这里主要介绍基于spring-boot使用

我写的实例代码在gitee上：<https://gitee.com/astorage/apollo-client>

引入依赖：

<dependency>  
 <groupId>com.ctrip.framework.apollo</groupId>  
 <artifactId>apollo-client</artifactId>  
 <version>1.7.0</version>  
</dependency>

配置：

apollo:  
 bootstrap:  
 enabled: true  
 meta: http://10.0.92.111:8080  
 autoUpdateInjectedSpringProperties: true  
  
app:  
 id: SampleApp

8080端口是config service服务

这样就可以从apollo拉去配置信息了。下面基于spring-boot用三种方式来获取配置：

1、@Value注解，可以实现自动更新

2、@ConfigurationProperties注解，不能自动更新，可以借助下面两种方法可以实现自动更新。

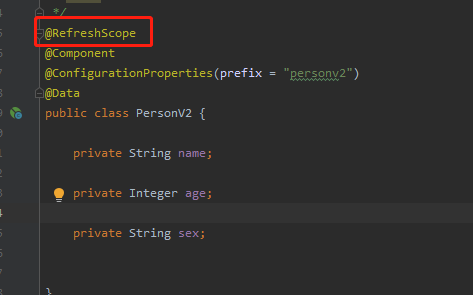
@ApolloConfigChangeListener

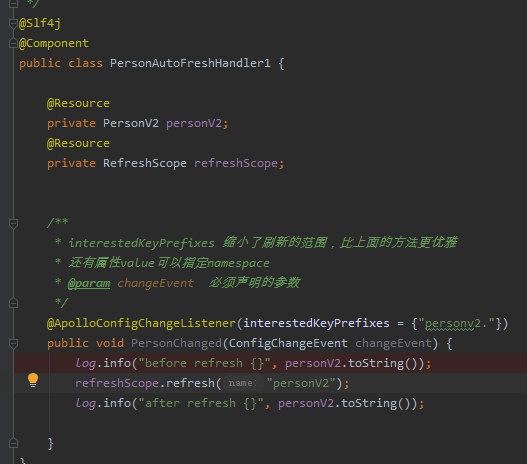
*/\*\*  
 \* interestedKeyPrefixes 缩小了刷新的范围，比上面的方法更优雅  
 \* 还有属性value可以指定namespace  
 \** ***@param*** *changeEvent  
 \*/*@ApolloConfigChangeListener(interestedKeyPrefixes = {"person."})  
public void PersonChanged1(ConfigChangeEvent changeEvent) {  
 if (changeEvent.isChanged("person.name")) {  
 *log*.info("person.name changed");  
 person.setName(config.getProperty("person.name", person.getName()));  
 }  
  
 if (changeEvent.isChanged("person.age")) {  
 *log*.info("person.age changed to {}", config.getIntProperty("person.age", person.getAge()));  
 person.setAge(config.getIntProperty("person.age", person.getAge()));  
 }  
  
 if (changeEvent.isChanged("person.sex")) {  
 *log*.info("person.sex changed");  
 person.setSex(config.getProperty("person.sex", person.getSex()));  
 }  
  
  
}

另一种借助spring-cloud的功能。首先引入依赖：

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-context</artifactId>  
 <version>2.2.4.RELEASE</version>  
</dependency>

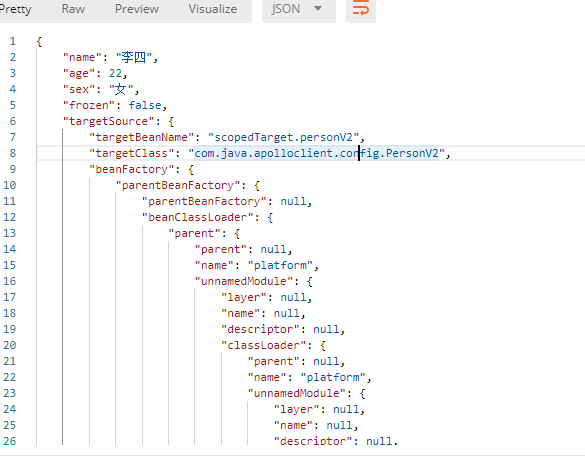
使用@RefreshScope注解和类RefreshScope





@RefreshScope注解能帮助我们做局部的参数刷新，但侵入性较强，需要开发阶段提前预知可能的刷新点，并且该注解底层是依赖于cglib进行代理的

生成的代理对象也很大，主要是代理对象中有非业务字段targetSource字段。大概有5200多行，这是我们不需要的。



参考文档：https://github.com/ctripcorp/apollo/wiki/Java客户端使用指南

<https://github.com/ctripcorp/apollo-use-cases>