# 第五周总结：

## 大致总结

### (一)SSM框架整合——简单CRM管理系统练习

在周一周二两天的时间里，通过视频教程，学习一个SSM框架的简单CRM项目如何从环境搭建到一步步实现。

这个CRM项目包含对用户信的息增删改查，用户信息展示需要进行分页处理，这也是项目的难点之一。另外的难点包括如何做到灵活搭建环境、以及多条件查询和页面展示。

### (二) 学习Vue

必要性：由于自己没有学习过前端框架，前端又比较薄弱，再加上工作中使用会比较多，所有有必要尽快熟悉并掌握Vue开发。

感想：对自己来说，通过视频学习进行入门会更好。

在学习时，起初去硬看官方文档，一边看一边练习，实在痛苦，并且效率太低，理论太多，好多看不懂。看了一天后去找视频学习，感觉好太多，学习别人对技术的理解总比自己研究技术来得容易，虽然是拿来主义，却对于入门级的学习者来说，更加容易理解，能快速入手，而文档学习门槛太高，不易学习。

## 学习笔记

包含SSM框架整合笔记、Vue学习笔记

### (一) SSM框架整合

\*\*\*\*\*\*springmvc简单介绍\*\*\*\*\*\*

一、整合思路

Dao层：

1、SqlMapConfig.xml，空文件即可，但是需要文件头。

2、applicationContext-dao.xml

a) 数据库连接Druid

b) SqlSessionFactory对象，需要spring和mybatis整合包下的。

c) 配置mapper文件扫描器。Mapper动态代理开发 增强版

Service层：

1、applicationContext-service.xml包扫描器，扫描@service注解的类。

2、applicationContext-trans.xml配置事务。

Controller层：

1、Springmvc.xml

a) 包扫描器，扫描@Controller注解的类。

b) 配置注解驱动

c) 配置视图解析器

Web.xml文件：

1、配置spring监听器

2、配置前端控制器。

二、分页问题

目的：获取Page<Customer>对象，返回前台

分析：分析拼接Page<Customer>对象所需的数据 前台/后台

易错点：1.模糊查询的书写 <where> <if>标签

2.serivice层边界的控制

QueryVo非空

过滤掉空值和null

### （二）Vue

\*\*\*\*\*\*Vue简单介绍\*\*\*\*\*\*

\*\*Vue基础篇\*\*

一、Vue是什么

渐进式JavaScript 框架

作用：动态构建用户界面

二、入门实例——输入框的内容同步到页面显示

1. 引入Vue.js

<script type="text/javascript" src="../js/vue.js"></script>

2. 创建Vue对象

el : 指定根element(选择器)

data : 初始化数据(页面可以访问)

<script type="text/javascript">

const vm=new Vue({//配置对象 options

// 配置选项(option)

el:"#app",//element: 选择器字符串 表明将哪个页面元素交给vm实例管理

data:{

username:'atguigu'

}

})

</script>

3. 双向数据绑定 : v-model

<div id="app">

<input type="text" v-model="username">

..

</div>

4. 显示数据 : {{xxx}}

<div id="app">

..

<p>hello{{username}}</p>

</div>

5. 理解vue的mvvm实现

m: model 模型，数据对象(data)

v: View 视图，模板页面

vm：viewModel 视图模型(Vue实例对象) new Vue()

三、模板语法

动态的html页面

包含了一些JS语法代码

大括号表达式

指令(以v-开头的自定义标签属性)

1.双大括号表达式

语法：{{exp}}

功能：输出数据

可以调用对象的方法

v-text="msg" :将msg看做文本 textContent

v-html="msg" ：将msg看作html页面标签 innnerHtml

2.指令一：强制数据绑定v-bind

功能：

指定变化的属性值

完整语法:

v-bind：xxx='yyy'//yyy会作为表达式解析执行

简洁写法:

:xxx='yyy'

示例说明:

<a v-bind:href="url">访问指定站点2</a><br>

或<a :href="url">访问指定站点2</a><br>

data: {

url: 'http://www.atguigu.com'

}

3.指令二：绑定事件监听v-on

功能:

绑定指定事件名的回调函数

完整语法:

v-on:click=''

简洁写法:

@click=''

示例说明:

<button v-on:click="test">绑定监听事件</button>

或<button @click="test">绑定监听事件</button>

methods: {

test () {

alert('好啊!!!')

}

}

四、计算属性和监视

1.计算属性computed

功能：

相关得datas属性数据发生改变时/初始化显示

语法：

在computed属性对象中定义计算属性的方法

在页面中使用{{方法名}}来显示计算的结果

示例说明：

computed:{//计算属性配置: 值为对象

fullName1(){//计算属性中得一个方法，方法的返回值作为属性值

return this.firstName+' '+this.lastName

}

}

调用：<input type="text" v-model="fullName1"><br>

2.监视属性watch

功能：

当属性变化时, 回调函数自动调用, 在函数内部进行计算

语法：

通过通过vm对象的$watch()或watch配置来监视指定的属性

vm.$watch('监视对象',fucntion(newValue){做什么})

示例说明：

方法一：

watch: {

// 配置监视firstName

firstName: function (newValue) { // 相当于属性的set

// 更新fullName2

this.fullName2 = newValue + '-' + this.lastName

}

}

调用：<input type="text" v-model="fullName2"><br>

方法二：

// 监视firstName

vm.$watch('firstName', function (newValue) {

this.fullName2 = newValue + '-' + this.lastName

})

调用：<input type="text" v-model="fullName2"><br>

3.计算属性高级

功能：

通过getter/setter实现对属性数据的显示和监视

计算属性存在缓存, 多次读取只执行一次getter计算

getter:属性的get()

setter:属性的set()

示例说明：

computed:{

fullName3:{//监视fullName3属性

// 当获取当前属性值时自动调用, 将返回值(根据相关的其它属性数据)作为属性值

get(){

return this.firstName+' '+this.lastName

},

// 当fullName3属性值发生了改变时自动调用, 监视当前属性值变化, 同步更新相关的其它属性值

set(value){//value为fullName3最新的值

const names=value.split(' ')

// 更新firstName和lastName

this.firstName=names[0]

this.lastName=names[1]

}

}

}

五、class与style绑定

class/style绑定就是专门用来实现动态样式效果的技术

1.class绑定

:class='xxx'

xxx是字符串

xxx是对象 对象需要用{}包住

xxx是数组

示例说明：

<!--css书写-->

<style>

.classA {

color: red;

}

.classB {

background: blue;

}

</style>

<!--class绑定-->

<p :class="myClass">xxx是字符串</p>

<p :class="{classA: hasClassA, classB: hasClassB}">xxx是对象</p>

或<p :class="{classA: true, classB: false}">xxx是对象</p>

<!--data数据书写-->

data: {

myClass: 'classA',

hasClassA: true,

hasClassB: false,

activeColor: 'red',

fontSize: '20px'

}

2.style绑定

:style="{ color: activeColor, fontSize: fontSize + 'px' }"

其中activeColor/fontSize是data属性

六、条件渲染

1. 条件渲染指令

v-if

v-else

v-show

2. 比较v-if与v-show

如果需要频繁切换 v-show 较好

v-if:直接移除

v-show:使用dispaly:none隐藏

七、列表渲染

1. 列表显示

a.数组: v-for / index [{key:value},{key:value}]

示例说明：

</ul>

<li v-for="(p,index) in persons" key="index">

{{index}}—-{{p.name}}--{{p.age}}

</li>

</ul>

b.对象(用得少): v-for / key {key:value,key:value}

示例说明：

<ul>

<li v-for="(item, key) in items :key="key">

{{key}}={{item}}

</li>

</ul>

2. 列表的更新显示——修改数据即可

删除item

this.persons.splice(index, 1)//index表示数组下标

替换item

this.persons.splice(index, 1, newP)/newP表示新增的对象

添加

this.persons.push(newP)

3. 列表过滤

4. 列表排序

\*\*涉及知识点\*\*

一、Vue是什么

Vue.js就是一个用于搭建类似于网页版知乎这种表单项繁多，且内容需要根据用户的操作进行修改的网页版应用。

二、单页应用程序(SPA)

单页应用一般指的就是一个页面就是应用，当然也可以是一个子应用.

单页应用程序中一般交互处理非常多，而且页面中的内容需要根据用户的操作动态变化

三、响应式的数据绑定

vue.js会自动对页面中某些数据的变化做出响应 同步

四、Webpack

前端打包工具

一次性把静态资源加载 不用每次都请求(每次握手与挥手消耗时间一般比加载这些资源更长)

五、Node.js与NPM

Node.js是一个服务端的JavaScript运行环境，通过Node.js可以实现用JavaScript写独立程序。

NPM是一个node.js的包管理器。

六、Vue-CLi

一个vue.js的脚手架工具。

就是一个自动帮你生成好项目目录，配置好Webpack，以及各种依赖包的工具

七、Vuex和Vue-route

Vuex是vue的一个状态管理器。用于集中管理一个单页应用程序中的各种状态。

Vue-route是vue的一个前端路由器，这个路由器不是我们上网用的路由器，而是一个管理请求入口和页面映射关系的东西。它可以实现对页面局部进行无刷新的替换，让用户感觉就像切换到了网页一样。

八、Vue-resource和Axios

单页应用程序属于前后端分离开发，那么这个单页应用程序中的数据就得通过ajax的方式获取，也要通过ajax的方式提交到后端。

vue.js本身没有封装ajax操作库，所以我们要通过Vue-resource和Axios来进行ajax操作