

用户管理

1.用户管理需求

- 1.用户的查询
- 2.高级查询
- 3.用户的添加
- 4.批量删除用户
- 5.单个删除用户
- 6.修改密码

2.页面设计

2.1 设计主页 “user/index.html”



2.2 设计添加页面 “user/add.html”

十

里

里

里

里

里

反

面

面

首页 / 常规管理 / 用户管理

添加用户

用户名 *

请输入用户名

填写用户名称

密 码 *

请输入密码

填写密码

性别 *

☒ 男 ☐ 女

年 龄 *

请输入年龄

邮 箱 *

请输入邮箱地址

格言

请输入个人格言信息

确认保存

2.3 修改密码页面 “user/updatePwd.html”

用户检索

用户姓名

加 删除

ID

10000

10001

10002

10003

操作

修改密码

修改密码

修改密码

修改密码

修改密码

旧的密码 *

请输入旧的密码

填写自己账号的旧的密码。

新的密码 *

请输入新的密码

新的密码 *

请输入新的密码

确认保存

3.后端接口提供

3.1 添加数据源及 Easy Code 插件

3.2 dao 层接口和实现

3.3 service 层接口和实现

接口

```
/**
 * 分页查询
 * @param page
```

```

* @param pageSize
* @param user
* @return
*/

```

```

PageInfo<User> findAll(int page,int pageSize,User user);

```

实现:

```

@Override
public PageInfo<User> findAll(int page, int pageSize, User user) {
    PageHelper.startPage(page,pageSize);
    List<User> userList = userDao.queryAll(user);
    PageInfo<User> userPageInfo = new PageInfo<>(userList);
    return userPageInfo;
}

```

3.4 controller 层接口和实现

```

/**
 * 分页查询 高级查询
 */

@RequestMapping("/queryAll")
public JSONObject<User> queryAll(@RequestParam(defaultValue = "1") Integer
page,
                                @RequestParam(defaultValue = "10") Integer
pageSize,
                                User user){

    //创建返回的对象
    JSONObject object = new JSONObject();

    //查询类型列表信息
    PageInfo<User> pageInfo = userService.findAll(page,pageSize,user);
    //填充返回的前端数据对象
    object.setCode(0);
    object.setMsg("ok");
    object.setCount(pageInfo.getTotal());
    object.setData(pageInfo.getList());

    return object;
}

```

4.前端接口的调用和实现

4.1 分页查询接口调用

```

table.render({
  elem: '#currentTableId',
  url: 'http://127.0.0.1:8888/user/queryAll',
  toolbar: '#toolbarDemo',

  {field: 'username', width: 80, title: '用户名'},
  {field: 'type', width: 80, title: '类型',
    templet: function (res) {
      if (res.type == 0) {
        return '<span class="layui-badge layui-bg-green">普通用户</span>';
      } else {
        return '<span class="layui-badge">管理员</span>';
      }
    }
  },
  {field: 'sex', width: 60, title: '性别'},
  {field: 'age', width: 60, title: '年龄'},
  {field: 'email', width: 180, title: '邮箱'},
  {field: 'personalSay', width: 160, title: '格言'},

```

4.2 高级查询接口调用及实现

页面设计：

```

<div class="layui-inline">
  <label class="layui-form-label">用户姓名</label>
  <div class="layui-input-inline">
    <input type="text" name="username" autocomplete="off" class="layui-input">
  </div>
</div>
<div class="layui-inline">
  <!-- 下拉框 -->
  <label class="layui-form-label">用户类型</label>
  <div class="layui-input-block">
    <select name="type" lay-verify="required" lay-search>
      <option value="">请选择</option>
      <option value="0">普通用户</option>
      <option value="1">管理员</option>
    </select>
  </div>
</div>

```



4.3 监听事件重载 table

```

// 监听搜索操作
form.on('submit(data-search-btn)', function (data) {
  var username = data.field.username;

```

```

var type = data.field.type;

//执行搜索重载
table.reload('currentTableId', {
  page: {
    curr: 1
  }
  , where: {
    username: username,
    type: type
  }
}, 'data');

return false;
});

```

效果图:

5.用户添加

5.1 后端实现代码:

mapper.xml:

```

insert into
tb_user(tb_user.admin,tb_user.username,tb_user.password,tb_user.email,tb_user.
sex,tb_user.age,tb_user.photo,tb_user.reward_code,tb_user.personal_say,tb_user
.last_time,tb_user.type)
values({admin},{username},{password},{email},{sex},{age},{photo},{rewa
rdCode},{personalSay},now(),{type})

```

注意表名, 刚开始表名为 user 可能与 mysql 数据库自带的 user 表冲突一直报错。
 解决方案: user → tb_user

UserController.java

```
@RequestMapping("/add")
public R add(@RequestBody User user) {
    int flag = userService.insert(user);
    if (flag > 0) {
        return R.ok();
    }
    return R.fail("添加用户失败");
}
```

其他层由 EasyCode 插件自己生成。

5.2 前端接口实现

```
// 监听提交
form.on('submit(saveBtn)', function (data) {
    var datas=data.field;
    // 获取信息发送信息到后台服务器
    axios.post('http://localhost:8888/user/add',datas).then(function
(response){
    if(response.code==200){
        layer.msg("添加成功");
        // 刷新主页面 table 一块
        window.reload();
    }
    // 关闭弹出信息
    layer.close(parentIndex);
}).catch(function (error){
    layer.msg(data.msg);
})
return false;
});
```

6.删除用户--批量删除&单个删除

6.1 批量删除

前端接口实现:

```
user/index.html:
else if (obj.event === 'delete') { // 监听删除操作
    var data = table.checkStatus('currentTableId').data;
    if(data.length==0){
        layer.msg("请选择要删除的用户");
    }else{
```

```

        // 获取要删除记录的id 集合
        var ids=getCheckId(data);

        axios.get('http://127.0.0.1:8888/user/deleteByIds?ids='+ids).then(function
        (response){
            if(response.code==200){
                layer.msg("删除成功");
                // 刷新主页面 table 一块
                window.reload();
            }
            // 关闭弹出信息
            // layer.close(parentIndex);
        }).catch(function (error){
            layer.msg(data.msg);
        })
    })
}
}

```

```

/**
 * 获取删除id的集合
 */
function getCheckId(data){
    var arr=new Array();
    for(var i=0;i<data.length;i++){
        arr.push(data[i].id);
    }
    return arr.join(",");
}

```

// 防止表格重复操作

```

147
148     });
149     } else if (obj.event === 'delete') { // 监听删除操作
150         var data = table.checkStatus('currentTableId').data;
151         if(data.length==0){
152             layer.msg("请选择要删除的用户");
153         }else{
154             // 获取要删除记录的id集合
155             var ids=getCheckId(data);
156             axios.get('http://127.0.0.1:8888/user/deleteByIds?ids='+ids).then(function (response){
157                 if(response.code==200){
158                     layer.msg("删除成功");
159                     // 刷新主页面 table 一块
160                     window.reload();
161                 }
162                 // 关闭弹出信息
163                 // layer.close(parentIndex);
164             }).catch(function (error){
165                 layer.msg(data.msg);
166             })
167         }
168     }
169 }
170
171 });

```

后端接口实现

```
@RequestMapping("/deleteByIds")
public R deleteByIds(String ids){ //ids 为传入的参数 /deleteByIds?id=ids
    //将id 字符串 ("1,2,3") 转为集合对象
    List<String> idslist = Arrays.asList(ids.split(","));
    // 遍历集合对象，逐个删除
    for (String id : idslist){
        userService.deleteById(Integer.parseInt(id));
    }
    return R.ok();
}
```

6.2 单个删除

后端代码与批量删除一样，仅前台的监听组件项和接口不一样

```
else if (obj.event === 'delete') {
    layer.confirm('真的删除行么', function (index) {

        axios.get('http://localhost:8888/user/deleteByIds?id=' + data.id).then(function (response){
            if(response.code==200){
                layer.msg("删除成功");
                // 刷新主页面 table 一块
                window.reload();
            }
            //关闭弹出信息
            // layer.close(parentIndex);
        }).catch(function (error){
            layer.msg(data.msg);
        })
        layer.close(index);
    });
});
});
```

7.用户修改密码

步骤：

1. 接收数据
2. 判断两次输入的新密码是否一致
3. 向后台发送请求

4. 提示信息，刷新页面 table

前端

```
/**
 * 设置form信息
 */
function setFormValue(data){
    form.val("updatePwd",{
        id:data.id
    })
}

table.on('tool(currentTableFilter)', function (obj) {
    var data = obj.data;
    if (obj.event === 'edit') { // 监听修改密码操作
        var content = miniPage.getHrefContent('page/user/updatePwd.html');
        // var openWH = miniPage.getOpenWidthHeight();

        var index = layer.open({
            title: '修改密码',
            type: 1,
            shade: 0.2,
            maxmin:true,
            shadeClose: true,
            area: ['50%', '50%'],
            // offset: [openWH[2] + 'px', openWH[3] + 'px'],
            content: content,
        });
        //设置值信息 给id输入框设置值
        setFormValue(data);
    }
});
```

```

<div class="layui-form layuimini-form" lay-filter="updatePwd">
  <input type="hidden" name="id">
  <div class="layui-form-item">
    <label class="layui-form-label required">旧密码</label>
    <div class="layui-input-block">
      <input type="password" name="oldpwd" lay-verify="required" lay-reqtext="旧的密码不能为空" placeholder="请输入旧密码" />
      <tip>填写自己账号的旧的密码。</tip>
    </div>
  </div>

  <div class="layui-form-item">
    <label class="layui-form-label required">新密码</label>
    <div class="layui-input-block">
      <input type="password" name="newpwd" lay-verify="required" lay-reqtext="新的密码不能为空" placeholder="请输入新密码" />
    </div>
  </div>

  <div class="layui-form-item">
    <label class="layui-form-label required">确认密码</label>
    <div class="layui-input-block">
      <input type="password" name="newpwd2" lay-verify="required" lay-reqtext="新的密码不能为空" placeholder="请再次输入" />
    </div>
  </div>

  <div class="layui-form-item">
    <div class="layui-input-block">
      <button class="layui-btn layui-btn-normal" lay-submit lay-filter="updatePwd">确认修改</button>
    </div>
  </div>
</div>

```

//向后台发送请求

```

/**
 * 步骤
 * 1.接收数据
 * 2.判断两次输入的新密码是否一致
 * 3.向后台发送请求
 * 4.提示信息，刷新页面table
 */
var datas = data.field;
if (datas.newpwd != datas.newpwd2){
  layer.msg("两次输入密码不一致，请重新输入密码!");
}else {
  axios.post('http://localhost:8888/user/update',datas).then(function (response){
    if(response.code==200){
      layer.msg("修改成功");
      // 刷新主页面 table一块
      window.reload();
    }
    //关闭弹出信息
    layer.close(parentIndex);
  }).catch(function (error){
    layer.msg(data.msg);
  })
}
}

```

后台

```
@RequestMapping("/update")
public R updatePwd(@RequestBody Map<String,String> map){
    // 获取数据
    String id = map.get("id");
    String oldpwd = map.get("oldpwd");
    String newpwd = map.get("newpwd");
    User user = userService.queryById(Integer.parseInt(id));

    if (user.getPassword().equals(oldpwd)){
        // 重新设置密码
        User u = new User();
        u.setId(Integer.parseInt(id));
        u.setPassword(newpwd);
        userService.update(u);
        return R.ok();
    }else {
        return R.fail("你输入的旧密码不正确...");
    }
}
```