21 我的页面UI及后台接口开发

更新时间：2019-09-04 10:42:27

> 构成我们学习最大障碍的是已知的东西，而不是未知的东西。
>
> —— 贝尔纳

接下来就要进入我的个人页面逻辑开发了，我的个人页面是指的用户自己的页面，可以用来更改头像和昵称等等，入口可以从朋友圈首页的头像点击进来，逻辑相对简单。主要用到的 weui 的 uploader，我们之前多次使用过的，比较容易掌握，下面就进入开发把。

本章节完整源代码地址，大家可以事先浏览一下：

Github-mypage

Github-changenickname

Github-changedesc

Github-user.js

页面效果图

我们先来看一下页面的效果UI图，主要由顶部的navBar和底部的表单元素组成：

开发公用**navHeader**

由于 navHeader 是一个公用组件，使用的页面比较多，所以我们要把代码放在公共组件目录下面，在前端项目的 components 文件夹下新建 navHeader 文件夹，同时新建`index.vue`：

下面这段代码主要是`navHeader`组件的UI，主要包括顶部标题和左侧的返回按钮:

```
<template>
  <div class="header-bar scale-1px">
    <p class="title">{{title}}</p>
    <div class="left-icon" @click="goBack"></div>
  </div>
</template>
```

`goBack()`方法来监听按钮点击：

```
methods: {
  goBack () {
    //表示是返回的标志位
    this.$router.backFlag = true
    // 返回上一页
    this.$router.back()
  }
}
```

这里设置返回标志位`this.$router.backFlag = true`，是为了后面开发页面专场逻辑而使用的标示。

我的页面逻辑开发

这部分UI主要是头像模块，名字模块，性别模块，个性签名模块，电话号码模块和私信跳转按钮模块。
首先是头像模块的UI代码逻辑：

```
<a class="weui-cell weui-cell_access" href="javascript:;" @click="changeAvatar">
  <div class="weui-cell__bd">
    <p class="name">头像</p>
    <div style="display:none" id="uploaderMyAvatar">
      <input
        ref="uploaderInputAvatar"
        id="uploaderInputAvatar"
        class="weui-uploader__input"
        type="file"
        accept="image/*"
      >
    </div>
  </div>
  <img class="avatar" :src="myUser.avatar">
  <div class="weui-cell__ft"></div>
</a>
```

这里同样预埋了`uploader`，目的是提供更换头像的图片上传组件，注意我们之前在很多页面也有预埋`uploader`，这里要保证id即`uploaderMyAvatar`在整个项目的唯一性。
下面这段代码是修改头像的逻辑：

```
/*
* 修改头像
*/
async changeAvatarCallback (obj) {

  let resp = await service.post('users/update', {
    userId: this.myUser._id,
    avatar: obj.data.url
  })

  if (resp.code === 0) {
    //修改成功之后要通知vuex来更新store的数据
    this.$store.dispatch('setUser', {
      ...this.myUser,
      avatar: obj.data.url
    })

    weui.toast('修改成功', 1000)
  }
},
changeAvatar () {
  this.$refs.uploaderInputAvatar.click()
},
```

1. changeAvatar这个方法是用来启动图片上传组件，在点击事件回调中，来触发一下uploader的点击。
2. changeAvatarCallback这个方法是修改头像后的回调，在uploader组件上传图片成功后触发，主要是调用service.post()方法来将头像数据保存到后端。

接下来是名字模块UI代码逻辑：

```
<a class="weui-cell weui-cell_access" href="javascript:;" @click="goChangeName">
  <div class="weui-cell__bd">
    <p class="name">名字</p>
  </div>
  <div class="weui-cell__ft">{{myUser.nickname}}</div>
</a>
```

goChangeName这个方法会跳转到新的页面来修改名字。

接下来是性别模块UI代码逻辑：

```
<a class="weui-cell weui-cell_access" href="javascript:;" @click="goChangeGender">
  <div class="weui-cell__bd">
    <p class="name">性别</p>
  </div>
  <div class="weui-cell__ft">{{myUser.gender == 1 ? '男':'女'}}</div>
</a>
```

goChangeGender这个方法会调用actionSheet组件。

接下来是个性签名模块UI代码逻辑：

```
<a class="weui-cell weui-cell_access" href="javascript:;" @click="goChangeDesc">
  <div class="weui-cell__bd">
    <p class="name">个性签名</p>
  </div>
  <div class="weui-cell__ft">{{myUser.desc}}</div>
</a>
```

goChangeDesc这个方法会跳转到新的页面来修改个性签名。

总结一下：

修改性别使用`weui`的`actionSheet`组件，这里之前有讲过，就不在赘述了，而修改名字，个性签名是挑战到修改页面，代码如下:

修改姓名的actionSheet代码逻辑：

```
weui.actionSheet(
 [
  {
   label: '男',
   onClick: () => {
    this.changeGender(1)
    // console.log('拍照');
   }
  },
  {
   label: '女',
   onClick: () => {
    this.changeGender(0)
    // console.log('拍照');
   }
  }
 ],
 [
  {
   label: '取消',
   onClick: function () {
    // console.log('取消');
   }
  }
 ]
)
```

修改完性别之后，要通知 vuex 来修改`store`的数据，保证数据的实时性，这也是 vuex 的优势所在，代码如下：

```
/*
* 修改性别
*/
async changeGender (gender) {
 let resp = await service.post('users/update', {
  userId: this.myUser._id,
  gender: gender
 })
 //修改成功之后要通知vuex来更新store的数据
 if (resp.code === 0) {
  this.$store.dispatch('setUser', {
   ...this.myUser,
   gender: gender
  })
  weui.toast('修改成功', 1000)
 }
},
```

修改名字，个性签名需要跳转页面，并且将数据带过去，代码如下：

```
goChangeName () {
  this.$router.push({
    name: 'changenickname',
    params: {
      name: this.myUser.nickname
    }
  })
},
goChangeDesc () {
  this.$router.push({
    name: 'changedesc',
    params: {
      desc: this.myUser.desc
    }
  })
},
```

然后在前端项目的 view 文件夹下，新建 changenickname 文件夹，同时新建 index.vue 。

新增修改页面的UI，主要由一个输入框和一个提交按钮组成。另外，需要将之前开发的 navHeader 组件引入，代码如下：

```
<template>
  <div class="container">
    <navHeader title="修改昵称"/>
    <div class="weui-cell">
      <div class="weui-cell__bd">
        <input class="weui-input" type="text" placeholder="请输入文本" v-model="name" maxLength="15">
      </div>
    </div>
    <div class="weui-btn-area">
      <a class="weui-btn weui-btn_primary" href="javascript:" id="showTooltips" @click="submit">确定</a>
    </div>
  </div>
</template>
```

修改完昵称之后，同样要通知 vuex 来修改 store 的数据，保证数据的实时性，代码如下：

```
async submit () {
  let resp = await service.post('users/update', {
    userId: this.$store.state.currentUser._id,
    nickname: this.name
  })

  if (resp.code === 0) {
    this.$store.dispatch('setUser', {
      ...this.$store.state.currentUser,
      nickname: this.name
    })

    weui.toast('修改成功', 1000)
    setTimeout(() => {
      this.$nextTick(() => {
        this.$router.backFlag = true
        this.$router.back()
      })
    }, 400)
  }
}
```

在修改完store数据之后，我们需要返回到上一页，这里利用了setTimeout来增加一个延时，并使用this.$nextTick方法来确保数据修改之后在执行返回操作。这里只展示修改昵称的页面的代码，修改个性签名和这个类似，就不在展示了。

更新个人页面接口

在开发完前端页面之后，就要开发对应接口，在后端项目的 routes 文件夹下，在 users.js 里新增一个路由。

下面这段代码主要是创建了一个post方法的路由，路径是/update，当浏览器请求http://xx.xx.xx/update就会进入这个方法。

```
/*
* 更新个人信息
*/
router.post('/update', async (req, res, next)=> {
  try {
    //根据id找到并更新
    var user = await User.findByIdAndUpdate(req.user._id, req.body).exec();

    res.json({
      code:0,
      data:user
    });
  }catch(e){
    console.log(e)
    res.json({
      code:1,
      data:e
    });
  }


});
```

这里的流程是：

1. 通过当前用户的id直接调用findByIdAndUpdate()来实现User的Model的更新操作。

其中findByIdAndUpdate()是mongoose提供的更新方法，返回的是更新后的结果，类似的方法还有update()，updateMany()，等等。

- update()：更新数据库中的一个文档但不返回它:第一个参数是筛选条件，第二个参数是更新的内容

```
MyModel.update({ age: { $gt: 18 } }, { oldEnough: true }, fn);
MyModel.update({ name: 'Tobi' }, { ferret: true }, { multi: true }, function (err, raw) {
  if (err) return handleError(err);
  console.log('The raw response from Mongo was ', raw);
});
```

- updateMany()：新多个符合匹配条件的文档。

```
MyModel.updateMany({ age: { $gt: 18 } }, { oldEnough: true }, fn);
```

这里可以参考文档来开发。

小结

本章节主要讲解了我的页面的逻辑开发，内容相对简单。

相关知识点：

1. 在开发公用navHeader组件时，我们需要抽离成公共组件，将代码写在在前端项目的components文件夹下，这样其他页面也可以从这里引入。
2. 在点击返回时，需要加上`this.$router.backFlag = true`标识出此次页面切换时返回，在后续进行页面转场开发时会用到。
3. 修改完头像，昵称，和性别之后，都要通知vuex来修改`store`的数据，保证数据的实时性。
4. mongoose 的更新方法讲解。

本章节完整源代码地址：

Github-mypage

Github-changenickname

Github-changedesc

Github-user.js

}

20 首页后台接口开发

22 用户资料页及后台接口开发