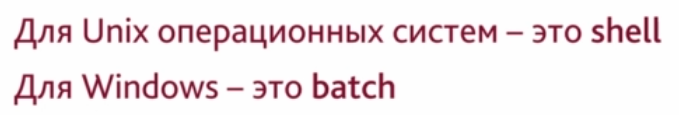
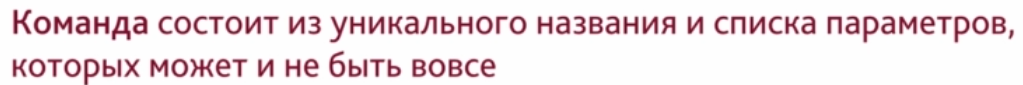


Команды в PowerShell называются командлеты.









Пример:



“~” – в командах обозначает директорию текущего пользователя

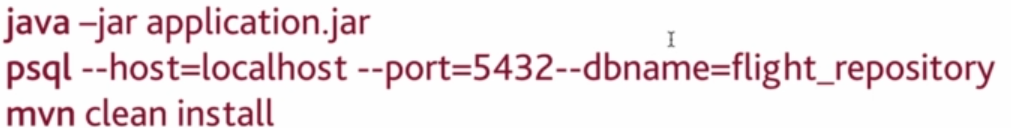
cat first.json – отобразить содержимое файла first.json на консоли

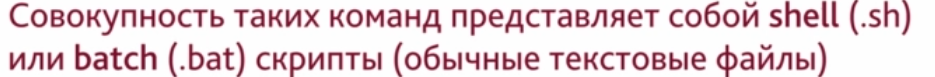
 - скопировать из директории текущего пользователя папки Download файл apache-maven.zip в директорию текущего пользователя в папку maven.

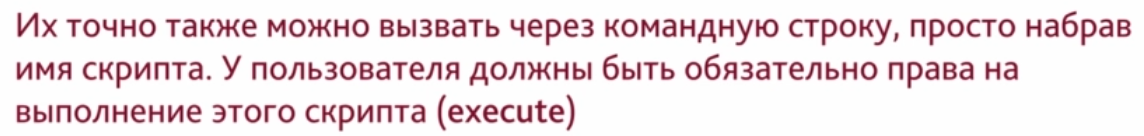
 - меняем текущую директорию на Work/test-project

Это стандартные команды, которые отличаются от того это shell или batch.

Но есть команды и для установленных приложений

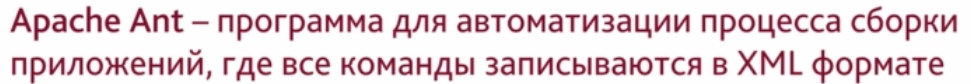


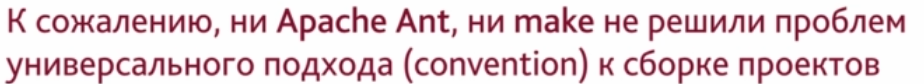


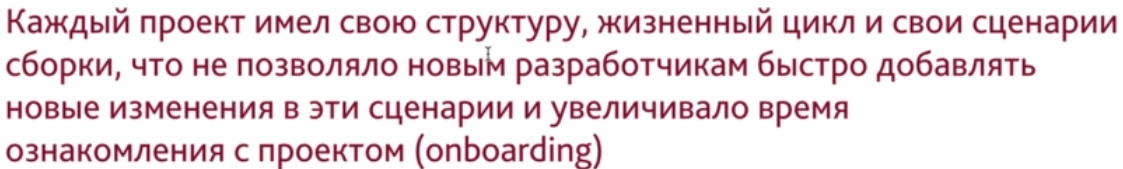


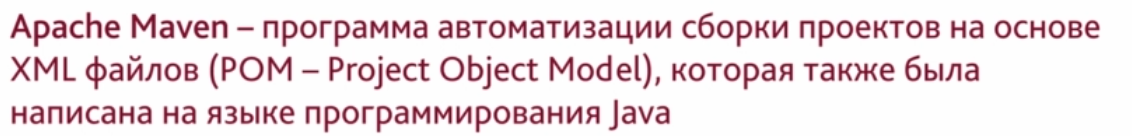
Обычно, чтобы выполнить какой-либо скрипт вызвают файл из командной строки, в котором указаны несколько таких команд.

Shell скрипты используются только для linux-систем, batch – для windows.

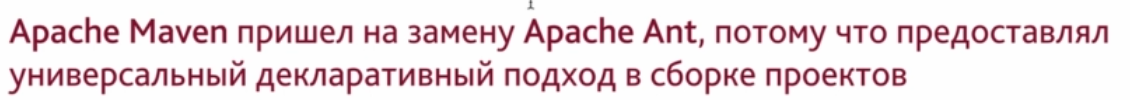


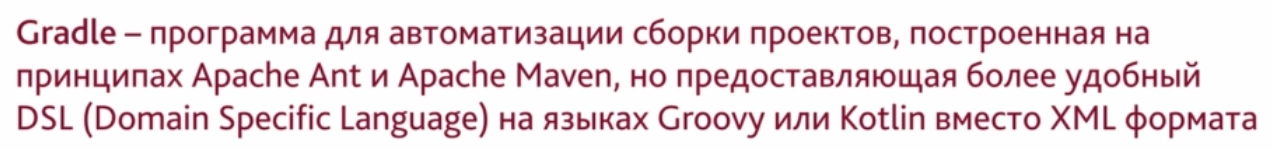














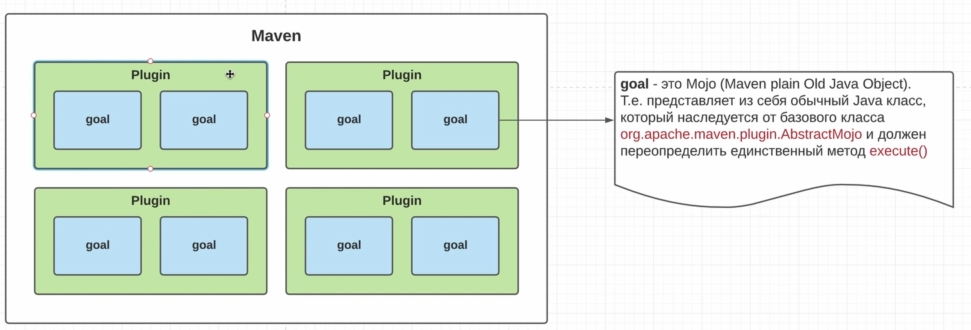
Maven – совокупность плагинов, плагин – основная часть как ядро(core) и при общении с maven мы общаемся с этими плагинами.

core – ядро maven

goal – цели

Плагин – java-проект состоящий из набора классов, и главные из этих классов это goal-ы, от 1 до нескольких. При этом каждый плагин содержит обязательно одну цель это help(служит для описания плагина)

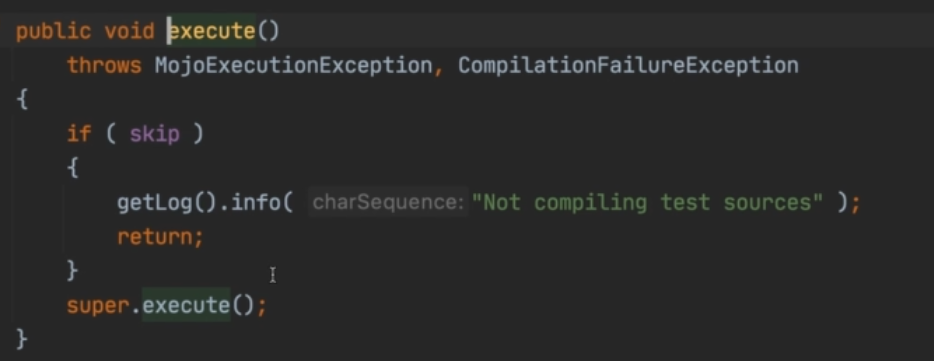
Кроме того, goal-ы в maven называются MOJO



Mojo java класс который наследуется от AbstractMojo, который в свою очередь реализует интерфейс Mojo c методами execute(), setLog() и getLog()



Пример переопределения метода execute()



А реализация метода execute() это и есть то, что выполняет goal в плагине.

При работе с плагинами вызывается тот или иной goal или их совокупность.

В core содержаться плагины которые всегда используются в java-приложениях:

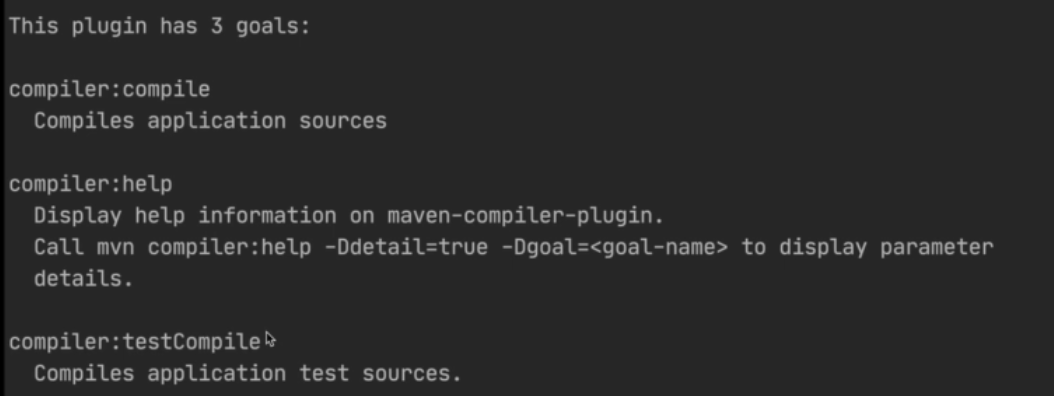
* Clean
* Compiler
* Deploy
* Install
* Resources
* Site
* Verifier
* Surefire

Чтобы набрать команду для maven пишем

mvn – основная команда, далее идет набор параметров

Чтобы обратиться к плагину нужно набрать его имя compiler и далее после двоеточия набираем цель(goal), которая нас интересует

**mvn compiler: help** - таким образом вызывается необходимый goal в нужном плагине



Как видно из рисунка выше, во время запуска нашего приложения мы можем дополнительно передавать какие-то параметры через -D, это характерно для любых java приложений. Такие параметры называются ещё JVM-аргументы. Есть также и обычные аргументы которые принимаются в виде массива строк в нашем методе **main(String[] args)**

Создадим **java-**приложение выводящее на консоль Hello world.

Зайдем в конфигурацию ранера Edit configuration и здесь видно, что есть два способа передать наши аргументы

* VM options – это с -D
* Program arguments – то, что мы принимаем в качестве массива строк в методе main

И то и другое мы можем передавать через командную строку, (на примере плагина compiler выше)

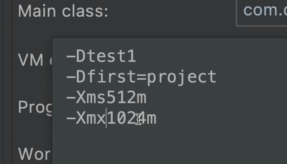
Добавим JVM options – кликаем в строку – в развернувшемся окне пишем

(этот вид аргументов делится на 3 типа:

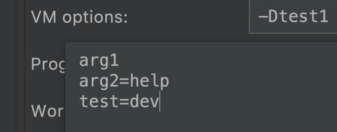
1. -D Format: -Dkey=value это user jvm args и могут принимать любые значения, это то что мы сами придумаем. Разделяются они друг от друга пробелом или переносятся на новую строку
2. Зарезервированные jvm options, начинаются с -Х, мы их не можем придумывать сами, они есть уже существующие и их можно увидеть в документации. Так например можно задать размер нашего heap и максимальный его размер, который он может принимать во время работы

-Xms=512m -Xmx=1024m

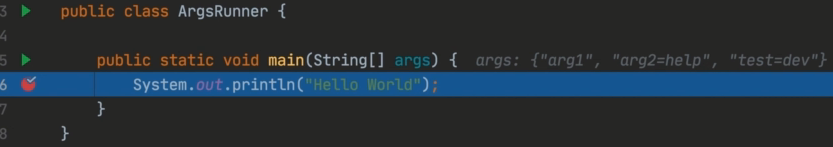
1. -XX для конфигурации компилятора или garbage коллектор, метаспэйсов и пр. и они также зарезервированы и мы не придумываем их сами.



Programm arguments – это аргументы, которые мы можем принимать во время запуска java-приложения в качестве массива строк в методе main (они также могут принимать любые значения, также перечисляются через пробел и именуются как угодно так как не зарезервированы)

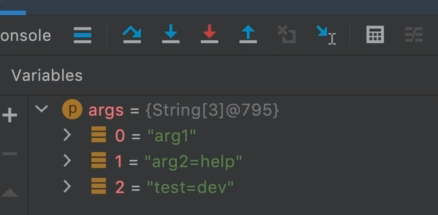


Запускаем приложение в режиме дебаг и попытаемся найти заданные аргумены.

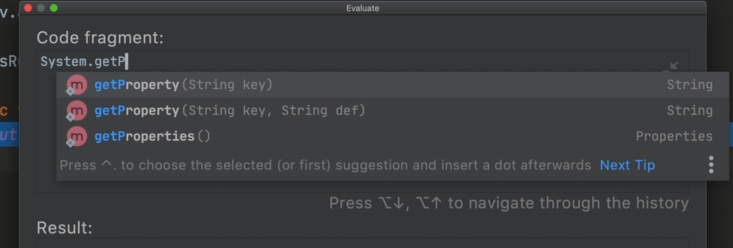


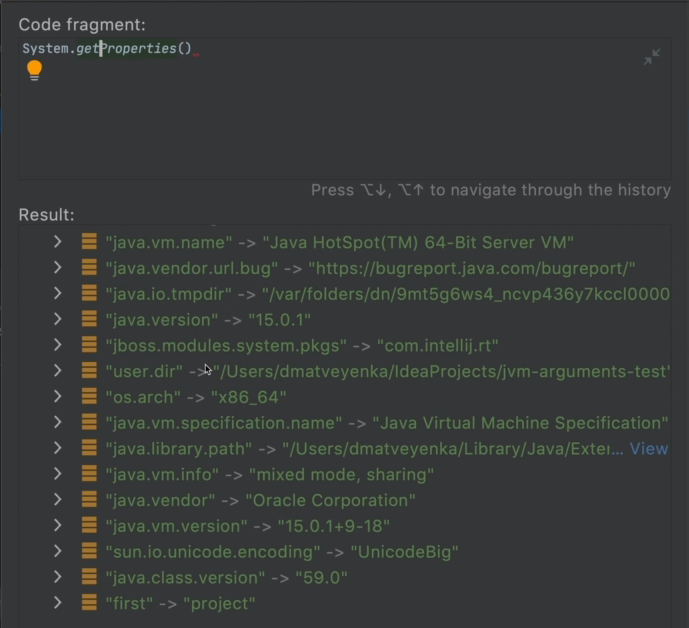
Важное замечание: аргументы args[] метода main не имеют формата в отличие от пользовательских JVM –D

Также знак «=» является пользовательским сепараторам аргумента, он может быть любой а может и не быть его вообще как в первом аргументе.



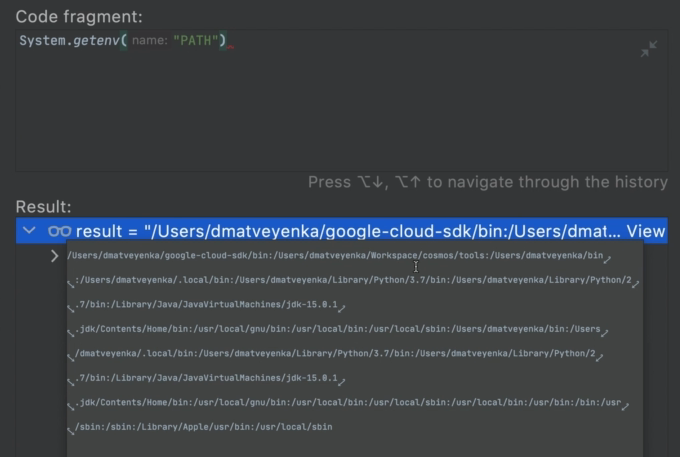
Нас интересуют jvm options. Чтобы их получить есть спец класс System (alt + F8) и по ключу можно получить эти аргументы или все сразу.





Аргументы которые заданы через -Х -ХХ мы получить не можем, потому как мы их использовать не должны, так как они используются нашей JVM

Также через System можно получить environment переменные, то что мы задаем в нашем компьютере(нашей операционной системе). Так мы например можем получить переменную PATH:



Задание таким образом аргументов используется, чтобы настраивать процесс запуска и конфигурации нашего проекта.

Чтобы передавать пользовательские значения настраиваются VM Options со значком -Dключ=значение,

Чтобы передавать системные значения, они именуются -Xзначение, -XXзначение, но получить на стороне приложения мы эти значения не можем, потому что они нами не могут использоваться.

Также можно передавать Program arguments во время запуска приложения

А через класс System можем получать JVM options пользовательские или environment