<https://github.com/modood/Administrative-divisions-of-China>

省市县区街道的数据来源于如上url，[（截止时间：2018-10-31，发布时间：2019-01-31）](https://github.com/modood/Administrative-divisions-of-China/commit/31aaa3c31dc767a01f8f5e724c795f02f4540437" \o "feat: 数据更新（截止时间：2018-10-31，发布时间：2019-01-31）)

1. 工程主要实现的内容在DemoController中都有样例说明，主要实现了如下内容
   1. 工程使用HikariDataSource数据源
   2. 工程依赖redis，请在application.yml文件中配置相关参数
   3. 使用@Data注解生成get set方法
   4. 使用@Slf4j 调用日志打印
   5. /demo2中调用的DemoService使用了基于redis的缓存注解
   6. /demo4 中抛出自定义的异常，会封装成{code:’’,message:’’}的json对象到前台
   7. /demo5 演示的是本地锁，数据存储到内存中
   8. /demo6?a=1&b=2 演示的是分布式锁，锁的数据存储于redis中。锁的key值是demo6:a的值
   9. /demo7 演示的是本地缓存锁，另外@log2db的注解会将该方法的日志输出到数据库t\_sys\_log表中
   10. /demo8 演示的是限流

单点登录及权限体系说明：



说明:

系统将拦截所有url请求，判断该请求的cookie或者header中是否含有\_TOKEN的值，如果没有，则跳转到认证系统（如是ajax请求则返回json数据结构。认证系统地址需在yml文件中到配置，稍后将提及）。如果有\_TOKEN的值，则到redis中判断该\_TOKEN的值有没有过期，如果没有过期则是有效的已登录的用户，解析redis中的value值，并建立user对象放到ThreadLocal中，同时刷新redis中的该\_TOKEN的过期时间延长30分钟；如果没有该redis值，说明已过期，需要跳转到登录界面（如果是ajax请求，则返回json数据结构）。跳转到认证中心的时候需要将serverId作为参数传递过来（否则登录成功后不知道通知哪个系统），认证中心会将原请求地址及serverId作为参数带到sso登录页面。如果是form表单登录登录成功后将回跳到serverId对应的回调地址（同时将原始地址作为参数传递过去），同时在认证中心写redis数据及cookie，业务服务器回调地址接收到请求后将\_TOKEN存储到cookie，同时redirect到原始地址；如果登录使用的ajax请求，将返回\_TOKEN的值，由前台自己保存，后续的请求都必须携带该值，否则将认为未登录。

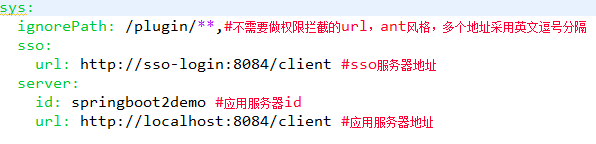
从已登录的a系统页面跳转到b系统的时候，b系统会拦截该请求，同时转发至认证服务器，由于认证服务器已经有cookie值了，顾将该cookie值去redis中校验是否有效，如果有效，则回调到b系统，后面的逻辑处理就跟a系统拦截登录一样了；如果无效则跳转到登录系统，后面的处理逻辑也跟a系统的拦截登录一致了。

退出的时候，可以在任意业务系统退出，实际上是将\_TOKEN对应的value在redis中删除，由于所有的拦截都要通过redis判定是否存在对应的value，且未过期，所以只要删除redis中对应的数值即可。

另外在业务系统及认证系统中cookie存储的key值均为\_TOKEN，且对应的value都是一样的，没有做不同系统的cookie值不一致，否则登出不好处理，太复杂。

部署：

业务系统和认证系统可以是一个工程，也可以是不同的工程，主要通过yml文件的如下进行配置：



Sys.sso.url的认证服务器的地址，sys.server.url是业务服务器的地址，sys.server.id是业务服务器的id（该id在所有业务服务器中唯一）。业务系统必须配这三个值，sys.sso.url和sys.server.id可以是同一个地址，表示业务服务器和认证服务器是同一个地址；也可以是不同值，代表不同地址。若是认证服务器，则这三个值都不需要配置。

已经使用的映射地址：

/auth 认证服务区登录成功后回调业务系统地址，若为ajax登录请求，则该url不需要关心。如果应用服务器为非webapp分离项目，则系统会自动处理，也不需要关心。

/logout 业务系统登出url地址，需在请求的cookie或header中传递\_TOKEN的key value键值对。若为ajax登出则返回json数据结构{code:’’,message:’’},code为1代表登出成功。

/ssoauth 业务服务器询问认证服务器当前请求是否已登录的地址。该地址在框架内部自己会处理，无需业务单独处理

/login 认证服务器的登录请求地址，接收post请求，username，password参数。如果是ajax请求将返回{code:’’,message:’’}。Code为1表示成功，此时message为token的值，由前端自己保存处理，以后每次请求都必须携带改值。若为0则表示登录失败，此时message为失败原因。

若为webapp分离项目，则需要自己通过post方法调用请求/login，并在客户端存储\_TOKEN的值，logout也需要业务端自己调用。

若不是webapp分离项目，以上4个url业务系统和认证系统会自己处理，不需要code，只需要配置yml中的三个参数即可。

认证中心登录页面为：？？？（注意登录都是在认证系统做的，包括url地址及页面都在认证系统，而不是在业务系统）

需要注意的是业务系统及认证系统都需要配置默认欢迎页面（即地址为/映射的页面），因为业务系统登出后会自动跳转到系统根路径对应的页面地址，而认证系统若没有收到实际跳转的业务系统地址也将跳转到认证系统的根路径页面地址。

关于当前用户

使用UserUtils.getUser()获取当前登录用户。

权限拦截

自定义注解 @HasRole

在controller的方法中增加@HasRole("admin && ox")，表示必须含admin角色及ox角色才能访问该方法，同时支持&&及||还有()小括号注解，admin及ox代表角色英文名称，对应tb\_role表中的enname