

Java Development Kit

Урок 6

Управление проектом



План курса

1

Фреймворки для автоматизации процесса сборки и тестирования проекта

2

Паттерны и принципы авто тестирования

3

Тестирование REST API

4

Применение SQL в автоматизации тестирования

5





Мокирование и логирование

6

CI и отчеты о тестировании



Чему посвящен урок

-  Инициализация проекта;
-  Знакомство с Apache Maven;
-  Подключение зависимостей;
-  Знакомство с Gradle;



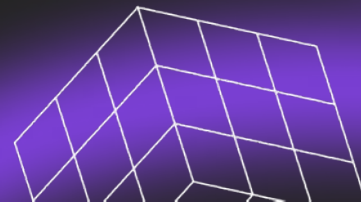
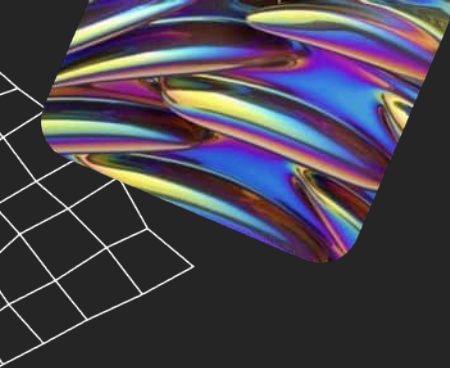


Создание и сборка приложения





Императивное vs Декларативное программирование





Декларативное программирование

— парадигма программирования, в которой задается спецификация решения задачи, то есть описывается конечный результат, а не способ его достижения.





Соглашение важнее конфигурации

— это принцип проектирования программного обеспечения, целью которого является уменьшение количества лишних действий, необходимых разработчику при создании своего проекта.





Maven





Преимущества Apache Maven

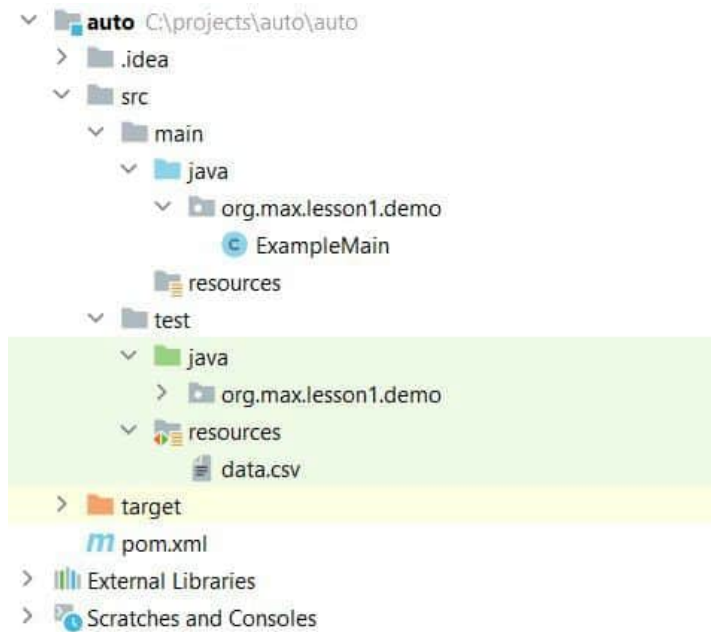
- 💡 Простая настройка проекта
- 💡 Управление зависимостями
- 💡 Большая коллекция библиотек и готовых решений
- 💡 Расширяемость
- 💡 Компиляция и запуск тестов
- 💡 Генерация отчетов по результатам тестирования





Структура каталогов Java

- `src/main/java` – исходные коды проекта, файлы с расширением `.java`. Здесь мы будем писать бизнес логику, сервисы, интерфейсы проекта.
- `src/main/resources` – «остальные» файлы проекта
- `src/test/java` – исходные коды тестов проекта, файлы расширения `.java`. Здесь мы будем писать тесты нашего проекта.
- `src/test/resources` – «остальные» файлы, необходимые для работы тестов.





Установка Apache Maven



Скачать дистрибутив Maven



Проверить системные требования



Распаковать архив в инсталляционную директорию



Настроить переменные окружения



Перезагрузить компьютер



Проверка установки Apache Maven

```
Командная строка
Microsoft Windows [Version 10.0.19044.2728]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\MAKravchenko>mvn -version
Apache Maven 3.8.6 (84538c9988a25aec085021c365c560670ad80f63)
Maven home: C:\projects\apache-maven-3.8.6-bin\apache-maven-3.8.6
Java version: 11.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-11.0.2
Default locale: ru_RU, platform encoding: Cp1251
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Users\MAKravchenko>
```



Настройка Apache Maven



Указать место расположения `setting.xml`



Изменить адрес локального репозитория



Изменить политику работы с версиями артефактов и механизмов синхронизации репозитория



Указать используемую JDK для импорта зависимостей



Указать используемые репозитории



Создание проекта Apache Maven



В IDEA выполнить команду File/New/Project...



Выбрать проект Maven.



Указать SDK



Указать GAV



Инициализировать проект



Зависимости

библиотеки, которые непосредственно используются
в проекте для компиляции кода или его тестирования





Плагины

используются самим Maven при сборке проекта
или для других целей (установка, архивирование,
генерация отчетов и так далее)





Структура pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5                               http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7 </project>
```



GAV

```
1 <groupId>org.max</groupId>  
2 <artifactId>auto</artifactId>  
3 <version>1.0-SNAPSHOT</version>
```



Стратегия работы с версиями

- 💡 Maven использует GAV для однозначного определения необходимой зависимости
- 💡 Maven в первую очередь обращается в локальному репозиторию, для проверки наличия соответствующего .jar файла совпадающего с искомым GAV.
- 💡 В случае если объект не найден, Maven обращается к центральному репозиторию и пытается вытянуть зависимость
- 💡 Если зависимость есть в локальном репозитории, Maven не обращается к центральному репозиторию и использует локальный файл



Переменные проекта

```
1 <properties>
2     <maven.compiler.source>17</maven.compiler.source>
3     <maven.compiler.target>17</maven.compiler.target>
4     <junit.version>5.7.2</junit.version>
5     <surefire.version>2.22.2</surefire.version>
6 </properties>
```



Кастомизация информации о проекте

```
1 <name>Демо проект</name>  
2 <description>Демонстрационный проект</description>  
3 <url>http://demo.max.org</url>
```



Архивирование

```
1 <packaging>jar</packaging>
```



Зависимости

```
1 <dependencies>
2   <dependency>
3     <groupId>org.junit.jupiter</groupId>
4     <artifactId>junit-jupiter-engine</artifactId>
5     <version>${junit.version}</version>
6     <scope>test</scope>
7   </dependency>
8 </dependencies>
```

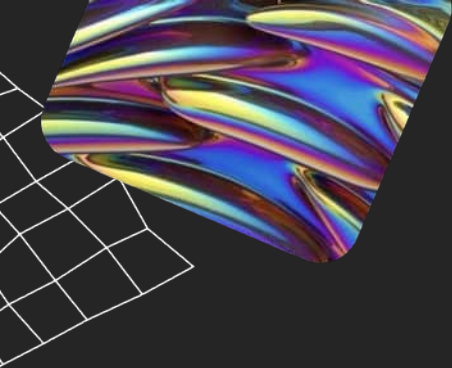


Конфликт зависимостей

- 💡 Всегда выбирается библиотека, которую указал разработчик, — то есть та, которую вы сами прописали в pom-файле. Это решение можно использовать, если Maven добавляет в сборку библиотеку меньшей версии.
- 💡 Предпочтение отдается той библиотеке, что ближе к корню (уровень узла дерева).



Иногда при конфликте библиотек
нельзя выбрать никакую из версий





Область видимости зависимостей



Compile



Provided



Runtime

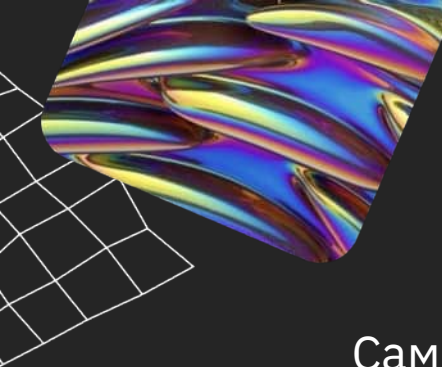


Test



Блок `<build>`

```
1 <build>
2     <plugins>
3         <plugin>
4             <groupId>org.apache.maven.plugins</groupId>
5             <artifactId>maven-compiler-plugin</artifactId>
6             <configuration>
7                 <source>17</source>
8                 <target>17</target>
9             </configuration>
10        </plugin>
11        <plugin>
12            <groupId>org.apache.maven.plugins</groupId>
13            <artifactId>maven-surefire-plugin</artifactId>
14            <version>${surefire.version}</version>
15        </plugin>
16    </plugins>
17 </build>
```



Самые частые изменения, которое вы будете вносить в pom.xml это добавление новых зависимостей или изменения версий используемых библиотек.

Сами библиотеки публикуются в **репозиториях**.





Репозиторий



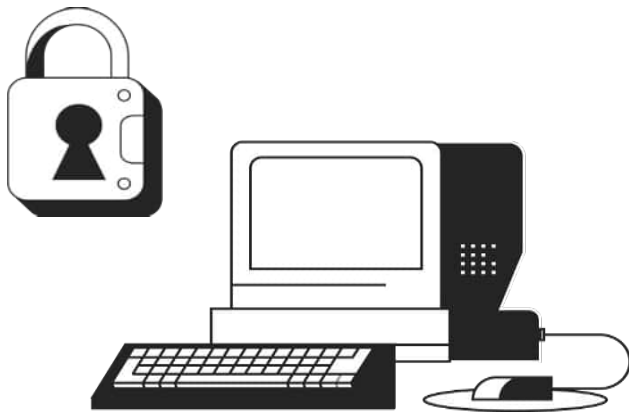
Локальный



Центральный



Удаленный





Репозиторий



Локальный — место для хранения и обновления файлов проекта;

Ваш локальный репозиторий по умолчанию располагается по адресу:

«C:\Users\\${UserName}\.m2\repository», где UserName это имя вашей учетной записи



Репозиторий



Центральный — общее онлайн-хранилище, здесь находятся все библиотеки, плагины и модули, созданные разработчиками сообщества Maven.

Адрес центрального репозитория указывается в `setting.xml`.



Репозиторий



Удаленный — обычно предназначен для хранения коммерческих библиотек или используются для предоставления доступа только к разрешенным библиотекам.

Вы можете указать дополнительные репозитории в своем `pom.xml` файле `<repositories>`, где maven также будет искать зависимости.



Рекомендаций по работе с зависимостями



Выбирайте артефакты известных вендоров



Выбирайте артефакты, для которых выпускаются обновления



Не останавливайтесь на первом ответ



Некоторые фреймворки, которые вы будете использовать, уже содержат необходимый пул библиотек



Жизненный цикл Maven

1

Clean

в общем случае используется для удаления ранее скомпилированных файлов

2

Site

используется для генерации документации

3

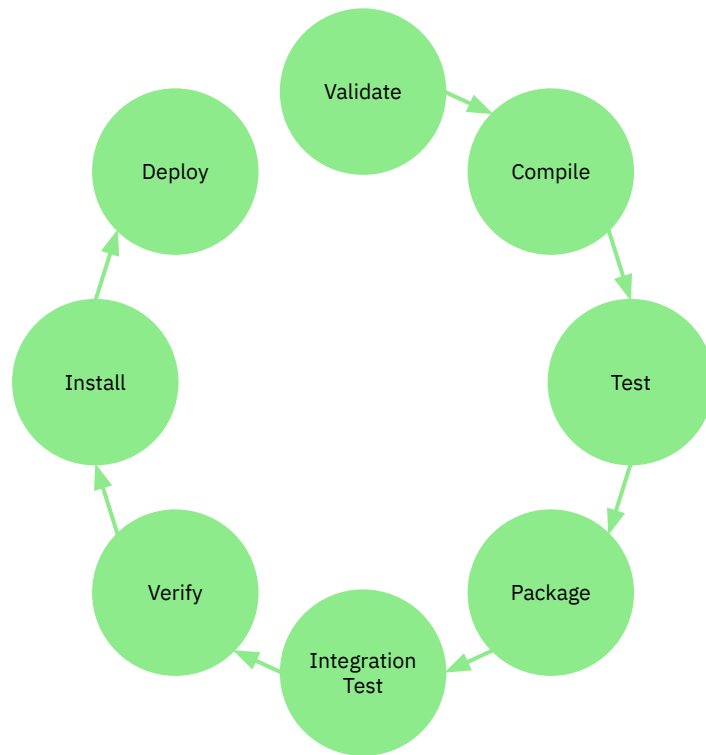
Default

основной или жизненный цикл по умолчанию, отвечает за создание приложения



Default

- **validate** – проверка структуры нашего pom.xml;
- **compile** – компиляция исходного кода нашего проекта и исходного кода тестов;
- **test** – запуск тестовых классов;
- **package** – упаковка проекта (по умолчанию jar);
- **verify** – запуск интеграционных тестов;
- **install** – копирование jar (war, ear) в локальный репозиторий;
- **deploy** – публикация файла в удалённый (внешний) репозиторий.





Gradle





Преимущества Gradle



Не только декларативный стиль



Не только java



Постоянно развивается



Быстрее



Установка Gradle



Скачать дистрибутив Gradle



Проверить системные требования



Распаковать архив в инсталляционную директорию



Настроить переменные окружения



Перезагрузить компьютер



Проверка установки Gradle

```
C:\Users\kravm>gradle -v

Welcome to Gradle 8.2!

Here are the highlights of this release:
- Kotlin DSL: new reference documentation, assignment syntax by default
- Kotlin DSL is now the default with Gradle init
- Improved suggestions to resolve errors in console output

For more details see https://docs.gradle.org/8.2/release-notes.html


-----
Gradle 8.2
-----

Build time:   2023-06-30 18:02:30 UTC
Revision:     5f4a070a62a31a17438ac998c2b849f4f6892877

Kotlin:       1.8.20
Groovy:       3.0.17
Ant:          Apache Ant(TM) version 1.10.13 compiled on January 4 2023
JVM:          17.0.7 (Oracle Corporation 17.0.7+8-LTS-224)
OS:           Windows 11 10.0 amd64
```



Создание проекта Gradle



В IDEA выполнить команду File/New/Project...



Выбрать проект Gradle.



Указать SDK



Указать GAV



Инициализировать проект



Структура Gradle проекта



`settings.gradle` — настройки проекта



`build.gradle` — конфигурация gradle модуля.



`gradle.properties` — содержит переменные (ключ + значение)



Проект (Project)

– это то, что мы создаем (например, файл JAR) или делаем (разворачиваем наше приложение в производственной среде). Проект состоит из одной или нескольких задач





Задача (Task)

– это атомарная работа, выполняемая нашей сборкой (например, компиляция нашего проекта или выполнение тестов).





Процесс сборки проекта на Gradle



Gradle запускается как новый процесс JVM



Gradle анализирует файл `gradle.properties` и соответствующим образом настраивает процесс



Создается экземпляр `Settings` для сборки



Сравнивается файл `settings.gradle` с объектом `Settings`



Создается иерархия `Projects` на основе настроенного объекта `Settings`



Gradle выполняет каждый файл `build.gradle` для своего проекта



Плагины Gradle

```
1 plugins {  
2     id 'java'  
3     id 'application'  
4 }  
5
```



Репозитории Gradle

```
1 repositories {  
2     mavenCentral()  
3 }
```



Зависимости Gradle

```
1 dependencies {  
2     testImplementation 'org.junit.jupiter:junit-jupiter-api:5.7.0'  
3     testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.7.0'  
4 }
```



Подведем итоги

- Научились писать графические интерфейсы на Swing;
- Узнали как проектировать программные интерфейсы и реализовывать множественное наследование в Java;
- Воспользовались Generic для проектирования классов и интерфейсов;
- Познакомились с Java Collection Framework и основными интерфейсами для работы с коллекциями;
- Узнали принципы организации памяти Java (Java Memory Model);
- Научились создавать потокобезопасные приложения;
- Рассмотрели механизмы автоматизации сборки и управления проектами с использованием Gradle и Apache Maven



Чек-лист подготовки к семинару

Для подготовки к семинару рекомендуем вам:

- Установить локально Maven и Gradle
- Создать проекты с использованием Maven
- Создать проекты с использованием Gradle





Спасибо за внимание

