目录

[1. 设计目的 2](#_Toc508616223)

[2. 阅读对象 2](#_Toc508616224)

[3. 项目简介 2](#_Toc508616225)

[3.1 开发工具 2](#_Toc508616226)

[3.2 采用技术 2](#_Toc508616227)

[3.3 示意图 2](#_Toc508616228)

[4. 开发介绍 4](#_Toc508616229)

[4.1 数据库表 4](#_Toc508616230)

[4.2 Server端介绍 5](#_Toc508616231)

[4.3 应用端开发 7](#_Toc508616232)

## 设计目的

提供企业级的组织架构主数据管理，单点登录，会话管理，权限认证，基础服务。

子应用只需依赖sso-common，简单配置后项目中即可调用sso-server提供的主数据，权限，日志，短信等服务；轻量级、易扩展。

## 阅读对象

此文档的阅读对象包括：

* 项目开发人员

## 项目简介

### 开发工具

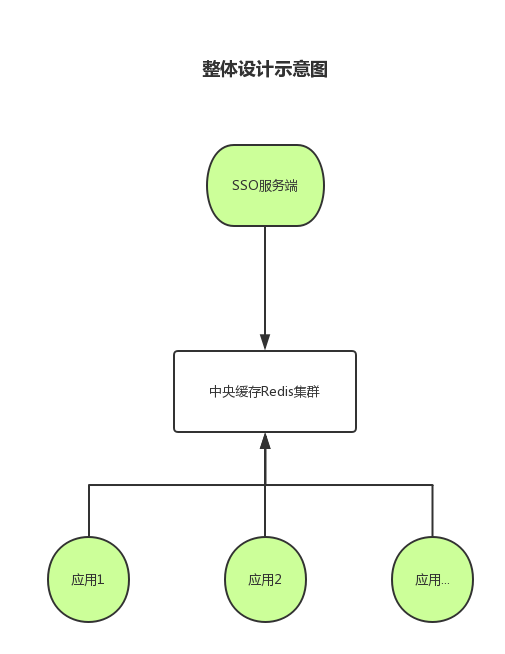
Eclipse/IDEA Maven构建 Mysql数据库 Redis缓存

### 采用技术

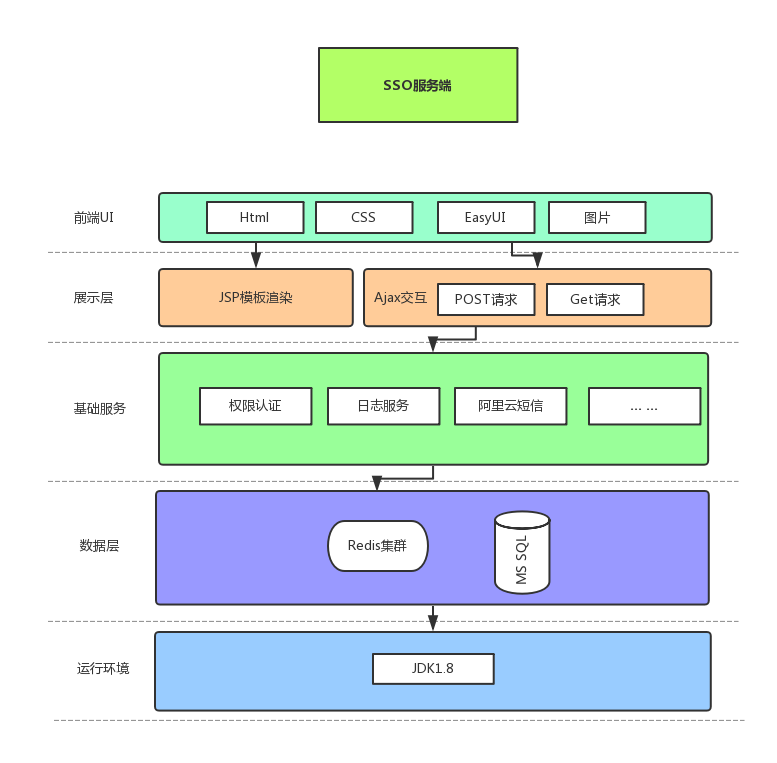
SpringBoot + Spring-Session + Mybatis + Redis + Mysql + 服务端前端easyui

### 示意图

整体设计示意图



sso-server端示意图



## 开发介绍

### 数据库设计

数据库表说明，具体字段含义详见sql脚本。



### Server端介绍

#### 组织架构权限

组织架构权限设计按照 组织(公司/部门等)/岗位/用户组/角色/菜单/权限

RBAC模型: 组织/岗位/用户组/用户 均可进行角色授予，菜单和权限分配给角色。使用Redis队列实现权限分配实时生效（参考代码实现：com.tongwei.sso.redis.UserChangeHandler/ com.tongwei.sso.redis.RoleChangeHandler）。

1.父组织、父岗位、父用户组拥有的角色，其子默认拥有。

2.用户账号可以选择挂在组织，岗位，用户组下面享有对应的角色。

3.岗位必须挂在组织下面。

4.部门和岗位从基础部门表和基础岗位表中选取。比如：基础部门财务部，唯一编码code为：cwb，测试公司编码为test，在测试测试公司下添加部门选择财务部：生成组织编码为test\_cwb，岗位同理。

5.菜单为树形结构，菜单按照应用appCode来区分，访问子应用的时候AuthUtil.getMenuTree()返回当前子应用，该用户拥有的菜单树。特别说明菜单pattern为权限控制AntPattern语法。当子应用@EnableSso注解，enableMenuPattern=true时才会验证菜单的pattern，如用户菜单，pattern为/user/\*\*，以为着用户可以访问/user/\*\*路径，其他路径不允许访问。

6.角色为树形结构，子角色默认拥有父角色的菜单、权限。角色按照应用区分，当在server端添加应用test后，会在sso端自动初始化角色system\_test，该角色即为sso端的test应用管理员，应用管理员角色只能管理其负责应用的权限与菜单等信息。system\_sso为超级管理员，负责管理组织架构，管理sso端与应用端。验证登录用户是否拥有app1下的角色role1,只需要访问app1，调用AuthUtil. hasRole(String roleCode)即可。

7.权限按照权限分组（父子）与应用appCode存储。例如访问app1，验证用户是否拥有app1的test组下的test权限。鉴权调用AuthUtil. hasPerm(String permGroupCode, String permCode)。同一app下权限分组code不能相同，同一分组下，权限code不能相同。

#### 基础服务

实现原理：redis队列，应用向中央缓存队列中添加业务数据，sso-server端负责消费数据。

1. 日志服务，使用参见sso-common包中com.tongwei.auth.util.LogUtil，实现参见sso-server端com.tongwei.sso.redis.LogHandler。
2. 短信服务，使用参见service-aliyun包中的com.tongwei.aliyun.AliyunSmsUtil，使用该类的全局开关@EnableAliyun注解。实现参见sso-server包com.tongwei.aliyun.AliyunSmsHandler
3. 开发其他服务

### 应用端开发

#### 快速开发

环境：JDK1.8

基于SpringBoot的应用：引用依赖sso-common包，启用注解@EnableSso。

1.激活sso配置文件(sso-dev,sso-pro参见sso-common包)spring.profiles.active=sso-dev,other

2.必须配置 sso.sys.appCode=your\_app\_code 且该code为在server端中添加了的应用appcode

3.其他配置介绍：

sso.sys.anno 允许应用匿名访问的地址

sso.sys.staticSuffixs 应用静态资源后缀（若该web应用存在静态资源）

sso.sys.loginUrl 应用登录地址

sso.sys.setCookieUrl 登录成功后setCookie地址（若选择sso-server的登录页面，且应用的HttpSessionStrategy为默认CookieHttpSessionStrategy）

sso.sys.successUrl 应用登录成功后默认跳转的地址

[4. @EnableSso](mailto:4.@EnableSso) 注解介绍，参见源码

非Springboot应用：配置ssoFilter，配置sso-common的redis连接，暂未提供代码，自行配置。

#### 使用示例

参见sso-server端com.tongwei.sso.QuickStartExample

sso-common提供了访问日志的注解@AccessLog，权限认证注解@RequireUsers，@RequireRoles，@RequirePerms。

Jsp标签鉴权：参见/sso-common/src/main/resources/META-INF/Auth.tld

工具类：

com.tongwei.auth.util.AESUtil 加密算法，盐值由sso-server端配置，应用共享

com.tongwei.auth.util.AuthUtil 认证工具

com.tongwei.auth.util.LogUtil 日志工具

com.tongwei.auth.util.SessionUtil 会话工具

接口：

登录成功后访问验证：

com.tongwei.auth.security.rule.LoginAccessRule 进入应用后的验证规则，默认实现为com.tongwei.auth.security.rule.AllUsersAcess用户登录即可访问。Sso-server端自定义实现参见代码：com.tongwei.sso.config.SsoAcessRule 只允许system管理员访问。注意：@Primary注解。

记住登录规则接口：

com.tongwei.auth.security.rule.RememberMeRule

默认实现cookie策略：com.tongwei.auth.security.rule.CookieRememberMeRule

注意：当应用为headr策略时，记住登录验证规则实现为：com.tongwei.auth.security.rule.HeaderRememberMeRule @EnableSso中配置。