第一章 Shiro 框架（一）

课程目标

1、 权限管理原理

2、 权限管理解决方案

3、 基于 url 的权限管理

4、 Shiro 简介

课程内容

1、 权限管理原理

1.1）什么是权限管理？

只要有用户参与的系统一般都要有权限管理，权限管理实现对用户访问系统的控制，按照安全规则或者安全策略控制用户可以访问而且只能访问自己被授权的资源。

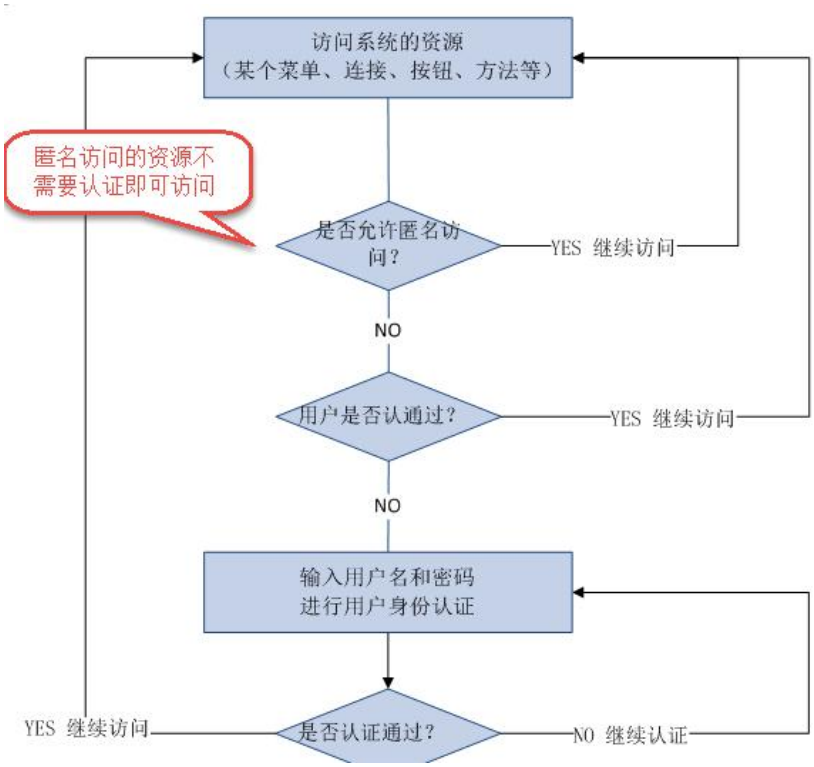
权限管理包括用户认证和授权两部分。

1.2）用户认证：

1.2.1）认证的概念：

用户认证，用户去访问系统，系统要验证用户身份的合法性。最常用的用户身份验证的方法：1、用户名密码方式、2、指纹打卡机、3、基于证书验证方法。。系统验证用户身份合法，用户方可访问系统的资源。

1.2.2）用户认证的流程（原理图）：



1.2.3）了解认证流程中的关键对象：

subject：主体，理解为用户,可能是程序，都要去访问系统的资源，系统需要对 subject 进行身份认证。

principal：身份信息，通常是唯一的，一个主体还有多个身份信息，但是都有一个主身份信息（primary principal）

credential：凭证信息，可以是密码 、证书、指纹。

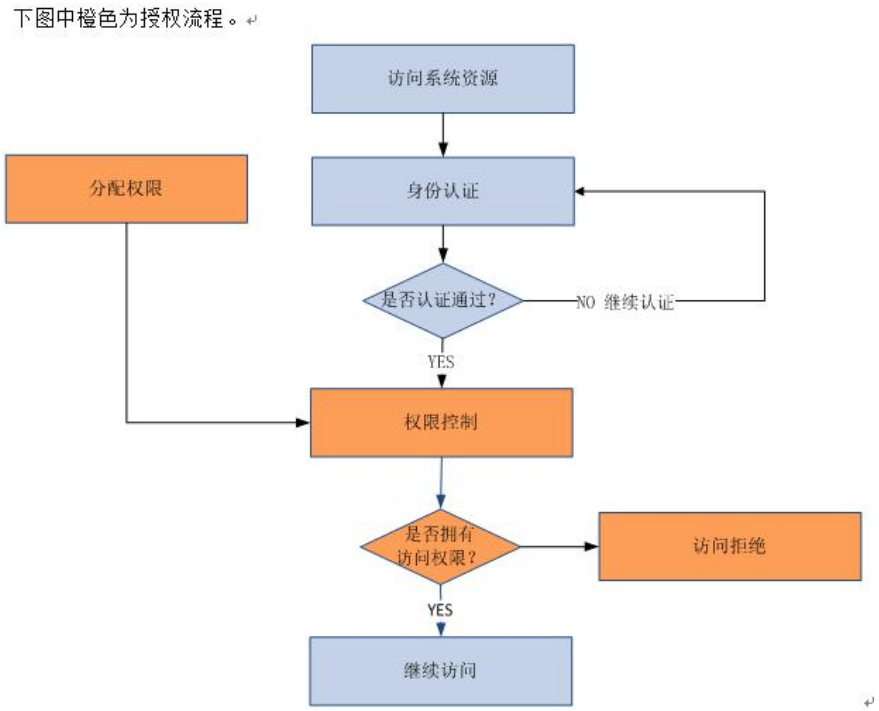
总结：主体在进行身份认证时需要提供身份信息和凭证信息。

1.3）用户授权：

1.3.1）什么是用户授权?

用户授权，简单理解为访问控制，在用户认证通过后，系统对用户访问资源进行控制，用户具有资源的访问权限方可访问。

1.3.2) 用户授权流程图？



1.3.2）用户授权过程中的关键对象？

Subject,resource,permission 分别代表主体，资源，权限。也可以简单理解为：

Who 对 what(which) 有 how 的操作。

what(which)：资源(**Resource**)，subject 必须具备资源的访问权限才可访问该 资源。资源比如：系统用户列表页面、商品修改菜单、商品 id 为 001 的商品信息。

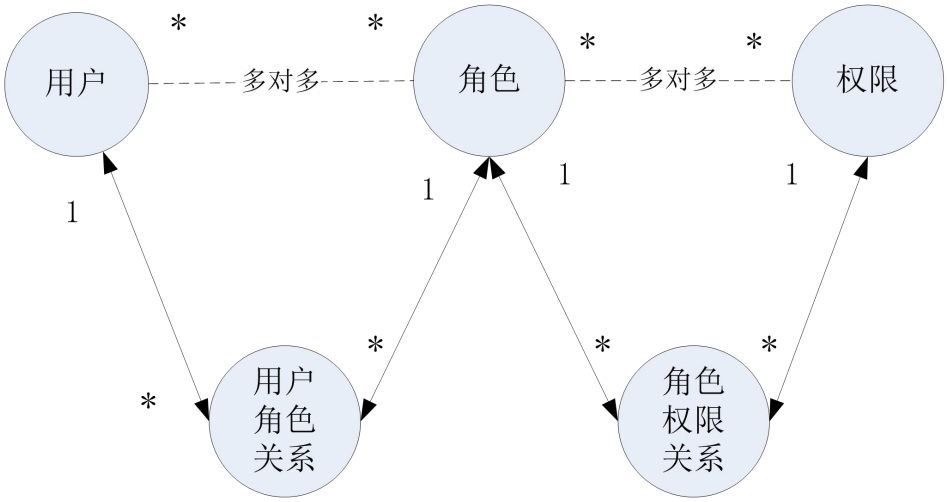
资源分为**资源类型和资源实例**：

系统的用户信息就是资源类型，相当于 java 类。

系统中 id 为 001 的用户就是资源实例，相当于 new 的 java 对象。

how：权限/许可(**permission**) ，针对资源的权限或许可，subject 具有 permission 访问资源，如何访问/操作需要定义 permission，权限比如：用户添加、用户修改、商品删除。

1.3.3）权限管理模型：



上图常被称为权限管理的通用模型，不过企业在开发中根据系统自身的特点还会对上图进行修改，但是用户、角色、权限、用户角色关系、角色权限关系是需要去理解的。

1.3.3）分配权限：

用户需要分配相应的权限才可访问相应的资源。权限是对于资源的操作许可。
通常给用户分配资源权限需要将权限信息持久化，比如存储在关系数据库中。
把用户信息、权限管理、用户分配的权限信息写到数据库（权限数据模型）

1.3.4）权限控制：

1.3.4.1）基于角色的权限控制：

RBAC（Role Based Access Control）:基于角色的权限控制
基于角色的权限管理，比如：某个角色可以访问某些资源，一旦角色发生了变化，就要修改 java 的代码，这样，维护起来不方便，比如：

```
If(user.hasRole(“经理”)){
    //查看报表
    //查看系统资源内容
}
```

此时，如果“总经理”也要能够访问“查看报表”和”查看系统资源”，则伪代码改为：

```
If(user.hasRole(“经理”) || user.hasRole(“总经理”)){
    //查看报表
    //查看系统资源内容
}
```

结论：基于角色的访问控制是不利于系统维护(可扩展性不强)。

1.3.4.2）基于资源的权限控制：

RBAC（Resource Based Access Control）:基于资源的权限控制
基于资源的权限控制与当前用户的权限有关，而与其角色无关，所以，当用户的角色发生变化后，也不用修改相应的 java 代码，如：

```
If(user.hasPermission(“查看报表权限”)){
```

```
    //查看报表及系统资源
}
```

如果，此时，”总经理”的权限发生了变更（可以通过分配权限模块,为其增加”查看报表”权限），我们不用再去修改 java 代码，此时，只需要在分配权限模块添加“总经理”的“查看报表”的权限即可，完全不用修改代码。

结论：一般使用基于资源的权限控制。

2、 权限管理解决方案

2.1）什么是粗粒度和细粒度？

粗粒度权限管理，对资源类型的权限管理。资源类型比如：菜单、url 连接、用户添加页面、用户信息、类方法、页面中按钮。。

粗粒度权限管理比如：超级管理员可以访问户添加页面、用户信息等全部页面。

部门管理员可以访问用户信息页面包括 页面中所有按钮。

细粒度权限管理，对资源实例的权限管理。资源实例就资源类型的具体化，比如：用户 id 为 001 的修改连接，1110 班的用户信息、行政部的员工。

细粒度权限管理就是数据级别的权限管理。

细粒度权限管理比如：部门经理只可以访问本部门的员工信息，用户只可以看到自己的菜单，大区经理只能查看本辖区的销售订单。。

粗粒度和细粒度例子：

系统有一个用户列表查询页面，对用户列表查询分权限，如果粗颗粒管理，张三和李四都有用户列表查询的权限，张三和李四都可以访问用户列表查询。

进一步进行细颗粒管理，张三（行政部）和李四(开发部)只可以查询自己本部门的用户信息。张三只能查看行政部 的用户信息，李四只能查看开发部门的用户信息。**细粒度权限管理就是数据级别的权限管理。**

2.2）如何实现粗粒度与细粒度的权限管理？

2.2.1）如何实现粗粒度权限管理？

粗粒度权限管理比较容易将权限管理的代码抽取出来在系统架构级别统一处理。比如：通过 springmvc 的拦截器实现授权。

2.2.2）如何实现细粒度权限管理？

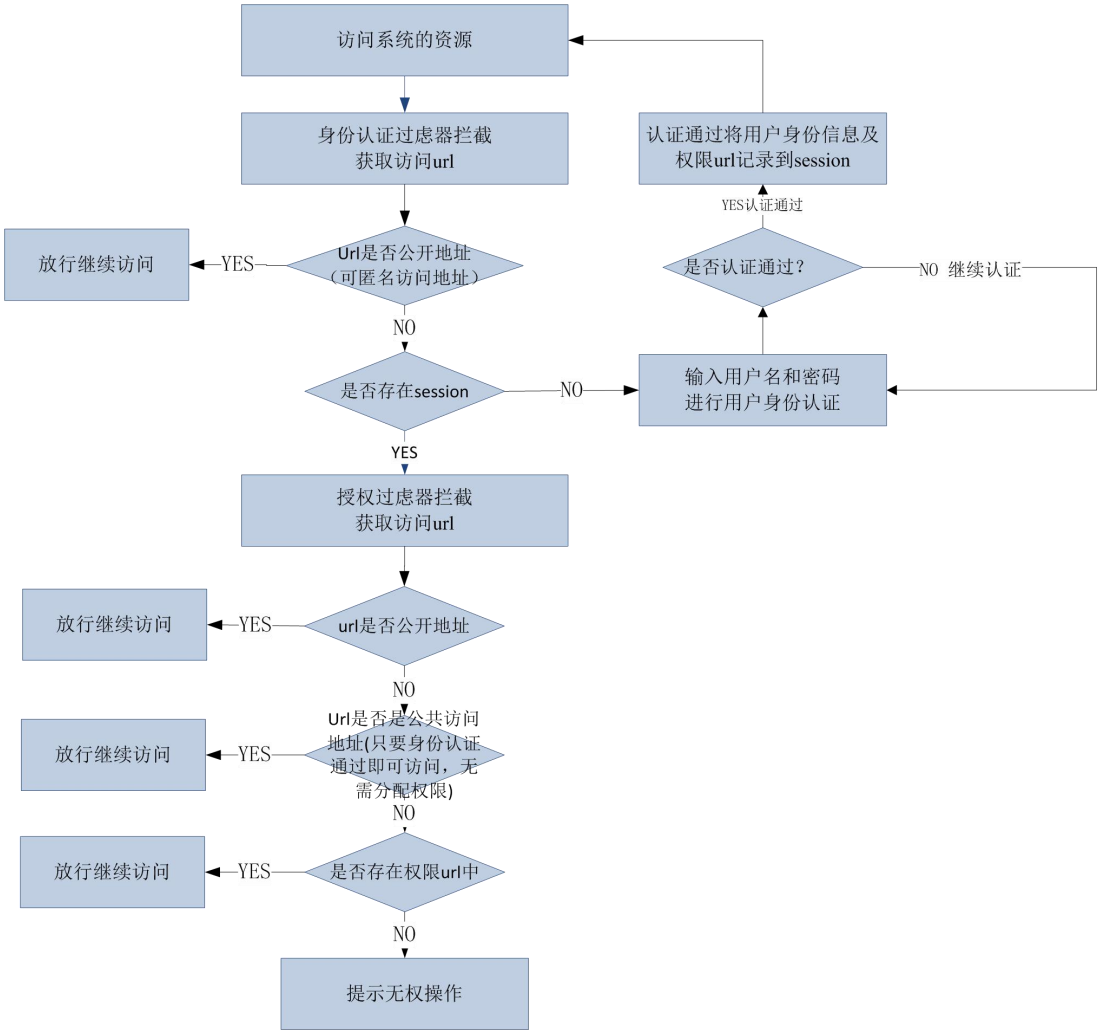
对细粒度权限管理在数据级别是没有共性可言，针对细粒度权限管理就是系统业务逻辑的一部分，如果在业务层去处理相对比较简单，如果将细粒度权限管理统一在系统架构级别去抽取，比较困难，即使抽取的功能可能也存在扩展不强。

建议细粒度权限管理在业务层去控制。
比如：部门经理只查询本部门员工信息，在 service 接口提供一个部门 id 的参数，controller 中根据当前用户的信息得到该用户属于哪个部门，调用 service 时将部门 id 传入 service，实现该用户只查询本部门的员工。

3、 基于 url 的权限管理（重点）

基于 url 拦截的方式实现在实际开发中比较常用的一种方式。
对于 web 系统，通过 **filter 过虑器实现 url 拦截**，也可以 **springmvc 的拦截器**实现基于 url 的拦截。

3.1）基于 url 权限管理的流程：



3.2）使用 url+springmvc 拦截器实现权限控制：

3.2.1）用户登录，登录成功后，把当前用户存放到 session 中：

后台代码：

```
@Controller
@RequestMapping("user")
public class SysUserController {
    @Autowired
    private SysUserService sysUserService;
    @RequestMapping("login")
    public void login(HttpServletResponse response,HttpSession session,
                      String usercode,String password) throws Exception{
        response.setContentType("text/html;charset=utf-8");
        String success = "1";       //代表登录是否成功
        //1.根据用户名、密码查询此用户是否存在
        SysUser sysUser =
sysUserService.findUserByUsercodeAndPassword(usercode,password);
        //2.如果查询到用户存在，就放到 session 中
        if(sysUser != null) {
            session.setAttribute("user", sysUser);
        }
        else {
            success = "0";
        }
        response.getWriter().println(success);
    }
}
```

前台 login.jsp 代码：

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@include file="/WEB-INF/jsp/baseUrl.jsp" %>
<!-- 加载 jquery easyUI 库 -->
<%@include file="/WEB-INF/jsp/loadEasyUI.jsp" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>登录页面</title>
<script>
    $(function(){
        //点击“登录”按钮后进行异步提交表单
```

```
        $('#loginBtn').click(function(){
            $('#form1').form('submit',{
                url:'${baseUrl}/user/login.action',
                onSubmit:function(){
                    return $(this).form("validate");
                },
                success:function(data){
                    if(data == 1){ //登录成功，跳转到首页
                        location.href = "${baseUrl}/index.action"
                    }else{
                        $('#user_error').html("<span style='color:red'>用户名
或密码输入有误！");
                    }
                }
            })
        })
    })
</script>
</head>
<body>

    <div id="win" class="easyui-dialog" title="用户登录"
style="width:400px;height:190px"

data-options="iconCls:'icon-user',buttons:'#buttons',modal:true,collapsible:
false,minimizable:false,maximizable:false,closable:false">
     <form id="form1" method="post">
         <table width=100% cellspacing=15 align=center>
            <tr>
                <td colspan=2 id="user_error"></td>
            </tr>
            <tr>
                <td>用户名：</td>
                <td><input class="easyui-textbox" name="usercode"
data-options="required:true"
                        missingMessage="对不起，用户名不能为空！"></td>
            </tr>
            <tr>
                <td>密码：</td>
                <td><input class="easyui-textbox" name="password"
data-options="type:'password',required:true"
                    missingMessage="对不起，密码不能为空！"></td>
            </tr>
```

```
        </table>
    </form>
</div>
<div id="buttons">
    <a href="#" class="easyui-linkbutton" data-options="iconCls:'icon-user'"
id="loginBtn">登录</a>
    <a href="#" class="easyui-linkbutton"
data-options="iconCls:'icon-cancel'">取消</a>
</div>
</body>
</html>
```

3.2.2）登录 SysUserServiceImpl.java 类编写：

```
@Service
public class SysUserServiceImpl implements SysUserService {
    @Autowired
    private SysUserMapper userMapper;
    @Override
    public SysUser findUserByUsercodeAndPassword(String usercode,String
password)  throws Exception{
        //1.根据用户名查询到用户对象
        SysUser sysUser = userMapper.findUserByUsercode(usercode);
        //1.1）得到当前用户下的所有菜单 集合
        List<SysPermission> menus =
userMapper.getMenusByUsercode(usercode);
        //1.2）得到当前用户下的所有权限 集合
        List<SysPermission> permissions =
userMapper.getPermissionsByUsercode(usercode);
        //1.3) 为当前用户指定上面两个集合
        sysUser.setMenus(menus);
        sysUser.setPermissions(permissions);
        //2.判断用户是否为空，如果不为空，再判断密码
        //将用户输入的密码经过 md5 加密后，再与数据库中的密码进行比对
        MD5 md5 = new MD5();
        password = md5.getMD5ofStr(password); //对用户输入的密码进行 md5 加
密
        if(sysUser != null && sysUser.getPassword().equals(password)){
                return sysUser;
        }
        return null;
    }

}
```

3.2.3）登录 SysUserMapper 及其映射文件编写：

接口：

```
//根据 usercode 查询得到用户
    public SysUser findUserByUsercode(String usercode)  throws
Exception;
    //根据 usercode 查询此用户下的所有菜单
    public List<SysPermission> getMenusByUsercode(String usercode)
throws Exception;
    //根据 usercode 查询此用户下的所有权限
    Public     List<SysPermission>     getPermissionsByUsercode(String
usercode) throws Exception;
```

映射文件：

```
 <select id="findUserByUsercode" parameterType="string"
resultType="sysUser">
    select * from sys_user where usercode = #{value}
  </select>
  <!-- 根据 usercode 查询当前用户下的所有菜单  -->
  <select id="getMenusByUsercode" resultType="SysPermission"
parameterType="string">
    SELECT
    *
    FROM
    sys_permission
    WHERE TYPE = 'menu'
    AND id IN
    (SELECT
    sys_permission_id
    FROM
    sys_role_permission
    WHERE sys_role_id IN
    (SELECT
    sys_role_id
    FROM
    sys_user_role
    WHERE sys_user_id IN
    (SELECT
    id
    FROM
    sys_user
    WHERE usercode = #{value})))
  </select>
  <!-- 根据 usercode 查询当前用户下的所有权限  -->
```

```
  <select id="getPermissionsByUsercode" resultType="SysPermission"
parameterType="string">
  SELECT
    *
    FROM
    sys_permission
    WHERE TYPE = 'permission'
    AND id IN
    (SELECT
    sys_permission_id
    FROM
    sys_role_permission
    WHERE sys_role_id IN
    (SELECT
    sys_role_id
    FROM
    sys_user_role
    WHERE sys_user_id IN
    (SELECT
    id
    FROM
    sys_user
    WHERE usercode = #{value})))
  </select>
```

3.2.4）实体类 SysUser.java

```
 private List<SysPermission> menus;           //得到当前用户下的所有菜
单集合
    private List<SysPermission> permissions; //得到当前用户下的所有权
限集合
```

3.2.5）认证拦截器的定义

```
public class AuthenticateInterceptor extends HandlerInterceptorAdapter {
    @Override
    public boolean preHandle(HttpServletRequest request,
HttpServletResponse response, Object handler)
            throws Exception {
        //第一部分：判断当前的 url 是否在 niming 的资源集合中,在就放行
        //1、得到请求的 url
        String url = request.getRequestURI();
        //只需要得到最后一个/后的路径即可,另外排除？后的部分
        url = CommUtils.getPath(url);
        System.out.println(url);
        //2.读取指定资源文件的 key 的集合（匿名 URL 地址的集合，所谓匿名是指不经
登录就可以访问的资源 url）
```

```
        List<String> keys = ResourcesUtil.gekeyList("anonymousURL");
        //3.看当前的 url 是否在其中
        boolean contains = keys.contains(url);
        //4.如果当前的 url 在匿名访问的资源集合中，就放行
        if(contains) return true;

        //第二部分：判断 session 中是否有 user
        HttpSession session = request.getSession();
        SysUser user = (SysUser) session.getAttribute("user");
        //如果当前的 user 不为空，证明登录成功，则放行
        if(user != null) return true;

        //第三部分:就是真正进行拦截
        response.sendRedirect(request.getContextPath() + "/login.jsp");
        return false;
    }

}
```

3.2.6) 授权拦截器的定义：

```
public class AuthorizationInterceptor extends HandlerInterceptorAdapter {
    @Override
    public boolean preHandle(HttpServletRequest request,
HttpServletResponse response, Object handler)
            throws Exception {
        //第一部分：判断当前的 url 是否在匿名的资源集合中,在就放行
        //1、得到请求的 url
        String url = request.getRequestURI();
        //只需要得到最后一个/后的路径即可,另外排除？后的部分
        url = CommUtils.getPath(url);
        System.out.println(url);
        //2.读取指定资源文件的 key 的集合（匿名 URL 地址的集合，所谓匿名是指不经
登录就可以访问的资源 url）
        List<String> keys = ResourcesUtil.gekeyList("anonymousURL");
        //3.看当前的 url 是否在其中
        boolean contains = keys.contains(url);
        //4.如果当前的 url 在匿名访问的资源集合中，就放行
        if(contains) return true;

        //第二部分：判断当前的 url 是否在公有的资源集合中,在就放行
        List<String> commUrls = ResourcesUtil.gekeyList("commonURL");
        contains = commUrls.contains(url);
        if(contains) return true;
```

```
        //第三部分：根据当前 url 是否在指定的权限列表 url 中
        //3.1) 取得 session 对象
        HttpSession session = request.getSession();
        SysUser user = (SysUser) session.getAttribute("user");
        //3.2) 取得当前用户下的权限集合
        List<SysPermission> permissions = user.getPermissions();
        System.out.println("permissions:" +  permissions);
        //3.3) 判断当前 url 是否在此权限集合中
        for (SysPermission permission : permissions) {
            if(permission.getUrl().contains(url)){
                return true;
            }
        }
        //第四部分：跳转到权限验证失败页面
    request.getRequestDispatcher("/WEB-INF/jsp/resure.jsp").forward(requ
est, response);
        return false;
    }
}
```

3.2.7）配置拦截器：

```
<!-- 配置拦截器 -->
    <mvc:interceptors>
        <!-- 1.配置认证拦截器 -->
        <mvc:interceptor>
            <mvc:mapping path="/**"/>
            <bean
class="com.zelin.web.interceptor.AuthenticateInterceptor"/>
        </mvc:interceptor>
        <!-- 2.配置授权拦截器 -->
        <mvc:interceptor>
            <mvc:mapping path="/**"/>
            <bean
class="com.zelin.web.interceptor.AuthorizationInterceptor"/>
        </mvc:interceptor>
    </mvc:interceptors>
```

3.2.8）配置文件：

anonymousURL.properties:

```
Login.action=登录
```

commURL.properties:

```
index.action=首页
logout.action=退出
menu.action=菜单功能
welcome.action=欢迎页面
```

3.2.9）工具类：（参见案例）

3.2.10）权限菜单/WEB-INF/jsp/layout/menu.jsp 页面：

```
<div title="权限维护" data-options="iconCls:'icon-group_key'"
          style="overflow: auto; padding: 10px;">
          <c:forEach items="${user.menus }" var="menu">
              <a href="#" class="easyui-linkbutton"
data-options="plain:true"
              style="display: block"
onclick="managerOpe(this,'${baseUrl}${menu.url}')">
                  ${menu.name }
              </a>
          </c:forEach>
      </div>
```

3.2.11）最终运行效果：

4、 Shiro 简介