国际物流云商系统第五天

一. 回顾

1.Shiro 安全框架

认证，授权，加密，会话管理（单点登录）,缓存，与 web 集成

2.Shiro 配置

- 应用程序与 Shiro 交互的运行过程
  ApplicationCode ----------→Subject----------→SecurityManager----------→Realm 域

- 配置
  1.jar 包或 Shiro 的 maven 坐标的引入
  2.web.xml 配置过滤器
  3.auto-proxy 代理生成方式，cglib

```
<!--Shiro生成代理时，使用cglib方式来生成,该代码最好放在事务管理器声明之前-->
<aop:aspectj-autoproxy proxy-target-class="true" />
```

  4.shiro 配置文件
  1.密码比较器配置
  2.自定义 realm 域的配置
  3.shiroFilter 的代理子类的配置（web.xml 过滤器名称一致）
  4.安全管理器的配置

  5.编写密码比较器(Md5Hash)
  就是在 Md5 加密算法的基础上进行改良，加上盐和打乱次数。
  明文----------密文
  6.编写自定义的 Realm 域
  1.继承 AuthorizingRealm
  2.重写认证方法，授权方法

  8.总结 Shiro 执行流程

二．角色分配

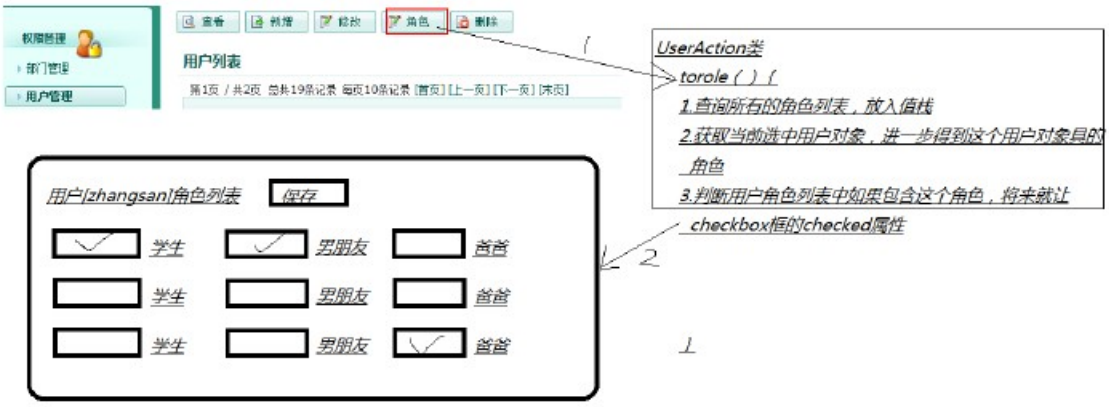
1.左侧菜单的修改

每个模块的 **left.jsp** 页面进行修改

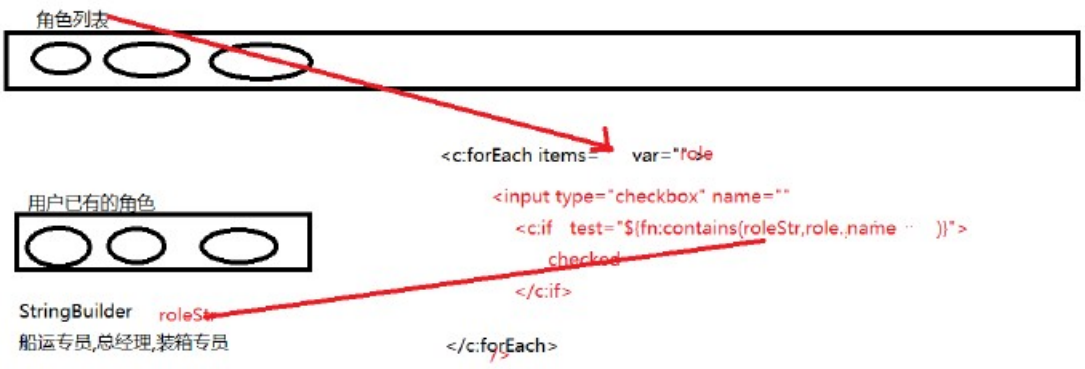
```
<!-- 引入动态的菜单生成 -->
<%@include file="../leftmenu.jsp" %>
```

2.进入角色分配页面的 action

思路分析：



3.JSP 页面中处理复选框的思路



EL 函数的使用：

1. 导入

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
```

2. 使用

解决页面的样式问题：

```
<style type="text/css">
  span{display: inline-block;width: 200px;}
</style>
```

4.进入角色修改界面

```
@Action(value = "userAction_torole", results = {
        @Result(name = "torole", location = "/WEB-INF/pages/sysadmin/user/jUserRole.jsp") })
public String torole() throws Exception {
    // 1.调用业务方法，得到当前用户对象及他的角色列表
    User user = userService.get(model.getId());

    // 加载当前用户的角色列表
    Set<Role> roles = user.getRoles();
    // 2.将当前用户的角色列表处理成字符串形式
    StringBuilder sb = new StringBuilder();
    for (Role role : roles) {
        sb.append(role.getName()).append(",");
    }
    // 将用户信息及角色信息保存到值栈中
    super.push(user);
    super.put("roleStr", sb.toString());

    // 3.得到所有的角色列表
    List<Role> roleList = roleService.find(null);

    // 将角色列表保存到值栈中
    super.put("roleList", roleList);
    // 4.跳页面
    return "torole";
}
```

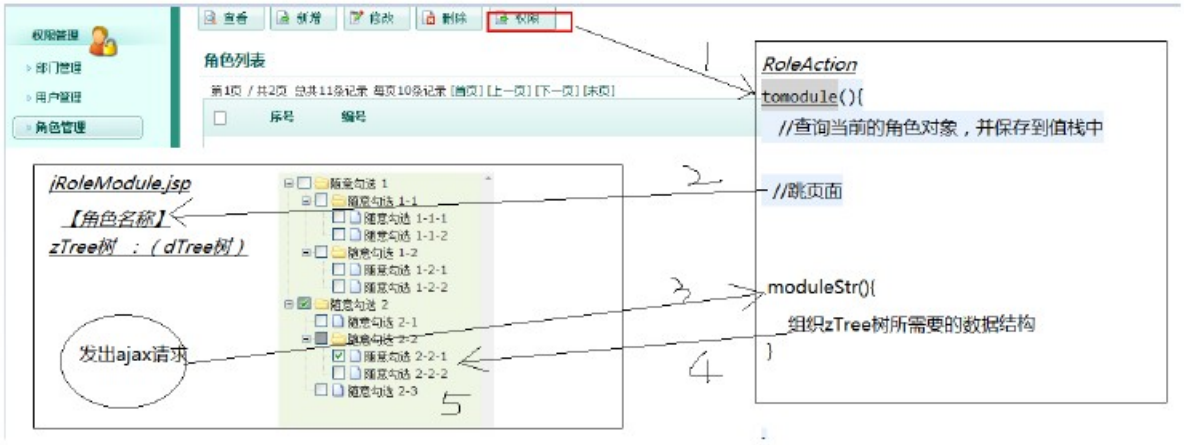
5.实现角色保存

当存在多对多关系时，操作其中一方，就可以自动维护中间表的关系。这样就可以不用配置 cascade="all" ,大多数情况不配置 cascade 属性

```
@Action(value="userAction_role")
public String role() throws Exception{
    //1.得到当前的用户对象
    User obj = userService.get(model.getId());

    //2.将用户选择的角色加载出来
    HashSet<Role> roleSet = new HashSet<>();
    for (String id : roleIds) {
        Role role = roleService.get(id);
        roleSet.add(role);
    }

    //3.设置关系
    obj.setRoles(roleSet);

    //3.更新用户的角色列表
    userService.saveOrUpdate(obj);

    return "alist";
}
private String[] roleIds;
public void setRoleIds(String[] roleIds) {
```

三．模块分配（权限分配）

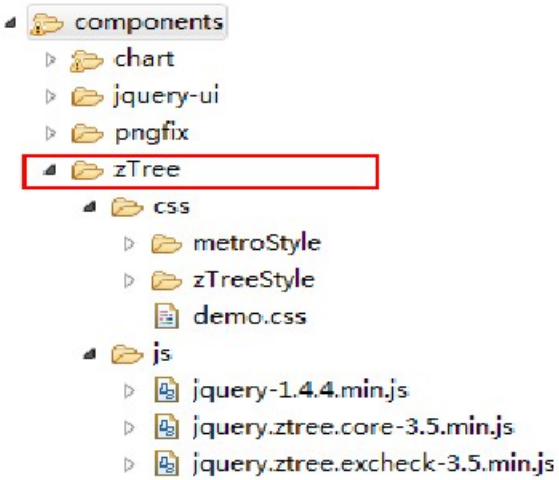
1.模块分配的思路分析



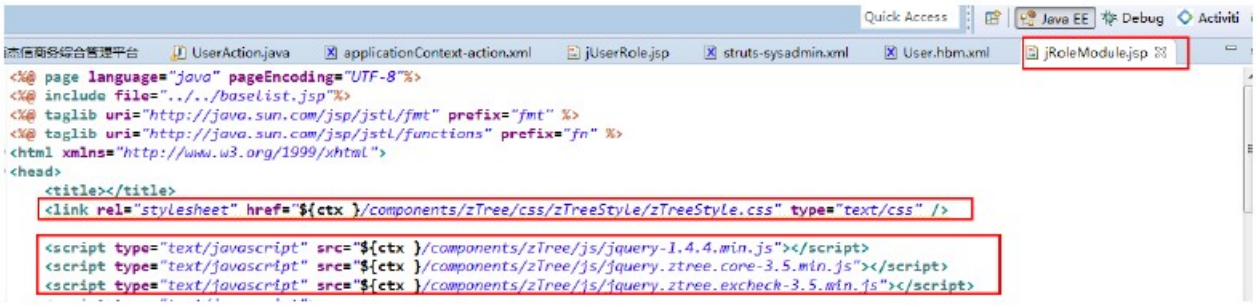
2.引入相关的 **js** 文件到项目中

1. 引入 zTree 树

2. zTree 树在项目中位置



3.在 jsp 页面上引入这些 js 文件及 css 样式

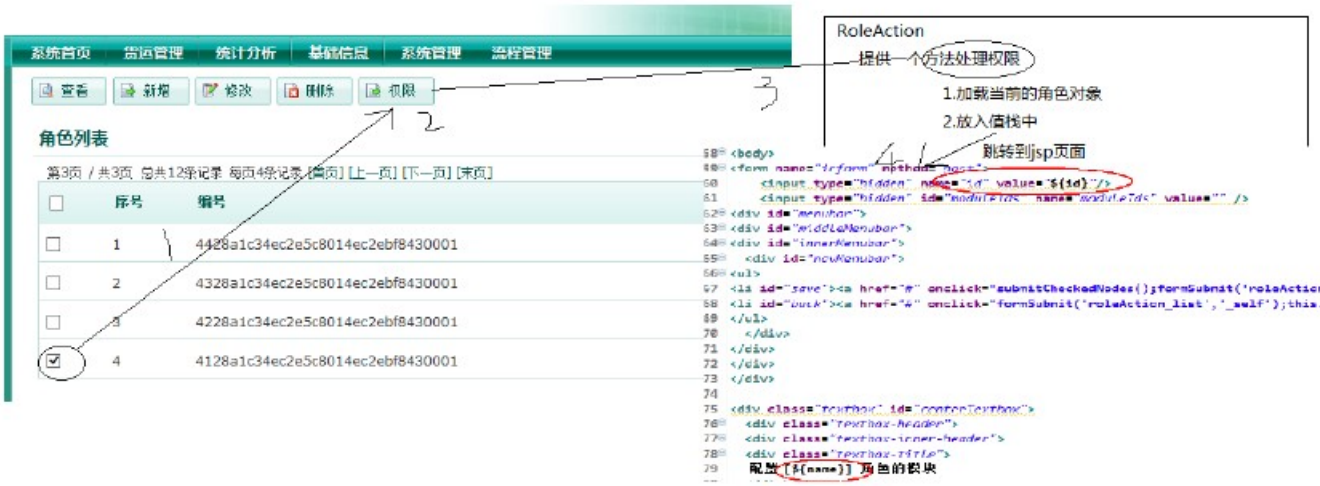


4.编写 js 实现动态生成一颗树（数据来源是一个 json 串）

```
<script type="text/javascript">
        var zTreeObj;
        var setting = {
            check : {
                enable : true
            },
            data : {
                simpleData : {
                    enable : true
                }
            }
        };

        $(document).ready(function() {
            $.ajax( {
            url:    "${ctx}/sysadmin/roleAction_roleModuleJsonStr.action?id=${id}",
                type : "get",
                dataType : "text",
                success : initZtree
            });
        });

        //初始化 ZTree 树
        function initZtree(data) {
            var zNodes = eval("(" + data + ")");       //动态 js 语句
            zTreeObj = $.fn.zTree.init($('#jkTree'), setting, zNodes);      //jkTree    树
的 id，支持多个树
            zTreeObj.expandAll(true);          //展开所有树节点
        }
```

```
        //获取所有选择的节点
        function submitCheckedNodes(treeNode) {
            var nodes = new Array();
            nodes = zTreeObj.getCheckedNodes(true);          //取得选中的结点
            var str = "";
            for (i = 0; i < nodes.length; i++) {
                if (str != "") {
                    str += ",";
                }
                str += nodes[i].id;
            }
            $('#moduleIds').val(str);
        }
    </script>
```

5.分析 RoleAction 中的方法



6.RoleAction 中添加的方法

```
@Action(value = "roleAction_tomodule", results = {
        @Result(name = "tomodule", location = "/WEB-INF/pages/sysadmin/role/jRoleModule.jsp") })
public String tomodule() throws Exception {
    //1.加载角色对象
    Role role = roleService.get(model.getId());
    //2.将角色对象放入值栈中
    super.push(role);
    return "tomodule";
}
```

7.RoleAction 中用于处理 ajax 请求的方法

客户端发送 Ajax 请求

```
$(document).ready(function(){
    $.ajax( {
        url : "${ctx}/sysadmin/roleAction_roleModuleJsonStr.action?id=${id}",
        type : "get",
        dataType : "text",
        success : initZtree
    });
});
```

当窗体加载完成后就会执行内部的function

RoleAction

新加一个方法：roleModuleJsonStr(),它返回一个json字符串，返回的json字符串的数据结构：

[{},{}]

一个{}内部就包含了一个树结点的完整信息

作为一个完整结点信息，包含哪些属性：

{id:"",pId:"",name:"",checked:"true|false"}

id代表结点编号，pId代表父结点编号

name代表结点名，checked代表当前结点是否选中

服务端响应的处理方法

```
/***
 * 实现Ajax请求的处理
 * [{"id":"","pId":"","name":"","checked":""}]
 */
@Action(value="roleAction_genzTreeNodes")
public String genzTreeNodes() throws Exception {
    //1.根据角色编号，获取角色对象
    Role role = roleService.get(model.getId());
    //2.通过role对象加载当前role的模块集合
    Set<Module> moduleSet = role.getModules();

    //加载所有可用的模块列表
    Specification<Module> spec = new Specification<Module>() {
        public Predicate toPredicate(Root<Module> root, CriteriaQuery<?> query, CriteriaBuilder cb) {
            return cb.equal(root.get("state").as(Integer.class),1);
        }
    };
    List<Module> moduleList = moduleService.find(spec);//加载状态为1的模块列表

  //将结果转化为zTree树要求的json串
  StringBuilder sb = new StringBuilder();
  int size=moduleList.size();
  sb.append("[");
  for (Module module : moduleList) {
      size--;
      sb.append("{\"id\":\"").append(module.getId());
      sb.append("\",\"pId\":\"").append(module.getParentId());
      sb.append("\",\"name\":\"").append(module.getName());
      sb.append("\",\"checked\":\"");
      if(moduleSet.contains(module)){
          sb.append("true");
      }else{
          sb.append("false");
      }
      sb.append("\"}");
      if(size>0){
          sb.append(",");
      }
  }
  sb.append("]");
  System.out.println(sb.toString());
```

```
//处理json的响应
HttpServletResponse response = ServletActionContext.getResponse();
response.setContentType("application/json;charset=UTF-8");
response.setHeader("cache-control", "no-cache");
response.getWriter().write(sb.toString());

return NONE;
```

8.客户端接收 json 串并加载到 ztree

zTreeObj = $.fn.zTree.init($('#jkTree'), setting, zNodes);

第一个参数：树展示的位置

第二个参数：一些初始配置

第三个参数：服务器返回的 json 对象

zTreeObj.expandAll(true);　　//展开所有树节点

9 如何获取 zTree 树上被选中的结点？

nodes = zTreeObj.getCheckedNodes(true);　　//取得选中的结点

它可以在表单提交时，把 zTree 树上选中结点组织成提交的数据

```
//获取所有选择的节点
function submitCheckedNodes() {
    var nodes = new Array();
    nodes = zTreeObj.getCheckedNodes(true);      //取得选中的结点
    var str = "";
    for (i = 0; i < nodes.length; i++) {
        if (str != "") {
            str += ",";
        }
        str += nodes[i].id;
    }
    $('#moduleIds').val(str);
}
```

此函数调用：

```
onclick="submitCheckedNodes();formSubmit('roleAction_module','_self');this.blur();">保存</a></li>
```

10.RoleAction 如何保存用户选中的结点。

在 RoleAction 类中添加角色分配的 module()方法

```
/**
 * 实现模块分配
 * <input type="hidden" name="id" value="${id}"/>
   <input type="hidden" id="moduleIds" name="moduleIds" value="" />
 */
@Action("roleAction_module")
public String module() throws Exception {
    //1.加载角色对象
    Role obj = roleService.get(model.getId());
    //2.加载模块列表
    Set<Module> modules = new HashSet<Module>(0);
    String ids[] =  moduleIds.split(",");
    for (String id :ids) {
        modules.add(moduleService.get(id));
    }

    //3.设置角色对象与模块之间的关系
    obj.setModules(modules);

    //4.保存角色对象
    roleService.saveOrUpdate(obj);

    return "alist";
}
private String moduleIds;
public void setModuleIds(String moduleIds) {
    this.moduleIds = moduleIds;
}
```

四．异常处理框架

1.异常处理框架存在的意义

在系统中什么时候实现异常处理是最好的，企业中如何实现异常的处理，是在 DAO，Service,Controller 的哪个部分?
答案是：在 Controller 中结合框架提供的异常处理机制，实现异常的处理是现在比较主流的方式。

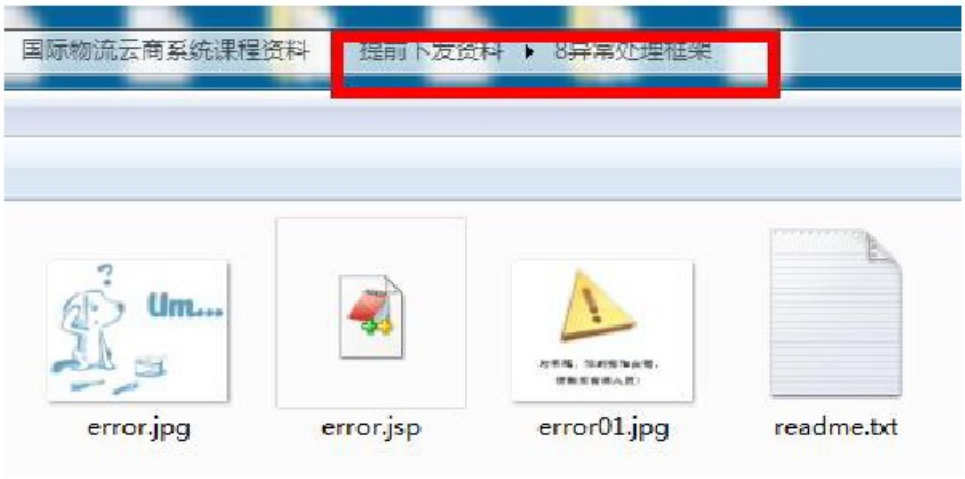
Struts2 提供了异常的处理机制，通过这种机制我们只需要在 struts.xml 配置文件中进行简单配置，就可以实现全站的异常统一处理。

2.使用 Struts2 异常处理框架的步骤

- 自定义相关异常类

```java
/**
 * 自定义的异常处理类
 *
 */
public class SysException extends Exception {
    private String message;
    public String getMessage() {
        return message;
    }
    public SysException(String message) {
        this.message = message;
    }
}
```

- 导入异常处理页面



- 在 struts.xml 文件中配置

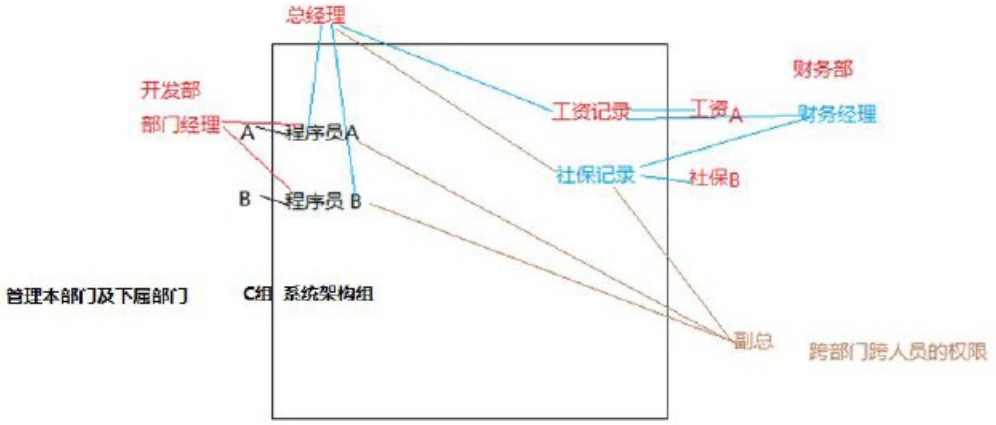
```xml
<!--全局结果视图配置-->
<global-results>
    <result name="error">/WEB-INF/pages/error.jsp</result>
</global-results>
<!--借助于struts2的异常框架来实现异常处理-->
<global-exception-mappings>
    <exception-mapping exception="cn.itcast.jk.exception.SysException" result="error"></exception-mapping>
    <exception-mapping exception="java.lang.Exception" result="error"></exception-mapping>
</global-exception-mappings>
```

五．细粒度权限控制

1．细粒度权限的定义

粗粒度权限控制：只能控制不同角色的用户登录系统时，他们所看到的主菜单，左侧菜单，按钮，超链接是不同的。
在可以控制顶部菜单 ，左侧菜单 ，超链接的基础上，可以进一步控制数据。

2.细粒度分类



1.自己添加的记录自己可以看到
2.部门经理可以看到该部门员工所添加的记录（不包括下属部门）
3.总经理可以看到所有记录

3.细粒度权限的实现思路

细粒度权限的实现思路和过程，需要配合数据库表的设计，再结合程序代码一起来实现。需要大家先思考，后面第七天会具体实现。

六．作业

1.扩展实现三级菜单的管理。
2.思考：将 zTree 树结点的手动 json 串拼接工作，改为使用 FastJson 来实现，并分析利弊。
3.思考：细粒度权限的实现。

七．面试

1.[面试：在 Hibernate 中实现数据检索的 5 种方式]

1.关联级别的数据检索
2.HQL 语句
3.SQL 语句
4.QBC 语句
5.通过 OID 加载　get/load()

2.[面试：get/load]

1.get 是立即加载，load 延迟加载
2.get 加载数据失败时返回 null,而 load 加载数据失败会抛出异常

3.[面试：StringBuffer 与 StringBuilder 区别]

1.StringBuilder 效率高，存在线程安全问题
2.StringBuffer 不存在线程安全问题，效率低