租户隔离

技术摘要

租户隔离的本质主要是数据行级的隔离，根据租户列的值得不同进行隔离，同时租户的隔离级别和用户自定义的数据是同级，即每个用户可以查阅关联租户和自己创建的数据。默认字段：`tenant_id`、`crt_user_id`。 核心类：com.github.wxiaoqi.security.common.data.MybatisDataInterceptor

强制条件

用户具有两个关键属性：部门信息`depart_id`，租户信息`tenant_id`。这两个信息会在用户操作自己的相关业务的数据的时候，同步写入到业务表中，因此要求所有的需要进行租户隔离的数据都必须要以上两个属性。否则无法进行数据的行级隔离。

如何进行租户数据隔离（参考admin模块）

- 配置mybatis拦截器

```
package com.github.wxiaoqi.security.admin.config;

import
com.github.wxiaoqi.security.common.data.MybatisDataInterceptor;
import
com.github.wxiaoqi.security.common.data.IUserDepartDataService;
import org.apache.ibatis.session.SqlSessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;

import javax.annotation.PostConstruct;

/**
 * 租户\部门数据隔离
 * @author ace
 * @create 2018/2/11.
```

```
 */
@Configuration
public class MybatisDataConfig {

    @Autowired
    private SqlSessionFactory sqlSessionFactory;

    /**
     * 该方法主要是为了让当前用户可以获取授权的数据权限部门
     */
    @Autowired
    private IUserDepartDataService userDepartDataService;

    @PostConstruct
    public void init(){
        /**
         * 有些mapper的某些方法不需要进行隔离，则可以在配置忽略，按逗号隔开.
         *
如:"com.github.wxiaoqi.security.admin.mapper.UserMapper.selectOne",
表示该mapper下不进行租户隔离
         */
        sqlSessionFactory.getConfiguration().addInterceptor(new
MybatisDataInterceptor(userDepartDataService,"com.github.wxiaoqi.se
curity.admin.mapper.UserMapper.selectOne"));
    }
}
```

- Mapper开启租户隔离配置

开启租户隔离`@Tenant`，以部门模块为例

```
@Tenant
public interface DepartMapper extends CommonMapper<Depart> {

    List<User> selectDepartUsers(@Param("departId") String
departId,@Param("userName") String userName);

    void deleteDepartUser(@Param("departId")String departId,
@Param("userId") String userId);

    void insertDepartUser(@Param("id") String id,
@Param("departId") String departId, @Param("userId") String
userId,@Param("tenantId") String tenantId);

}
```

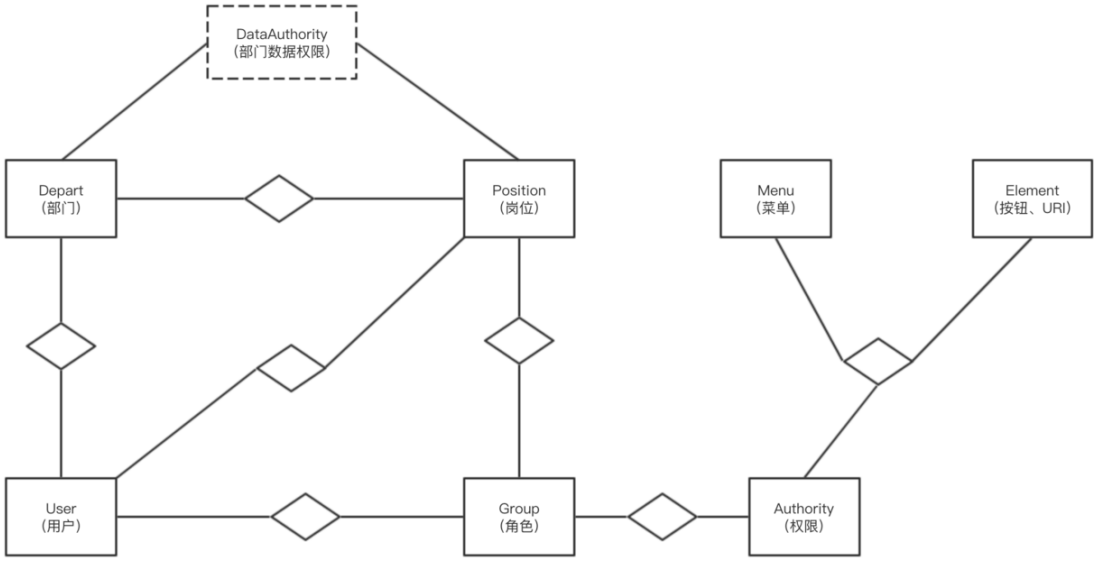
强制条件：要求mapper查询条件的主表必须具有上文提到的两个关键属性.如上述mapper中的`selectDepartUsers`，查询主表中就有具有上诉的两个属性.示例sql如下：

```
select u.name,u.username,u.id,u.sex,u.description,u.depart_id from
base_depart d
  inner join base_depart_user bdu
on bdu.depart_id = d.id
  inner join
  base_user u
on bdu.user_id = u.id
where bdu.depart_id = #{departId}
    <if test="userName!=null">
    and u.name like #{userName}
    and u.is_deleted = '0'
    and u.is_disabled = '0'
    </if>
```

至此，我们就完成了一个服务进行租户隔离的配置，下文，我们将会介绍如何在租户下进行部门的数据隔离。

部门数据权限

数据对象关系说明



技术摘要

默认情况下，用户只能看到自己的创建的数据（要求数据必须具有`crt_user_id`，`depart_id`这两个属性）。当要授予他可以查看部门的数据的时候，必须进行岗位数据权限的关联。即：用户具有某个岗位权限， 某个岗位拥有某些部门的数据权限。

如何配置部门权限（参考ace-demo/ace-demo-depart-data模块）

- mybatis拦截器配置

与租户隔离同样，只需要配置一遍mybatis的数据权限拦截器：com.github.wxiaoqi.security.common.data.MybatisDataInterceptor，即可

- Mapper配置`@Depart`注解

```
@Depart
@Tenant
public interface DepartDataTestMapper extends
CommonMapper<DepartDataTest> {

}
```

- 配置获取用户授权数据部门接口

因为用户的授权部门的数据是存放在admin数据库中，所以我们必须用FeignClient的方式进行服务调用来获取当前用户所授权的部门。

```
// Feign调用服务
@FeignClient(value = "ace-admin",configuration =
FeignApplyConfiguration.class)
public interface IUserFeign {
    /**
     * 获取所有菜单和按钮权限
     * @return
     */
    @RequestMapping(value="/user/dataDepart",method =
RequestMethod.GET)
    List<String> getUserDataDepartIds(@RequestParam("userId")
String userId);
}

// 实现本地获取部门信息的接口
@Component
public class UserDepartDataService implements
IUserDepartDataService {
    @Autowired
    private IUserFeign userFeign;
    @Override
    public List<String> getUserDataDepartIds(String userId) {
        // 获取用户授权的数据权限部门ID列表
        return userFeign.getUserDataDepartIds(userId);
    }
}
```

通过上诉的配置，我们就可以进行用户在同租户下得部门数据权限控制了。