

**P2P后台管理系统讲义**

# 项目概述

## 项目概述

该项目是某互联网金融平台的后台管理项目，主要使用对象为公司内部员工，比如债权运营人员、市场运营人员、产品人员、客服人员、公司领导等

P2P后台系统是支撑P2P前台系统的

P2P后台系统可以为公司的业务运营提供一些自动化的处理，减少人工处理，提高运营效率

P2P后台系统是与P2P前端业务系统一起统筹设计的

开发团队：3人

## 项目功能

* 用户账户管理（有管理员账号和员工账号，管理员拥有至高无上的权限，可以为公司内部员工开通账户）
* 轮播图管理（管理pc站、H5站、APP首页上展示的轮播图，可以对轮播图开启或禁用，设置展示的时间等）
* 产品管理（管理pc站、H5站、APP展示的理财产品、产品列表，能发布产品、审核产品）
* 合同管理（主要对债权合同进行管理，定时生成合同及手动生成合同，pdf版本合同将调用签章系统进行电子签章，H5站、APP合同是一个html的版本 ）
* 债权的管理（对借款人的借款信息进行管理，包括录入债权、接收第三方债权等）
* ......还有很多功能，暂时没有开发

## 技术选型&工具

前端：ACE (bootstrap框架) + jQuery (一些插件) + layer +html + Thymeleaf

后端：Spring Boot + MyBatis + RBAC权限 + SpringCloud

相关组件：HttpComponents、Apache commons、Druid、iText 等；

服务器：Tomcat、MySQL、FastDFS

项目依赖管理：Maven

项目开发工具：IDEA

# 集成已有项目p2p-admin

p2p-admin项目部署步骤

* 将项目导入到idea工具中；
* 在本地Mysql上创建一个数据库叫 p2padmin
* 在项目工程下有一个p2padmin.sql脚本，使用该脚本将表和数据导入到p2padmin数据库；
* 在本地Mysql上创建一个数据库叫 p2p2
* 在项目工程下有一个p2p2.sql脚本，使用该脚本将表和数据导入到p2p2数据库；
* 修改项目的application.properties中的数据库连接信息配置；
* 运行项目springboot的P2pAdminApplication类的main方法；
* 访问项目，在浏览器地址栏输入：http://localhost:9500/p2p-admin/
* 输入账号：admin 密码：123456 登录系统；

# 根据已有项目介绍模块功能

## 账户管理功能模块

### 登录

### 七天内免登录

### 用户个人详细信息展示

### 用户列表

#### 分页

#### 切换显示条数

#### 分页信息

### 添加用户

### 修改用户

### 删除用户

### 禁用用户

### 权限管理

#### 菜单权限

#### 按钮权限

## 轮播图管理功能模块

### 轮播图列表

#### 分页

#### 切换显示条数

#### 分页信息

### 添加轮播图

### 修改轮播图

### 删除轮播图

### 禁用轮播图

## 产品管理功能模块

### 优选产品

分页

切换显示条数

分页信息

添加产品

删除产品

修改产品

产品发布

### 体验产品

分页

切换显示条数

分页信息

添加产品

删除产品

修改产品

产品发布

### 散标产品

分页

切换显示条数

分页信息

**注意：散标产品没有添加，它是通过债权添加的，在债权中有一个发布散标，相当于添加了一个散标产品**

删除产品

修改产品

**发布产品**

## 债权管理功能模块

### 自主债权

分页

切换显示条数

分页信息

录入债权

发布散标

修改债权

债权发布

删除债权

### 三方债权

分页

切换显示条数

分页信息

接收三方债权（webservice）

发布散标

修改债权

债权发布

删除债权

## 合同管理功能模块

### 合同列表

分页

切换显示条数

分页信息

### 合同定时生成

### 合同重新生成

### 合同下载

# 动手实践

**在后台管理系统中，权限控制是比较复杂的功能了，讲解RBAC权限模型**

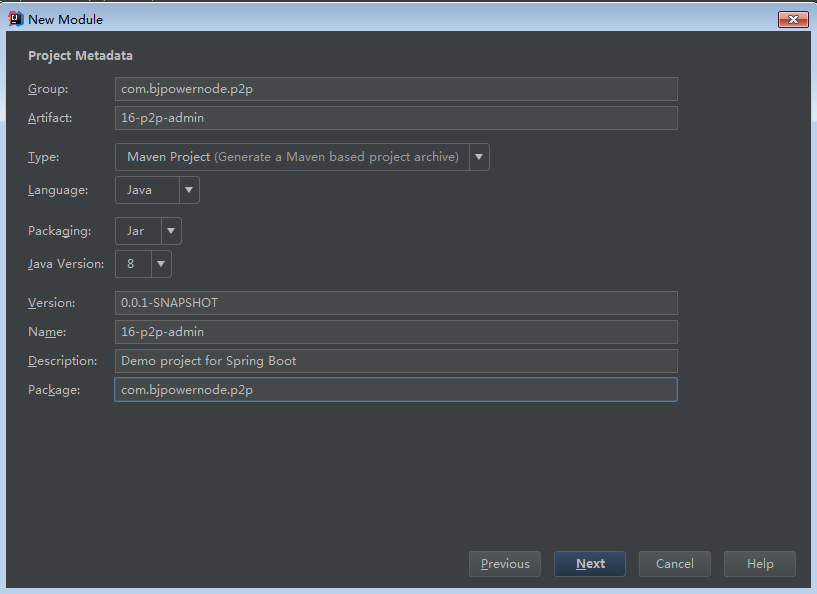
## 我们需要完成的功能

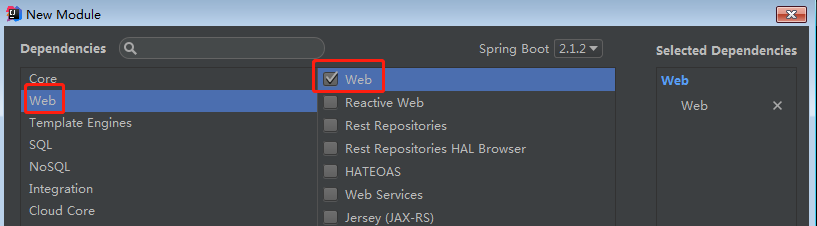
* 项目原型搭建
* 用户登录
* 获取用户菜单权限及按钮权限
* 7天内免登陆
* 个人中心页面中，根据登录用户拥有的菜单权限显示菜单
* 功能权限（按钮权限的控制）

## 项目原型搭建

### 搭建SpringBoot项目基础结构

#### 创建Springboot项目16-p2p-admin





#### 在16-p2p-admin的pom.xml中配置基础依赖

根据我们的技术选型，需要添加Jsp，Mybatis相关依赖

<dependencies>  
 <!--SpringBoot开发web项目起步依赖-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <!--SpringBoot测试依赖-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <!--引入Spring Boot内嵌的Tomcat对JSP的解析包，必须加，否则不能解析jsp页面-->  
 <dependency>  
 <groupId>org.apache.tomcat.embed</groupId>  
 <artifactId>tomcat-embed-jasper</artifactId>  
 </dependency>  
 <!-- servlet依赖的jar包start ，可选-->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 </dependency>  
 <!-- servlet依赖的jar包end -->  
  
 <!-- jsp依赖jar包start ，可选-->  
 <dependency>  
 <groupId>javax.servlet.jsp</groupId>  
 <artifactId>javax.servlet.jsp-api</artifactId>  
 <version>2.3.1</version>  
 </dependency>  
 <!-- jsp依赖jar包end -->  
  
 <!--jstl标签依赖的jar包start ，可选-->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>jstl</artifactId>  
 </dependency>  
 <!--jstl标签依赖的jar包end -->  
  
 <!--热部署插件-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <optional>true</optional>  
 </dependency>  
  
 <!-- 加载mybatis整合springboot -->  
 <dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <!--在springboot的父工程中没有指定版本，我们需要手动指定-->  
 <version>1.3.2</version>  
 </dependency>  
 <!-- MySQL的jdbc驱动包 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <!--在springboot的父工程中指定了版本，我们就不需要手动指定了-->  
 <artifactId>mysql-connector-java</artifactId>  
 </dependency>

<!-- commons-lang依赖jar包 -->  
<dependency>  
 <groupId>commons-lang</groupId>  
 <artifactId>commons-lang</artifactId>  
 <version>2.5</version>  
</dependency>

</dependencies>

#### 在16-p2p-admin的pom.xml文件中添加将指定资源编译到classpath的配置

<resources>  
 <!--mybatis的mapper.xml-->  
 <resource>  
 <directory>src/main/java</directory>  
 <includes>  
 <include>\*\*/\*.xml</include>  
 </includes>  
 </resource>  
 <!--src/main/resources下的所有配置文件编译到classes下面去-->  
 <resource>  
 <directory>src/main/resources</directory>  
 <includes>  
 <include>\*\*/\*.\*</include>  
 </includes>  
 </resource>  
 <resource>  
 <!--指定将jsp编译到指定位置-->  
 <directory>src/main/webapp</directory>  
 <!--编译到META-INF/resources，该目录不能随便写-->  
 <targetPath>META-INF/resources</targetPath>  
 <includes>  
 <!--要把哪些文件编译过去，\*\*表示webapp目录及子目录，\*.\*表示所有-->  
 <include>\*\*/\*.\*</include>  
 </includes>  
 </resource>  
</resources>

#### 在16-p2p-admin的核心配置文件application.properties中配置项目端口及上下文

#配置Spring boot内嵌的tomcat的端口号  
server.port=9600  
#配置项目访问工程名  
server.servlet.context-path=/p2p-admin

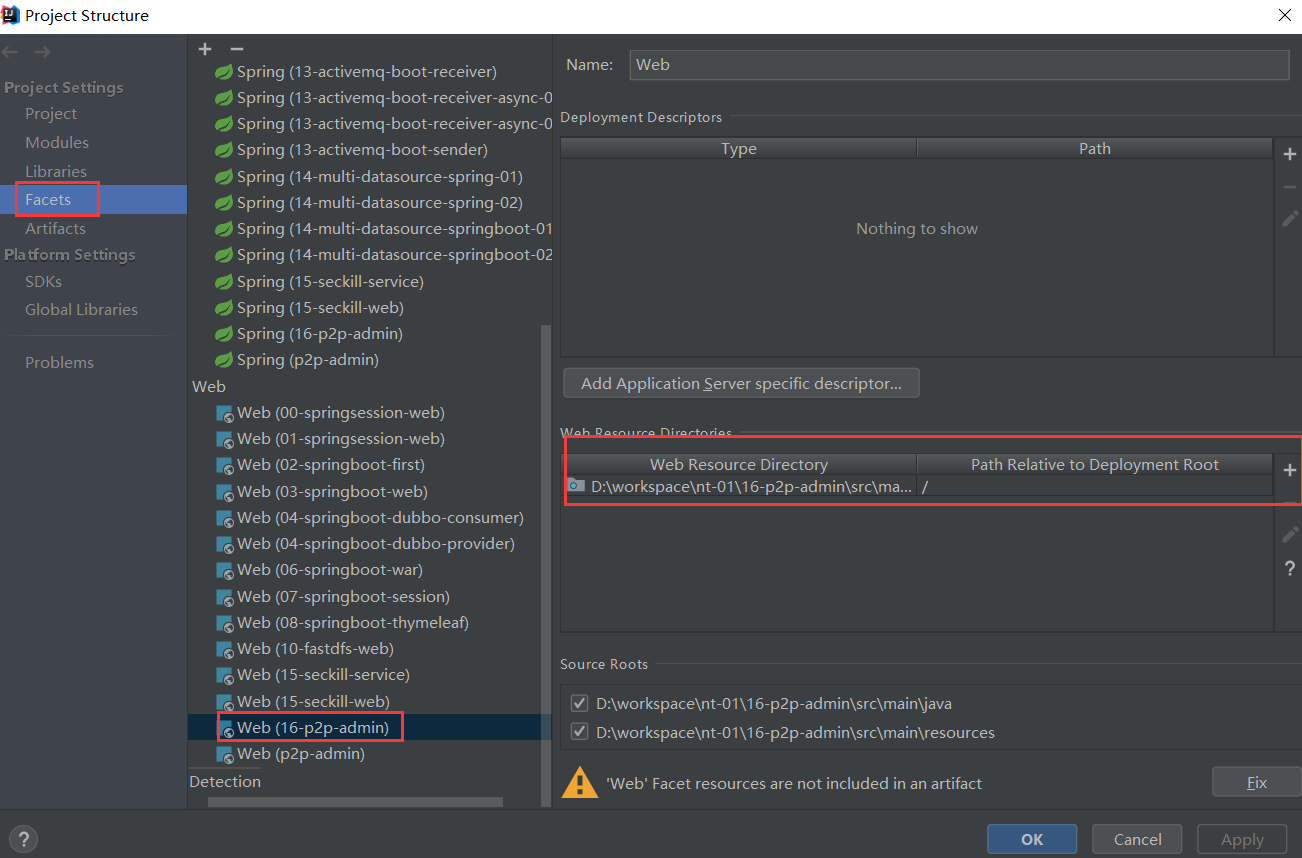
#### 在16-p2p-admin的核心配置文件application.properties中配置SpringBoot使用JSP展示数据

#配置springmvc视图展示层访问前缀，/斜杠表示访问src/main/webapp目录下的jsp页面  
spring.mvc.view.prefix=/WEB-INF/jsp/  
#配置springmvc视图展示层访问后缀，.jsp表示是jsp页面  
spring.mvc.view.suffix=.jsp

#### 在16-p2p-admin的核心配置文件application.properties中配置数据库连接信息

#配置数据库连接(p2padmin)  
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/p2padmin?serverTimezone=GMT%2B8&useUnicode=true&characterEncoding=utf8&useSSL=false  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.username=root  
spring.datasource.password=123456

#### 在16-p2p-admin下创建webapp/WEB-INF/jsp目录，并指定为WebResources Directory



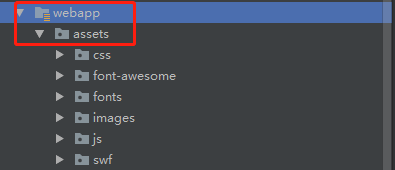
### 基于ACE搭建项目原型

#### 从ACE原型中拷贝登录页面login.html，修改为index.jsp

添加jsp头信息

<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>

#### 将ACE需要的js和css导入到我们的16-p2p-admin中



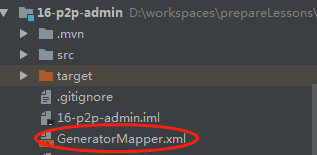
#### 放到webapp的WEB-INF目录下，WEB-INF下的资源是受保护的，不能直接访问。在16-p2p-admin的com.bjpowernode.p2p.admin.controller包下创建IndexController，并提供index方法，用于跳转到登录页面

@Controller  
public class IndexController {  
 @RequestMapping("/")  
 public String index(){  
 return "index";  
 }  
}

#### 参照已有p2p-admin进行修改，独立将ACE页面原型改造成我们项目的原型（培养这种能力）

### 使用Mybatis反向工程生成内容

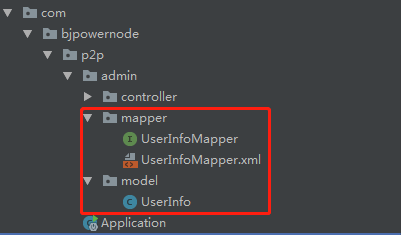
#### 拷贝逆向工程文件到16-p2p-admin中



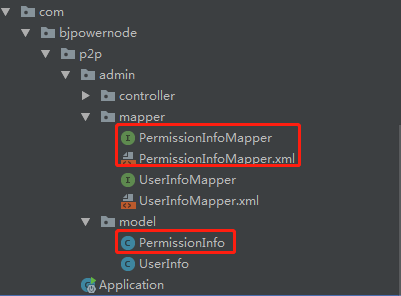
#### 在pom.xml文件中添加反向工程插件

<!--mybatis代码自动生成插件-->  
<plugin>  
 <groupId>org.mybatis.generator</groupId>  
 <artifactId>mybatis-generator-maven-plugin</artifactId>  
 <version>1.3.7</version>  
 <configuration>  
 <!--配置文件的位置-->  
 <configurationFile>GeneratorMapper.xml</configurationFile>  
 <verbose>true</verbose>  
 <overwrite>true</overwrite>  
 </configuration>  
</plugin>

#### 修改GeneratorMapper.xml生成用户UserInfo相关model和mapper



#### 修改GeneratorMapper.xml生成权限PermissionInfo相关model和mapper



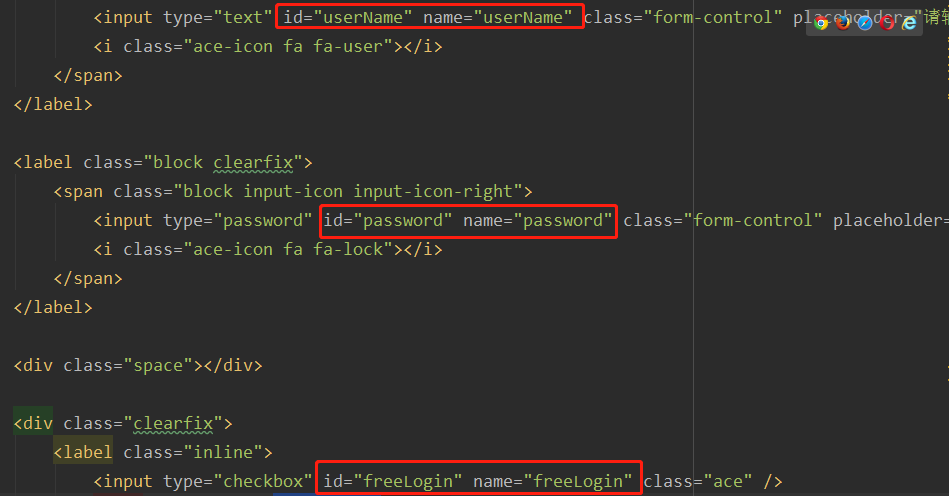
#### 在生成的Mapper接口上添加@Mapper注解，被SpringBoot扫描到

#### 为了操作方便，在UserInfo中添加上两个属性，代表用户所拥有的菜单权限和按钮权限，并提供get|set方法

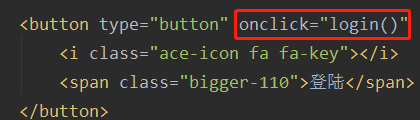
//当前用户拥有的所有菜单权限  
private List<PermissionInfo> menuPermissionInfoList;  
//当前用户拥有的所有按钮权限  
private List<PermissionInfo> buttonPermissionInfoList;

## 用户登录及获取用户拥有权限功能实现

### 在16-p2p-admin的index.jsp中为用户名、密码、免登陆的input添加id和name属性



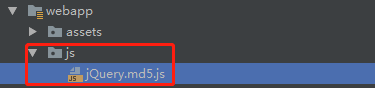
### 在16-p2p-admin的index.jsp中给登录按钮添加单击事件



### 引入jQuery的md5插件

**注意：因为数据库中存的是加密后的密码，所有我们这里需要对密码进行加密，加密可以在页面中通过jquery插件加密，也可以在控制层通过java代码加密，我们这里在页面通过jQuery插件加密**

**从已有的p2p-admin中拷贝jQuery.md5.js到16-p2p-admin/webapp/js目录下，并在index.jsp中引入**

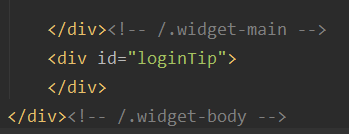


<script src="**${**pageContext.request.contextPath**}**/js/jQuery.md5.js"></script>

### 在16-p2p-admin的index.jsp的login函数中发送ajax请求进行登录

**function** login() {  
 **var** userName = $("#userName").val();  
 **var** password = $("#password").val();  
 **var** freeLogin = $("#freeLogin").is(":checked");  
 **if**(userName ==""){  
 $("#loginTip").html("<span style='color: red;'>请输入登陆账号</span>");  
 **return false**;  
 }**else if**(password ==""){  
 $("#loginTip").html("<span style='color: red;'>请输入登陆密码</span>");  
 **return false**;  
 }  
 //MD5加密  
 password = $.md5(password);  
 $.ajax({  
 url:"**${**pageContext.request.contextPath**}**/admin/login",  
 type:"POST",  
 dataType:"json",  
 data:{  
 "userName":userName,  
 "password":password,  
 "freeLogin":freeLogin  
 },  
 success:**function** (rtnMessage) {  
 **if**(rtnMessage.errorCode == 1){  
 //登陆成功 跳转到主页面  
 window.location.href = "**${**pageContext.request.contextPath**}**/admin/profile";  
 }**else**{  
 //登陆失败 提示用户  
 $("#loginTip").html("<span style='color: red;'>"+rtnMessage.errorMessage+"</span>");  
 }  
 }  
 });  
 }

### 在index.jsp页面中添加loginTip提示



### 在16-p2p-admin的IndexController中添加login方法，处理登录请求

@PostMapping("/admin/login")  
public @ResponseBody ReturnObject login(HttpServletRequest request,HttpServletResponse response,  
 @RequestParam("userName") String userName,  
 @RequestParam("password") String password,  
 @RequestParam("freeLogin") boolean freeLogin){  
 ReturnObject returnObject = new ReturnObject();  
 UserInfo userInfo = userInfoService.getUserInfoByLogin(userName,password);  
 if(userInfo == null){  
 //登陆失败  
 returnObject.setErrorCode(Constants.*ZERO*);  
 returnObject.setErrorMessage("账号或者密码不匹配");  
 return returnObject;  
 }else{  
 //登陆成功  
 //将用户信息放到session中  
 request.getSession().setAttribute(Constants.*SESSION\_USER*,userInfo);  
 returnObject.setErrorCode(Constants.*ONE*);  
 returnObject.setErrorMessage("账号成功");  
 return returnObject;  
 }  
}

### 在16-p2p-admin的com.bjpowernode.p2p.admin.rto包下创建ReturnObject类封装返回的对象

public class ReturnObject {  
 private int errorCode;  
 private String errorMessage;  
 private Object data;

//省略get|set

### 在16-p2p-admin的com.bjpowernode.p2p.admin.constant包下创建Constants类，并配置常用常量

public class Constants {  
 //session中的用户  
 public static final String *SESSION\_USER* = "userInfo";  
 //处理失败  
 public static final int *ZERO* = 0;  
 //处理成功  
 public static final int *ONE* = 1;  
}

### 在16-p2p-admin的com.bjpowernode.p2p.admin.service包下创建UserInfoService接口，提供getUserInfoByLogin方法

public interface UserInfoService {  
 UserInfo getUserInfoByLogin(String userName,String password);  
}

### 在16-p2p-admin的com.bjpowernode.p2p.admin.service.impl包下创建UserInfoServiceImpl实现类，对getUserInfoByLogin方法进行实现

@Service  
public class UserInfoServiceImpl implements UserInfoService{  
 @Autowired  
 private UserInfoMapper userInfoMapper;  
 @Autowired  
 private PermissionInfoMapper permissionInfoMapper;  
 @Override  
 public UserInfo getUserInfoByLogin(String userName, String password) {  
 UserInfo userInfo = userInfoMapper.selectByLogin(userName,password);  
 if(userInfo != null){  
 //登录成功，获取用户的权限信息  
 //获取菜单权限  
 List<PermissionInfo> menuPermissionInfoList  
 = permissionInfoMapper.selectPermissionInfoByUserIdAndType(userInfo.getId(),"menu");  
 //获取按钮权限  
 List<PermissionInfo> buttonPermissionInfoList  
 = permissionInfoMapper.selectPermissionInfoByUserIdAndType(userInfo.getId(),"button");  
 userInfo.setMenuPermissionInfoList(menuPermissionInfoList);  
 userInfo.setButtonPermissionInfoList(buttonPermissionInfoList);  
 }  
 return userInfo;  
 }  
}

### 在16-p2p-admin的UserInfoMapper接口中添加selectByLogin方法

*/\*\*  
 \* 如果Mapper接口中有多个参数，我们一般有以下几种处理方式  
 \* 1.封装Map集合  
 \* 2.封装参数对象  
 \* 3.@param注解  
 \* 4.根据索引获取  
 \*/*UserInfo selectByLogin(@Param("userName") String userName,   
 @Param("password") String password);

### 在16-p2p-admin的UserInfoMapper.xml文件中对selectByLogin进行实现

<select id="selectByLogin" resultMap="BaseResultMap">  
 select  
 <include refid="Base\_Column\_List" />  
 from u\_user\_info  
 where username = #{userName,jdbcType=VARCHAR} and password = #{password,jdbcType=VARCHAR}  
</select>

### 在16-p2p-admin的PermissionInfoMapper接口中添加selectPermissionInfoByUserIdAndType方法

List<PermissionInfo> selectPermissionInfoByUserIdAndType(@Param("userId") Integer userId,  
 @Param("type") String type);

### 在16-p2p-admin的PermissionInfoMapper.xml文件中对selectPermissionInfoByUserIdAndType进行实现

**这条语句比较复杂，现在Navicat中测试通过后再编写**

<sql id="My\_Base\_Column\_List">  
 pi.id, pi.name, pi.type, pi.url, pi.code, pi.parentid, pi.sort, pi.available  
</sql>  
  
<select id="selectPermissionInfoByUserIdAndType" resultMap="BaseResultMap">  
 select  
 <include refid="My\_Base\_Column\_List" />  
 from  
 u\_permission\_info pi  
 left join  
 u\_role\_permission rp on pi.id = rp.permission\_id  
 left join  
 u\_role\_info ri on ri.id = rp.role\_id  
 left join  
 u\_user\_role ur on ur.user\_id = ri.id  
 where  
 ur.user\_id = #{userId,jdbcType=INTEGER} and type = #{type,jdbcType=VARCHAR}  
</select>

### 登录测试

## 七天内免登陆

如果用户登录成功，判断是否勾选了7天内免登陆，如果勾选了，将用户名和密码放到cookie中，下次使用该浏览器访问的时候，直接从浏览器中读取用户名和密码，自动为用户登录。

### 在16-p2p-admin的com.bjpowernode.p2p.admin.cookie包下创建MyCookieUtil类，并创建addCookie方法，用于向浏览器放cookie

public class MyCookieUtil {  
 */\*\*  
 \* 向浏览器中放cookie  
 \*/* public static void addCookie(HttpServletRequest request,HttpServletResponse response,  
 String cookieName,String cookieValue){  
 Cookie cookie = new Cookie(cookieName,cookieValue);  
 //7天  
 cookie.setMaxAge(7\*24\*60\*60);  
 //将cookie放到项目的上下文根下  
 cookie.setPath(request.getContextPath());  
 //通过response向浏览器响应cookie  
 response.addCookie(cookie);  
 }  
}

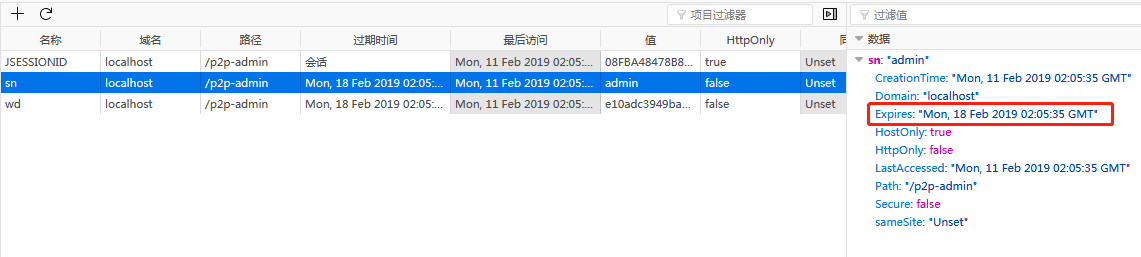
### 在16-p2p-admin的IndexController的login方法中，加判断

else{  
 //登陆成功  
 //判断是否勾选了7天内免登陆  
 if(freeLogin){  
 //将用户名和密码放到cookie中  
 MyCookieUtil.*addCookie*(request,response,Constants.*COOKIE\_USERNAME*,userName);  
 MyCookieUtil.*addCookie*(request,response,Constants.*COOKIE\_PASSWORD*,password);  
 }  
 //将用户信息放到session中  
 request.getSession().setAttribute(Constants.*SESSION\_USER*,userInfo);  
 returnObject.setErrorCode(Constants.*ONE*);  
 returnObject.setErrorMessage("账号或者密码不匹配");  
 return returnObject;  
}

### 在Constants类中添加cookie中用于存放用户名和密码的名字常量

//cookie中存放用户名的常量名  
public static final String *COOKIE\_USERNAME* = "sn";  
//cookie中存放密码的常量名  
public static final String *COOKIE\_PASSWORD* = "wd";

### 浏览器访问登录页面，并勾选7天内免登陆，登录成功后，测试cookie是否设置成功



### 在16-p2p-admin的MyCookieUtil类中创建getCookie方法，用于获取浏览器中携带的cookie

*/\*\*  
 \* 获取浏览器中携带cookie  
 \*/*public static String getCookie(HttpServletRequest request, String cookieName) {  
 String cookieValue = null;  
 Cookie[] cookies = request.getCookies();  
 if(cookies != null){  
 for (Cookie cookie : cookies) {  
 //如果遍历出的cookie名字就是你要获取的cookie名  
 if(cookie.getName().equals(cookieName)){  
 cookieValue = cookie.getValue();  
 break;  
 }  
 }  
 }  
 return cookieValue;  
}

### 修改16-p2p-admin的IndexController的index方法，实现自动登录

@RequestMapping("/")  
public String index(HttpServletRequest request,Model model){  
 //读取浏览cookie中携带的用户名和密码，然后自动登录  
 String userName = MyCookieUtil.*getCookie*(request,Constants.*COOKIE\_USERNAME*);  
 String password = MyCookieUtil.*getCookie*(request,Constants.*COOKIE\_PASSWORD*);  
 //如果cookie中携带了用户名和密码并且不为null  
 if(StringUtils.*isNotEmpty*(userName)&&StringUtils.*isNotEmpty*(password)){  
 //通过model将用户名和密码传递到index.jsp页面，并传递一个免登陆标记  
 model.addAttribute("userName",userName);  
 model.addAttribute("password",password);  
 //是否免登陆标记，1为免登陆  
 model.addAttribute("loginFlag",1);  
 }  
 return "index";  
}

### 在index.jsp页面中，给用户名和密码加value属性



### 页面加载完成之后，判断是否为免登陆

//判断是否为免登陆  
**var** loginFlag = "**${**loginFlag**}**";  
**if** (loginFlag == "1"){  
 login();  
}

### 修改index.jsp中的login函数，如果是自动登录，密码从后台获取的就是加密的，所以不需要加密

**var** loginFlag = "**${**loginFlag**}**";  
**if** (loginFlag !="1"){  
 //MD5加密  
 password = $.md5(password);  
}

### 浏览器访问测试

## 个人中心页面中，根据登录用户拥有的菜单权限显示菜单

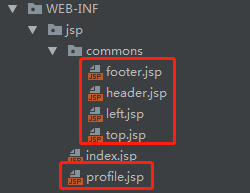
用户登录成功之后，已经将用户拥有的所有菜单权限和按钮权限查询出来，并放到用户对象上了，不同的用户登录，能看到的菜单权限是不一样的。

一般在后台管理系统中，有些页面的信息是保持不变的，比如：header、footer等，所有可以将这些页面单独提取出来，在需要的地方通过jsp的静态include或者动态include导入，**注意动态include中不能出现${request.contextPath}这样的语法**

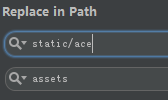
### 在16-p2p-admin的IndexController中创建profile方法，跳转到个人中心页面

*/\*\*  
 \* 跳转到个人中心页面  
 \*/*@RequestMapping("/admin/profile")  
public String profile(){  
 return "profile";  
}

### 为16-p2p-admin准备个人中心页面，在个人中页面profile.jsp中引入公共的jsp，直接从已有项目p2p-admin复制即可

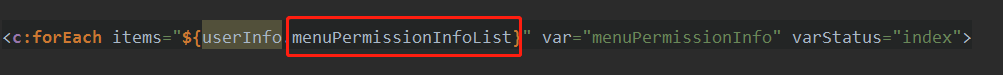


复制过来的路径需要修改，选中16-p2p-admin,CTRL+SHIFT+R,将static/ace替换为assets

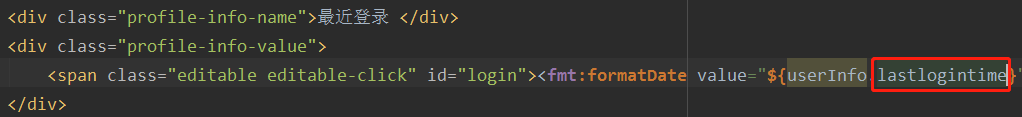




### 将left.jsp中，遍历属性名修改为和自己定义的保持一致



### 将profile.jsp中，用户的最后登录时间修改为小写



### 在PermissionInfoMapper.xml文件中，查询用户拥有的菜单权限时过滤掉管理面板



**在写SQL语句的时候，如果出现<>会报错，解决方案**

* !=
* &lt;&gt;
* <![CDATA[ ]]>

<select id="selectPermissionInfoByUserIdAndType" resultMap="BaseResultMap">  
 select  
 <include refid="My\_Base\_Column\_List" />  
 from  
 u\_permission\_info pi  
 left join  
 u\_role\_permission rp on pi.id = rp.permission\_id  
 left join  
 u\_role\_info ri on ri.id = rp.role\_id  
 left join  
 u\_user\_role ur on ur.user\_id = ri.id  
 where  
 ur.user\_id = #{userId,jdbcType=INTEGER} and type = #{type,jdbcType=VARCHAR}  
 /\*and pi.parentid != 0\*/  
 /\*and pi.parentid &lt;&gt; 0\*/  
 <![CDATA[ and pi.parentid <> 0]]>  
</select>

### 浏览器用不同的用户登录查看访问菜单情况

admin->123456

shichang->123456

zhaiquan->123456

## 功能权限（按钮权限的控制）

* 需求

用户有这个权限，才能访问这个功能，例如

**债权用户**

有 合同管理 查看权限 http://localhost:9600/p2p-admin/admin/contract

没有 轮播图 查看权限 http://localhost:9600/p2p-admin/admin/slideShow

**市场用户**

没有 合同管理 查看权限 http://localhost:9600/p2p-admin/admin/contract

有 轮播图 查看权限 http://localhost:9600/p2p-admin/admin/slideShow

* 实现思路
  + 用户登录成功后，会将用户拥有的权限获得
  + 当用户访问某一个功能的时候，会发起对应的url请求
  + 使用拦截器对请求进行拦截
  + 在拦截器中，从用户拥有的权限中，判断是否包含该请求的URL，如果包含继续访问，否则没有权限

### 在16-p2p-admin中创建SlideShowController类，并提供slideShow方法，接收跳转到轮播图页面的请求

@Controller  
public class SlideShowController {  
 @RequestMapping("/admin/slideShow")  
 public String slideShow(){  
 return "slideShow";  
 }  
}

### 在16-p2p-admin中创建ContractController类，并提供contract方法，接收跳转合同页面的请求

@Controller  
public class ContractController {  
 @RequestMapping("/admin/contract")  
 public String contract(){  
 return "contract";  
 }  
}

### 拷贝profile.jsp，命名为slideShow.jsp，修改paga-content的div里面的内容为轮播图列表（模拟）

<div class="page-content">  
 <h1>轮播图列表</h1>  
</div>

### 拷贝profile.jsp，命名为contract.jsp，修改paga-content的div里面的内容为轮播图列表（模拟）

<div class="page-content">  
 <h1>合同列表</h1>  
</div>

### 在16-p2p-admin的com.bjpowernode.p2p.admin.interceptor包下创建PermissionInterceptor类，并重写preHandle方法

public class PermissionInterceptor implements HandlerInterceptor{  
 @Override  
 public boolean preHandle(HttpServletRequest request, HttpServletResponse response,  
 Object handler) throws Exception {  
 //获取用户在页面上请求的URI /项目上下文/admin/contract  
 String requestURI = request.getRequestURI();  
 //从session获取用户  
 UserInfo userInfo = (UserInfo) request.getSession().getAttribute(Constants.*SESSION\_USER*);  
 //获取用户拥有的菜单权限  
 List<PermissionInfo> menuPermissionInfoList = userInfo.getMenuPermissionInfoList();  
 //获取用户拥有的功能权限  
 List<PermissionInfo> buttonPermissionInfoList = userInfo.getButtonPermissionInfoList();  
 //对用户拥有的菜单权限进行遍历  
 for (PermissionInfo permissionInfo : menuPermissionInfoList) {  
 //用户拥有的菜单权限对应的URL /admin/contract  
 String url = permissionInfo.getUrl();  
 //如果菜单权限URL不为空，并且用户请求的URI包含用户拥有的url，说明有访问该请求的权限，继续访问  
 if (StringUtils.*isNotEmpty*(url) && requestURI.indexOf(url) >=0){  
 return true;  
 }  
 }  
 //对用户拥有的功能权限进行遍历  
 for (PermissionInfo permissionInfo : buttonPermissionInfoList) {  
 //用户拥有的按钮权限对应的URL /admin/contract  
 String url = permissionInfo.getUrl();  
 //如果按钮权限URL不为空，并且用户请求的URI包含用户拥有的url，说明有访问该请求的权限，继续访问  
 if (StringUtils.*isNotEmpty*(url) && requestURI.indexOf(url) >=0){  
 return true;  
 }  
 }  
  
 //如果执行到这里，说明没有访问该请求的权限，不在继续访问，重定向到提示页面  
 response.sendRedirect(request.getContextPath() +"/admin/refuse");  
 return false;  
 }  
}

### 在IndexController中添加refuse方法，跳转到没有权限提示页面

*/\*\*  
 \* 跳转到权限不足提示页面  
 \*/*@RequestMapping("/admin/refuse")  
public String refuse(){  
 return "refuse";  
}

### 拷贝contract.jsp，命名为refuse.jsp，修改paga-content的div里面的内容为权限不足，无法访问

<div class="page-content">  
 <h1>权限不足，无法访问</h1>  
</div>

### 为了让拦截器生效，需要拦截器配置类，在16-p2p-admin的com.bjpowernode.p2p.admin.config包下创建WebMVCConfig类

@Configuration  
public class WebMVCConfig implements WebMvcConfigurer {  
 @Override  
 public void addInterceptors(InterceptorRegistry registry) {  
 //要拦截的路径  
 String[] patterns = {  
 "/\*\*"  
 };  
 //不需要权限拦截的url，也就是公共url，任何用户登录后都可以访问  
 String[] excludePathPatternsForPermission = {  
 "/admin/login",  
 "/admin/logout",  
 "/admin/profile",  
 "/",  
 "/admin/refuse",  
 "/assets/\*\*",  
 "/js/\*\*"  
 };  
 //权限拦截器  
 registry.addInterceptor(new PermissionInterceptor()).addPathPatterns(patterns)  
 .excludePathPatterns(excludePathPatternsForPermission);  
 }  
}

### 登录不同的用户，访问用户没有权限的功能进行测试

## 合同管理功能解读

* 底层mapper涉及跨库查询，如果不同的库在同一个IP和端口下，是可以通过库名.表名的方式直接查询的
* 有两个地方可以生成合同，页面和定时任务
* 调用webservice流程
* e-seal生成pdf代码解读