4、修改

笔记本:	ssm		
创建时间:	2021/7/31 12:27	**更新时间:**	2021/7/31 19:03
作者:	清风渡		
URL:	about:blank		

1、大概流程

1、点击编辑
2、弹出用户修改的模态框（显示用户信息）
3、点击更新，完成用户修改

2、点击编辑打开模态框

新增模态框

```
<!-- 员工修改的模态框 -->
<div class="modal fade" id="empUpdateModal" tabindex="-1"
role="dialog" aria-labelledby="myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-
dismiss="modal" aria-label="Close"><span aria-
hidden="true">&times;</span></button>
                <h4 class="modal-title">员工修改</h4>
            </div>
            <div class="modal-body">


                <%--表单-----------------------------------
--------------%>
                <form class="form-horizontal">
                    <%--姓名--%>
                    <div class="form-group">
                        <label for="empName_add_input"
class="col-sm-2 control-label">姓名</label>
                        <div class="col-sm-10">
                            <input type="text" name="empName"
class="form-control" id="empName_update_input"
placeholder="empName">
                            <%--显示错误信息--%>
                            <span class="help-block"></span>
                        </div>
                    </div>
```

```
                    <%--邮箱--%>
                    <div class="form-group">
                        <label
for="email_add_input"  class="col-sm-2 control-label">邮箱
</label>
                        <div class="col-sm-10">
                            <input type="text" name="email"
class="form-control" id="email_update_input"
placeholder="email@qq.com">
                            <span class="help-block"></span>
                        </div>
                    </div>
                    <%--性别--%>
                    <div class="form-group">
                        <label for="gender1_add_input"
class="col-sm-2 control-label">性别</label>
                        <div class="col-sm-10">
                            <label class="radio-inline">
                                <input type="radio"
name="gender" id="gender1_update_input" value="M" checked> 男
                            </label>
                            <label class="radio-inline">
                                <input type="radio"
name="gender" id="gender2_update_input" value="F"> 女
                            </label>
                        </div>
                    </div>


                    <%--部门名--%>
                    <div class="form-group">
                        <label class="col-sm-2 control-
label">部门名</label>
                        <div class="col-sm-4">
                            <%--部门提交部门id就行--%>
                            <select name="dId" class="form-
control">
                            </select>
                        </div>
                    </div>
                </form>
            </div>
            <%-----------------------------------------------
----------%>
            <div class="modal-footer">
                <button type="button" class="btn btn-default"
data-dismiss="modal">关闭</button>
                <button type="button" class="btn btn-primary"
id="emp_update_btn">修改</button>
            </div>
        </div>
    </div>
```

```
</div>
```

给编辑按钮绑上属性update_btn，然后使用jquery提供的on方法实现点击按钮弹出模态框，记得要将查询的部门信息进行清空再查询

```
// 得到部门信息函数
function getDepts(ele) {
    // 清空下拉 列表的值
    $(ele).empty()
    // 发送ajax请求
    $.ajax({
        url:"${APP_PATH}/depts",
        type: "GET",
        success:function (result){
            // {code: 100, msg: "处理成功", extend: {…}}
            // code: 100
            // extend:
            //     depts: (2) [{…}, {…}]
            // 将显示信息添加到下拉部门列表中
            $("#empAddModal select").append()
            $.each(result.extend.depts,function(){
                var optionEle = $("<option>
</option>").append(this.deptName).attr("value",this.deptId)
                optionEle.appendTo(ele)
            })
        }
    })
}

// 点击编辑按钮触发的事件
// 参考jQuery中的on方法，按钮创建之后还是会继续绑定
$(document).on("click",".edit_btn",function (){
    // alert("edit");
    //1、查出员工信息。显示员工信息
    //2、查出部门信息。并且显示部门列表
    getDepts("#empUpdateModal select")
    $("#empUpdateModal").modal({
        backdrop:"static"
    })
})
```

3、修改，回显修改信息

点击编辑按钮，发起两个ajax请求，先查出部门信息，再查出员工信息，部门信息调用之前的部门查询，将部门信息的函数改为同步，因为不改同步的话，后面表单回显的部门信息将会随机显示。也就是加上async:false。然后再查询员工信息，查询员工信息要传当前电机的编辑按钮的员工id，因为后端要根据这个id去数据库查询该员工，那么

这个id怎么来，当我们创建编辑按钮时，给编辑按钮添加属性，属性值为id，作为记号，然后现在取出来传给后端。后端controller层使用路径接收值拿到id

```
// 编辑按钮
var editBtn =$("<button></button>").addClass("btn btn-info
btn-sm edit_btn")
    .append($("<span></span>").addClass("glyphicon glyphicon-
pencil").append("编辑"));
// 为编辑按钮添加一个自定义的属性，来表示当前员工id
editBtn.attr("edit-id",item.empId)


// 得到部门信息函数
function getDepts(ele) {
    // 清空下拉 列表的值
    $(ele).empty()
    // 发送ajax请求
    $.ajax({
        url:"${APP_PATH}/depts",
        async:false,
        type: "GET",
        success:function (result){
            // {code: 100, msg: "处理成功", extend: {…}}
            // code: 100
            // extend:
            //     depts: (2) [{…}, {…}]
            // 将显示信息添加到下拉部门列表中
            $("#empAddModal select").append()
            $.each(result.extend.depts,function(){
                var optionEle = $("<option>
</option>").append(this.deptName).attr("value",this.deptId)
                optionEle.appendTo(ele)
            })
        }
    })
}

// 员工查询
function getEmp(id){
    $.ajax({
        url:"${APP_PATH}/emp/"+id,
        type:"GET",
        success:function (result){
            var empData = result.extend.emp;
            console.log(empData)
            $("#empName_update_static").text(empData.empName)
            $("#email_update_input").val(empData.email)
            $("#empUpdateModal input[name =
gender]").val([empData.gender])
```

```
            $("#empUpdateModal select").val([empData.dId])
        }
    })
}

// 点击编辑按钮触发的事件
// 参考jQuery中的on方法，按钮创建之后还是会继续绑定
$(document).on("click",".edit_btn",function (){
    // alert("edit");

//清除错误属性
// 找到表单下面任何元素，只要具有这两个class属性的就移除属性
$("#empUpdateModal form").find("*").removeClass("has-error
has-success")
$("#empUpdateModal form").find(".help-block").text("")

    //1、查出部门信息。并且显示部门列表
    getDepts("#empUpdateModal select")


    //2、查出员工信息。显示员工信息
    getEmp($(this).attr("edit-id"))
    $("#empUpdateModal").modal({
        backdrop:"static"
    })
})

Controller
/**
* @Description: 根据员工id查询员工对象
* @Author: tao
* @Date: 2021/7/31 15:49
* @param id:
* @return: com.study.bean.Msg
**/
@RequestMapping(value = "/emp/{id}",method =
RequestMethod.GET)
@ResponseBody
/*取路径上的参数信息使用PathVariable 取请求参数使用
RequestParam*/
public Msg getEmp(@PathVariable("id")Integer id){
    Employee employee =  employeeService.getEmp(id);
    return Msg.success().add("emp",employee);
}

Service
/**
* @Description: 根据员工id查询员工
* @Author: tao
* @Date: 2021/7/31 14:49
* @param id:
* @return: com.study.bean.Employee
```

```
**/
public Employee getEmp(Integer id) {
    Employee employee =
employeeMapper.selectByPrimaryKeyWithDept(id);
    return employee;
}
```

4、点击修改按钮，将数据修改

- 首先要前端校验邮箱是否合法
- 邮箱校验完成发送ajax请求保存更新的员工数据。ajax请求使用put请求，需要传递一个参数为要修改的员工id，这个id怎么来的呢？当我们创建页面的员工一行行元素时，我们将id的值传到了各自的编辑按钮上，编辑按钮的这个属性就保存了id值，当我们点击编辑按钮时，我们将这个id取出来又给修改按钮绑上新的属性值来保存这个id，这样点击按钮就可以得到员工id了。
- 有两个方法发送put请求，一个是type为POST，但是要在请求体中加入`"&_method=PUT"`
- 还有一个方法是配置过滤器`FormContentFilter`　这个过滤器支持delete、put方法
- 在controller中配置请求地址的时候注意参数要和Javabean属性一致的话就可以直接将地址栏传递的参数值封装到对象中
- 最后关闭模态框并跳转到修改页

index.jsp

```
// 点击更新 更新员工信息
$("#emp_update_btn").click(function () {
    // 验证邮箱
    // 1、校验邮箱信息
    var email = $("#email_update_input").val();
    var regEmail = /^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]
{2,6})$/
    if(!regEmail.test(email)){
        show_validate_msg("#email_update_input","error","邮箱
格式不正确")
        return false
    }else{
        show_validate_msg("#email_update_input","success","")
    }


    //2、发送ajax请求保存更新的员工数据
    $.ajax({
        url:"${APP_PATH}/emp/"+$(this).attr("edit-id"),
```

```
        // 方法一
        // type:"POST",
        // data:$("#empUpdateModal
form").serialize()+"&_method=PUT",
        // 方法二
        type:"PUT",
        data:$("#empUpdateModal form").serialize(),
        success:function (result){
            // alert(result.msg)
            // 关闭对话框
            $("#empUpdateModal").modal("hide")
            // 跳转到修改页
            to_page(currentPage);
        }
    })
});
```

controller

```
/**
* 如果ajax直接发送PUT请求，我们封装的数据为：
* Employee
*      {empId=1036, empName='null', gender='null',
email='null', dId=null, department=null}
* 而在请求头中是有数据的，
email=ast12123%40nn.com&gender=M&dId=1
*
* 问题就在于请求体中有数据，但是Employee对象封装不上
* 这样的话SQL语句就变成 update tbl_employee where emp_id =
1014 ，没有set字段，所以sql语法就有问题
*
* 而封装不上的原因在于
* Tomcat:
*      tomcat会将请求体中的数据封装为一个map，
request.getParameter("empName")就会从这个map中取值
*      而SpringMVC封装POJO对象的时候，会把POJO中每个属性的值调用
request.getParameter()方法来获取
*
*      但是如果AJAX发送PUT请求，tomcat看到是PUT请求，就不会将请求
体中的数据封装为map，
*      只有POST请求才会封装请求体为map
*
*解决方案：
* 我们要能支持直接发送PUT之类的请求，还要封装请求体中的数据
*      在web.xml中配置上FormContentFilter过滤器
*      他的作用是将请求体中的数据解析包装成map
*      request被重新包装，request.getParameter()被重写，就会从
自己封装的map中取出数据
*
* 员工更新方法
```

```
* @param employee
* @return
*/
/*这里的empId和Javabean里面的属性名一致，就会将这个地址栏传递过来的参数字自动封装到这个员工对象中，
没有这个id就不能根据id来修改员工信息了*/
@RequestMapping(value = "/emp/{empId}",method =
RequestMethod.PUT)
@ResponseBody
public Msg updateEmp(Employee employee){
    employeeService.updateEmp(employee);
    return Msg.success();
}
```

web.xml

```
<!--解决ajax发送put请求取不到值-->
<filter>
  <filter-name>formContentFilter</filter-name>
  <filter-
class>org.springframework.web.filter.FormContentFilter</filter-
class>
</filter>
<filter-mapping>
  <filter-name>formContentFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Service

```
/**
* @Description: 员工更新
* @Author: tao
* @Date: 2021/7/31 16:33
* @param employee:
* @return: void
**/
public void updateEmp(Employee employee) {
    employeeMapper.updateByPrimaryKeySelective(employee);
}
```