# 分布式配置中心DISCONF使用手册

|  |  |  |
| --- | --- | --- |
| 文件状态：  【√】初稿  【 】正式发布  【 】正在修改 | 文件标识： | 理赔平台规划 |
| 当前版本： | 1.0 |
| 作 者： | 理赔服务开发团队-朱欢敏 |
| 完成日期： | 2018/2/11 |

## 什么内容可以放配置中心

1. 系统开关类，修改后，同步到各服务器，并自动重新加载
2. 服务器主机信息类
3. 调用其他服务的URL配置

## 系统接入配置中心

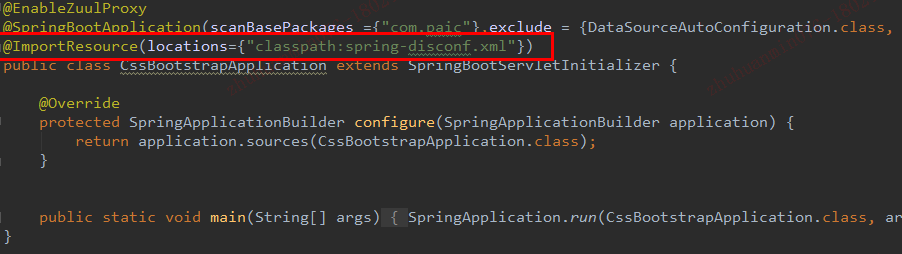
1. 增加依赖包（版本请使用最新版，

<http://maven.paic.com.cn/#browse/browse/components:maven-public> 搜索)

<dependency>  
 <groupId>com.paic.icore</groupId>  
 <artifactId>claim-framework-disconf-client</artifactId>  
 <version>1.0.14</version>  
</dependency>

1. 增加spring-disconf.xml 配置文件

对应SpringBoot 应用，可以参考如下配置



这个配置文件，主要是对于非侵入式配置文件的管理。

例如：（demo 代码的配置为例）



下面的两个bean 是默认要的，主要作用有侵入式注解的扫描，以及配置的校验；注入数据到实体类等功能。

<bean id="disconfMgrBean" class="com.paic.icore.framework.disconf.client.DisconfMgrBean" destroy-method="destroy">  
 <property name="scanPackage" value="com.paic"/>  
</bean>  
<bean id="disconfMgrBean2" class="com.paic.icore.framework.disconf.client.DisconfMgrBeanSecond" init-method="init" destroy-method="destroy"/>

下面的配置是非侵入式，需要自动重新加载的配置

（bean id =”configproperties\_disconf“）：

<bean id="propertyConfigurer" class="com.paic.icore.framework.disconf.client.addons.properties.ReloadingPropertyPlaceholderConfigurer">  
 <property name="ignoreResourceNotFound" value="true"/>  
 <property name="ignoreUnresolvablePlaceholders" value="true"/>  
 <property name="propertiesArray">  
 <list>  
 <ref bean="configproperties\_disconf"/>  
 </list>  
 </property>  
</bean>  
  
<!-- 使用托管方式的disconf配置(无代码侵入, 配置更改会自动reload)-->  
<bean id="configproperties\_disconf" class="com.paic.icore.framework.disconf.client.addons.properties.ReloadablePropertiesFactoryBean">  
 <property name="locations">  
 <list>  
 <value>classpath\*:acl.properties</value>  
 <!--<value>classpath\*:business-on-off.properties</value>-->  
 <value>classpath\*:xml-config.properties</value>  
 <value>classpath\*:product.properties</value>  
 <value>classpath\*:error.properties</value>  
 <value>classpath\*:um-security.properties</value>  
 <value>classpath\*:mybatis-refresh.properties</value>  
 </list>  
 </property>  
</bean>

下面是spring 属性的默认配置项，以及可以在配置中心管理的配置

<bean id="disconfService" class="com.paic.property.icore.css.service.impl.DisconfServiceImpl">  
 <property name="xmlConfigFlag" value="${xml.config.flag=123}"></property>  
 <property name="productdto" ref="productDto"></property>  
</bean>  
  
<bean id="productDto" class="com.paic.property.icore.css.dto.ProductDTO">  
 <property name="price" value="${product.price=0.0}"></property>  
 <property name="maxStore" value="${product.max-store=100}"></property>  
</bean>

1. 增加自己的disconf.properties配置



# 是否使用远程配置文件  
# true(默认)会从远程获取配置 false则直接获取本地配置  
enable.remote.conf=true  
  
#  
# 配置服务器的 HOST,用逗号分隔 127.0.0.1:8000   
#  
conf\_server\_host=30.4.142.226:80  
  
# 版本, 请采用 X\_X\_X\_X 格式  
version=1\_0\_0  
  
# APP 请采用 产品线\_服务名 格式  
app=ICORE-CSS  
  
# 环境  
env=dev  
  
# debug  
debug=true  
  
# 忽略哪些分布式配置，用逗号分隔  
ignore=  
  
# 获取远程配置 重试次数，默认是3次  
conf\_server\_url\_retry\_times=1  
# 获取远程配置 重试时休眠时间，默认是5秒  
conf\_server\_url\_retry\_sleep\_seconds=1

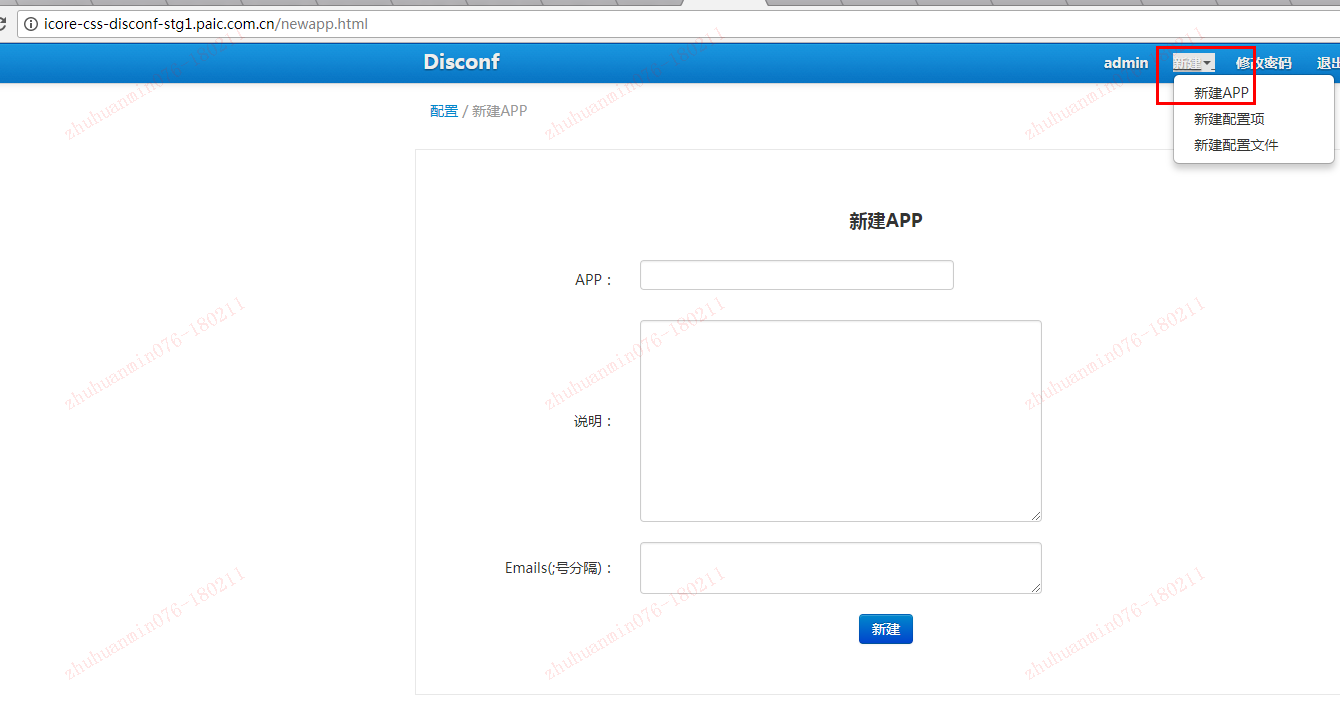
#默认配置中心下载的配置存放位置  
user\_define\_download\_dir=./disconf/config/download

## 创建应用级配置管理

### 登录disconf 控制台

<http://icore-css-disconf-stg1.paic.com.cn> admin/angus

### 新建APP（应用）



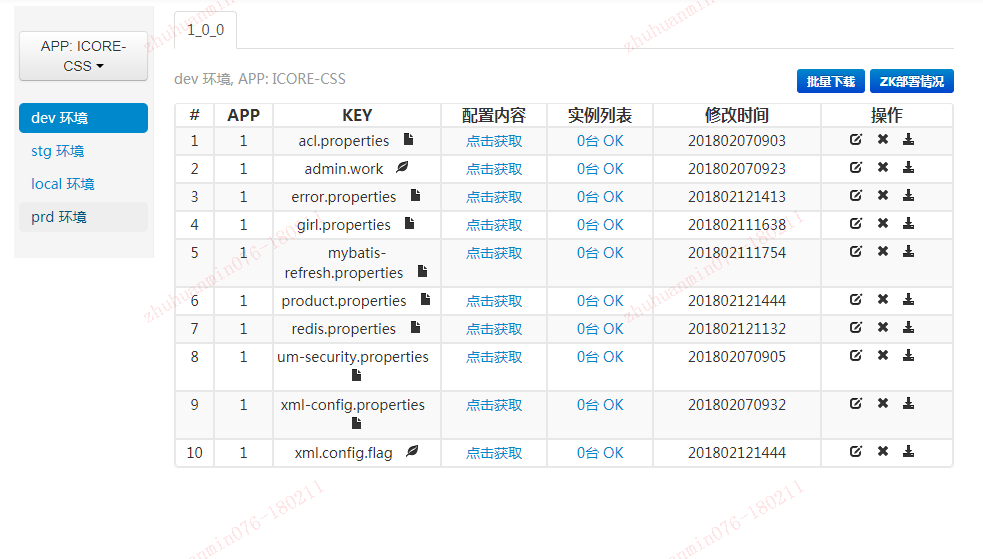
一般你的系统英文名

### 新建配置文件项



选择你的APP ,设置版本号（下划线格式如 1\_0\_0），选择环境，这里可以一般选择dev,stg。生产环境有另外的一个配置中心。

配置完以后可以看到，类似如下情况：



实例列表：是指当前使用该配置项的依赖方实例数。

### 新建配置文件



新建配置文件可以选择是上传文件，还是直接在线编辑配置，推荐上传已有的配置，防止手工写错。

注： 以上设置的APP ,版本号，以及选择的环境，待会在程序的配置文件disconf.properties中会用到，通过这三个维度，来区分不同的配置文件/项。

## 使用场景&demo 说明

Disconf 使用场景有：

1. 配置文件统一管理
2. 配置文件重新加载，实时刷新
3. 配置项重新加载，实时刷新

以下demo 以<http://git-ma.paic.com.cn/ZHUHUANMIN076/icore-claim-demo/tree/master/icore-claim-css> 为例说明：

### 非侵入式

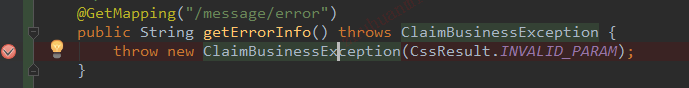
#### 配置文件管理

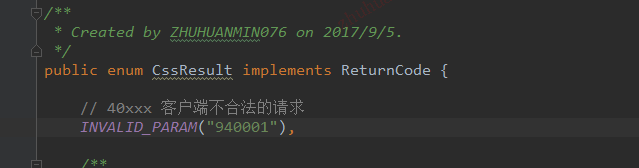
##### Demo1 非侵入式error.properties 自动刷新

* spring-disconf.xml 中增加对error.properties 的自动刷新配置。

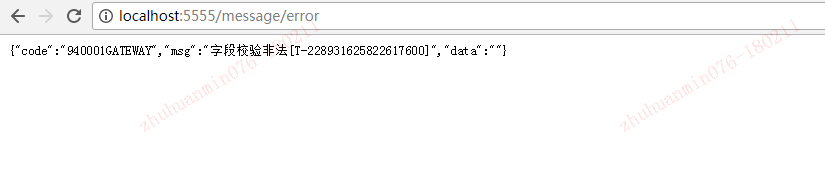


* Java 代码





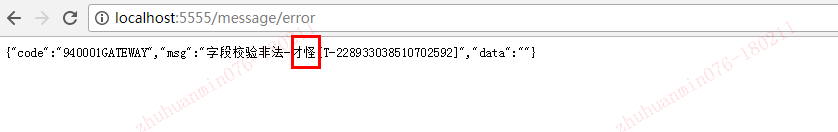
* 访问效果



* 修改配置中心该值：



* 再次访问：

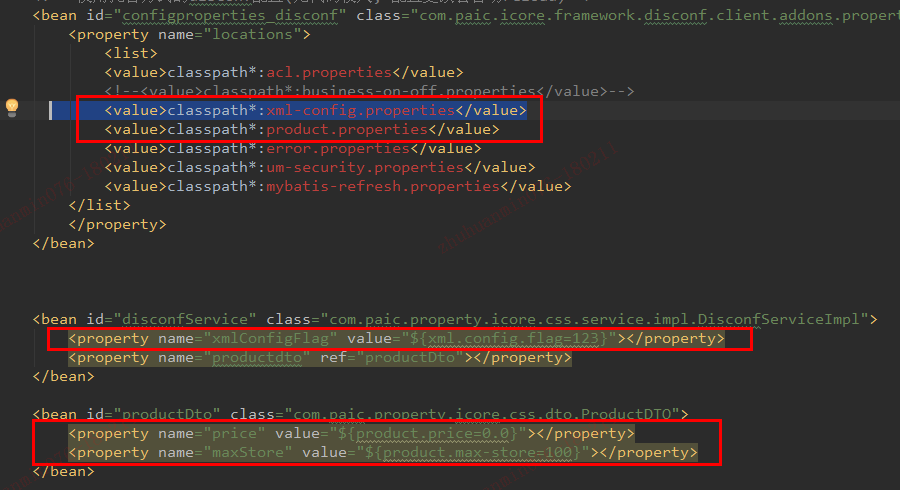


可以看到，代码结构中无error.properties ,只在配置中心中统一管理着，可以无侵入原有代码，自动做到配置可刷新。

##### Bean属性的自动刷新

* spring-disconf.xml配置

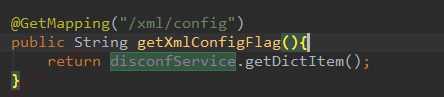
应用中spring-disconf.xml 对bean 的配置，以及配置项所属配置文件，添加到自动刷新的文件列表中，如下图



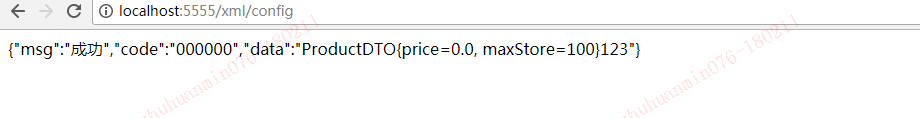
* Java 代码（com.paic.property.icore.css.service.impl.DisconfServiceImpl）



* 验证方法（com.paic.property.icore.css.action.DisconfController）



* 配置中心未修改前：

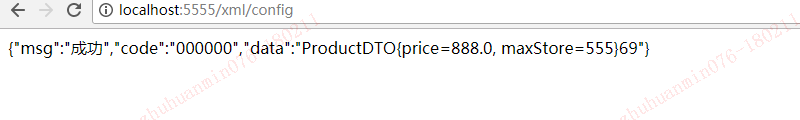


* 修改　product.properties, xml-config.properties文件





* 浏览器刷新



#### 配置项管理

非侵入式，单配置项无法自动刷新。

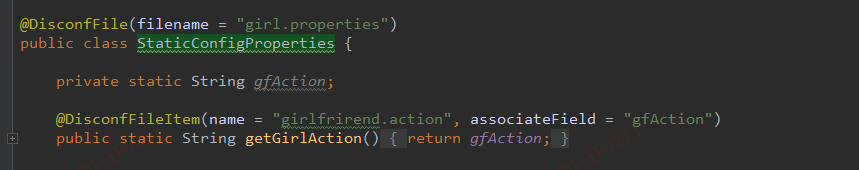
### 侵入式

看demo 之前，介绍几个Disconf 自定义注解：

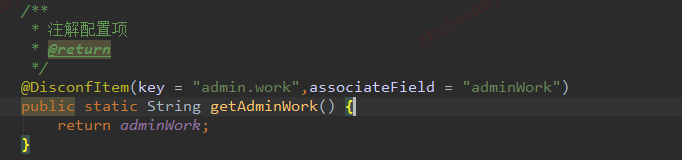
com.paic.property.icore.css.config.StaticConfigProperties

@DisconfFile 表示这个配置文件会被分别式配置中心管理，自动加载

@DisconfFileItem 用在方法上注解，表示配置文件的取值来源与哪个托管字段，并设置到哪个字段去。



@DisconfItem 用在方法上注解，表示方法取值来源于哪个配置项

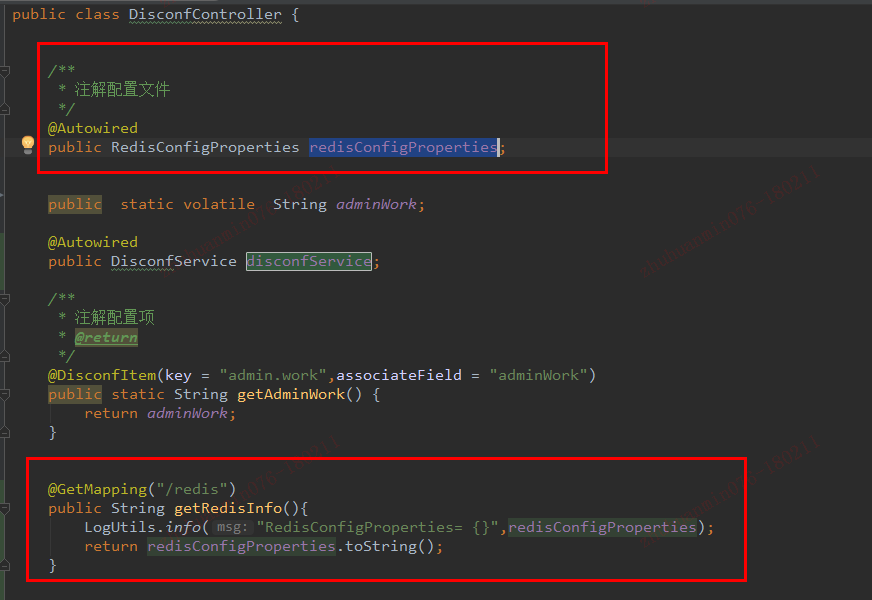


#### 配置文件管理

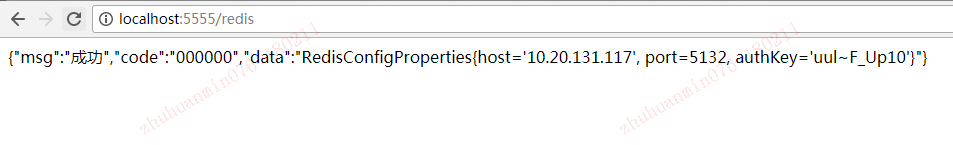
* 注解配置



* Java 代码



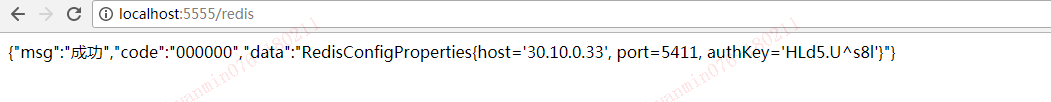
* 首次访问



* 修改配置，提交

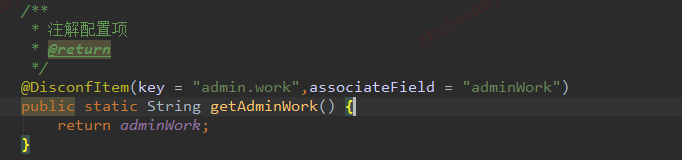


* 再次访问，刷新浏览器

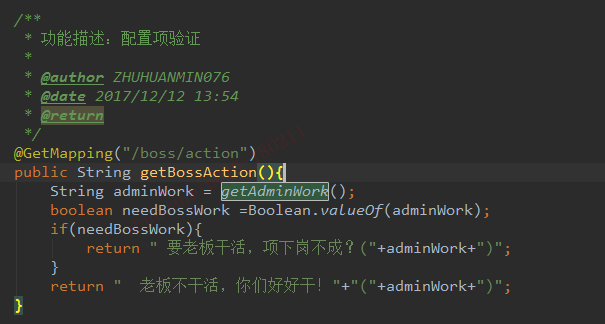


#### 配置项

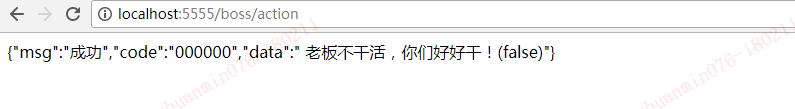
* 配置项



* Java 代码



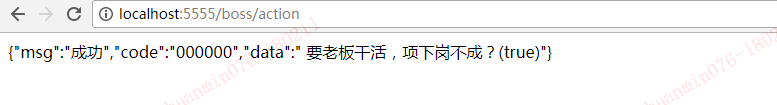
* 首次访问情况



* 修改配置



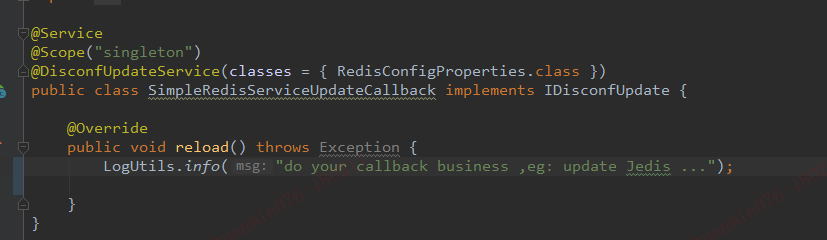
* 再次访问：



### 刷新回调

回调类，实现IDisconfUpdate 接口，重写reload方法实现刷新后，自定义逻辑。

比如：redis 主机切换后，需要重新设置Jedis ，这里可以重新生成Jedis 实例，或者连接池。

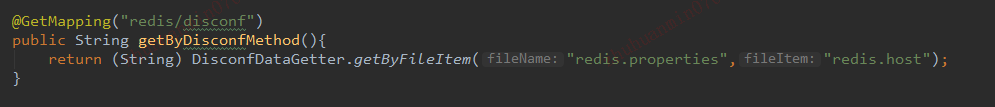


修改配置后，可以看到日志，reload 方法被触发



### 获取配置方法

从配置文件中，获取某个配置项的值



## 代码位置

测试环境：<http://icore-css-disconf-stg1.paic.com.cn>

生产环境：<http://icore-css-app-disconf.paic.com.cn>

## DISCONF原理

参考之前分享的PPT.

http://git-ma.paic.com.cn/ZHUHUANMIN076/icore-claim-demo/blob/master/tech\_share/2017-11-10%20%E5%88%86%E5%B8%83%E5%BC%8F%E9%85%8D%E7%BD%AE%E4%B8%AD%E5%BF%83%EF%BC%88%E6%9C%B1%E6%AC%A2%E6%95%8F%EF%BC%89/%E5%88%86%E5%B8%83%E5%BC%8F%E9%85%8D%E7%BD%AE%E4%B8%AD%E5%BF%83.pptx

## 常见问题