简单练习

库表相关信息

1. 用户信息管理
2. 角色信息管理
3. 用户-角色关联
4. 用例树表

   具有层级结构

   同一父节点下名称不能重复

   注意父节点和叶子节点的概念

   > 一个节点没有下属节点的时候，应该转变为叶子节点
   >
   > 一个叶子节点添加了下属节点时，变成父节点
   >
   > 一般用字段进行区别`IS_LEAF`

5. 测试用例管理

   1. 用例的增删改查功能

   > 新增时，需要创建人信息
   >
   > 查询时，用例名称支持模糊查询、分页

6. 部分需要用的技术栈

> mysql：用于记录信息 redis：用于存放登录信息

库表信息

```
drop table if exists user;
/*==============================================================*/
/* Table: user                                                  */
/*==============================================================*/
create table user
(
   ID                   national varchar(32) not null comment '主键',
   LOGIN_NAME           national varchar(50) comment '用户登录名',
   USER_CHN_NAME        national varchar(50) comment '用户中文名称',
   CREATE_TIME          datetime comment '创建时间',
   UPDATE_TIME          datetime comment '更新时间',
   primary key (ID)
);
alter table user comment '用户信息表';

drop table if exists role;

/*==============================================================*/
/* Table: role                                                  */
/*==============================================================*/
create table role
(
```

```
   ID                   national varchar(32) not null comment '主键',
   ROLE_NAME            national varchar(50) comment '角色名称',
   ROLE_CODE            national varchar(50) comment '角色代号（英文代号）',
   CREATE_TIME          datetime comment '创建时间',
   UPDATE_TIME          datetime comment '更新时间',
   primary key (ID)
);

alter table role comment '角色表';

drop table if exists user_role_relation;

/*==============================================================*/
/* Table: user_role_relation                                    */
/*==============================================================*/
create table user_role_relation
(
   ID                   national varchar(32) not null comment '主键',
   ROLE_NAME            national varchar(50) comment '角色名称',
   ROLE_CODE            national varchar(50) comment '角色代号（英文代号）',
   CREATE_TIME          datetime comment '创建时间',
   primary key (ID)
);
alter table user_role_relation comment '用户-角色关系表';

drop table if exists test_repo.case_tree;

/*==============================================================*/
/* Table: case_tree                                             */
/*==============================================================*/
create table test_repo.case_tree
(
   ID                   national varchar(32) not null comment '主键',
   NODE_NAME            national varchar(50) comment '树节点名称',
   PARENT_NODE_ID       national varchar(32) comment '父节点ID',
   IS_LEAF              char(10) default '1' comment '是否叶子节点（1-是 0-否）',
   ID_PATH              national varchar(350) comment 'ID全路径',
   CREATOR              national varchar(32) comment '创建人',
   CREATE_TIME          datetime comment '创建时间',
   EDITOR               national varchar(32) comment '修改人',
   UPDATE_TIME          datetime comment '修改时间',
   primary key (ID)
);

alter table test_repo.case_tree comment '用例树表';


drop table if exists test_repo.case_info;

/*==============================================================*/
/* Table: case_info                                             */
/*==============================================================*/
create table test_repo.case_info
(
   ID                   national varchar(32) not null comment '主键',
   CASE_NAME            national varchar(100) comment '用例名称',
   NODE_ID              national varchar(32) comment '用例树节点',
   CREATOR              national varchar(32) comment '创建人',
```

```
    CREATE_TIME           datetime comment '创建时间',
    EDITOR                national varchar(32) comment '修改人',
    UPDATE_TIME           datetime comment '修改时间',
    primary key (ID)
);

alter table test_repo.case_info comment '用例信息表';
```

库表字段：

用户表：

> id、用户登录名、用户名、密码、状态

用例信息表:

> id、用例名称、状态、创建人，创建时间

打包工程

例用`maven`的插件将工程成打包成可执行的`jar`包

部署工程

1. 一个linux虚拟机环境`CentOS7`、`RedHat7`以上都可以
2. 添加工程运行环境：`Java`、`mysql`、`redis`、`nginx`
3. 将工程的`jar`包上传到服务器上启动

```
# 启动java程序的命令，退出后程序会关闭
java -jar xxxx.jar
# 后台启动java
nohup java -jar xxx.jar &
# 后台启动java并把日志输出到指定文件
nohup java -jar xxx.jar >> xxx.log 2>&1 &
# 跟踪日志
tail -f xxx.log
```

日志输出

给工程添加一个日志框架，输出相应的日志信息

日志级别	描述
OFF	关闭：最高级别，不输出日志
FATAL	致命：输出非常严重的可能会导致应用程序终止的错误。
ERROR	错误：输出错误，但应用还能继续运行。
WARN	警告：输出可能潜在的危险状况。
INFO	信息：输出应用运行过程的详细信息。
DEBUG	调试：输出更细致的对调试应用有用的信息。
TRACE	跟踪：输出更细致的程序运行轨迹。
ALL	所有：输出所有级别信息。

日志级别顺序：`ALL < TRACE < DEBUG < INFO < WARN < ERROR < FATAL < OFF` 每个日志级别都应该有其对应的信息

> 1. 存在判断分支时，要在分支的首行打印日志，用来确定进入了哪个分支
> 2. 确定通过日志可以看到整个流程
> 3. 异常捕获时，注意异常的堆栈日志不要丢失

常用的日志框架

log4j

slf4j

logback

log4j2

缓存使用

Redis

需求(2020-08-13 改动)

1. 用户登录
    1. 通过密码校验是否登录成功
    2. 密码错误时提示密码错误次数，超过5次后锁定；登录成功后清除错误次数
2. 用户的管理（增删改查）
3. 角色的管理（增删改查）
4. 用户关联角色、角色关联用户
    1. 一个用户可以关联多个角色
    2. 一个角色可以被多个用户关联
5. 用例树管理（增删改查）
6. 用例信息表（增删改查）
    1. 用例挂在用例树的叶子节点
    2. 用例树只有叶子节点才能新增用例
7. 统计相关

1. 按用户维度，统计每个用户的用例数
2. 按时间维度，统计每天的例数
3. 统计到指定日期，每个用户的用例数
4. 统计角色关联的用户数

文件上传下载