芋道源码 2022-03-07

数据权限

数据权限，实现指定用户可以操作**指定范围**的数据。例如说，针对员工信息的数据权限：

用户	数据范围
普通员工	自己
部门领导	所属部门的所有员工
HR 小姐姐	整个公司的所有员工

上述的这个示例，使用硬编码是可以实现的，并且也非常简单。但是，在业务快速迭代的过程中，类似这种数据需求会越来越多，如果全部采用硬编码的方式，无疑会给我们带来非常大的开发与维护成本。
因此，项目提供 `yudao-spring-boot-starter-biz-data-permission` 技术组件，只需要少量的编码，无需入侵到业务代码，即可实现数据权限。

> **友情提示：数据权限是否支持指定用户只能查看数据的某些字段?**
> 不支持。权限可以分成三类：功能权限、数据权限、字段权限。
> 字段权限的控制，不属于数据权限，而是属于字段权限，会在未来提供，敬请期待。

1. 实现原理

`yudao-spring-boot-starter-biz-data-permission` 技术组件的实现原理非常简单，每次对数据库操作时，他会**自动**拼接 `WHERE data_column = ?` 条件来进行数据的过滤。
例如说，查看员工信息的功能，对应 SQL 是 `SELECT * FROM system_users` ，那么拼接后的 SQL 结果会是：

用户	数据范围	SQL
普通员工	自己	`SELECT * FROM system_users WHERE id = 自己`
部门领导	所属部门的所有员工	`SELECT * FROM system_users WHERE dept_id = 自己的部门`
HR 小姐姐	整个公司的所有员工	`SELECT * FROM system_users` **无需拼接**

明白了实现原理之后，想要进一步加入理解，后续可以找时间 Debug 调试下 DataPermissionDatabaseInterceptor 类的这三个方法：

- `#processSelect(...)` 方法：处理 SELECT 语句的 WHERE 条件。
- `#processUpdate(...)` 方法：处理 UPDATE 语句的 WHERE 条件。
- `#processDelete(...)` 方法：处理 DELETE 语句的 WHERE 条件。

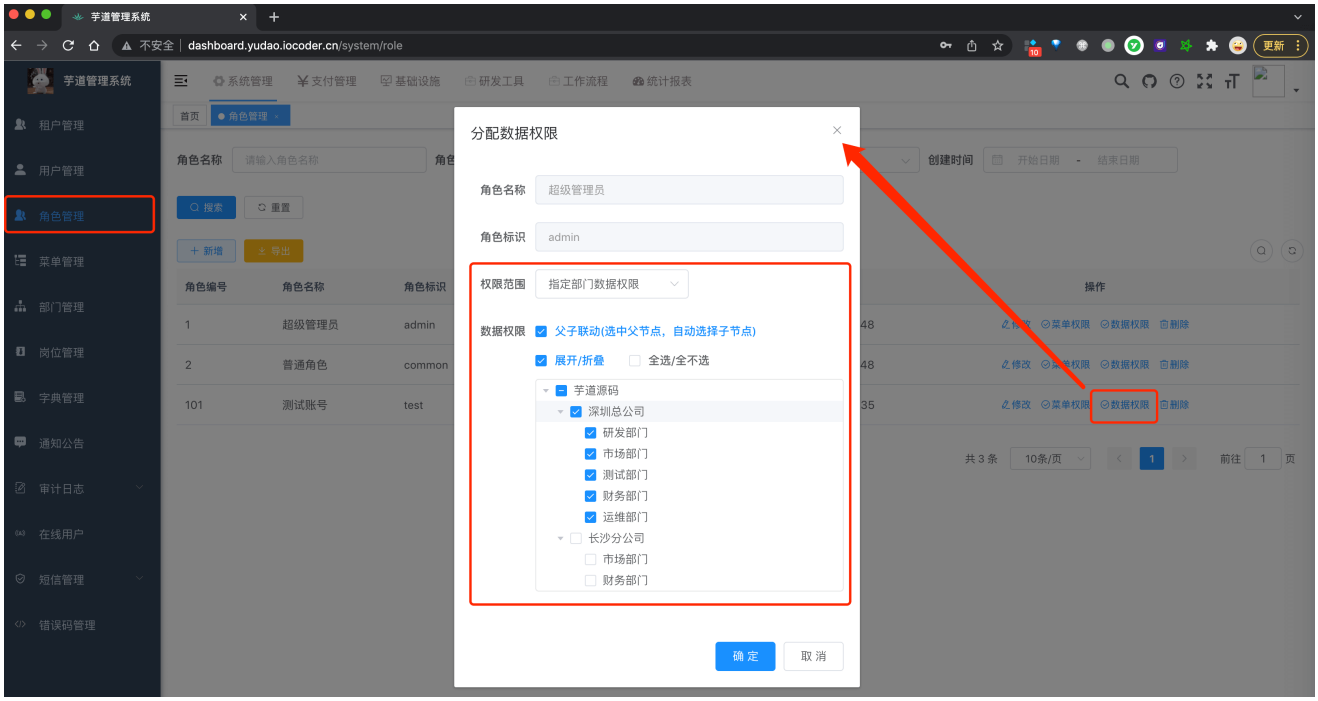
2. 基于部门的数据权限

项目**内置**了基于部门的数据权限，支持 5 种数据范围：

1. 全部数据权限：无数据权限的限制。
2. 指定部门数据权限：根据实际需要，设置可操作的部门。
3. 本部门数据权限：只能操作用户所在的部门。
4. 本部门及以下数据权限：在【本部门数据权限】的基础上，额外可操作子部门。
5. 仅本人数据权限：**相对特殊**，只能操作自己的数据。

2.1 后台配置

可通过管理后台的 [系统管理 -> 角色管理] 菜单，设置用户角色的数据权限。



> **实现代码?**
>
> 可见 DeptDataPermissionRule 数据权限规则。

2.2 字段配置

每个 Maven Module， 通过自定义 DeptDataPermissionRuleCustomizer Bean，配置哪些表的哪些字段，进行数据权限的过滤。以 `yudao-module-system` 模块来举例子，代码如下：

```
@Configuration(proxyBeanMethods = false)
public class DataPermissionConfiguration {

    @Bean
    public DeptDataPermissionRuleCustomizer sysDeptDataPermissionRuleCustomizer(
        return rule -> {
            // dept 基于部门的数据权限
            rule.addDeptColumn(AdminUserDO.class); // WHERE dept_id = ?
            rule.addDeptColumn(DeptDO.class, "id"); // WHERE id = ?

            // user 基于用户的数据权限
            rule.addUserColumn(AdminUserDO.class, "id"); // WHERE id = ?
//              rule.addUserColumn(OrderDO.class); // WHERE user_id = ?
        };
    }

}
```

注意，数据库的表字段必须添加：

- 基于【部门】过滤数据权限的表，需要添加部门编号字段，例如说 `dept_id` 字段。
- 基于【用户】过滤数据权限的表，需要添加部门用户字段，例如说 `user_id` 字段。

3. @DataPermission 注解

`@DataPermission` 数据权限注解，可声明在类或者方法上，配置使用的数据权限规则。

① `enable` 属性：当前类或方法是否开启数据权限，默认是 `true` 开启状态，可设置 `false` 禁用状态。

也就是说，数据权限默认是开启的，无需添加 `@DataPermission` 注解

也就是说，数据权限默认是开启的，无需添加 `@DataPermission` 注解

也就是说，数据权限默认是开启的，无需添加 `@DataPermission` 注解

使用示例如下，可见 UserProfileController 类：

```
// UserProfileController.java

@GetMapping("/get")
@Operation(summary = "获得登录用户信息")
@DataPermission(enable = false) // 关闭数据权限，避免只查看自己时，查询不到部门。
public CommonResult<UserProfileRespVO> profile() {
    // .. 省略代码
    if (user.getDeptId() != null) {
        DeptDO dept = deptService.getDept(user.getDeptId());
```

```
            resp.setDept(UserConvert.INSTANCE.convert02(dept));
        }
        // .. 省略代码
    }
```

② `includeRules` 属性，配置生效的 DataPermissionRule 数据权限规则。例如说，项目里有 10 种 DataPermissionRule 规则，某个方法**只想**其中的 1 种生效，则可以使用该属性。
③ `excludeRules` 属性，配置排除的 DataPermissionRule 数据权限规则。例如说，项目里有 10 种 DataPermissionRule 规则，某个方法**不想**其中的 1 种生效，则可以使用该属性。

4. 自定义的数据权限规则

如果想要自定义数据权限规则，只需要实现 DataPermissionRule 数据权限规则接口，并声明成 Spring Bean 即可。需要实现的只有两个方法：

```
public interface DataPermissionRule {

    /**
     * 返回需要生效的表名数组
     * 为什么需要该方法？Data Permission 数组基于 SQL 重写，通过 Where 返回只有权限的
     *
     * 如果需要基于实体名获得表名，可调用 {@link TableInfoHelper#getTableInfo(Class)
     *
     * @return 表名数组
     */
    Set<String> getTableNames();

    /**
     * 根据表名和别名，生成对应的 WHERE / OR 过滤条件
     *
     * @param tableName 表名
     * @param tableAlias 别名，可能为空
     * @return 过滤条件 Expression 表达式
     */
    Expression getExpression(String tableName, Alias tableAlias);

}
```

- `#getTableNames()` 方法：**哪些数据库表**，需要使用该数据权限规则。
- `#getExpression(...)` 方法：当操作这些数据库表，需要额外拼接**怎么样的 WHERE 条件**。

下面，芋艿带你写个自定义数据权限规则的示例，它的数据权限规则是：

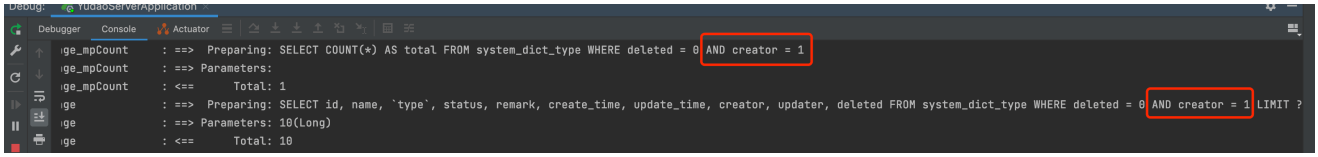
- 针对 `system_dict_type` 表，它的创建人 `creator` 要是当前用户。
- 针对 `system_post` 表，它的更新人 `updater` 要是当前用户。

具体实现代码如下：

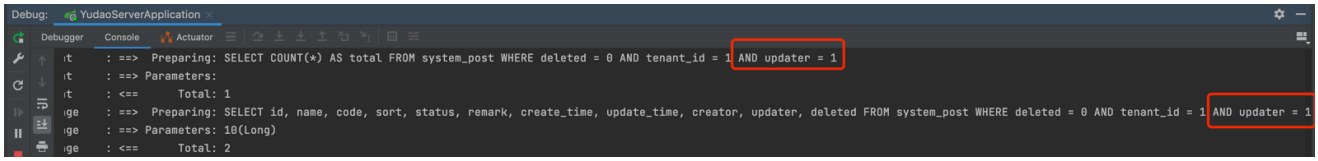
```
package cn.iocoder.yudao.module.system.framework.datapermission;

import cn.iocoder.yudao.framework.datapermission.core.rule.DataPermissionRule;
import cn.iocoder.yudao.framework.mybatis.core.util.MyBatisUtils;
import cn.iocoder.yudao.framework.security.core.util.SecurityFrameworkUtils;
import com.google.common.collect.Sets;
import net.sf.jsqlparser.expression.Alias;
import net.sf.jsqlparser.expression.Expression;
import net.sf.jsqlparser.expression.LongValue;
import net.sf.jsqlparser.expression.operators.relational.EqualsTo;
import org.springframework.stereotype.Component;

import java.util.Set;

@Component // 声明为 Spring Bean，保证被 yudao-spring-boot-starter-biz-data-permi
public class DemoDataPermissionRule implements DataPermissionRule {

    @Override
    public Set<String> getTableNames() {
        return Sets.newHashSet("system_dict_type", "system_post");
    }

    @Override
    public Expression getExpression(String tableName, Alias tableAlias) {
        Long userId = SecurityFrameworkUtils.getLoginUserId();
        assert userId != null;
        switch (tableName) {
            case "system_dict_type":
                return new EqualsTo(MyBatisUtils.buildColumn(tableName, tableAli
            case "system_post":
                return new EqualsTo(MyBatisUtils.buildColumn(tableName, tableAli
            default: return null;
        }
    }

}
```

① 启动前端 + 后端项目。

② 访问 [系统管理 -> 字典管理] 菜单，查看 IDEA 控制台，可以看到 `system_dict_type` 表的查询自动拼接了 `AND creator = 1` 的查询条件。



② 访问 [系统管理 -> 岗位管理] 菜单，查看 IDEA 控制台，可以看到 `system_post` 表的查询自动拼接了 `AND updater = 1` 的查询条件。



← 功能权限 用户体系→