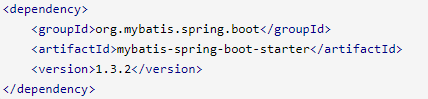
1. mybatis-spring-boot-starter

1）



2）



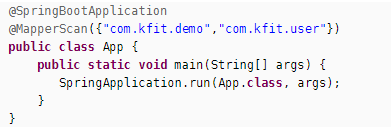
3）

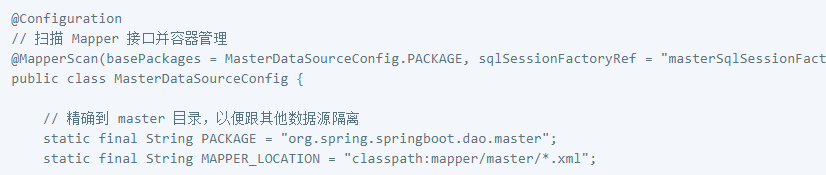


4）Scanning for mappers

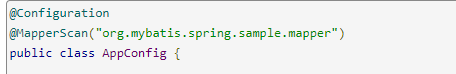
① <mybatis:scan base-package=”mapper路径”>，路径之间可以用分号或逗号隔开。如果是单个数据源，可以自动映射，多个的话要在配置文件中分别配置，见[多数据源配置](#多数据源配置)

② @MapperScan，为了避免使用太多@Mapper注解





也可以按照以上来指定数据特定的数据源



相当于在appCofig配置文件中，加了一个mapper扫描功能

③MapperScannerConfigurer

也可以在以上bean中加上：以指定特定的数据源



④ 多数据源的管理见 Spring-Mybatis多数据源的配置与管理.docx

1. Spring Data JPA 可以理解为 JPA 规范的再次封装抽象，底层还是使用了 Hibernate 的 JPA 技术实现。要说差异，主要是比较的是 MyBatis 和 Hibernate。
2. @Bean @Configuration注解

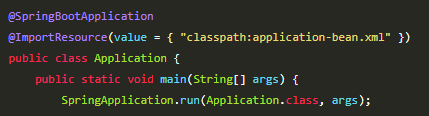
1）@Configuration：标注在类上，相当于把该类作为spring的xml配置文件中的<beans>

2) @Bean标注在方法上（返回值为一个类），相当于xml配置中的<bean>

3) 详情见 @bean与@configuration注解.docx



1. springboot配置文件
2. pring Boot使用一个全局的配置文件application.properties，Spring Boot 支持使用YAML语言的配置文件，YAML是以数据位中心的语言，所以使用application.yml作为全局配置也是同样的效果，如果使用YAML替代properties注意写法，冒号后面要加个空格，否则会解析不出来。
3. 引入XML配置文件
4. 通过在入口启动类上加上@ImportResource(value = { "路径" })或者使用@ImportResource(locations= { "路径",””,… })，一样的效果，多个XML文件的话你可以用逗号“,”分隔，就这样轻而易举的引用XML配置。



1. 在一个configuration类上添加也可以



1. 引入多个@Configuration配置类

在实际项目中可能不会把所有的配置都放在一个配置类（用@Configuration注解的类）中，可能会分开配置。这时可以用@Import注解引用。

这样来说，各个configuration类上下文直接不互通？



1. 读取配置文件中的属性
2. 通过@value注解

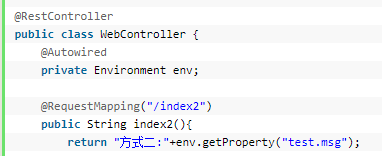


1. 通过@configurationProperties将属性注入到Bean中

现在下面那个方法已经过时了，spirng2.x没有了，用[**代替方法**](#代替的配置文件属性注入方法)



③ 通过Environment类 ，测试就没有getProperty这个方法。。



③ @configurationProperties注解需要用@EnableConfigurationProperties来使之生效

④ @SpringBootApplication注解：



1. Netty

.netty是一套在java NIO的基础上封装的便于用户开发网络应用程序的api. 应用场景很多,诸如阿里的消息队列(RocketMQ),分布式rpc(Dubbo)通信层都使用到了netty(dubbo可以用服务发现自由选择通信层).

1. Springboot四大核心
2. 自动配置
3. 起步依赖
4. Actuator
5. 命令行界面（CLI）：虽然功能强大，但开发模型非常规

7、因为容器中可能有多个BeanFactoryPostProcessor，可能还需要实现org.springframework.core.Ordered接口，以保证BeanFactoryPostProcessor按照顺序执行。

8、占位符什么时候被替换

运行后置处理器时被替换

PropertyPlaceholderConfigurer作为BeanFactoryPostProcessor被应用时，它会使用properties配置文件中的值来替换相应的BeanDefinition中占位符所表示的属性值。当需要实例化bean时，bean定义中的属性值就已经被替换成我们配置的值。

9、beforePostProcessor->initializingBean->init-methd->BeanPostProcessor

10、[spring Bean初始化步骤](#springbean初始化步骤)

11、@ComponentScan扫描范围

我们可以通过 basePackages等属性来指定 @ComponentScan自动扫描的范围，如果不指定，默认从声明 @ComponentScan所在类的 package进行扫描。正因为如此，SpringBoot的启动类都默认在 src/main/java下。

12、[@Conditional注解](#conditional注解)

13、@ConfigurationProperties与@EnableConfigurationProperties与属性注入

1）通过@value注入

使用 @Value注解注入的属性通常都比较简单，如果同一个配置在多个地方使用，也存在不方便维护的问题（考虑下，如果有几十个地方在使用某个配置，而现在你想改下名字，你改怎么做？）。对于更为复杂的配置，Spring Boot提供了更优雅的实现方式，那就是 @ConfigurationProperties注解。

注意使用@Value的类如果被其他类作为对象引用，必须要使用注入的方式，而不能new。

2) @ConfigurationProperties注解



14、读取配置文件属性

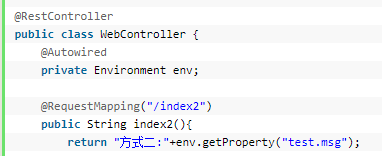
1）读取核心配置文件，只能读取核心文件

Classpath文件下的properties和yml文件，子文件不算

① 通过@value注解



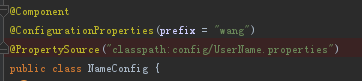
② 通过注入Environment，测试无此方法。。

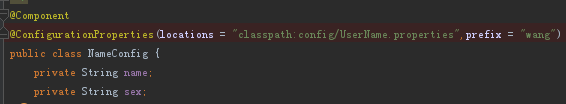


2）读取自定义配置文件

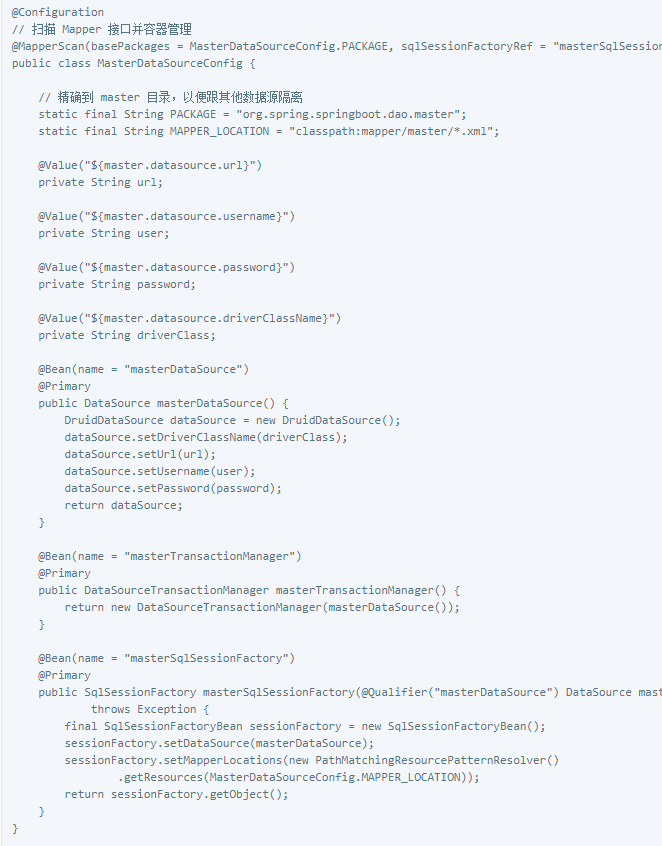
通过@ConfigurationProperties注解

1）spring2.x+



2）spirng1.x 

一、多数据源配置



二、@conditional注解





三、spring bean初始化步骤

