# 1.spring的缺点

Spring虽然有诸多优点，但其配置过于繁杂，让人头大。另一方面，项目依赖的管理也是一件耗时耗力的事情。

Springboot基于约定优于配置的思想，帮助我们快速搭建项目，让我们的spring应用变得更加轻量化，它具有如下优点：

1. **为所有Spring开发者更快的入门**
2. **开箱即用，提供各种默认配置来简化项目配置**
3. **内嵌式容器简化Web项目，直接打成jar包即可运行**
4. **没有冗余代码生成和XML配置的要求**

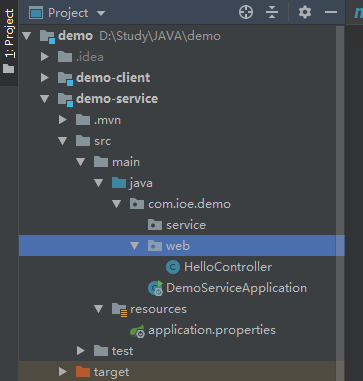
**Spring Boot让我们的Spring应用变的更轻量化。我们不必像以前那样繁琐的构建项目、打包应用、部署到Tomcat等应用服务器中来运行我们的业务服务。通过Spring Boot实现的服务，只需要依靠一个Java类，通过mvn install把它打包成jar，并通过java -jar命令就可以运行起来。这一切相较于传统Spring应用来说，已经变得非常的轻便、简单。**

# 2.快速构建

## 2.1 Spring Initializr工具

Spring官方提供了工具Spring Initializr来帮助创建Spring Boot应用，其地址为：<https://start.spring.io/>，按照提示内容填好，添加依赖后生成并下载项目，导入编译器

项目结构图如下：



Spring Boot的基础结构共三个文件：

1. src/main/java下的程序入口：DemoApplication
2. src/main/resources下的配置文件：application.properties
3. src/test/下的测试入口：DemoApplicationTests

**注意，入口文件应当放在根目录下，这样才能扫描到其他文件**

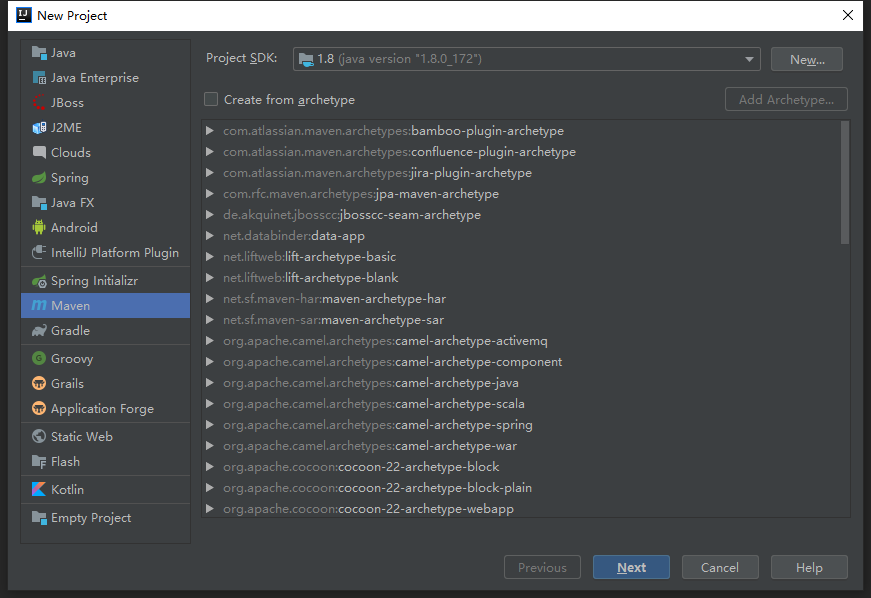
编写HelloController文件：

@RestController  
public class HelloController {  
 @RequestMapping("/hello")  
 public String index() {  
 return "Hello World";  
 }  
}

访问localhost:8080/hello即可看到对应的页面

## 2.2 手动创建

1.创建一个maven项目



2.导入依赖

Parent-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>com.dj.springboot</groupId>  
 <artifactId>springboot-study</artifactId>  
 <packaging>pom</packaging>  
 <version>1.0-SNAPSHOT</version>  
  
 <modules>  
 <module>springboot-study-client</module>  
 <module>springboot-study-service</module>  
 </modules>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.0.6.RELEASE</version>  
 </parent>  
  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.3</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 <fork>true</fork>  
 <verbose>true</verbose>  
 <encoding>UTF-8</encoding>  
 <meminitial>256m</meminitial>  
 <maxmem>1024m</maxmem>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

Service-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">  
 <parent>  
 <artifactId>springboot-study</artifactId>  
 <groupId>com.dj.springboot</groupId>  
 <version>1.0-SNAPSHOT</version>  
 </parent>  
 <modelVersion>4.0.0</modelVersion>  
  
 <artifactId>springboot-study-service</artifactId>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.dj.springboot</groupId>  
 <artifactId>springboot-study-client</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 </dependency>  
 </dependencies>  
</project>

3.编写主程序

*/\*\*  
 \* Author:RicardoDing  
 \* Date :2020/8/18  
 \* Time :15:20  
 \* Desc :主程序，springboot启动入口  
 \*/*@SpringBootApplication  
public class MainApplication {  
 public static void main(String[] args) {  
 //启动springboot应用  
 SpringApplication.*run*(MainApplication.class, args);  
 }  
}

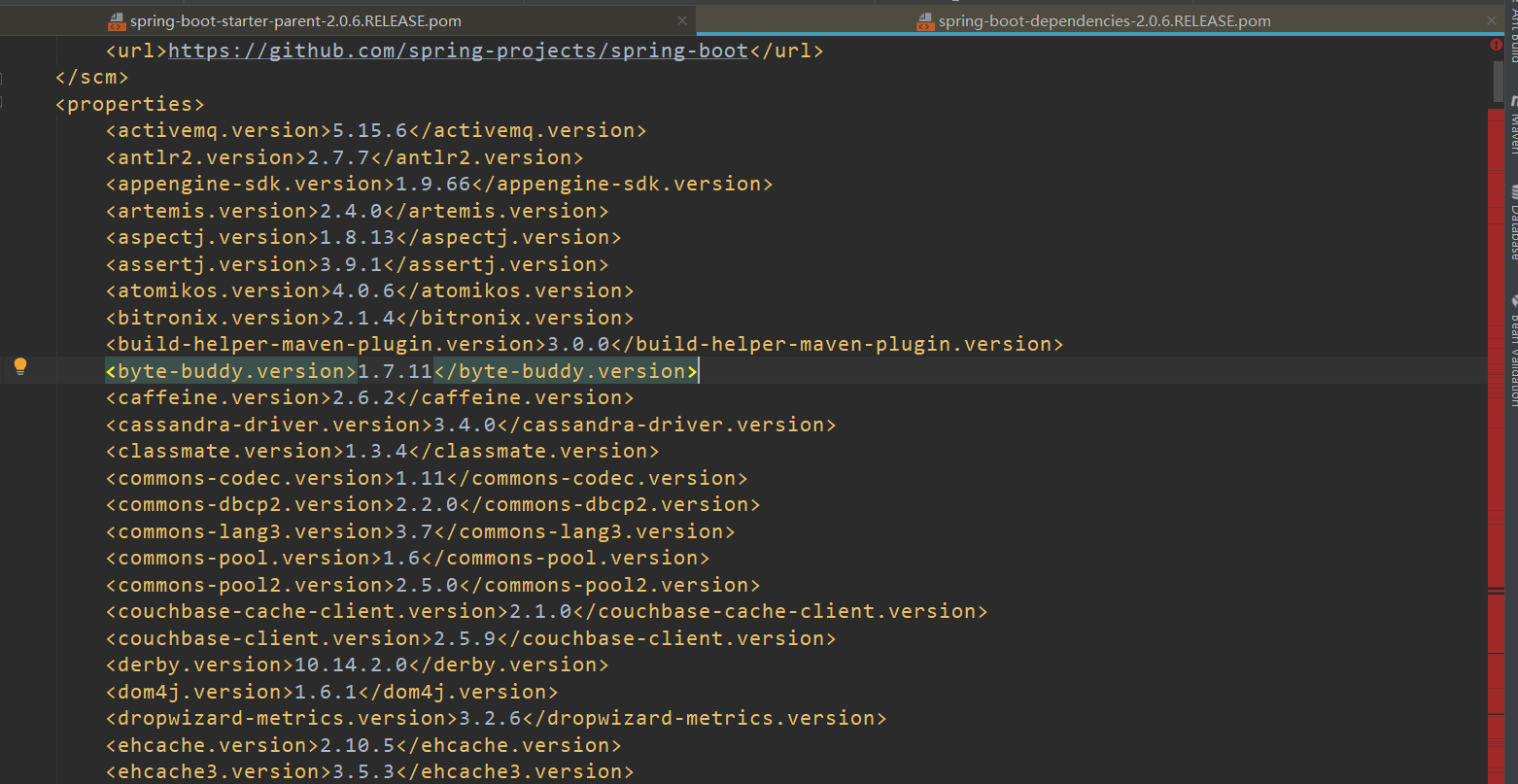
## 2.3 打包发布

在parent的pom文件中，加入如下配置：

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.3</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 <fork>true</fork>  
 <verbose>true</verbose>  
 <encoding>UTF-8</encoding>  
 <meminitial>256m</meminitial>  
 <maxmem>1024m</maxmem>  
 </configuration>  
 </plugin>  
 </plugins>  
</build>

## 2.4 springboot的依赖管理

上面我们可以知道，springboot引入了一个父项目，在这个父项目中，springboot帮助我们对常用的依赖包进行了版本管理，在<artifactId>spring-boot-starter-parent</artifactId>的父项目 <artifactId>spring-boot-dependencies</artifactId>中我们可以看到springboot的版本限定：



如果希望使用自定义版本号，则需要在当前项目的pom文件中，通过覆盖 祖先项目 的<properties>属性来完成

# 3.配置springboot

从快速构建一节中我们可以看到，虽然我们的Controller代码和spring一致，但我们几乎未进行任何配置就让一个springboot的helloworld项目运行起来了，这就是springboot无需配置的优点。但我们仍要学习如何配置springboot以应对特殊的场景需求

src/main/resources目录是Spring Boot的配置目录，其配置的文件名默认为application.properties，几乎所有的配置内容都可以集中在该文件中。除了可以使用传统的properties文件进行配置以外，我们还可以采用YAML文件进行配置

## 3.1自定义属性

application.properties中除了可以定义springboot的属性，我们还可以自定义属性，如：

book.name=SpringBootApp  
book.author=QiuKu

通过@Value注解可以加载这部分自定义属性，eg：

@Value("${book.name}")  
private String name;

甚至自定义属性之间也可以通过占位符**${}**互相引用

book.name=SpringBootApp  
book.author=QiuKu  
book.desc=The book is by ${book.author}

## 3.2多环境配置

我们在开发任何应用的时候，通常同一套程序会被应用和安装到几个不同的环境，比如：开发、测试、生产等。其中每个环境的数据库地址、服务器端口等等配置都会不同，如果在为不同环境打包时都要频繁修改配置文件的话，那必将是个非常繁琐且容易发生错误的事。

我们可以通过为不同环境配置多个配置文件，并在打包时进行区分打包来解决这个问题。

在配置多环境配置时，多个环境的配置文件需要满足**application-{profile}.properties**的格式，其中**{profile}**对应你的环境标识

eg：

**application-dev.properties：开发环境**

**application-test.properties：测试环境**

**application-product.properties：生产环境**

至于加载哪一个具体的配置文件，可以在application.properties文件中通过spring.profiles.active属性来设置，其值对应配置文件中的{profile}值。如：spring.profiles.active=test就会加载application-test.properties配置文件内容。

