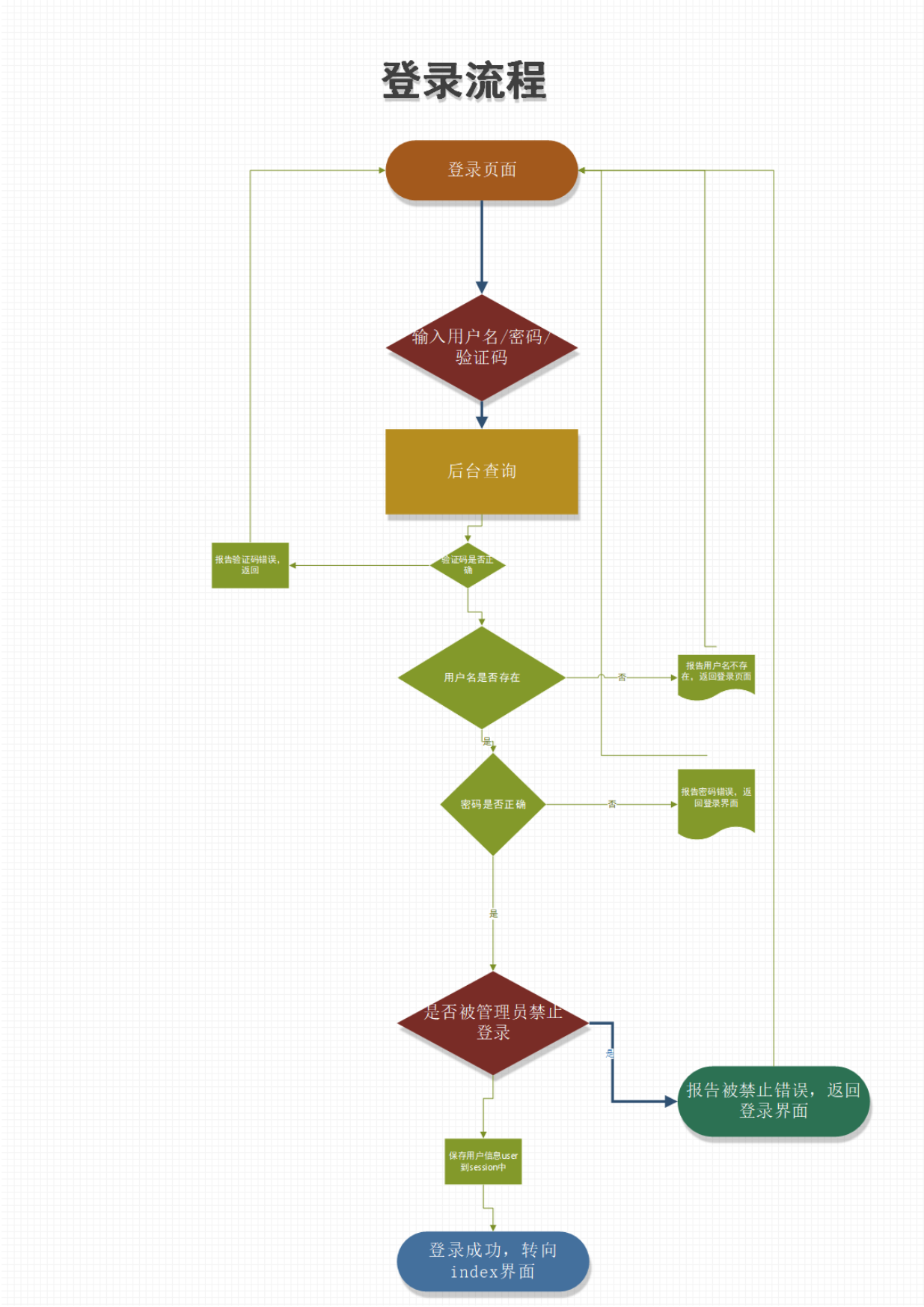
# 登录，注册页面详解

注册，登录页面非常类似，这里先着重讲登录页面。

登录页面样式：


登录后端实现思路：




(我后来实现的时候没有加验证码，请忽略验证码的哪块)

为了让文章更简洁，html代码我不在这里给出，需要看的话，直接去github里找login.html页面即可。

因为这里提交前端信息用到了vue（也是本项目唯一一次用到vue），所以将这个页面的js代码帖出来，并给出相关讲解。

```
<script src="https://cdn.jsdelivr.net/npm/vue-resource@1.5.1"></script>
<script type='text/javascript'>
    Vue.http.options.emulateJSON = true;
    var loginPage = new Vue({
        el: '#loginPage', //绑定id=loginPage的标签
        data: {
            'username': '',
            'password': ''
        },
        methods: {
            login: function(event){
                var ok = $('#form').parsley().isValid({force: true});
                //验证用户名，密码是否为空
                if(!ok){
                    return;
                }
                var datas={
                //将你填写的用户名和密码复制给userName,password两个变量里。
                    userName: this.username,
                    password: this.password
                };
                this.$http.post('/loginPage',datas).then(function(respons
e){
                //类似于提交表单的操作,"/loginPage"是提交到后端对应的地方,datas是
携带的数据，也就是说，我将要把datas提交给后端的"/loginPage"，(then)接着后端会返回给
我一个数据类型，这里我用response来接收。
                    if(response.data.rspCode == '000000'){
                    //后端会返回登录成功与否的信息
                        window.open(response.data.data, '_self');
                    }else{
                        $("#errorMsg").html(response.data.rspMsg);
                        $("#errorMsg").show();
                    }
                }, function(response){
                    console.log('error');
                })
            }
        }
    })
</script>
```

注意我这里引用vue是引用了网页版的，如果没有联网的话是不能使用vue的。

前端逻辑大致讲完了。
现在讲讲后端。处理login的函数在loginController.java里面

```
@RequestMapping("loginPage")
//RequestMapping是一种映射，刚刚我们前端提交的地方是"/loginPage"，也就是说通过@Req
uestMapping,前端的信息将来到这个函数里面。
    public ResponseData login(HttpServletRequest request, User user) {
```

```
            //直接在这里讲一下接收前端传来的数据的规则，因为前端我们定义了userName和password来存储信息，所以我们后端只要定义相同的变量名称就可以实现接收了。而我们的user实体类刚好有userName和password，所以直接放一个user对象即可（千万注意，命名一定要和前端完全一致），如果你不想用user去接收，那你就直接用(String userName,String password)去接收。
        try {
            User dbUser = userServiceImp.getUserByName(user.getUserName());
            //先查找有没有这个用户
            System.out.println(dbUser);
            if (dbUser == null) {
                System.out.println("1");
               //没有这个用户，返回错误信息
                return new ResponseData(ExceptionMsg.LoginNameNotExists);
            } else if (!dbUser.getPassword().equals(user.getPassword())) {
            //密码不对
                System.out.println("2");
                return new ResponseData(ExceptionMsg.LoginNameOrPassWordError);
            } else if (dbUser.getLocked() == User.user_locked) {
            //这个用户被管理员锁定了，暂时不允许登录
                System.out.println("'3");
                return new ResponseData(ExceptionMsg.Locked);
            } else {
            //成功,保存登录信息到数据库
                System.out.println("'4");
                dbUser.setLastIp(request.getRemoteAddr());
                dbUser.setLastVisit(new Date());

                userServiceImp.loginSuccess(dbUser);
                setSessionUser(request, dbUser);
                String toUrl = (String)
                request.getSession().getAttribute(Constantion.LOGIN_TO_URL);//记忆化历史浏览记录,这个主要是解决用户之前在浏览一个页面，后来再次登录的时候，可以让用户直接回到之前定的页面
                request.getSession().removeAttribute(Constantion.LOGIN_TO_URL);
                if (toUrl == null) {
                    toUrl = "/index";
                    //如果没有记录，则跳转到index页面
                }
                return new ResponseData(ExceptionMsg.SUCCESS, toUrl);
            }
        } catch (Exception e) {
            System.out.println("操作失败");
            return new ResponseData(ExceptionMsg.FAILED);
        }
    }
```

注册页面不再叙述，可以直接看registerController.java和register.html，里面都有注释。