5、删除

笔记本:	ssm		
创建时间:	2021/7/31 20:03	**更新时间:**	2021/8/1 8:24
作者:	清风渡		
URL:	about:blank		

删除流程

1、单个删除

url：/emp/{id} delete请求

点击删除按钮，将员工id取到，然后通过url传参的方式发送给后端，这个id怎么取，当我们创建这个删除按钮的时候可以加入自定义属性，来保存员工的id值。删除成功同样的是返回删除的页面

```
//删除按钮
var delBtn = $("<button></button>").addClass("btn btn-danger
btn-sm delete_btn")
    .append($("<span></span>").addClass("glyphicon glyphicon-
trash").append("删除"));
// 为删除按钮添加自定义属性来表示当前删除的员工id
delBtn.attr("del-id",item.empId)

// 点击单个删除按钮触发的事件
$(document).on("click",".delete_btn",function (){
    // 弹出是否删除确认框
    var empName =
$(this).parents("tr").find("td:eq(2)").text()
    var empId = $(this).attr("del-id");
    if (confirm("确认删除【"+empName+"】吗？")){
        $.ajax({
            url:"${APP_PATH}/emp/"+empId,
            type:"DELETE",
            success:function (result){
                alert(result.msg)
                to_page(currentPage)
            }
        })
    }

})
```

controller

```
/**
* @Description: 员工删除
* @Author: tao
* @Date: 2021/7/31 21:00
* @param id:
* @return: com.study.bean.Msg
**/
@RequestMapping(value = "/emp/{id}",method =
RequestMethod.DELETE)
@ResponseBody
public Msg deleteEmpById(@PathVariable("id")Integer id){
    employeeService.deleteEmp(id);
    return Msg.success();
}
```

service

```
/**
* @Description: 员工删除
* @Author: tao
* @Date: 2021/7/31 20:29
* @param id:
* @return: void
**/
public void deleteEmp(Integer id) {
    employeeMapper.deleteByPrimaryKey(id);
}
```

2、复选框全选/全不选

当我们选中全选后，点击页码换一页，会发现全选按钮还是选中，但是员工没被选中，所以要在切换页面的时候，将全选按钮改为未选中

```
<tr>
    <th>
        <%-- 一键全选 --%>
        <input type="checkbox" id="check_all">
    </th>
    <th>#</th>
    <th>姓名</th>
    <th>性别</th>
    <th>邮箱</th>
    <th>部门</th>
    <th>操作</th>
</tr>
```

```
//在员工数据的最左边加上多选框
var checkBoxTd = $("<td><input type='checkbox'
class='check_item'/></td>");
var empIdTd = $("<td></td>").append(item.empId)
var empNameTd = $("<td></td>").append(item.empName)
var genderTd = $("<td></td>").append(item.gender == 'M' ?
"男" : "女")
var emailTd = $("<td></td>").append(item.email)
var deptNameTd = $("<td>
</td>").append(item.department.deptName)



// 全选/不全选按钮
$("#check_all").click(function (){
    //   attr获取checked是undefined，因为我们没有定义checked属
性，attr获取的是自定义属性值
    //     而我们这些dom原生的属性，可以用prop来获取这些值
    //     alert($(this).prop("checked"));
    //    让所有复选框状态同步
    // .check_item每个员工的选择框
    $(".check_item").prop("checked",$(this).prop("checked"))
})


// 当本页面所有复选框都选上时，自动将全选复选框选上
$(document).on("click", ".check_item", function () {
    //判断当前选择中的元素是否是当前页面所有check_item的个数
    var flag = $(".check_item:checked").length ==
$(".check_item").length


    $("#check_all").prop("checked",flag)
})


// 指定跳转到哪一页
function to_page(pageNum){
    //每次页面跳转时，都将全选/全不选设置为false
    $("#check_all").prop("checked", false);


    $.ajax({
        url:"${APP_PATH}/emps",
        data:{pageNum},
        type:"get",
        success:function (result) {
            // 解析员工数据
            build_emps_table(result);
            // 解析并显示分页数据
            build_page_info(result.extend.pageInfo);
```

```
            // 解析显示分页条数数据
            build_page_nav(result.extend.pageInfo);
        }
    })
}
```

3、批量删除的实现

- 给删除按钮绑上单击事件，将要删除的员工，即选中的员工的姓名和id循环遍历拼接成字符串，姓名用','分隔，id用'-'分隔。最后要删除多余的','和'-'，也就是最后一个。将要删除的id字符串通过参数传给后端。

```
// 点击删除按钮
$("#emp_delete_all_btn").click(function (){
    // 要删除的名字
    var empNames = "";
    // 要删除的id
    var del_idstr = "";
    $.each($(".check_item:checked"),function () {
        //this 值的是被选中的员工复选框
        empNames +=
$(this).parents("tr").find("td:eq(2)").text()+","
        // 组装员工id字符串
        del_idstr +=
$(this).parents("tr").find("td:eq(1)").text()+"-"
    })
    // 去除多余的,
    empNames = empNames.substring(0,empNames.length-1)
    // 删除id多余的-
    del_idstr = del_idstr.substring(0,del_idstr.length-1)
    if(confirm("确认删除【"+empNames+"】吗？")){
        // 发送ajax请求删除
        $.ajax({
            url:"${APP_PATH}/emp/"+del_idstr,
            type:"DELETE",
            success:function (result) {
                alert(result.msg)
                // 回到当前页面
                to_page(currentPage)
            }
        })
    }
})
```

- 后端接收到了一个参数就是批量删除的字符串，这个字符串存放的是id，以'-'为分隔符分隔的字符串。后端拿到将其转换为数组，再将其数组的每个值转为int型

```
/**
* @Description: 员工删除
* 单个批量二合一
* 单个删除：1
* 多个删除：1-2-3
* @Author: tao
* @Date: 2021/7/31 21:00
* @param ids: 字符串id
* @return: com.study.bean.Msg
**/
@RequestMapping(value = "/emp/{ids}",method =
RequestMethod.DELETE)
@ResponseBody
public Msg deleteEmpById(@PathVariable("ids")String ids){
    // 批量删除
    if(ids.contains("-")){
        // 要删除的id集合
        List<Integer> del_ids = new ArrayList<>();
        String[] str_ids = ids.split("-");
        // 组装id的集合
        for (String id : str_ids) {
            // 将字符串id转为数值型 存到集合中
            del_ids.add(Integer.parseInt(id));
        }
        // 调用批量删除方法
        employeeService.deleteBatch(del_ids);
    }else{
        //单个删除
        Integer id = Integer.parseInt(ids);
        employeeService.deleteEmp(id);
    }
    return Msg.success();
}
```

Service

```
/**
* @Description: 员工删除
* @Author: tao
* @Date: 2021/7/31 20:29
* @param id:
* @return: void
**/
```

```
public void deleteEmp(Integer id) {
    employeeMapper.deleteByPrimaryKey(id);
}

/**
* @Description: 根据id批量删除员工
* @Author: tao
* @Date: 2021/7/31 23:04
* @param ids:
* @return: void
**/
public void deleteBatch(List<Integer> ids) {
    EmployeeExample example = new EmployeeExample();
    EmployeeExample.Criteria criteria =
example.createCriteria();
    // delete from tb_emp where id in (1,2,3...)
    criteria.andEmpIdIn(ids);
    employeeMapper.deleteByExample(example);
}
```