

Sever-conf 配置中心使用手册

目 录

一、简介.....	1
1.1 概述	1
1.2 特性	2
二、快速入门.....	2
2.1 “接入 Server-CONF 的示例项目”、相关配置	3
A、引入 maven 依赖.....	3
B、添加 “Server-CONF 配置信息”	3
C、设置 “Server-CONF 配置工厂”	3
三、客户端配置获取.....	4
3.1 方式 1: API 方式	4
3.2 方式 2: @ServerConf 注解方式	5
3.3 方式 3: XML 占位符方式	5
3.4 其他方式: 配置变更监听	6
四、管理端使用.....	6
五、示例 demo 代码	6

一、简介

1.1 概述

Server-CONF 是一个轻量级分布式配置管理平台，拥有“轻量级、秒级动态推送、多环境、跨语言、跨机房、配置监听、权限控制、版本回滚”等特性，开箱即用。

1.2 特性

- 简单易用: 接入灵活方便, 一分钟上手;
- 轻量级: 部署简单, 不依赖第三方服务;
- 配置中心 HA: 配置中心支持集群部署。
- 多环境支持: 单个配置中心集群, 支持自定义多套环境, 管理多个环境的配置数据; 环境之间相互隔离;
- 多数据类型配置: 支持多种数据类型配置, 如: String、Boolean、Short、Integer、Long、Float、Double 等;
- 跨语言: 底层通过 http 服务 (long-polling) 拉取配置数据并实时感知配置变更, 从而实现多语言支持。
- 跨机房: 得益于配置中心集群关系对等特性, 集群各节点提供平等的配置服务; 因此, 异地跨机房部署时, 只需要请求本机房配置中心即可, 实现异地多活;
- 高性能: 得益于配置中心的 "磁盘配置" 与客户端的 "LocalCache", 因此配置服务性能非常高; 单机可承担大量配置请求;
- 实时性: 秒级动态推送; 配置更新后, 实时推送配置信息, 项目中配置数据会实时更新并生效, 不需要重启线上机器;
- 配置变更监听功能: 可开发 Listener 逻辑, 监听配置变更事件, 可据此动态刷新等高级功能;
- 配置备份: 配置数据同时在磁盘与 MySQL 中存储和备份, 并定期同步, 提高配置数据的安全性;
- 多种获取配置方式: 支持 "API、注解、XML 占位符" 等多种方式获取配置, 可灵活选择使用;
- 15、兼容 Spring 原生配置: 兼容 Spring 原生配置方式 "@Value"、"\${}" 加载本地配置功能; 与分布式配置获取方式隔离, 互不干扰;
- 16、分布式: 支持多业务线接入并统一管理配置信息, 支撑分布式业务场景;
- 17、项目隔离: 以项目为维度管理配置, 方便隔离不同业务线配置;
- 18、客户端断线重连: 设置守护线程, 周期性检测客户端连接、配置同步, 提高异常情况下配置稳定性和时效性;
- 19、空配置处理: 主动缓存 null 或不存在类型配置, 避免配置请求穿透到远程配置 Server 引发雪崩问题;
- 20、访问令牌(accessToken): 为提升系统安全性, 配置中心和客户端进行安全性校验, 双方 AccessToken 匹配才允许通讯;

二、快速入门

2.1 “接入 Server-CONF 的示例项目”、相关配置

参考 demo 项目：server-conf-demo-springboot 供大家参考学习。

A、引入 maven 依赖

```
<!-- server-conf-client -->
<dependency>
    <groupId>com.wshsoft</groupId>
    <artifactId>server-conf-core</artifactId>
    <version>${project.parent.version}</version>
</dependency>
```

B、添加 “Server-CONF 配置信息”

参考配置文件：/src/main/resources/application.properties 内容：

```
# 配置中心跟地址，必填；
server.conf.admin.address=http://localhost:8080/server-conf-admin

# 环境配置，必填；如"test、product"等，指定配置加载环境；
server.conf.env=test

# 配置中心接入验证TOKEN，选填，非空时启用，进行安全严重
server.conf.access.token=

# 配置快照文件地址，必填；会周期性缓存到本地快照文件中，当从配置中心获取配置失败时，将会使用使用本地快照文件中的配置数据；提高系统可用性；
server.conf.mirrorfile=/data/applogs/server-conf/server-conf-mirror-sample.properties
```

C、设置 “Server-CONF 配置工厂”

可参考配置文件

src/main/java/com/wshsoft/conf/sample/config/ServerConfConfig.java 配置:

```
@Bean
    public ServerConfFactory ServerConfFactory() {

        ServerConfFactory ServerConf = new
ServerConfFactory();
        ServerConf.setAdminAddress(adminAddress);
        ServerConf.setEnv(env);
        ServerConf.setAccessToken(accessToken);
        ServerConf.setMirrorfile(mirrorfile);

        logger.info(">>>>>>>>> server-conf config init.");
        return ServerConf;
    }
```

三、客户端配置获取

Server-CONF 提供多种配置方式, 包括 "API、 @ServerConf、XML" 等多种配置方式, 介绍如下。

3.1 方式 1: API 方式

参考 "IndexController" 代码如下:

```
String paramByApi =ServerConfClient.get("default.key01",null);
```

用法:

代码中直接调用 API 即可, 示例代码 ""ServerConfClient.get("key", null)

优点:

- ✓ 配置从配置中心自动加载;
- ✓ 存在 LocalCache, 不用担心性能问题;
- ✓ 支持动态推送更新;
- ✓ 支持多数据类型;

3.2 方式 2: @ServerConf 注解方式

参考 "DemoConf.paramByAnno" 属性配置；示例代码

```
@ServerConf("default.key02")

public String paramByAnno;
```

用法：对象 Field 上加注解 "@ServerConf("key")"，支持设置默认值，支持设置是否开启动态刷新；

优点：

- ✓ 配置从配置中心自动加载；
- ✓ 存在 LocalCache，不用担心性能问题；
- ✓ 支持动态推送更新；
- ✓ 支持设置配置默认值；
- ✓ 可配置是否开启 "动态推送更新"；

"@ServerConf"注解属性	说明
value	配置 Key
defaultValue	配置为空时的默认值
callback	配置更新时，是否需要同步刷新配置

3.3 方式 3: XML 占位符方式

参考 "xml-test.xml" 中 "DemoConf.paramByXml" 属性配置；示例代码如下：

```
<bean id="demoConf"
class="com.wshsoft.conf.sample.demo.DemoConf">
    <property name="paramByXml"
value="$ServerConf{default.key04}" />
</bean>
```

用法：占位符方式 "\$ServerConf{key}"；

优点：

- ✓ 配置从配置中心自动加载；
- ✓ 存在 LocalCache，不用担心性能问题；
- ✓ 支持动态推送更新；

3.4 其他方式: 配置变更监听

可开发 Listener 逻辑, 监听配置变更事件; 可据此实现动态刷新等高级功能;

参考 "IndexController" 代码如下:

```
static {  
    /**  
     * 配置变更监听示例: 可监听配置变更事件等高级功能;  
     */  
    ServerConfClient.addListener("default.key01", new  
ServerConfListener() {  
    @Override  
        public void onChange(String key, String value)  
throws Exception {  
            logger.info("配置变更事件通知: {}={}", key, value);  
        }  
    });  
}
```

四、管理端使用

略...

五、示例 demo 代码

demo 示例代码