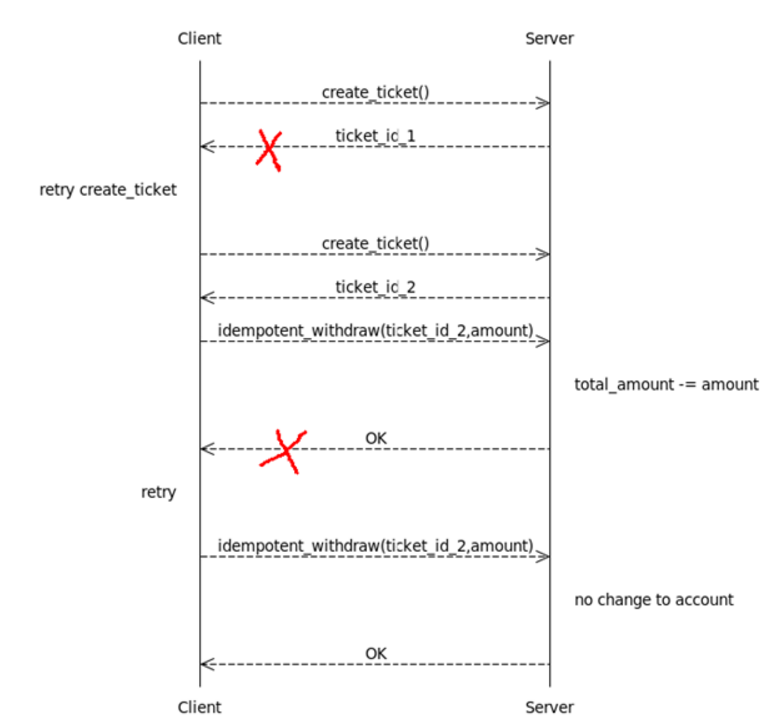
## HTTP幂等性

HTTP方法的幂等性是指一次和多次请求某一个资源应该具有同样的副作用。即每次请求的结果是一样的，GET表示获取操作，是幂等的；PUSH是更新操作，是幂等的(更新后，多次提交结果任然不变，即更新后的结果)；DELETE是删除操作，和PSUH类似；POST保存操作，多次提交，多次保存，非幂等。

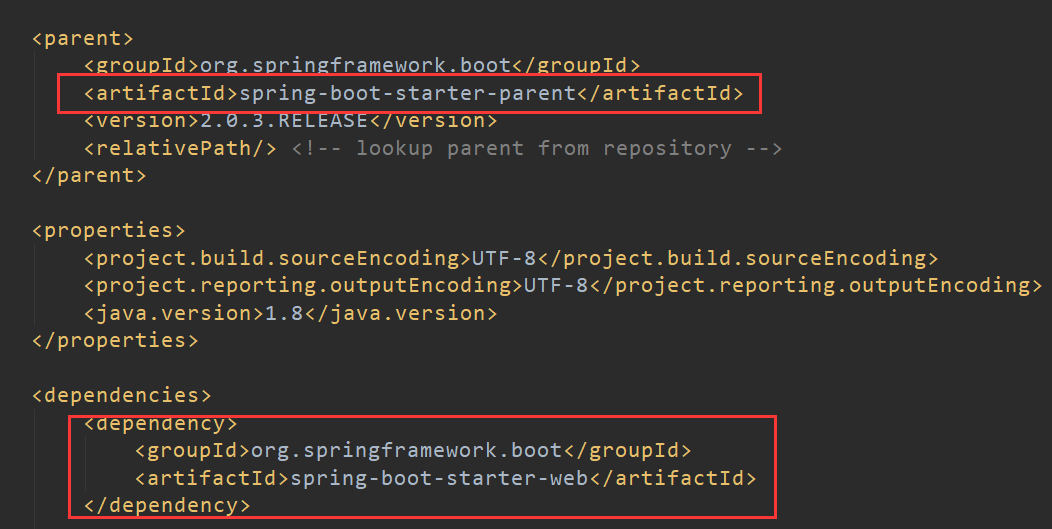
非幂等实现幂等的操作：在扣钱操作中，服务器端扣款成功，但是在将扣款成功返回给客户端时，丢包，客户端认为操作未成功，重复提交，这样就会导致多次扣款，非幂等；在扣款操作前，客户端首先发送一个请求获取一个id，获取到id后，执行扣款操作(会将此id同时发送)服务器端扣款成功，但是返回结果时丢包，客户端可以操作失败，重新发起扣款请求(带上id)，服务器端接收到请求后，发出此id以扣款，且成功，服务器端返回操作成功给客户端



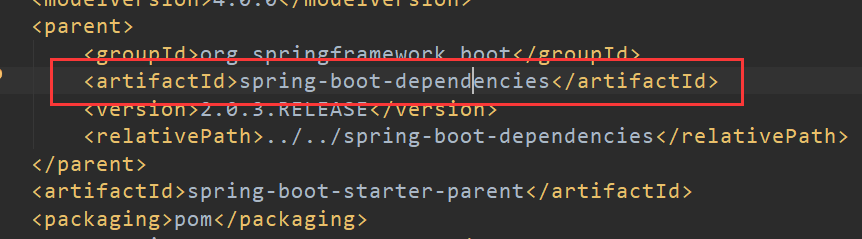
@Controller，@Service，@Respository注解都是@Component注解的子注解

@RequestParam实现自动类型转换

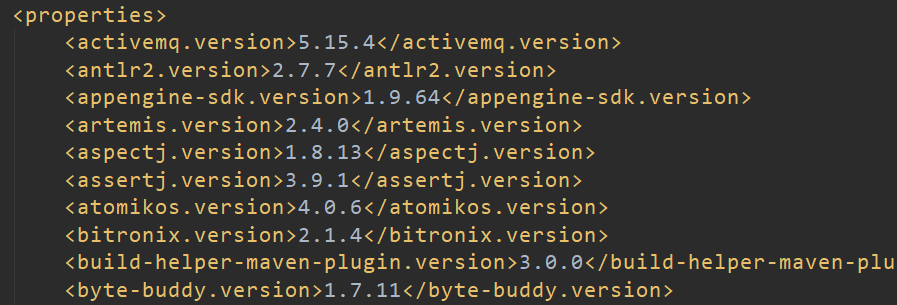
在SpringBoot中在引入依赖的时候，为什么不需要定义依赖的版本(大部分)



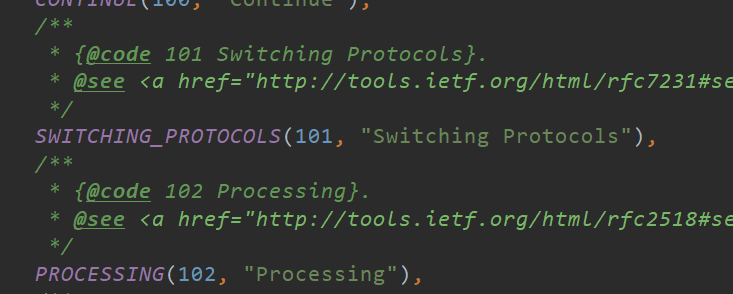
进入spring-boot-parent中



进入spring-boot-dependencies下



当出错时，会显示响应码，可以在HttpStatus中查找对应的响应码及其错误原因



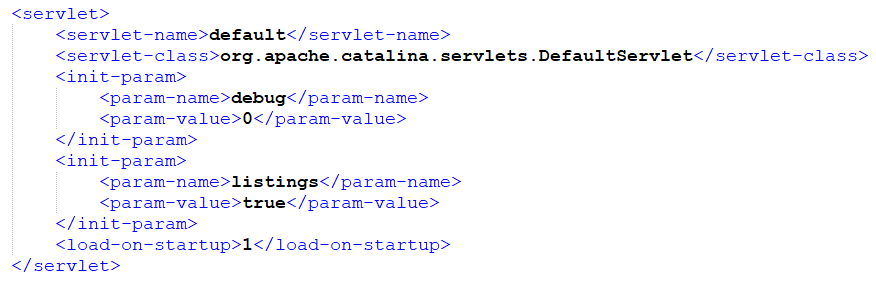
@SpringBootApplication注解实现了@Configuration，@ComponentScan，@EnableAutoConfiguration

@Configuration注解主要和@Bean结合使用

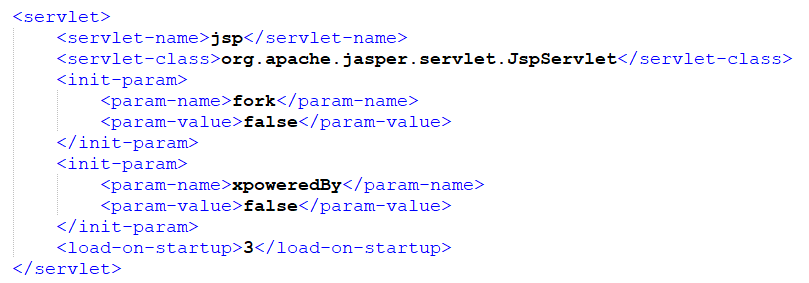
@ComponentScan注解用于指定扫描的路径，SpringBoot默认会扫描main下类及其子包下的类

@EnableAutoConfiguration注解会在你开启某些功能的时候自动配置，这个注解会告诉SpringBoot根据添加的jar依赖猜测你想如何配置Spring，比如导入spring-boot-starter-web时，这可能会是一个web项目，SpringBoot就会自动配置web相关

Tomcat：在tomcat的web.xml文件中声明了两个Servlet，一个DefaultServlet，一个JspServlet，



DefaultServlet的作用是处理静态资源，英文描述时，当其他请求为映射到Servlet时，执行此Servlet进行静态资源的查询，这里的listings表示是否罗列静态资源文件，默认false



JspServlet的作用就是映射.jsp请求，找到对应的jsp页面，并渲染为Servlet

Tomcat在运行的时候先查询自身的web.xml，再查询自定义的web.xml

### SpringBoot中常用注解及其分析：

@SpringBootApplication：main方法所在类上，主要由以下4个注解组成

@EnableAutoConfiguration：自动装配，会根据导入的包将对象加载到IOC中

@SpringBootConfiguration：和Spring的@Configuration一样，@SpringBootConfiguration就继承了@Configuration注解，@Configuration和@Bean注解一起使用

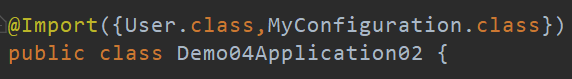
@ComponentScan：去扫描当前类及其类所在的子包的所有类

@Inherited：这是Java的注解，注解A添加@Inherited，在类B添加注解A，类C继承类B，类C上也有注解A，类在继承时是不会继承类的注解的，但是使用@Inherited的注解的类会继承注解

@Configuration/@Bean：

@Configuration用于修饰类，@Bean用于修饰方法，如果通过name获取@Bean的对象，name是方法名，而不是类名

@Import：

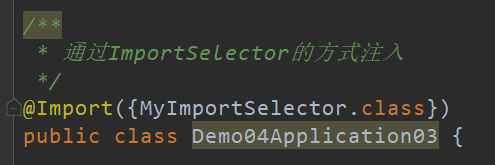


@Import注解的作用是将指定类加入到IOC中

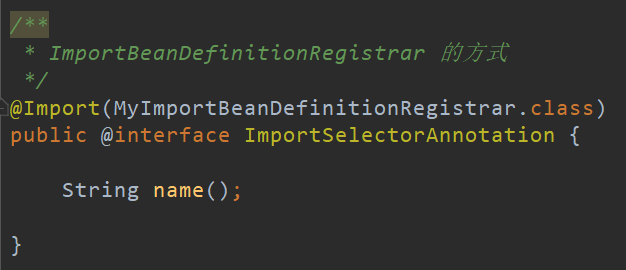
其他实现方式：

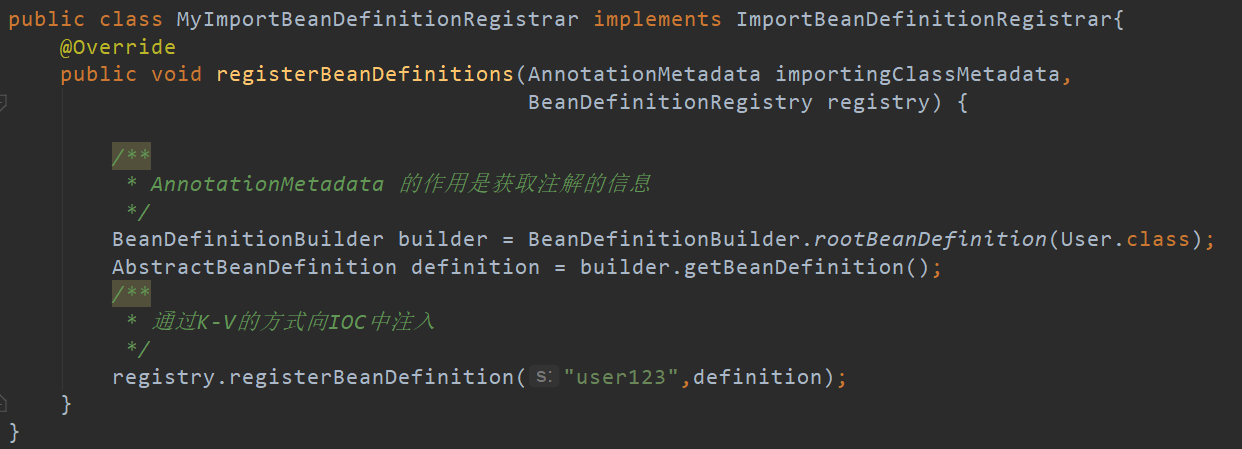


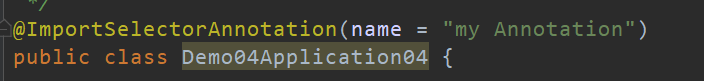
继承ImportSelector，将需要加入到IOC容器中的类的路径作为数组返回，这些类就会被加入到IOC中



自定义注解的方式：

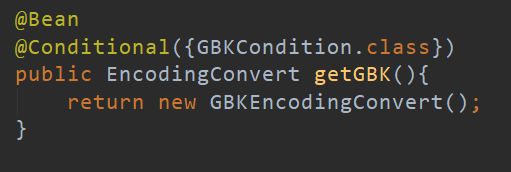


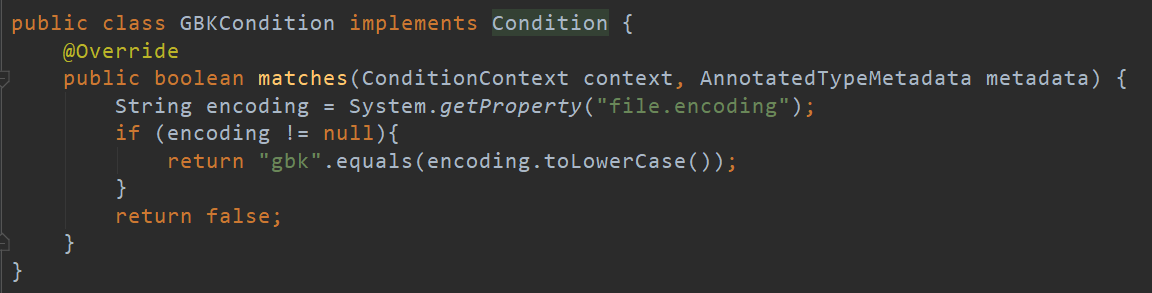




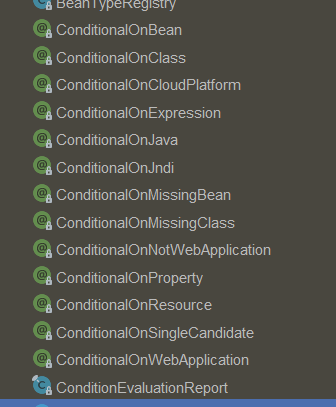
一个是ImportSelector，一个是ImportBeanDefinitionRegister

@Conditional(有条件的)一般和@Bean一起使用，用于判断是否要将当前Bean装载到IOC中



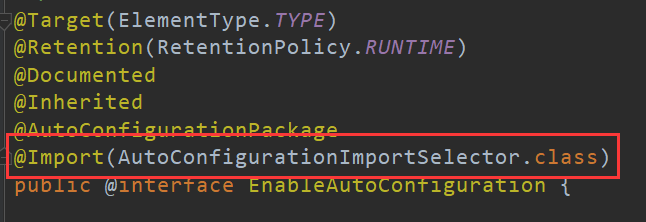


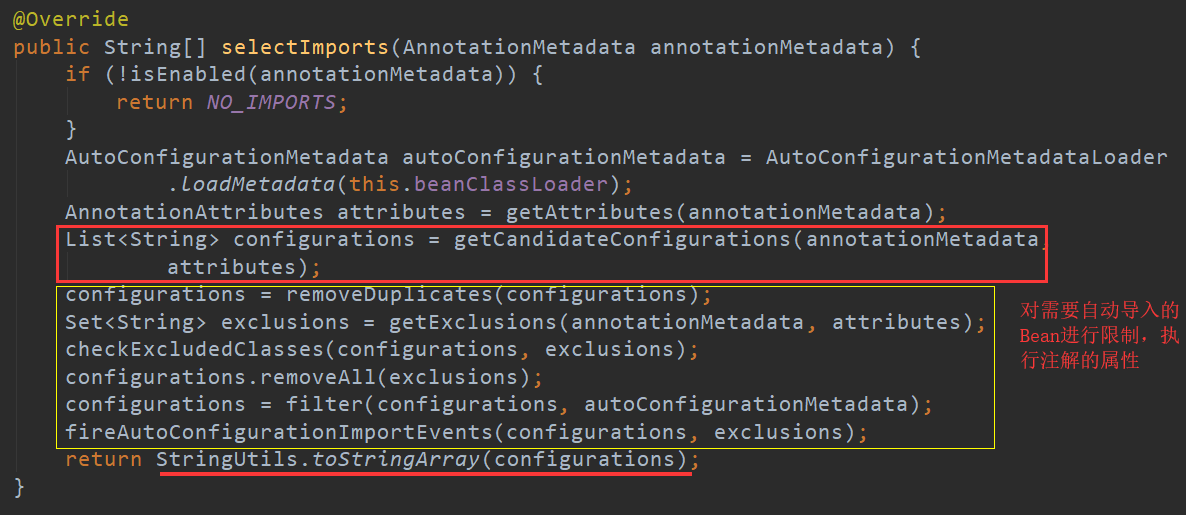
@Conditional传入Class，这个Class必须继承Condition接口，如果接口的matche方法返回true时，才会执行@Bean注解，将Bean装载到IOC中



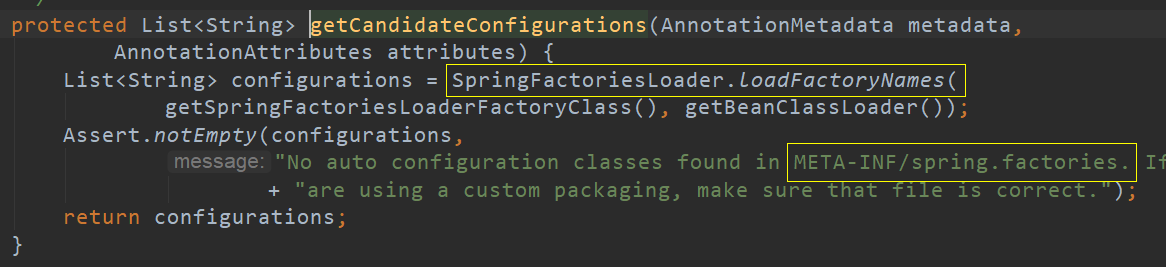
在autoconfigure包下的condition下有很多定义好的@Conditional注解，可以直接使用

@EnableAutoConfiguration自动装配原理：



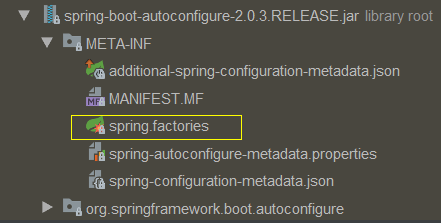


AutoConfigurationImportSelector是ImportSelector的实现类，AutoConfigurationImportSelector中重写selectImports方法

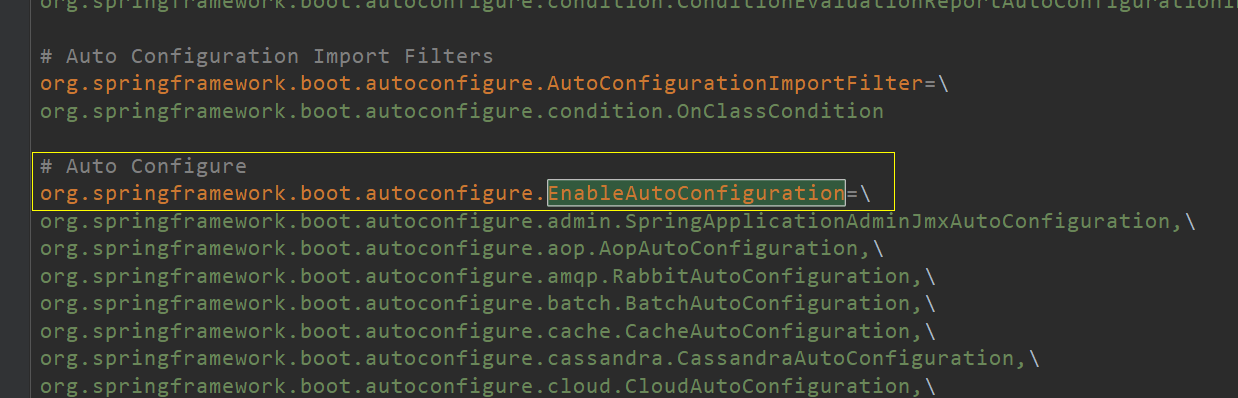


SpringFactoriesLoader.loadFactoryNames会加载ClassPath下的META-INF的spring.factories文件

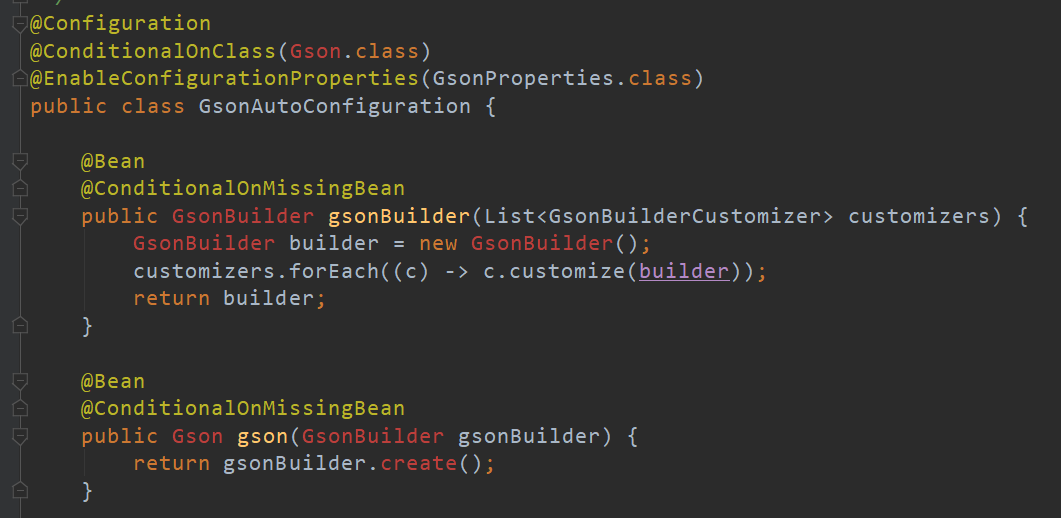
比如，Spring-Boot-autoconfigure.jar下的spring.factories



spring.factories文件，这里的\是换行符



其它的先不管，那么Spring-boot如何实现判断到底要装入哪些Bean



这以GsonAutoConfiguration为例

@Configuration和@Bean就是实例化Bean

@ConditionalOnClass才是关键，判断当前classpath下是否存在Gson类，才会执行实例化此Bean，也就是说，如果没有导入Gson相关的包，那么Spring-boot将不会实例化Gson，这也是为什么在导入包后，Spring-boot会自动装配的原因，Spring-Boot会去猜测是否导入了响应的包的关键就是@ConditionalOnClass注解

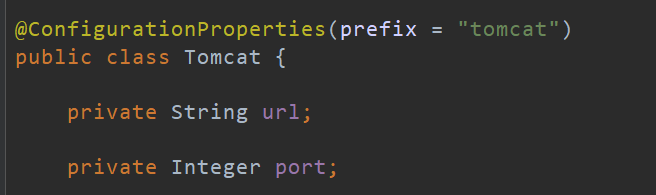
@ConditionalOnMissingBean注解表示，如果IOC中已经有此Bean了，那么不再执行@Bean，确保系统中只有一个Bean

我们也可以在ClassPath下自己创建一个spring.factories文件，以org.springframework.boot.autoconfigure.EnableAutoConfiguration为K，将需要自动装配的Bean的类路径(包含第三方Jar包)写到这里(也可以根据加载实际情况，使用@ConditionalXxx注解)，这样在项目启动的时候，就会自动装配到IOC中

自动装配的判断条件就是根据Conditional注解判断，实现自动装配

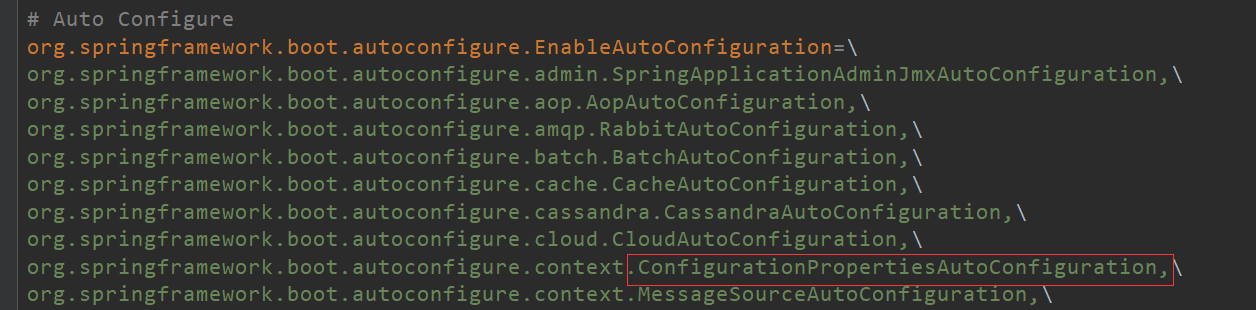
这里的K是org.springframework.boot.autoconfigure.EnableAutoConfiguration，表示@EnableAutoConfiguration注解自动装配的，还有可能其他注解也有类似功能

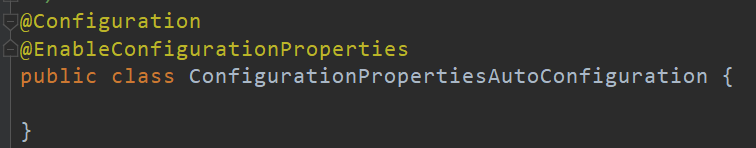
@ConfigurationProperties

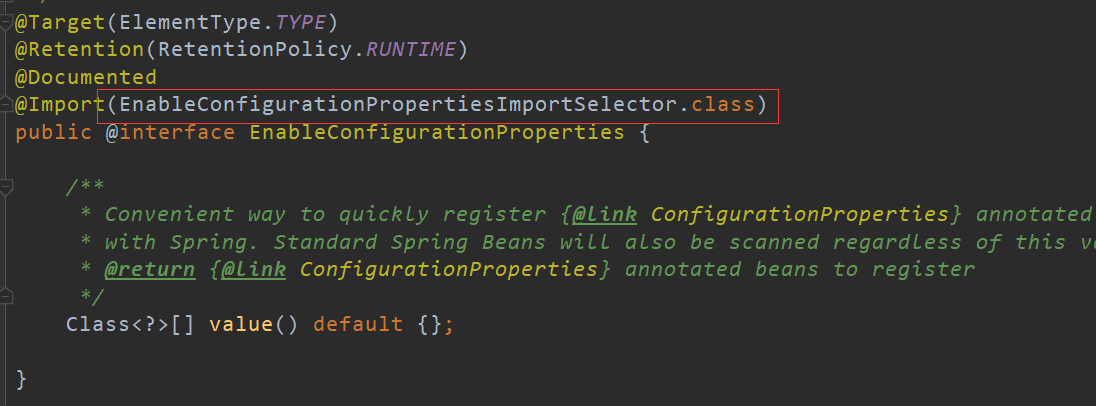


读取application.properties配置文件中的属性，并将Bean对象存放到IOC中

@EnableConfigurationProperties的作用是启动@ConfigurationProperties







@EnableAutoConfiguration在判断导入IOC中的类时，会创建ConfigurationPropertiesAutoConfiguration，此类上有一个@EnableConfigurationProperties，@Import会进实现@ConfigurationProperties类的加入到IOC中，所以就不再需要@Component，同时会实现@ConfigurationProperties的功能，将 application.properties 配置文件的属性装配到Bean中



异常拦截：



在每个类中对异常进行拦截



@ControllerAdvice定义全局异常，@ControllerAdvice是定义全局的地方

SpringBoot的pom文件

在SpringBoot中最主要的几个pom文件：

spring-boot-dependencies：定义一些jar包的版本

spring-boot-starter-parent：定义一些maven，git，idea，eclipse的插件，默认使用JDK1.8，UTF-8编码

spring-boot-dependencies(父)

spring-boot-starter-parent

spring-boot-starter：定义一些SpringBoot的基础(必须的)

spring-boot-starters(父)

spring-boot-parent(父)

spring-boot-dependencies(父)

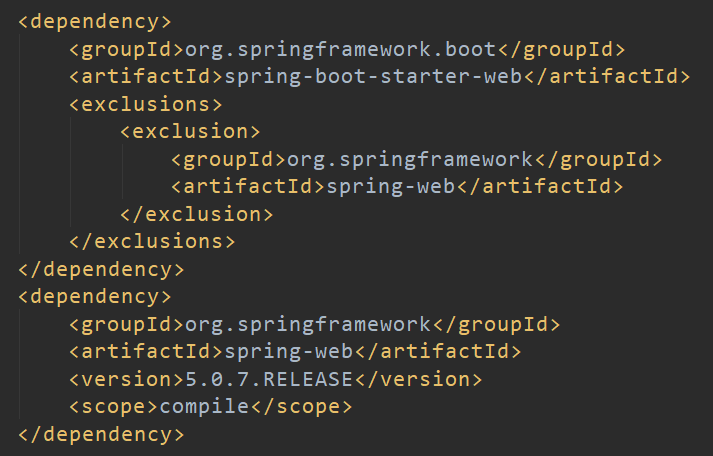
spring-boot-starter-web：web项目需要导入的

spring-boot-starters(父)

spring-boot-parent(父)

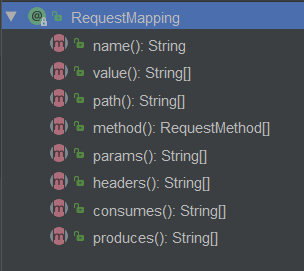
spring-boot-dependencies(父)

spring-boot-starter-web是spring-boot-starters的子pom，但是spring-boot-starter-web引入了spring-boot-starter的依赖，因为这是SpringBoot启动必须的，其他spring-boot-starter-Xxx都是类似的，spring-boot-starter和spring-boot-starters是必须的



Pom文件有多个依赖，可以根据项目的实际需求导入依赖，比如在web中不是使用spring-boot-starter-web自定义的spring-web版本，而使用自定义的5.0.7版本

@ResponseBody注解的作用是将方法的返回值存放到响应的body中



@RequestMapping注解属性详情

Name：实际使用少

Value和path都一样请求的url

Method：请求方式 post，get，put…..

Params：缩小请求范围，params={age!=10&&name==jack}，请求参数中name属性等于jack，并且age属性不等于10才执行方法

Headers：请求头必须包含的

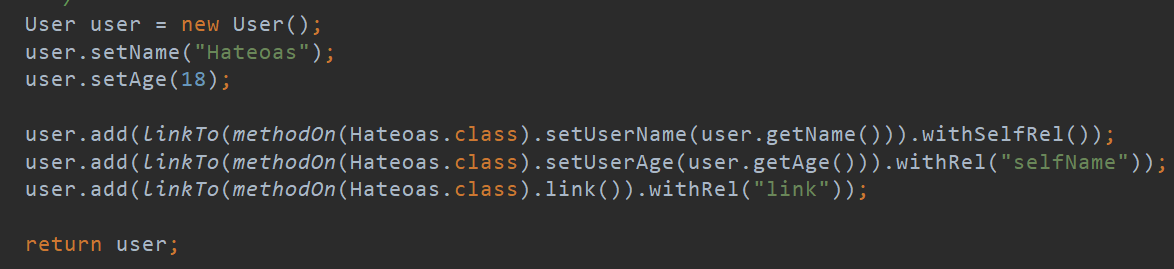
produces：生产者 告诉用于用什么方式解析返回的结果

consumes：消费者 当浏览器能够识别的类型(Accept)和consumes相同时，才访问此方法

hateoas：可以往返回的结果添加Url连接



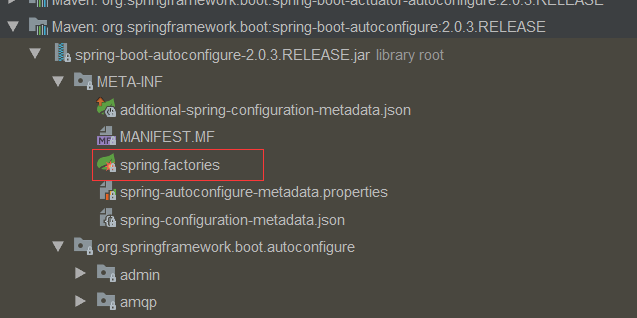
先继承ResourceSupport类，add为ResourceSupport的方法



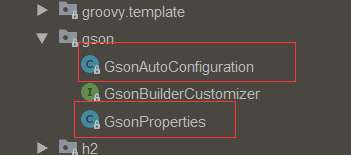
返回结果



autoconfigure包：

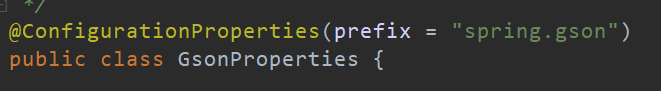


自动装配的依据

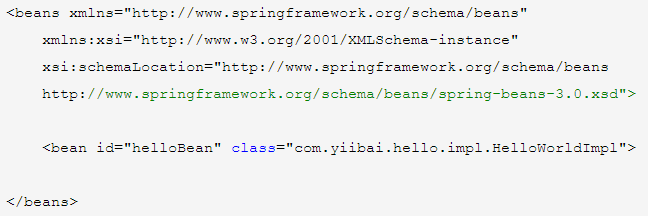


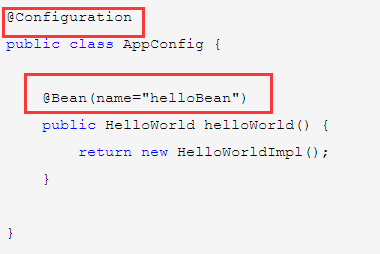
比如装配GSON相关，GsonAutoConfiguration是否装配的条件

GsonProperties配置Gson相关的参数



SpringBoot使用javaConfig的方式替代xml(也可以直接使用注解)





@Configuration对应beans，@Bean对应bean；@Bean如果未声明name，默认是方法名

spring容器最核心的三个jar包是bean，context，core。bean是spring基石，一切结尾bean，context维护了应用的上下文，如果bean是演员，那么context就是舞台，而core则是道具。

IOC：向ConcurrentHashMap中添加对象和取出对象

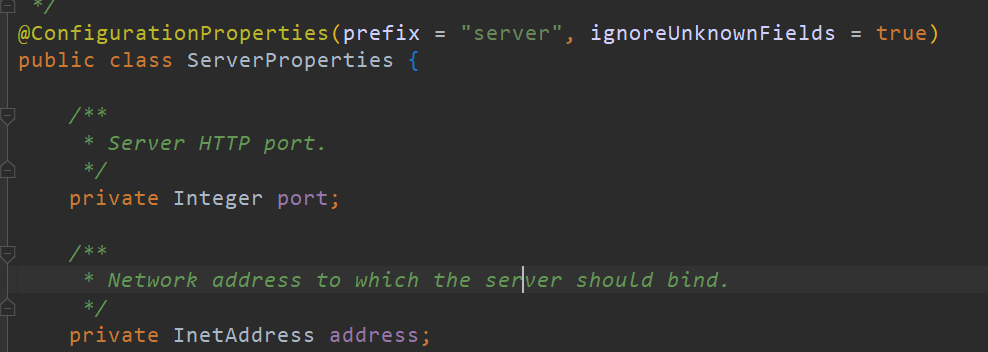
@Autowired、@Inject、@Resource三者区别：都用于Spring的自动装配

@AutoWire：Spring提供的，可作用于变量，set方法，构造方法

@Inject：JSR330提供，可作用于变量，set方法，构造方法，按名注入@Named

@Resource：JSR250提供，作用于变量，set方法(需导包)

@ConfigurationProperties：



为实体类注入属性，在application.properties中配置server.port=6064，实例化的时候会自动set

@EnableScheduling：开启定时任务，作用在Main方法所在类

@Scheduled：定时任务

@Enable\*表示开启某些特定功能

Spring boot，约定优于配置，简化了spring的配置流程。

Dispatcher Servlet(C的部分+Controller) ModelAndView(M) View Resolver(V)

Spring是通过Listener来创建IOC的(ContextLoaderListener，ServletContextListener的实现类)

SpringMVC是在创建DispatcherServlet的时候，在FrameeorkServlet的initServletBean中创建的WebApplicationContext

什么是IOC，什么又是DI，他们有什么区别：

依赖注入DI是一个程序设计模式和架构模型， 一些时候也称作控制反转，尽管在技术上来讲，依赖注入是一个IOC的特殊实现，依赖注入是指一个对象应用另外一个对象来提供一个特殊的能力；控制反转和依赖注入的基本思想就是把类的依赖从类内部转化到外 部以减少依赖

应用控制反转，对象在被创建的时候，由一个调控系统内所有对象的外界实体，将其所依赖的对象的引用，传递给它。也可以说，依赖被注入到对象中。所以，控制反转是，关于一个对象如何获取他所依赖的对象的引用，这个责任的反转。

Spring中设计模式：

工厂模式：BeanFactory

单例模式：Singleto n的Bean

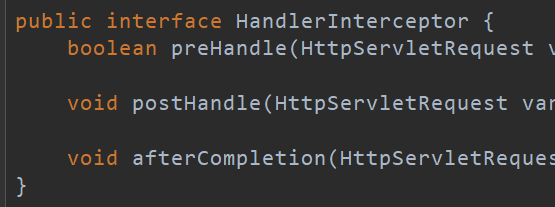
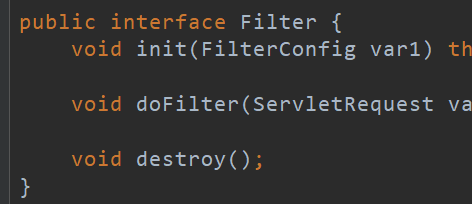
适配起模式：SpringMVC中的HandlerAdapter

代理模式：AOP

观察者模式：ApplicationListener

拦截器和过滤器：

过滤器Filter，拦截器HandlerInterceptor



Filter基于Servlet，对所有请求进行拦截，但是Filter只会在容器初始化的时候调用，然后每次只会执行doFilter方法

HandlerInterceptor基于SpringMVC，基于AOP的思想，preHandle，postHandle，afterCompiletion都会执行，作用方法

注解：

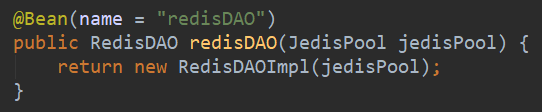
ConfigurationProperties：用于类上，类中的属性可以读取properties文件中的属性

和@Value的区别：



@Configuration和@Bean的作用：

@Configuration类似Spring的xml文件中的beans标签，而@Bean则是bean标签



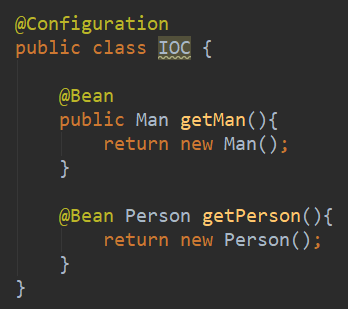
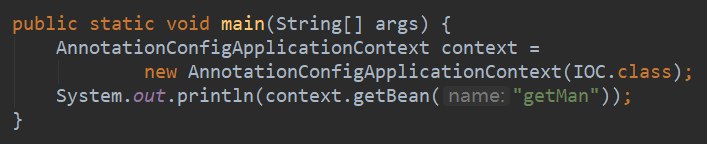
@Configuration作用于类，@Bean作用于方法，方法必须返回的是一个对象，返回的对象存放到IOC中

这样的好处，用注解代替了XML，@Componet用于类上实例化对象，而@Bean用于方法上，将方法的返回对象存放到IOC中

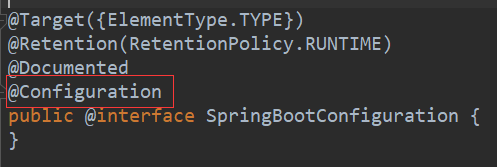
相当于是@Configuration注解类，此类用于替代XML文件一样，在初始化加载的时候，首先加载的还是是此@Configuration类，XML文件有啥功能@Configuration和@Bean基本就有对应的功能

XML文件有ClassPathXMLApplication加载XML文件，那么@Configuration用什么加载呢？AnnotationConfigApplicationContext；这里只是单纯的一个Spring项目，如果是一个web项目，还是得去扫描知道的包

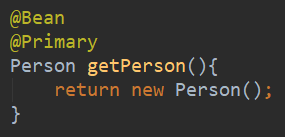
但是在通过getBean获取Bean时，通过Byname方式获取时，name是方法的名称

@Configuration和@SpringBootConfiguration的区别：@Configuration是Spring的注解，而SpringBootConfiguration注解是SpringBoot的，实际上@SpringBootConfiguration就是@Configuration



@Primary：如果在IOC中有多个同类型的Bean，在根据class获取的时候就会抛出异常，当有多个时，就会去找有@Primary的Bean，@Primary常和@Bean使用



@AutoWire @Resource @Inject

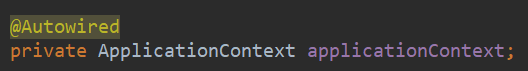
@AutoWire是Spring自带的，默认是ByType，可以使用@Qualifier通过ByName获取

@Inject是JSR330规范，需要导入包，默认ByType，可以使用@Named通过ByName获取，和AutoWire类似，差别是AutoWire有一个required属性

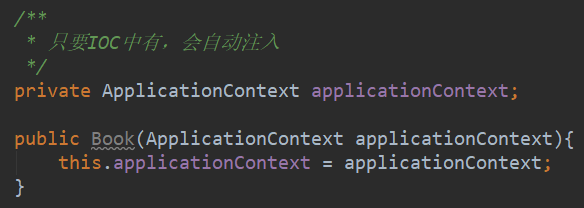
@Resource是JSR250规范，不需要导包，默认是ByName

在Bean中获取ApplicationContext的方式：

实现BeanFactory/ApplicationContextAware接口，并重写其方法

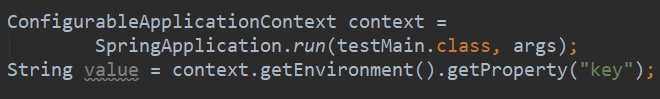


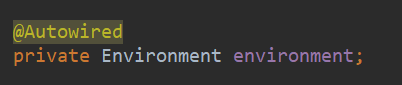
直接在Bean中使用AutoWire注解



构造函数中

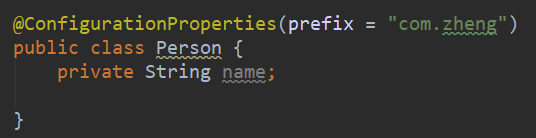
获取application.properties文件中属性的方式：







使用Value注解获取application.properties文件中属性，: jack 表示当未找到对应属性时，默认值

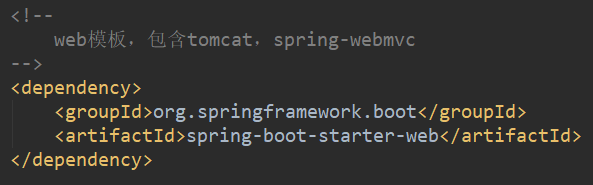


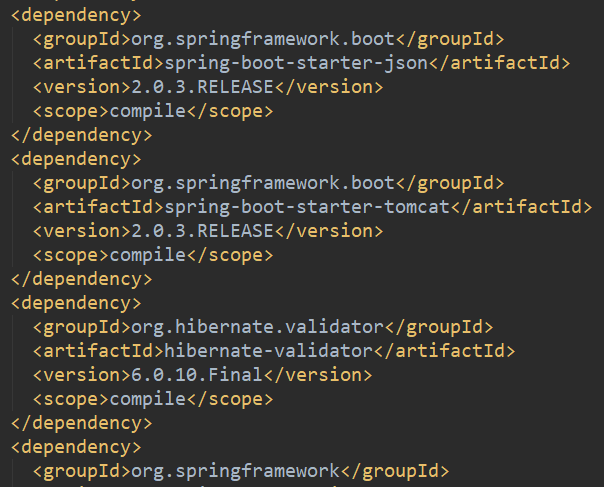
每个都写Value和前缀太麻烦，会自动获取com.zheng.name对应的属性值

@PropertySource注解：SpringBoot会默认加载application.properties文件，当有其他properties文件时，可以使用@PropertySource注解将properties文件加载到Environment中

@PropertySource("classpath:configuration.properties")

SpringBoot实际是在Spring的基础上进行封装，在SpringBoot的pom文件中，点进start-web中可以看到很多其他的依赖关系





Spring中初始化方法和销毁方法的三种方式

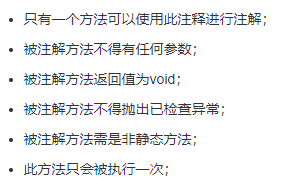
在<bean>或者@Bean中声明init和destroy属性

实现InitializingBean和DisposableBean接口

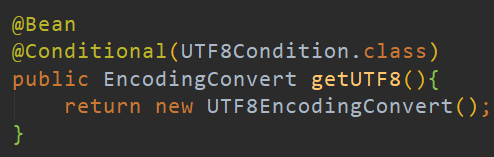
@PostConstruct/@ PreDestroy在类的方法中实现这个两个注解，但是这两个注解是基于java的

顺序：Constructor > @PostConstruct > InitializingBean > init-method

@PostConstruct的特点

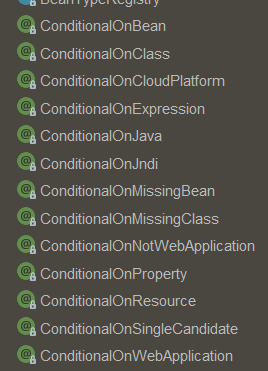


在springboot中创建Servlet，Filter，Listener等可以不用再配置配置文件，而直接在类上使用注解(@WebServlet….)，然后在注入到对应的RegistrationBean中



@Conditional注解和Condition接口结合使用，UTF8EncodingConvert实现Condition接口，并重写matches方法，如果matches返回为ture，才会将返回结果注入到IOC中

SpringBoot中提供了很多类似的@ConditionalXxx注解，来判断是否对bean进行装配



比如@ConditionalOnBean表示，如果在ioc中有对应的bean，才将返回的bean注入IOC

在SpringBoot中@EnableXxx的注解，此注解的作用是开启某些功能

@EnableConfigurationProperties的作用开启@ConfigurationProperties的功能