Java Pro 3.3

TEL-RAN
by Starta Institute

Системы сборки

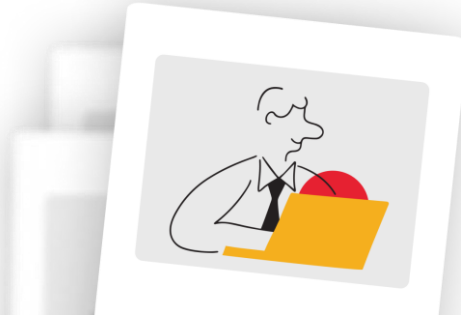
ПРЕПОДАВАТЕЛЬ

TEL-RAN
by Starta Institute

Юрий Костяной

Java/Kotlin backend-разработчик

- 3+ года опыта в коммерческой разработке
- 2+ года опыта в преподавании
- Проекты по интеграции сторонних платформ, CRM
- Проблемно-ориентированный подход в преподавании



ВАЖНО:

- Камера должна быть включена на протяжении всего занятия.
- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

TEL-RAN
by Starta Institute

ПЛАН ЗАНЯТИЯ

1. Повторение
2. Основной блок
3. Вопросы по основному блоку
4. Домашняя работа

TEL-RAN
by Starta Institute

1 ПОВТОРЕНИЕ ИЗУЧЕННОГО

Повторение

TEL-RAN
by Starta Institute

1. Дайте определение понятию “исключение”
2. Какова иерархия исключений.
3. Какие существуют способы обработки исключений?
4. О чем говорит ключевое слово throws?
5. В чем особенность блока finally? Всегда ли он исполняется?
6. Может ли не быть ни одного блока catch при отлавливании исключений?
7. В чем особенность RuntimeException?
8. Как написать собственное (“пользовательское”) исключение? Какими мотивами вы будете руководствоваться при выборе типа исключения: checked/unchecked?
9. Какой оператор позволяет принудительно выбросить исключение?
10. Может ли метод main выбросить исключение во вне и если да, то где будет происходить обработка данного исключения?
11. Если оператор return содержится и в блоке catch и в finally, какой из них “главнее”?
12. Что вы знаете о OutOfMemoryError?
13. Предположим, есть блок try-finally. В блоке try возникло исключение и выполнение переместилось в блок finally. В блоке finally тоже возникло исключение. Какое из двух исключений “выпадет” из блока try-finally? Что случится со вторым исключением?
14. Предположим, есть метод, который может выбросить IOException и FileNotFoundException в какой последовательности должны идти блоки catch? Сколько блоков catch будет выполнено?

Ответы https://javastudy.ru/interview/exceptions/

Повторение

TEL-RAN
by Starta Institute

Что будет выведено в консоль?

```
try {
    int a = 0;
    int b = 42/a;
    System.out.print("A");
} catch (Exception e) {
    System.out.print("C");
} catch (ArithmeticException e) {
    System.out.print("B");
}
```

Повторение

TEL-RAN
by Starta Institute

Что будет выведено в консоль?

```
try {
    int a = 0;
    int b = 42/a;
    System.out.print("A");
} catch (Exception e) {
    System.out.print("C");
} catch (ArithmeticException e) {
    System.out.print("B");
}
```

java: exception java.lang.ArithmeticException has already been caught

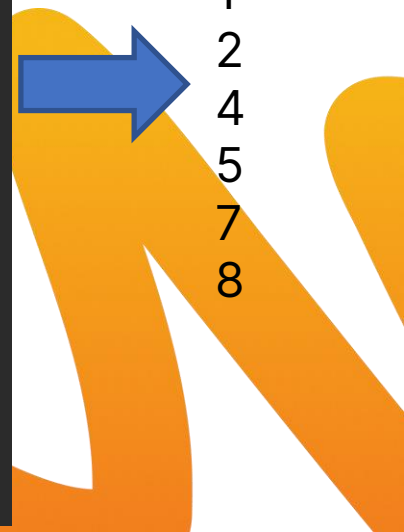
Подкласс исключения должен следовать раньше своего суперкласса в серии catch-операторов. Если это не так, то будет создан недостижимый код и будет ошибка компиляции.

Повторение

TEL-RAN
by Starta Institute

Что будет выведено в консоль?

```
try {
    System.err.println("0");
    try {
        System.err.println("1");
        // Нет исключения
        System.err.println("2");
    } catch (RuntimeException e) {
        System.err.println("3");
    } finally {
        System.err.println("4");
    }
    System.err.println("5");
} catch (Exception e) {
    System.err.println("6");
} finally {
    System.err.println("7");
}
System.err.println("8");
```



Повторение

Что будет выведено в консоль?

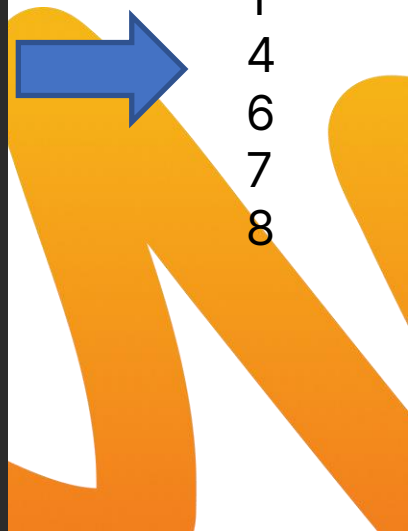
```
try {
    System.err.println("0");
    try {
        System.err.println("1");
        if (true) {
            throw new RuntimeException();
        }
        System.err.println("2");
    } catch (RuntimeException e) {
        System.err.println("3");
    } finally {
        System.err.println("4");
    }
    System.err.println("5");
} catch (Exception e) {
    System.err.println("6");
} finally {
    System.err.println("7");
}
System.err.println("8");
```

0
1
3
4
5
7
8

Повторение

Что будет выведено в консоль?

```
try {
    System.err.println("0");
    try {
        System.err.println("1");
        if (true) {
            throw new Exception();
        }
        System.err.println("2");
    } catch (RuntimeException e) {
        System.err.println("3");
    } finally {
        System.err.println("4");
    }
    System.err.println("5");
} catch (Exception e) {
    System.err.println("6");
} finally {
    System.err.println("7");
}
System.err.println("8");
```

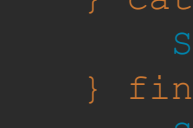


0
1
4
6
7
8

Повторение

Что будет выведено в консоль?

```
try {
    System.err.println("0");
    try {
        System.err.println("1");
        if (true) {
            throw new Error();
        }
        System.err.println("2");
    } catch (RuntimeException e) {
        System.err.println("3");
    } finally {
        System.err.println("4");
    }
    System.err.println("5");
} catch (Exception e) {
    System.err.println("6");
} finally {
    System.err.println("7");
}
System.err.println("8");
```



```
0
1
4
7
Exception in thread "main" java.lang.Error
```

Повторение

TEL-RAN
by Starta Institute

В какой строке допущена ошибка?

```
class MyException1 extends Exception { }                    // 1
class MyException2 extends RuntimeException { }             // 2
class A {
    void m1() { throw new MyException1(); }                 // 3
    void m2() { throw new MyException2(); }                 // 4
}
```

Повторение

TEL-RAN
by Starta Institute

В какой строке допущена ошибка?

```
class MyException1 extends Exception { }                    // 1
class MyException2 extends RuntimeException { }             // 2
class A {
    void m1() { throw new MyException1(); }                 // 3
    void m2() { throw new MyException2(); }                 // 4
}
```

Ошибка будет выброшена в строке 3, т.к. checked исключение MyException1 не указано в сигнатуре метода.

Повторение

В чём прикол мема?

TEL-RAN
by Starta Institute

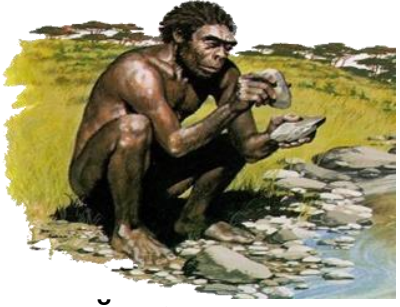
2 ОСНОВНОЙ БЛОК

Введение

TEL-RAN
by Starta Institute

- Системы сборки
- Apache Ant
- Apache Maven
- Gradle

До систем сборки



Исходный код

```
class Crm {
  public static void main(String[] args) {
    Account account = createAccount(args);
    sendOrder(account);
  }
}
```

Crm.java

Компилятор

javac

Скомпилированный код

☻ • ’ ☺ “ main ☻ – — B ▒ ☺ “ ☻ љ ќ њ ☻ ħ — ☻ ☻ Γ — ☻ ¦ ‡ Crm ☻ § Ë ↕ © є ☻ « ¬ 8 ☺ ® ☺ ï є ♥ ♥ @*· ☺*☻µ ☻*♦µ ♥*♦µ ♦*♦µ ♣*☻µ •*▶◘µ O*¶ ◙µ ♀*» ♪Y·

Crm.class

Crm.class

Account.class

Order.class

Упаковщик

jar

Приложение (библиотека)

Crm.jar

Проблема

TEL-RAN
by Starta Institute

Сборка jar зависит от других библиотек. Хотим все разобрать и собрать все вместе

В приложении несколько jar. Как все быстро собрать?

Как собрать для разных окружений?

Хотим после сборки сразу запустить

А тесты запустим?

Разное окружение при попытках автоматизации

А можно сразу номер сборки генерировать?

Системы сборки

Концепция

TEL-RAN
by Starta Institute

Для того, чтобы избавиться от монотонной работы сборки проекта и устранить встречающиеся проблемы, было решено унифицировать и автоматизировать процесс сборки.

Система сборки – это программа, которая из сходных файлов и папок проекта собирает приложение в соответствии с заданными настройками.

Системы сборки проектов на языке Java обычно тоже являются Java-программами и требуют установки JRE/JDK.

Apache Ant

Основные характеристики

- Пошаговое описание сборки
- *Target* - сценарий сборки, включает последовательность команд
- Команды в *XML-формате*
- Большой набор низкоуровневых команд: *javac*, *copy*, *delete*, *junit*, *exec*, *mail*...
- Настройка через *property*
- Возможность подключать дочерние проекты

Apache Ant

Пример файла настроек проекта

Обычно такой файл кладётся в корень проекта. Современные IDE при обнаружении файла системы сборки подключают необходимые модули для работы с соответствующей системой сборки.

```
<project default="build" basedir=".">
    <property name="name" value="CrmJar"/>
    <property name="src.dir" location="${basedir}/src"/>
    <target name="build"> <!-- Сборка приложения -->
        <mkdir dir="${build.classes}"/><!-- Создания директории-->
        <javac srcdir="${src.dir}" /> <!-- Компиляция-->
        <copy todir="${build.classes}"> <!-- Копирование файлов в директорию-->
          <fileset dir="${src.dir}" />
        </copy>
        <jar jarfile="${build}/${name}.jar"><!-- Сборка jar-->
          <fileset dir="${build.classes}"/>
        </jar>
       <junit /> <!-- Запуск тестов -->
       <exec executable="C:\utils\sendToServer.exe" /> <!-- Отправка приложения-->
    </target>
    <target name="build_production"><!-- Сборка приложения для PROD-->
     ...
    </target>
```

Проблема

TEL-RAN
by Starta Institute

Зависимости библиотек
От других библиотек

Однотипные многошаговые сценарии

Кто положил
sales-v2.jar и
sales-ver1.2.1.jar
В папку с библиотеками?

У разных проектов
отличаются имена target,
разный жизненный
цикл сборки.

service.jar зависит
от dictionary.jar.
dictionary.jar —
забыли положить.
Проект соберется,
но упадет потом...
у заказчика ((((

Дубли библиотек
в проектах

Apache Maven

Основные характеристики

Apache Maven (идиш «собиратель знания») — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке *POM* (англ. Project Object Model), являющемся подмножеством *XML*.

Главным нововведением Apache Maven стала возможность использовать *централизованные хранилища библиотек*.

- все артефакты в хранилище имеют уникальный набор идентификаторов
- известны зависимости всех библиотек
- библиотеки больше не нужно хранить с исходным кодом, они скачиваются из хранилищ
- доступны новые подходы при разработке. Например: Каждая команда собирает свою библиотеку и помещает её в хранилище. Другие команды пользуются «чужими» библиотеками из хранилища, без самостоятельной пересборки.

Искать зависимости для Вашего проекта нужно в https://mvnrepository.com/repos/central

Apache Maven

Основные характеристики

- декларативное (а не императивное) описание сборки, т.е. описывались не шаги выполнения сборки, а то, какой результат нужно получить.

Пример содержимого pom.xml

```
<project>
  <groupId>com.company.kit</groupId><!-- Уникальные идентификаторы артефакта -->
  <artifactId>crm-app</artifactId><!-- исходники которого находятся в директории -->
  <version>2.24.8</version><!-- в которой расположен данный pom.xml -->
  <properties>
    <lib.version>1.3.2</lib.version><!-- Настройка -->
  </properties>
  <dependencies>
    <dependency><!-- Описание используемых зависимостей -->
      <groupId>org.apache.commons</groupId>
      <artifactId>commons-lang3</artifactId>
      <version3.12.0</version>
    </dependency>
  <dependency><!-- Описание используемых зависимостей -->
      <groupId> com.company. lib </groupId>
      <artifactId> lib </artifactId>
      <version>${lib.version}</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin><!-- Собрать стандартным плагином в jar -->
        <artifactId>maven-jar-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

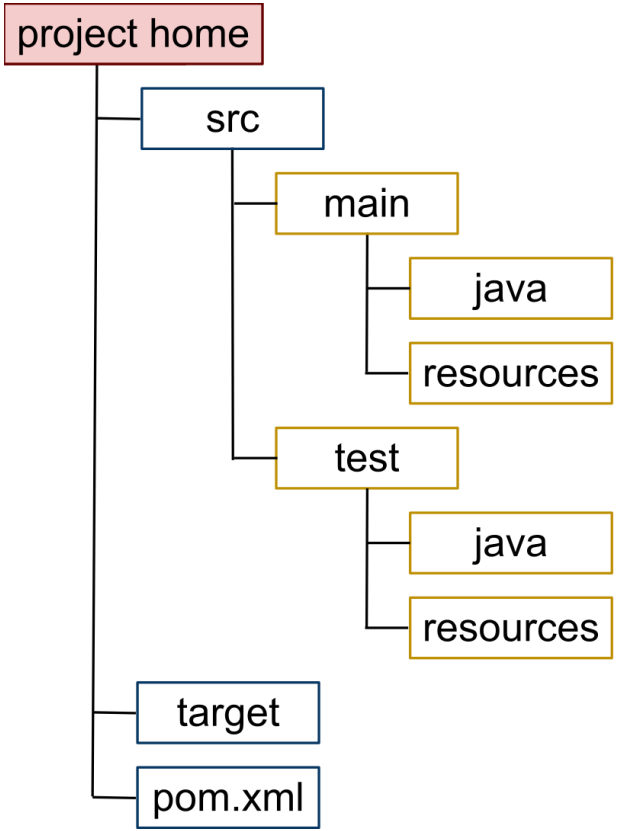
Apache Maven

Основные характеристики

- *Соглашения по конфигурации*: рассматриваемые аспекты нуждаются в конфигурации тогда и только тогда, когда этот аспект не удовлетворяет некоторой спецификации. Это позволяет сократить количество требуемой конфигурации без потери гибкости. Например, нет необходимости указывать пути к файлам, что упрощает содержимое pom.xml.

Apache Maven

Основные характеристики



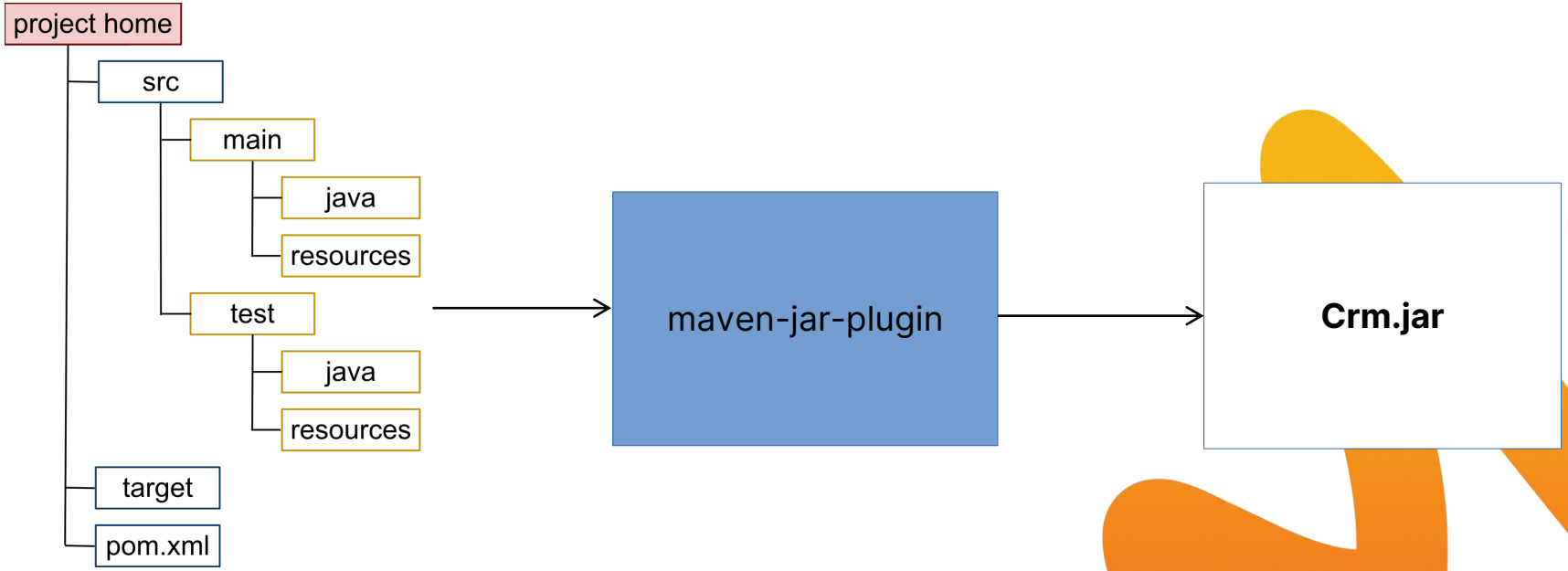
- Maven использует принцип *Maven-архетипов* (англ. Archetypes). **Архетип** – это инструмент шаблонов, каждый из которых определён паттерном или моделью, по аналогии с которой создаются производные.

Стандартная структура каталогов – одна из реализаций принципа архетипов в Maven.

Apache Maven

Основные характеристики

- *Плагины*: все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов (гибко подключаемых дополнений) без необходимости их в явном виде инсталлировать.

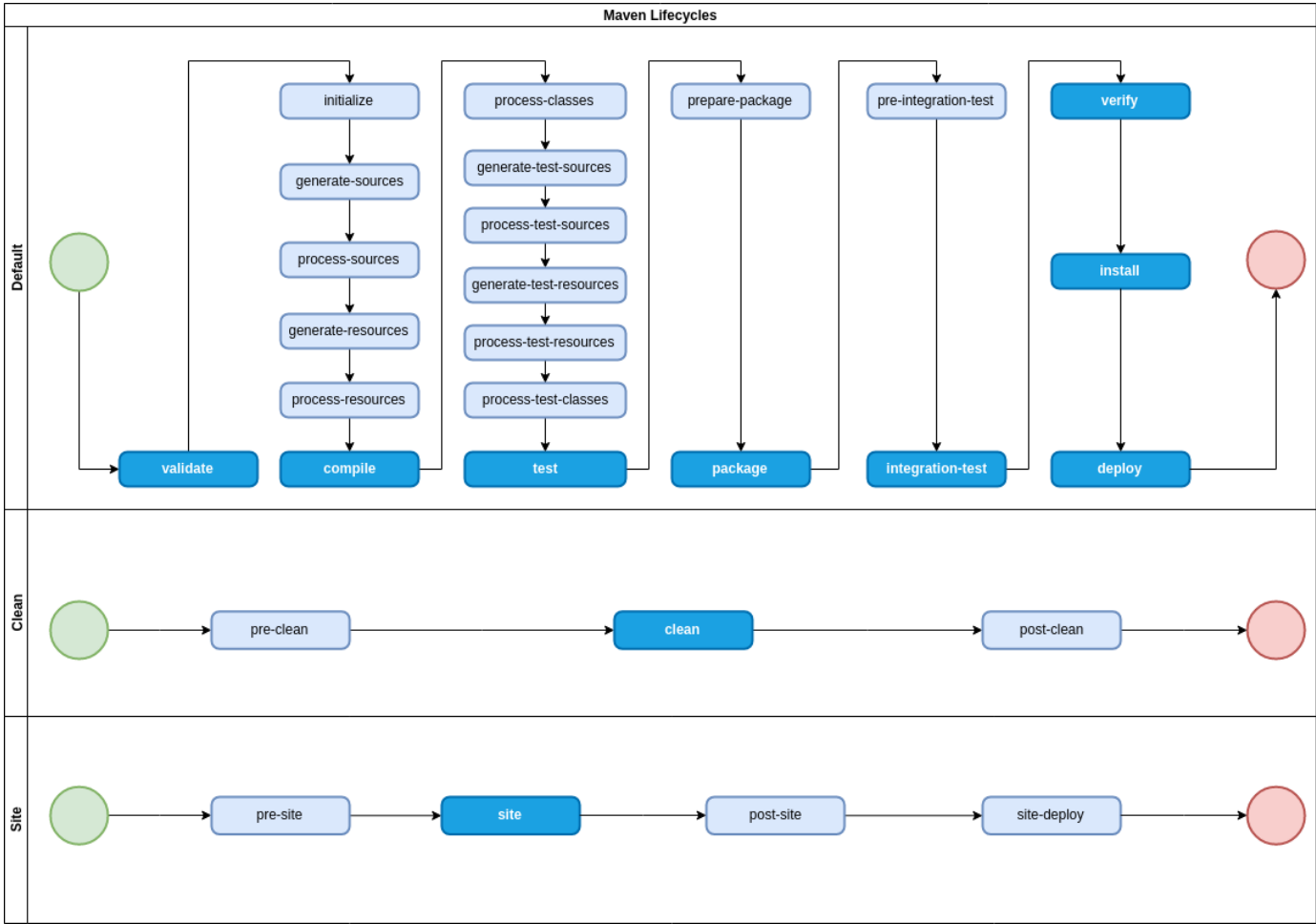


Apache Maven

Жизненный цикл проекта – это список поименованных фаз, определяющий порядок действий при его построении. Жизненный цикл Maven содержит три независимых порядка выполнения:

Apache Maven

Жизненный цикл проекта – это список поименованных фаз, определяющий порядок действий при его построении. Жизненный цикл Maven содержит три независимых порядка выполнения:



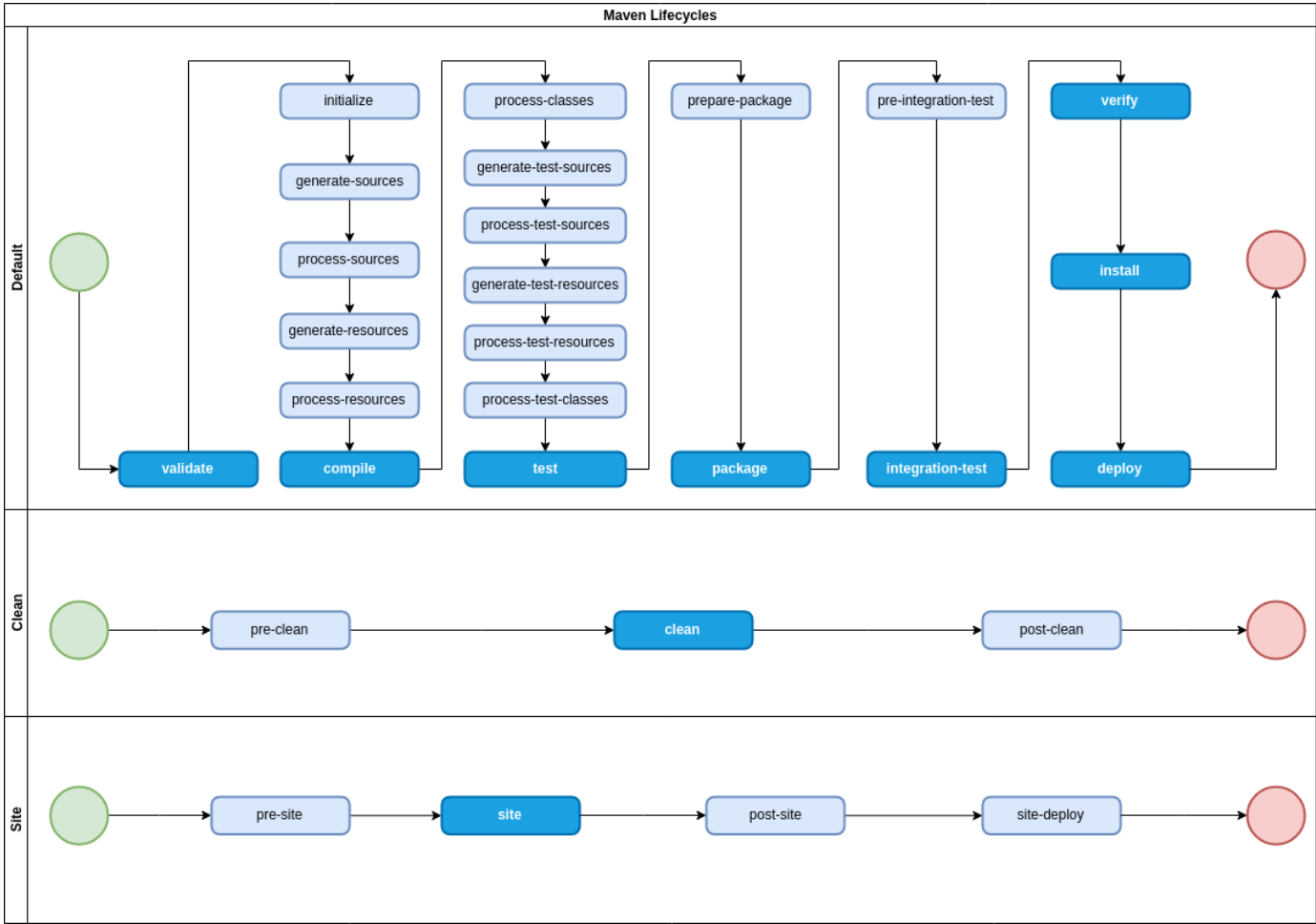
Apache Maven

default – основной жизненный цикл.

Поясним некоторые:

validate – проверка, является ли структура проекта полной и правильной.

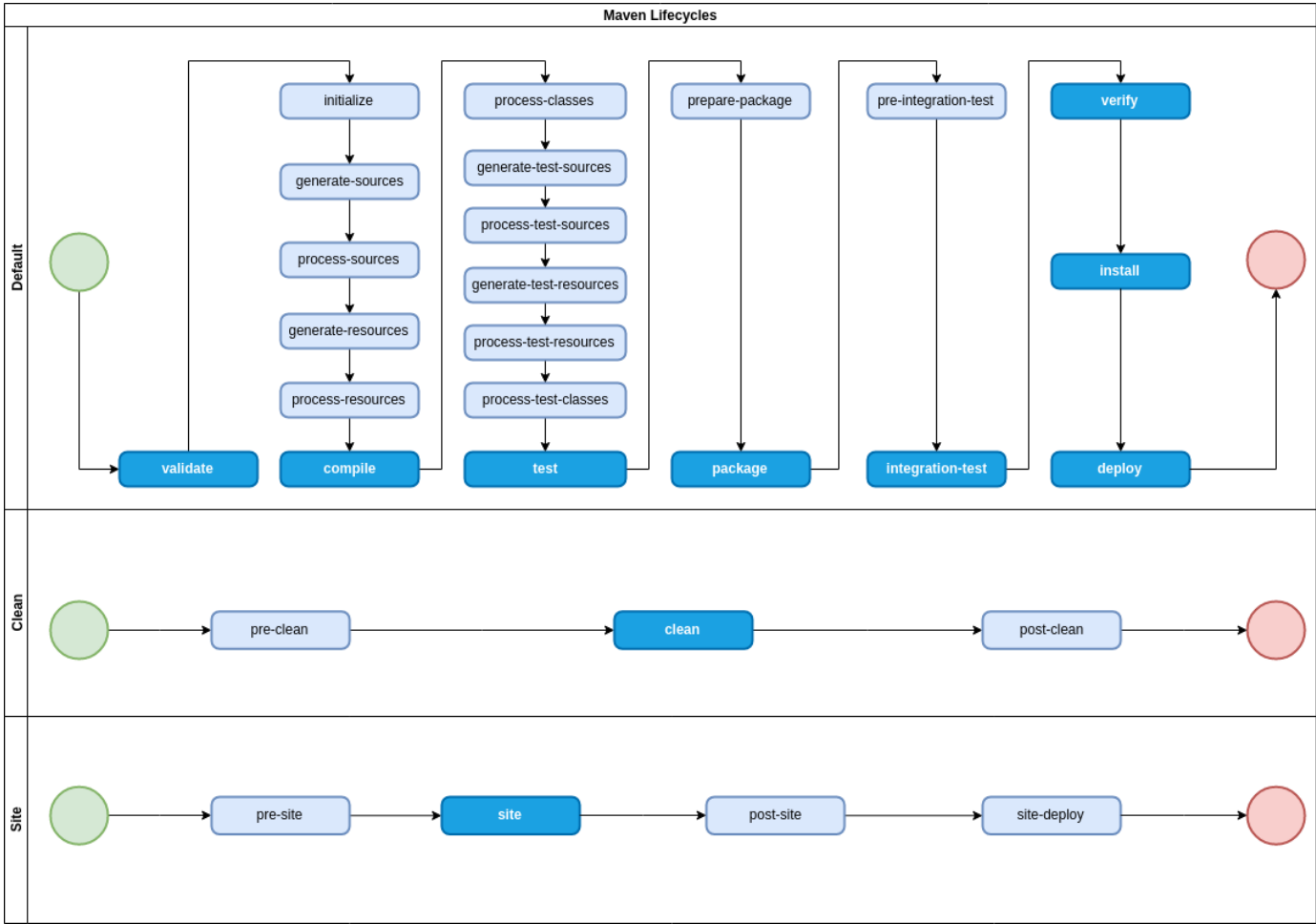
compile – компилируются исходные тексты.



Apache Maven

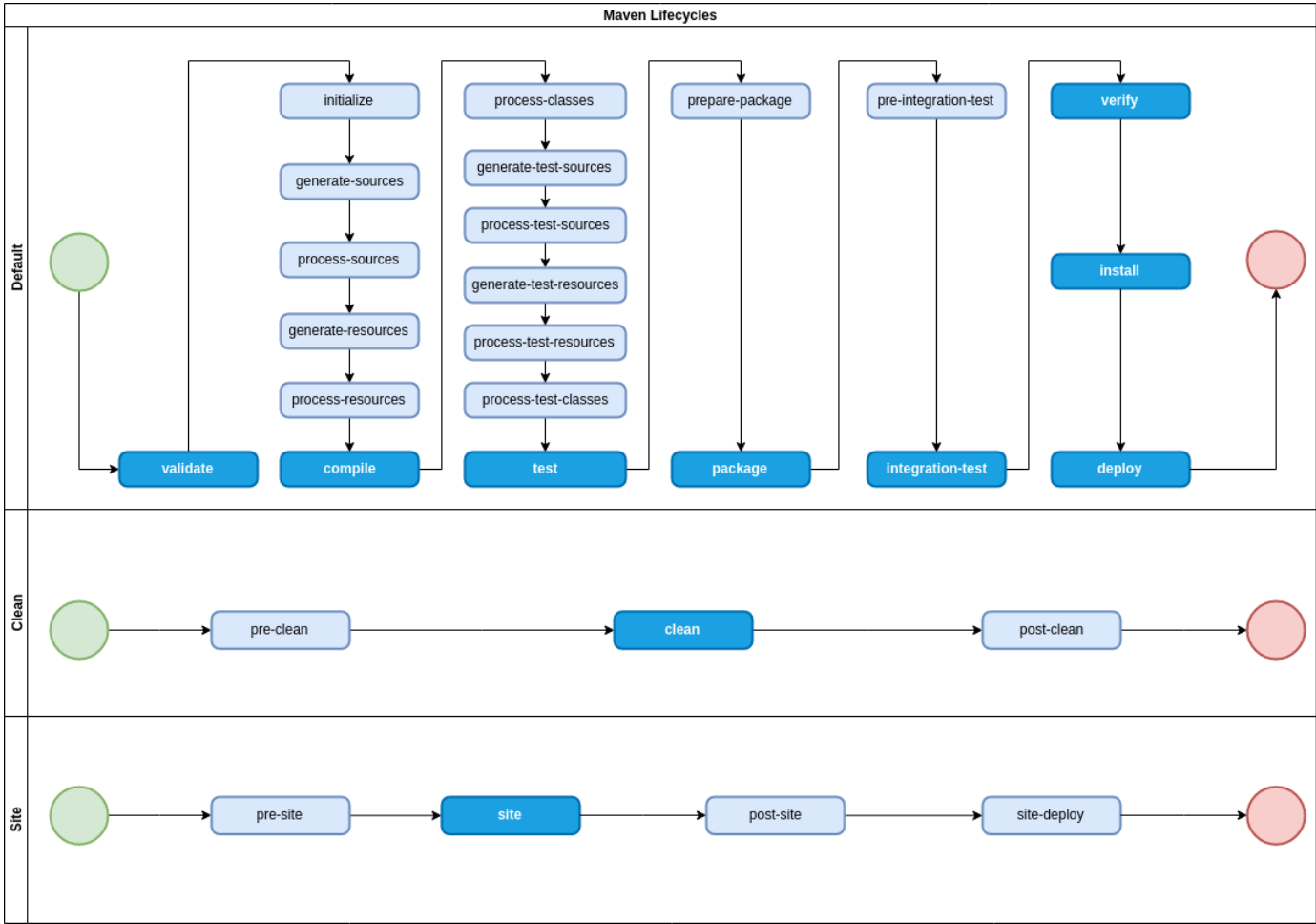
test – собранный код тестируется заранее подготовленным набором тестов (юнит-тесты).

package – упаковка откомпилированных классов и прочих ресурсов. Например, в JAR-файл.



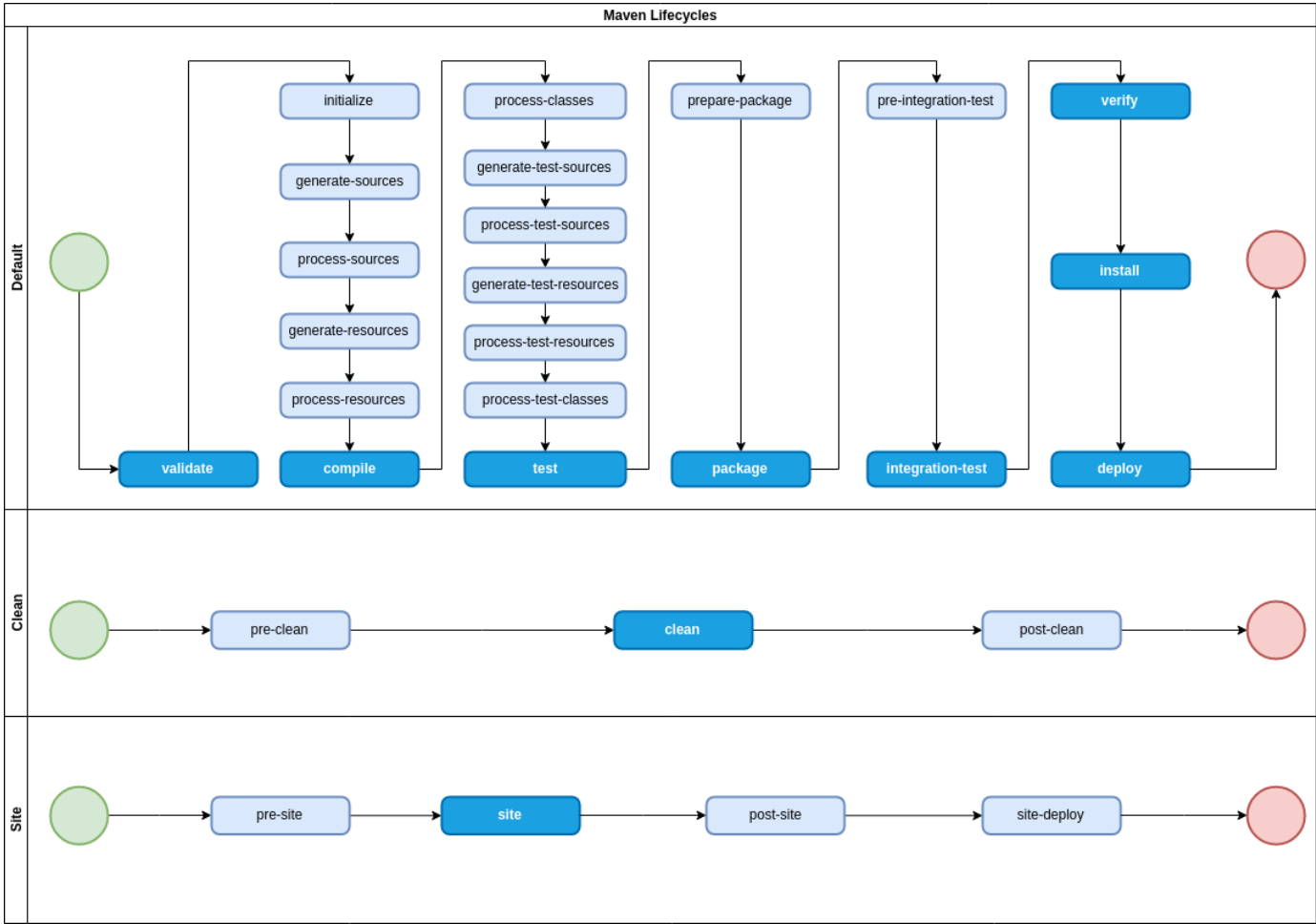
Apache Maven

integration-test – программное обеспечение в целом или его крупные модули подвергаются интеграционному тестированию. Проверяется взаимодействие между составными частями программного продукта.



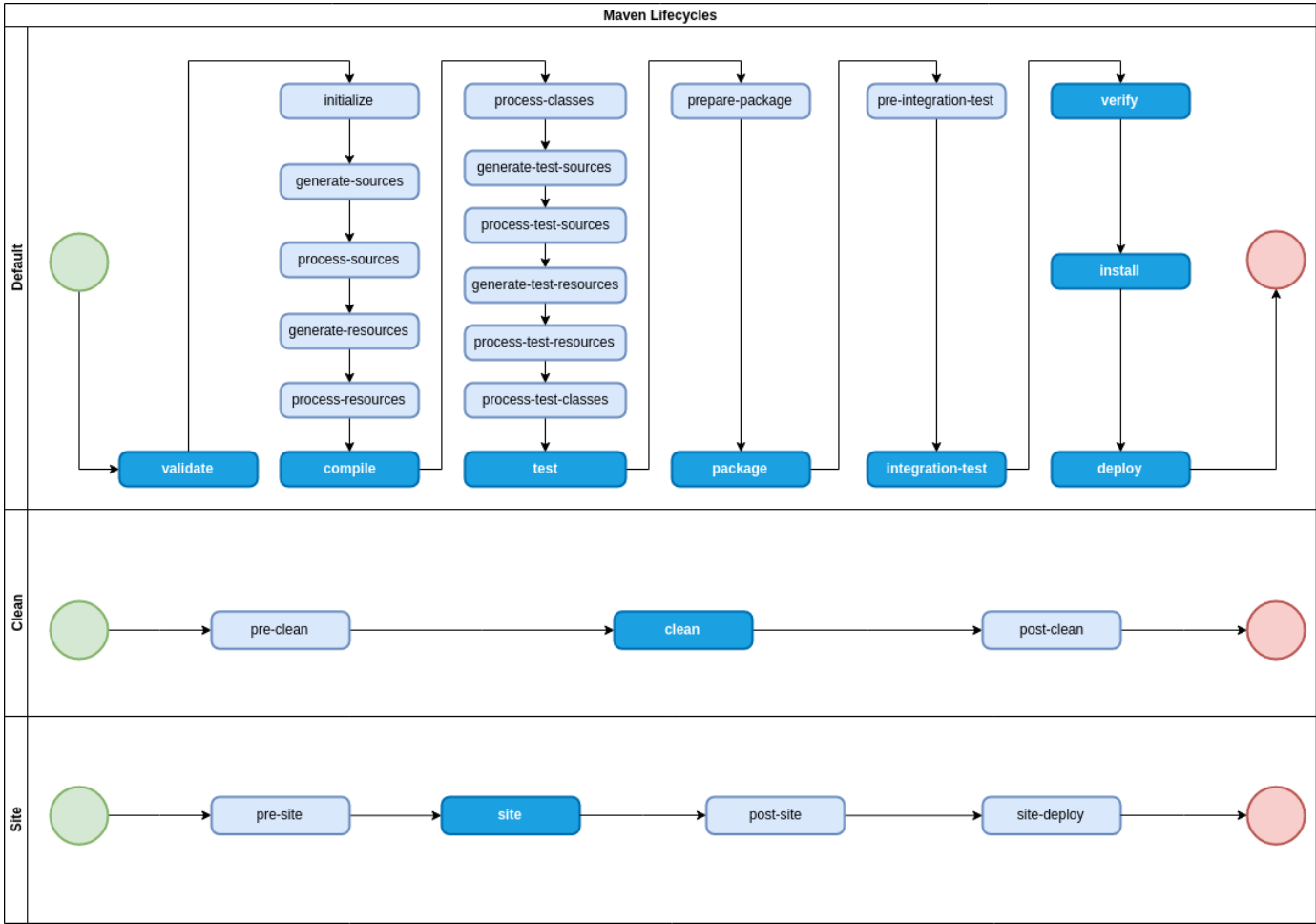
Apache Maven

install – установка программного обеспечения в локальный Maven-репозиторий, чтобы сделать его доступным для других проектов текущего пользователя.



Apache Maven

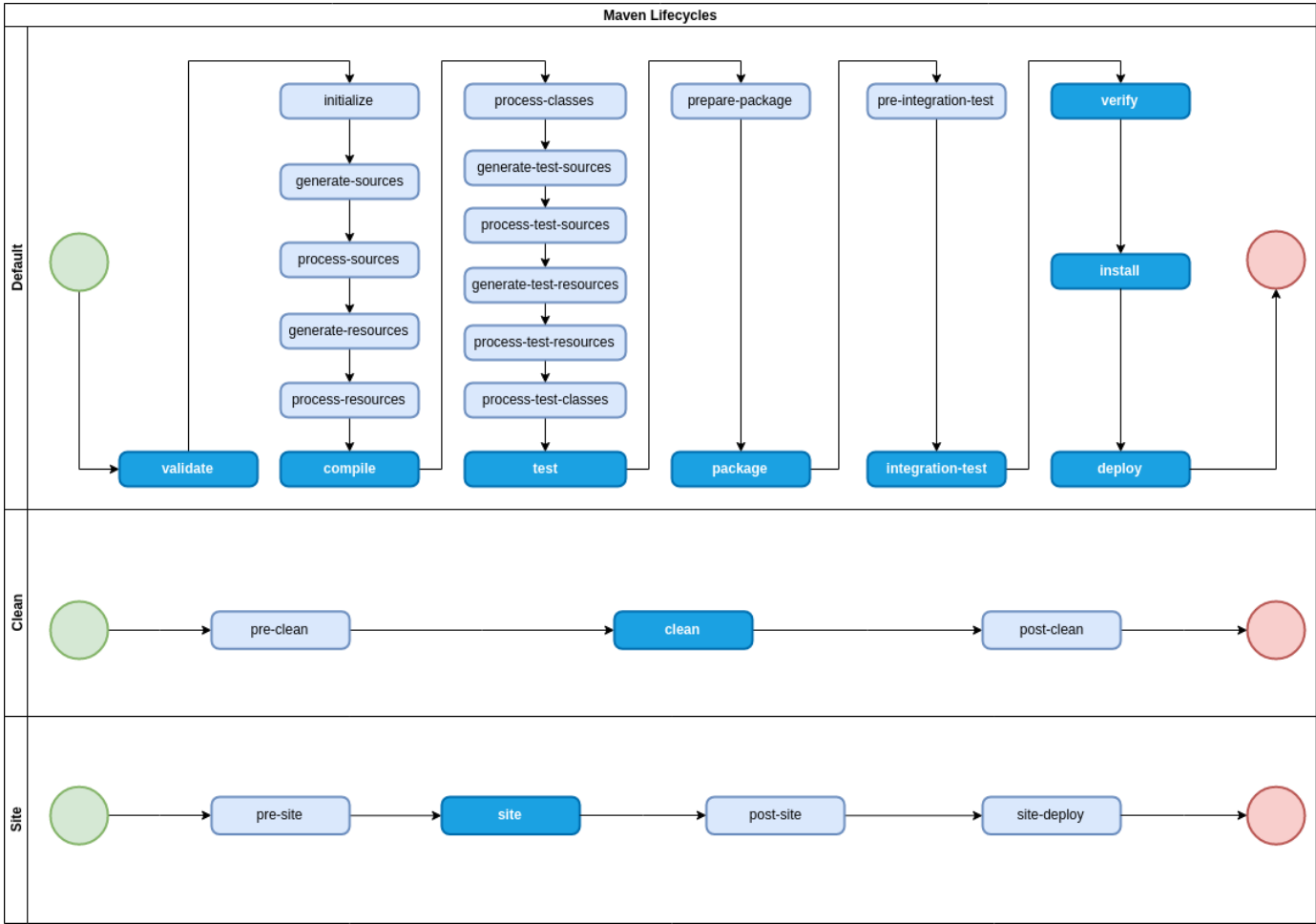
deploy – стабильная версия программного обеспечения распространяется на удаленный Maven-репозиторий, чтобы сделать его доступным для других пользователей.



Apache Maven

site – жизненный цикл генерации проектной документации.

Стандартные жизненные циклы могут быть дополнены функционалом с помощью Maven-плагинов.



Apache Maven

Команды

Пример команды Maven (вводится в командную строку):

mvn [имя плагина]:[имя цели]

Собрать проект: *mvn compile*

Скомпилировать пакет для распространения (напр., jar): *mvn package*

Собрать и упаковать, *пропустив* модульные тесты: *mvn package -DskipTests*

Установить собранный пакет в локальном maven репозитории. (Это может вызвать команды сборки и пакетирования также): *mvn install*

Удалить артефакты сборки из указанной директории: *mvn clean*

Запустить очистку и запустить фазу пакетирования: *mvn clean package*

Очистить, а затем запакетить код для указанного профиля сборки:

mvn clean -P{{profile}} package

Запустить класс из main метода:

mvn exec:java -Dexec.mainClass="{{com.example.Main}}" -Dexec.args="{{arg1 arg2}}"

Задание

TEL-RAN
by Starta Institute

1.1 Скачайте maven (Binary zip archive).

https://maven.apache.org/download.cgi

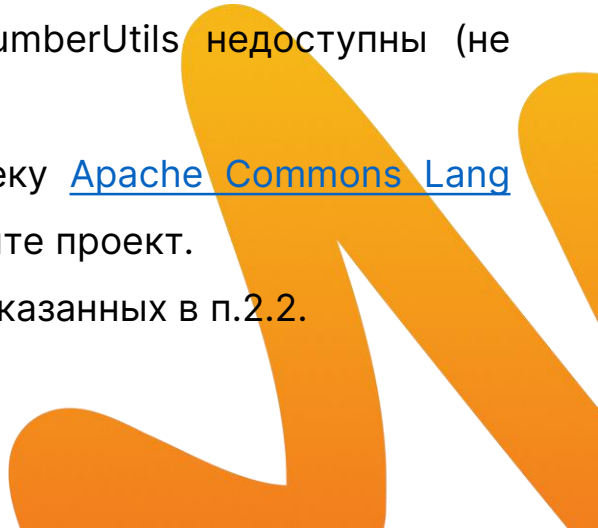
1.2 Установите maven на ПК согласно README.txt.

2.1 Создайте maven-проект в IntelliJ IDEA. Изучите структуру проекта и содержимое файла pom.xml.

2.2 Проверьте в коде, что класс StringUtils, ArrayUrils и NumberUtils недоступны (не импортируются).

2.3 В https://mvnrepository.com/repos/central найдите библиотеку Apache Commons Lang последней версии и добавьте её зависимость в pom.xml. Обновите проект.

2.4 Повторно проверьте возможность использования классов, указанных в п.2.2.



Проблема

TEL-RAN
by Starta Institute

Нет гибкости.
Сложно реализовать
нестандартное поведение

Проблемы с поддержкой
нестандартных плагинов

Скорость сборки можно
было бы увеличить

Много pom.xml

Многословное описание
зависимостей

Gradle

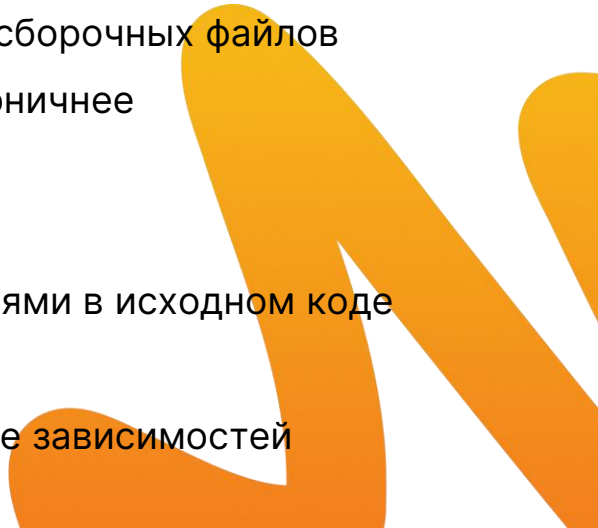
Преимущества

Преимущества, взятые у Maven:

- Плагины
- Tasks
- Хранилища библиотек

Собственные преимущества Gradle:

- Гибкое описание на *Groovy*, *Kotlin*
- Компактное описание:
- Зависимости в одну строку
- Меньшее количество сборочных файлов
- Описание сборки лаконичнее
- Быстрая сборка
- Gradle Deamon
- Контроль за изменениями в исходном коде
- Параллельная сборка
- Расширенное описание зависимостей



Gradle

Показатели перехода

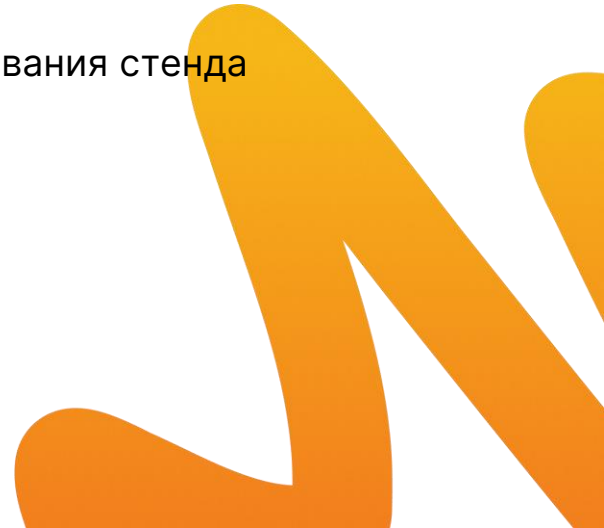
При переводе одного из крупных Enterprise-проектов с Maven на Gradle удалось достичь следующих показателей:

Время сборки (с тестами) уменьшилось (до ≈30 мин, после ≈8 мин)

Время выдачи стенда уменьшилось (до ≈30 мин, после ≈2 мин)

Появилась возможность использовать debug в Idea без раскатывания стенда

Появилась возможность добавлять различные Tasks



Gradle

Команды

Вызов страницы помощи
> **gradle help**
Список доступных задач (build, clean, test, help...)
> **gradle tasks**
Список доступных проектов (crm-be, integration-api, commons-utils..)
> **gradle projects**
Сборка и выполнение тестов
> **gradle build**
Запуск тестов
> **gradle test**
Запуск очистки
> **gradle clean**

Gradle

Команды

Список из нескольких задач. Например: Сборка и выполнение тестов.

```
> gradle clean build
```

Запуск одной задачи для одного проекта. Например: Сборка crm-api

```
\crm-be\crm-api> gradle :build
> gradle crm-be:crm-api:build
```

Исключение задачи

```
> gradle test -x crm-be:crm-kit:crm-kit-test:test
```

Передача параметра

```
> gradle build -PbuildProfile=dev-stand
```

Gradle

Пример конфигурации 1


```
plugins {
    `kotlin-dsl` // применяем плагин подключающий Kotlin
    id `application` // плагин настройки приложения
}
allprojects { // Настройка всех проектов
    apply plugin: 'java' // плагин java, подключаем tasks для работы с java
    group = `com.company.kit` // Настройка группы
    if (project.getProperty('env') == 'prod') { // Выбор настроек для разных стендов
        apply from: 'prod-stand.gradle'
    } else {
        apply from: 'dev-stand.gradle'
    }
}
dependencies { //Настройки зависимостей
        implementation(project(`:crm-api`)) //внутренняя зависимость
        implementation('com.company:integration:1.7.25') //внешняя зависимость
}

description = `crm-main` // Настройка описания данного проекта
application {
    mainClass = `com.company.kit.crm.Application` // Настройка главного класса
}
```

Gradle

Пример конфигурации 2


```
//Плагины
plugins {
    `kotlin-dsl` // применяем плагин подключающий Kotlin
    id `application` // плагин настройки приложения
    `basis-cluster-support` // плагин для работы с кластером
    id(`com.company.kit.swagger`) // плагин для работы с swagger
}
//Репозитории
repositories {
    mavenCentral() // подключение главного репозитория maven
}
//Настройка всех проектов
allprojects {
    apply plugin: `java` // плагин java, подключаем tasks для работы с java
}
//Настройка дочерних проектов, текущего проекта
subprojects {
    version = rootVersion.version //Перекладываем версию из родительского проекта
}
```

Gradle

Пример конфигурации 2


```
dependencies { //Настройки зависимостей
        compile('com.company:integration:1.7.25') //устаревший стиль описания зависимости
        api('org.connector:connector:2.1'))// аналогично устаревшему compile, менее оптимально по скорости сборки. С транзитивным подключения дочерних зависимостей
        implementation(project(`:crm-api`)){ //подключает зависимость, без транзитивного подключения ее дочерних зависимостей, ускоряет время сборки.
            exclude('org.yaml:snakeyaml:1.0') //исключает зависимость
        }
        testImplementation(`org.yaml:snakeyaml:1.23`) //зависимость для тестов
}
//Настройка задачи
tasks.jar { // настройка задачи jar
    enabled = false
}
//Настройка задач с типом
tasks.withType<Test>{
    maxHeapSize = `4G`
}
```

Gradle

Пример конфигурации 2


```
//Настройка приложения
application {
    mainClass.set(`com.company.crm.CrmSpringBootApplication`)
}
//Настройка кластера (при использовании плагина)
basisCluster {
    localPort = 12345
    ktunelPort = 28000
}
//Произвольный код
val confVersion: String by settings //Получение настройки gradle.properties
val buildVersion = version ?: `1.0`
//Настройка поведения на определенной фазе жизненного цикла
afterEvaluate { //Вызов после этапа конфигурирования (до выполнения)
//Если вызвать выше, то конфигурируемая задача не будет найдена
    tasks.findByName(`distZip`)?.apply {//Настройка задачи
        enabled = false
        group = null
    }
}
```

Задание

TEL-RAN
by Starta Institute

1 Скачайте и установите gradle (раздел *Installing manually*).

https://gradle.org/install/

2 Создайте gradle-проект в IntelliJ IDEA. Изучите структуру проекта и содержимое файлов build.gradle и settings.gradle.

3 Изучите описание бибилиотек Google Guava

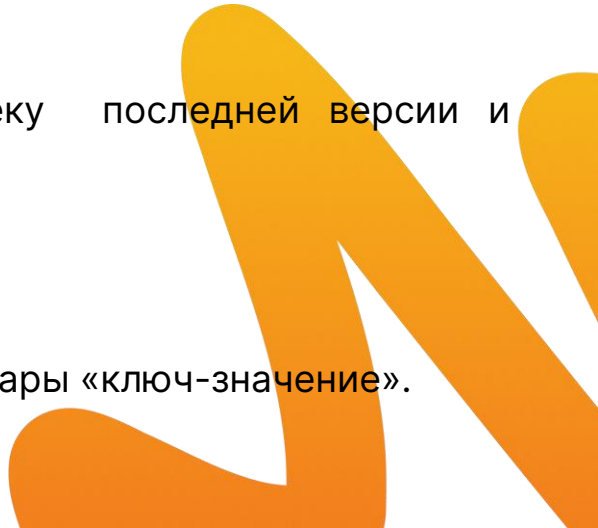
https://habr.com/ru/articles/244347/

3 В https://mvnrepository.com/repos/central найдите библиотеку последней версии и добавьте её зависимость в build.gradle. Обновите проект.

4 Создайте коллекцию Multimap из библиотеки Guava

https://www.baeldung.com/guava-multimap

Наполните её произвольным образом и выведите на экран все пары «ключ-значение».



TEL-RAN
by Starta Institute

4 Домашнее задание

Домашнее задание

TEL-RAN
by Starta Institute

1 Для выполнения домашнего задания нужно выполнить классное задание, точнее, установить maven.

2 Изучите статью по библиотеке Lombok

https://habr.com/ru/companies/piter/articles/676394/

3 Создайте maven-проект, выполните Enabling Annotation Processing и добавьте зависимости, как описано в статье https://www.baeldung.com/lombok-ide, но для последней версии Lombok (Вам нужны разделы о IntelliJ IDEA и maven). Обновите проект.

4 Создайте в проекте класс с 2-3 полями. Не генерируйте и не пишите конструктор, геттеры и сеттеры, методы toString, equals и hashcode. Вместо них укажите подходящие аннотации.

5 В main создайте экземпляр написанного класса. Доступны ли геттеры, toString и другие методы?

ЗАКЛЮЧЕНИЕ

TEL-RAN
by Starta Institute