#### 1、yml文件

这里设置参数的时候往往设置两层前缀，避免在调用的时候该属性中的部分参数名与计算机中的某些名字冲突。

spring:

redis:  
 database: 1  
 host: localhost  
 port: 6379  
 password:  
 timeout: 0

#### 2、在JavaBean中读取参数值，并且将读取出来的值赋予到指定的属性上

@Configuration

@ConfigurationProperties(prefix = "spring.redis")  
public class RedisProperties {  
  
 @Value("${host: #{null}}")  
 private String host;  
 @Value("${port: #{null}}")  
 private Integer port;  
 @Value("${password: #{null}}")  
 private String password;  
  
 @Value("${timeout: #{2000}}")  
 private Integer timeout = 2000;

// 省略getter、setter

}

@ConfigurationProperties(prefix=“spring.redis”)将配置文件中配置的每个属性和配置文件中相关的配置进行绑定；

prefix=“spring.redis”配置文件中哪个下面的所有属性进行一一映射（yml文件中的参数名称要和JavaBean中的属性名称保持一致）

#### 3、pom.xml文件中导入依赖配置

<!--导入配置文件处理器，配置文件进行绑定就会有提示-->

<dependency>

<groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-configuration-processor</artifactId>  
 <optional>true</optional>  
</dependency>

#### 4、@ConfigurationProperties(prefix = "person") 和 @Value的区别

|  |  |  |
| --- | --- | --- |
|  | @ConfigurationProperties | @Value |
| 功能 | 批量注入配置文件中的属性 | 一个个指定 |
| 松散绑定（松散语法） | ✅ | ❌ |
| SpEL | ❌ | ✅ |
| JSR303 | ✅ | ❌ |
| 复杂类型封装 | ✅ | ❌ |

如果说，我们只是在某个业务逻辑中需要获取一下配置文件中的某项值，使用@Value；

如果说，我们专门编写了一个javaBean来和配置文件进行映射，我们就直接使用@ConfigurationProperties；