SSM权限管理项目流程

准备工作

- 1.创建数据库,添加表格
- 2.创建项目,添加依赖
- 3.配置资源文件

```
applicationContext.xml
db.properties
generatorConfig.xml
log4j.properties
mvc.xml
```

- 4.配置main/w ebapp/WEB-INF/w eb.xml

```
springDispatcherServlet
characterEncodingFilter
```

Department CRUD

- 1.配置generatorConfig.xml,生成domain/mapper接口/mapper映射文件
- 2.生成Service接口及其实现类
- 3.生成测试类测试CRUD
- 4.分页查询

```
编写QueryObject/PageResult类
Service接口添加query方法
实现类实现方法
mapper接口添加queryForCount/queryForList方法
映射文件编写sql
测试
```

- 5.Controller

```
list
input
saveOrupdate
delete
```

- 6.w eb页面

```
webapp
css
js
WEB-INF
views
common
department
input.jsp
list.jsp
```

- 7.匹配对应的变量名/测试

Role CRUD

(这里就很爽了,一顿复制粘贴,咳咳咳)

- 1.修改generatorConfig.xml的表名/类名
- 2.生成domain/mapper接口/mapper.xml
- 3.mapper接口/mapper.xml添加分页方法
- 4.拷贝/修改Service接口/实现类/Controller
- 5.页面调整

Employee CRUD

同上

Employee 高级查询

```
传入一个关键词
对多个列进行查询
先在domain的getter判断是否为空字符串
sql只要判断是否为null
and(....or....)
```

Employee x Department 多对一

- 1.查询

```
domain中deptId改为dept对象
xml中
sql改为多表查询
resultType->resultMap
resultMap追加association封装dept
注意别名,前缀,javaType
```

- 2.新增或修改

```
主体employee维护关系,外键列dept_id
对应的属性是dept.id
注意别名
update不用更新password
```

- 3.删除

```
根据employe的id删除,无特别修改
```

Employee x Role 多对多

- 1.查询

```
domain中添加List<Role> roles;
xml中:
ResultMap 中追加Collection封装roles
发送额外sql
查中间表/role表 注意条件不要写错
```

- 2.新增或修改

```
employee插入数据
同时中间表也插入关系(两方id)
mapper接口新增关系插入/删除方法
修改Service实现类saveOrUpdate方法
更新要先删除关系
最后都要添加关系
遍历前,注意判断roleIds是否为空
```

- 3.删除

```
删除时,要删除对应的关系
```

select 的value 就是选中的 option的value

值改变事件

- 修改超级管理员复选框的选中状态时
- 删除或者显示下面的角色的页面片段

```
var role;
$("#admin").change(function(){
  if(this.checked){
    //如果为true,执行删除;
    //detach保留元素的事件
    role = $("#role").detach();
  }else{
    //如果为false,执行显示
    //closest:找最近的匹配的上级元素
    $(this).closest(".form-group").after(role);
  }
  })

  //当编辑的时候
  //页面加载完成之后,如果当前员工是超级管理员,将下面的角色片段删除
  <c:if test="${!empty e}">
  if(${e.admin}){
        role = $("#role").detach();

  }
  </c:if>
```

select框

```
<div class="form-group " id="role">
    <label for="role" class="col-sm-2 control-label">分配权限: </label><br/>
    <div class="row" style="...">
        <div class="col-sm-2 col-sm-offset-2">
            <select multiple class="form-control allPermissions" size="15">
                <c:forEach items="${permissions}" var="p">
                    <option value="${p.id}">${p.name}</option>
                </c:forEach>
            </select>
        </div>

        <div class="col-sm-1" style="..." align="center">
            <div>

                <a type="button" class="btn btn-primary  " style="..." title="右移动"
                   onclick="moveSelected('allPermissions', 'selfPermissions')">
                    <span class="glyphicon glyphicon-menu-right"></span>
                </a>
            </div>
            <div>
                <a type="button" class="btn btn-primary " style="..." title="左移动"
                   onclick="moveSelected('selfPermissions', 'allPermissions')">
                    <span class="glyphicon glyphicon-menu-left"></span>
                </a>
            </div>
            <div>
                <a type="button" class="btn btn-primary " style="..." title="全右移动"
                   onclick="moveAll('allPermissions', 'selfPermissions')">
                    <span class="glyphicon glyphicon-forward"></span>
                </a>
            </div>
            <div>
                <a type="button" class="btn btn-primary " style="..." title="全左移动"
                   onclick="moveAll('selfPermissions', 'allPermissions')">
                    <span class="glyphicon glyphicon-backward"></span>
                </a>
            </div>
        </div>

        <div class="col-sm-2">
            <select multiple class="form-control selfPermissions" size="15" name="permissionIds">
                <c:forEach items="${entity.permissions}" var="p">
                    <option value="${p.id}">${p.name}</option>
                </c:forEach>
            </select>
        </div>
    </div>
</div>
```

```
<script>
        $(function () {
            //获取用户拥有的权限ids,封装到ids
            var ids = $.map($(".selfPermissions option"), function (item) {
                //console.log(item.value);
                return item.value;
            })

            //遍历全部权限,将与用户重复的权限去除
            $.each($(".allPermissions option"), function (index, item) {
                if ($.inArray(item.value, ids) >= 0) {
                    $(item).remove();
                }
            });

            //提交表单
            $("#submitBtn").click(function () {
                //提交表单之前,先将右边框选中
                $(".selfPermissions option").prop("selected", true);
                $("#editForm").submit();
            })
        })

        //移动选中
        function moveSelected(srcCls, targetCls) {
            $("." + srcCls + " option:selected").appendTo($("." + targetCls));
        }
```

```
        //移动全部
        function moveAll(srcCls, targetCls) {
            $("." + srcCls + " option").appendTo($("." + targetCls))
        }
</script>
```

权限管理模块

```
访问对应的资源需要对应的权限
权限分配给角色:
权限表:id,name,
expression(权限表达式唯一)
中间表:多对多关系

1.生成权限的数据:
程序员来生成权限数据(name/expression)
@requiredPermission(name="员工列表",expression="employee:list")
public String list(){}
@requiredPermission(name="员工编辑",expression="employee:list")
public String input(){}

2.扫描所有controller中的所有方法,获取到对应的权限注解中的数据,保存到数据库
3.生成权限数据的时间(监听器)
  1.服务器启动时候
  2.用户控制生成的时间

//定义注解:
//指定能贴的位置
@Target(ElementType.METHOD)
//指定保存的时期(源码/字节码/运行时期)
@Retension(RetentionPolicy.RUNTIME)
public @interface RequiredPermission{
  String[] value():
}

@RequiredPermission({"员工列表","employee:list"})
public String list(){}
```

```
重载权限:reload
获取当前数据库中的权限,并将其权限表达式取出到封装到List<> expressions中
通过容器ApplicationContext ->controllers ->methods ->annotation >expression
判断expression是否在expressions,如果不在就添加到数据库中
不能直接删除全部然后加载,因为会同时清除之前配置的关系
```

```
关系管理:
新增role时,要插入关系到role_permission
删除role/permission时,要删除role_permission里面的关系
```

```
登陆:
前台传入name和password到Controller
调用业务方法查找,
不为null:共享EMPLOYEE_IN_SESSION和EXPRESSIONS_IN_SESSION
若为null:抛出异常.

若controller
没捕获到异常,跳转到主页;
若捕获到异常,共享异常信息,返回登陆页面
```

```
注销:
```

清空session中的对象,返回登陆页面
session.invalidate();

登陆拦截器
不允许用户未经登陆,直接访问指定资源,返回登陆页面
2.覆写preHandle方法
1.继承HandlerInterceptorAdapter
3.判断Session中是否有EMPLOYEE_IN_SESSION
4.有,放行,跳转到指定页面
  无,拦截,跳到登陆页面

权限检查拦截器
用户需要有对应的权限才能访问对应的资源
1.继承HandlerInterceptorAdapter
2.覆写preHandle方法
3.获取访问当前资源所需的权限表达式
  ((HandlerMethod) handler).getMethod();
  method.getAnnotation(...).value()[1];
4.判断当前用户是否拥有相应的权限
5.有,放行,跳转到指定资源;
  无,拦截,跳转到"权限不足"页面;