Создание Maven-проекта

JSP & Servlets
1 уровень, 1 лекция

ОТКРЫТА

Объектная модель описания проекта

Одна из вещей, которую Maven стандартизировал в первую очередь, — это описание проекта. До Maven у каждой IDE был свой project-файл, который хранил информацию о проекте и его сборке (и зачастую в бинарном виде).

Maven предложил универсальный открытый стандарт на основе XML, в котором с помощью различных тегов описывается, что это за проект, как его нужно собирать и какие у него зависимости. Описание проекта заключено в одном файле, обычно с именем **pom.xml**.

Пример файла **pom.xml**:

```
<?xml version="1.0" encoding="UTF-8"?>
<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">

  <modelVersion>4.0.0</modelVersion>

   <groupId>example.com</groupId>
   <artifactId>example</artifactId>
   <version>1.0-SNAPSHOT</version>

   <dependencies>
       <dependency>
           <groupId>commons-io</groupId>
           <artifactId>commons-io</artifactId>
        <version>2.6</version>
        </dependency>
   </dependencies>

</project>
```

В этом примере записано три вещи:

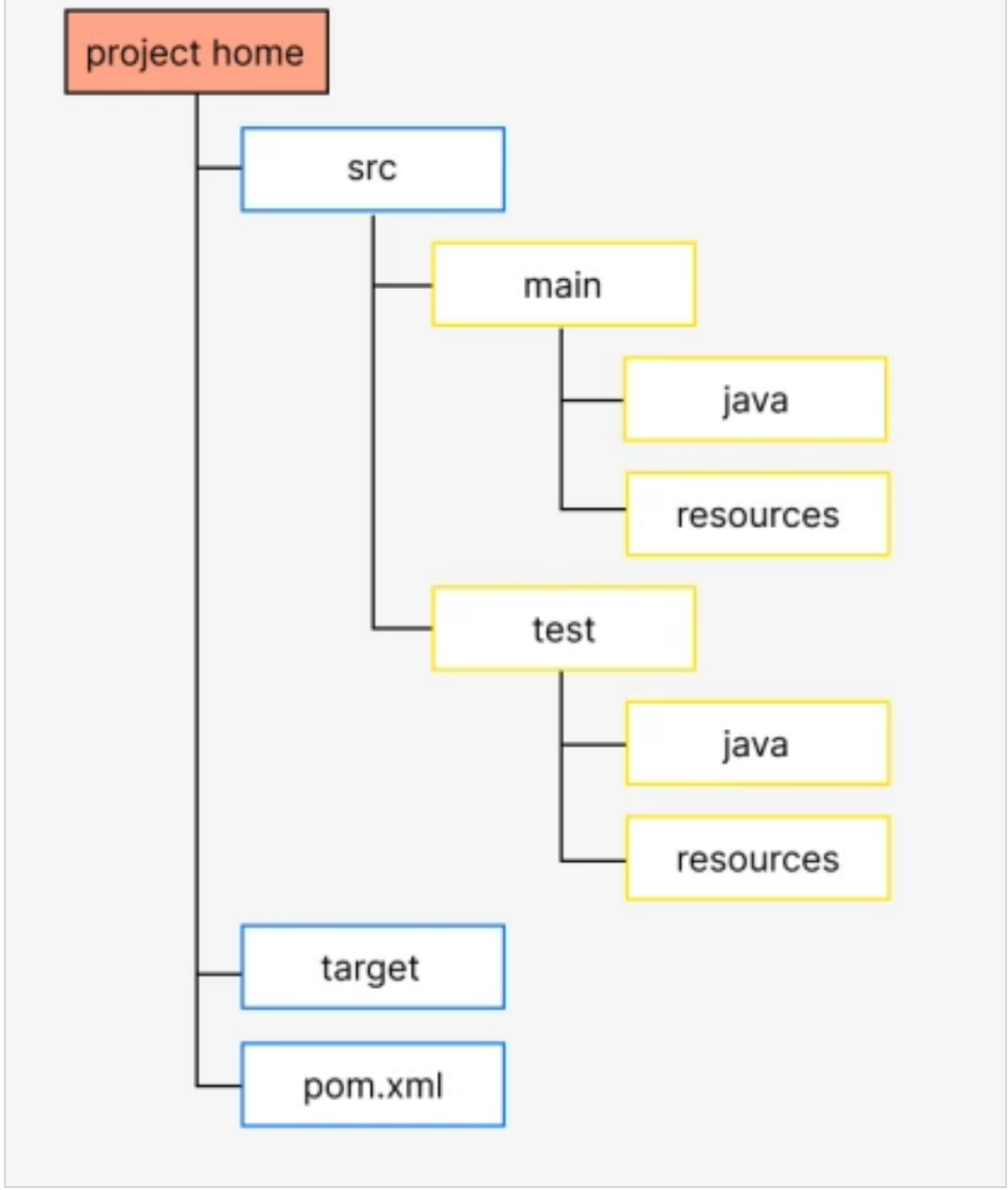
- Информация о версии стандарта maven-проекта — синим.
- Информация о самом проекте — красным.
- Информация об используемых библиотеках — зеленым.

Давай разберем устройство pom-файла подробнее.

Устройство Maven-проекта

И сразу вопрос: обратили внимание на странность в прошлом примере? В нем нет информации о самом коде проекта! Нет ни слова о том, где хранятся java-файлы, ресурсы, файлы свойств, html, скрипты сборки и тому подобное.

А ответ прост — Maven стандартизировал устройство проекта. Есть несколько вариантов организации кода внутри проекта и самый распространенный имеет вид:



Структура немного непривычная после стандартных проектов IDEA, но на то она и универсальная. 90% проектов, с которыми вы будете сталкиваться в своей жизни, будут иметь именно такую структуру папок.

Если ты создашь Maven-проект (с помощью IDEA или с помощью консоли), то он примет указанный вид. Давай разберем, как тут все устроено.

Папка **src**, как ты уже догадываешься, содержит исходный код проекта. В ней есть две подпапки: **main** и **test**.

Папка **/src/main/java** является корнем для всех Java-классов проекта. Если у тебя есть класс com.javarush.Test, то он будет лежать в папке /src/main/java/com/javarush/Test.java. Если есть текстовые или бинарные ресурсы, то они должны храниться в папке /src/main/resources.

Устройство папки **/src/test** аналогично устройству папки **/src/main**, но в ней содержатся тесты и их ресурсы. Maven сам умеет запускать нужные тесты при сборке проекта, но об этом поговорим в отдельной лекции.

Также в проекте есть папка **/target**, в которую Maven будет сохранять проект после его сборки. Так как у крупных проектов зачастую нетривиальные сценарии сборки, то в этой папке чего только не хранится.

Второе назначение папки **/target** — это кеширование промежуточных результатов сборок. При сборке большого проекта Maven может пересобрать только ту его часть, которая изменилась, ускорив, таким образом время сборки в несколько раз.

Ну и как вишенка на торте — в самом корне проекта лежит файл pom.xml. В нем-то и содержится вся необходимая информация о проекте, о котором мы поговорим ниже.

Устройство pom.xml

Начнем с того, что pom-файл — это xml, поэтому он содержит стандартные заголовки и информацию о namespaces. Это все касается чисто XML-стандарта, так что подробно об этом говорить не будем. Имеется в виду это:

```
<?xml version="1.0" encoding="UTF-8"?>
<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">

        …
```

```
</project>
```

Также обычно первой строкой внутри тега <project> идет описание версии стандарта pom-файла. Почти всегда оно 4.0. В этом тоже для нас ничего интересного.

Первые строки, которые нас интересуют, выглядят так:

```
<project>
  <modelVersion>4.0.0</modelVersion>
   <groupId>com.sample.app</groupId>
  <artifactId>new-app</artifactId>
  <version>1.0-SNAPSHOT</version>
</project>
```

Чтобы не разбираться лишний раз, что мы описываем (программу, проект, модуль, библиотеку и тому подобное) в стандарте Maven это все называется словом **артефакт**. Уж в чем создателям Maven не откажешь, так это в любви к стандартизации.

Три тега, которые ты видишь, означают:

- **groupId** – пакет, к которому принадлежит приложение, с добавлением имени домена;
- **artifactId** – уникальный строковый ключ (id проекта);
- **version – версия проекта.**

Трех указанных параметров достаточно, чтобы однозначно описать любой артефакт.

Далее, после описания проекта обычно идет список артефактов (библиотек), которые проект использует. Выглядит это примерно так:

```
   <dependencies>

       <dependency>
           <groupId>commons-io</groupId>
           <artifactId>commons-io</artifactId>
           <version>2.6</version>
       </dependency>

   </dependencies>
```

В данном примере мы добавляем в свой проект библиотеку commons-io из пакета commons-io, с версией 2.6.

Во время сборки Maven найдет такую библиотеку в своем глобальном репозитории и добавит ее в твой проект. И кстати, так умеет не только Maven.

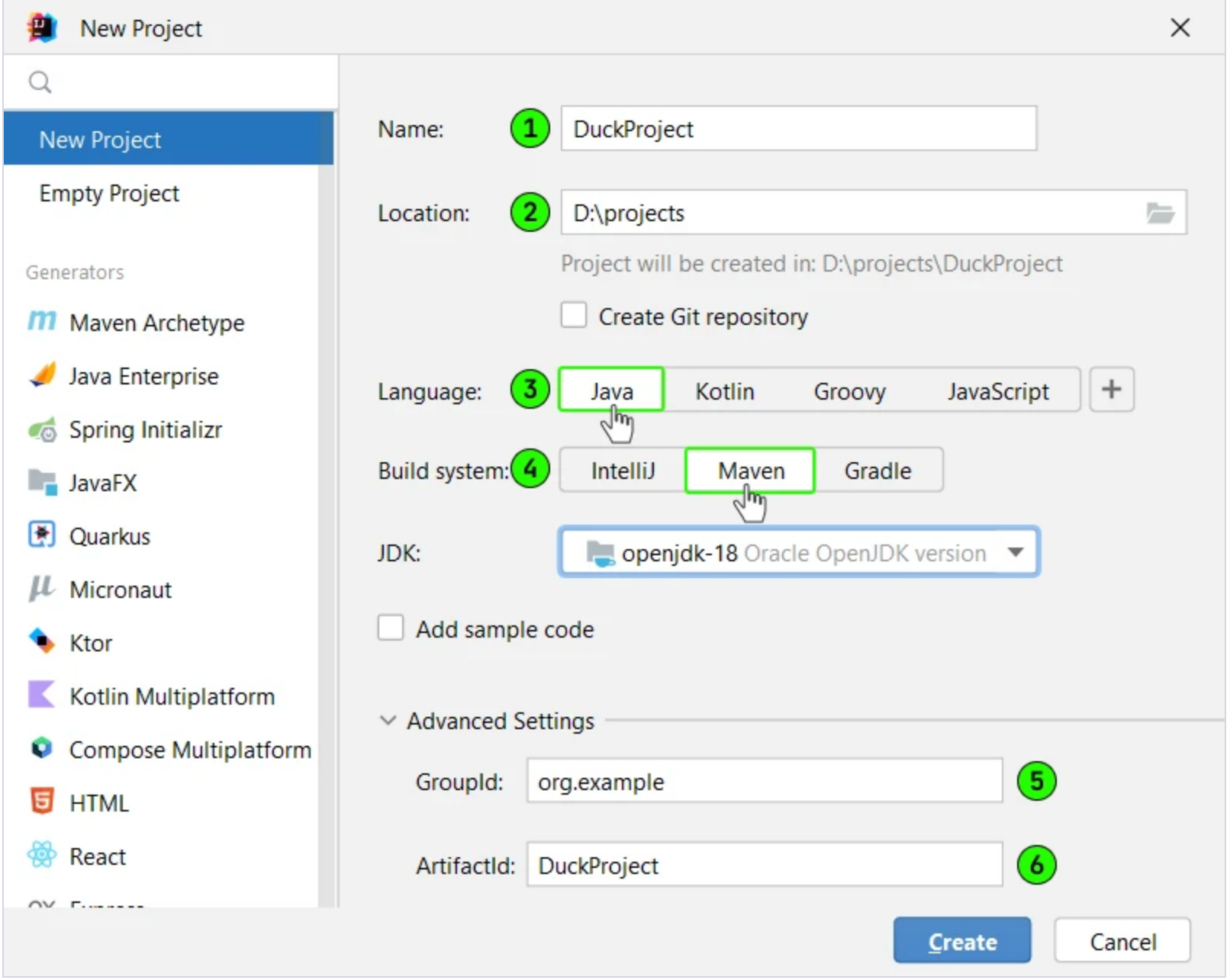
Как IDEA работает с Maven

Intellij IDEA отлично умеет работать с Maven. Она умеет открывать такие проекты, самостоятельно создавать, запускать различные сценарии сборки и отлично понимает подключаемые библиотеки.

У нее даже с некоторых пор есть свой встроенный Maven, но вам все равно нужно уметь ставить и настраивать его самостоятельно, поэтому об этой особенности IDEA ранее не упоминалось. Чисто теоретически у IDEA может быть конфликт двух Maven’ов, так что вам полезно знать, что их два.

Как создать в IDEA новый Maven-проект:

Кликаем меню Files > New Project. Выбираем слева в меню пункт **New Project**.

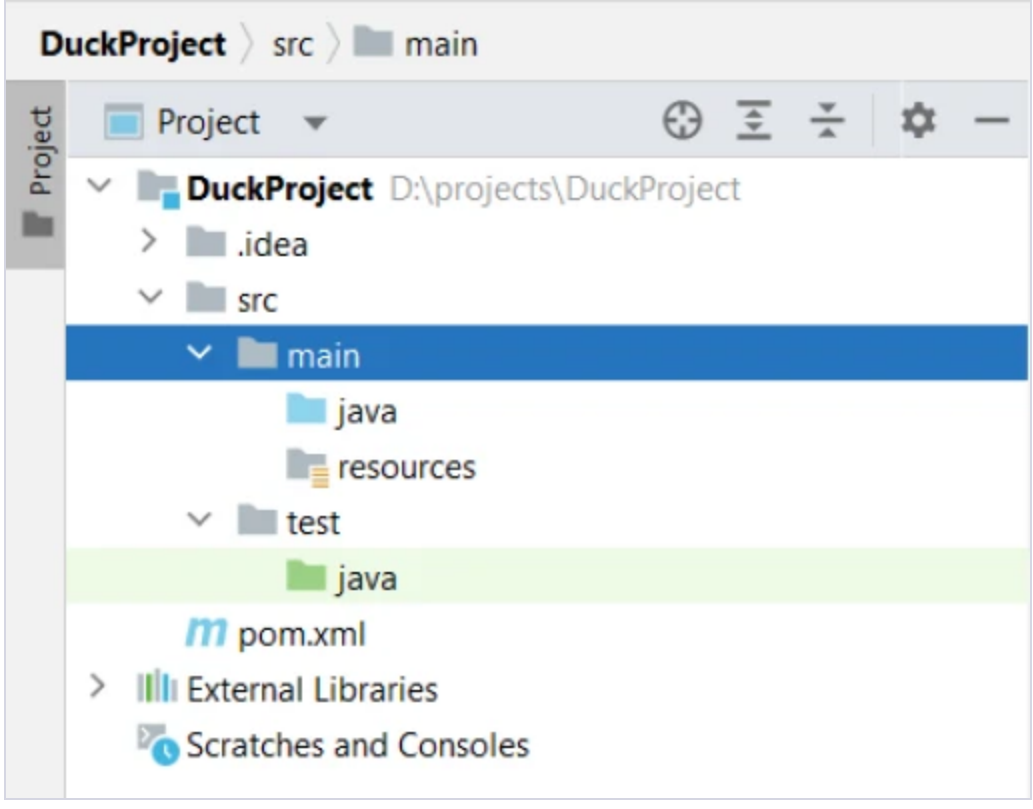


Поясним некоторые пункты:

1. Имя проекта;
2. Папка для проекта;
3. Язык проекта — Java;
4. Тип проекта — Maven.

В разделе Advanced Settings внизу IDEA предложит указать goupID, artifactID и версию нашего нового проекта. Эти данные всегда легко можно сменить позже. Выбираем предложенные или указываем свои:

Далее стандартно создаем проект в необходимом месте. В итоге видим структуру:



Классы и пакеты нужно создавать в папке java, об этом мы уже говорили. И, думаю, ты с этим легко справишься. Мы готовы идти дальше, но давай немного вернемся назад, к одному важному вопросу, который мы немного "проскочили".

Предыдущая лекция

Следующая лекция

+36

Комментарии (6) популярные **новые** старые

JavaCoder

Введите текст комментария

Михаил Уровень 51, Санкт-Петербург, Russian Federation 8 сентября, 12:57

Побурчу немного. В примере pom файла в первой вставке хорошо выделили структуру цветом. Зачем дальше было цвета менять? Сделали депенданси зеленым - продолжайте так до конца статьи.

Ответить +2

フシギダネ справедливый в **opex** 15 августа, 10:23

Все создается через JavaFX, только вот почему-то нет папки test.

Ответить 0

LuneFox Уровень 41, Москва, Россия EXPERT 11 июня, 01:12

"Если у тебя есть класс com.javarush.Test, то он будет лежать в папке /src/main/java/com/javarush/Test.java. "

Не очень удачный выбор имени для класса, лучше Cat.java или Airplane.java, потому что место для тестов в папке test :)

Ответить +11

Facepalm Уровень 32, Москвачкала, Россия 9 июня, 01:09

Как создать в IDEA новый Maven-проект: File > New > Project >(вместо обычной java выбираете внизу JavaFX) > и дальше как на картинке

Ответить +7

Тёма Панфилов Уровень 18, Подольск 28 мая, 02:04

First

Ответить 0

Владимир Уровень 68, Тель-Авив, Израиль 18 августа, 17:16

а вот и нет))

Ответить 0