## 【Java框架型项目从入门到装逼】第十四节 查询用户列表展现到页面

这一节,我们来实现一下用户列表搜索,最终的效果如下:

用户列表					
		用户名	密码	姓名	性别
1		zsf	123	张三丰	男
2		ខ្យ	123456	郭靖	男
3		888	123	阿阿阿	男
4		pikaqiu	123	皮卡丘	男
5		a	1	2	男
10	▼	₩ 4 第 1 共1页	▶ н о		显示1到5, 共5记

这边我们要使用easyUI给我们提供的datagrid组件。

#### HTML结构如下

注意,姓名和性别的field值已经变更,和数据库保持一致,这一点和之前的代码是不一样的。至于datagrid的具体用法,因为本系列课程不是专门讲前端框架EasyUI,所以我也就不多阐述了。感兴趣的童鞋可以去参考EasyUI的官方文档。

我们这边看到,在 data-options 中设置了 url 为 getUsers.do。这个代表在刷新页面的时候,就会自动去寻找对应的Controller,然后加载数据。

如图,我们还设置了分页,每页条数为10。

现在我们来看一下控制器部分:

```
@RequestMapping("/getUsers")
@ResponseBody
public Map<String,Object> getUsers(HttpServletRequest request , HttpServletResponse response,Student student){
    System.out.println("开始搜索用户...");

    int page = Integer.parseInt(request.getParameter("page")); //第几页
    int rows = Integer.parseInt(request.getParameter("rows")); //每页多少行
    System.out.println(student);

    Map<String,Object> pageBean = userService.getUsers(student,page,rows);
    return pageBean;
}
```

注意,我们在参数列表中加了一个Student:

```
HttpServletRequest request , HttpServletResponse response Student student) {
.");
```

为什么要添加这个参数呢,那是因为我们最终会通过form表单输入用户名和姓名进行一个模糊查询,所以要在controller层做一个数据绑定。

```
@RequestMapping("/getUsers")
@ResponseBody
public Map<String,Object> getUsers(HttpServletRequest request , HttpServletResponse response,Student student){
    System.out.println("开始搜索用户...");

    int page = Integer.parseInt(request.getParameter("page")); //第几页
    int rows = Integer.parseInt(request.getParameter("rows")); //每页多少行

    System.out.println(student);

    Map<String,Object> pageBean = userService.getUsers(student,page,rows);

    return pageBean;
}
```

再看这两个东西,因为datagrid已经设置了分页,所以加载数据的时候就会默认传过来这两个参数,它会自己告诉controller本次请求要查询的是第几页,每页多少条数据?

```
int page = Integer.parseInt(request.getParameter("page")); //第几页
int rows = Integer.parseInt(request.getParameter("rows")); //每页多少行

System.out.println(student);

Map<String,Object> pageBean = userService.getUsers(student,page,rows);

return pageBean;
```

接下来,我们调用service层的方法,因为是分页查询,所以我们不仅仅要传入需要查询的条件,还要传入分页参数。

最终,我们走到了dao层的方法:

```
@Override
public Map<String, Object> getUsers(Student student, int page, int rows) {
    String sql = "select * from t_student where l=1 ";
    List args = new ArrayList();

    if(student.getId() != 0) {
        sql += " and id = ?";
        args.add(student.getId());
    }

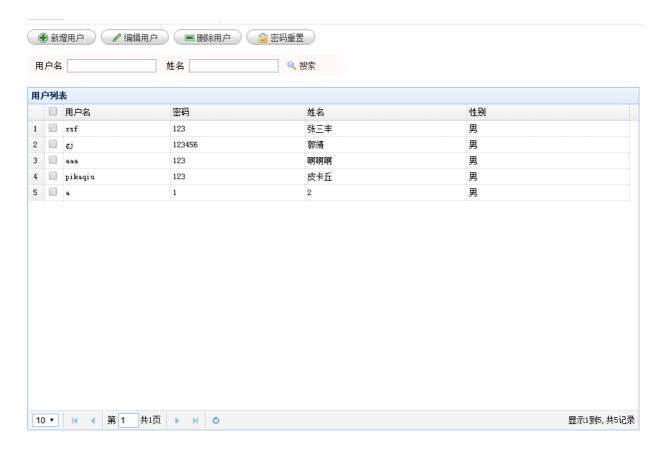
    if(student.getUsername() != null && !"".equals(student.getUsername())) {
        sql += " and username = ?";
        args.add(student.getUsername());
    }

    if(student.getName() != null && !"".equals(student.getName())) {
        sql += " and name = ?";
        args.add(student.getName());
    }

    SimpleDao dao = new SimpleDao();
    Map<String, Object> pageBean = dao.queryForPage(sql, page, rows, args.toArray());
    return pageBean;
}
```

相信只要你有一定的java基础和jdbc基础,就一定能看懂这个方法做了些什么?没错,我们挨个判断查询条件,如果有传过来,就拼接到sql语句中去。最后,用queryForPage方法去查询一个分页对象过来。

现在,我们来测试一下,刷新index.html。



### 可以看到,数据成功的载入了我们的datagrid!

接下来,我们给搜索按钮添加点击事件,达到条件查询的目的:

```
//搜索用户
function searchUser() {
   var username = $('#username_search').val(); //用户名
   var name = $('#nickname_search').val(); //姓名

  $("#gridO").datagrid('load', {
      username : username,
      name : name
   });
}
```

效果(截图不清晰,请下载后观看):



## 我要下载源码

您的支持是我写作的最大动力:

# 打赏不分好坏,一毛也是真爱





支付宝

微信