wtypora-copy-images-to: img

day31-Vue

今日内容介绍

- VueJS介绍
- VueJS常用系统指令------->重点掌握
  - 常用的事件
  - v-text
  - v-html
  - v-bind
  - v-model
  - v-for
  - v-if 或 v-show
- VueJS生命周期
- VueJS的ajax请求------->重点掌握
  - get请求
  - post请求
- 综合案例

第一章-VueJS介绍与快速入门

1.1 VueJS介绍

什么是VueJS

Vue.js是一个渐进式JavaScript 框架。Vue.js 的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。它不仅易于上手，还便于与第三方库或既有项目整合。

官网:https://cn.vuejs.org/

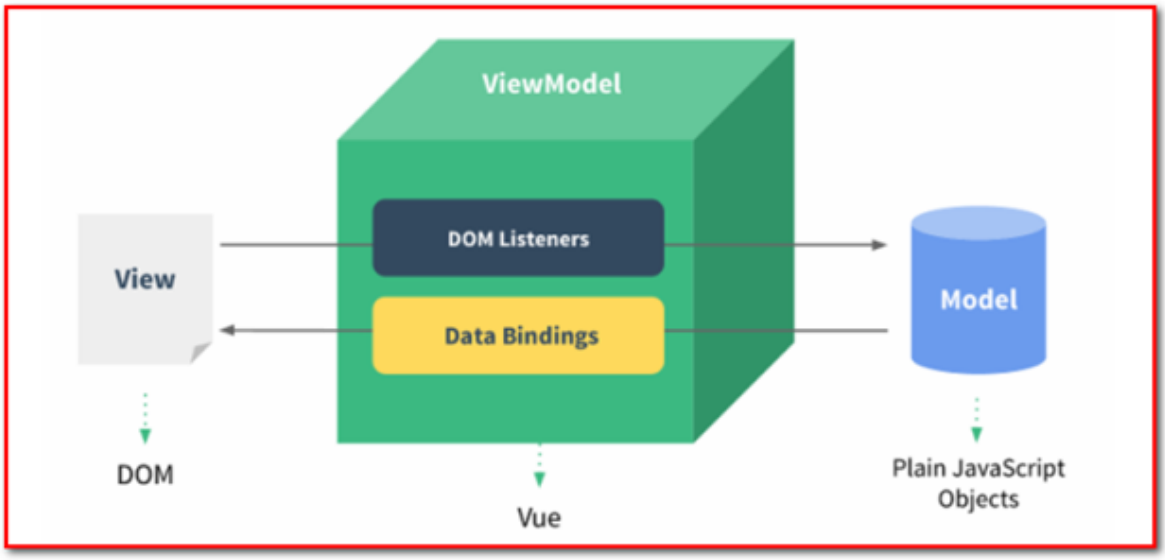
特点

- 易用
- 灵活
- 高效

MVVM模式

MVVM是Model-View-View-Model的简写。它本质上就是MVC 的改进版。

MVVM 就是将其中的View 的状态和行为抽象化，让我们将视图UI 和业务逻辑分开. MVVM模式和MVC模式一样，主要目的是分离视图（View）和模型（Model）Vue.js 是一个提供了 MVVM 风格的==双向数据绑定==的 Javascript 库，专注于View 层。它的核心是 MVVM 中的 VM，也就是 ViewModel。ViewModel负责连接 View 和 Model，保证视图和数据的一致性，这种轻量级的架构让前端开发更加高效、便捷.



1.2 案例-VueJs快速入门

1.需求

使用vue，对message赋值，并把值显示到页面

2.步骤oi

1. 创建工程,拷贝vue.js到工程
2. 创建demo01.html 把vue.js引入到页面
3. 创建vue实例, 定义message 显示

3.实现

1. 创建工程(war),导入vuejs
2. 创建demo01.js(引入vuejs,定义div,创建vue实例)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>01_vue的入门</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
    <!--插值表达式{{}}-->
    {{message}}<br/>
    {{num}}<br/>
    {{flag}}<br>

</div>

</body>
<script>
    // 1.创建Vue对象
    new Vue({
        // 当前vue对象接管了id为app的div
```

```
        el:"#app",
        // 定义数据模型
        data:{
            message:"hello vue...",
            num: 10,
            flag: true
        },
        // 定义函数
        methods:{}
    });
</script>

</html>
```

data： 用于定义数据。

methods： 用于定义的函数，可以通过 return 来返回函数值。

4.注意

数据绑定最常见的形式就是使用“Mustache”语法 (双大括号) 的文本==插值表达式==，Mustache 标签将会被替代为对应数据对象上属性的值。无论何时，绑定的数据对象上属性发生了改变，插值处的内容都会更新.

第二章-VueJS 常用系统指令

2.1 常用的事件【重点】

- 绑定事件:
  - 方式一: v-on:事件名="函数名(参数)"
  - 方式二: @事件名="函数名(参数)"

@click

说明: 点击事件(等同于v-on:click)

【需求】： 点击按钮事件，改变message的值

- 02_vue的点击事件.html

```
<!DOCTYPE html>
<html lang="en" xmlns:v-on="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <title>02_vue的点击事件</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
    {{message}}<br/>
    <!--<input type="button" value="点击事件,改变message的值" v-on:click="f1()">-->
    <input type="button" value="点击事件,改变message的值" @click="f1()">
</div>
```

```
</body>
<script>
    var a = new Vue({
        el: "#app",
        data: {
            message:"hello world..."
        },
        methods: {
            f1:function (){
                // alert("点击事件");
                // 修改message的值    this:表示当前vue对象
                // this.message = "hello vue...";
                a.message = "hello vue...";
            }
        }
    });

</script>
</html>
```

@keydown

说明: 键盘按下事件(等同于v-on:keydown)

【需求】：对文本输入框做校验，使用键盘按下事件，如果按下0-9的数字，正常显示，其他按键则阻止事件执行。

- 03_vue的键盘按下事件

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>03_vue的键盘按下事件</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
    {{message}}<br/>
    <input type="text" @keydown="f1($event)">
</div>
</body>
<script>
    new Vue({
        el:"#app",
        data:{
            message:""
        },
        methods:{
            f1:function (o) {
                // 1.获取按下的键----代码
                var keyCode = o.keyCode;
                // console.log(keyCode);
                // 把值赋值给message,修改数据模型
                this.message = keyCode;

                // 2.判断如果按下的键不是0-9,就阻止事件执行
```

```
                if (keyCode < 48 || keyCode > 57) {
                    // 阻止事件执行
                    o.preventDefault();
                }
            }
        }
    });
</script>
</html>
```

@mouseover

说明:鼠标移入区域事件(等同于v-on:mouseover)

【需求1】：给指定区域大小的div中添加样式，鼠标移到div中，弹出窗口。

- 04_vue的鼠标移入事件.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>04_vue的鼠标移入事件</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
    <div style="border: 1px solid red; width: 100px;height: 100px"
@mouseover="f1">1111</div>
</div>
</body>
<script>
    new Vue({
        el:"#app",
        data:{},
        methods:{
            f1:function() {
                alert("鼠标移入事件...")
            }
        }
    });
</script>
</html>
```

2.2 v-text与v-html

v-text：输出文本内容，不会解析html元素---->文本

v-html：输出文本内容，会解析html元素----->标签体

用在标签的属性里面

【需求】：分别使用v-text, v-html 赋值 `<h1>hello world<h1>` ，查看页面输出内容。

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>05_v-text与v-html</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
    <!--v-text\v-html写在标签的属性位置-->

    <!--<span>{{msg1}}</span><br/>-->
    <!--<span>{{msg2}}</span><br/>-->

    <!--<span v-text="msg1"></span><br/>-->
    <!--<span v-text="msg2"></span><br/>-->

    <span v-html="msg1"></span><br/>
    <span v-html="msg2"></span><br/>

</div>
</body>
<script>
    new Vue({
        el:"#app",
        data:{
            msg1:"hello vue1...",
            msg2:"<h1>hello vue2...</h1>"
        }
    });
</script>
</html>
```

> 用在标签的属性

2.3 v-bind和v-model【重点】

v-bind

==插值语法不能作用在HTML 属性上==，遇到这种情况应该使用 v-bind指令,v-bind可以省略,直接在属性名前面写冒号

【需求】：使用vue定义变量ys，对页面中的字体标签color属性赋值。

使用vue定义变量info，对页面中的超链接href属性赋值。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>06_v-bind</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
    <font v-bind:color="ys">字体标签</font>
    <a v-bind:href="info">黑马程序员</a><br/>
    <a v-bind:href="'http://www.baidu.com?id='+id">百度</a><br/>
```

```
    <!--省略了v-bind-->
    <a :href="'http://www.baidu.com?id='+id">百度</a><br/>

    <input type="button" value="改变字体标签的颜色" @click="f1()"><br/>

</div>
</body>
<script>
    new Vue({
        el:"#app",
        data:{
            ys:"red",
            info:"http://www.itheima.com",
            id:3
        },
        methods:{
            f1:function () {
                this.ys = "blue";
            }
        }
    });
</script>
</html>
```

v-model

用于数据的绑定,数据的读取

【需求】：使用vue赋值json(对象)数据，并显示到页面的输入框中（表单回显）.点击按钮，改变json数据，验证同时输入框的内容也发生改变。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>07_v-model  </title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
    <form>
        用户名:<input type="text" v-model="user.username" name="username"><br/>
        密码:<input type="text"  v-model="user.password" name="password"><br/>
        性别:<input type="radio"  v-model="user.sex" name="sex" value="男">男
             <input type="radio"  v-model="user.sex" name="sex" value="女">女
<br/>
        地址:
            <select name="address"  v-model="user.address">
                <option value="sz">深圳</option>
                <option value="bj">北京</option>
                <option value="sh">上海</option>
            </select><br/>
            <input type="button" value="提交(改变数据模型中的数据,视图跟着改变)"
@click="f1()"><br/>
            <input type="button" value="提交(视图中的数据改变了,数据模型跟着改变)"
@click="f2()"><br/>
    </form>
```

```
</div>
</body>
<script>
    new Vue({
        el:"#app",
        data:{
            user:{}
        },
        methods:{
            // 数据模型变了,视图也跟着变了
            f1:function () {
                // 改变数据模型user对象的值
                this.user = {
                    username:"ls",
                    password:"abcdef",
                    sex:"男",
                    address:"sh",
                };
            },

            // 视图变了,数据模型也变了
            f2:function () {
                // 输出数据模型中的数据
                console.log(this.user.username);
                console.log(this.user.password);
                console.log(this.user.sex);
                console.log(this.user.address);
            }
        }
    });
</script>
</html>
```

2.4 v-for,v-if,v-show

v-for【重点】

用于操作array/集合，遍历

```
<标签 v-for="(元素,索引) in 数组"></标签>
//1.元素的变量名随便取
//2.索引的变量名随便取
```

【需求】：使用v-for遍历数组，并把数据遍历到页面上的

- 标签中。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title>08_v-for</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
    <ul>
        <li v-for="(e,i) in arr" v-text="e"></li>
    </ul>
    <hr/>
    <ul>
        <li v-for="(e,i) in list">
            {{e.username}}---{{e.password}}
        </li>
    </ul>
</div>
</body>
<script>
    new Vue({
        el:"#app",
        data:{
            arr:["深圳","东莞","惠州","广州"],
            list:[
                {username:"zs",password:"123456"},
                {username:"ls",password:"abcdef"},
                {username:"zl",password:"ABCDEF"}
            ]
        },
        methods:{

        }
    });
</script>
</html>
```

v-if【重点】与v-show

v-if是根据表达式的值来决定是否渲染元素(**标签都没有了**)

v-show是根据表达式的值来切换元素的display css属性(**标签还在)**

```
<标签 v-if="boolean类型的"></标签>
//1.v-if里面是true, 展示
//2.v-if里面是false, 不展示,标签都没有
```

【需求】：使用vue赋值flag变量（boolean类型），用来判断元素中的内容是否显示。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>09_v-if【重点】与v-show</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
<div id="app">
```

```
    <!--v-if:如果表达式的值为true,就显示,如果为false,就不显示(标签也没有了)-->
    <!--v-show:如果表达式的值为true,就显示,如果为false,就不显示(标签还在,只是设置了
style="display: none;")-->
    <span v-if="flag">span标签1</span><br/>
    <span v-show="flag">span标签2</span><br/>
    <input type="button" value="切换状态" @click="f1()">
</div>
</body>
<script>
    new Vue({
        el:"#app",
        data:{
            flag:true,
        },
        methods:{
            f1:function () {
                // 切换状态
                this.flag = !this.flag;
            }
        }
    });
</script>
</html>
```

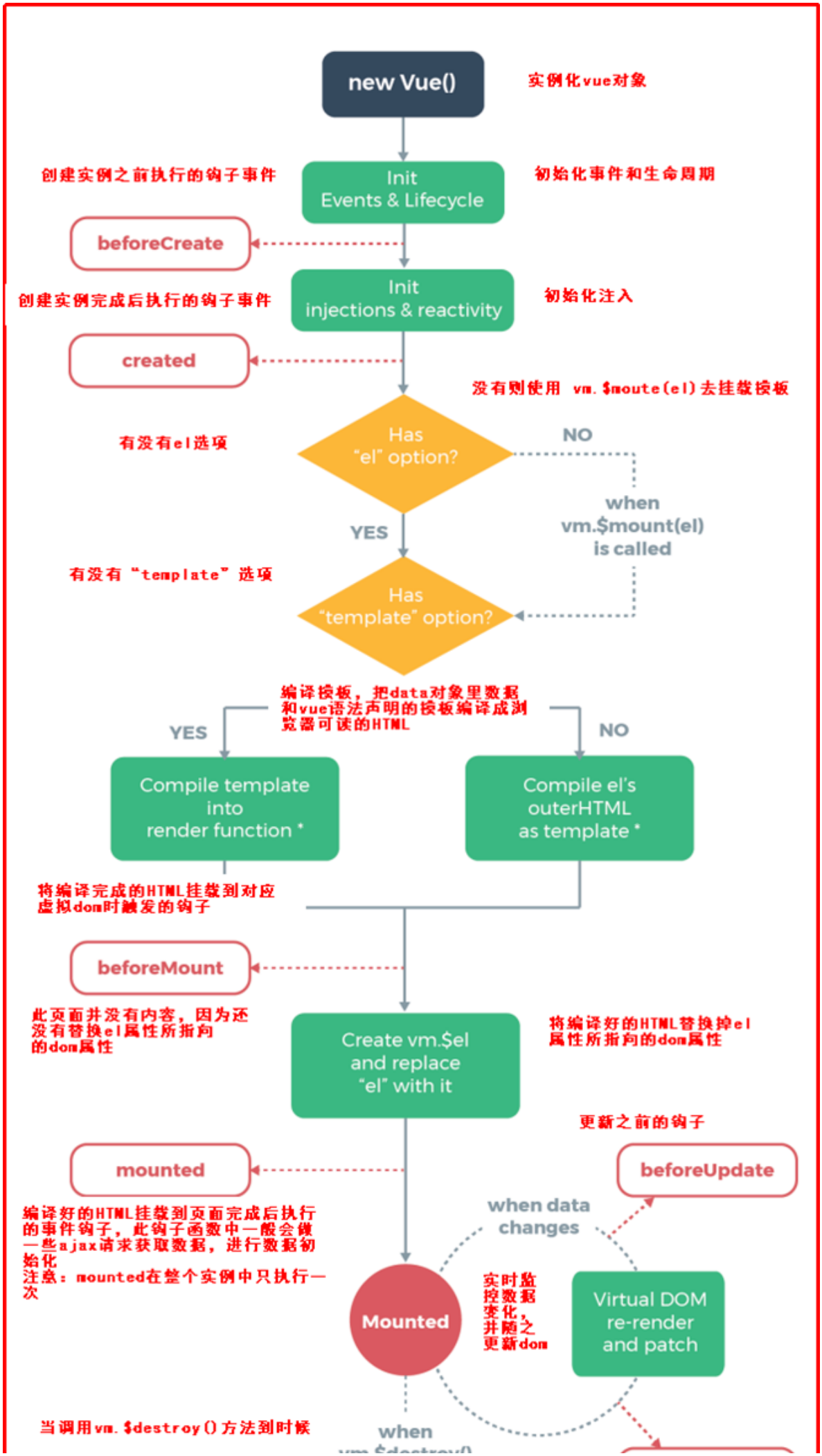
第三章-VueJS生命周期

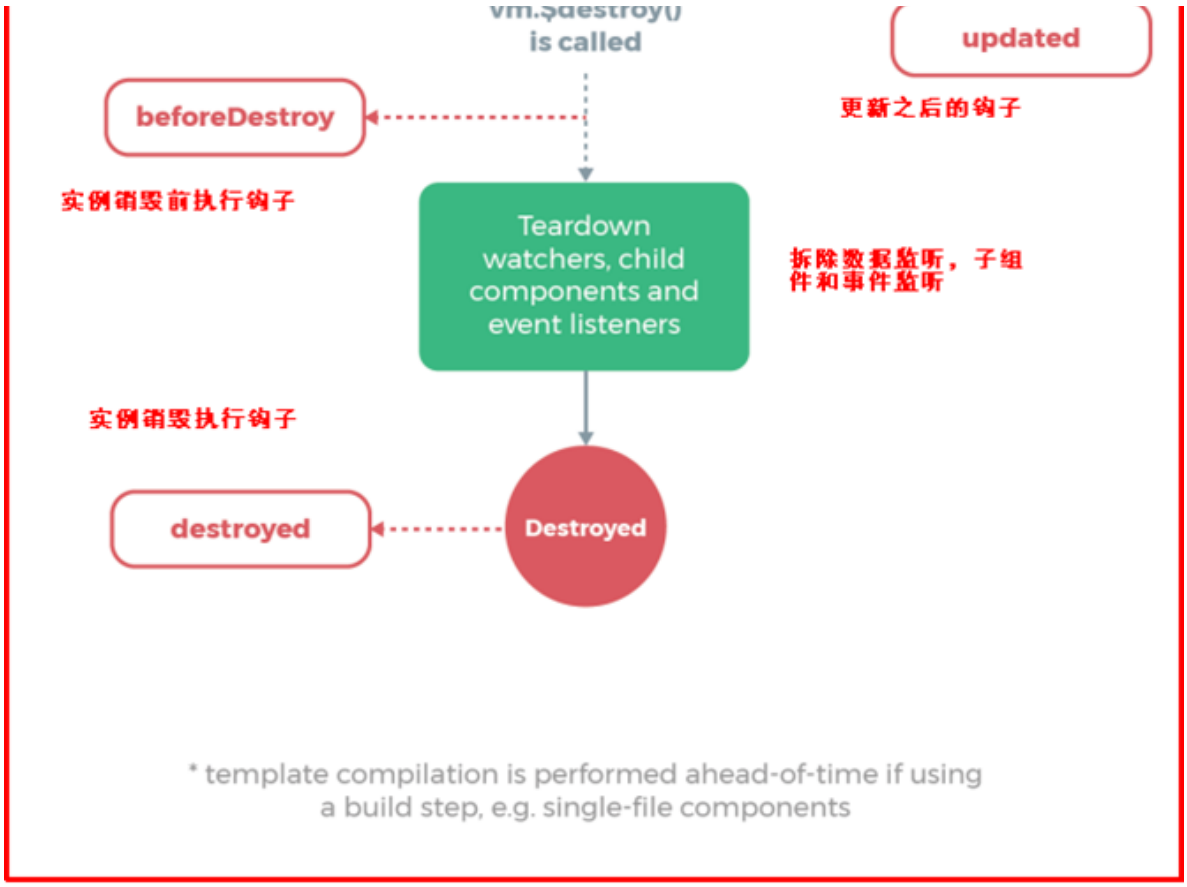
3.1VueJS生命周期

什么是VueJS生命周期

就是vue实例从创建到销毁的过程.

每个 Vue 实例在被创建到销毁都要经过一系列的初始化过程——例如，需要设置数据监听、编译模板、将实例挂载到 DOM 并在数据变化时更新 DOM 等。同时在这个过程中也会运行一些叫做生命周期钩子的函数，这给了用户在不同阶段添加自己的代码的机会。


new Vue()
实例化vue对象
创建实例之前执行的钩子事件
Init
Events & Lifecycle
初始化事件和生命周期
beforeCreate
创建实例完成后执行的钩子事件
Init
injections & reactivity
初始化注入
created
没有则使用 vm.$moute(el)去挂载模板
有没有el选项
Has
"el" option?
NO
when
vm.$mount(el)
is called
YES
有没有"template"选项
Has
"template" option?
编译模板，把data对象里数据
和vue语法声明的模板编译成浏
览器可读的HTML
YES
NO
Compile template
into
render function *
Compile el's
outerHTML
as template *
将编译完成的HTML挂载到对应
虚拟dom时触发的钩子
beforeMount
此页面并没有内容，因为还
没有替换el属性所指向
的dom属性
Create vm.$el
and replace
"el" with it
将编译好的HTML替换掉el
属性所指向的dom属性
更新之前的钩子
mounted
beforeUpdate
编译好的HTML挂载到页面完成后执行
的事件钩子，此钩子函数中一般会做
一些ajax请求获取数据，进行数据初
始化
注意：mounted在整个实例中只执行一
次
when data
changes
Mounted
实时监
控数据
变化，
并随之
更新dom
Virtual DOM
re-render
and patch
当调用vm.$destroy()方法到时候
when



- beforeCreate： 数据还没有监听，没有绑定到vue对象实例，同时也没有挂载对象
- ==created==： 数据已经绑定到了对象实例，但是还没有挂载对象（使用ajax可在此方法中查询数据，调用函数）
- beforeMount: 模板已经编译好了，根据数据和模板已经生成了对应的元素对象，将数据对象关联到了对象的

  el属性，el属性是一个HTMLElement对象，也就是这个阶段，vue实例通过原生的createElement等方法来创 建这个html片段，准备注入到我们vue实例指明的el属性所对应的挂载点
- ==mounted==:将el的内容挂载到了el，相当于我们在jquery执行了(el).html(el),生成页面上真正的dom，上面我们 就会发现dom的元素和我们el的元素是一致的。在此之后，我们能够用方法来获取到el元素下的dom对象，并 进行各种操作当我们的data发生改变时，会调用beforeUpdate和updated方法
- beforeUpdate： 数据更新到dom之前，我们可以看到$el对象已经修改，但是我们页面上dom的数据还 没有发生改变
- updated: dom结构会通过虚拟dom的原则，找到需要更新页面dom结构的最小路径，将改变更新到 dom上面，完成更新
- beforeDestroy,destroyed :实例的销毁，vue实例还是存在的，只是解绑了事件的监听、还有watcher对象数据 与view的绑定，即数据驱动

vuejs生命周期的演示

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>10_vuejs生命周期的演示</title>
    <script src="js/vuejs-2.5.16.js"></script>
</head>
<body>
```

```
<div id="app">
    {{message}}
</div>
</body>
<script>
    /*
        - beforeCreate ：数据还没有监听，没有绑定到vue对象实例，同时也没有挂载对象
        - ==created== ：数据已经绑定到了对象实例，但是还没有挂载对象（使用ajax可在此方法中查询数据，调用函数）
        - beforeMount: 模板已经编译好了，根据数据和模板已经生成了对应的元素对象，将数据对象关联到了对象的
          el属性，el属性是一个HTMLElement对象，也就是这个阶段，vue实例通过原生的createElement等方法来创
        建这个html片段，准备注入到我们vue实例指明的el属性所对应的挂载点
        - ==mounted==:将el的内容挂载到了el，相当于我们在jquery执行了(el).html(el),生成页面上真正的dom，上面我们
        就会发现dom的元素和我们el的元素是一致的。在此之后，我们能够用方法来获取到el元素下的dom对象，并
        进行各种操作当我们的data发生改变时，会调用beforeUpdate和updated方法
        - beforeUpdate ：数据更新到dom之前，我们可以看到$el对象已经修改，但是我们页面上dom的数据还
        没有发生改变
        - updated: dom结构会通过虚拟dom的原则，找到需要更新页面dom结构的最小路径，将改变更新到
        dom上面，完成更新
        - beforeDestroy,destroyed :实例的销毁，vue实例还是存在的，只是解绑了事件的监听、还有watcher对象数据
        与view的绑定，即数据驱动

     */
    var a = new Vue({
        el: "#app",
        data: {
            message:"hello vue..."
        },
        methods: {},
        beforeCreate:function () {
            // console.log("===beforeCreate===")
            showMsg("===beforeCreate===",this);
        },
        created:function () {
            // console.log("===created===")
            showMsg("===created===",this);
        },
        beforeMount:function () {
            // console.log("===beforeMount===")
            showMsg("===beforeMount===",this);
        },
        mounted:function () {
            // console.log("===mounted===")
            showMsg("===mounted===",this);
        },
        beforeDestroy:function () {
            console.log("===beforeDestroy===")
        },
        destroyed:function () {
            console.log("===destroyed===")
        }
```

```
    });

    function showMsg(msg, obj) {
        // msg:钩子函数名, obj:vue对象
        console.log("方法名:"+msg);
        console.log("数据模型data:" + obj.message);
        console.log("el元素:" + obj.$el);
        console.log("视图元素的内容:" + document.getElementById("app").innerHTML);
    }

    // vue死亡
    // a.$destroy();

</script>
</html>
```

- 结果

小结

1. **一般情况下 我们可以在==created或者mounted进行初始化(请求服务器获得数据绑定)==**

第四章-VueJS ajax

4.1 VueJS ajax

vue-resource【了解】

vue-resource是Vue.js的插件提供了使用XMLHttpRequest或JSONP进行Web请求和处理响应的服务。当vue更新到2.0之后，作者就宣告不再对vue-resource更新，而是推荐的axios，在这里大家了解一下vue-resource就可以。

vue-resource的github: https://github.com/pagekit/vue-resource

- Get方式

Get:

```
get:function () {
  this.$http.get("package.json",{
    params:{
      uesrId:"101"
    },
    headers:{
      token:"abcd"
    }
  }).then(res=>{
    this.msg = res.data;
  },error=>{
    this.msg = error;
  });
},
```

- Post方式

Post:

```
post:function () {
  this.$http.post("package.json",{
    userId:"102"
  },{
    headers:{
      access_token:"abc"
    }
  }).then(function (res) {
    this.msg = res.data;
  });
},
```

axios【重点】

什么是axios

Axios 是一个基于 promise 的 HTTP 库，可以用在浏览器和 node.js 中

注: Promise 对象用于表示一个异步操作的最终状态（完成或失败），以及其返回的值。

axios的github:https://github.com/axios/axios

中文说明: https://www.kancloud.cn/yunye/axios/234845

axios的语法

- get请求

```
// 为给定 ID 的 user 创建请求    url路径携带参数
axios.get('/user?ID=12345')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });

// 可选地，上面的请求可以这样做    json方式携带参数
axios.get('/user', {
    params: {
```

```
      ID: 12345,
    }
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

- post请求

```
// 为给定 ID 的 user 创建请求    url路径携带参数
axios.post('/user?ID=12345')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });

// 为给定 ID 的 user 创建请求    json携带参数
axios.post('/user', {
    firstName: 'Fred',
    lastName: 'Flintstone'
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

axios的使用

需求:使用axios发送异步请求给Servlet，获取响应的数据, 显示到页面上

步骤:

1. 把axios,vue导入项目中,并引入到页面
2. 使用get(), post() 请求
3. 创建Servlet

get请求

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>01_axios的get请求</title>
    <script src="js/vuejs-2.5.16.js"></script>
    <script src="js/axios-0.18.0.js"></script>
</head>
<body>
<div id="app">
    {{message}}
</div>
```

```
</body>
<script>
    var a = new Vue({
        el: "#app",
        data: {
            message:""
        },
        methods: {

        },
        created:function () {
            // 发送axios的异步请求----> get方式--->不携带参数
            /*axios.get("ServletDemo1").then(function (response) {
                // response参数名可以自定义
                // console.log(response);
                // console.log(JSON.stringify(response));
                // 获取服务器响应的数据
                // console.log(response.data);
                // 获取服务器响应的数据赋值给数据模型中的message
                // 如果在axios里面写的普通函数,this不是当前vue对象
                // this.message = response.data;// 赋值失败
                a.message = response.data;// 赋值成功

            });*/


            // 发送axios的异步请求----> get方式--->不携带参数
            /*axios.get("ServletDemo1").then(response=>{
                // 获取服务器响应的数据赋值给数据模型中的message
                // 如果在axios里面写的是普通函数(es5),this不是当前vue对象
                // 如果在axios里面写的是箭头函数(es6),this就是当前vue对象
                this.message = response.data;// 赋值成功
                // a.message = response.data;// 赋值成功

            });*/


            // 发送axios的异步请求----> get方式--->url方式携带参数
           /* axios.get("ServletDemo1?
username=zhangsan&password=123456").then(response=>{
                // 获取服务器响应的数据赋值给数据模型中的message
                this.message = response.data;// 赋值成功

            });*/

            // 发送axios的异步请求----> get方式--->json方式携带参数
            axios.get("ServletDemo1",{
                params:{
                    username:"lisi",
                    password:"abcdef"
                }
            }).then(response=>{
                // 获取服务器响应的数据赋值给数据模型中的message
                this.message = response.data;// 赋值成功

            });

        }
```

```
    });
</script>
</html>
```

```
package com.itheima.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * @Author: pengzhilin
 * @Date: 2021/5/15 12:01
 */
@WebServlet("/ServletDemo1")
public class ServletDemo1 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        System.out.println("来到了ServletDemo1...");
        // 1.处理乱码
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");

        // 2.获得请求参数
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        System.out.println("username:" + username + ",password:" + password);

        // 响应数据给页面
        response.getWriter().println("ServletDemo1......");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

post请求

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>02_axios的post请求</title>
    <script src="js/vuejs-2.5.16.js"></script>
    <script src="js/axios-0.18.0.js"></script>
</head>
<body>
<div id="app">
    {{message}}
```

```
</div>
</body>
<script>
    new Vue({
        el:"#app",
        data:{
            message:""
        },
        methods:{},
        created:function () {
           /* // axios发送post请求---->不携带请求参数
            axios.post("ServletDemo2").then(response=>{
                // response:响应回来的数据对象 (包含很多响应信息)
                // response.data:服务器响应回来的数据
                // console.log(response);
                // console.log(response.data);
                // 服务器响应回来的数据设置给数据模型message
                this.message = response.data;
            }).catch(error=> {
                // console.log(JSON.stringify(error));
                // alert("服务器异常");
                this.message = "服务器异常";
            });*/


            /*// axios发送post请求---->url路径方式携带请求参数
            axios.post("ServletDemo2?
username=zhangsan&password=123456").then(response=>{
                // 服务器响应回来的数据设置给数据模型message
                this.message = response.data;
            }).catch(error=> {
                // console.log(JSON.stringify(error));
                // alert("服务器异常");
                this.message = "服务器异常";
            });*/

            // axios发送post请求---->json格式方式携带请求参数
            axios.post("ServletDemo2",{
                username:"lisi",
                password:"abcdef"
            }).then(response=>{
                // 服务器响应回来的数据设置给数据模型message
                this.message = response.data;
            }).catch(error=> {
                // console.log(JSON.stringify(error));
                // alert("服务器异常");
                this.message = "服务器异常";
            });
        }
    });
</script>
</html>
```

```
/**
 * @Author: pengzhilin
 * @Date: 2021/5/15 14:33
```

```
 */
@WebServlet("/ServletDemo2")
public class ServletDemo2 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        System.out.println("来到了ServletDemo2...");
        // 1.处理乱码
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");

        //int r = 1/0;

        // 2.获取请求参数 ------>url路径方式携带请求参数--->使用请求对象的
getParameter(),getParameterMap(),...
       /* String username = request.getParameter("username");
        String password = request.getParameter("password");
        System.out.println("username:"+username+",password:"+password);*/

        // 2.获取请求参数 ------>json格式方式携带请求参数---->使用工具类JsonUtils
        //Map map = JsonUtils.parseJSON2Object(request, Map.class);
        //System.out.println("map:"+map);

        User user = JsonUtils.parseJSON2Object(request, User.class);
        System.out.println(user);

        // 响应数据到页面
        response.getWriter().println("ServletDemo2...");

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

注意

1. get方式携带请求参数(url路径携带,json格式携带)，以及post方式通过url携带请求参数，实际上携带给服务器的参数的格式是"name=value&name=value&name=value", 服务器接收到这种格式的请求参数的时候，可以使用request的getParameter(name)或者getParameterValues(name)或者getParameterMap()方法获取请求参数
2. post方式通过json格式携带请求参数，那么提交给服务器的参数的格式是{name:value,name:value},服务器就无法通过以前的getParameter(name)等方法获取请求参数。那么**服务器要通过解析json获取，我们直接使用工具类就行了**
3. get请求josn格式携带参数 {params: {key:value,key:value}} ,post请求json格式携带参数{key:value,key:value}

JsonUtils工具类和Result类

JsonUtils工具类(JsonUtils)

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>03_工具类JsonUtils的讲解</title>
    <script src="js/vuejs-2.5.16.js"></script>
    <script src="js/axios-0.18.0.js"></script>
</head>
<body>
<div id="app">
    {{message}}<br/>
    {{username}}<br/>
    {{password}}<br/>
</div>
</body>
<script>
    new Vue({
        el: "#app",
        data: {
            message:"",
            username:"",
            password:""
        },
        methods: {},
        created:function () {
            // 发送异步请求---->post方式,携带参数(json格式)
            axios.post("ServletDemo3",{
                username:"wangwu",
                password:"654321"
            }).then(response=>{
                console.log(response.data);
                // 把服务器数据赋值给数据模型message
                // this.message = response.data;
                this.username = response.data.username;
                this.password = response.data.password;
            })
        }
    });
</script>
</html>
```

```
package com.itheima.web;

import com.alibaba.fastjson.JSON;
import com.itheima.bean.User;
import com.itheima.utils.JsonUtils;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * @Author: pengzhilin
 * @Date: 2021/5/15 15:03
 */
@WebServlet("/ServletDemo3")
```

```
public class ServletDemo3 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        System.out.println("来到了ServletDemo3...");
        // 1.处理乱码
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");

        // 2.获得请求参数--->post请求,参数是以json格式携带的--->使用工具类JsonUtils
        User user = JsonUtils.parseJSON2Object(request, User.class);
        System.out.println("user:"+user);

        // 3.响应数据
        User u = new User("szitheima113","1234567");

        // 工具类:把u对象转换为json,再通过response对象把json响应到页面
        JsonUtils.printResult(response,u);

        /*// 把u对象转换为json
        String json = JSON.toJSONString(u);
        // 通过response对象把json响应到页面
        response.getWriter().println(json);*/


        //response.getWriter().println("ServletDemo3...");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

Result类

```
public class Result implements java.io.Serializable {
    private boolean flag;//执行结果，true为执行成功 false为执行失败
    private String message;//返回结果信息
    private Object result;//返回数据---json格式的字符串

    public Result(boolean flag,String message){
        this.flag = flag;
        this.message = message;
    }

    public Result(boolean flag, String message, Object result) {
        this.flag = flag;
        this.message = message;
        this.result = result;
    }
```

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <title>04_Result的JavaBean的讲解</title>
    <script src="js/vuejs-2.5.16.js"></script>
    <script src="js/axios-0.18.0.js"></script>
</head>
<body>
<div id="app">
    {{message}}<br/>
    {{username}}<br/>
    {{password}}<br/>
</div>
</body>
<script>
    new Vue({
        el: "#app",
        data: {
            message:"",
            username:"",
            password:""
        },
        methods: {},
        created:function () {
            // 发送异步请求---->post方式,携带参数(json格式)
            axios.post("ServletDemo4",{
                username:"wangwu",
                password:"654321"
            }).then(response=>{
                // console.log(response.data);
                // 把服务器数据赋值给数据模型message
                if (response.data.flag){
                    this.username = response.data.result.username;
                    this.password = response.data.result.password;
                } else{
                    this.message = "服务器异常";
                }

            })
        }
    });
</script>
</html>
```

```
/**
 * @Author: pengzhilin
 * @Date: 2021/5/15 15:03
 */
@WebServlet("/ServletDemo4")
public class ServletDemo4 extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        System.out.println("来到了ServletDemo4...");
        // 1.处理乱码
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
```

```
        // 2.获得请求参数--->post请求,参数是以json格式携带的--->使用工具类JsonUtils
        User user = JsonUtils.parseJSON2Object(request, User.class);
        System.out.println("user:"+user);

        // 3.响应数据
        User u = new User("szitheima113","1234567");

        if (u != null){
            // 封装数据
            Result result = new Result(true,"查询成功",u);
            // 把数据转换为json,响应到页面
            JsonUtils.printResult(response,result);
        }else{
            // 封装数据
            Result result = new Result(false,"查询失败");
            // 把数据转换为json,响应到页面
            JsonUtils.printResult(response,result);
        }

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

第五章-综合案例

案例一-显示所有联系人案例

一，案例需求

显示所有联系人

编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三	男	11	广州	766335435	766335435@qq.com	修改 删除
2	李四	男	12	上海	243424242	243424242@qq.com	修改 删除
3	王五	女	13	广州	474574574	474574574@qq.com	修改 删除
4	赵六	男	14	北京	987069697	987069697@qq.com	修改 删除
5	钱七	女	15	广州	412132145	412132145@qq.com	修改 删除

添加联系人

- 查询数据库里面所有的联系人, 展示在页面

二, 思路分析

1. 在list.html导入vue和axios,创建vue实例
2. 在钩子函数created里面, 使用axios请求LinkManServlet

```
axios.get('linkMan?method=findAll').then(response=>{
   //进行数据的赋值和绑定,展示
})
```

3. 在LinkManServlet的findAll()方法里面, 封装Result 响应json

三, 代码实现

1.准备工作

- 数据库----还是用day28数据库中的linkMan表

- 拷贝day28_findAll模块到项目中---->记得该模块名和配置文件名
- 删除索引jsp页面
- 拷贝资料中的所有html页面到模块中
- 拷贝JsonUtils工具类到utils包
- 拷贝Result类到bean包
- 拷贝FastJson的jar包

2.代码

- list.html

```
<!DOCTYPE html>
<!-- 网页使用的语言 -->
<html lang="zh-CN">
<head>
    <!-- 指定字符集 -->
    <meta charset="utf-8">
    <!-- 使用Edge最新的浏览器的渲染方式 -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- viewport视口：网页可以根据设置的宽度自动进行适配，在浏览器的内部虚拟一个容器，容器的宽度与设备的宽度相同。
    width: 默认宽度与设备的宽度相同
    initial-scale: 初始的缩放比，为1:1 -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>Bootstrap模板</title>

    <!-- 1. 导入CSS的全局样式 -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- 2. jQuery导入，建议使用1.9以上的版本 -->
    <script src="js/jquery-2.1.0.min.js"></script>
    <!-- 3. 导入bootstrap的js文件 -->
    <script src="js/bootstrap.min.js"></script>
    <style type="text/css">
        td, th {
            text-align: center;
        }
    </style>
    <script src="js/vuejs-2.5.16.js"></script>
    <script src="js/axios-0.18.0.js"></script>
```

```
</head>
<body>
<div class="container" id="app">
    <h3 style="text-align: center">显示所有用户</h3>
    <table border="1" class="table table-bordered table-hover">
        <tr class="success">
            <th>编号</th>
            <th>姓名</th>
            <th>性别</th>
            <th>年龄</th>
            <th>籍贯</th>
            <th>QQ</th>
            <th>邮箱</th>
            <th>操作</th>
        </tr>

        <tr v-for="(lm,i) in linkMans">
            <td v-text="lm.id"></td>
            <td v-text="lm.name"></td>
            <td v-text="lm.sex"></td>
            <td v-text="lm.age"></td>
            <td v-text="lm.address"></td>
            <td v-text="lm.qq"></td>
            <td v-text="lm.email"></td>
            <td><a class="btn btn-default btn-sm" href="修改联系人.html">修改
</a> <a class="btn btn-default btn-sm"

  href="修改联系人.html">删除</a></td>
        </tr>

        <tr>
            <td colspan="8" align="center"><a class="btn btn-primary"

 href="${pageContext.request.contextPath }/add.jsp">添加用户</a></td>
        </tr>
    </table>
</div>
</body>
<script>
    // 1.引入vue,axios的js文件
    // 2.创建vue对象
    new Vue({
        el: "#app",
        data: {
            linkMans:[]
        },
        methods: {},
        created: function () {
            // 3.在vue对象的created钩子函数中发送异步请求(get)到LinkManServlet
            axios.post("linkMan?method=findAll").then(response => {
                // console.log(response.data);
                // 3.响应成功后,得到服务器返回的json数据,赋值给数据模型(同步更新到视图---
数据模型和视图进行绑定)
                if (response.data.flag){
                    this.linkMans = response.data.result;
                } else{
                    alert("服务器异常");
                }
```

```
            });
        }
    });
</script>
</html>
```

- LinkManServlet

```
// 查询所有联系人
    private void findAll(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        try {
            //1. 调用service层,获得所有联系人List<LinkMan>
            List<LinkMan> list = service.findAll();

            //3. 响应页面
            if (list != null) {
                // 封装成Result对象
                Result result = new Result(true,"查询所有联系人成功",list);
                // 把result对象转换为json,响应到页面
                JsonUtils.printResult(response,result);
            } else {
                // 封装成Result对象
                Result result = new Result(false,"查询所有联系人失败");
                // 把result对象转换为json,响应到页面
                JsonUtils.printResult(response,result);
            }
        } catch (Exception e) {
            e.printStackTrace();
            // 封装成Result对象
            Result result = new Result(false,"查询所有联系人失败");
            // 把result对象转换为json,响应到页面
            JsonUtils.printResult(response,result);
        }
    }
```

四,小结

1. 在list.html里面, 导入vue和axios 创建vue实例
2. 在钩子函数里面 created()里面,使用axios请求LinkManServlet获得数据 进行赋值,绑定
3. 在LinkManServlet的findAll()方法里面, 封装Result 响应

案例二:添加联系人

一,案例需求

1. 点击添加联系人跳转添加联系人页面

2. 在添加联系人页面，点击提交按钮,把数据提交到服务器,保存到数据库

3. 在添加完成，可以查看到新建的联系人信息

二,思路分析

1. 在add.html 导入vue,axios ,创建vue实例
2. 创建linkMan:{} ,和表单进行绑定
3. 给提交设置一个点击事件,创建函数
4. 在这个函数里面

```
//1.发送ajax请求服务器 携带linkMan
//2.获得响应的结果，  判断是否增加成功
```

三,代码实现

- add.html

```
<!-- HTML5文档-->
<!DOCTYPE html>
<!-- 网页使用的语言 -->
<html lang="zh-CN">
<head>
    <!-- 指定字符集 -->
    <meta charset="utf-8">
    <!-- 使用Edge最新的浏览器的渲染方式 -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- viewport视口：网页可以根据设置的宽度自动进行适配，在浏览器的内部虚拟一个容器，容器的宽度与设备的宽度相同。
    width: 默认宽度与设备的宽度相同
    initial-scale: 初始的缩放比，为1:1 -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>添加用户</title>

    <!-- 1. 导入CSS的全局样式 -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- 2. jQuery导入，建议使用1.9以上的版本 -->
    <script src="js/jquery-2.1.0.min.js"></script>
    <!-- 3. 导入bootstrap的js文件 -->
    <script src="js/bootstrap.min.js"></script>
    <script src="js/vuejs-2.5.16.js"></script>
    <script src="js/axios-0.18.0.js"></script>
</head>
<body>
<div class="container" id="app">
    <center><h3>添加用户页面</h3></center>
    <form>
        <input type="hidden" name="method" value="add"/>
        <div class="form-group">
            <label for="name">姓名：</label>
            <input type="text" v-model="linkMan.name" class="form-control"
id="name" name="name" placeholder="请输入姓名">
        </div>

        <div class="form-group">
            <label>性别：</label>
            <input type="radio"  v-model="linkMan.sex"  name="sex" value="男"
checked="checked"/>男
            <input type="radio"  v-model="linkMan.sex" name="sex" value="女"/>女
        </div>

        <div class="form-group">
            <label for="age">年龄：</label>
            <input type="text"  v-model="linkMan.age" class="form-control"
id="age" name="age" placeholder="请输入年龄">
        </div>

        <div class="form-group">
            <label for="address">籍贯：</label>
```

```
            <select name="address"  v-model="linkMan.address"  class="form-
control" id="jiguan">
                <option value="广东">广东</option>
                <option value="广西">广西</option>
                <option value="湖南">湖南</option>
            </select>
        </div>

        <div class="form-group">
            <label for="qq">QQ：</label>
            <input type="text" v-model="linkMan.qq"  class="form-control"
name="qq" placeholder="请输入QQ号码"/>
        </div>

        <div class="form-group">
            <label for="email">Email：</label>
            <input type="text" v-model="linkMan.email"  class="form-control"
name="email" placeholder="请输入邮箱地址"/>
        </div>

        <div class="form-group" style="text-align: center">
            <input class="btn btn-primary" type="button" value="提交"
@click="add()"/>
            <input class="btn btn-default" type="reset" value="重置" />
            <input class="btn btn-default" type="button" value="返回" />
        </div>
    </form>
</div>
</body>
<script>
    new Vue({
        el: "#app",
        data: {
            linkMan:{}
        },
        methods: {
            add:function () {
                // 发送异步请求到服务器LinkManServlet,携带参数method=add
                axios.post("linkMan?method=add",this.linkMan).then(response=>{
                   // console.log(response.data);
                    if (response.data.flag){
                        // 跳转到list.html
                        location.href="list.html";
                    } else{
                        alert(response.data.message);
                    }
                });
            }
        }
    });
</script>
</html>
```

- LinkManServlet

```
// 添加联系人
```

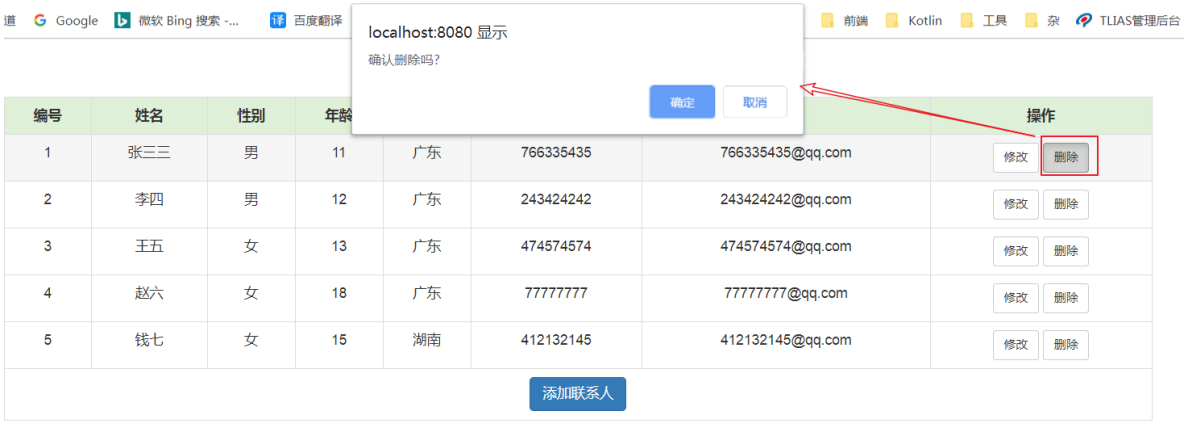
```
    private void add(HttpServletRequest request, HttpServletResponse response)
throws IOException {
        try {
            //1 获取所有提交的表单数据
            LinkMan linkMan = JsonUtils.parseJSON2Object(request,
LinkMan.class);

            //3 调用Service层,传入LinkMan对象,得到boolean值
            boolean flag = service.add(linkMan);

            //4 根据boolean值,响应页面
            if (flag) {
                // 封装数据
                Result result = new Result(true,"添加联系人成功");
                // 转换为json,响应到页面
                JsonUtils.printResult(response,result);
            } else {
                // 封装数据
                Result result = new Result(false,"添加联系人失败");
                // 转换为json,响应到页面
                JsonUtils.printResult(response,result);
            }
        } catch (Exception e) {
            e.printStackTrace();
            // 封装数据
            Result result = new Result(false,"添加联系人失败");
            // 转换为json,响应到页面
            JsonUtils.printResult(response,result);
        }
    }
```

四,小结

1. 在add.html里面 创建linkMan:{}, 和表单进行绑定
2. 点击提交按钮, 把linkMan提交到LinkManServlet

案例三:删除联系人

一,案例需求



点击确定删除之后, 再重新查询所有全部展示,

二,思路分析

1. 点击了确定, 发送axios请求LinkManServlet 携带id
2. 获得响应的结果 判断

三,代码实现

- list.html

```
<a class="btn btn-default btn-sm" @click="deleteById(lm.id)">删除</a>
methods: {
            deleteById:function (id) {
                var flag = confirm("确定要删除id为"+id+"的联系人吗?");
                // 如果flag为true
                if (flag){
                    // 发送axios异步请求--->post方式,携带id
                    axios.post("linkMan?method=delete&id="+id).then(response=>{
                        if (response.data.flag){
                            // 如果删除成功,跳转到list.html
                            location.href = "list.html";
                        } else{
                            alert(response.data.message);
                        }
                    });
                }
            }
        },
```

- LinkManServlet

```
  // 删除联系人
    private void delete(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        try {
            //1 获得id请求参数
            int id = Integer.parseInt(request.getParameter("id"));

            //2 调用service层,传入id.得到boolean类型结果
            boolean flag = service.delete(id);
```

```
            //3 判断boolean值
            if (flag) {
                // 封装
                Result result = new Result(true,"删除成功");
                // 转换为json,响应
                JsonUtils.printResult(response,result);
            } else {
                // 封装
                Result result = new Result(false,"删除失败");
                // 转换为json,响应
                JsonUtils.printResult(response,result);
            }
        } catch (Exception e) {
            e.printStackTrace();
            // 封装
            Result result = new Result(false,"删除失败");
            // 转换为json,响应
            JsonUtils.printResult(response,result);
        }
    }
```

四,小结

1. 点击`删除`弹出确定框
2. 在确定框里面点击了确定 请求LinkManServlet 携带method=delete&id=xxx
3. 在LinkManServlet的dlete方法里面

```
//1.获得id
//2.调用业务删除
//3.响应
```

案例四:更新联系人【作业】

一,案例需求

显示所有联系人

编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三三	男	11	广东	766335435	766335435@qq.com	修改 删除
2	李四	男	12	广东	243424242	243424242@qq.com	修改 删除

二,思路分析

三,代码实现

数据回显

- list.html

```
<a class="btn btn-default btn-sm" :href="'update.html?id='+lm.id">修改</a>
```

- update.html

```
1.页面和数据模型需要绑定-->参考代码  v-model
<script>
        new Vue({
            el: "#app",
            data: {
                linkMan:{}
            },
            methods: {},
            created:function () {
                // 获取id
                var id = getParameter("id");
                // 发送axios异步请求,携带id过去
                axios.post("linkMan?method=findOne&id="+id).then(response=>{
                    if (response.data.flag){
                        // 查询成功,把数据赋值
                        this.linkMan = response.data.result;
                    } else{
                        alert(response.data.message);
                    }
                });
            }
        });
    </script>
```

- LinkManServlet

```
 // 查询要修改的联系人信息
    private void findOne(HttpServletRequest request, HttpServletResponse
response) throws IOException {
```

```
        //1.调用findOne方法
        //2.实现findOne()方法:
        try {
            //2.1 获取要修改的id
            int id = Integer.parseInt(request.getParameter("id"));

            //2.2 调用service层,传入id,得到LinkMan对象
            LinkMan linkMan = service.findOne(id);


            if (linkMan != null) {
                // 封装成Result对象
                Result result = new Result(true,"查询联系人成功",linkMan);
                // 把result对象转换为json,响应到页面
                JsonUtils.printResult(response,result);
            }else {
                // 封装成Result对象
                Result result = new Result(false,"查询联系人失败");
                // 把result对象转换为json,响应到页面
                JsonUtils.printResult(response,result);
            }
        } catch (Exception e) {
            e.printStackTrace();
            // 封装成Result对象
            Result result = new Result(false,"查询联系人失败");
            // 把result对象转换为json,响应到页面
            JsonUtils.printResult(response,result);

        }
    }
```

修改联系人

- 提交按钮绑定点击事件 update.html

```
<input class="btn btn-primary" type="button" value="提交"
@click="update()"/>
```

```
 methods: {
                update:function () {
                    // 发送axios异步请求
                    axios.post("linkMan?
method=update",this.linkMan).then(response=>{
                        if (response.data.flag){
                            location.href="list.html";
                        } else{
                            alert(response.data.message);
                        }
                    });
                }
            },
```

- LinkManServlet

```
 // 修改联系人
```

```
    private void update(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        try {
            //1.调用update()方法
            //2.获取表单所有提交的数据(Map)
            LinkMan linkMan = JsonUtils.parseJSON2Object(request,
LinkMan.class);


            //4.调用service层,传入LinkMan对象,得到boolean结果
            boolean flag = service.update(linkMan);

            if (flag) {
                // 封装
                Result result = new Result(true,"修改成功");
                // 转换为json,响应
                JsonUtils.printResult(response,result);
            }else {
                // 封装
                Result result = new Result(false,"修改失败");
                // 转换为json,响应
                JsonUtils.printResult(response,result);
            }
        } catch (Exception e) {
            e.printStackTrace();
            // 封装
            Result result = new Result(false,"修改失败");
            // 转换为json,响应
            JsonUtils.printResult(response,result);

        }
    }
```

四,小结

总结

必须练习:

1.vue常用指令--->2.1,2.2,2.3,2.4
2.axios发送异步请求---->4.1 axios[重点]
3.综合案例------>第五章
    1.发送axios请求
    2.数据模型和视图绑定---->vue的语法很熟悉
    3.服务器把数据转换为json格式,响应到页面

- 了解vue
  vue可以实现视图和数据模型双向绑定

- 掌握vue常用系统指令
  事件: @click,@keydown,@mouseover,...
  v:text: 绑定标签的文本
  v:html: 绑定标签的标签体

```
    v-bind: 绑定标签的属性
    v:model: 绑定表单元素
    v-for: 循环遍历
    v-if: 根据表达式的值来决定是否渲染元素(标签都没有了)
    v-show: 根据表达式的值来切换元素的display css属性(标签还在)

- 了解vue生命周期
    略

- 掌握vue的ajax的使用
    axios.get(路径,参数).then(...).catch(...)
    axios.post(路径,参数).then(...).catch(...)

- 能够完成显示所有联系人案例
    1.在list.html页面中引入vue,axios库
    2.创建vue对象
    3.在created钩子函数中发送异步请求到LinkManServlet
    4.进行数据模型和视图绑定,显示到页面
    5.在LinkManServlet中封装结果成Result对象,转换为json响应到页面
- 能够完成增加联系人案例
    1.在add.html页面中引入vue,axios库
    2.修改提交按钮为button按钮,为按钮绑定点击事件
    3.创建vue对象
    4.在methods中定义点击事件的函数
    5.在函数中发送axios异步请求到LinkManServlet
    6.如果修改成功跳转到list.html,否则显示错误信息
    7.在LinkManServlet中,将结果封装成Result对象,转换为json响应到页面

- 能够完成删除联系人案例
    1.在list.html页面中为删除按钮绑定一个点击事件,传入要删除元素的id
    2.创建vue对象
    3.在methods中定义点击事件的函数
    4.在函数中发送axios异步请求到LinkManServlet
    5.如果修改成功跳转到list.html,否则显示错误信息
    6.在LinkManServlet中,将结果封装成Result对象,转换为json响应到页面
```

附录

1.键盘ascii码

http://tool.oschina.net/commons?type=4

15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DCI	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	TB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y