1. Что такое Maven? Для чего создан Maven? Как узнать какую версию Maven вы используете?

[Apache Maven](https://maven.apache.org/) - это Java фреймворк для автоматизации сборки проектов, компиляции, создания jar и генерации документации.

Если собирать большие проекты с командной строки, то команда для сборки будет очень длинной, поэтому её иногда записывают в bat/sh скрипт. Но такие скрипты зависят от платформы. Для того чтобы избавиться от этой зависимости и упростить написание скрипта, используют инструменты для сборки проекта.

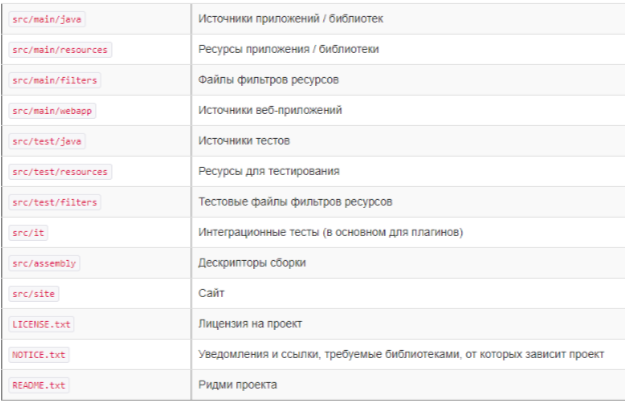
Maven обеспечивает декларативную, а не императивную сборку проекта. То есть, в файлах проекта pom.xml содержится его описание, а не отдельные команды. Все задачи по обработке файлов в Maven выполняется через плагины.

**Maven** — инструмент для управления и сборки проектов — настоящий помощник Java-программиста. Он облегчает жизнь девелоперу на всех стадиях работы: от создания структуры проекта и подключения необходимых библиотек до развертывания продукта на сервере. При работе с любым фреймворком придется использовать Maven.

С помощью mvn --v

2. Какая структура каталогов в Maven?

В Maven [стандартная структура каталогов](https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html), благодаря ей отпадает необходимость прописывать пути к файлам проекта. В корневом каталоге проекта находится pom.xml и несколько текстовых файлов. Всё остальное хозяйство аккуратно разложено в подкаталогах. Главные из них - **src** и **target**. Однако, порядок сохраняется и вглубь:



**3. Что такое pom.xml? Какую информацию содержит pom.xml?**

pom.xml - это XML-файл, который содержит информацию о конфигурации и деталях проекта, используемых при создании проекта на Maven. Он всегда находится в базовом каталоге проекта. Этот файл также содержит описание задач, список и параметры плагинов.

Во время выполнения задач, Maven ищет файл pom.xml в базовой директории проекта. Он читает его и получает необходимую информацию, после чего выполняет задачи.

Корневой элемент <project> содержит ссылку на схему XML, которая облегчает редактирование и проверку pom.xml:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

**4. Что такое супер POM?**

**Супер POM** обеспечивает конфигурацию по умолчанию, например, стандартная структура каталогов, используемые по умолчанию плагины, привязка к фазам жизненного цикла и прочее.

**5. Какие элементы необходимы для минимального POM?**

Обязательные элементы для минимального POM это корневой элемент, modelVersion, GroupID, artifactID и версия. Минимальный POM файл:

<project>

<modelVersion>4.0.0</modelVersion>

<groupId>org.codehaus.mojo</groupId>

<artifactId>my-project</artifactId>

<version>1.0</version>

</project>

**6. Что такое артефакт? Что является полным именем артефакта?**

Артефакт - это элемент, который проект может использовать или создавать. В терминологии Maven **артефакт** **- это вывод, созданный после сборки проекта Maven.**Это может быть, например, jar, war или любой другой исполняемый файл.

Кроме того, артефакты Maven включают пять ключевых элементов: groupId, artifactId, version, packaging и classifier . Это элементы, которые мы используем для идентификации артефакта и известны как координаты Maven .

**Полное** **имя** **артефакта** (координата) представляет четыре слова, разделенные знаком двоеточия в следующем порядке groupId:artifactId:packaging:version.

**7. Что такое зависимости в Maven? Что такое транзитивная зависимость в Maven?**

**Зависимость (dependency)** - это те библиотеки, которые непосредственно используются в вашем проекте для компиляции кода или его тестирования.

**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.

**8. Что такое плагин в Maven?**

**Плагин (plugin)** - это зависимости Maven'а, расширяющие его функционал.

**9. Что такое задача в Maven?**

**Задача (goal)** - это специальная задача, которая относится к сборке проекта и его управлению. Она может привязываться как к нескольким фазам, так и ни к одной. Задача, которая не привязана ни к одной фазе, может быть запущена вне фаз сборки с помощью прямого вызова.

**10.Что такое репозиторий в Maven? Какие типы репозитория существуют в Maven?**

**репозитарий (repository)** - глобальное хранилище всех библиотек, доступных для Maven, это место где хранятся артефакты: jar файлы, pom-файлы, javadoc, исходники, плагины.

В Maven существуют три типа репозитариев:

* **Локальный (local) репозитарий** - это директория, которая хранится на нашем компьютере. Она создаётся в момент первого выполнения любой команды Maven. По умолчанию она расположена в <home директория>/.m2/repository - персональная для каждого пользователя. Фактически она является кэшем для центрального и удалённого репозитариев.
* **Центральный (central) репозитарий** - это репозитарий, который обеспечивается сообществом Maven. Он содержит огромное количество часто используемых библиотек. Расположен по адресу <http://repo1.maven.org/maven2/> и доступен на чтение для всех пользователей в интернете. Если Maven не может найти зависимости в локальном репозитарии, то автоматически начинается поиск необходимых файлов в центральном репозитарии. Для поиска по центральному репозитарию можно использовать сайт <https://mvnrepository.com/>
* **Удалённый (remote) репозитарий**. Иногда Maven не может найти необходимые зависимости в центральном репозитарии, например при отсутствии интернета. В этом случае процесс сборки прерывается, и в консоль выводится сообщение об ошибке. Для того, чтобы предотвратить подобную ситуацию, в Maven предусмотрен механизм **удалённого репозитария**, который является репозитарием, определённым самим разработчиком. Там могут храниться все необходимые зависимости.

**11.Какой порядок поиска зависимостей Maven?**

Когда мы выполняем собрку проекта в Maven, автоматически начинается поиск необходимых зависимостей в следующем порядке:

1. Поиск зависимостей в локальном репозитарии. Если зависимости не обнаружены, происходит переход к шагу 2.
2. Поиск зависимостей в центральном репозитарии. Если они не обнаружены и удалённый репозитарий определён, то происходит переход к шагу 4.
3. Если удалённый репозитарий не определён, то процесс сборки прекращается и выводится сообщение об ошибке.
4. Поиск зависимостей на удалённом репозитарии, если они найдены, то происходит их загрузка в локальный репозитарий, если нет - выводится сообщение об ошибке.

**12.Назовите основные фазы жизненного цикла сборки Maven?**

В Maven есть следующие три стандартных жизненных цикла:

* **Очистка (clean)** - очищает артефакты, созданные до сборки.
* **Сборка (default or build)** - используется для создания приложения.
* **Создание сайта проекта (site)** - генерирует документацию сайта для проекта.

**13.Что делает команда mvn site? Что делает команда mvn clean?**

mvn site - создает веб-сайт проекта

mvn clean - эта команда очищает целевую директорию от созданных в процессе сборки файлов

14.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 репозитарий, откуда он будет доступен для использования как зависимость в других проектах.
* **deploy** - копирует финальный пакет (архив) в удалённый репозитарий 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).

**15.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.

**16.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.

**17.Какая команда создает новый проект на основе архетипа?**

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

mvn archetype:generate

**18.Перечислите теги pom.xml.**

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 необязательое 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** - репозитарии для артефактов
* **pluginRepositories** - репозитарии для плагинов Maven