# springboot学习

## springboot优缺点

### 优点：

– 快速创建独立运行的Spring项目以及与主流框架集成

– 使用嵌入式的Servlet容器，应用无需打成WAR包

– starters自动依赖与版本控制

– 大量的自动配置，简化开发，也可修改默认值

– 无需配置XML，无代码生成，开箱即用

– 准生产环境的运行时应用监控

– 与云计算的天然集成

### 缺点

如果对spring不是很熟悉那么对springboot应该也是入门容易精通难

## springboot简介

简化spring应用开发的一个框架

整个spring技术栈的一个大整合

J2EE开发的一站式解决方案

## 微服务

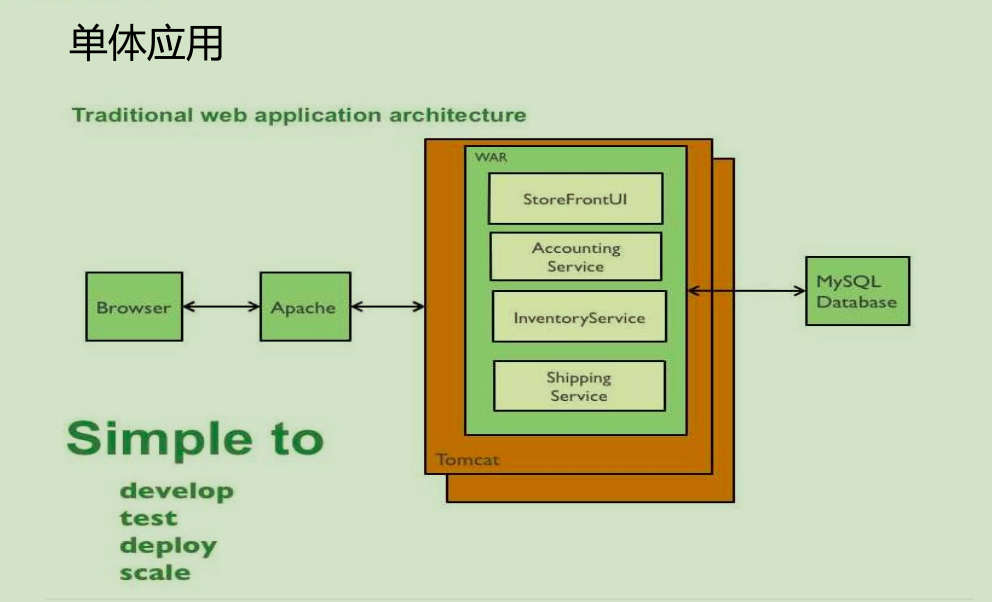
**Martin Fowler关于微服务的原文翻译**

<https://blog.csdn.net/u013970991/article/details/53333921>

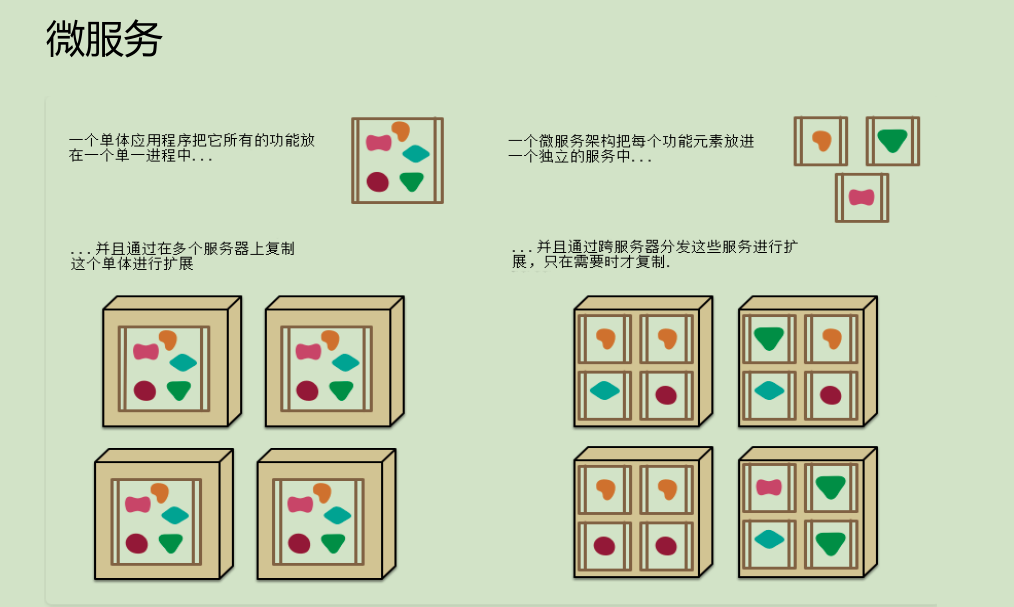
微服务:是一个架构风格

一个应用应该是一组小型服务:可以通过HTTP的方式进行胡同;

单体应用:ALL IN ONE ,缺点:修改一个功能可能就会影响整个应用程序的部署,需要重新启动tomcat



微服务:每一个功能元素最终都是一个可独立替换和独立升级的软件单元



## 环境搭建

环境约束

– jdk1.8

– maven3.x

– eclipse

– Spring Boot 1.5.9.RELEASE

给maven的setting.xml的profiles标签添加如下内容(设置maven编译java环境为jdk1.8)

|  |
| --- |
| <profile>  <id>jdk-1.8</id>  <activation>  <jdk>1.8</jdk>  </activation>  <properties>  <maven.compiler.source>1.8</maven.compiler.source>  <maven.compiler.target>1.8</maven.compiler.target>  <maven.compiler.compilerVersion>1.8</maven.compiler.compilerVersion>  </properties>  </profile> |

## springboot-helloworld

### 创建maven工程(jar工程)

### 导入springboot相关依赖

在pom.xml中添加如下内容

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.9.RELEASE</version>  </parent>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  </dependencies> |

### 编写主程序App类:作用启动springboot应用

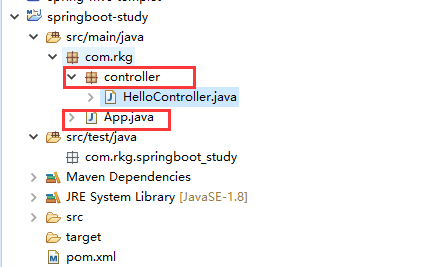
|  |
| --- |
| /\*\*  \*  \***@SpringBootApplication来标注一个主程序类**,说明这是一个springboot应用  \*/  @SpringBootApplication  **public** **class** App {  **public** **static** **void** main(String[] args) {  // spring应用启动入口,固定写法  SpringApplication.*run*(App.**class**, args);  }  } |

### 编写controller

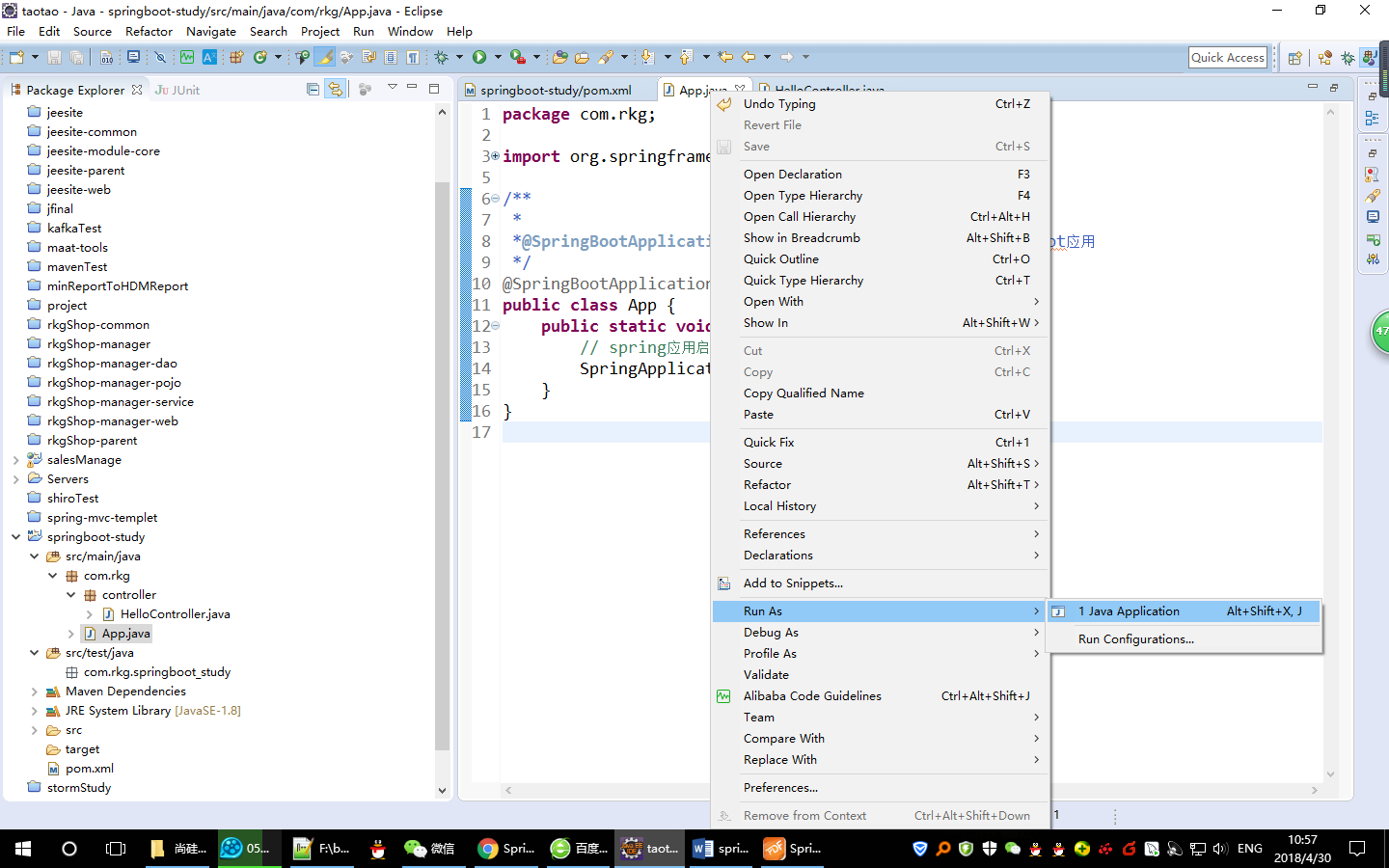
|  |
| --- |
| @Controller  **public** **class** HelloController {  @ResponseBody  @RequestMapping("hello")  **public** String hello() {  **return** "hello world";  }  } |

### 启动项目

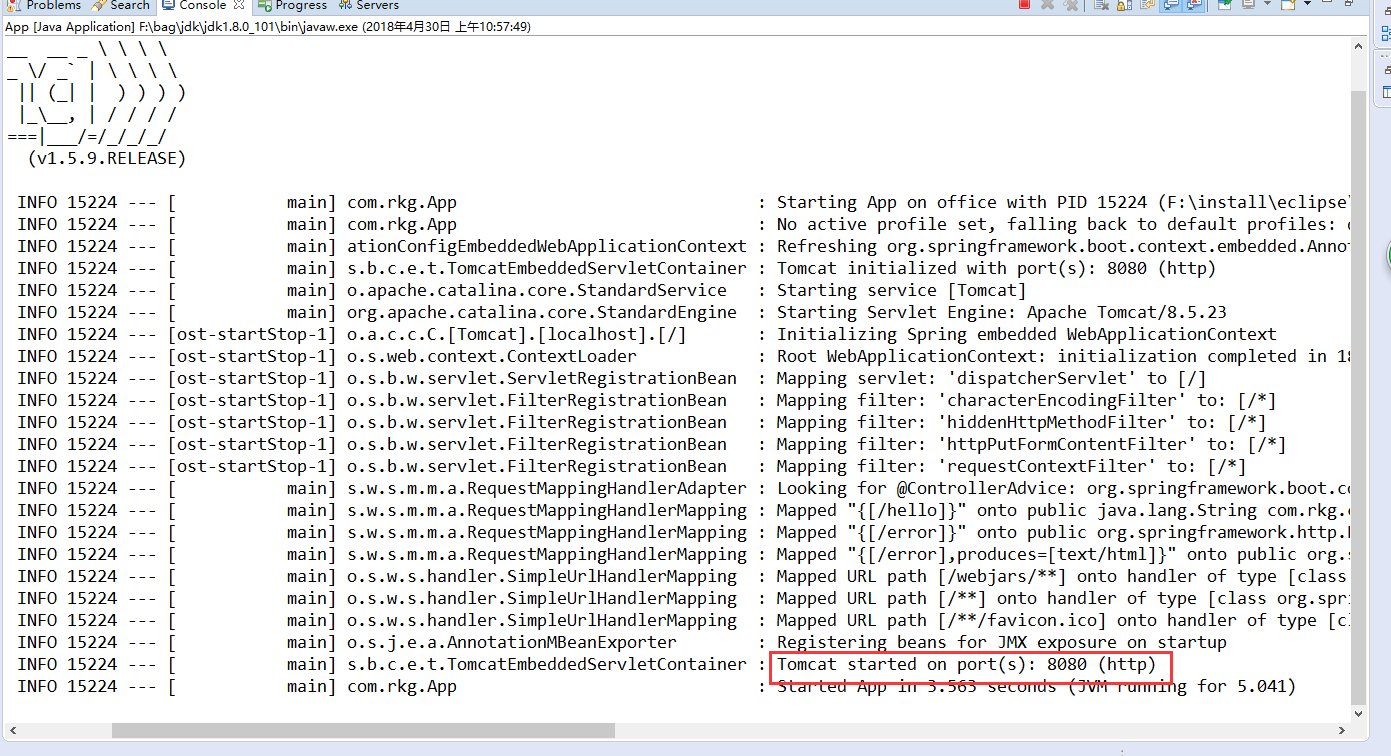
注意App.java必须在整个项目第一级(在最上面的包里面)

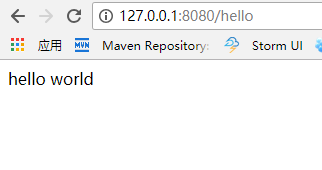


在app.java右键Run As🡪java application启动项目



控制台输出在8080端口启动tomcat成功,springboot内置有tomcat



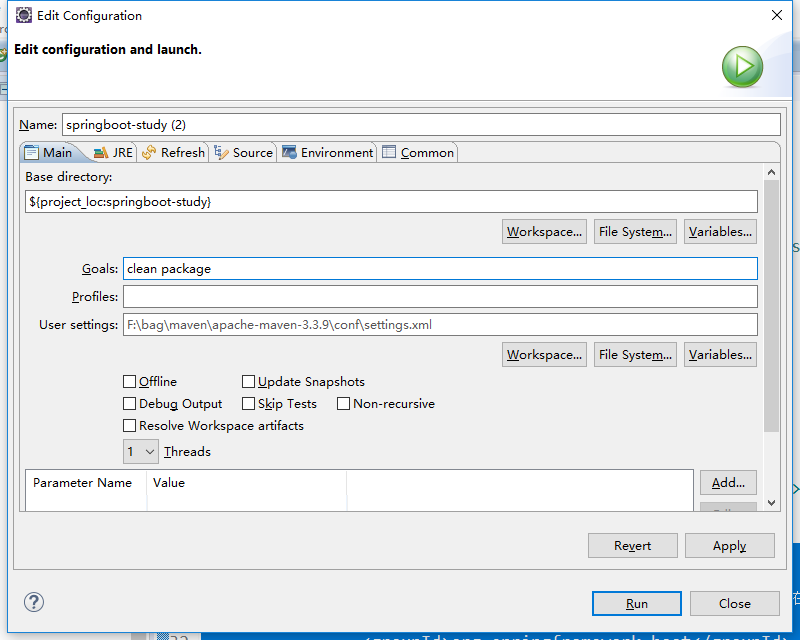


### 简化部署

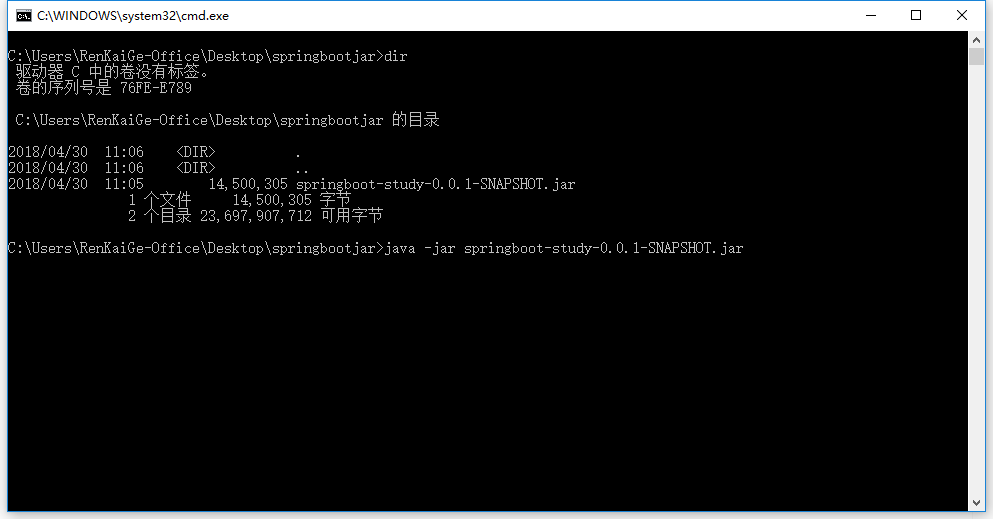
#### 在pom.xml中添加

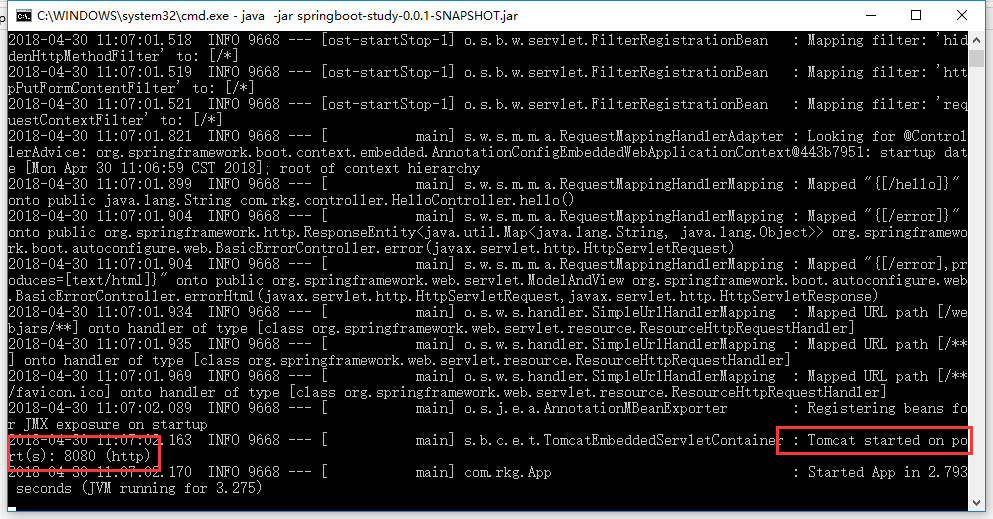
|  |
| --- |
| <build>  <plugins>  <!-- 这个插件可以将springboot应用打包成一个可执行的jar文件,无需在服务器安装配置tomcat容器 -->  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build> |

#### 输入打包命令在target目录下生成了jar文件



#### 运行jar文件





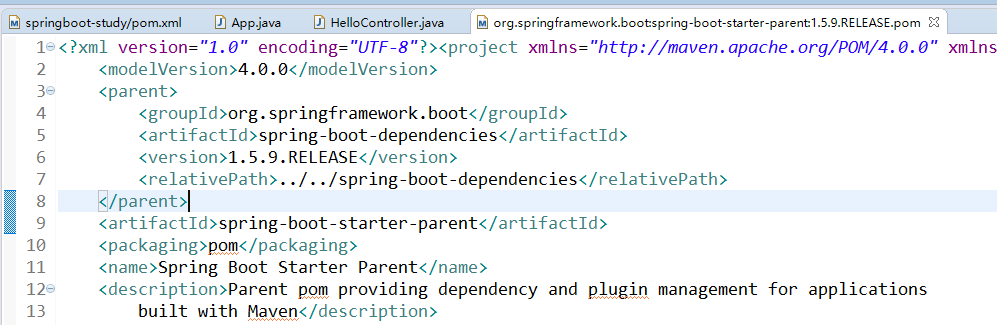
## springboot-helloworld探究

### 首先从pom.xml说起

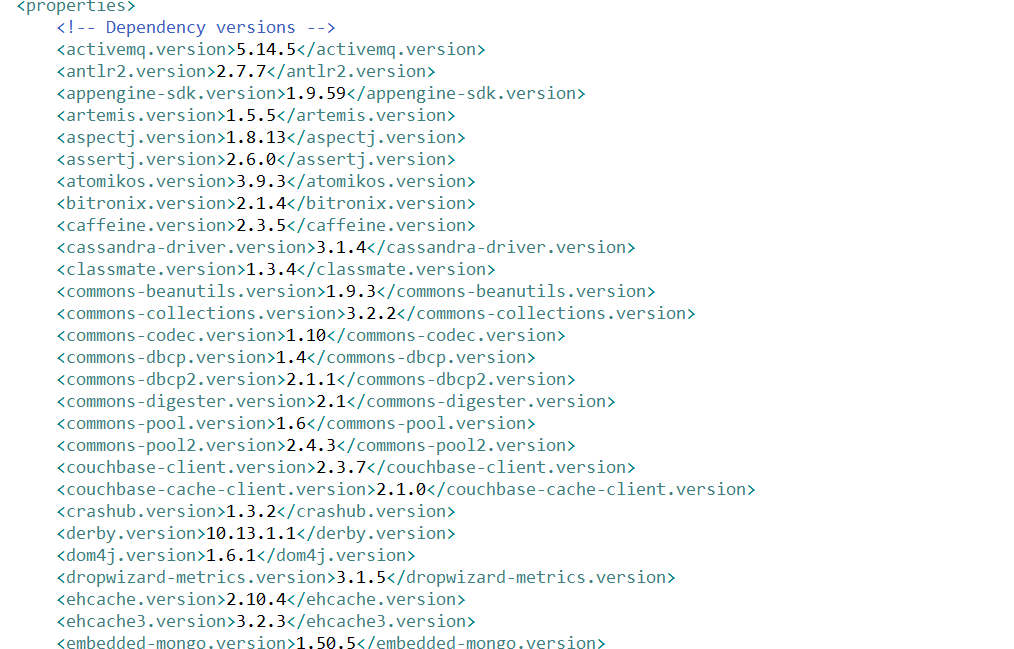
#### 发现里面有一个parent项目

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.9.RELEASE</version>  </parent> |

##### 点进去发现他也有一个父项目



##### 点进去会发现里面定义了好多依赖,这个pom文件管理了springboot的依赖



##### 所以我们在使用依赖时不用写版本号了,(spring-boot-dependencies中未定义的依赖还是需要写版本号)

#### 导入的依赖

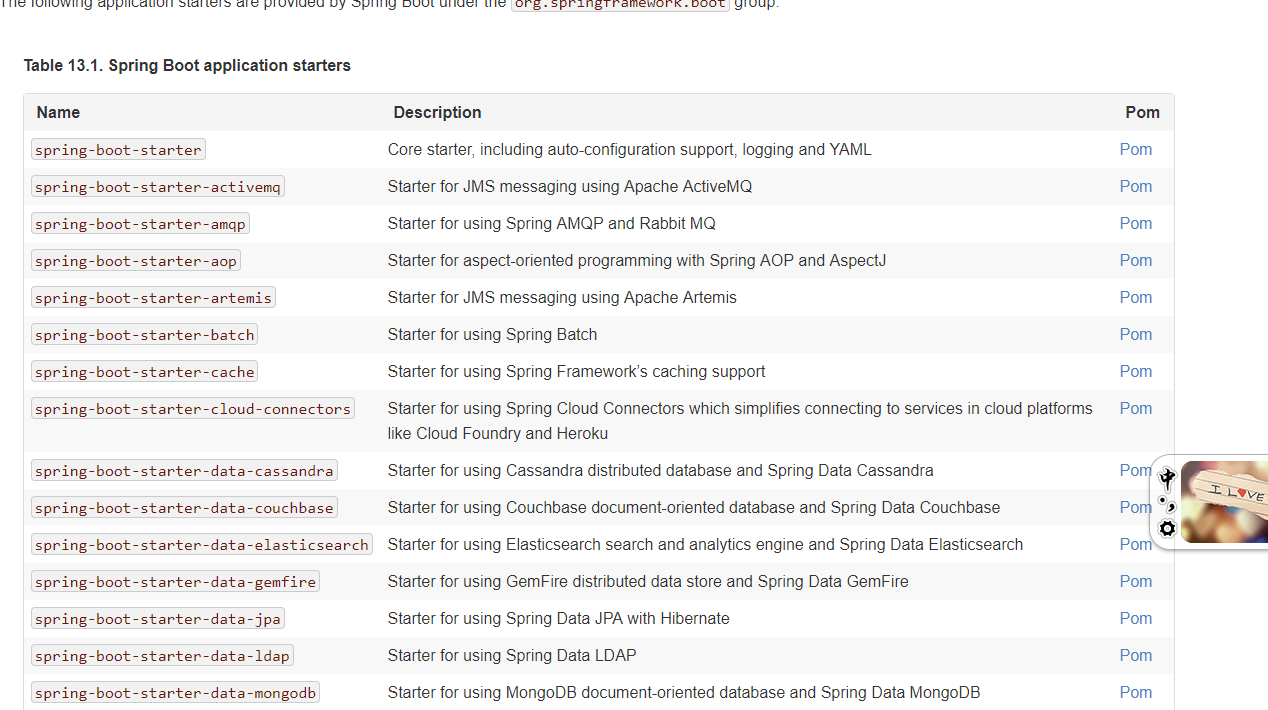
|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  </dependencies> |

spring-boot-starter-web:

spring-boot-starter是场景启动器,它帮我们导入web正常运行需要导入的依赖

springboot将所有的功能场景都抽取出来,做成了一个个staters(启动器),使用时只需要将对应的staters引用过来,其对应的依赖就会自动引用过来,要用什么功能就导入什么样的启动器

具体starters参照

[https://docs.spring.io/spring-boot/docs/1.5.9.RELEASE/reference/htmlsingle/#using-boot-starter](https://docs.spring.io/spring-boot/docs/1.5.9.RELEASE/reference/htmlsingle/#using-boot-starter ) 

### 主程序类(主入口类)

|  |
| --- |
| /\*\*  \*  \***@SpringBootApplication来标注一个主程序类**,说明这是一个springboot应用  \*/  @SpringBootApplication  **public** **class** App {  **public** **static** **void** main(String[] args) {  // spring应用启动入口,固定写法  SpringApplication.*run*(App.**class**, args);  }  } |

**SpringBootApplication:springboot引用标注在某个类上说明这个类是springboot的主配置类,springboot就应该运行这个类的main方法来启动springboot应用**

|  |
| --- |
| **@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 {** |

#### SpringBootConfiguration

**springboot的配置类,标注在某个类上,表示这是一个springboot的配置类;**

|  |
| --- |
| @Target({ElementType.***TYPE***})  @Retention(RetentionPolicy.***RUNTIME***)  @Documented  @Configuration  **public** **@interface** SpringBootConfiguration { |

Configuration表名这是一个配置类

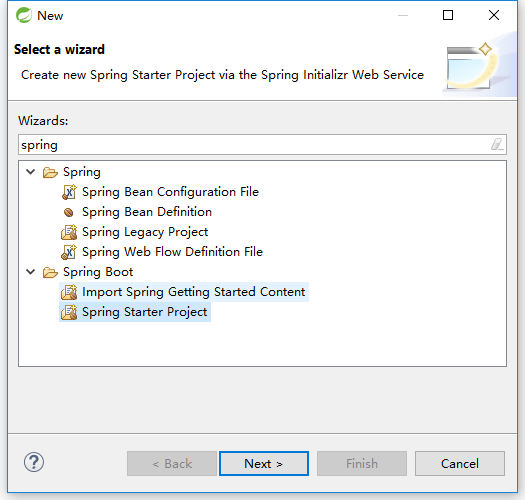
|  |
| --- |
| @Target({ElementType.***TYPE***})  @Retention(RetentionPolicy.***RUNTIME***)  @Documented  @Component  **public** **@interface** Configuration {  String value() **default** "";  } |

#### EnableAutoConfiguration

告诉springboot开启自动配置功能,

## 使用eclipse快速创建springboot应用

### 使用sts创建spring stater project



# 配置文件

springboot使用一个全局配置文件(文件名称是固定的)

application.properties或者application.yml

配置文件的作用是:修改springboot的自动配置的默认值

比如修改tomcat端口号

server.port=8888