用户模块

一 学习目标

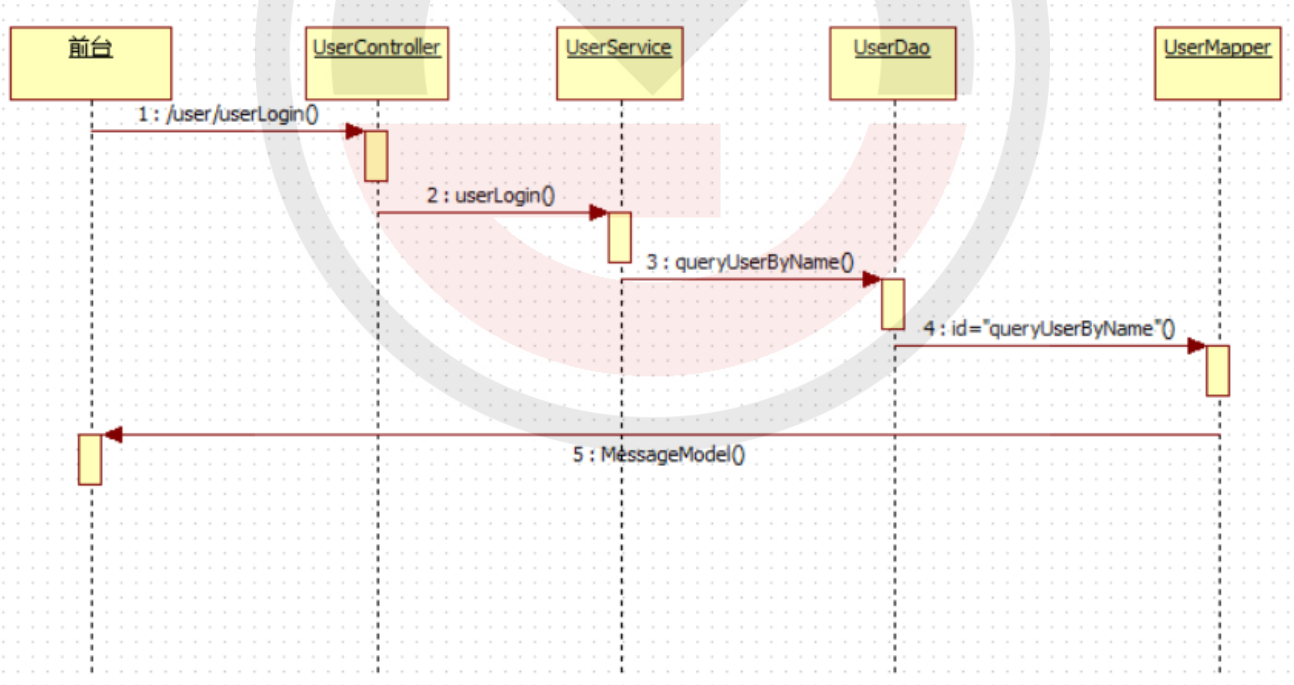
二 登陆

1 需求分析

点击登陆，判断用户名密码是否正确

2 页面原型

3 流程图



4 数据库信息

t_user

	列名	数据类型
主建名	id	int
用户名	user_name	varchar
用户密码	user_pwd	varchar
手机号	phone	varchar
邮箱号	email	varchar
标志位	is_valid	int
创建时间	create_date	datetime
更新时间	update_date	datetime
用户真实姓名	true_name	varchar

5 编写Controller

```
        @Controller
        @RequestMapping("user")
        public class UserController extends BaseController {

            @Resource
            private UserService userService;

            @RequestMapping("userLogin")
            @ResponseBody
            public MessageModel userLogin(String userName, String userPwd){
                MessageModel messageModel=new MessageModel();
                try{
                    UserModel userModel = userService.userLogin(userName, userPwd);
                    messageModel.setResult(userModel);
                }catch (ParamsException e){
                    e.printStackTrace();
                    messageModel.setCode(CrmConstant.LOGIN_FAILED_DODE);
                    messageModel.setMsg(e.getMsg());
                }catch (Exception e){
                    e.printStackTrace();
                    messageModel.setCode(CrmConstant.OPS_FAILED_DODE);
                    messageModel.setMsg(CrmConstant.OPS_FAILED_MSG);
                }
                return messageModel;
            }
        }
         @RequestMapping("main")
            public String main(HttpServletRequest request) throws
UnsupportedEncodingException {
                /**
                 * 从request中，获取cookie 得到userName,和trueName
```

```
                 */
                /*Cookie[]  cookies = request.getCookies();
                for (int i = 0; i <cookies.length ; i++) {
                    Cookie cookie = cookies[i];
                    if("userName".equals(cookie.getName())){
                        request.setAttribute("userName",cookie.getValue());
                    }else if("trueName".equals(cookie.getName())){
                        request.setAttribute("trueName",
URLDecoder.decode(cookie.getValue(),"utf-8"));
                    }
                }*/
                request.setAttribute("userName",
CookieUtil.getCookieValue(request,"userName"));
                request.setAttribute("trueName",
CookieUtil.getCookieValue(request,"trueName"));
                return "main";
            }
```

6 编写Service

```
@Service
public class UserService {

    @Resource
    private UserDao userDao;

    /**
     * 用户合法性的检测
     * @param userName
     * @param userPwd
     * @return
     */
    public UserModel userLogin(String userName, String userPwd){
        User user= userDao.queryUserByName(userName);
        AssertUtil.isTrue(null==user,"用户不存在");
        AssertUtil.isTrue("0".equals(user.getIsValid()),"用户已经注销");
        String pwd = Md5Util.encode(userPwd);
        AssertUtil.isTrue(!pwd.equals(user.getUserPwd()),"密码错误");
        return createUserModel(user);
    }

    /**
     * 创建UserModel
     * @param user
     * @return
     */
    public UserModel createUserModel(User user){
        UserModel userModel=new UserModel();
        userModel.setTrueName(user.getTrueName());
        userModel.setUserName(user.getUserName());
        String id= Base64Util.encode(user.getId());
        userModel.setUserId(id);
```

```
            return userModel;
        }
    }
```

7 编写Dao

```
    public interface UserDao {
        public User queryUserByName(String userName);
    }
```

8 编写Mapper

```
    <?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE mapper
            PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
            "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
    <mapper namespace="com.mage.crm.dao.UserDao">

        <sql id="user_columns">
            id , user_name as userName, user_pwd as userPwd,true_name as trueName,
            phone,email,is_valid as isValid,create_date as createDate,update_date as
updateDate

        </sql>

        <select id="queryUserByName" parameterType="string" resultType="user">

            SELECT <include refid="user_columns"/>
            from t_user where user_name = #{userName}
        </select>

    </mapper>
```

9 编写Vo

```
    public class User {

        private String id;
        private String userName;
        private String userPwd;
        private String trueName;
        private String phone;
        private String email;
        private String isValid;
```

```
    private String createDate;
    private String updateDate;

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getUserPwd() {
        return userPwd;
    }

    public void setUserPwd(String userPwd) {
        this.userPwd = userPwd;
    }

    public String getTrueName() {
        return trueName;
    }

    public void setTrueName(String trueName) {
        this.trueName = trueName;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getIsValid() {
        return isValid;
```

```
        }

        public void setIsValid(String isValid) {
            this.isValid = isValid;
        }

        public String getCreateDate() {
            return createDate;
        }

        public void setCreateDate(String createDate) {
            this.createDate = createDate;
        }

        public String getUpdateDate() {
            return updateDate;
        }

        public void setUpdateDate(String updateDate) {
            this.updateDate = updateDate;
        }
    }
```

10 编写model

10.1 UserModel

```
    public class UserModel {

        public String userName;
        public String trueName;
        public String userId;

        public String getUserId() {
            return userId;
        }

        public void setUserId(String userId) {
            this.userId = userId;
        }

        public String getUserName() {
            return userName;
        }

        public void setUserName(String userName) {
            this.userName = userName;
        }

        public String getTrueName() {
```

```
            return trueName;
        }

        public void setTrueName(String trueName) {
            this.trueName = trueName;
        }
    }
```

10.2 MessageModel

```
    public class MessageModel {
        private Integer code= CrmConstant.OPS_SUCCESS_DODE;
        private String msg=CrmConstant.OPS_SUCCESS_MSG;
        private Object result;

        public Integer getCode() {
            return code;
        }

        public void setCode(Integer code) {
            this.code = code;
        }

        public String getMsg() {
            return msg;
        }

        public void setMsg(String msg) {
            this.msg = msg;
        }

        public Object getResult() {
            return result;
        }

        public void setResult(Object result) {
            this.result = result;
        }
    }
```

11 封装Utils

11.1 AssertUtil

```
    public class AssertUtil {

        public static void isTrue(boolean flag,String msg){
            if(flag){
```

```
                    throw new ParamsException(300,msg);
                }
            }

            public static void isTrue(boolean flag,Integer code,String msg){
                if(flag){
                    throw new ParamsException(code,msg);
                }
            }
        }
```

11.2 Base64Util

```
        public class Base64Util {

            public static  String encode(String id){
                String encodeString = Base64.encodeBase64String(id.getBytes());
                String timeString = Base64.encodeBase64String((System.currentTimeMillis() +
"").getBytes());
                String encode=timeString+encodeString+timeString.substring(4,8);
                encode = new StringBuilder(encode).reverse().toString();
                return encode.replaceAll("=","#");
            }

            public static String decode(String encodeId){
                String encode = encodeId.replaceAll("#", "=");
                encode = new StringBuilder(encode).reverse().toString();
                byte[] idBytes = Base64.decodeBase64(encode.split("==")[1]);
                return new String (idBytes);
            }

            public static void main(String[] args) {
                System.out.println(decode(encode("15465sa465df46a5s4df")));
            }
        }
```

11.3 CookieUtil

```
        public class CookieUtil {

            public static String getCookieValue(HttpServletRequest request,String key){
                Cookie[] cookies = request.getCookies();
                for (int i = 0; i <cookies.length ; i++) {
                    Cookie cookie = cookies[i];
                    if(key.equals(cookie.getName())){
                        try {
                            return URLDecoder.decode(cookie.getValue(),"UTF-8");
                        } catch (UnsupportedEncodingException e) {
                            e.printStackTrace();
                        }
                    }
                }
```

```
            return null;
        }
    }
```

11.4 Md5Util

```
public class Md5Util {
    public static String encode(String pwd) {
        try {
            MessageDigest messageDigest = MessageDigest.getInstance("md5");
            return Base64.encodeBase64String(messageDigest.digest(pwd.getBytes()));
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

11.5 UserLoginUtil

```
public class UserLoginUtil {

    public static String realseUserId(HttpServletRequest request){
        return CookieUtil.getCookieValue(request,"userId");
    }
}
```

12 封装CrmConstant

```
public class CrmConstant {

    public  static  final  Integer OPS_SUCCESS_CODE=200;
    public  static  final  String  OPS_SUCCESS_MSG="操作成功";

    public  static  final  Integer OPS_FAILED_CODE=300;
    public  static  final  String  OPS_FAILED_MSG="操作失败";

    public  static  final  Integer LOGIN_FAILED_CODE=305;
    public  static  final  String  LOGIN_FAILED_MSG="用户登陆失败";
}
```

13 封装异常

```
public class ParamsException  extends  RuntimeException{

    private Integer code;

    private String msg;

    public ParamsException(Integer code,String msg){
```

```
            this.code=code;
            this.msg=msg;
        }

        public Integer getCode() {
            return code;
        }

        public void setCode(Integer code) {
            this.code = code;
        }

        public String getMsg() {
            return msg;
        }

        public void setMsg(String msg) {
            this.msg = msg;
        }
    }
```

14 前台js

```
    function userLogin() {
        var userName=$("#userName").val();
        var userPwd=$("#userPwd").val();
        if(isEmpty(userName)){
            alert("用户名不能为空！");
            return;
        }
        if(isEmpty(userPwd)){
            alert("密码不能为空！");
            return;
        }

        var params={};
        params.userName=userName;
        params.userPwd=userPwd;
        $.ajax({
            type:"post",
            url:ctx + "/user/userLogin",
            data:params,
            dataType:"json",
            success:function (data) {
                if(data.code==200){
                    var result=data.result;
                    $.cookie("userName",result.userName);
                    $.cookie("trueName",result.trueName);
                    $.cookie("userId",result.userId);
                    window.location.href="main";
                }else{
```

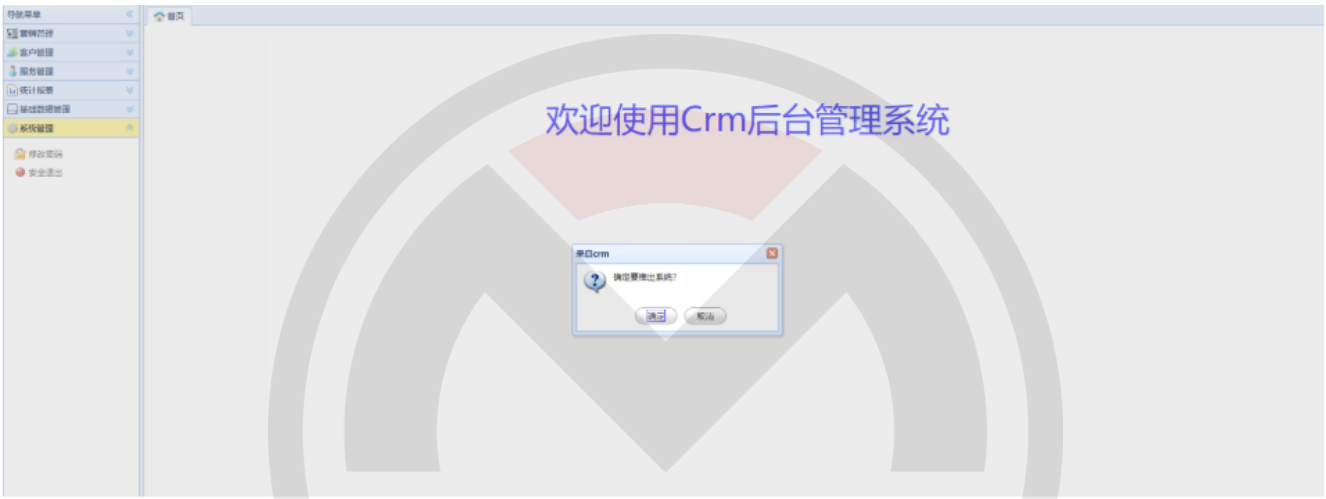
```
                    alert(data.msg);
                }
            }
        })
    }
```

三 退出登陆

1 需求分析

用户点击突出登陆，跳转到登陆页面

2 页面原型



3 前台js

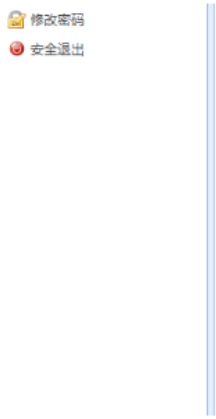
```
function logout() {
    $.messager.confirm("来自crm","确定要推出系统? ",function (r) {
        if(r){
            $.messager.alert("来自crm系统","两秒后推出系统","info");
            setTimeout(function () {
                /*$.cookie("userName","");
                $.cookie("trueName","");*/
                $.removeCookie("userName");
                $.removeCookie("trueName");
                window.location.href="index";
            },2000)
        }
    })
}
```

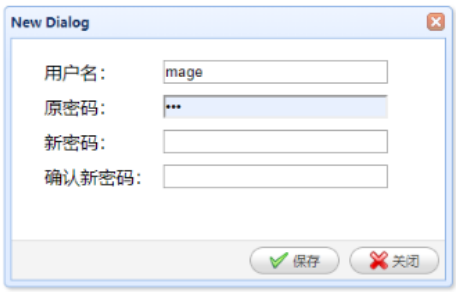
四 密码修改

1 需求分析

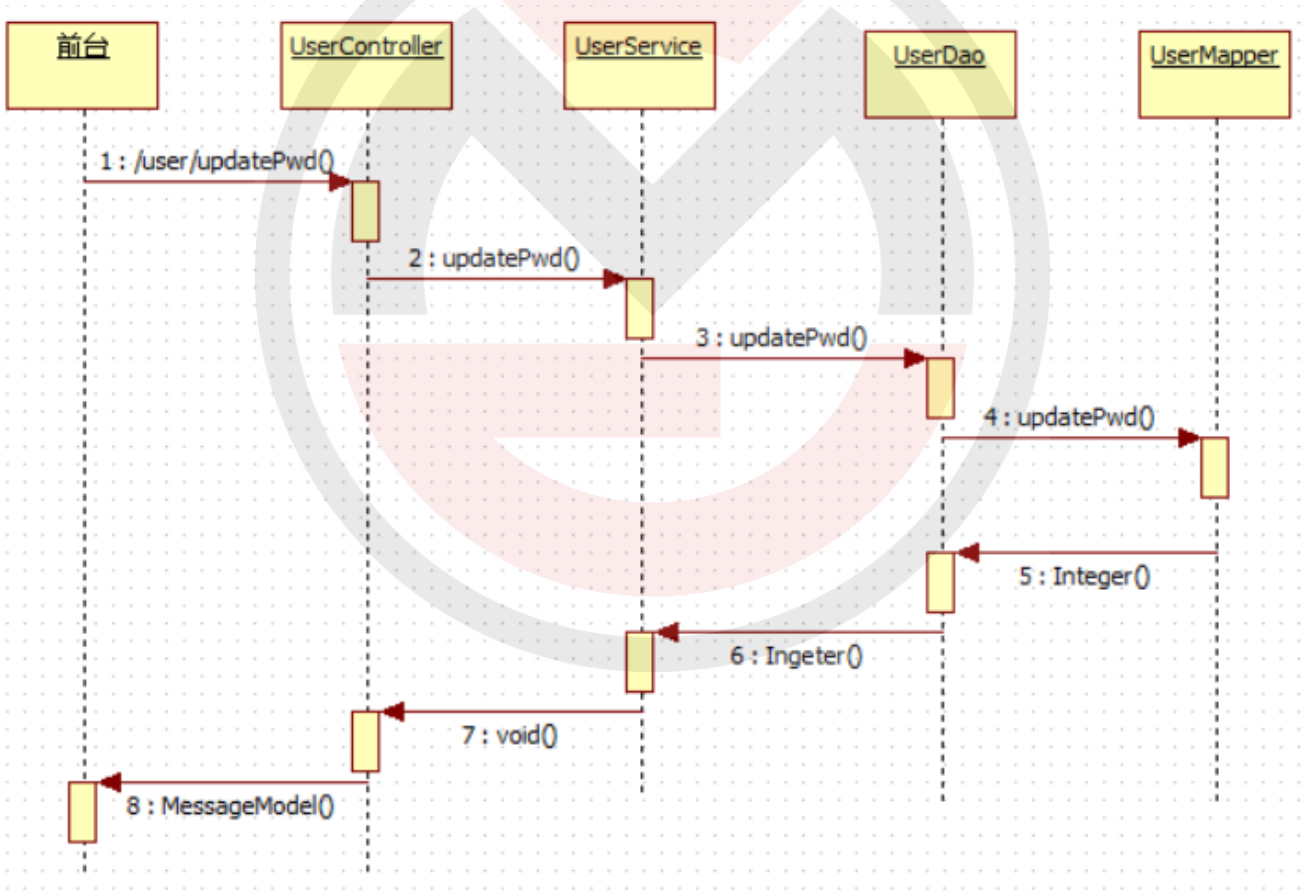
修改当前登陆用户的登陆密码

2 页面原型





3 流程图



4 编写Controller

```
            @ResponseBody
        @RequestMapping("updatePwd")
        public MessageModel updatePwd(HttpServletRequest request, String oldPassword,
String newPassword, String confirmPassword){
            MessageModel messageModel=new  MessageModel();
```

```
                String userId = UserLoginUtil.realseUserId(request);
                try{
                    userService.updatePwd(userId,oldPassword,newPassword,confirmPassword);
                }catch (ParamsException e){
                    e.printStackTrace();
                    messageModel.setCode(e.getCode());
                    messageModel.setMsg(e.getMsg());
                }catch (Exception e){
                    e.printStackTrace();
                    messageModel.setCode(CrmConstant.OPS_FAILED_DODE);
                    messageModel.setMsg(CrmConstant.OPS_FAILED_MSG);
                }
                return messageModel;
            }
```

5 编写Service

```
        /**
         * 用户修改密码
         * @param userId
         * @param oldPassword
         * @param newPassword
         * @param confirmPassword
         */
        public void updatePwd(String userId,String oldPassword,String newPassword,String
confirmPassword){
            AssertUtil.isTrue(null==userId,"非法用户");
            AssertUtil.isTrue(StringUtils.isBlank(newPassword),"新密码不能为空");
            AssertUtil.isTrue(!newPassword.equals(confirmPassword),"两次密码输入不一致");
            User user = userDao.queryUserById(userId);
            AssertUtil.isTrue(null==user,"用户被冻结，不允许修改密码");
            oldPassword = Md5Util.encode(oldPassword);
            AssertUtil.isTrue(!oldPassword.equals(user.getUserPwd()),"原始密码错误");
            newPassword= Md5Util.encode(newPassword);
            AssertUtil.isTrue(userDao.updatePwd(userId,newPassword)<1,"操作失败");
        }
```

6 编写Dao

```
                public User queryUserById(String id);
                public Integer updatePwd(@Param("id") String id, @Param("userPwd") String
userPwd);
```

7 编写Mapper

```
        <sql id="user_columns">
            id , user_name as userName, user_pwd as userPwd,true_name as trueName,
            phone,email,is_valid as isValid,create_date as createDate,update_date as
updateDate
```

```
</sql>
<select id="queryUserById" parameterType="string" resultType="user">

    SELECT <include refid="user_columns"/>
    from t_user where id = #{id} and is_valid=1
</select>

<update id="updatePwd">

  UPDATE t_user set user_pwd=#{userPwd}

  where id =#{id}
</update>
```

8 前台js

```
function modifyPassword() {
    $("#fm").form("submit",{
        url:ctx+"/user/updatePwd",
        onSubmit:function () {
            return $("#fm").form("validate");
        },
        success:function (data) {
            data=JSON.parse(data);
            if (data.code==200){
                $.messager.alert("来自crm系统","修改密码成功，两秒后推出系统","info");
                setTimeout(function () {
                    $.removeCookie("userName");
                    $.removeCookie("trueName");
                    $.removeCookie("userId");
                    window.location.href="index";
                },2000)
            }else{
                $.messager.alert("来自crm系统",data.msg,"info");
            }
        }
    })
}
```

五 总结

了解项目分层

Controller：接收请求，向前台返回数据

Service：在其中实现具体的业务逻辑

Dao/Mapper：与数据库进行交互，负责增删改查

Vo：数据库与项目中实体类的映射

Model：向前台返回我们所需要的自定义的类型

Utils：封装我们需要的公共的方法