第3章-SaaS系统用户权限设计

学习目标：

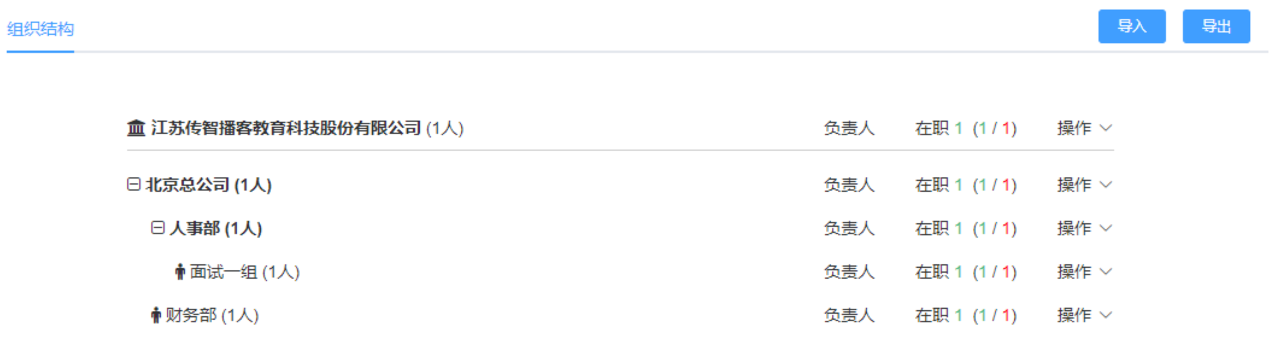
- 理解RBAC模型的基本概念及设计思路
- 了解SAAS-HRM中权限控制的需求及表结构分析
- 完成组织机构的基本CRUD操作
- 完成用户管理的基本CRUD操作
- 完成角色管理的基本CRUD操作

1 组织机构管理

1.1 需求分析

1.1.1 需求分析

实现企业组织结构管理，实现部门的基本CRUD操作



江苏传智播客教育科技股份有限公司 (1人)	负责人	在职 1 (1 / 1)	操作
北京总公司 (1人)	负责人	在职 1 (1 / 1)	操作
人事部 (1人)	负责人	在职 1 (1 / 1)	操作
面试一组 (1人)	负责人	在职 1 (1 / 1)	操作
财务部 (1人)	负责人	在职 1 (1 / 1)	操作

1.1.2 数据库表设计

```sql
CREATE TABLE `co_department` (
  `id` varchar(40) NOT NULL,
  `company_id` varchar(255) NOT NULL COMMENT '企业ID',
  `parent_id` varchar(255) DEFAULT NULL COMMENT '父级部门ID',
  `name` varchar(255) NOT NULL COMMENT '部门名称',
  `code` varchar(255) NOT NULL COMMENT '部门编码',
  `category` varchar(255) DEFAULT NULL COMMENT '部门类别',
  `manager_id` varchar(255) DEFAULT NULL COMMENT '负责人ID',
  `city` varchar(255) DEFAULT NULL COMMENT '城市',
  `introduce` text COMMENT '介绍',
  `create_time` datetime NOT NULL COMMENT '创建时间',
  `manager` varchar(40) DEFAULT NULL COMMENT '部门负责人',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

1.2 微服务实现

1.2.1 抽取公共代码

（1）在公共controller

ihrm_commoncom.模块下的 `ihrm.common.controller` 包下添加公共controller

```
package com.ihrm.common.controller;



import org.springframework.web.bind.annotation.ModelAttribute;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * 公共controller
 *      获取request，response
 *      获取企业id，获取企业名称
 */
public class BaseController {

    protected HttpServletRequest request;
    protected HttpServletResponse response;

    @ModelAttribute
    public void setReqAndResp(HttpServletRequest request, HttpServletResponse response)
{
        this.request = request;
        this.response = response;
    }

    //企业id，（暂时使用1,以后会动态获取）
    public String parseCompanyId() {
        return "1";
    }
    public String parseCompanyName() {
        return "江苏传智播客教育股份有限公司";
    }
}
```

（2）公共service

ihrm_commoncom.模块下的 `ihrm.common.service` 包下添加公共BaseService

```java
public class BaseService<T> {
    protected Specification<T> getSpecification(String companyId) {
        return new Specification<T>() {
            @Override
            public Predicate toPredicate(Root<T> root, CriteriaQuery<?> criteriaQuery,
CriteriaBuilder cb) {
                return cb.equal(root.get("companyId").as(String.class),companyId);
            }
        };
    }
}
```

1.2.2 实现基本CRUD操作

（1）实体类

在 `com.ihrm.domain.company` 包下创建Department实体类

```java
package com.ihrm.domain.company;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Transient;
import java.io.Serializable;
import java.util.Date;
import java.util.List;

/**
 * (Department)实体类
 */
@Entity
@Table(name = "co_department")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Department implements Serializable {
    private static final long serialVersionUID = -9084332495284489553L;
    //ID
    @Id
    private String id;
    /**
     * 父级ID
     */
    private String pid;
    /**
     * 企业ID
     */
    private String companyId;
    /**
     * 部门名称
     */
    private String name;
    /**
     * 部门编码，同级部门不可重复
     */
    private String code;

    /**
     * 负责人ID
     */
    private String managerId;
    /**
    *  负责人名称
    */
    private String manager;

    /**
     * 介绍
     */
    private String introduce;
    /**
     * 创建时间
     */
    private Date createTime;
}
```

（2）持久化层

在 `com.ihrm.company.dao` 包下创建DepartmentDao

```
package com.ihrm.company.dao;


import com.ihrm.domain.company.Department;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;

/**
 * 部门操作持久层
 */
public interface DepartmentDao extends JpaRepository<Department, String>,
JpaSpecificationExecutor<Department> {
}
```

（3）业务层

在 `com.ihrm.company.service` 包下创建DepartmentService

```java
package com.ihrm.company.service;

import com.ihrm.common.entity.ResultCode;
import com.ihrm.common.exception.CommonException;
import com.ihrm.common.service.BaseService;
import com.ihrm.common.utils.IdWorker;
import com.ihrm.company.dao.DepartmentDao;

import com.ihrm.domain.company.Department;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Service;
import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Predicate;
import javax.persistence.criteria.Root;
import java.util.Date;
import java.util.List;

/**
 * 部门操作业务逻辑层
 */
@Service
public class DepartmentService extends BaseService {

    @Autowired
    private IdWorker idWorker;
    @Autowired
    private DepartmentDao departmentDao;

    /**
     * 添加部门
     */
    public void save(Department department) {
        //填充其他参数
        department.setId(idWorker.nextId() + "");
        department.setCreateTime(new Date());
        departmentDao.save(department);
    }

    /**
     * 更新部门信息
     */
    public void update(Department department) {
        Department sourceDepartment = departmentDao.findById(department.getId()).get();
        sourceDepartment.setName(department.getName());
        sourceDepartment.setPid(department.getPid());
        sourceDepartment.setManagerId(department.getManagerId());
        sourceDepartment.setIntroduce(department.getIntroduce());
        sourceDepartment.setManager(department.getManager());
        departmentDao.save(sourceDepartment);
    }
    /**
     * 根据ID获取部门信息
     *
     * @param id 部门ID
     * @return 部门信息
     */
    public Department findById(String id) {
        return departmentDao.findById(id).get();
    }

    /**
     * 删除部门
     *
     * @param id 部门ID
     */
    public void delete(String id) {
        departmentDao.deleteById(id);
    }

    /**
     * 获取部门列表
     */
    public List<Department> findAll(String companyId) {
        return departmentDao.findAll(getSpecification(companyId));
    }
}
```

（4）控制层

在 `ihrm.company.controller` 创建控制器类DepartmentController

```java
package com.ihrm.company.controller;


import com.ihrm.common.controller.BaseController;
import com.ihrm.common.entity.Result;
import com.ihrm.common.entity.ResultCode;
import com.ihrm.company.service.CompanyService;
import com.ihrm.company.service.DepartmentService;
import com.ihrm.domain.company.Company;
import com.ihrm.domain.company.Department;
import com.ihrm.domain.company.response.DeptListResult;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import java.util.*;
import java.util.stream.Collectors;

/**
 * 控制器层
 */
@RestController
```

```
@RequestMapping("/company")
public class DepartmentController extends BaseController{

    @Autowired
    private DepartmentService departmentService;

    @Autowired
    private CompanyService companyService;

    /**
     * 添加部门
     */
    @RequestMapping(value = "/departments", method = RequestMethod.POST)
    public Result add(@RequestBody Department department) throws Exception {
        department.setCompanyId(parseCompanyId());
        departmentService.save(department);
        return Result.SUCCESS();
    }

    /**
     * 修改部门信息
     */
    @RequestMapping(value = "/departments/{id}", method = RequestMethod.PUT)
    public Result update(@PathVariable(name = "id") String id, @RequestBody Department
department) throws Exception {
        department.setCompanyId(parseCompanyId());
        department.setId(id);
        departmentService.update(department);
        return Result.SUCCESS();
    }

    /**
     * 删除部门
     */
    @RequestMapping(value = "/departments/{id}", method = RequestMethod.DELETE)
    public Result delete(@PathVariable(name = "id") String id) throws Exception {
        departmentService.delete(id);
        return Result.SUCCESS();
    }

    /**
     * 根据id查询
     */
    @RequestMapping(value = "/departments/{id}", method = RequestMethod.GET)
    public Result findById(@PathVariable(name = "id") String id) throws Exception {
        Department department = departmentService.findById(id);
        return new Result(ResultCode.SUCCESS,department);
    }

    /**
     * 组织架构列表
     */
    @RequestMapping(value = "/departments", method = RequestMethod.GET)
    public Result findAll() throws Exception {
        Company company = companyService.findById(parseCompanyId());
        List<Department> list = departmentService.findAll(parseCompanyId());
        return  new Result(ResultCode.SUCCESS,new DeptListResult(company,list));
    }

}
```

1.3 前端实现

1.3.1 创建模块

（1）使用命令行创建module-departments模块并引入到工程中

```
itheima moduleAdd departments
```

（2）`在src/main.js` 中注册模块

```
import departments from '@/module-departments/' // 组织机构管理
Vue.use(departments, store)
```

（3）在 `/module-departments/router/index.js` 配置路由

```
import Layout from '@/module-dashboard/pages/layout'
const _import = require('@/router/import_' + process.env.NODE_ENV)

export default [
  {
    root: true,
    path: '/departments',
    component: Layout,
    redirect: 'noredirect',
    name: 'departments',
    meta: {
      title: '组织架构管理',
      icon: 'architecture'
    },
    children: [
      {
        path: 'index',
        component: _import('departments/pages/index'),
        name: 'organizations-index',
        meta: {title: '组织架构', icon: 'architecture', noCache: true}
      }
    ]
  }
]
```

1.3.2 配置请求API

在 `/src/api/base/` 创建departments.js作为组织机构管理的API公共接口方法

```
import {
  createAPI, createFileAPI
} from '@/utils/request'

export const organList = data => createAPI('/company/departments', 'get', data)
export const add = data => createAPI('/company/departments', 'post', data)
export const update = data => createAPI(`/company/departments/${data.id}`, 'put', data)
export const detail = data => createAPI(`/company/departments/${data.id}`, 'get', data)
export const remove = data => createAPI(`/company/departments/${data.id}`, 'delete',
data)
export const changeDept = data => createAPI(`/company/departments/changeDept`, 'put',
data)
export const saveOrUpdate = data => {return data.id?update(data):add(data)}
```

1.3.3 构造列表

（1）构造基本页面样式

找到 `/module-departments/page/index.vue` ,使用element-ui提供的Card组件构造卡片式容器

```
<template>
  <div class="dashboard-container">
    <div class="app-container">
      <el-card shadow="never">
            <div class='organization-index'>
              <div class='organization-index-top'>
                <div class='main-top-title'>
                  <el-tabs v-model="activeName">
                    <el-tab-pane label="组织结构" name="first"></el-tab-pane>
                    <div class="el-tabs-report">
                      <a class="el-button el-button--primary el-button--mini" title="导
出" >导入</a>
                      <a class="el-button el-button--primary el-button--mini" title="导
出" >导出</a>
                    </div>
                  </el-tabs>
                </div>
              </div>
              <div style="overflow: scroll;white-space:nowrap"  class="treBox">
                <div class="treeCon clearfix">
                    <span>
                      <i class="fa fa-university" aria-hidden="true"></i>
                      <span ><strong>{{departData.name}}</strong></span>
                    </span>
                    <div class="fr">
                        <div class="treeRinfo">
```

```
                          <span>负责人</span>
                          <span>在职  <em class="colGreen" title="在职人数">---
</em>&nbsp;&nbsp;(<em class="colGreen" title="正式员工">---</em>&nbsp;/&nbsp;<em
class="colRed" title="非正式员工">---</em>)</span>
                        </div>
                        <div class="treeRinfo">
                          <el-dropdown class="item">
                            <span class="el-dropdown-link">
                              操作<i class="el-icon-arrow-down el-icon--right"></i>
                            </span>
                            <el-dropdown-menu slot="dropdown">
                                <el-dropdown-item>
                                  <el-button type="text" @click="handlAdd('')">添加子部门
</el-button>
                                </el-dropdown-item>
                              <el-dropdown-item>
                                <el-button type="text"
@click="handleList(organizationTree,0)">查看待分配员工</el-button>
                              </el-dropdown-item>
                            </el-dropdown-menu>
                          </el-dropdown>
                        </div>
                    </div>
                  </div>

                  <!--
                    构造树形列表
                  -->
              </div>
            </div>
      </el-card>
    </div>
</div>
</template>

<style rel="stylesheet/scss" lang="scss">
.el-dropdown {
  color: #000000
}
.el-tree-node__content>.el-tree-node__expand-icon {
  padding:0px;
}
.el-tree-node__expand-icon {
  color:#ffffff
}
.generalClassNode {
  padding-left: 20px;
}
.el-tree-node__content{
  font-size: 16px;
  line-height: 36px;
  height:36px;
}
```

```
.custom-tree-node{
  padding-left: 20px;
}
.objectTree {
  overflow: auto;
  z-index: 100;
  width: 300px;
  border: 1px solid #dcdfe6;
  margin-top: 5px;
  left: 70px;
}
.el-tabs__content {
  overflow: initial;
}
.boxpad {
  margin-left: -40px;
}
.boxpad > div:first-child,
.objectTree > div:first-child.el-tree-node > div:first-child {
  display: none;
}
</style>
<style  rel="stylesheet/scss" lang="scss" scoped>
.el-tree-node__expand-icon{

}
.el-icon-caret-right{}
.el-tree-node__content{
  font-size: 14px;
  line-height: 36px;
}
.generalClass {
  font-size: 14px;
  line-height: 36px;
  color:#000000
}
.all {
  position: relative;
  min-height: 100%;
  padding-bottom: 200px;
}
.organization-main:after,
.organization-index-top:after {
  display: block;
  clear: both;
  content: '';
  visibility: hidden;
  height: 0;
}
.organization-main {
  font-size: 14px;
  font-size: 14px;
}
```

```css
.organization-index {
  padding-bottom: 20px;
  margin-left: 20px;
}
.main-top-title {
  padding-left: 20px;
  padding-top: 20px;
  text-align: left;
}

::-webkit-scrollbar-thumb {
  background-color: #018ee8;
  height: 50px;
  outline-offset: -2px;
  outline: 8px solid #fff;
  -webkit-border-radius: 4px;
}
::-webkit-scrollbar-track-piece {
  background-color: #fff;
  -webkit-border-radius: 0;
}
::-webkit-scrollbar {
  width: 8px;
  height: 8px;
}
::-webkit-scrollbar-thumb:hover {
  background-color: #fb4446;
  height: 50px;
  -webkit-border-radius: 4px;
}
.modal-total {
  width: 100%;
  height: 100%;
  position: fixed;
  top: 0;
  left: 0;
  background: #000;
  z-index: 90;
  opacity: 0.2;
}
.modal {
  width: 400px;
  height: 300px;
  background-color: #ffffff;
  z-index: 999;
  position: absolute;
  left: 45%;
  top: 20%;
  text-align: center;
}
.treBox {
  padding: 30px 120px 0;
}
.organization-index-top {
  position: relative;

  .el-tabs-report {
    position: absolute;
    top: -50px;
    right: 15px;
  }
}
.treeCon {
  border-bottom: 1px solid #cfcfcf;
  padding: 10px 0;
  margin-bottom: 10px;
  .el-dropdown {
    color: #333;
  }
}
.treeRinfo {
  display: inline-block;
}
.treeRinfo span {
  padding-left: 30px;
}
</style>
```

（2）树形机构列表

```
                <el-tree :data="departData.children" :indent="20">
                  <div class="generalClass" slot-scope="{node,data}" style="width:99%">
                    <span>
                      <span>
                        <span>
                            <i v-if="node.isLeaf" class="fa fa-male"></i>
                            <i v-else :class="node.expanded ? 'fa fa-minus-square-o':
'fa fa-plus-square-o'"></i>
                            <span><strong>{{ data.name }}</strong></span>
                        </span>
                      </span>
                    </span>
                    <div class=fr>
                      <span class="treeRinfo">
                        <div class="treeRinfo">
                          <span>负责人</span>
                          <span>在职  <em class="colGreen" title="在职人数">---
</em>&nbsp;&nbsp;(<em class="colGreen" title="正式员工">---</em>&nbsp;/&nbsp;<em
class="colRed" title="非正式员工">---</em>)</span>
                        </div>
                        <el-dropdown class="item">
                          <span class="el-dropdown-link">
                            操作<i class="el-icon-arrow-down el-icon--right"></i>
                          </span>
                          <el-dropdown-menu slot="dropdown">
```

```
                              <el-dropdown-item>
                                <el-button type="text" @click="handlAdd(data.id)">添加子
部门</el-button>
                              </el-dropdown-item>
                              <el-dropdown-item>
                                <el-button type="text" @click="handleEdit(data.id)">编辑
部门</el-button>
                              </el-dropdown-item>
                            <el-dropdown-item>
                              <el-button type="text" @click="handleList(treeRoot,1)">查
看员工</el-button>
                            </el-dropdown-item>
                            <el-dropdown-item>
                              <el-button type="text" @click="handleDelete(data)">删除
</el-button>
                            </el-dropdown-item>
                          </el-dropdown-menu>
                        </el-dropdown>
                      </span>
                    </div>
                  </div>
                </el-tree>
```

（3）构造数据

```
  //数据绑定模型
  data() {
     return {
       activeName: 'first',   //激活pane的名称
       dialogFormVisible:false,//是否显示弹出层标识
       parentId:'',           //父id
       departData:{},         //部门列表
       formData:{}            //表单提交数据
     }
   },
   //自定义方法
   methods: {
     getObject(params) {
       organList().then(res => {
         this.departData = res.data.data
       })
     }
   },
   //钩子函数
   created: function() {
     this.getObject()
   },
```

1.3.4 组织机构的增删改查

（1）新增部门

使用element-ui提供的dialog的弹出层构造弹出添加页面

```
<el-dialog title="编辑部门" :visible.sync="dialogFormVisible">
    <el-form ref="dataForm" :model="formData" label-width="120px">
        <el-form-item label="部门名称">
            <el-input v-model="formData.name" placeholder='请输入部门名称'></el-input>
        </el-form-item>
        <el-form-item label="部门编码">
            <el-input v-model="formData.code" placeholder='请输入部门编码'></el-input>
        </el-form-item>
        <el-form-item label="部门负责人">
            <el-input v-model="formData.manager" placeholder='请输入负责人'></el-input>
        </el-form-item>
        <el-form-item label="部门介绍">
            <el-input v-model="formData.introduce" placeholder='请输入介绍'></el-input>
        </el-form-item>
    </el-form>
    <div slot="footer" class="dialog-footer">
        <el-button type="primary" @click="createData">确定</el-button>
        <el-button @click="dialogFormVisible=false">取消</el-button>
    </div>
</el-dialog>
```

配置保存方法

```
    createData:function() {
      this.formData.parentId = this.parentId
      saveOrUpdate(this.formData)
        .then(res => {
     this.$message({message:res.data.message,type:res.data.success?"success":"error"});
          location.reload()
          this.dialogFormVisible=false
      })
    }
```

（2）修改部门

1. 根据id查询部门

```
    handleEdit(id) {
      detail({id}).then( res=> {
        this.formData = res.data.data
        this.dialogFormVisible = true
        this.parentId = res.data.data.parentId
      })
    }
```

2. 调用方法更新部门

（3）删除部门

```
      handleDelete(obj) {
        this.$confirm(
          `本次操作将删除${obj.name},删除后将不可恢复，您确认删除吗？`
        ).then(() => {
          remove({id:obj.id}).then( res=> {

   this.$message({message:res.data.message,type:res.data.success?"success":"error"});
            location.reload()
          })
        })
      },
```

1.3.5 抽取组件

组件（Component）是Vue.js 最强大的功能。可以通过将不同的业务拆分为不同的组件进行开发，让代码更加优雅提供可读性。当然页可以封装可重用的代码，通过传入对象的不同，实现组件的复用。

（1）抽取新增/修改页面到 `/module-departments/components/add.vue` 中

```
<template>
    <el-dialog title="编辑部门" :visible.sync="dialogFormVisible">
      <el-form ref="dataForm" :model="formData" label-width="120px">
        <el-form-item label="部门名称">
          <el-input v-model="formData.name" placeholder='请输入部门名称'></el-input>
        </el-form-item>
        <el-form-item label="部门编码">
          <el-input v-model="formData.code" placeholder='请输入部门编码'></el-input>
        </el-form-item>
        <el-form-item label="部门负责人">
          <el-input v-model="formData.manager" placeholder='请输入部门负责人'></el-input>
        </el-form-item>
        <el-form-item label="部门介绍">
          <el-input v-model="formData.introduce" placeholder='请输入部门介绍'></el-input>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button type="primary" @click="createData">确定</el-button>
        <el-button @click="dialogFormVisible=false">取消</el-button>
      </div>
    </el-dialog>
</template>
<script>
import { saveOrUpdate } from '@/api/base/departments'
export default {
  name: 'dept-add',
  data() {
    return {
      dialogFormVisible:false,
      formData:{},
      parentId:''
    }
  },
  methods: {
    createData:function() {
      this.formData.parentId = this.parentId
      saveOrUpdate(this.formData)
        .then(res => {

 this.$message({message:res.data.message,type:res.data.success?"success":"error"});
          location.reload()
          this.dialogFormVisible=false
      })
    }
  }
}
</script>
```

（2）在 `/module-departments/page/index.vue` 中引用组件

- 导入组件

```
import deptAdd from './../components/add'  //导入组件
export default {
  //声明引用组件
  components: { deptAdd },                   //声明组件
  data() {
    return {
      deptAdd: 'deptAdd',                    //配置组件别名
      activeName: 'first',
      departData:{},
    }
  },
  ....
}
```

- 使用组件

```
//v-bind:is （绑定的组件名称）
//ref ： 引用子组件中内容的别名
<component v-bind:is="deptAdd" ref="deptAdd"></component>
```

- 改造新增修改方法

```
    handlAdd(parentId) {
      //对子组件中的属性复制
      this.$refs.deptAdd.formData = {};
      this.$refs.deptAdd.parentId = parentId
      this.$refs.deptAdd.dialogFormVisible = true;
    },
    handleEdit(id) {
      detail({id}).then( res=> {
        this.$refs.deptAdd.formData = res.data.data
        this.$refs.deptAdd.dialogFormVisible = true
        this.$refs.deptAdd.parentId = res.data.data.parentId
      })
    },
```

2 RBAC模型

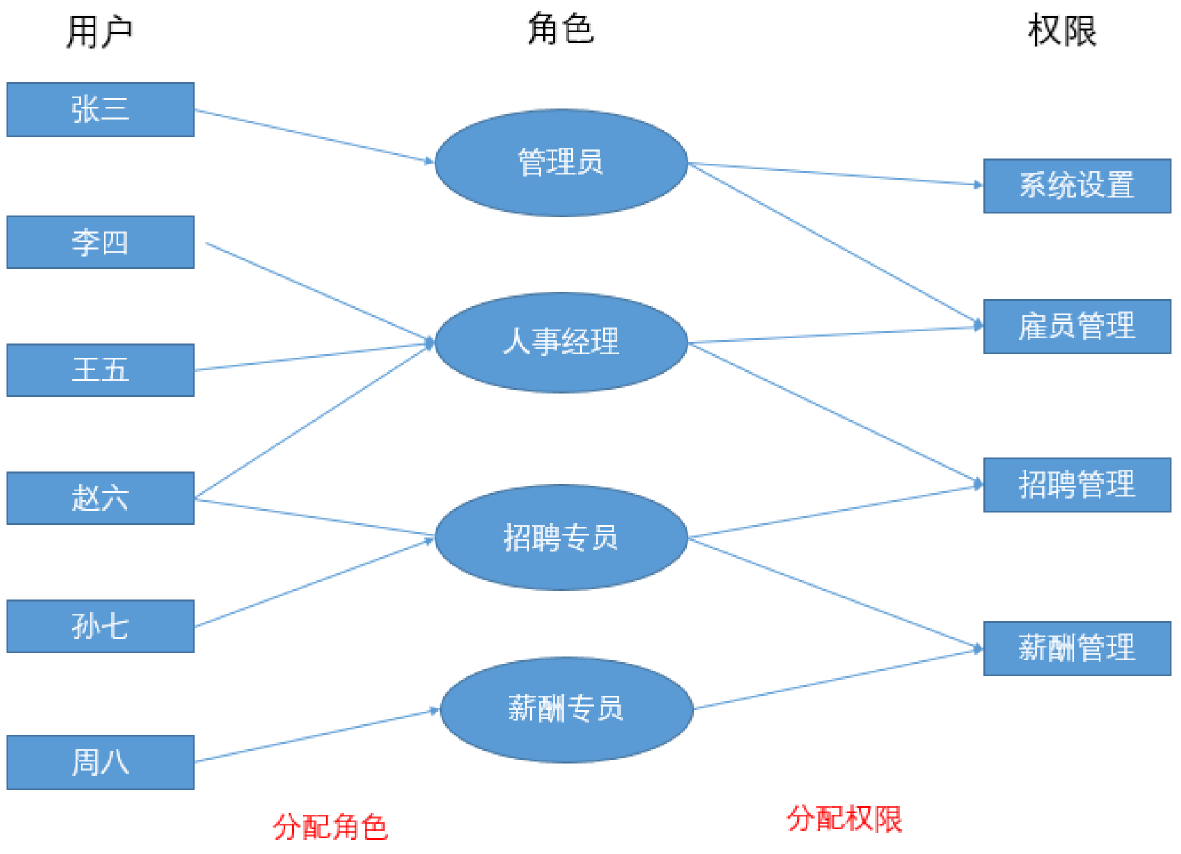
2.1 什么是RBAC

RBAC（全称：Role-Based Access Control）基于角色的权限访问控制，作为传统访问控制（自主访问，强制访问）的有前景的代替受到广泛的关注。在RBAC中，权限与角色相关联，用户通过成为适当角色的成员而得到这些角色的权限。这就极大地简化了权限的管理。在一个组织中，角色是为了完成各种工作而创造，用户则依据它的责任和资格来被指派相应的角色，用户可以很容易地从一个角色被指派到另一个角色。角色可依新的需求和系统的合并而赋予新的权限，而权限也可根据需要而从某角色中回收。角色与角色的关系可以建立起来以囊括更广泛的客观情况。

访问控制是针对越权使用资源的防御措施，目的是为了限制访问主体（如用户等） 对访问客体（如数据库资源等）的访问权限。企业环境中的访问控制策略大部分都采用基于角色的访问控制（RBAC）模型，是目前公认的解决大型企业的统一资源访问控制的有效方法

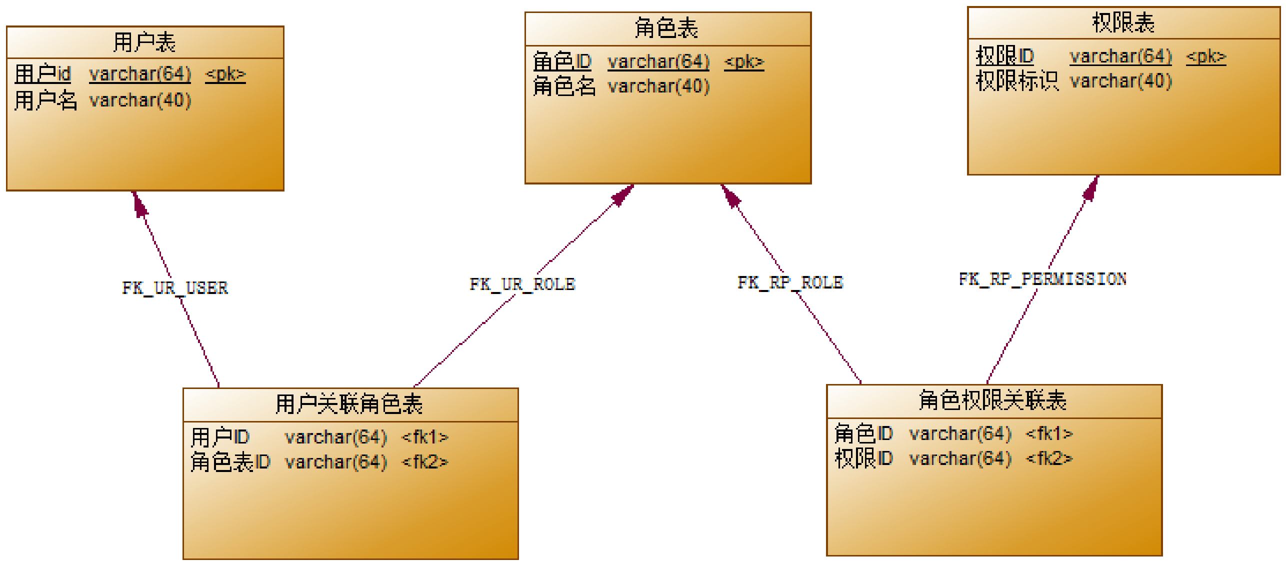
2.2 基于RBAC的设计思路

基于角色的访问控制基本原理是在用户和访问权限之间加入角色这一层，实现用户和权限的分离，用户只有通过激活角色才能获得访问权限。通过角色对权限分组，大大简化了用户权限分配表，间接地实现了对用户的分组，提高了权限的分配效率。且加入角色层后，访问控制机制更接近真实世界中的职业分配，便于权限管理。



在RBAC模型中，角色是系统根据管理中相对稳定的职权和责任来划分，每种角色可以完成一定的职能。用户通过饰演不同的角色获得角色所拥有的权限，一旦某个用户成为某角色的成员，则此用户可以完成该角色所具有的职能。通过将权限指定给角色而不是用户，在权限分派上提供了极大的灵活性和极细的权限指定粒度。

3.3 表结构分析

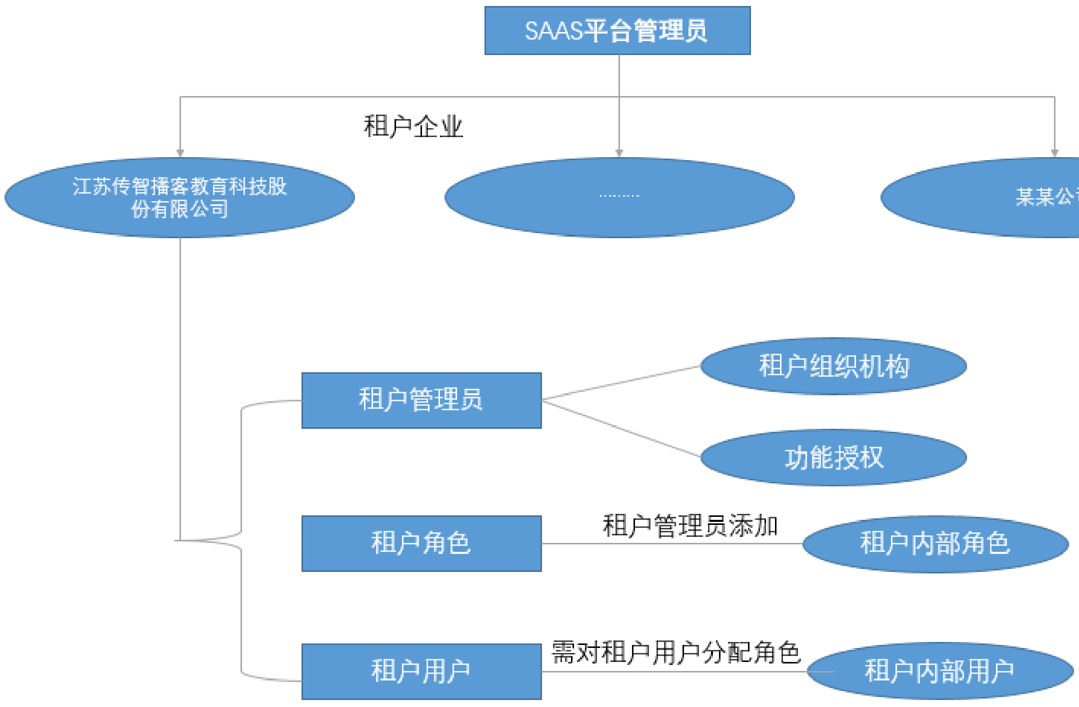


一个用户拥有若干角色，每一个角色拥有若干权限。这样，就构造成“用户-角色-权限”的授权模型。在这种模型中，用户与角色之间，角色与权限之间，一般者是多对多的关系。

3 SAAS-HRM中的权限设计

3.1 需求分析

3.1.1 SAAS平台的基本元素



SAAS平台管理员：负责平台的日常维护和管理，包括用户日志的管理、租户账号审核、租户状态管理、租户费用的管理，要注意的是平台管理员不能对租户的具体业务进行管理

企业租户：指访问SaaS平台的用户企业，在SaaS平台中各租户之间信息是独立的。

租户管理员：为租户角色分配权限和相关系统管理、维护。

租户角色：根据业务功能租户管理员进行角色划分，划分好角色后，租户管理员可以对相应的角色进行权限分配

租户用户：需对租户用户进行角色分配，租户用户只能访问授权的模块信息。

3.1.2 需求分析

在应用系统中，权限是以什么样的形式展现出来的？对菜单的访问，页面上按钮的可见性，后端接口的控制，都要进行充分考虑

- **前端**

前端菜单：根据是否有请求菜单权限进行动态加载

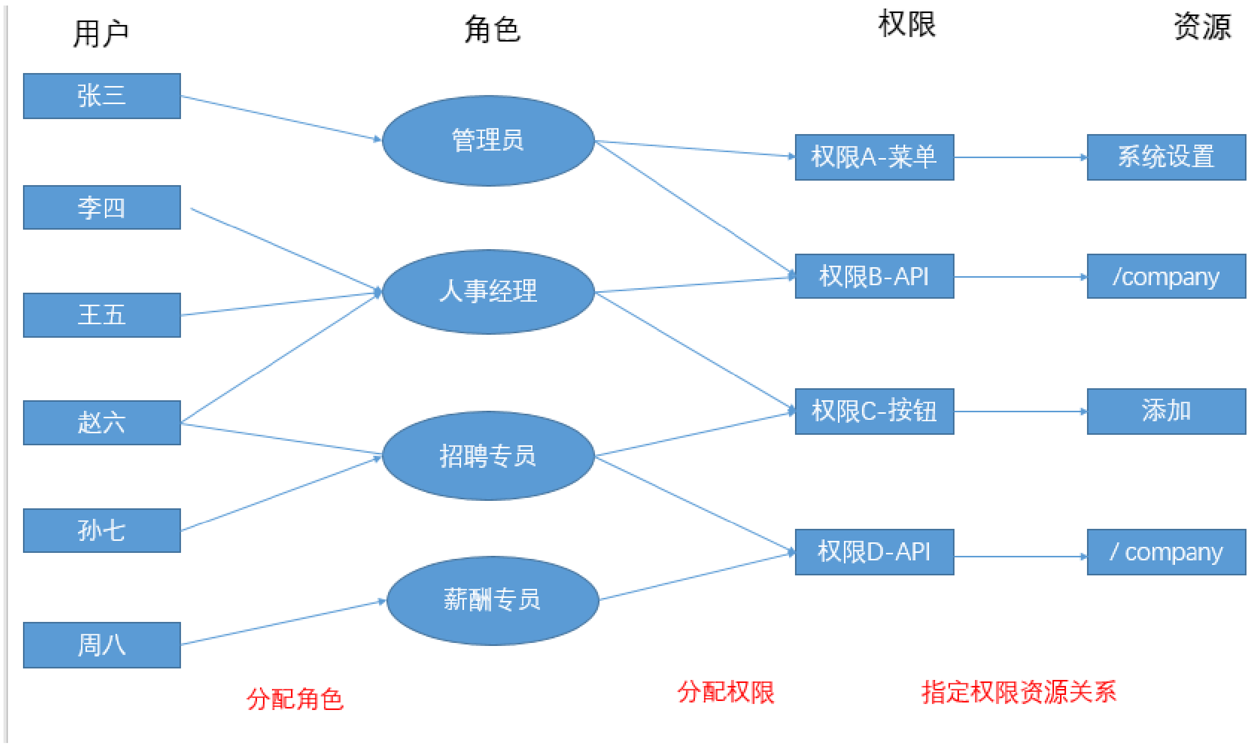
按钮：根据是否具有此权限点进行显示/隐藏的控制

- **后端**

前端发送请求到后端接口，有必要对接口的访问进行权限的验证

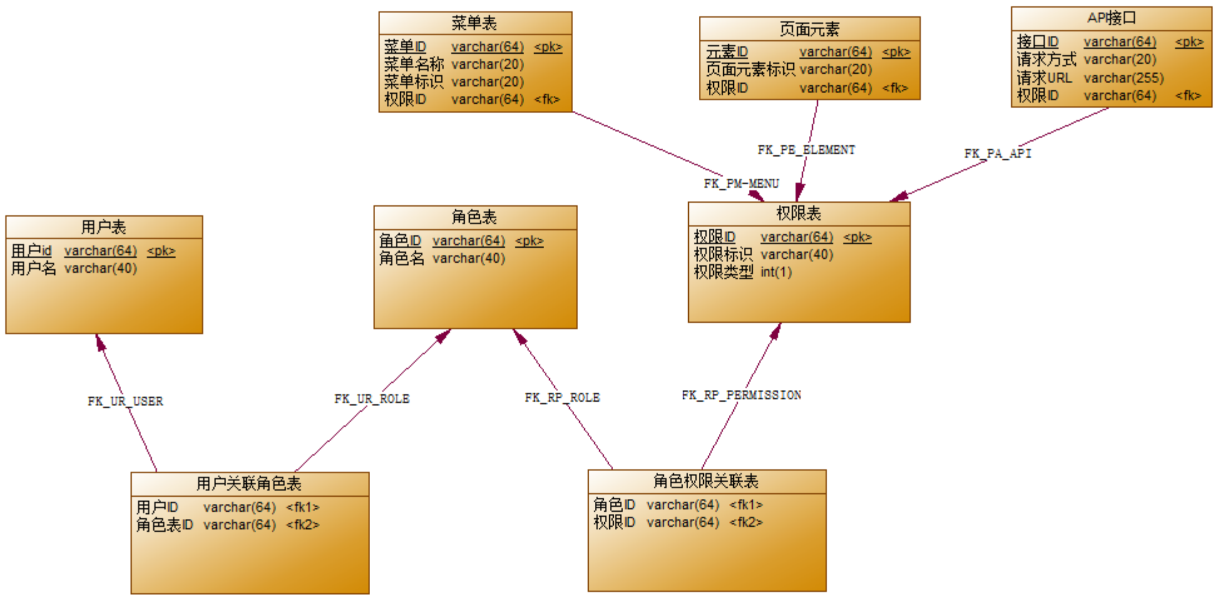
3.2 权限设计

针对这样的需求，在有些设计中可以将菜单，按钮，后端API请求等作为资源，这样就构成了基于RBAC的另一种授权模型（用户-角色-权限-资源）。在SAAS-HRM系统的权限设计中我们就是才用了此方案



针对此种权限模型，其中权限究竟是属于菜单，按钮，还是API的权限呢？那就需要在设计数据库权限表的时候添加类型加以区分（如权限类型 1为菜单 2为功能 3为API）。

3.3 表结构分析



这里要注意的是，权限表与权限菜单表、页面元素表与API接口表都是一对一的关系

与传统的RBAC模型对比不难发现此种设计的好处：

1. 不需要区分哪些是操作，哪些是资源
2. 方便扩展，当系统要对新的东西进行权限控制时，我只需要建立一个新的资源表，并确定这类权限的权限类型标识即可。

4 用户管理

4.1 需求分析

用户其实就是saas企业访问的员工，对企业员工完成基本的CRUD操作

表结构如下：

```sql
CREATE TABLE `bs_user` (
  `id` varchar(40) NOT NULL COMMENT 'ID',
  `mobile` varchar(40) NOT NULL COMMENT '手机号码',
  `username` varchar(255) NOT NULL COMMENT '用户名称',
  `password` varchar(255) DEFAULT NULL COMMENT '密码',
  `enable_state` int(2) DEFAULT '1' COMMENT '启用状态 0是禁用，1是启用',
  `create_time` datetime DEFAULT NULL COMMENT '创建时间',
  `department_id` varchar(40) DEFAULT NULL COMMENT '部门ID',
  `time_of_entry` datetime DEFAULT NULL COMMENT '入职时间',
  `form_of_employment` int(1) DEFAULT NULL COMMENT '聘用形式',
  `work_number` varchar(20) DEFAULT NULL COMMENT '工号',
  `form_of_management` varchar(8) DEFAULT NULL COMMENT '管理形式',
  `working_city` varchar(16) DEFAULT NULL COMMENT '工作城市',
  `correction_time` datetime DEFAULT NULL COMMENT '转正时间',
  `in_service_status` int(1) DEFAULT NULL COMMENT '在职状态 1.在职  2.离职',
  `company_id` varchar(40) DEFAULT NULL COMMENT '企业ID',
  `company_name` varchar(40) DEFAULT NULL,
  `department_name` varchar(40) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `idx_user_phone` (`mobile`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
```

4.2 配置系统微服务

（1）搭建系统微服务模块（ihrm_system），pom引入依赖

```xml
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <dependency>
            <groupId>com.ihrm</groupId>
            <artifactId>ihrm_common</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
```

（2）配置application.yml

```yaml
server:
  port: 9002
spring:
  application:
    name: ihrm-system #指定服务名
  datasource:
    driver-class-name: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/ihrm?useUnicode=true&characterEncoding=utf8
    username: root
    password: 111111
  jpa:
    database: MySQL
    show-sql: true
    open-in-view: true
```

（3）配置启动类

```java
package com.ihrm.system;

import com.ihrm.common.utils.IdWorker;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.annotation.Bean;

@SpringBootApplication(scanBasePackages = "com.ihrm")
@EntityScan("com.ihrm.domain.system")
public class SystemApplication {

    public static void main(String[] args) {
        SpringApplication.run(SystemApplication.class, args);
    }

    @Bean
    public IdWorker idWorkker() {
        return new IdWorker(1, 1);
    }
}
```

4.3 后端用户基本操作

（1）实体类

```java
package com.ihrm.domain.system;

```

```java
import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;
import java.io.Serializable;
import java.util.Date;
import java.util.HashSet;
import java.util.Set;

/**
 * 用户实体类
 */
@Entity
@Table(name = "bs_user")
@Getter
@Setter
public class User implements Serializable {
    private static final long serialVersionUID = 4297464181093070302L;
    /**
     * ID
     */
    @Id
    private String id;
    /**
     * 手机号码
     */
    private String mobile;
    /**
     * 用户名称
     */
    private String username;
    /**
     * 密码
     */
    private String password;

    /**
     * 启用状态 0为禁用 1为启用
     */
    private Integer enableState;
    /**
     * 创建时间
     */
    private Date createTime;

    private String companyId;

    private String companyName;

    /**
     * 部门ID
     */
    private String departmentId;

    /**
     * 入职时间
     */
    private Date timeOfEntry;

    /**
     * 聘用形式
     */
    private Integer formOfEmployment;

    /**
     * 工号
     */
    private String workNumber;

    /**
     * 管理形式
     */
    private String formOfManagement;

    /**
     * 工作城市
     */
    private String workingCity;

    /**
     * 转正时间
     */
    private Date correctionTime;

    /**
     * 在职状态 1.在职  2.离职
     */
    private Integer inServiceStatus;

    private String departmentName;


    @ManyToMany
    @JsonIgnore
    @JoinTable(name="pe_user_role",joinColumns=
{@JoinColumn(name="user_id",referencedColumnName="id")},
            inverseJoinColumns={@JoinColumn(name="role_id",referencedColumnName="id")}
    )
    private Set<Role> roles = new HashSet<Role>();//用户与角色   多对多
}
```

（2）持久化层

```
package com.ihrm.system.dao;


import com.ihrm.system.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;

/**
  * 企业数据访问接口
  */
public interface UserDao extends JpaRepository<User, String>,
JpaSpecificationExecutor<User> {

}
```

（3）业务逻辑层

```
package com.ihrm.system.service;

import com.ihrm.common.utils.IdWorker;
import com.ihrm.domain.system.Role;
import com.ihrm.domain.system.User;
import com.ihrm.system.dao.RoleDao;
import com.ihrm.system.dao.UserDao;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Predicate;
import javax.persistence.criteria.Root;
import java.util.*;

/**
 * 部门操作业务逻辑层
 */
@Service
public class UserService {
    @Autowired
    private IdWorker idWorker;
    @Autowired
    private UserDao userDao;

    @Autowired
    private RoleDao roleDao;

    public User findByMobileAndPassword(String mobile, String password) {
        User user = userDao.findByMobile(mobile);
        if (user != null && password.equals(user.getPassword())) {
            return user;
        } else {
            return null;
        }
    }

    /**
     * 添加用户
     */
    public void save(User user) {
        //填充其他参数
        user.setId(idWorker.nextId() + "");
        user.setCreateTime(new Date()); //创建时间
        user.setPassword("123456");//设置默认登录密码
        user.setEnableState(1);//状态
        userDao.save(user);
    }

    /**
     * 更新用户
     */
    public void update(User user) {
        User targer = userDao.getOne(user.getId());
        targer.setPassword(user.getPassword());
        targer.setUsername(user.getUsername());
        targer.setMobile(user.getMobile());
        targer.setDepartmentId(user.getDepartmentId());
        targer.setDepartmentName(user.getDepartmentName());
        userDao.save(targer);
    }

    /**
     * 根据ID查询用户
     */
    public User findById(String id) {
        return userDao.findById(id).get();
    }

    /**
     * 删除部门
     *
     * @param id 部门ID
     */
    public void delete(String id) {
        userDao.deleteById(id);
    }


    public Page<User> findSearch(Map<String,Object> map, int page, int size) {
        return userDao.findAll(createSpecification(map), PageRequest.of(page-1, size));
    }
    /**
     * 调整部门
     */
    public void changeDept(String deptId,String deptName,List<String> ids) {
        for (String id : ids) {
            User user = userDao.findById(id).get();
            user.setDepartmentName(deptName);
            user.setDepartmentId(deptId);
            userDao.save(user);
        }
    }

    /**
     * 分配角色
     */
    public void assignRoles(String userId,List<String> roleIds) {
        User user = userDao.findById(userId).get();
        Set<Role> roles = new HashSet<>();
        for (String id : roleIds) {
            Role role = roleDao.findById(id).get();
            roles.add(role);
        }
        //设置用户和角色之间的关系
        user.setRoles(roles);
        userDao.save(user);
    }

    /**
     * 动态条件构建
     * @param searchMap
     * @return
     */
    private Specification<User> createSpecification(Map searchMap) {
        return new Specification<User>() {
            @Override
            public Predicate toPredicate(Root<User> root, CriteriaQuery<?> query,
CriteriaBuilder cb) {
                List<Predicate> predicateList = new ArrayList<Predicate>();
                // ID
                if (searchMap.get("id") !=null && !"".equals(searchMap.get("id"))) {
                    predicateList.add(cb.equal(root.get("id").as(String.class),
(String)searchMap.get("id")));
                }
                // 手机号码
                if (searchMap.get("mobile")!=null &&
!"".equals(searchMap.get("mobile"))) {
                    predicateList.add(cb.equal(root.get("mobile").as(String.class),
(String)searchMap.get("mobile")));
                }
                // 用户ID
                if (searchMap.get("departmentId")!=null &&
!"".equals(searchMap.get("departmentId"))) {
 predicateList.add(cb.like(root.get("departmentId").as(String.class),
(String)searchMap.get("departmentId")));
                }
                // 标题
                if (searchMap.get("formOfEmployment")!=null &&
!"".equals(searchMap.get("formOfEmployment"))) {

 predicateList.add(cb.like(root.get("formOfEmployment").as(String.class),
(String)searchMap.get("formOfEmployment")));
                }
                if (searchMap.get("companyId")!=null &&
!"".equals(searchMap.get("companyId"))) {
                    predicateList.add(cb.like(root.get("companyId").as(String.class),
(String)searchMap.get("companyId")));
                }
                if (searchMap.get("hasDept")!=null &&
!"".equals(searchMap.get("hasDept"))) {
                    if("0".equals((String)searchMap.get("hasDept"))) {
                        predicateList.add(cb.isNull(root.get("departmentId")));
                    }else{
                        predicateList.add(cb.isNotNull(root.get("departmentId")));
                    }
                }
                return cb.and( predicateList.toArray(new
Predicate[predicateList.size()]));
            }
        };
    }
}
```

（4）控制器层

```
package com.ihrm.system.controller;

import com.ihrm.common.controller.BaseController;
import com.ihrm.common.entity.PageResult;
import com.ihrm.common.entity.Result;
import com.ihrm.common.entity.ResultCode;
import com.ihrm.domain.system.User;
import com.ihrm.system.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Map;

@RestController
@RequestMapping("/sys")
public class UserController extends BaseController {

    @Autowired
    private UserService userService;

    //保存用户
    @RequestMapping(value = "/user", method = RequestMethod.POST)
    public Result add(@RequestBody User user) throws Exception {
        user.setCompanyId(parseCompanyId());
        user.setCompanyName(parseCompanyName());
        userService.save(user);
        return Result.SUCCESS();
    }

    //更新用户
    @RequestMapping(value = "/user/{id}", method = RequestMethod.PUT)
    public Result update(@PathVariable(name = "id") String id, @RequestBody User user)
throws Exception {
        userService.update(user);
        return Result.SUCCESS();
    }

    //删除用户
    @RequestMapping(value = "/user/{id}", method = RequestMethod.DELETE)
    public Result delete(@PathVariable(name = "id") String id) throws Exception {
        userService.delete(id);
        return Result.SUCCESS();
    }

    /**
     * 根据ID查询用户
     */
    @RequestMapping(value = "/user/{id}", method = RequestMethod.GET)
    public Result findById(@PathVariable(name = "id") String id) throws Exception {
        User user = userService.findById(id);
        return new Result(ResultCode.SUCCESS,user);
    }

    /**
     * 分页查询用户
     */
    @RequestMapping(value = "/user", method = RequestMethod.GET)
    public Result findByPage(int page,int pagesize,@RequestParam Map<String,Object>
map) throws Exception {
        map.put("companyId",parseCompanyId());
        Page<User> searchPage = userService.findSearch(map, page, pagesize);
        PageResult<User> pr = new
PageResult(searchPage.getTotalElements(),searchPage.getContent());
        return new Result(ResultCode.SUCCESS,pr);
    }
}
```

4.4 前端用户基本操作

由于课程时间有限,本着不浪费时间的原则，页面部分的基本功能都是大致相似的。课上我们使用提供的基本模块代码构建模块信息。

4.4.2 配置接口请求路径

在 `config/index.js` 中通过proxyTable配置代理转发的请求后端地址

```
      '/api/sys': {
        target: 'http://localhost:9002/sys',
        changeOrigin: true,
        pathRewrite: {
          '^/api/sys': ''
        }
      },
```

4.4.1 导入员工模块

注册模块

```
import employees from '@/module-employees/' // 员工管理
Vue.use(employees, store)
```

在 `/src/api/base/` 下配置API（user.js）

```
import {createAPI} from '@/utils/request'

export const list = data => createAPI('/sys/user', 'get', data)
export const simple = data => createAPI('/sys/user/simple', 'get', data)
export const add = data => createAPI('/sys/user', 'post', data)
export const update = data => createAPI(`/sys/user/${data.id}`, 'put', data)
export const remove = data => createAPI(`/sys/user/${data.id}`, 'delete', data)
export const detail = data => createAPI(`/sys/user/${data.id}`, 'get', data)
```

4.4.2 用户列表展示

（1）页面代码

```
        <el-table :data="dataList" fit style="width: 100%;" border>
          <el-table-column type="index" :index="1" label="序号" width="150"> </el-table-
column>
          <el-table-column sortable prop="username" label="姓名" width="150"></el-table-
column>
          <el-table-column sortable prop="mobile" label="手机号" width="150"></el-table-
column>
          <el-table-column sortable prop="workNumber" label="工号" width="120"></el-
table-column>
```

```
          <el-table-column sortable prop="formOfEmployment" label="聘用形势"
width="200"></el-table-column>
          <el-table-column sortable prop="departmentName" label="部门" width="200"></el-
table-column>
          <el-table-column sortable prop="timeOfEntry" label="入职时间" width="150">
</el-table-column>
          <el-table-column sortable label="状态" width="120">
            <template slot-scope="scope">
              <el-switch
              v-model="scope.row.accountStatus"
              active-color="#13ce66"
              inactive-color="#ff4949"
               @change="handleStatus(scope.row)">
              </el-switch>
            </template>
          </el-table-column>
          <el-table-column fixed="right" label="操作" align="center" width="220">
            <template slot-scope="scope">
              <router-link :to="{'path':'/employees/details/' + scope.row.id}"
class="el-button el-button--text el-button--small">
                查看
              </router-link>
              <el-button @click="handleDelete(scope.row)" type="text" size="small">删除
</el-button>
            </template>
          </el-table-column>
        </el-table>
        <!-- 分页 -->
        <div class="pagination">
          <PageTool :paginationPage="requestParameters.page"
:paginationPagesize="requestParameters.pagesize" :total="counts"
@pageChange="handleCurrentChange" @pageSizeChange="handleSizeChange">
          </PageTool>
        </div>
```

（2）js构造数据

```
import constantApi from '@/api/constant/employees'
import {list,remove} from "@/api/base/users"
import PageTool from './../../components/page/page-tool'
import employeesAdd from './../components/add'
var _this = null
export default {
  name: 'employeesList',
  components: {
    PageTool,employeesAdd
  },
  data() {
    return {
      employeesAdd: 'employeesAdd',
      baseData: constantApi,
      dataList: [],
      counts: '',
      requestParameters:{
        page: 1,
        pagesize: 10,
      }
    }
  },
  methods: {
    // 业务方法
    doQuery(params) {
        list(this.requestParameters).then(res => {
          this.dataList = res.data.data.rows
          this.counts = res.data.data.total
        })
    }
  },
  // 创建完毕状态
  created: function() {
    this.doQuery()
  },
}
```

4.4.4 用户详情

（1）配置路由

```
        {
          path: 'details/:id',
          component: _import('employees/pages/employees-details'),
          // hidden: true // 是否显示在左侧菜单
          name: 'details',
          meta: {
            title: '详情'
          }
        }
```

（2）完成用户详情页面

```
  <template>
    <div class="dashboard-container">
      <div class="app-container">
        <el-card  :style="{minHeight:boxHeight}">
            <el-tabs v-model="activeName" class="infoPosin">
              <el-tab-pane name="first" class="rInfo">
                <span slot="label">登录账户设置</span>
                <component v-bind:is="accountInfo" :objId='objId' ref="user"></component>
              </el-tab-pane>
              <el-tab-pane name="two" class="rInfo">
                  <span slot="label">个人详情</span>
              </el-tab-pane>
              <el-tab-pane name="third" class="rInfo">
                  <span slot="label">岗位信息</span>
              </el-tab-pane>
          </el-tabs>
        </el-card>
      </div>
    </div>
  </template>

  <script>
  import accountInfo from './../components/details-account-info'
  export default {
    name: 'employeesDetails',
    components: { accountInfo},
    data() {
      return {
        accountInfo:'accountInfo',
        activeName: 'first',
        objId: this.$route.params.id,
        dataList: []
      }
    }
  }
  </script>
```

（3）用户信息组件

```
<template>
  <div class="boxInfo">
    <!-- 表单内容 -->
    <div class="formInfo">
      <div>
        <!-- 头部信息  -->
        <div class="userInfo">
             <div class="headInfo clearfix">
               <div class="headText">
                 <el-form ref="formData" :model="formData" label-width="215px">
                     <el-form-item label="姓名：">
                       <el-input v-model="formData.username" placeholder="请输入"
class="inputW"></el-input>
                     </el-form-item>
                     <el-form-item label="密码：">
                       <el-input v-model="formData.password" placeholder="请输入"
class="inputW"></el-input>
                    </el-form-item>
                    <el-form-item label="部门：">
                        <el-input
                          placeholder="请选择"
                          v-model="formData.departmentName"
                          icon="caret-bottom"
                          class="inputW"
                          @click.native="isShowSelect = !isShowSelect">
                        </el-input>
                        <input v-model="formData.departmentId" type="hidden" >
                        <el-tree v-if="isShowSelect"
                          :expand-on-click-node="false"
                          :data="inspectionObjectOptions"
                          :props="{label:'name'}"
                          default-expand-all
                          :filter-node-method="filterNode"
                           @node-click="handleNodeClick"
                          class="objectTree"
                          ref="tree2">
                        </el-tree>
                    </el-form-item>
                      <el-form-item>
                        <el-button type="primary" @click="saveData">更新</el-button>
                        <router-link :to="{'path':'/employees/index'}" class="el-button
el-button--text el-button--small">
                          取消
                        </router-link>
                      </el-form-item>
                 </el-form>
               </div>
             </div>
        </div>
      </div>
    </div>
    </div>
</template>

<script>
import constantApi from '@/api/constant/employees'
import {detail,update} from "@/api/base/users"
import { organList } from '@/api/base/departments'
export default {
  name: 'accountInfo',
  props: ['objId'],
  data() {
    return {
      baseData: constantApi,
      inspectionObjectOptions: [],
      isShowSelect:false,
      formData: {
        id: this.objId,
      }
    }
  },
  methods: {
    handleNodeClick(data) {
      this.formData.departmentName = data.name
      this.formData.departmentId = data.id
      this.isShowSelect = false
    },
    // 获取详情
    getObjInfo() {
      detail({ id: this.objId }).then(res => {
          this.formData = res.data.data
      })
    },
    saveData(obj) {
      update(this.formData)
        .then(res => {
          this.formData = res.data
          this.$message.success('保存成功！')
          this.getObjInfo()
      })
    },
  },
  // 创建完毕状态
  created: function() {
    this.getObjInfo()
    organList().then(ret => {
      this.inspectionObjectOptions.push(ret.data.data)
    })
  }
}
</script>
```

4.4.3 用户的新增

和组织机构的增删改查大同小异，学员可以参照代码自行实现

5 作业-角色管理

5.1 需求分析

完成角色的基本CRUD操作

5.2 后端实现

（1）实体类

```
package com.ihrm.domain.system;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;
import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;

@Entity
@Table(name = "pe_role")
@Getter
@Setter
public class Role implements Serializable {
    private static final long serialVersionUID = 594829320797158219L;
    @Id
    private String id;
    /**
     * 角色名
     */
    private String name;
    /**
     * 说明
     */
    private String description;
    /**
     * 企业id
     */
    private String companyId;

    @JsonIgnore
    @ManyToMany(mappedBy="roles")
    private Set<User> users = new HashSet<User>(0);//角色与用户   多对多


    @JsonIgnore
    @ManyToMany
    @JoinTable(name="pe_role_permission",
            joinColumns={@JoinColumn(name="role_id",referencedColumnName="id")},
            inverseJoinColumns=
{@JoinColumn(name="permission_id",referencedColumnName="id")})
    private Set<Permission> permissions = new HashSet<Permission>(0);//角色与模块  多对多
}
```

（2）持久化层

```
package com.ihrm.system.dao;

import com.ihrm.system.domain.Role;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.JpaSpecificationExecutor;

/**
  * 企业数据访问接口
  */
public interface RoleDao extends JpaRepository<Role, String>,
JpaSpecificationExecutor<Role> {

}
```

（3）业务逻辑层

```
package com.ihrm.system.service;

import com.ihrm.common.utils.IdWorker;
import com.ihrm.system.dao.CompanyDao;
import com.ihrm.system.dao.RoleDao;
import com.ihrm.system.dao.UserDao;
import com.ihrm.system.domain.Company;
import com.ihrm.system.domain.Role;
import com.ihrm.system.domain.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Service;

import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Predicate;
import javax.persistence.criteria.Root;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * 角色操作业务逻辑层
 */
@Service
public class RoleService {

    @Autowired
    private IdWorker idWorker;
    @Autowired
    private RoleDao roleDao;


    /**
     * 添加角色
     */
    public void save(Role role) {
        //填充其他参数
        role.setId(idWorker.nextId() + "");
        roleDao.save(role);
    }

    /**
     * 更新角色
     */
    public void update(Role role) {
        Role targer = roleDao.getOne(role.getId());
        targer.setDescription(role.getDescription());
        targer.setName(role.getName());
        roleDao.save(targer);
    }

    /**
     * 根据ID查询角色
     */
    public Role findById(String id) {
        return roleDao.findById(id).get();
    }

    /**
     * 删除角色
     */
    public void delete(String id) {
        roleDao.deleteById(id);
    }

    public Page<Role> findSearch(String companyId, int page, int size) {
        Specification<Role> specification = new Specification<Role>() {
            @Override
            public Predicate toPredicate(Root<Role> root, CriteriaQuery<?> query,
CriteriaBuilder cb) {
                return cb.equal(root.get("companyId").as(String.class),companyId);
            }
        };
        return roleDao.findAll(specification, PageRequest.of(page-1, size));
    }

}
```

（4）控制器层

```
package com.ihrm.system.controller;

import com.ihrm.common.entity.PageResult;
import com.ihrm.common.entity.Result;
import com.ihrm.common.entity.ResultCode;
import com.ihrm.system.domain.Role;
import com.ihrm.system.domain.User;
import com.ihrm.system.service.RoleService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.web.bind.annotation.*;

import javax.persistence.criteria.CriteriaBuilder;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Predicate;
import javax.persistence.criteria.Root;
import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/sys")
public class RoleController {

    @Autowired
    private RoleService roleService;

    //添加角色
    @RequestMapping(value = "/role", method = RequestMethod.POST)
    public Result add(@RequestBody Role role) throws Exception {
        String companyId = "1";
        role.setCompanyId(companyId);
        roleService.save(role);
        return Result.SUCCESS();
    }

    //更新角色
    @RequestMapping(value = "/role/{id}", method = RequestMethod.PUT)
    public Result update(@PathVariable(name = "id") String id, @RequestBody Role role)
throws Exception {
        roleService.update(role);
        return Result.SUCCESS();
    }

    //删除角色
    @RequestMapping(value = "/role/{id}", method = RequestMethod.DELETE)
    public Result delete(@PathVariable(name = "id") String id) throws Exception {
        roleService.delete(id);
        return Result.SUCCESS();
    }

    /**
     * 根据ID获取角色信息
     */
    @RequestMapping(value = "/role/{id}", method = RequestMethod.GET)
    public Result findById(@PathVariable(name = "id") String id) throws Exception {
        Role role = roleService.findById(id);
        return new Result(ResultCode.SUCCESS,role);
    }

    /**
     * 分页查询角色
     */
    @RequestMapping(value = "/role", method = RequestMethod.GET)
    public Result findByPage(int page,int pagesize,Role role) throws Exception {
        String companyId = "1";
        Page<Role> searchPage = roleService.findSearch(companyId, page, pagesize);
        PageResult<Role> pr = new
PageResult(searchPage.getTotalElements(),searchPage.getContent());
        return new Result(ResultCode.SUCCESS,pr);
    }
}
```

5.3 前端实现

学员参考资料自行实现