Vue进阶二

一、Vue中的表单

1.v-model修饰符

创建表单，并通过v-model绑定data中的属性

```
<template>
  <div id="app">
    <div style="width:50%" class="container">
        <div>
            <h3>Regist</h3>
            <h5>Email</h5>
            <input type="text" class="form-control" v-model.trim="mail" /><br />
            {{mail}}
            <h5>Password</h5>
            <input type="password" class="form-control" v-model.lazy="password" /><br />
            {{password}}
            <h5>Gender</h5>
            <input type="radio" name="gender" v-model="gender" value="female" />男
            <input type="radio" name="gender" v-model="gender" value="male" />女<br />
            <h5>Hobby</h5>
            <input type="checkbox" name="hobby" v-model="hobby" value="music">音乐
            <input type="checkbox" name="hobby" v-model="hobby" value="movie">电影
            <input type="checkbox" name="hobby" v-model="hobby" value="sport">运动
            {{hobby}}
        </div>
    </div>
  </div>
</template>

<script>
export default {
    data(){
        return {
            mail:"xxx@126.com",
            password:"",
            gender:"",
            hobby:[]
        }
    },
    methods:{
        changeName:function(name){
            this.name = name
        }
    }
}
</script>
```

2.使用v-model绑定表单控件，v-model修饰符

- lazy：失去焦点时才会绑定内容
- trim： 绑定的内容自动去除开头和结尾的所有的空格，字符串内部的空格不去除。

二、http请求--ajax

1. 什么是 Axios

Axios 是一个开源的可以用在浏览器端和 NodeJS 的异步通信框架，她的主要作用就是实现 AJAX 异步通信，其功能特点如下：

- 从浏览器中创建 `XMLHttpRequests`
- 从 `node.js` 创建 `http` 请求
- 支持 `Promise API`
- 拦截请求和响应
- 转换请求数据和响应数据
- 取消请求
- 自动转换 `JSON` 数据
- 客户端支持防御 `XSRF` （跨站请求伪造）

GitHub：https://github.com/axios/axios

2.为什么要使用 Axios

由于 Vue.js 是一个 **视图层框架** 并且作者（尤雨溪）严格准守 `SoC` （关注度分离原则），所以 Vue.js 并不包含 AJAX 的通信功能，为了解决通信问题，作者单独开发了一个名为 `vue-resource` 的插件，不过在进入 2.0 版本以后停止了对该插件的维护并推荐了 Axios 框架

3.Axios的使用

1）安装vue axios

```
npm install --save axios vue-axios
```

2）在main.js中引入

在项目中使用axios模块

```
import Vue from 'vue'
import axios from 'axios'
import VueAxios from 'vue-axios'

Vue.use(VueAxios, axios)
```

3）发送ajax请求

```
this.axios({
  method:'get',
  url:'http://bit.ly/2mTM3nY',
  data:{}
})
  .then(function (response) {
    console.log(response.data)
  });
```

4）服务端解决跨域问题

```
<mvc:cors>
    <mvc:mapping path="/**"
        allowed-origins="*"
        allowed-methods="POST, GET, OPTIONS, DELETE, PUT,PATCH"
        allowed-headers="Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-
With"
        allow-credentials="true" />
</mvc:cors>
```

在spring-mvc.xml中加入上述这一段。其中，allowed-origins指的是允许的访问源的域名，"*"表示任何人都可以访问，也可以指明具体的域名

5）解决axios无法传递data中的参数问题

原因：默认情况下发送axios时请求头中的内容类型为：

```
Content-Type:application/json;charset=UTF-8
```

而实际服务端需要的是：

```
Content-Type:application/x-www-form-urlencoded
```

因此，使用axios的qs内置库中的方法进行内容类型的转换。

```
import Qs from 'qs'

this.axios({
    method:'post',
    url:'http://localhost:8081/regist',
    transformRequest: [function (data) {
        return Qs.stringify(data)
    }],
    data:{
        email:this.email
    }
})
.then(function (response) {
    alert(response.data.message)
```

```
});
```

三、路由

路由是第三方模块提供

1.安装路由模块

```
npm install vue-router -s
```

2.设计路由界面

创建components文件夹，文件夹内分别创建user、Home组件

- user.vue

```
<template>
    <div>user</div>
</template>
```

- Home.vue

```
<template>
    <div>Home</div>
</template>
```

3.创建静态路由表

在src下创建routes.js

```
import Home from '@/components/Home.vue'
import User from '@/components/user/user.vue'

export const routes = [
    {path:'/',component:Home},
    {path:'/user',component:User}
]
```

4.引入路由模块并使用

在main.js中引入路由模块并使用

```
import Vue from 'vue'
import App from './App.vue'
import VueRouter from 'vue-router' //1.引入路由模块
```

```
import {routes} from './routes' //2.引入静态路由表

Vue.use(VueRouter); //3.使用路由模块

//4.创建一个VueRouter模块的实例
const router = new VueRouter({
    routes:routes
});

new Vue({
  el: '#app',
  router,//5.把router实例放入到vue实例中
  render: h => h(App)
})
```

5.路由初体验

- App.vue

```
<template>
  <div class="container">
        <div class="row">
            <div class="col-xs-12 col-sm-8">
                <h1>Routing</h1>
                <router-view></router-view>
            </div>
        </div>
    </div>
</template>
```

改变url，发现中的内容发生改变

- http://localhost:8080/#/　显示home
- http://localhost:8080/#/user　显示user

向router实例中添加mode属性：

- 值"hash": url带# 适用于调试模式
- 值"history" url不带#

6.链接路由的实现

创建Header.vue

```
<template>
    <ul class="nav nav-pills">
        <router-link to="/" tag="li" active-class="active" exact><a>Home</a></router-link>
        <router-link to="/user" tag="li" active-class="active"><a>User</a></router-link>
    </ul>
</template>
```

修改App.vue

```
<template>
  <div class="container">
        <div class="row">
            <div class="col-xs-12 col-sm-8">
                <h1>Routing</h1>
                <Home></Home>
                <router-view></router-view>
            </div>
        </div>
    </div>
</template>
<script>
    import Home from '@/components/Header.vue'
    export default{
        components:{"Home":Home}
    }
</script>
```

7.参数的传递

Header.vue传入参数

```
<template>
    <ul class="nav nav-pills">
        <router-link to="/" tag="li" active-class="active" exact><a>Home</a></router-link>
        <router-link to="/user/10" tag="li" active-class="active"><a>User</a></router-link>
    </ul>
</template>
```

路由表：router.js

```
export const routes = [
    {path:'/',component:Home},
    {path:'/user/:id',component:User}
]
```

修改user.vue

```
<template>
    <div>
    <div>user</div>
    <p>userId:{{id}}</p>
    </div>
</template>
<script>
    export default{
        data(){
```

```
            return {
                id:this.$route.params.id
            }
        }
    }
</script>
```

8.程序式路由的实现

在user.vue中加入回到首页按钮

```
<template>
    <div>
        <div>user</div>
        <p>userId:{{id}}</p>
        <button type="button" @click="goHome" class="btn btn-default btn-block">回到首页</button>
    </div>
</template>
<script>
    export default{
        data(){
            return {
                id:this.$route.params.id
            }
        },
        methods:{
            goHome(){
                this.$router.push({path:'/'})
            }
        }
    }
</script>
```