

Maven LifeCycle

Жизненный цикл Maven представляет собой (абстрактную) концепцию, охватывающую все **шаги** (или лучше: все шаги, которые разработчики Maven решили поддержать), **которые, как ожидается, появятся в процессе разработки проекта**. Эти шаги (или этапы) **называются фазами** в терминологии Maven. То есть под выполнением жизненного цикла maven имеется в виду последовательное выполнение определенных фаз. Выполнение фазы означает выполнение всех предыдущих фаз.

Существует три встроенных жизненных цикла сборки:

- default
- clean
- site

Clean Lifecycle Phases	Site Lifecycle
1. pre-clean	1. pre-site
2. clean	2. site
3. post-clean	3. post-site
	4. site-deploy

Default Lifecycle	
1. validate	13. test-compile
2. initialize	14. process-test-classes
3. generate-sources	15. test
4. process-sources	16. prepare-package
5. generate-resources	17. package
6. process-resources	18. pre-integration-test
7. compile	19. integration-test
8. process-classes	20. post-integration-test
9. generate-test-sources	21. verify
10. process-test-sources	22. install
11. generate-test-resources	23. deploy
12. process-test-resources	

Для того, чтобы maven предоставил вам описание фазы выполните команду

```
>mvn help:describe -Dcmd=phase
```

Например,

```
>mvn help:describe -Dcmd=site
>mvn help:describe -Dcmd=clean
```

```
Markers Properties Servers Data Source Explorer Snippets Terminal
D:\Workspaces\MavenExamples\maven-plugins-example>mvn help:describe -Dcmd=clean
[INFO] Scanning for projects...
[INFO] -----< by.itac.mavenex:maven-plugins-example >-----
[INFO] Building maven-plugins-example 0.0.1
[INFO] -----[ jar ]-----
[INFO] --- maven-help-plugin:3.1.1:describe (default-cli) @ maven-plugins-example ---
[INFO] 'clean' is a phase within the 'clean' lifecycle, which has the following phases:
* pre-clean: Not defined
* clean: org.apache.maven.plugins:maven-clean-plugin:2.5:clean
* post-clean: Not defined
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.394 s
[INFO] Finished at: 2019-03-18T20:56:50+03:00
[INFO] -----
```

Maven позволяет разработать свой собственный жизненный цикл с собственными фазами, однако такой вариант использования инструмента применим крайне редко.

Maven Plugins

Плагин – это компонента maven-фреймворка, позволяющая выполнять ряд задач (целей). Если говорить в целом, то Maven – это фреймворк, который выполняет плагины, а плагин представляет собой набор целей. Каждая из целей плагина может быть назначена/привязана к любой из фаз жизненного цикла.

Плагины Maven используются для создания jar-файла, создания war-файла, компиляции кода файлов, юнит-тестирования кода, создание отчётов проекта, создание документации проекта. Например компилятор – это основной плагин, он доступен по умолчанию, но практически в каждом проекте, его приходится переобъявлять т.к. настройки по умолчанию не очень подходящие.

Пример применения:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>2.0.2</version>
      <configuration>
        <source>1.6</source>
        <target>1.6</target>
        <encoding>UTF-8</encoding>
      </configuration>
    </plugin>
  </plugins>
</build>
```

При провале тестов построение проекта прекращается. Можно игнорировать результаты выполнения тестов, настроив соответствующий плагин:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <configuration>
    <testFailureIgnore>true</testFailureIgnore>
  </configuration>
</plugin>
```

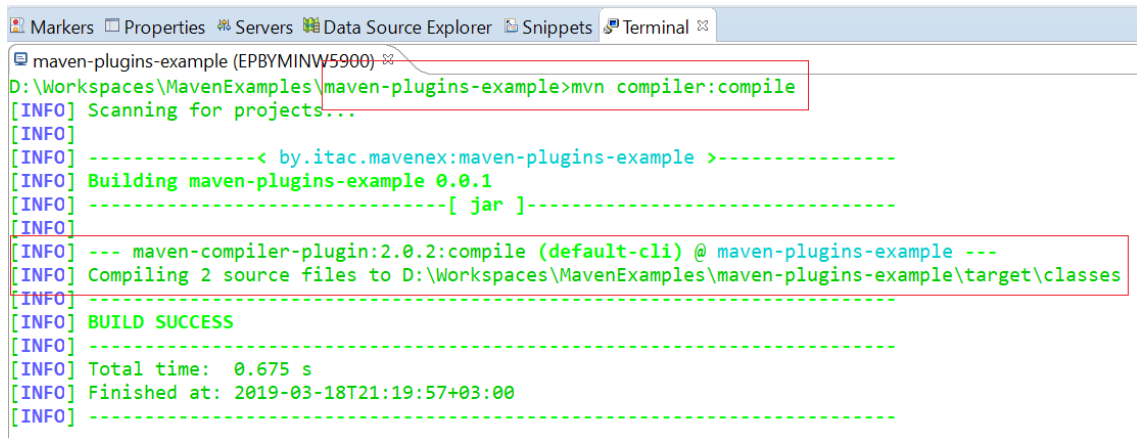
Также плагины позволяют вставить (заменить) в стандартный цикл новые шаги (например, распределение на сервер приложений) или расширить уже существующие.

В общей форме, плагин обеспечивает набор целей, которые могут быть выполнены с помощью такого синтаксиса:

>mvn [имя-плагины]:[имя-цели]

Например, проект Java может быть скомпилирован компилятором-плагином, с помощью следующей команды:

>mvn compiler:compile



```
Markers Properties Servers Data Source Explorer Snippets Terminal
maven-plugins-example (EPBYMINW5900)
D:\Workspaces\MavenExamples\maven-plugins-example>mvn compiler:compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----< by.itac.mavenex:maven-plugins-example >-----
[INFO] Building maven-plugins-example 0.0.1
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-compiler-plugin:2.0.2:compile (default-cli) @ maven-plugins-example ---
[INFO] Compiling 2 source files to D:\Workspaces\MavenExamples\maven-plugins-example\target\classes
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 0.675 s
[INFO] Finished at: 2019-03-18T21:19:57+03:00
[INFO]
[INFO] -----
```

При вызове >mvn *phase* maven перебирает все фазы (каждый раз) и выполняет все цели (поставляемые плагинами), которые были привязаны к любой фазе до и вплоть до (включительно) данной фазы.

Если к фазе не привязана ни одна цель – то ничего не произойдет, однако считается, что фаза все равно выполнится.

Цели, предоставляемые плагинами, могут быть связаны с различными этапами жизненного цикла. Например, по умолчанию цель *compiler:compile* связана с фазой *compile*, а цель *surefire:test* связана с фазой *test*.

Цель, не связанная с какой-либо фазой сборки, может выполняться за пределами жизненного цикла сборки путем прямого вызова.

>mvn *clean dependency:copy-dependencies package* – аргументы *clean* и *package* являются фазами сборки, а *dependency:copy-dependencies* является целью (плагина).

Кроме того, если цель связана с одной или несколькими фазами сборки, эта цель будет вызываться на всех этих этапах.

Если фаза сборки не имеет целей, связанных с ней, эта фаза сборки не будет выполняться. Но если у него есть один или несколько целей, связанных с ним, он выполнит все эти цели.

Для определения репозитория плагинов и плагинов, используемых по умолчанию, следует изучить *effective-pom*, элементы *<plugins>* и *<pluginRepositories>*.

>mvn *help:effective-pom*

Чтобы узнать о всех целях (goals) плагина

>mvn *help:describe -Dplugin=compiler*

Чтобы узнать о параметрах цели выполняем команду

```
>mvn compiler:help -Ddetail=true -Dgoal=compile
```

Как и зависимости плагины идентифицируются с помощью GAV (groupId, artifactId, version)

В простейшем случае запустить плагин можно и так

```
>mvn org.apache.maven.plugins:maven-checkstyle-plugin:check
```

```
D:\Workspaces\MavenExamples\maven-plugins-example>mvn org.apache.maven.plugins:maven-checkstyle-plugin:check
[INFO] Scanning for projects...
```

Плагины с groupId `org.apache.maven.plugins` можно запускать в более короткой форме.

```
>mvnmaven-checkstyle-plugin:check
>mvn checkstyle:check
```

После того, как плагин объявлен, его можно настроить так, чтобы он автоматически запускался в нужный момент. Это делается с помощью привязки плагина к фазе сборки проекта.