проекта, а не пути к ее достижению, в отличие от Ant.

Полезные ссылки
Официальная страница Maven
Документация
Центральный репозиторий
Репозиторий iBiblio
Поиск артефактов по названию
Неплохой форум по Maven
Maven: The Definitive Guide — хорошая книга по Maven