2、查询

笔记本:	ssm		
创建时间:	2021/7/27 14:20	**更新时间:**	2021/7/29 13:43
作者:	清风渡		
URL:	about:blank		

1、首页跳转，进入首页发起请求，查询出所有员工，在index.jsp页面做出修改

```
<jsp:forward page="/emps"></jsp:forward>
```

2、处理请求，编写service层，注入dao层，这里为了方便就没有使用接口。编写查询所有员工的业务逻辑处理。

```
@Service
public class EmployeeService {


    @Autowired
    EmployeeMapper employeeMapper;


    // 查询所有员工
    public List<Employee> getAllEmployees(){
        List<Employee> employees = employeeMapper.selectByExampleWithDept(null);
        return employees;
    }
}
```

3、编写controller层，处理请求，注入service层。并将查询到的所有员工使用分页插件进行分页处理。

引入分页插件依赖 https://pagehelper.github.io/docs/howtouse/#1-%E5%BC%95%E5%85%A5%E5%88%86%E9%A1%B5%E6%8F%92%E4%BB%B6

```
<!--分页插件-->
<dependency>
  <groupId>com.github.pagehelper</groupId>
  <artifactId>pagehelper</artifactId>
  <version>5.0.0</version>
</dependency>

@Controller
public class EmployeeController {


    @Autowired
    EmployeeService employeeService;
```

```
    /**
     * @Description: 查询员工数据（分页）
     * @Author: tao
     * @Date: 2021/7/27 14:39
     * @return: java.lang.String
     **/
    @RequestMapping("/emps")
    public String getEmps(@RequestParam(value = "pageNum",defaultValue =
"1")Integer pageNum,
                          Model model){


        // 引入PageHelper分页插件
        // 在查询之前调用，并传入参数(当前页码，每一页的容量(这里配置的5))
        PageHelper.startPage(pageNum, 5);
        // 紧跟着startPage后面的查询方法就是一个分页查询
        List<Employee> allEmployees = employeeService.getAllEmployees();
        // 使用PageInfo保证查询后的结果，只要将PageInfo的实例化对象交给页面就行了
        // 封装了详细的分页信息，包括有我们的查询出来的数据,传入连续显示的页数
        PageInfo page = new PageInfo(allEmployees,5);
        model.addAttribute("pageInfo",page);
        return "list";
    }
}
```

4、使用单元测试 对分页查询进行测试 测请求测试

```
package com.study.test;


import com.github.pagehelper.PageInfo;
import com.study.bean.Employee;
import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mock.web.MockHttpServletRequest;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MockMvcBuilder;
import org.springframework.test.web.servlet.MvcResult;
import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
import org.springframework.test.web.servlet.setup.MockMvcBuilders;
import org.springframework.web.context.WebApplicationContext;


import java.util.List;


/**
* @author tao
* @create 2021-07-27 16:23
* @Description 使用Spring测试模块提供的测试请求功能，测试CRUD请求的准确性
*/


/*spring单元测试类*/
// Spring4测试的时候，需要servlet3.0的支持
@RunWith(SpringJUnit4ClassRunner.class)
@WebAppConfiguration
```

```
/*读取配置文件*/
@ContextConfiguration(locations =
{"classpath:applicationContext.xml","file:src/main/webapp/WEB-
INF/dispatcherServlet-servlet.xml"})
public class MvcTest {


    //传入springmvc的IOC
    @Autowired
    WebApplicationContext context;


    // 虚拟MVC请求，获取到处理结果
    MockMvc mockMvc;


    @Before
    public void initMockMvc(){
       mockMvc = MockMvcBuilders.webAppContextSetup(context).build();
    }


    @Test
    public void testPage() throws Exception {
        // 模拟请求拿到返回值
        MvcResult result =
mockMvc.perform(MockMvcRequestBuilders.get("/emps").param("pageNum", "5"))
                .andReturn();


        // 请求成功以后，请求域中会有pageInfo,我们可以取出来验证
        MockHttpServletRequest request = result.getRequest();
        PageInfo pi  =  (PageInfo) request.getAttribute("pageInfo");
        System.out.println("当前页码 = " + pi.getPageNum());
        System.out.println("总页码 = " + pi.getPages());
        System.out.println("总记录数 = " + pi.getTotal());
        System.out.println("在页面需要连续显示的页码");
        int[] navigatePageNums = pi.getNavigatepageNums();
        for(int i : navigatePageNums){
            System.out.println(" " + i);
        }


        // 获取员工数据
        List<Employee> list = pi.getList();
        for (Employee employee : list) {
            System.out.println("ID = " +
employee.getEmpId()+"==>NAME: "+employee.getEmpName());
        }
    }


}
```

5、搭建BootStrap分页页面

- 引入jsp依赖

```
<!--    jsp-api   -->
<dependency>
```

```
    <groupId>javax.servlet</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.0</version>
</dependency>
```

- 编写jsp页面，因为我们请求的员工信息存放在request域中，叫pageInfo

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%--
  Created by IntelliJ IDEA.
  User: hu tao
  Date: 2021/7/27
  Time: 14:38
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>员工列表</title>
    <%
        /*将项目名取出来*/
        pageContext.setAttribute("APP_PATH",request.getContextPath());
    %>
    <%--web路径：
        不以/开头的相对路径，找资源，以当前的资源路径为基准，经常出问题
        以/开头的相对路径，找资源，以服务器的路径为标准(https://localhosst:8080);需要
加上项目名
            https://localhosst:8080/crud
    --%>
    <script type="text/javascript" src="${APP_PATH}/static/js/jquery-
1.12.4.min.js"></script>
    <%--引入样式--%>
    <link rel="stylesheet" href="${APP_PATH}/static/bootstrap-3.4.1-
dist/css/bootstrap.min.css">
    <script src="${APP_PATH}/static/bootstrap-3.4.1-dist/js/bootstrap.min.js">
</script>
</head>
<body>
    <%--搭建页面--%>
    <div class="container">
        <%--标题--%>
        <div class="row">
            <div class="col-md-12">
                <h1>SSM-CRUD</h1>
            </div>
        </div>
        <%--按钮--%>
        <div class="row">
            <div class="col-md-4 col-md-offset-8">
                <button class="btn btn-primary">新增</button>
                <button class="btn btn-danger">删除</button>
            </div>
        </div>
        <%--显示表格数据--%>
        <div class="row">
            <div class="col-md-12">
                <div class="table-responsive">
                    <table class="table table-hover">
                        <thead>
                            <tr>
```

```
                                <th>#</th>
                                <th>姓名</th>
                                <th>性别</th>
                                <th>邮箱</th>
                                <th>部门</th>
                                <th>操作</th>
                            </tr>
                        </thead>
                        <tbody>
                            <c:forEach items="${pageInfo.list}" var="emp">
                                <tr>
                                    <td>${emp.empId}</td>
                                    <td>${emp.empName}</td>
                                    <td>${emp.gender=="M"?"男":"女"}</td>
                                    <td>${emp.email}</td>
                                    <td>${emp.department.deptName}</td>
                                    <td>
                                        <button class="btn btn-primary btn-sm">
                                            <span class="glyphicon glyphicon-
pencil" aria-hidden="true"></span>编辑
                                        </button>
                                        <button class="btn btn-danger btn-sm">
                                            <span class="glyphicon glyphicon-
trash" aria-hidden="true"></span>删除
                                        </button>
                                    </td>
                                </tr>
                            </c:forEach>
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
        <%--显示分页信息--%>
        <div class="row">
            <%--分页文字信息--%>
            <div class="col-md-6">
                当前${pageInfo.pageNum}页,总${pageInfo.pages}页,总
${pageInfo.total}条记录
            </div>
            <%--分页条信息--%>
            <div class="col-md-6">
                <nav aria-label="Page navigation">
                    <ul class="pagination">
                        <li><a href="${APP_PATH}/emps?pageNum=1">首页</a></li>
                        <%--如果有上一页，就显示上一页按钮--%>
                        <c:if test="${pageInfo.hasPreviousPage}">
                            <li>
                                <a href="${APP_PATH}/emps?
pageNum=${pageInfo.pageNum - 1}" aria-label="Previous">
                                    <span aria-hidden="true">&laquo;</span>
                                </a>
                            </li>
                        </c:if>


                        <c:forEach items="${pageInfo.navigatepageNums}"
var="page_num">
                            <c:if test="${page_num == pageInfo.pageNum}">
                                <li class="active"><a href="">${page_num}</a>
</li>
                            </c:if>
                            <c:if test="${page_num != pageInfo.pageNum}">
```

```
                                <li><a href="${APP_PATH}/emps?
pageNum=${page_num}">${page_num}</a></li>
                            </c:if>


                        </c:forEach>
                        <%--如果有下一页，就显示下一页按钮--%>
                        <c:if test="${pageInfo.hasNextPage}">
                            <li>
                                <a href="${APP_PATH}/emps?
pageNum=${pageInfo.pageNum+1}" aria-label="Next">
                                    <span aria-hidden="true">&raquo;</span>
                                </a>
                            </li>
                        </c:if>


                        <li><a href="${APP_PATH}/emps?pageNum=${pageInfo.pages}">
末页</a></li>
                    </ul>
                </nav>
            </div>
        </div>
    </div>
</body>
</html>
```

6、为了实现各个客户端的通用性，我们希望服务端返回的数据是以json格式返回的。

- index.jsp页面直接发送ajax请求进行员工分页数据的查询
- 服务器将查出的数据以json字符串的形式返回给浏览器
- 浏览器收到js字符串。可以是以js对json进行解析，使用js通过dom增删改 改变页面
- 返回json，实现客户端的无关性

1.编写一个用来返回json数据的通用类

```
package com.study.bean;


import java.util.HashMap;
import java.util.Map;


/**
* @author tao
* @create 2021-07-28 10:24
* @Description 用来返回json数据的通用类
*/
public class Msg {


    // 状态码 100成功 200失败
    private int code;
    // 提示信息
    private String msg;
    // 用户要返回给浏览器的数据
```

```
    private Map<String,Object> extend = new HashMap<String, Object>();


    public static Msg success(){
        Msg msg = new Msg();
        msg.setCode(100);
        msg.setMsg("处理成功");
        return msg;
    }
    public static Msg fail(){
        Msg msg = new Msg();
        msg.setCode(200);
        msg.setMsg("处理失败");
        return msg;
    }


    // 处理链式调用
    public Msg add(String key,Object value){
        this.getExtend().put(key,value);
        return this;
    }


    public int getCode() {
        return code;
    }


    public void setCode(int code) {
        this.code = code;
    }


    public String getMsg() {
        return msg;
    }


    public void setMsg(String msg) {
        this.msg = msg;
    }


    public Map<String, Object> getExtend() {
        return extend;
    }


    public void setExtend(Map<String, Object> extend) {
        this.extend = extend;
    }
}
```

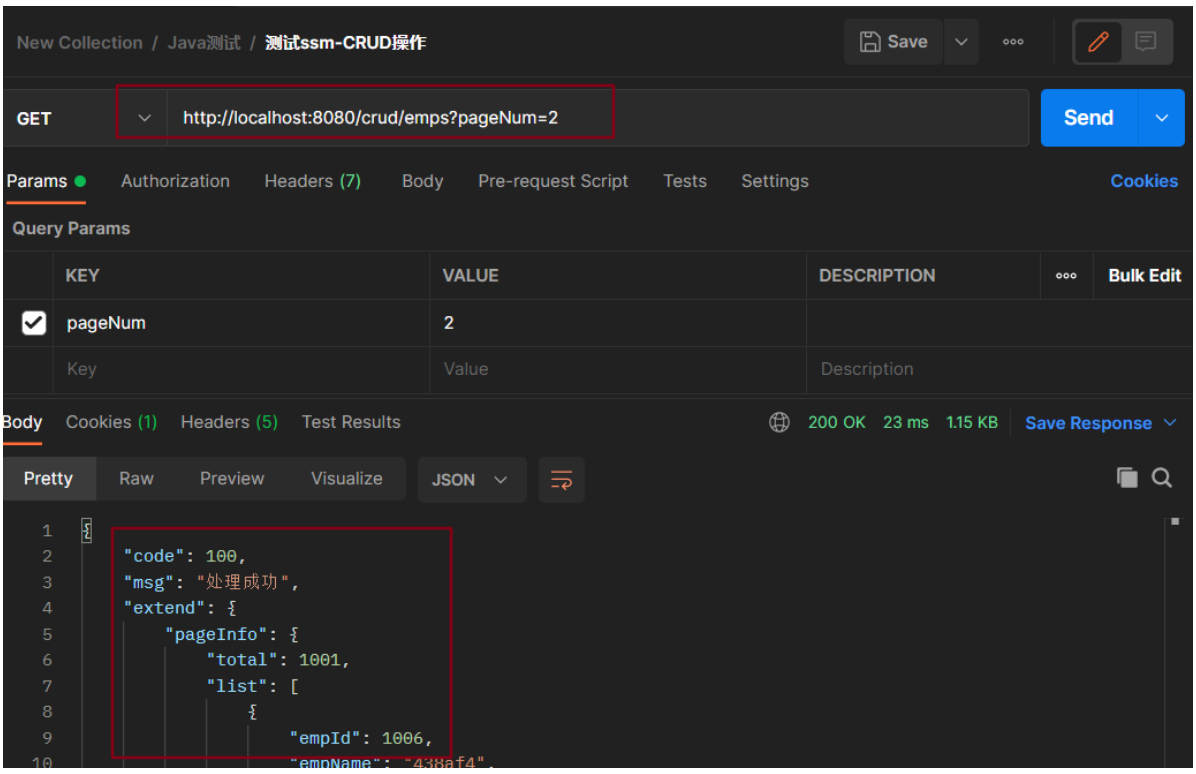
2.改写contoller层的方法，导入Jackson包

```
<!--    引入jackson，返回json字符串的支持-->
<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-
databind -->
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
```

```
  <version>2.11.0</version>
</dependency>

/**
* @Description: 要让ResponseBody正常工作，我们需要导入Jackson包，
* 才能将返回给客户端的对象转换为json字符串
* @Author: taoz
* @Date: 2021/7/28 10:47
* @param pageNum:  当前页码
* @return: com.study.bean.Msg
**/
@RequestMapping("/emps")
@ResponseBody
public Msg getEmpsWithJson(@RequestParam(value = "pageNum",defaultValue =
"1")Integer pageNum){
    //引入PageHelper分页插件
    // 在查询之前调用，并传入参数(当前页码，每一页的容量(这里配置的5))
    PageHelper.startPage(pageNum, 5);
    // 紧跟着startPage后面的查询方法就是一个分页查询
    List<Employee> allEmployees = employeeService.getAllEmployees();
    // 使用PageInfo保证查询后的结果，只要将PageInfo的实例化对象交给页面就行了
    // 封装了详细的分页信息，包括有我们的查询出来的数据,传入连续显示的页数
    PageInfo page = new PageInfo(allEmployees,5);
    return Msg.success().add("pageInfo",page);
}
```

测试：



7、改造index.jsp页面，使其进来就发送ajax请求数据，而不是从请求域拿数据

- 表格展示数据

```
//1、页面加载完成以后，直接发送一个ajax请求，要到分页数据
```

```
/*页面加载完成*/
$(function (){
   $.ajax({
       url:"${APP_PATH}/emps",
       data:{pageNum:1},
       type:"get",
       success:function (result) {
           build_emps_table(result);
       }
   })


    /*创建员工表格信息*/
    function build_emps_table(result){
        var emps = result.extend.pageInfo.list;
        /*jquery的遍历方法，item为遍历数组中的每一个*/
        $.each(emps,function (index,item){
            //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)


            // 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"));


            //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"));


            //把两个按钮放到一个单元格中，并且按钮之间留点空隙
            var btnTd = $("<td></td>").append(editBtn).append("
").append(delBtn);


            //append方法执行完成以后还是会返回原来的元素，所以可以一直.append添加元素，
            //将上面的td添加到同一个tr里
            $("<tr></tr>").append(checkBoxTd)
                .append(empIdTd)
                .append(empNameTd)
                .append(genderTd)
                .append(emailTd)
                .append(deptNameTd)
                .append(btnTd)
                .appendTo("#emps_table tbody");//将tr添加到tbody标签中
        })
    }
```

- 分页条处理

- 为了让分页更加的合理化，我们需要在配置分页插件的配置文件中配置
- https://pagehelper.github.io/docs/howtouse/#2-%E9%85%8D%E7%BD%AE%E6%8B%A6%E6%88%AA%E5%99%A8%E6%8F%92%E4%BB%B6

1. reasonable：分页合理化参数，默认值为**false**。当该参数设置为 **true** 时，**pageNum<=0** 时会查询第一页， **pageNum>pages**（超过总数时），会查询最后一页。默认**false** 时，直接根据参数进行查询。

```
<!--plugins标签要放在typeAliases标签后面-->
    <plugins>
        <plugin interceptor="com.github.pagehelper.PageInterceptor">
<!--            分页合理化，如果pageNum > pages，就让他查询最后一页。如果pageNum < 0，就查询第一页-->
            <property name="reasonable" value="true"/>
        </plugin>
    </plugins>
```

```
<script>
    //1、页面加载完成以后，直接发送一个ajax请求，要到分页数据
    /*页面加载完成*/
    $(function (){
        to_page(1)
    });


    function to_page(pageNum){
        $.ajax({
            url:"${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);
            }
        })
    }


    // 解析显示分页信息
    function build_page_info(result){
        // 清空分页信息
        $("#page_info_area").empty();
        $("#page_info_area").append("当前"+result.pageNum+"页,总"+result.pages+"页,总"+result.total+"条记录")
    }


        /*分页导航信息*/
    function build_page_nav(result){
        // 清空分页导航信息
        $("#page_nav_area").empty();
        var ul = $("<ul></ul>").addClass("pagination");
        // page_nav_area 构建元素
        // 首页
        var firstPageLi = $("<li></li>").append($("<a></a>").append("首页").attr("href","#"))
        // 上一页
        var prePageLi = $("<li></li>").append($("<a></a>").append("&laquo;"))
```

```
        // 如果没有上一页
        if(result.hasPreviousPage==false){
            firstPageLi.addClass("disable")
            prePageLi.addClass("disable")
        }else{
            // 为元素添加点击翻页的事件
            firstPageLi.click(function (){
                to_page(1);
            })
            prePageLi.click(function (){
                to_page(result.pageNum - 1)
            })
        }




        //添加首页和前一页 构建元素
        ul.append(firstPageLi).append(prePageLi)
        // 下一页
        var nextPageLi = $("<li></li>").append($("<a></a>").append("&raquo;"))
        // 最后一页
        var lastPageLi = $("<li></li>").append($("<a></a>").append("末
页").attr("href","#"))


        // 如果没有下一页
        if(result.hasNextPage==false){
            nextPageLi.addClass("disable")
            lastPageLi.addClass("disable")
        }else{
            lastPageLi.click(function (){
                to_page(result.pages)
            })


            nextPageLi.click(function (){
                to_page(result.pageNum + 1)
            })
        }


        $.each(result.navigatepageNums,function (index,item) {
            var numLi = $("<li></li>").append($("<a></a>").append(item))
            if(item == result.pageNum){
                numLi.addClass("active")
            }
            numLi.click(function (){
                to_page(item)
            })
            ul.append(numLi)
        })


        // 添加下一个和末页
        ul.append(nextPageLi).append(lastPageLi)


        // 把ul加入到nav
        var navEle = $("<nav></nav>").append(ul)
        navEle.appendTo("#page_nav_area")
    }
```

```
</script>
```