Table of Contents

Вопросы и ответы по Maven

maven, java

Из каких фаз состоит жизненный цикл сборки Clean?

Жизненный цикл сборки **Clean** состоит из следующих этапов:

- **pre-clean**
- **clean**
- **post-clean**

Из каких фаз состоит жизненный цикл сборки Default (Build)?

Default (Build) - это основной жизненный цикл Maven, который используется для сборки проектов. Он включает в себя следующие фазы:

- **validate** - проверяет корректность метаинформации о проекте, подтверждает, является ли проект корректным и вся ли необходимая информация доступа для завершения процесса сборки.
- **initialize** - инициализирует состояние сборки, например, различные настройки.
- **generate-sources** - включает любой исходный код в фазу компиляции.
- **process-sources** - обрабатывает исходный код (подготавливает). Например, фильтрует определённые значения.
- **generate-resources** - генерирует ресурсы, которые должны быть включены в пакет.
- **process-resources** - копирует и отправляет ресурсы в указанную директори. Это фаза перед упаковкой.
- **compile** - комплирует исходный код проекта.
- **process-classes** - обработка файлов, полученных в результате компляции. Например, оптимизация байт-кода Java классов.
- **generate-test-sources** - генерирует любые тестовые ресурсы, которые должны быть включены в фазу компиляции.
- **process-test-sources** - обрабатывает исходный код тестов. Например, фильтрует значения.
- **test-compile** - компилирует исходный код тестов в указанную директорию тестов.
- **process-test-classes** - обрабатывает файлы, полученные в результате компиляции исходного кода тестов.
- **test** - запускает тесты классов, используя приемлемый фреймворк юнит-тестирования (например, Junit).
- **prepare-package** - выполняет все необходимые операции для подготовки пакета, непосредственно перед упаковкой.
- **package** - преобразует скомпилированный код и пакет в дистрибутивный формат. Такие как JAR, WAR или EAR.
- **pre-integration-test** - выполняет необходимые действия перед выполнением интеграционных тестов.
- **integration-test** - обрабатывает и распаковывает пакет, если необходимо, в среду, где

будут выполняться интеграционные тесты.
- **post-integration-test** - выполняет действия, необходимые после выполнения интеграционных тестов. Например, освобождение ресурсов.
- **verify** - выполняет любые проверки для подтверждения того, что пакет пригоден и отвечает критериям качества.
- **install** - переносит пакет в локальный репозиторий, откуда он будет доступен для использования как зависимость в других проектах.
- **deploy** - копирует финальный пакет (архив) в удалённый репозиторий для, того, чтобы сделать его доступным другим разработчикам и проектам.

Здесь также необходимо уточнить два момента:

- Когда мы выполняем команду Maven, например `install`, то будут выполнены фазы до `install` и фаза `install`.
- Различные задачи Maven будут привязаны к различным фазам жизненного цикла Maven в зависимости от типа архива (JAR/WAR/EAR).

Из каких фаз состоит жизненный цикл сборки Site?

Жизненный цикл сборки **Site** состоит из следующих этапов:

- **pre-site**
- **site**
- **post-site**
- **site-deploy**

Что сделает команда "mvn clean dependency:copy-dependencies package"?

Порядок выполнения зависит от порядка вызова целей и фаз. Рассмотрим данную команду:

```
mvn clean dependency:copy-dependencies package
```

Аргументы `clean` и `package` являются фазами сборки до тех пор, пока “`dependency:copy-dependencies`” является задачей. В этом случае, сначала будет выполнена фаза clean, после этого будет выполнена задача “`dependency:copy-dependencies`”. После чего будет выполнена фаза `package`.

Что такое профиль сборки (Build Profile)?

Профиль сборки - это множество настроек, которые могут быть использованы для установки или перезаписи стандартных значений сборки Maven.

Используя профиль сборки Maven, мы можем настраивать сборку для различных окружений, таких как Development или Production.

Профили настраиваются в файле `pom.xml` с помощью элементов `activeProfiles / profiles` и запускаются различными методами.

Какие типы профилей сборки (Build Profiles) вы знаете?

В Maven существует три основных **типа профилей сборки**:

- **Per Project** - определяется в POM файле, `pom.xml`
- **Per User** - определяется в настройках Maven - xml файл (`%USER_HOME%/.m2/settings.xml`).
- **Global** - определяется в глобальных настройках - xml файл (`%M2_HOME%/conf/settings.xml`).

Как вы можете активировать профили сборки?

Профиль сборки Maven может быть активирован различными способами:

- использованием команды в консоли
- с помощью настроек Maven
- с помощью переменных окружения
- в настройках ОС
- существующими или отсутствующими файлами

Для чего используются Maven плагины?

Maven плагины используются для:

- создания jar–файла
- создания war–файла
- компиляции кода файлов
- юнит-тестирования кода
- создания отчётов проекта
- создания документации проекта

Какие типы плагинов существуют в Maven?

В Maven существует два типа плагинов:

- **Плагины сборки (Build plugins)** - выполняются в процессе сборки и должны быть конфигурированны внутри блока

  ```
  <build></build>
  ```

  файла `pom.xml`.

- **Плагины отчётов (Reporting plugins)** - выполняются в процессе генерирования сайта и должны быть сконфигурированны внутри блока

  ```
  <reporting></reporting>
  ```

  файла `pom.xml`.

Когда Maven использует внешние зависимости?

Если необходимые файлы не найдены ни в центральном, ни на удалённом репозитории, тогда для решения этой проблемы используются внешние зависимости.

Что нужно определить для внешней зависимости?

Внешние зависимости могут быть сконфигурированы в файле `pom.xml` таким же образом, как и другие зависимости, для этого нужно:

- определить `groupId` таким же именем, как и имя файла
- определить `artifactId` таким же именем, как и имя файла
- определить область видимости зависимости как `system`
- указать абсолютный путь к файлу

Какая команда создает новый проект на основе архетипа?

Переходим в нужную нам директорию и выполняем в терминале следующую команду:

```
mvn archetype:generate
```

Что такое SNAPSHOT в Maven?

SNAPSHOT - это специальная версия, которая показывает текущую рабочую копию. При каждой сборке Maven проверяет наличие новой **snapshot** версии на удалённом репозитории.

В чем разница между snapshot и версией?

В случае с **обычной версией**, если Maven однажды загрузил версию `data-service:1.0`, то он больше не будет пытаться загрузить новую версию 1.0 из репозитория. Для того, чтобы скачать обновлённый продукт `data-service` должен быть обновлён до версии 1.1.

В случае со **snapshot**, Maven автоматически будет подтягивать крайний snapshot (`data-`

`service:1.0-SNAPSHOT`) каждый раз, когда будет выполнятся сборка проекта.

Что такое транзитивная зависимость в Maven?

Транзитивная зависимость - позволяет избегать необходимости изучать и указывать библиотеки, которые требуются для самой зависимости, и включает их автоматически. Необходимые библиотеки подгружаются в проект автоматически. При разрешении конфликта версий используется принцип «ближайшей» зависимости, то есть выбирается зависимость, путь к которой через список зависимых проектов является наиболее коротким.

Как Maven определяет, какую версию зависимостей использовать, когда встречается множественный вариант выбора?

Dependency mediation - определяет, какая версия зависимости будет использоваться, когда встречается несколько версий артефактов. Если две версии зависимости на той же глубине в дереве зависимостей, то будет использоваться та которая объявлена первой. Здесь важен порядок объявления: первое объявление выигрывает.

Что такое область видимости зависимостей (dependency scope)? Назовите значения dependency scope.

Существуют следующие **области видимости зависимостей**:

- **compile** - это область по умолчанию, используется, если ничего больше не определено. `Compile` зависимости доступны во всех `classpath` проекта.
- **provided** - это очень похоже на `compile`, но указывает на то, что вы ожидаете от JDK или контейнера предоставить зависимость в ходе выполнения. Эта область доступна только на `compilation` и `test classpath` и не является транзитивной.
- **runtime** - эта область указывает на то, что зависимость не обязательна для `compilation`, но для фаз выполнения.
- **test** - эта область указывает, что зависимость не обязательна для нормального использования приложения.
- **system** - эта область похожа на `provided` за исключением того, что вы предоставляете JAR. Артефакт всегда доступен и не смотрит в репозиторий.
- **import** - эта область используется в зависимости типа `pom` в `<dependencyManagement>` разделе. Это указывает на то, что определенный POM будет заменен зависимостями в этом POM `<dependencyManagement>` разделе.

Какой минимальный набор информации нужен для

составления ссылки зависимостей в разделе dependencyManagement?

Минимальный набор информации такой: `{groupId, artifactId, type, classifier}`.

Как сослаться на свойство(property) определенное в файле pom.xml?

На все свойства в `pom.xml`, можно сослаться с помощью префиксов “**project.**” или “**pom.**” Ниже приведён пример некоторых часто используемых элементов:

- `${project.build.directory}` - “target” директория, или тоже самое `${pom.project.build.directory}`
- `${project.build.outputDirectory}` - путь к директории куда компилятор складывает файлы, по умолчанию “`target/classes`”
- `${project.name}` или `${pom.name}` - имя проекта
- `${project.version}` или `${pom.version}` - версия проекта

Для чего нужен элемент <execution> в POM файле?

Элемент **<execution>** содержит информацию, необходимую для выполнения плагина.

Каким образом можно исключить зависимость в Maven?

Файл описания проекта предусматривает возможность исключить зависимость в случае обнаружения цикличности или отсутствия необходимости в определённой библиотеке. Зависимость может быть исключена используя элемент `exclusion`.

Что является полным именем артефакта?

```
<groupId>:<artifactId>:<version>
```

Если вы не определяете никакой информации, откуда ваш POM унаследует её?

Все POM-ы наследуются от родителя, несмотря на то, определен ли он явно или нет. Это базовый POM известный как “супер POM”, он содержит значения, которые наследуются по умолчанию.

При сборке проекта Maven постоянно проверяет наличие обновлений в интернете. Можете ли вы собрать проект без интернета?

Да, можете, если в вашем локальном репозитории есть все необходимые для сборки артефакты.

Если при сборке проекта в тестах произошла ошибка, как собрать проект без запуска тестов?

Для запуска сборки без выполнения тестов добавьте `-Dmaven.test.skip=true` к команде в строке запуска maven:

```
mvn install -Dmaven.test.skip=true
```

Как запустить только один тест?

Для запуска только одного теста добавьте `-Dtest=[Имя класса]` к команде в строке запуска maven. Например:

```
mvn install -Dtest=org.apache.maven.utils.ConverterTest
```

Как остановить распространение наследования плагинов для дочерних POM?

Установить `<inherited>` в **false**.

Какие теги pom.xml вы знаете?

Вот некоторые из них:

- **project** - описывает проект, это элемент верхнего уровня во всех файлах `pom.xml`
- **groupId** - по-сути, это имя пакета. Полностью отражается в структуре каталогов
- **artifactId** - название проекта. В структуре каталогов не отображается
- **version** - версия проекта
- **packaging** - определяет, какой тип файла будет собран. Варианты: `pom, jar, war, ear`
- **dependencies** - указываются зависимости
- **build** - информация о сборке проекта
- **name** - это уже необязательное описание проекта. В данном случае его название

- **description** - элемент представляет собой общее описание проекта. Это часто используется в генерации документации Maven
- **url** - интернет-страница проекта
- **repositories** - репозитории для артефактов
- **pluginRepositories** - репозитории для плагинов Maven

См. также

- Заметки по Java