



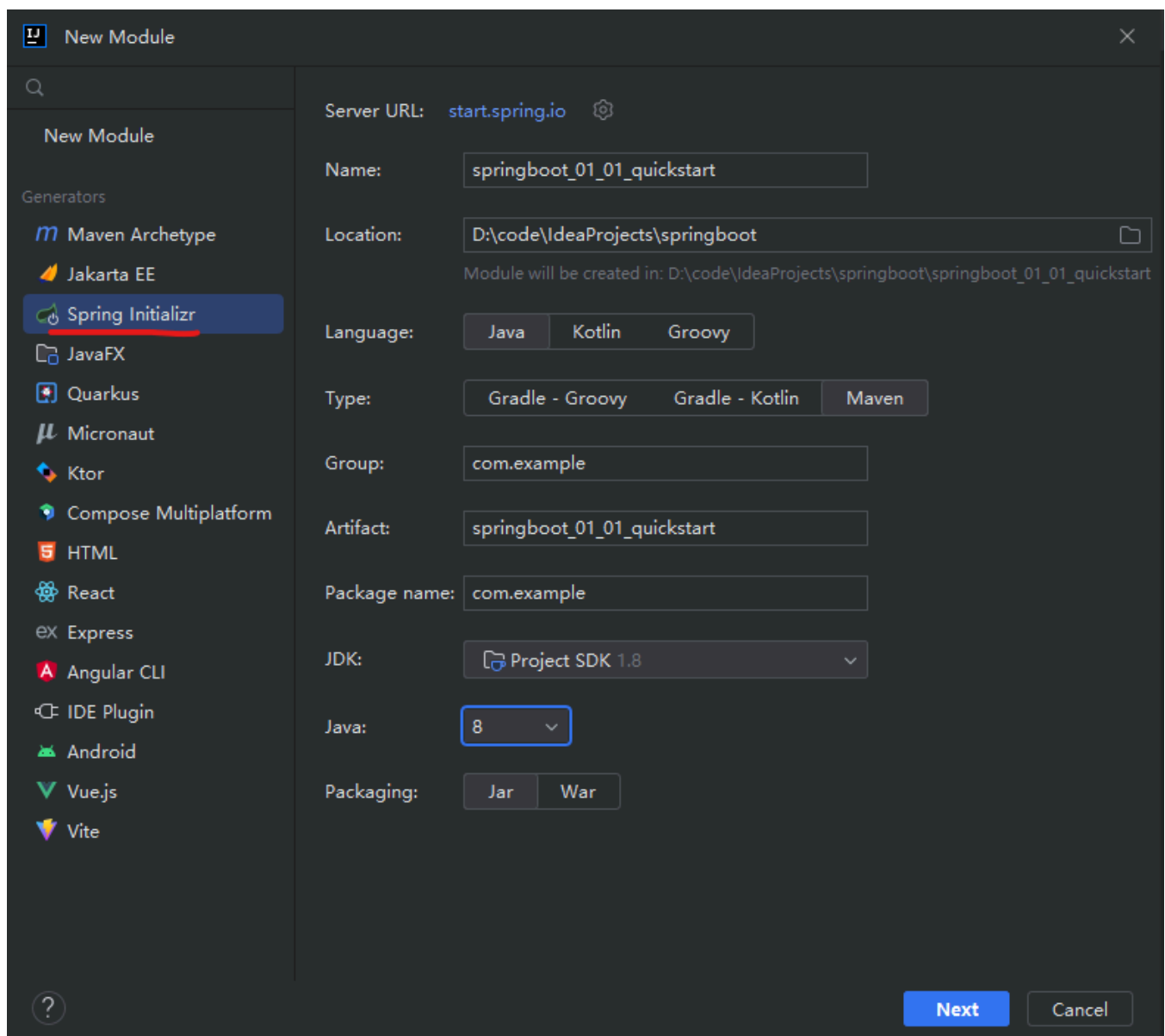
入门案例

SpringBoot是为了简化Spring框架的操作。

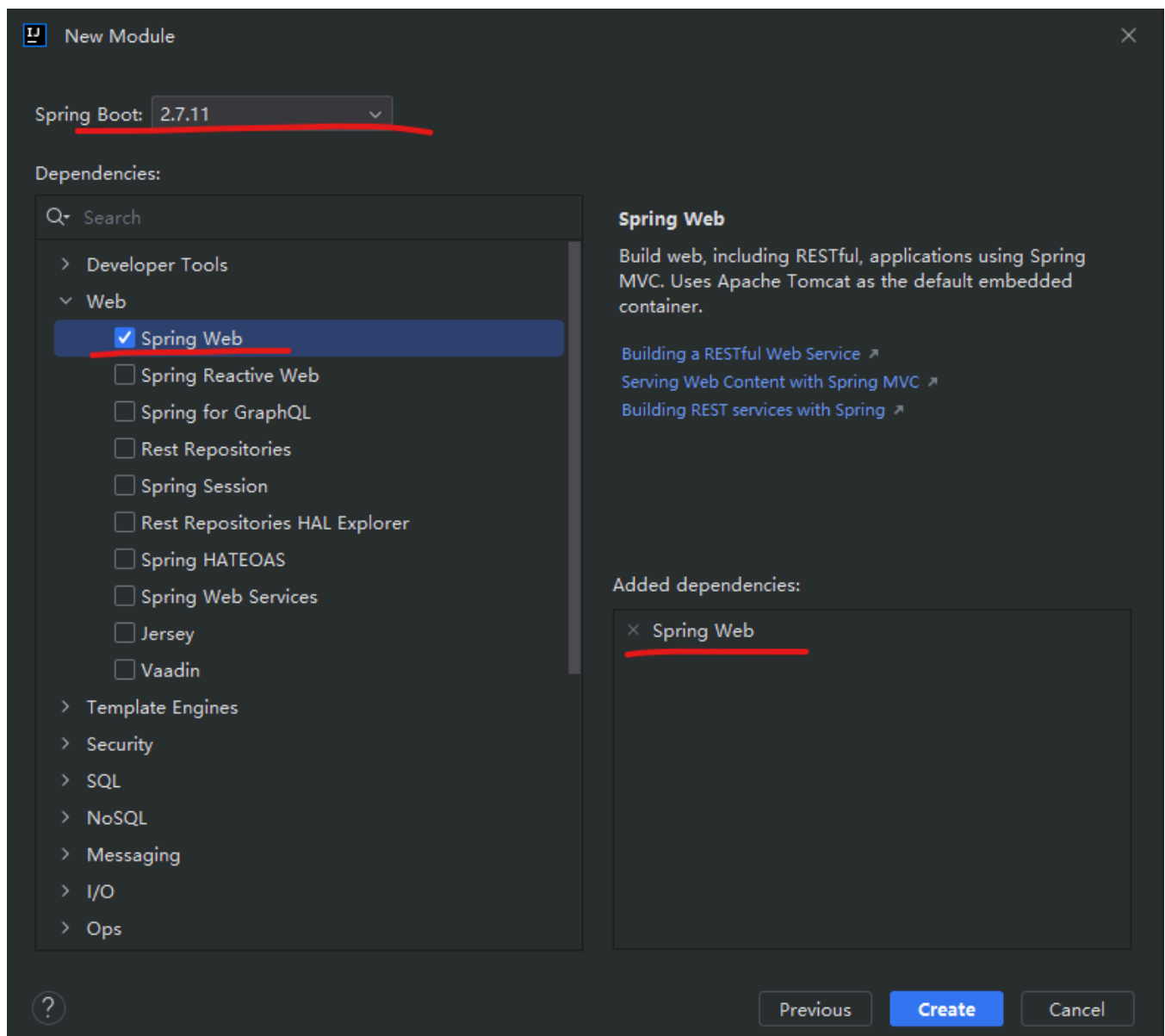
以下进行快速入门程序。

1. IDEA创建SpringBoot项目

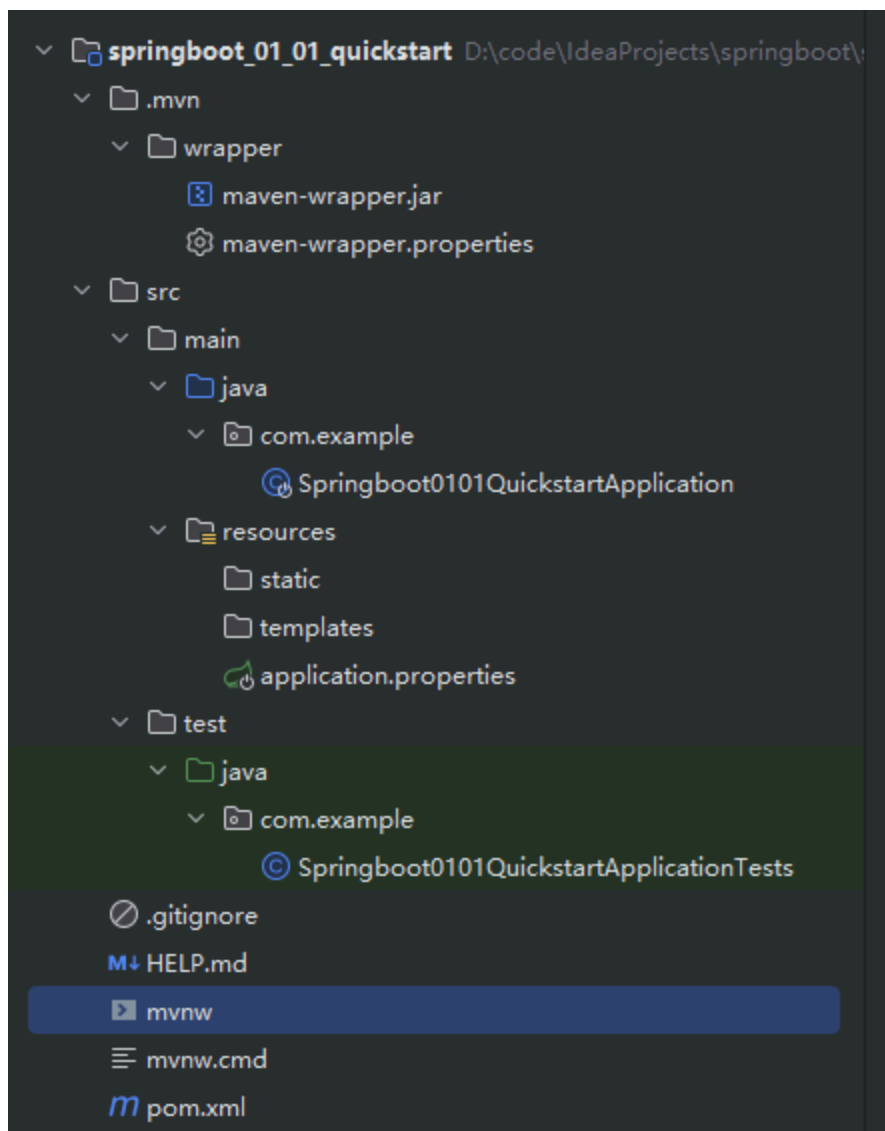
创建,



勾选依赖,



创建好的目录如下,



创建一个Controller,

```

7  @RestController
8  @RequestMapping("/books")
9  public class BookController {
10
11      @GetMapping
12      public String getById() {
13          System.out.println("springboot is running...");
14          return "springboot is running";
15      }
16
17  }
18

```

运行Application，其中带有一个main方法，

```

package com.example;

> import ...

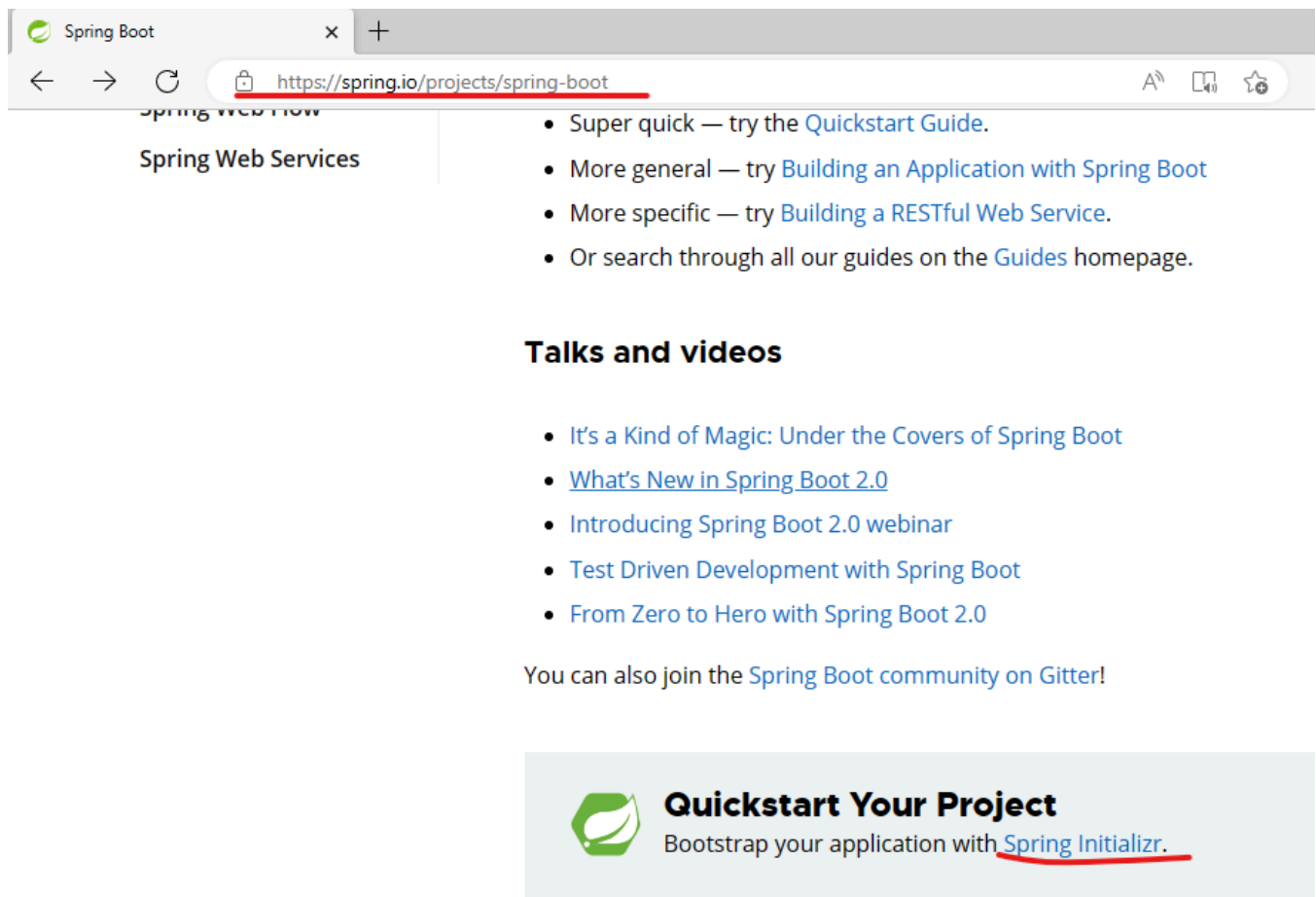
@SpringBootApplication
public class Springboot0101QuickstartApplication {

    public static void main(String[] args) {
        SpringApplication.run(Springboot0101QuickstartApplication.class, args);
    }

}

```

控制台显示如下，至此一个基于SpringBoot的Web项目创建完毕。



The screenshot shows a web browser window with the address bar displaying <https://spring.io/projects/spring-boot>. The page content includes a sidebar with "Spring Web Services" and a main area with a list of links: "Super quick — try the [Quickstart Guide](#)", "More general — try [Building an Application with Spring Boot](#)", "More specific — try [Building a RESTful Web Service](#)", and "Or search through all our guides on the [Guides](#) homepage". Below this is a section titled "Talks and videos" with a list of links: "It's a Kind of Magic: Under the Covers of Spring Boot", "[What's New in Spring Boot 2.0](#)", "Introducing Spring Boot 2.0 webinar", "Test Driven Development with Spring Boot", and "From Zero to Hero with Spring Boot 2.0". A paragraph follows: "You can also join the [Spring Boot community on Gitter](#)!". At the bottom is a light gray box with the Spring logo, the text "Quickstart Your Project", and "Bootstrap your application with [Spring Initializr](#)".

Spring Boot

<https://spring.io/projects/spring-boot>


Spring Web Services

- Super quick — try the [Quickstart Guide](#).
- More general — try [Building an Application with Spring Boot](#)
- More specific — try [Building a RESTful Web Service](#).
- Or search through all our guides on the [Guides](#) homepage.

Talks and videos

- [It's a Kind of Magic: Under the Covers of Spring Boot](#)
- [What's New in Spring Boot 2.0](#)
- [Introducing Spring Boot 2.0 webinar](#)
- [Test Driven Development with Spring Boot](#)
- [From Zero to Hero with Spring Boot 2.0](#)


You can also join the [Spring Boot community on Gitter](#)!

**Quickstart Your Project**
Bootstrap your application with [Spring Initializr](#).

创建选项与IDEA相同,

Spring Initializr

https://start.spring.io



Project

☐ Gradle - Groovy ☒ **Java** ☐ Kotlin

☐ Gradle - Kotlin ☐ Groovy

☒ **Maven**

Spring Boot

☐ 3.1.0 (SNAPSHOT) ☐ 3.1.0 (RC1)

☐ 3.1.0 (M2) ☐ 3.0.7 (SNAPSHOT) ☐ 3.0.6

☐ 2.7.12 (SNAPSHOT) ☒ **2.7.11**

Project Metadata

Group

Artifact

Name

Description

Package name



Packaging ☒ **Jar** ☐ War

Java ☐ 20 ☐ 17 ☐ 11 ☒ **8**

Dependencies ADD ... CTRL + B

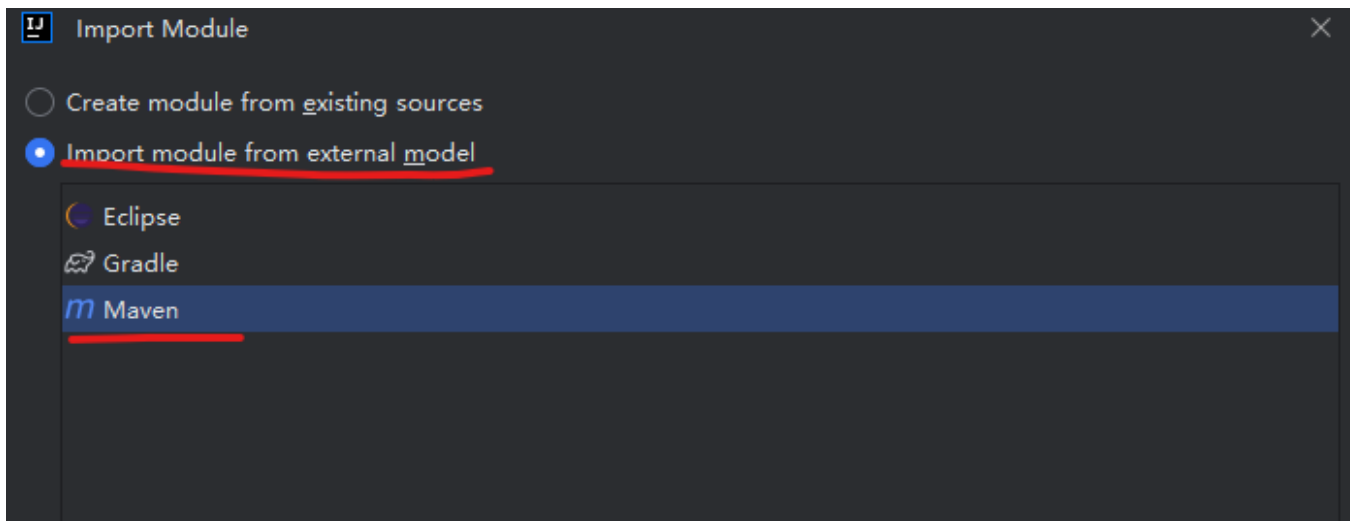
Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

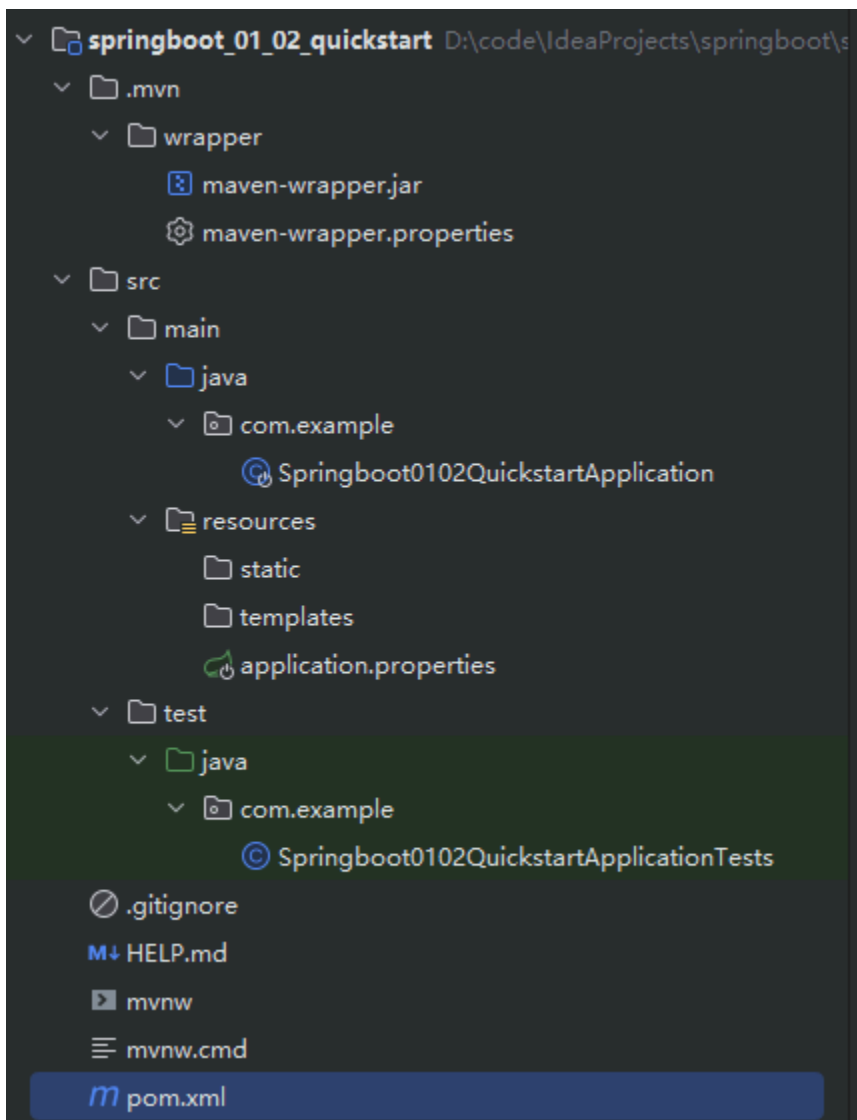
 

GENERATE CTRL + G EXPLORE CTRL + SPACE SHARE...

创建结果以压缩包的形式下载，解压后，导入到IDEA中，其中，导入选项选择Maven，

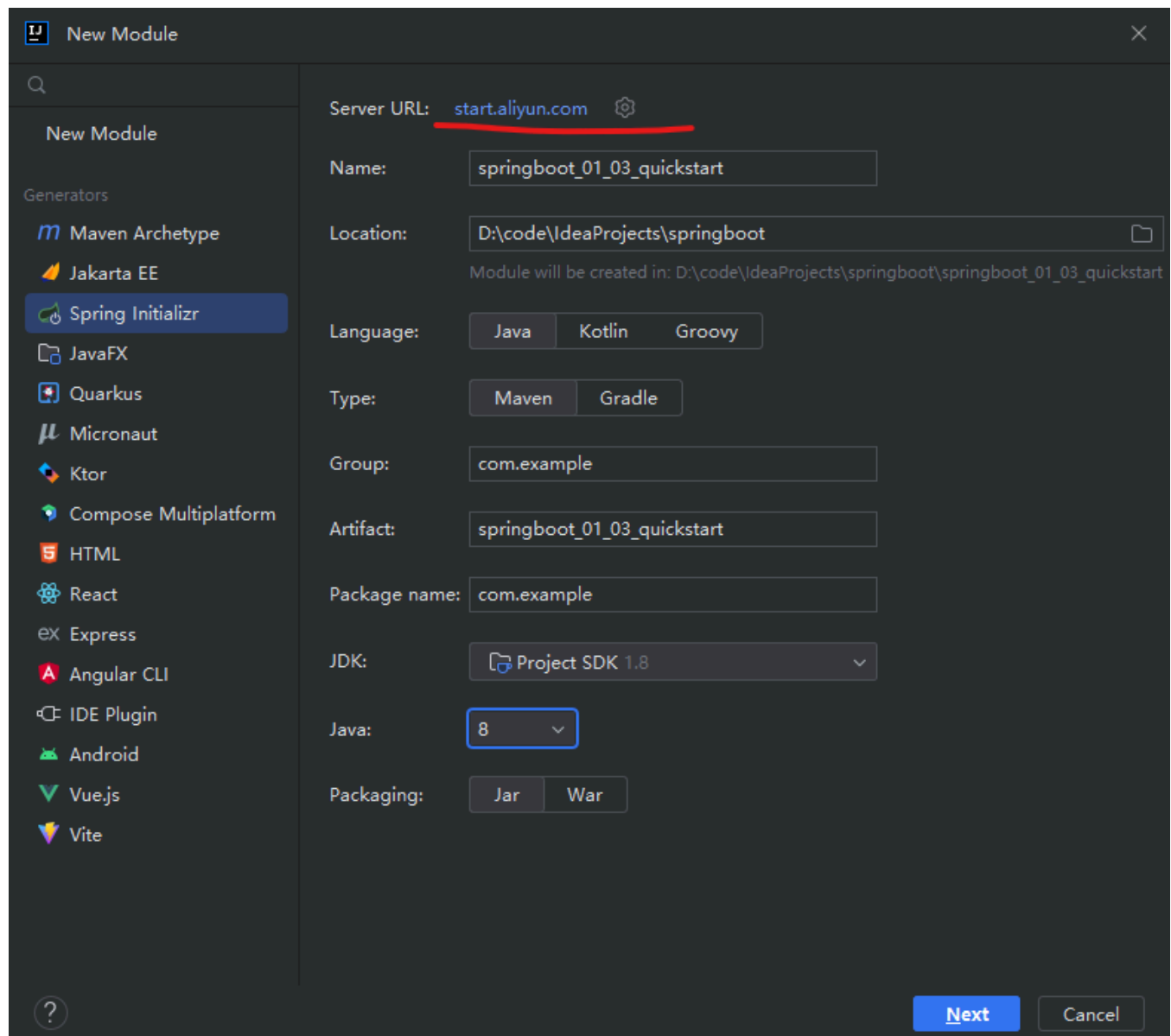


创建目录与IDEA相同，至此创建完毕。



3. 阿里云创建SpringBoot项目

通过访问阿里云创建SpringBoot项目，Server URL需要修改，



它所对应的页面如下，

Cloud Native App Initializer

https://start.aliyun.com

浅色主题 阿里巴巴微服务生态 我要反馈 解决方案

云原生应用脚手架

Cloud Native App Initializer

项目构建方式

开发语言

Spring Boot版本

项目基本信息

应用架构

组件与示例

© 1999-2023 Aliyun.com

start.aliyun.com is powered by Aliyun.com

Maven Project

Gradle Project

Java

Kotlin

Groovy

3.0.0

2.7.6

2.6.13

2.4.2

2.3.12.RELEASE

Group

com.example

Artifact

demo

> 高级选项

单模块

MVC架构

Q 搜索

分组列表

搜索依赖组件

Web, Security, JPA, Actuator, Devtools...

已选组件

暂未选择任何组件

获取代码 - Ctrl + C

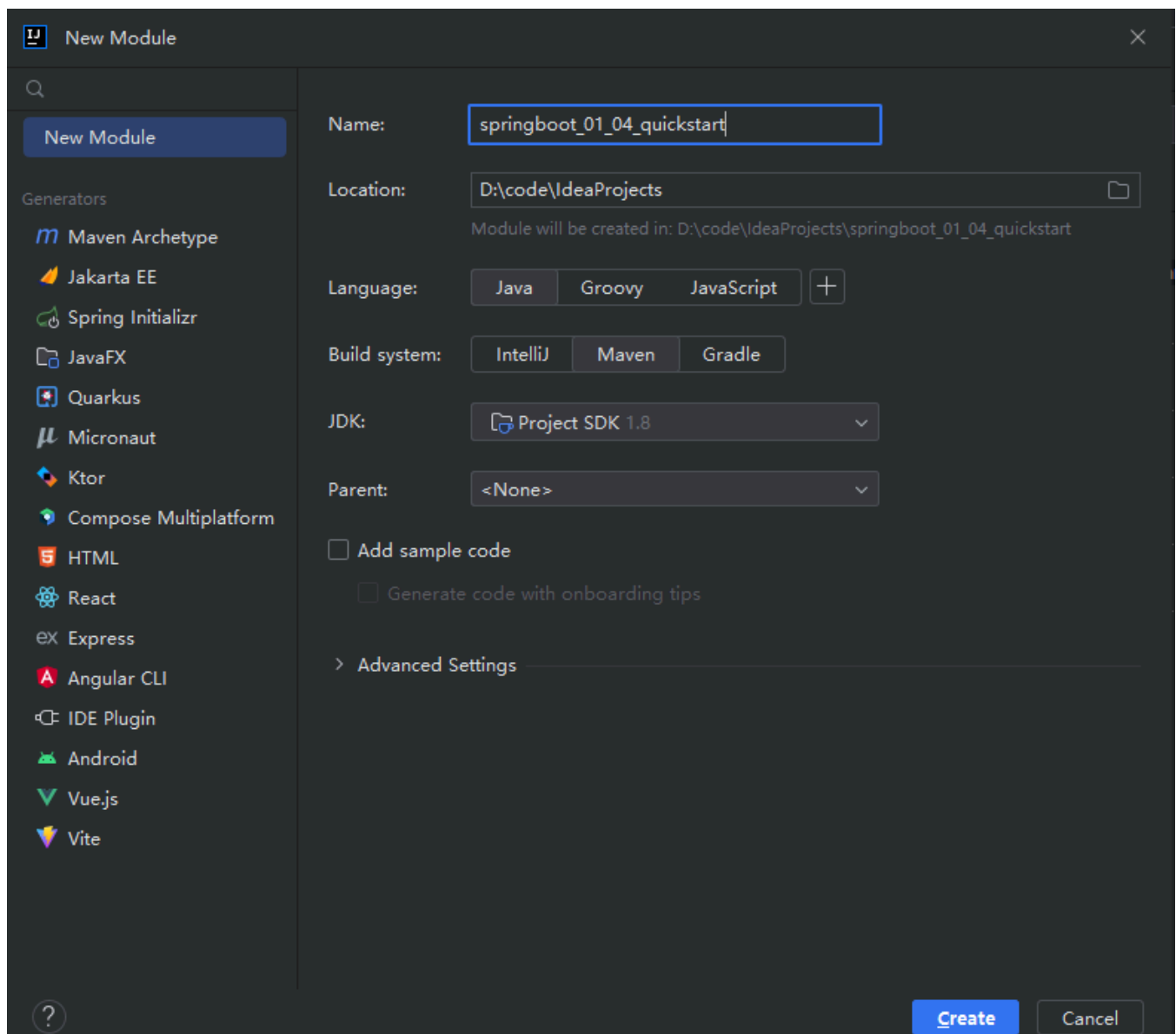
浏览代码 - Ctrl + Space

分享...

其他步骤与前面所述相同。

4. 手工创建SpringBoot项目

正常创建一个Maven项目，



添加继承和依赖,

```

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 http://maven.apach
5       <modelVersion>4.0.0</modelVersion>
6
7       <!--继承-->
8       <parent>
9         <groupId>org.springframework.boot</groupId>
10        <artifactId>spring-boot-starter-parent</artifactId>
11        <version>2.7.11</version>
12      </parent>
13
14      <groupId>org.example</groupId>
15      <artifactId>springboot_01_04_quickstart</artifactId>
16      <version>1.0-SNAPSHOT</version>
17
18      <properties>
19        <maven.compiler.source>8</maven.compiler.source>
20        <maven.compiler.target>8</maven.compiler.target>
21        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
22      </properties>
23
24      <!--依赖-->
25      <dependencies>
26        <dependency>
27          <groupId>org.springframework.boot</groupId>
28          <artifactId>spring-boot-starter-web</artifactId>
29        </dependency>
30      </dependencies>
31
32    </project>

```

创建Application,

```

1  package com.example;
2
3  import org.springframework.boot.SpringApplication;
4  import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6  @SpringBootApplication
7  public class Application {
8
9      public static void main(String[] args) {
10         SpringApplication.run(Application.class);
11     }
12
13 }
14

```

至此，手工创建SpringBoot项目完毕。

5. 简单分析

- SpringBoot是由Pivotal团队提供的全新框架，其设计目的是用来简化Spring应用的初始搭建以及开发过程
 - ◆ Spring程序缺点
 - 依赖设置繁琐
 - 配置繁琐
 - ◆ SpringBoot程序优点
 - 起步依赖（简化依赖配置）
 - 自动配置（简化常用工程相关配置）
 - 辅助功能（内置服务器，.....）

有以下内容需要分析，

- parent
- starter
- 引导类（Application类）
- 内嵌Tomcat

5.1 parent

pom中的parent负责管理依赖以及版本，避免了版本冲突，

```

<!--继承-->
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.7.11</version>
</parent>

```

spring-boot-starter-parent的父依赖是spring-boot-dependencies,

```

<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-dependencies</artifactId>
    <version>2.7.11</version>
</parent>

```

进入spring-boot-dependencies中发现，在properties下，定义了各种依赖的版本，使得我们在SpringBoot开发中，引入pom坐标时，**不需要指定version**，只需要指定其他两项即可。

```

28     <properties>
29         <activemq.version>5.16.6</activemq.version>
30         <antlr2.version>2.7.7</antlr2.version>
31         <appengine-sdk.version>1.9.98</appengine-sdk.version>
32         <artemis.version>2.19.1</artemis.version>
33         <aspectj.version>1.9.7</aspectj.version>
34         <assertj.version>3.22.0</assertj.version>
35         <atomikos.version>4.0.6</atomikos.version>
36         <awaitility.version>4.2.0</awaitility.version>

```

```

6     <dependencyManagement>
7         <dependencies>
8             <dependency>
9                 <groupId>org.apache.activemq</groupId>
10                <artifactId>activemq-amqp</artifactId>
11                <version>${activemq.version}</version>
12            </dependency>
13            <dependency>
14                <groupId>org.apache.activemq</groupId>
15                <artifactId>activemq-blueprint</artifactId>
16                <version>${activemq.version}</version>

```

而在阿里云创建的版本中，并没有使用parent，而是导入了spring-boot-dependencies的依赖，效果相同。

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-dependencies</artifactId>
      <version>${spring-boot.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

不同版本的SpringBoot，它的依赖的版本也存在差异！

5.2 starter

以spring-boot-starter-web为例，它通过依赖传递，导入了web开发所需要的web和webmvc等依赖，

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
    <version>2.7.6</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-json</artifactId>
    <version>2.7.6</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <version>2.7.6</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>5.3.24</version>
    <scope>compile</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.3.24</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

不同的starter通过这种方式，引入了相应的开发所需要的依赖，**简化了依赖的导入**。

当然，starter的作用不止如此，后续继续研究！

5.3 引导类

parent和starter简化了配置，引导类中的main方法，运行了SpringApplication的run方法，它的返回值是一个ApplicationContext的子类，即Spring容器，可以通过它获取Bean。

```
public static ConfigurableApplicationContext run(Class<?>[] primarySources, String[] args) {
    return new SpringApplication(primarySources).run(args);
}
```

```
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        ConfigurableApplicationContext ctx = SpringApplication.run(Application.class);
        BookController bean = ctx.getBean(BookController.class);
        System.out.println(bean);
    }
}
```

注解SpringBootApplication如下,

```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(
    excludeFilters = {@Filter(
        type = FilterType.CUSTOM,
        classes = {TypeExcludeFilter.class}
    ), @Filter(
        type = FilterType.CUSTOM,
        classes = {AutoConfigurationExcludeFilter.class}
    )}
)
public @interface SpringBootApplication {
    @AliasFor(
```

其中，注解ComponentScan未定义扫描路径，默认扫描类所在路径及其子包。

5.4 内嵌Tomcat

spring-boot-starter-web依赖了spring-boot-starter-tomcat，它依赖Tomcat内嵌的核心，Spring将内嵌的Tomcat作为对象运行，并交给Spring容器管理，

```
<dependency>
  <groupId>org.apache.tomcat.embed</groupId>
  <artifactId>tomcat-embed-core</artifactId>
  <version>9.0.74</version>
  <scope>compile</scope>
  <exclusions>
    <exclusion>
      <artifactId>tomcat-annotations-api</artifactId>
      <groupId>org.apache.tomcat</groupId>
    </exclusion>
  </exclusions>
</dependency>
```

可以通过exclusion去除Tomcat的starter，更换为Jetty的starter，达到更换服务器的效果。

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <exclusions>
    <exclusion>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-tomcat</artifactId>
    </exclusion>
  </exclusions>
</dependency>

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

SpringBoot支持以下三款服务器，

- Tomcat
- Jetty
- Undertow