配置中心V1.0

详细设计文档

# 背景介绍

配置中心主要是核心价值就是把目前开发中的系统，包括开发，测试，线上系统所有配置内容统一管理起来。

配置中心的优势如下：

1. 配置的统一管理。配置信息不会散落在各个项目中，统一管理，简化配置维护
2. 配置的依赖的同步变更和配置变更通知，保证相同配置内容，在不同项目配置内容不会有不一致的表述问题
3. 彻底解决线下，线上配置容易人为配置错误的问题

初期版本只支持K-V结构的配置信息管理。

# 配置中心支持的范围

配置中心可支持的配置类型如下：

1. 系统运行环境配置。包括数据库，缓存，新权限，天路等项目所依赖的外围服务配置
2. 非常量方式的项目内部配置参数。如是否开启日志，是否开启验证等，内容为k-v类型。
3. 需要经常变化的项目业务配置参数。与业务相关参数，需要时常修改需求的，内容为k-v类型。

配置中心不支持的配置需求

1. 不变的常量，建议放入到系统内部中处理
2. 非k-v结构的配置，建议放入到系统内部中处理

# 整体模块设计方案



配置中心服务采用独立部署的方案，项目是一个WEB工程项目。

SDK 客户端目前暂只支持Java. RPC协议采用http+json序列化？

# 配置中心模型设计



模型说明：

* Project 工程模型。工程的名称唯一
* Environment运行环境，如rd环境，qa环境，线上环境等。工程下可以建多个运行环境。
* Version 版本模型。某个运行环境下，可以建立多个版本号。部署时必须要指定版本号
* Group 分组模型。用于对配置项进行分类查看
* ConfigItem 配置项模型。
* ItemType 配置项类型。用于对填写的内容进行校验
* User 用户模型

用户模型有两个密码。

password是控制台登录密码，可以登入控制台

apiPassword是api访问密码，只能用于api访问时使用。不能用于登入控制台

* AccessSetting 权限设置，绑定用户与运行环境和用户分组

# 配置中心功能设计

## 用户管理&授权模块

* 查看用户列表界面

url: /config/listUsers.action

参数: 无

返回结果 WebContent/config/users.vm

权限(admin)

用户列表:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | 用户名 | 备注 | 创建时间 | 更新时间 | 操作 |
|  |  |  |  |  | 修改删除 |

* 创建用户列表

url: /config/addUser.action

参数: username, password,memo

返回结果: 返回用户列表

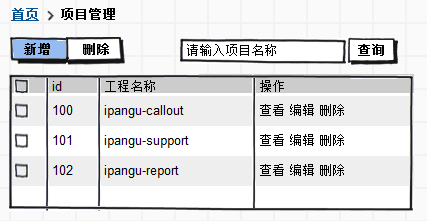
权限(admin)

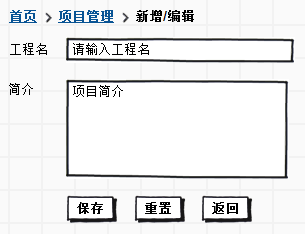
## 配置管理模块

配置管理模块提供key-value管理功能，主要有如下功能

4.2.1

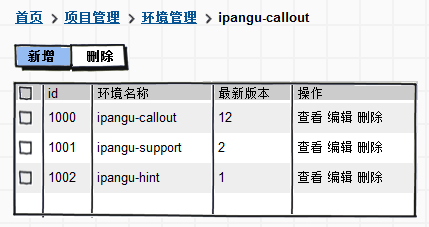
项目管理，提供新增，编辑，模糊查询功能

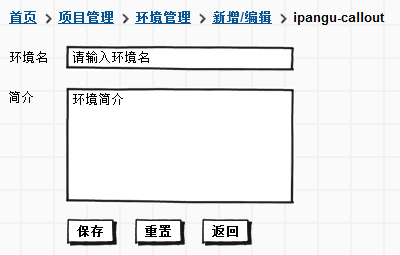




4.2.2 环境分类管理

运行环境，如rd环境，qa环境，线上环境等。工程下可以建多个运行环境



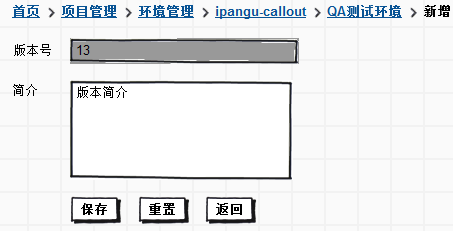


4.2.3 版本管理

版本管理，一个环境下可以有多个版本，便于回滚和查看

版本默认只增不减，新增时，版本号不可以输入，默认取当前最大版本号+1





4.2.4分组管理

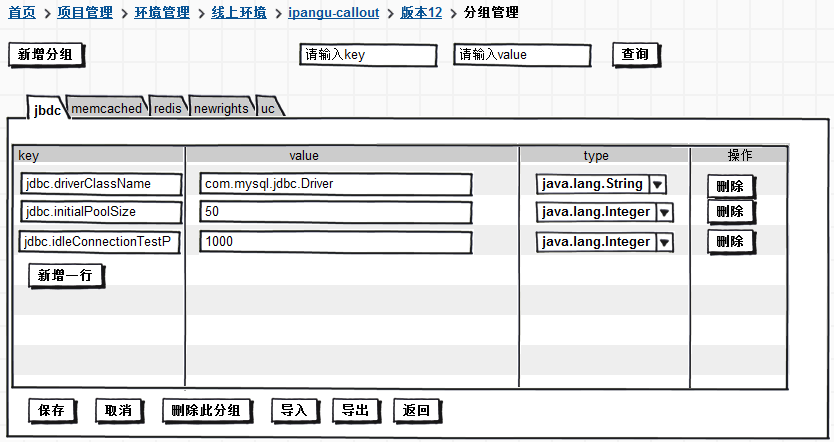
按类型管理key->value，如jdbc一组，memcached一组

提供按key和value的查询功能

录入时，自动校验key是否冲突

导入：支持上传properties，自动解析成key-value

导出：导出成properties文件

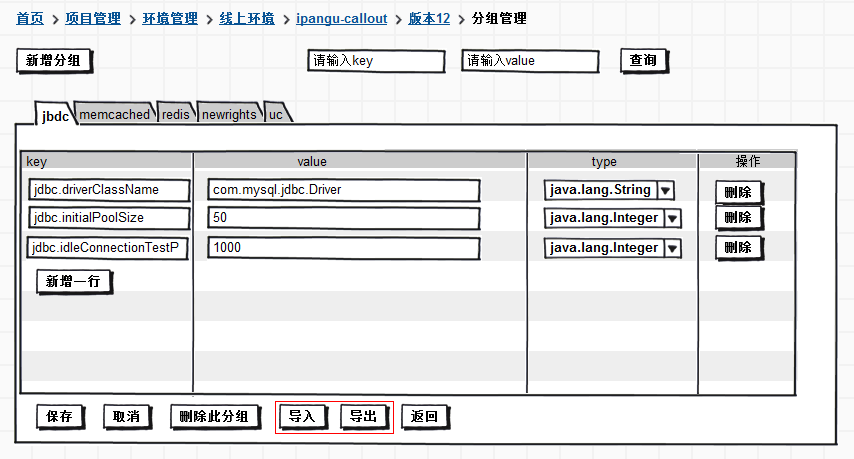


## 导入导出模块

* + 1. 导入导出功能

可以上传properties，自动解析成key-value，可将配置生成properties下载

功能放在分组管理每个tab面的下面



* + 1. 复制功能

为批量配置提供便利，支持指定project、environment、group、version复制所有配置项，填写相关直接生成新的配置

源和目标的选项必须相同，幂等，最少要选到版本号，如：

源 : ipangu-callout/QA测试环境/版本12/

目标： ipangu-callout/RD开发环境/版本1/

目标的子配置项必须为空



* + 1. Diff功能

实现两个版本的配置对比功能。特别是针对前后版本的变化情况对比查看功能。

## 系统设置模块

系统设置模块用于对系统的配置信息进行设置。如是否开启导入功能等。

# SDK设计方案

SDK是配置中心提供给项目访问配置中心服务的API集。

功能包括：提供RPC访问的API实现以及配置服务接口访问功能。

RPC访问时必须提供用户名，密码(加密)才可以访问.

线上项目登录配置中心的用户名和密码管理方案：

由OP在jvm启动参数中以环境变量的方式配置用户名和密码。

## 配置读取接口

/\*\*

\* 根据版本号获取所有配置项内容

\* **@param**user用户名

\* **@param**password密码

\* **@param** version 版本号

\* **@return** Map<String,String>返回所有的值类型都是String

\*/

**public**Map<String,String> getConfigItems(String user, String password, Long version);

/\*\*

\* 根据运行环境id获取最新一份版本的配置项内容

\* **@param**user用户名

\* **@param**password密码

\* **@param**envId运行环境id

\* **@return** Map<String,String>返回所有的值类型都是String

\*/

**public**Map<String,String> getLastestConfigItems(String user, String password, Long envId);

/\*\*

\* 根据版本号、key获得某项配置内容

\* **@param**user用户名

\* **@param**password密码

\* **@param** version 版本号

\* **@param** key key

\* **@return** String

\*/

**public** String getConfigItemValue(String user, String password, Long version, String key);

## 配置回调实现流程与接口定义

回调的流程说明：

每次修改后，在版本模型中都会更新 checkSumDate的属性，内容为最新的更新时间。同时会把createCheckSum设置为true，表示此时可以手工设置checkSum字段操作，生成checkSum字串符，客户端都会维护这个checkSum字符串（从配置中心加载服务后就会保存到本地内存中），如果这个checkSum发生变化配置中心会把当前所有的配置重新发到客户端，由客户端做diff找出变更的内容，再触发ConfigItemChangedCallable这个接口的回调，同时客户会也会更新最新的这个checkSumDate属性。

回调的监听由客户端定时发送http请求进行交互。checkSum状态不会进行缓存，由多个主库分担压力。变化的结果由于存在一定的查询数据量，可以加入缓存，缓存时间不能超过5秒，解决同时多个服务一起拉取变化结果的并发压力问题。

流程图如下：



**publicinterface** ConfigItemChangedCallable {

/\*\*

\* 配置中心值变更回调接口

\* **@param** changedConfigItemList 发生变更的key列表

\* ChangedConfigItem有三个属性 key/oldValue/newValue

\*

\*/

**publicvoid** changed(List<ChangedConfigItem> changedConfigItemList);

}

# 审计模块功能设计?

审计模块主要是提供日志的记录与查询功能。

审计日志将采用异步保存到数据库的实现。

# 与BJF集成方案

## 配置SDK加载支持

支持HTTP协议加载。

通过扩展Spring的PlaceholderConfigurer,可以保障对BJF无缝的迁移实现

配置示例如下：

<beanid=*"propertyConfigurer"*class=*"com.baidu.cc.spring.ConfigCenterPropertyPlaceholderConfigurer"*>

<propertyname=*"enableUpdateCallback"*value=*"false"*></property>

<propertyname=*"connectionTimeout"*value=*"2000"*></property>

<propertyname=*"readTimeout"*value=*"2000"*></property>

<propertyname=*"ccServerUrl"*value=*"127.0.0.1"*></property>

<propertyname=*"ccUser"*value=*"admin"*></property>

<propertyname=*"ccPassword"*value=*"admin"*></property>

<propertyname=*"ccVersion"*value=*"1"*></property>

<propertyname=*"ccEnvId"*value=*"1"*></property>

</bean>

*enableUpdateCallback*：是否支持配置变更回调操作

*connectionTimeout*：连接配置中心超时设置

*readTimeout*：访问配置中心超时设置

注：配置中心的用户名，密码，服务器域名,版本号，环境id可以在环境变量中指定后覆盖配置的值。

## 支持老系统应用BJF框架导入配置到指定的版本中

该功能可以很方便让原来不支持配置中心的项目快速使用配置中心应用。

配置如下：

<beanclass=*"com.baidu.cc.spring.ConfigCenterPropertyExtractor"*>

<propertyname=*"connectionTime"*value=*"2000"*></property>

<propertyname=*"readTime"*value=*"5000"*></property>

<propertyname=*"ccServerUrl"*value=*"127.0.0.1"*></property>

<propertyname=*"ccUser"*value=*"admin"*></property>

<propertyname=*"ccPassword"*value=*"admin"*></property>

<propertyname=*"ccVersion"*value=*"1"*></property>

<propertyname=*"ccEnvId"*value=*"1"*></property>

</bean>

为了安全，只有针对空的版本id，才可以导入。同时线上的服务将关闭该导入的功能

导入只显示成一组。

## 配置变更通知

为了简化配置变更的使用，提供注解的方式进行方法回调支持。

要求如下：

方法的第一个参数必须为List<ChangedConfigItem>类型

对象必须注入到Spring容器中。

方法必须是public方法。

使用示例

@ConfigChangeNotifer(keys = “key1,key2,myk\*”)

public void configChanged(List<ChangedConfigItem>) {

}

增加注解扫描配置到Spring配置文件中

<bean

class="com.baidu.bjf.beans.context.annotation.CommonAnnotationBeanPostProcessor">

<property name="callback">

<bean

class="com.baidu.cc.spring.ConfigChangeNotiferAnnotationParser">

</bean>

</property>

</bean>

## 配置导入接口

为了支持现有系统的配置快速迁移进入配置中心服务，提供迁移的API以支持从原有的系统把配置内容导入到配置中心服务。

待讨论，API可导入值的风险

## 配置本地缓存支持

配置本地物理缓存方案：

提供一个设置开关，由客户端决定是否启用缓存功能。缓存的内容包括checkSum内容与所有的配置项内容。默认是关闭状态，即不进行缓存。

本地缓存的过程如下：

1. 当应用启动时，会尝试连接配置中心服务器，如果配置中心服务器连接成功，则会把配置内容获取到本，更新本地的缓存配置文件。此时在正常运行过程中，有配置变更时都会更新本地配置文件（异步）。
2. 当应用启动时，如果配置中心服务器连接失败，则会使用本地缓存配置文件。在运行过程中，会心跳检测远程配置的中心服务器，如果配置中心服务器连接成功。会重新检查本地的checkSum与远程配置中心服务器的checkSum是否相同，如果不相同，则会把变更信息同步下来，同时更新本地的配置文件（异步）。

<beanid=*"propertyConfigurer"*class=*"com.baidu.cc.spring.ConfigCenterPropertyPlaceholderConfigurer"*>

<propertyname=*"enableUpdateCallback"*value=*"false"*></property>

<propertyname=*"connectionTimeout"*value=*"2000"*></property>

<propertyname=*"readTimeout"*value=*"2000"*></property>

<propertyname=*"enableLocalCache"*value=*"true"*></property>

<propertyname=*"localCacheFile"*value=*"/mfs/path"*></property>

</bean>

*enableLocalCache和localCacheFile两个参数可以通过环境变更进行覆盖*

# 配置中心部署模型

1. 配置中心部署在jpaas上，提供两台以上的WEB服务器作为集群提供服务。前面挂接BFE，提供http服务
2. 数据库挂接dbproxy，主从结构。



# 与乐高集成方案

乐高在项目管理时，可直接生成配置中心的所有需要的配置生成工作。

# 开发计划

|  |  |  |  |
| --- | --- | --- | --- |
| 开发任务 | 预估功能点 | 人力 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

RD 人力需求：缺口

FE 人力需求