

用户信息

一 学习目标



二 用户信息查询

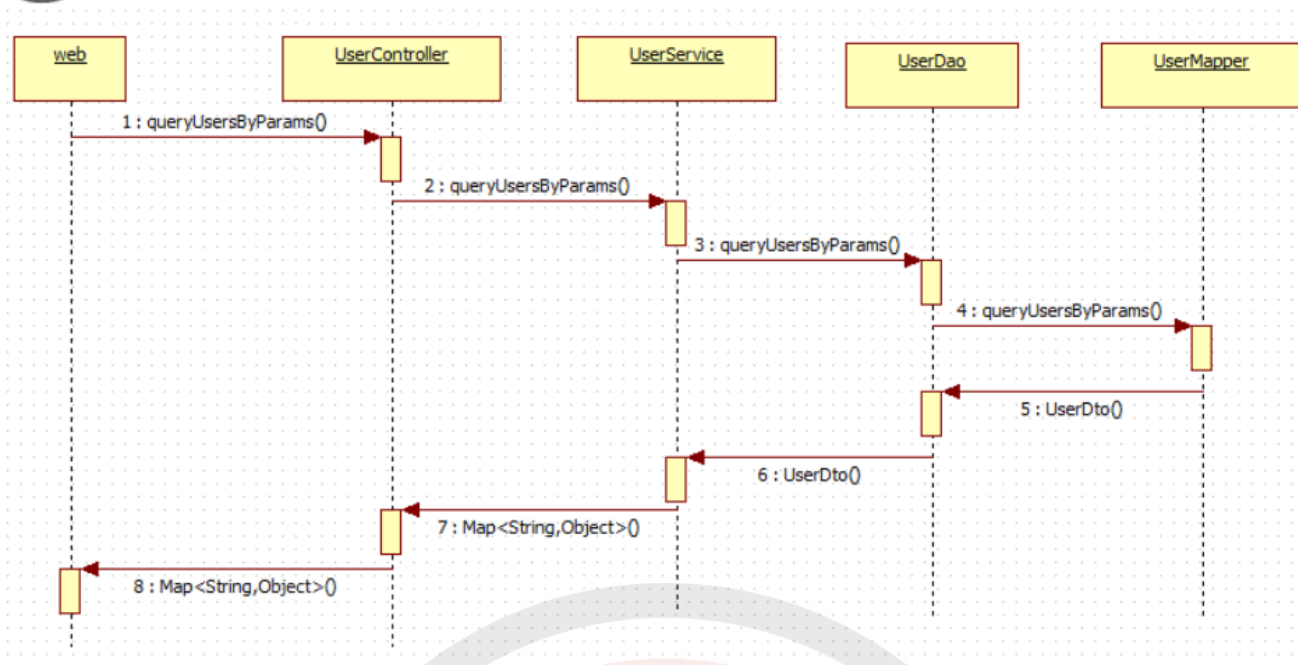
1 需求分析

可以查询所有用户信息

2 页面原型

<div>添加 修改 删除</div> <div>用户名: <input type="text"/> 真实名称: <input type="text"/> 联系方式: <input type="text"/> 邮箱: <input type="text"/> <input type="button" value="查询"/></div>									
	<input type="checkbox"/>	编号	用户名	真实名称	邮箱	手机号	角色	创建时间	更新时间
1	<input type="checkbox"/>	10	whsxt	whsxt	123@163.com	123213	系统管理员	2016-12-01	2018-01-20
2	<input type="checkbox"/>	48	老裴	老裴				2017-10-23	2017-10-24
3	<input type="checkbox"/>	52	shsxt02	shsxt	shsxt@163.com	123456		2018-01-11	2018-01-11
4	<input type="checkbox"/>	53	shsxt03	shsxt	shsxt@163.com	123456		2018-01-11	2018-01-11
5	<input type="checkbox"/>	55	shsxt04	shsxt	shsxt@163.com	23213213		2018-01-12	2018-01-12
6	<input type="checkbox"/>	57	shsxt05	shsxt	shsxt@163.com	2342343		2018-01-12	2018-01-12
7	<input type="checkbox"/>	58	shsxt06	shsxt06	shsxt@163.com	2342343		2018-01-12	2018-01-12
8	<input type="checkbox"/>	63	mage	mage			客户经理		

3 流程图



4 数据库信息

(t_user 用户表)

	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Cntrroller

```
@RequestMapping("index")
public String index(){
    return "user";
}
@RequestMapping("queryUsersByParams")
@ResponseBody
public Map<String, Object> queryUsersByParams(UserQuery userQuery){
    return userService.queryUsersByParams(userQuery);
}
```

6 编写Service

```
public Map<String, Object> queryUsersByParams(UserQuery userQuery) {
    PageHelper.startPage(userQuery.getPage(), userQuery.getRows());
    List<UserDto> list = userDao.queryUsersByParams(userQuery);

    if (list != null && list.size() > 0) {
        for (UserDto userDto : list) {
            String roleIdStr = userDto.getRoleIdsStr();
            if (roleIdStr != null) {
                String[] roleIdArray = roleIdStr.split(",");
                for (int i = 0; i < roleIdArray.length; i++) {
                    List<Integer> roleIds = userDto.getRoleIds();
                    roleIds.add(Integer.parseInt(roleIdArray[i]));
                }
            }
        }
    }

    PageInfo<UserDto> pageInfo = new PageInfo<>(list);
    Map<String, Object> map = new HashMap<>();
    map.put("rows", pageInfo.getList());
    map.put("total", pageInfo.getTotal());
    return map;
}
```

7 编写Dao

```
public List<UserDto> queryUsersByParams(UserQuery userQuery);
```

8 编写Mapper

```
<select id="queryUsersByParams" parameterType="userQuery" resultType="userDto">

    select <include refid="user_columns02"/>,
    GROUP_CONCAT(r.role_name SEPARATOR '-') AS roleName,
    GROUP_CONCAT(r.id ) AS roleIdsStr
    FROM
    t_user u
```



```
LEFT JOIN t_user_role ur ON u.id = ur.user_id
LEFT JOIN t_role r ON ur.role_id = r.id
<where>
    u.is_valid = 1
    <if test="userName !=null and userName!="">
        and u.user_name like concat('%',{userName},%')
    </if>
    <if test="trueName !=null and trueName!="">
        and u.true_name like concat('%',{trueName},%')
    </if>
    <if test="email !=null and email!="">
        and u.email like concat('%',{email},%')
    </if>
    <if test="phone !=null and phone!="">
        and u.phone like concat('%',{phone},%')
    </if>
</where>
GROUP BY u.id

</select>
```

9 编写Vo

```
public class User extends BaseVO{

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

    public List<Integer> roleIds = new ArrayList<>();
    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 List<Integer> getRoleIds() {
    return roleIds;
}

public void setRoleIds(List<Integer> roleIds) {
    this.roleIds = roleIds;
}
}
```

10 编写Query

```
public class UserQuery extends BaseQuery {
    private String userName;
    private String trueName;
    private String email;
    private String phone;
    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;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
}
```

11 编写Dto

```
public class UserDto extends User {

    private String roleName;

    private String roleIdsStr;

    public String getRoleName() {
        return roleName;
    }

    public void setRoleName(String roleName) {
        this.roleName = roleName;
    }

    public String getRoleIdsStr() {
        return roleIdsStr;
    }

    public void setRoleIdsStr(String roleIdsStr) {
        this.roleIdsStr = roleIdsStr;
    }
}
```

12 前台js

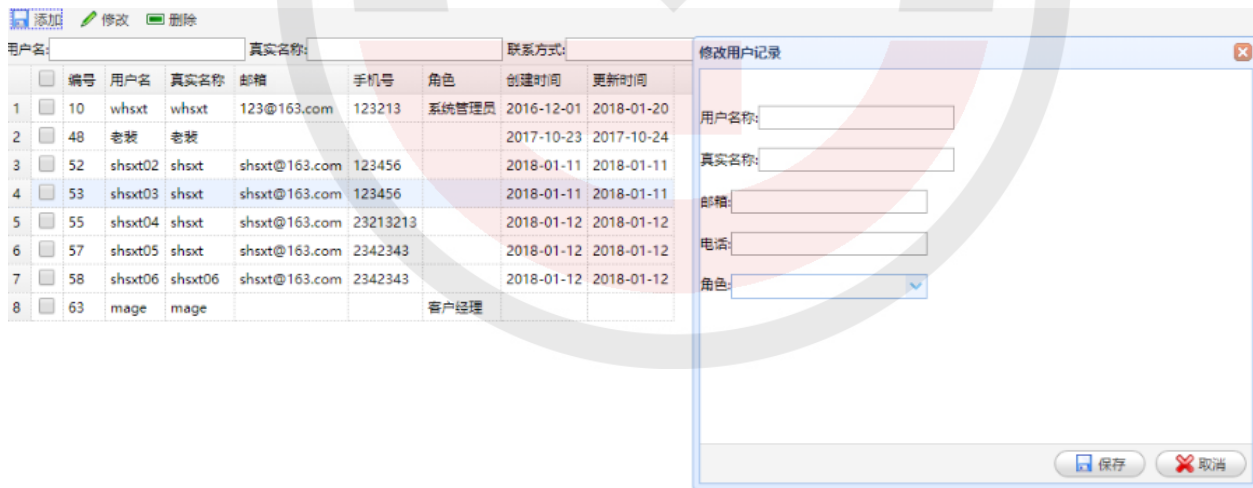
```
$(function () {  
    searchUsers();  
})  
function searchUsers() {  
  
    $("#dg").datagrid("load",{  
        userName:$("#userName").val(),  
        trueName:$("#trueName").val(),  
        phone:$("#phone").val(),  
        email:$("#email").val()  
    });  
}
```

二 添加用户

1 需求分析

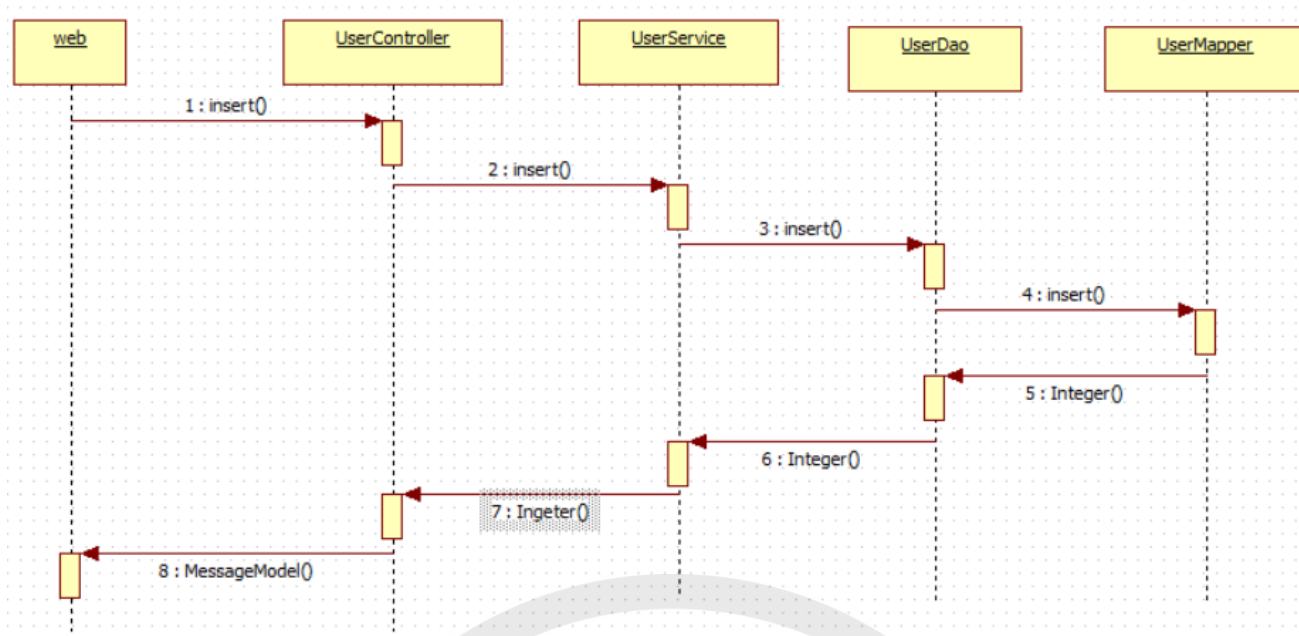
针对于公司新人，对其添加公司内部系统账号

2 页面原型



	编号	用户名	真实名称	邮箱	手机号	角色	创建时间	更新时间
1	10	whsxt	whsxt	123@163.com	123213	系统管理员	2016-12-01	2018-01-20
2	48	老装	老装				2017-10-23	2017-10-24
3	52	shsxt02	shsxt	shsxt@163.com	123456		2018-01-11	2018-01-11
4	53	shsxt03	shsxt	shsxt@163.com	123456		2018-01-11	2018-01-11
5	55	shsxt04	shsxt	shsxt@163.com	23213213		2018-01-12	2018-01-12
6	57	shsxt05	shsxt	shsxt@163.com	2342343		2018-01-12	2018-01-12
7	58	shsxt06	shsxt06	shsxt@163.com	2342343		2018-01-12	2018-01-12
8	63	mage	mage			客户经理		

3 流程图



4 编写Controller

```

@RequestMapping("update")
@ResponseBody
public MessageModel update(User user) {
    userService.update(user);
    return success("用户记录修改成功");
}

```

5 编写Service

```

/**
 * 数据验证
 * 用户名不能为空, 且不能已存在
 * 真实姓名不能为空
 * 电话不能为空
 * 额外数据的补录
 * 密码 : 123456
 * invalid 1
 * createDate
 * updateDate
 *
 * @param user
 */

public void insert(User user) {
    checkParams(user.getUserName(), user.getTrueName(), user.getPhone());
    //判断userName的唯一性
    AssertUtil.isTrue(userDao.queryUserByName(user.getUserName()) != null, "用户名已
    经存在");

    user.setUserPwd(Md5Util.encode("123456"));
    user.setInvalid(1);
}

```



```
user.setCreateDate(new Date());
user.setUpdateDate(new Date());
AssertUtil.isTrue(userDao.insert(user) < 1, "用户数据添加失败");
String userId = user.getId();
List<Integer> roleIds = user.getRoleIds();
if (roleIds != null && roleIds.size() > 0) {
    //级联操作,添加用户角色。
    relateRoles(userId, roleIds);
}
}

public void checkParams(String userName, String trueName, String phone) {
    AssertUtil.isTrue(StringUtils.isBlank(userName), "用户名不能为空");
    AssertUtil.isTrue(StringUtils.isBlank(trueName), "真实姓名不能为空");
    AssertUtil.isTrue(StringUtils.isBlank(phone), "手机号不能为空");
}

public void relateRoles(String userId, List<Integer> roleIds) {

    List<UserRole> list = new ArrayList<>();

    for (Integer roleId : roleIds) {

        UserRole userRole = new UserRole();

        userRole.setIsValid(1);
        userRole.setCreateDate(new Date());
        userRole.setUpdateDate(new Date());
        userRole.setUserId(Integer.parseInt(userId));
        userRole.setRoleId(roleId);
        list.add(userRole);
    }

    AssertUtil.isTrue(userRoleDao.insertBatch(list) < list.size(), "用户角色添加失
败");
}
```

6 编写Dao

```
public Integer insert(User user);
public Integer insertBatch(List<UserRole> userRoles);
```

7 编写Mapper

```
<insert id="insert" parameterType="user" useGeneratedKeys="true" keyProperty="id">
    insert into
t_user(user_name,user_pwd,true_name,email,phone,is_valid,create_date,update_date)
    values(#{userName},#{userPwd},#{trueName},#{email},#{phone},#{isValid},#{
createDate},#{updateDate})
</insert>
<insert id="insertBatch" parameterType="list">
    insert into t_user_role(user_id,role_id,create_date,update_date)
    values
    <foreach collection="list" item="item" separator=",">
        (#{item.userId},#{item.roleId},#{item.createDate},#{item.updateDate})
    </foreach>
</insert>
```

8 前台js

```
function openAddUserDialog() {
    $("#dlg").dialog("open");
}

function closeDialog(){
    $("#dlg").dialog("close");
}

function saveOrUpdateUser() {

    var id=$("#id").val();
    var url=ctx+"/user/insert";
    if(!isEmpty(id)){
        url=ctx+"/user/update";
    }

    $("#fm").form("submit", {
        url:url,

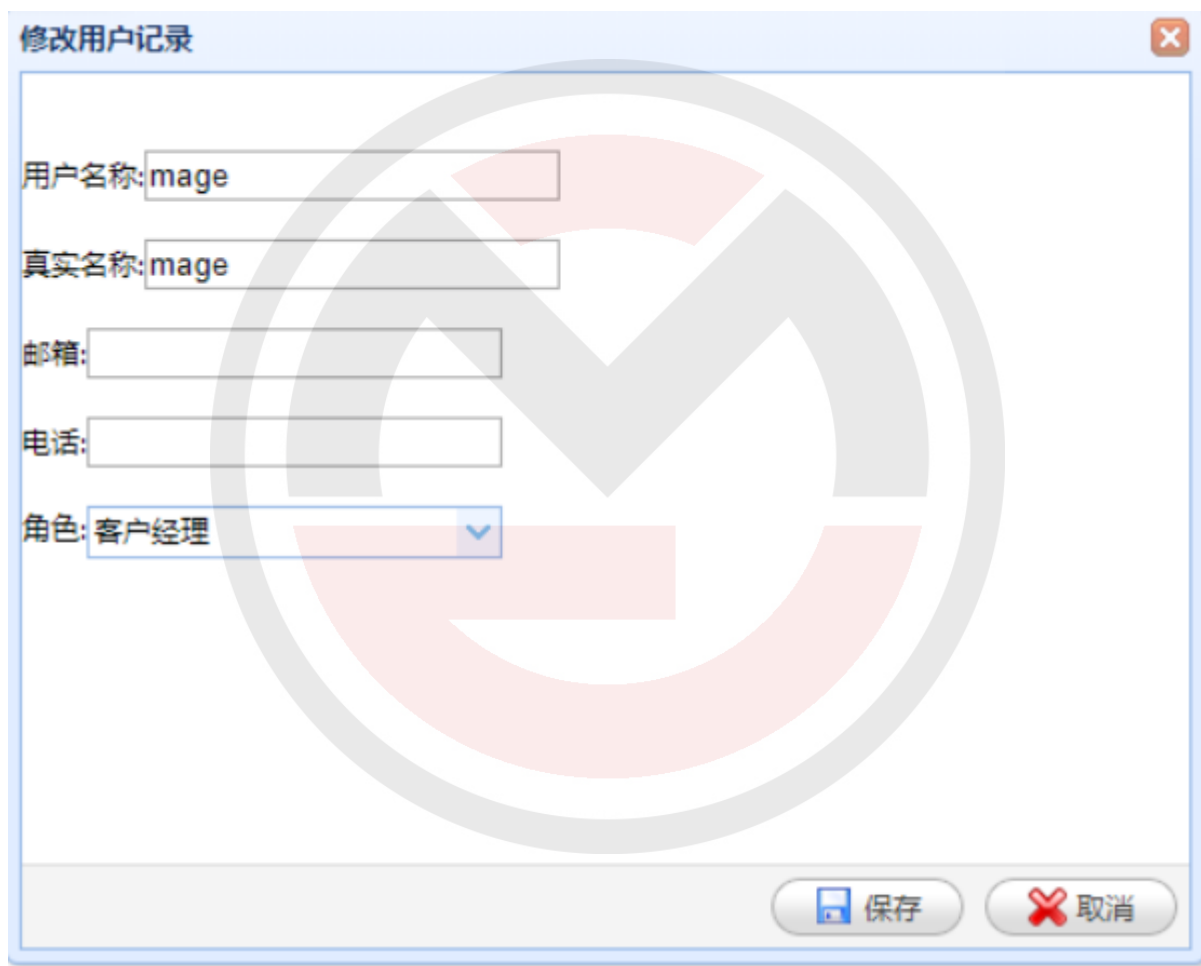
        onSubmit:function () {
            return $("#fm").form("validate");
        },
        success:function (data) {
            data = JSON.parse(data);
            if(data.code==200){
                $.messager.alert("来自crm系统",data.msg,"info")
                $("#fm").form("clear");
                closeDialog();
                searchUsers();
            }else{
                $.messager.alert("来自crm系统",data.msg,"info")
            }
        }
    });
}
```

三 更新用户

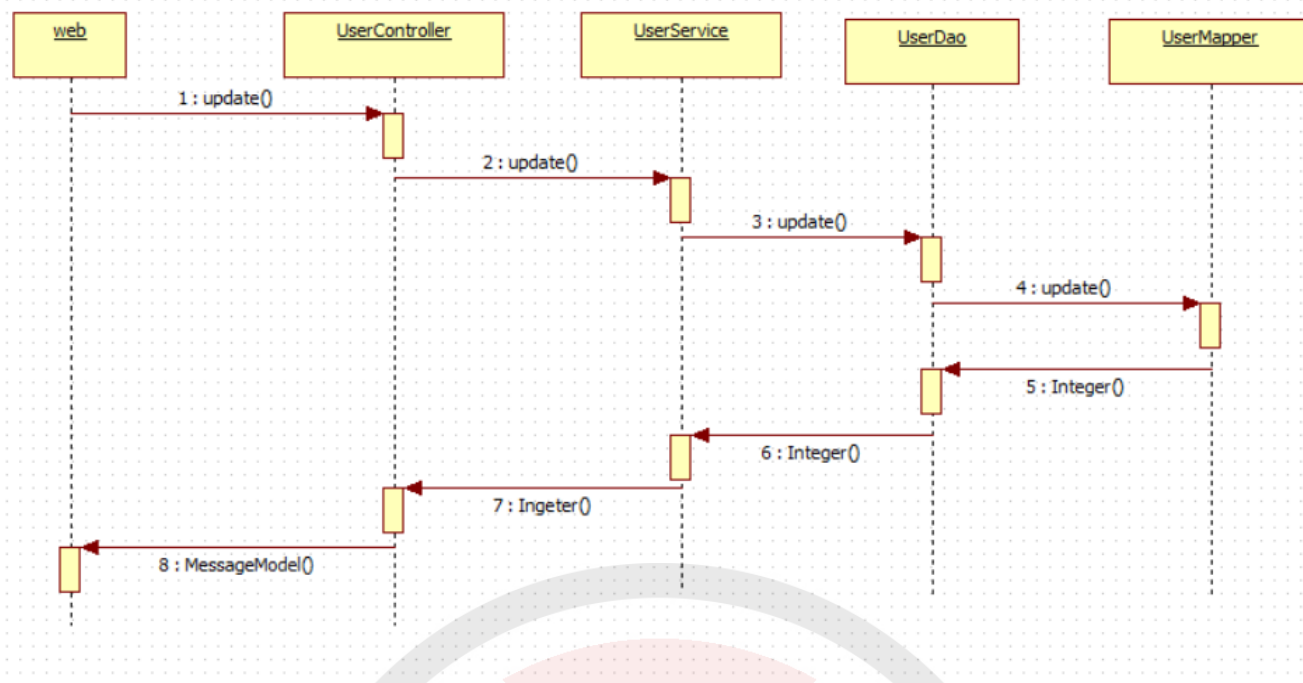
1 需求分析

对已经分配的账户进行修改

2 页面原型



3 流程图



4 编写Controller

```

@RequestMapping("update")
@ResponseBody
public MessageModel update(User user) {
    userService.update(user);
    return success("用户记录修改成功");
}

```

5 编写Service

```

/**
 * 数据验证
 * 用户名不能为空, 且不能已存在
 * 真实姓名不能为空
 * 电话不能为空
 * 额外数据的补录
 * updateDate
 *
 * @param user
 */
public void update(User user) {

    checkParams(user.getUserName(), user.getTrueName(), user.getPhone());

    user.setUpdateDate(new Date());

    User tmpUser = userDao.queryUserByName(user.getUserName());
}

```



```
AssertUtil.isTrue(tmpUser!=null&&!user.getId().equals(tmpUser.getId()),"用户名称已存在");

AssertUtil.isTrue(userDao.update(user) < 1, "用户数据添加失败");

String userId = user.getId();

userRoleDao.deleteUserRolesByUserId(Integer.parseInt(userId));

List<Integer> roleIds = user.getRoleIds();
if (roleIds != null && roleIds.size() > 0) {
    //级联操作,添加用户角色。
    relateRoles(userId, roleIds);
}
}
```

6 编写Dao

```
public Integer deleteUserRolesByUserId(Integer userId);
public Integer update(User user);
```

7 编写Mapper

```
<update id="update" parameterType="user">
    update t_user set user_name=#{userName},true_name=#{trueName},email=#{email},phone=#{phone},
    update_date=#{updateDate}
    where id=#{id} and is_valid=1
</update>
<delete id="deleteUserRolesByUserId" parameterType="int">
    delete from t_user_role
    where user_id=#{userId}
</delete>
```

8 前台js

```
function openModifyUserDialog() {
    var rows= $("#dg").datagrid("getSelections");
    if(rows.length==0){
        $.messager.alert("来自crm","请选中待更新记录!", "info");
        return;
    }

    if(rows.length>1){
        $.messager.alert("来自crm","只能选择一条记录执行更新!", "info");
        return;
    }
}
```

```
$("#fm").form("load",rows[0]);  
$("#dlg").dialog("open").dialog("setTitle","修改用户记录");  
}
```

四 删除用户

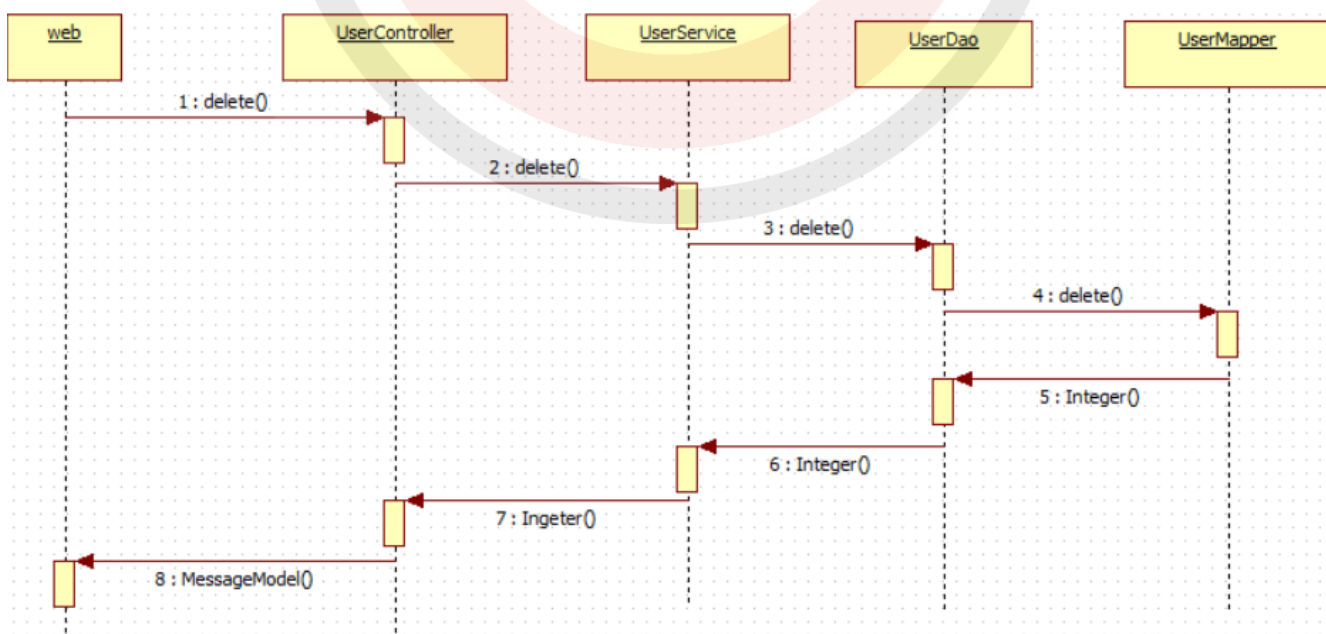
1 需求分析

对已经辞职的账户进行删除

2 页面原型



3 流程图



4 编写Controller



```
@RequestMapping("delete")
@ResponseBody
public MessageModel delete(Integer id){
    userService.delete(id);
    return success("用户记录删除成功");
}
```

5 编写Service

```
public void delete(Integer id){
    AssertUtil.isTrue(userDao.delete(id)<1,"用户数据删除失败");

    int count = userRoleDao.queryUserRoleCountsByUserId(id);

    if(count>0){
        AssertUtil.isTrue(userRoleDao.deleteUserRolesByUserId(id)<count,"用户角色级联
删除失败");
    }
}
```

6 编写Dao

```
@Delete("delete from t_user where id = #{id}")
public Integer delete(Integer id);
public Integer queryUserRoleCountsByUserId(Integer userId);
public Integer deleteUserRolesByUserId(Integer userId);
```

7 编写Mapper

```
<select id="queryUserRoleCountsByUserId" resultType="int">
    select count(1) from t_user_role
    where user_id=#{userId}
</select>

<delete id="deleteUserRolesByUserId" parameterType="int">
    delete from t_user_role
    where user_id=#{userId}
</delete>
```

8 前台js

```
function deleteUser() {
```



```
var rows=$("#dg").datagrid("getSelections");
if(rows.length==0){
    $.messager.alert("来自crm","请选中待删除记录!","info");
    return;
}

if(rows.length>1){
    $.messager.alert("来自crm","不允许同时删除多条记录!","info");
    return;
}

$.messager.confirm("来自crm","确定删除选中的"+rows.length+"条记录?",function(r){
    if(r){
        $.ajax({
            type:"post",
            url:ctx+"/user/delete",
            data:"id="+ rows[0].id,
            dataType:"json",
            success:function(data){
                $.messager.alert("来自crm",data.msg,"info");
                if(data.code==200){
                    closeDialog();
                    searchUsers();
                }
            }
        });
    }
});
}
```