SpringBoot的本质还是Spring。要想使用SpringBoot开发web项目，就一定要使用servlet3.0规范(没有web.xml)。要想使用SpringBoot，则jdk的版本一定要是1.6及以上版本。
SpringBoot的1.x版本和2.x版本是没有关系的，它们是不同的两个分支。SpringBoot的1.x版本和2.x版本有很大的区别。
SpringBoot和Spring的不同之处：Spring用的日志包是log4j，SpringBoot用的日志包是logback。

org.springframework.boot
spring-boot-starter-web

//maven核心程序根据这个给当前项目(可知是一个SpringBoot项目)的所有jar包的引用中含有与spring，springmvc，Tomcat相关的jar包的引用，其中与
Tomcat相关的jar包的引用引用的jar包组成了一个Tomcat服务器，当执行当前项目中的被@SpringBootApplication修饰的类中的main方法时，系统就会将当前项目部
署到这个Tomcat服务器上，并且启动这个Tomcat服务器。（不需要我们往这个Tomcat上部署当前项目和启动这个Tomcat服务器）。

只要项目中用到了与SpringBoot有关的东西，则这个项目就是一个SpringBoot项目。每一个SpringBoot项目中一定要有且仅有一个类是被@SpringBootApplication修饰的，并且这个被修饰的类的类名要以Application结尾(这是规范)，并且在这个被修饰的类中要有一个main方法，main方法中的内容是" SpringApplication.run(@SpringBootApplication修饰的类.class, args); "。
如果一个SpringBoot项目是一个Maven项目，则在这个SpringBoot项目的pom.xml中就一定要有：
org.springframework.boot
spring-boot-starter-parent
1.5.17.RELEASE

SpringBoot项目中不能出现.xml文件，所以SpringBoot项目中的配置文件只能是: .properties文件和.yml文件(更推荐)。 一定要取名为application.properties或application.yml。
当执行@SpringBootApplication修饰的类中的main方法时，SpringBoot框架会去读取src/main/resources/路径或类路径下/config路径中的application.properties文件或application.yml文件中的内容。

如果要创建一个Maven的SpringBoot项目，则在创建的时候只能选jar，不能选war。

SpringBoot框架内部含有许多由@Configuration修饰的类。就是由于有这些类的存在我们省去了很多的配置工作。

1.5.17版本SpringBoot流程分析:

```
SpringApplication.run(@SpringBootApplication修饰的类.class, args);  的实质就是： new SpringApplication(new Object[] { @SpringBootApplication修饰的类.class }).run(args);
构造方法SpringApplication(Object... sources)中调用了initialize(sources);
在initialize(sources); 方法中做了4件事：1. 判断当前项目是不是一个web项目，判断依据是：当前项目中是否含有类javax.servlet.Servlet和
类
org.springframework.web.context.ConfigurableWebApplicationContext。只要当前项目中含有这两个类，就认为当前项目是一个web项目。

2. 查找出项目中所有的META-INF/spring.factories文件。然后到每个spring.factories文件中获取以" ApplicationContextInitializer " 为键的
```

值。这些得到的值的内容就是各个ApplicationContextInitializer实现类的类名。然后根据这些ApplicationContextInitializer实现类的类名利
用反射创建出这些ApplicationContextInitializer实现类的对象。然后将这些ApplicationContextInitializer实现类的对象设置到当前
SpringApplication对象的成员变量initializers中。

注：
源码中的loadFactoryNames方法的上面地注释：param factoryClass the interface or abstract class representing the factory 的意思是
方法参数factoryClass用于接收一个能表示工厂的接口或抽象类。ApplicationContextInitializer类并不是工厂类，但是
ApplicationContextInitializer类和它的各个实现类都能表示一个工厂类，所以会将ApplicationContextInitializer类和它的各个实现类也称
为factoryClass。同样，ApplicationListener类也不是工厂类，但是ApplicationListener类和它的各个实现类都能表示一个工厂类。

3. 查找出项目中所有的META-INF/spring.factories文件。然后到每个spring.factories文件中获取以" ApplicationListener " 为键的
值。这些得到的值的内容就是各个ApplicationListener实现类的类名。然后根据这些ApplicationListener实现类的类名利用反射创建出这些
ApplicationListener实现类的对象。然后将这些ApplicationContextInitializer实现类的对象设置到当前SpringApplication对象的成员变
量listeners中。

4. 获取当前项目中的含有man方法的类，然后将这个类.class赋给当前SpringApplication对象的成员变量mainApplicationClass。

在run(String... args); 方法中做了这些事：创建IOC容器对象(如果之前判断出当前项目是一个web项目，则创建出的IOC容器对象就是AnnotationConfigEmbeddedWebApplicationContext对象，
如果之前判断出当前项目不是一个web项目，则创建出的IOC容器对象就是AnnotationConfigApplicationContext对象。)

注：只有所属类名是以AnnotationConfig开头的IOC容器对象能够认识@Configuration、@Bean ... 。

@SpringBootApplication注解：可以知道@SpringBootApplication被@SpringBootConfiguration，@EnableAutoConfiguration，@ComponentScan修饰了，这样的效果就是:
@SpringBootConfiguration，
@EnableAutoConfiguration，@ComponentScan隐式修饰了被@SpringBootApplication修饰的类。

@SpringBootConfiguration : 当一个所属类名是以AnnotationConfig开头的IOC容器对象被创建之后，这个IOC容器对象会去扫描@SpringBootApplication修饰的类上的@ComponentScan的value
属性值指定的包及这个包的子包中的被@Configuration/@Component/@Controller/@Service/@Repository修饰的类。当IOC容器扫描被@Configuration修饰的类时，会
看这个类中有哪些被@Bean修饰的方法上的@ConditionalXXX(一个方法上的@ConditionalXXX还包括了它所在的类的类名上的@ConditionalXXX)是能够被满足的，然后去
调用上面的@ConditionalXXX是能被满足的所有被@Bean修饰的方法。并将这些方法的返回值(是对象)都放到自己内部。可以利用
getBean(String name, Class requiredType)方法从IOC容器对象中获取到内部存放的一个被@Bean修饰的方法的返回值，传给参数name的值是这个被@Bean修饰的方法
的方法名。
补充说明： 要满足@ConditionalOnClass 就是 当前项目中要含有它的value属性值指定的所有类。

如果IOC容器对象所属的类不是以AnnotationConfig开头的，则这个IOC容器对象在扫描时是不会扫描被@Configuration修饰的类的。

被@SpringBootConfiguration修饰的类是被@Configuration隐式修饰的(@SpringBootConfiguration被@Configuration修饰)。@Configuration和@SpringBootConfiguration
可以互换。如果当前项目是一个SpringBoot项目，最好就只使用@SpringBootConfiguration而不要去使用@Configuration，这是规范。

@Bean： @Bean只能在被@Configuration修饰的类中使用，被@Bean修饰的方法必须要返回一个对象。只有所属类名是以AnnotationConfig开头的IOC容器对象能够认识
@Configuration、@Bean ... 。

注：被@Bean修饰的方法返回的对象并不是由IOC容器对象创建出来的，而是由我们自己创建出来的，从这里也可以看出在IOC容器中的对象并不一定就是由IOC容
器对象创建出来的。

@ComponentScan ： 如果在被@SpringBootApplication修饰的类的上方没有写@ComponentScan，那么被@SpringBootApplication修饰的类就是被@SpringBootApplication类上的@ComponentScan隐式
修饰，@SpringBootApplication类上的@ComponentScan的value属性值是，意思就是被@SpringBootApplication修饰的类所在的包。

@EnableAutoConfiguration ： @EnableAutoConfiguration类中内部引入了EnableAutoConfigurationImportSelector类，EnableAutoConfigurationImportSelector类的父类是
AutoConfigurationImportSelector类，在AutoConfigurationImportSelector类中有一个selectImports()方法，IOC容器会调用这个selectImports()方法。
selectImports()方法的作用是去获取SpringBoot框架自己定义的各个被@Configuration修饰的类的类名。(获取的过程是：selectImports()方法查找出
项目中所有的META-INF/spring.factories文件。然后到每个spring.factories文件中获取以" EnableAutoConfiguration " 为键的值。这些得到的值的
内容就是SpringBoot框架自己定义的各个被@Configuration修饰的类的类名。)根据这些类名，IOC容器会找到这些类，然后看各个类中有哪些被@Bean修饰的方法
上的@ConditionalXXX(一个方法上的@ConditionalXXX还包括了它所在的类的类名上的@ConditionalXXX)是能够被满足的，然后去调用上面的@ConditionalXXX是能被满
足的所有被@Bean修饰的方法。并将这些方法的返回值(是对象)都放到自己内部。

SpringBoot项目中是没有web.xml的，DispatcherServlet是由SpringBoot框架进行配置，而不是由我们进行配置。可以在日志信息中查看到SpringBoot框架对DispatcherServlet配置的信息：
2018-11-19 15:26:45.047 INFO 7260 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]

@RestController : 等价于 @Controller + @ResponseBody ， 只有当@Controller修饰的类中的所有方法都是被@ResponseBody修饰的时，才能在类上用@RestController替换@Controller(同时要将类中的所有@ResponseBody去掉)。

采用分布式系统架构的项目中，如果A系统要访问B系统，则A是发送Rest风格的URL到B系统，B系统会返回Json数据给A系统。???

Mybatis框架定义了@Select，@Insert，@Update，@Delete注解，用于写在Dao接口中的方法的上方。
标在Dao接口方法上的@Select(" xxx ") 的作用等价于 给这个Dao接口写了一个sql映射文件，并且在这个sql映射文件中写了一个

----------springmvc + springboot + mybatis整合 ----------

1.先创建出一个SpringBoot项目(Maven项目)

2. 在pom.xml中写上：

```
org.springframework.boot
spring-boot-starter-web
```

3.

```
mysql
mysql-connector-java

com.alibaba
druid
1.0.5

org.mybatis.spring.boot
mybatis-spring-boot-starter
1.1.1
```

4. 在src/main/resources下创建application.yml，在里面写上：

```
---                注意:一定只能写三个 - 。
spring:
  datasource:      注意:首字母一定要缩进两个字母的距离。
    name: mydb
    type: com.alibaba.druid.pool.DruidDataSource
    url: jdbc:mysql://127.0.0.1:3306/atcrowdfunding
    username: root
    password: root
    driver-class-name: com.mysql.jdbc.Driver
mybatis:
  mapper-locations: classpath:/mybatis/mapper-.xml
  type-aliases-package: com.atguigu.**.bean
```

5. 在被@SpringBootApplication修饰的类的类名上方写上：@MapperScan("Dao接口所在的包的包名")

   @MapperScan("Dao接口所在的包的包名")

   等价于Spring 整合 mybatis 时写在Spring.xml中的

```
//basePackage属性用于指定Mapper接口所在的包。
```

6. 在被@SpringBootApplication修饰的类的类名上方写上：@EnableTransactionManagement

   @EnableTransactionManagement

   等价于写在Spring.xml中的

<tx:annotation-driven transaction-manager="transactionManager"/>

----------草稿----------

@Bean只能在被@Configuration修饰的类中使用，被@Bean修饰的方法必须要返回一个对象，返回的这个对象会被系统放入IOC容器中。(可以知道返回的这个对象并不是
由IOC容器对象创建出来的，是由我们自己创建出来的，所以从这里也可以看出在IOC容器中的对象并不一定就是由IOC容器对象创建出来的。)系统会调用@Configration
修饰的类中的被@Bean修饰的各个方法，然后将这些方法的返回值都放到IOC容器对象中。可以利用getBean(String name, Class requiredType)从IOC容器对象中获取
到被系统放入的一个@Bean修饰的方法的返回值，传给参数name的值是这个@Bean修饰的方法的方法名。

```
@ComponentScan : @ComponentScan(excludeFilters = {
    @Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),
    @Filter(type = FilterType.CUSTOM, classes =
AutoConfigurationExcludeFilter.class) })是什么意思？
```

要有被@Configuration修饰的类的原因是它可以起到和配置文件一样的作用。

要有被@Configuration修饰的类的原因是它可以起到和配置文件一样的作用

@SpringBootConfiguration：一个被@Configuration修饰的类就等价于一个配置文件（注：如果一个被@Configuration修饰的类也被@ConditionalOnClass修饰了，如果项目中不含有
@ConditionalOnClass的value属性值指定的所有类，就等于在这个项目中不含有这个被@Configuration修饰的类。
项目中的一个被@Configuration修饰的类的上方如果有@ConditionalXXX，只要没有满足这个@ConditionalXXX所指定的条件，就等于在这个项目中不含有这个
被@Configuration修饰的类）。
@SpringBootConfiguration和@Configuration是等价的(@SpringBootConfiguration被@Configuration修饰)，@SpringBootConfiguration修饰的类等价于一个配置文件。
如果当前项目是一个SpringBoot项目，最好就只使用@SpringBootConfiguration而不要去使用@Configuration，这是规范。

```
@Bean只能在被@Configuration修饰的类中使用，被@Bean修饰的方法必须要返回一个对象。只有所属类名是以AnnotationConfig开头的IOC容器对象能够认识
@Configuration、@Bean ... 。当一个所属类名是以AnnotationConfig开头的IOC容器对象被创建之后，这个IOC容器对象会去扫描@SpringBootApplication修饰的类
上的@ComponentScan的value属性值指定的包及这个包的子包中的被@Configuration/@Component/@Controller/@Service/@Repository修饰的类。当IOC容器扫描被
@Configuration修饰的类时，会对这个类进行解析，然后
调用@Configration修饰的类中的被@Bean修饰的各个方法，然后将这些方法的返回值(是对象)都放到自己中。可以利用getBean(String name, Class<T> requiredType)
从IOC容器对象中获取到被系统放入的一个@Bean修饰的方法的返回值，传给参数name的值是这个@Bean修饰的方法的方法名。

注：被@Bean修饰的方法返回的对象并不是由IOC容器对象创建出来的，而是由我们自己创建出来的，从这里也可以看出在IOC容器中的对象并不一定就是由IOC容器对象创
建出来的。
```

当一个所属类名是以AnnotationConfig开头的IOC容器对象被创建之后，这个IOC容器对象会去扫描@SpringBootApplication修饰的类上的@ComponentScan的value
属性值指定的包及这个包的子包中的被@Configuration/@Component/@Controller/@Service/@Repository修饰的类。当IOC容器扫描被@Configuration修饰的类时，会
看这个类中有哪些被@Bean修饰的方法上的@ConditionalXXX(一个方法上的@ConditionalXXX还包括了它所在的类的类名上的@ConditionalXXX)是能够被满足的，然后去
调用上面的@ConditionalXXX是能被满足的所有被@Bean修饰的方法。并将这些方法的返回值(是对象)都放到自己内部。可以利用getBean(String name, Class requiredType)
方法从IOC容器对象中获取到内部存放的一个@Bean修饰的方法的返回值，传给参数name的值是这个@Bean修饰的方法的方法名。
补充说明： @ConditionalOnClass指定的条件就是当前项目中要含有它的value属性值指定的所有类。
如果IOC容器对象所属的类不是以AnnotationConfig开头的，则这个IOC容器对象在扫描时是不会扫描被@Configuration修饰的类的。

看这个类有没有被@ConditionalXXX修饰，如果有就会看有没有满足@ConditionalXXX指定的条件。如果满足了这个类的类名上的所有@ConditionalXXX指定的条件，
IOC容器对象就会到这个类中看。到类中看