3、新增

笔记本:	ssm		
创建时间:	2021/7/29 13:44	**更新时间:**	2021/7/31 11:36
作者:	清风渡		
URL:	about:blank		

1、总体流程

1、在index.jsp页面点击新增
2、弹出新增对话框
3、去数据库查询部门列表，显示在对话框中
4、用户输入数据，并进行校验
jQuery前端校验
ajax用户名重复校验
重要数据（后端校验（JSR303）,唯一约束）
5、完成保存

2、新增模态框 https://v3.bootcss.com/javascript/#modals-examples

- jsp 代码

```
<!-- 员工添加的模态框 -->
<div class="modal fade" id="empAddModal" 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" id="myModalLabel">员工添加</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_add_input" placeholder="empName">
                        </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_add_input" placeholder="email@qq.com">
                        </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_add_input" value="M" checked> 男
                            </label>
                            <label class="radio-inline">
                                <input type="radio" name="gender"
id="gender2_add_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"
id="deptName_add_input">
                            </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">保存
</button>
            </div>
        </div>
    </div>
</div>
```

- js代码，点击新增按钮时，打开模态框之前要发起请求，请求部门信息，然后将部门信息拼接到选择框中

```
// 点击新增按钮 弹出模态框
```

```
$("#emp_add_modal_btn").click(function (){
    // 发送ajax请求，查出部门信息，显示在下拉列表中
    getDepts()
    //打开模态框的方法
    $("#empAddModal").modal({
        backdrop:"static"
    })
})


// 得到部门信息函数
function getDepts() {
    // 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("#empAddModal select")
            })
        }
    })
```

3、新增员工，保存表单数据到数据库中

- 表单中的每个输入框的name属性值要和Javabean属性名一致，这样SpringMvc解析页面传过来的数据时会将数据封装为一个对象
- service层新增添加员工信息方法

```
public void saveEmp(Employee employee) {
    employeeMapper.insertSelective(employee);
}
```

- controller层添加新的请求emp,post 请求为添加员工

```
/**
* @Description:
* @Author: tao
* @Date: 2021/7/29 19:37
* @param employee:  只要前端页面表单中的输入框name属性值和我们javabean的属性名一致，
```

```
*                  这里会将前端页面的表单中的数据直接封装为员工对象
* @return: com.study.bean.Msg
**/
@RequestMapping(value = "/emp",method = RequestMethod.POST)
@ResponseBody
public Msg saveEmp(Employee employee){
    employeeService.saveEmp(employee);
    return Msg.success();
}
```

- 给保存按钮绑上单击事件，点击发送ajax请求请求添加员工，注意这个地方发起请求的时候会将页面的表单数据封装在data对象中，jquery有一个方法可以序列化表单数据为字符串，所以这里使用了这个$("表单").serialize()方法。当我们员工添加成功以后需要将模态框关闭，bootstrap有提供好的方法。然后还要将展示员工页面跳转到最后一页，这里插件的作用很强大，我们可以定义一个全局变量，在获取记录数的时候将总记录数保存下来，然后调用to_page()方法将总记录数作为参数传递进去，这个插件定位强大之处就是当你配置上了参数合理化，当你访问页数比总页数更大的时候，他会自动帮你找最后一页，所以我们可以将总记录数+1传递给这个函数，就会跳到最后一页了。

```
$("#emp_save_btn").click(function (){
    // 1、将模态框的表单数据提交给服务器进行保存
    // 2、发送ajax请求保存员工
    // serialize能将表格中的数据序列化为字符串，用于ajax请求
    $("#empAddModal form").serialize()
    $.ajax({
        url:'${APP_PATH}/emp',
        type:'POST',
        data: $("#empAddModal form").serialize(),
        success:function (result){


            // alert(result.msg)
            // 员工保存成功
            //1、关闭模态框
            $('#empAddModal').modal('hide')
            //2、来到最后一页，显示刚才保存的数据
            // 分页插件他会把大于总页码的页码查出总是最后一页的数据
            to_page(totalRecord+1)
        }


    })
```

4、校验功能

- jQuery前端校验（正则表达式）
- 校验信息的显示优化

```
// 显示校验信息函数
function show_validate_msg(ele,status,msg){
    // 每次都要清除当前元素校验的状态
    $(ele).parent().removeClass("has-success has-error")
    $(ele).next("span").text("")
    if("success" == status){
        $(ele).parent().addClass("has-success")
        $(ele).next("span").text(msg)
    }else if("error" == status){
        $(ele).parent().addClass("has-error")
        $(ele).next("span").text(msg)
    }
}


// 校验邮箱姓名函数
function validate_add_form(){
    // 清空
    //1、拿到要校验的数据，使用正则表达式
    var empName = $("#empName_add_input").val();
    var regName = /(^[a-zA-Z0-9_-]{6,16}$)|(^[\u2E80-\u9FFF]{2,5})/;
    if(!regName.test(empName)){
        // alert("姓名格式不正确")
        show_validate_msg("#empName_add_input","error","用户名可以是2-5位
中文或者6-16位英文和数字的组合")
        // $("#empName_add_input").parent().addClass("has-error")
        // $("#empName_add_input").next("span").text("用户名可以是2-5位中
文或者6-16位英文和数字的组合")
        return false
    }else{
        show_validate_msg("#empName_add_input","success","")
        // $("#empName_add_input").parent().addClass("has-success")
        // $("#empName_add_input").next("span").text("")


    }


    // 2、校验邮箱信息
    var email = $("#email_add_input").val();
    var regEmail = /^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/
    if(!regEmail.test(email)){
        show_validate_msg("#email_add_input","error","邮箱格式不正确")
        return false
    }else{
        show_validate_msg("#email_add_input","success","")
    }


    return true
}
```

- 后端校验，用户名是否存在，如果用户名存在，不能再继续添加。给姓名输入框绑上内容该改变事件，内容改变发起ajax请求检测用户名是否存在，存在调

用显示校验信息函数来通知用户。再给保存按钮添加属性，失败 和成功的属性值不同，再在点击保存按钮的单击事件中根据成功属性还是失败属性来判断程序是否继续执行。

```
// 用户名输入框内容改变监听 用户名是否存在
$("#empName_add_input").change(function (){
    var empName = this.value;
    $.ajax({
        url:"${APP_PATH}/checkUser",
        data:{empName},
        type:"GET",
        success:function (result){
            if(result.code == 100){
                show_validate_msg("#empName_add_input","success","用户名
可用")
                // 将按钮开放
                $("#emp_save_btn").attr("disabled",false)
                // 给保存按钮添加属性
                $("#emp_save_btn").attr("ajax-va","success")
            }else{
                show_validate_msg("#empName_add_input","error","用户名不
可用")
                // 将按钮禁用
                $("#emp_save_btn").attr("disabled",true)
                $("#emp_save_btn").attr("ajax-va","error")


            }
        }
    })
})


// 保存员工按钮点击
$("#emp_save_btn").click(function (){
    // 1、将模态框的表单数据提交给服务器进行保存
    // 1、先校验数据
    if(!validate_add_form()){return false}


    // 1、判断之前的ajax用户名校验是否成功。如果成功。
    if($(this).attr("ajax-va")=="error"){
        return false;
    }
    // 2、发送ajax请求保存员工
    // serialize能将表格中的数据序列化为字符串，用于ajax请求
    $("#empAddModal form").serialize()
    $.ajax({
        url:'${APP_PATH}/emp',
        type:'POST',
        data: $("#empAddModal form").serialize(),
        success:function (result){


            // alert(result.msg)
            // 员工保存成功
```

```
            //1、关闭模态框
            $('#empAddModal').modal('hide')
            //2、来到最后一页，显示刚才保存的数据
            // 分页插件他会把大于总页码的页码查出总是最后一页的数据
            to_page(totalRecord+1)
        }


    })
})
```

- 因为先进行后端校验，当我们输入的姓名不符合前端正则校验时，会出现“用户名可用”，然后点击保存按钮又会出现“用户名只能为....”，所以要在后端校验用户名是否存在之前，还要做一个和前端一样的正则检验。
- 同时，当我们点击新增按钮时，还要将之前的输入框的样式以及检验信息内容清空，不单单是将表单重置哦。

```
@RequestMapping("/checkUser")
@ResponseBody
public Msg checkUser(@RequestParam("empName")String empName){
    String reg = "(^[a-zA-Z0-9_-]{6,16}$)|(^[\\u2E80-\\u9FFF]{2,5})";
    if(!empName.matches(reg)){
        return Msg.fail().add("va_msg","用户名必须是6-16为字母组合或者2-5
位中文");
    }
    // 数据库用户名重复校验
    boolean flag = employeeService.checkUser(empName);
    if(flag){
        return Msg.success();
    }else{
        return Msg.fail().add("va_msg","用户名存在");
    }
}

// 重置表单函数
// ele 表单元素
function reset_form(ele){
    //重置表单内容
    $(ele)[0].reset()
    // 清空表单样式以及校验信息内容
    // 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
    $(ele).find("*").removeClass("has-error has-success")
    $(ele).find(".help-block").text("")
}



// 点击新增按钮 弹出模态框
$("#emp_add_modal_btn").click(function (){
    // 清除表单数据 重置表单 (表单的数据以及样式)
    //为啥要加[0]，因为jquery没有重置方法，dom对象有
    reset_form("#empAddModal form")
```

```
    // 发送ajax请求，查出部门信息，显示在下拉列表中
    getDepts()
    $("#empAddModal").modal({
        backdrop:"static"
    })
})
```

- **重要数据（后端校验（JSR303）,唯一约束）**

前端校验只能防君子不能防小人，需要进行后端校验

导入依赖，在实体类进行注解标识，在控制层拿到判断错误信息，有错误信息返回到页面，在页面调用信息校验函数进行页面显示。

```
<!--   JSR303数据校验支持
            注意：如果是tmocat7以下的服务器，EL表达式功能不强大，需要给lib包中替换新的标准的EL-->
<!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-validator -->
<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-validator</artifactId>
  <version>5.4.1.Final</version>
</dependency>
```

Employee

```
    @Pattern(regexp = "(^[a-zA-Z0-9_-]{6,16}$)|(^[\\u2E80-\\u9FFF]{2,5})",
            message = "用户名必须是2-5位中文或者6-16位英文和数字的组合")
    private String empName;


    private String gender;


//    @Email
    @Pattern(regexp = "^([a-z0-9_\\.-]+)@([\\da-z\\.-]+)\\.([a-z\\.]{2,6})$",
            message = "邮箱格式不正确")
```

EmployeeController

```
/**
* @Description:员工保存
* 要加入依赖 hibernate-validator
* 支持JSR303校验
* @Author: tao
* @Date: 2021/7/29 19:37
```

```
* @param employee:  只要前端页面表单中的输入框name属性值和我们javabean的属性名一致，
*                  这里会将前端页面的表单中的数据直接封装为员工对象
* @return: com.study.bean.Msg
**/
@RequestMapping(value = "/emp",method = RequestMethod.POST)
@ResponseBody
public Msg saveEmp(@Valid Employee employee, BindingResult result){
    if(result.hasErrors()){
        // 校验失败 返回失败 在模态框显示失败的错误信息
        Map<String,Object> map = new HashMap<>();
        List<FieldError> fieldErrors = result.getFieldErrors();
        for (FieldError fieldError : fieldErrors) {
            System.out.println("错误的字段名 = " +
fieldError.getField());
            System.out.println("错误信息 = " +
fieldError.getDefaultMessage());
            map.put(fieldError.getField(),
fieldError.getDefaultMessage());
        }
        return Msg.fail().add("errorFields",map);
    }else{
        employeeService.saveEmp(employee);
        return Msg.success();
    }


}

// 用户名输入框内容改变监听 用户名是否存在
$("#empName_add_input").change(function (){
    var empName = this.value;
    $.ajax({
        url:"${APP_PATH}/checkUser",
        data:{empName},
        type:"GET",
        success:function (result){
            if(result.code == 100){
                show_validate_msg("#empName_add_input","success","用户名
可用")
                // 将按钮开放
                $("#emp_save_btn").attr("disabled",false)
                // 给保存按钮添加属性
                $("#emp_save_btn").attr("ajax-va","success")
            }else{
                show_validate_msg("#empName_add_input","error",result.extend.va_msg)
                // 将按钮禁用
                $("#emp_save_btn").attr("disabled",true)
                $("#emp_save_btn").attr("ajax-va","error")
            }
        }
    })
})
```