

租户隔离

技术摘要

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有以上两个属性。否则无法进行数据的行级隔离。

租户传递

从前台到后台租户的传递，我们可以通过`url`上带有租户ID，也可以在请求头上带`X-TENANT-AUTH`来标志当前访问的租户。

核心类：

```
com.github.wxiaoqi.security.auth.filter.TenantFilter
com.github.wxiaoqi.security.auth.interceptor.TenantInterceptor
```

通过上述的两个类的初始化，后台接口可以通过`BaseContextHandler.getTenantID()`来获取当前访问的租户ID，从而配合数据插件进行行级数据的隔离。

租户数据隔离

租户隔离的本质主要是数据行级的隔离，根据租户列的值不同进行隔离，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
```

数据保存（参考部门模块）

方式一：自动对象赋值

通过封装`BusinessBiz` 保存数据方法

```
@Override
public void insertSelective(T entity) {
    // 此处会对创建对象自动赋值（包括租户、部门id）
    EntityUtils.setCreatAndUpdatInfo(entity);
    super.insertSelective(entity);
}
```

方式二：手动对象赋值

```
通过BaseContextHandler.getTenantID()来获取当前租户的id，自行对象数据赋值。
```

数据查询

方式一：Mapper 自定义注解@Tenant

- Mapper开启租户隔离配置
开启租户隔离`@Tenant`，通过这个注解，会对查询的数据sql自动进行加工。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。

方式二：通过显示传参来进行租户隔离

```
通过BaseContextHandler.getTenantID()来获取当前租户的ID，从而结合自定义查询sql来实现。
```

租户页面配置

1. 创建租户管理员用户



2. 创建租户并授予该用户



3. 创建角色并授予该用户可访问菜单和资源权限，同时分配用户可分配的菜单和资源权限



4. 特别说明

关于角色中权限分配/和权限下发的解释
权限分配：指的是该角色拥有哪些菜单的访问权限和页面按钮的操作权限
权限下发：指的是该角色能够给其自身创建的角色分配哪些菜单和按钮权限

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）。若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

- 只需要在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（已默认配置实现）

```
// 实现本地获取部门信息的接口
@Component
public class UserDepartDataService implements IUserDepartDataService {

    @Autowired
    private UserBiz userBiz;

    @Override
    public List<String> getUserDataDepartIds(String userId) {
        return userBiz.getUserDataDepartIds(userId);
    }
}
```

通过上述的配置，我们就可以进行用户在同租户下得部门数据权限控制了。

部门页面配置

1. 创建部门



2. 创建岗位，并关联用户或角色



3. 根据情况给岗位授予部门数据权限，使用用户具有某个部门的数据访问权限

