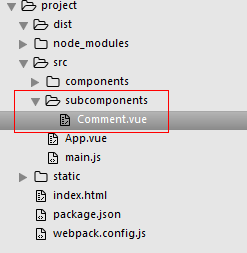
Vue项目步骤第02天

# 1、评论组件Comment.vue的实现

1.1、在src目录下创建subcomponents文件夹

**在此文件夹下创建Comment.vue，并建好对于的vue代码结构**



<template>

<div class="templateDiv">

</div>

</template>

<script>

export default {

}

</script>

<style scoped>

</style>

1.2、在Comment.vue中接收父组件传递过来的参数

<script>

export default {

props:['mynewsid']

}

</script>

1.3、在父组件中使用Comment.vue

**其实也就是在NewsInfo.vue中将Comment.vue进行使用，Comment.vue这个组件并没有匹配路由规则。**

1. **在NewsInfo.vue中导入Comment.vue**

<script>

import Comment from "../../subcomponents/Comment.vue";

export default {

1. **在NewsInfo.vue中注册Comment.vue的对象**

<script>

import Comment from "../../subcomponents/Comment.vue";

export default {

components:{

comment:Comment//前面是我将要使用的标签的名称，后面是组件的对象

},

data:function(){

return {

info:{}

};

1. **在NewsInfo.vue中使用Comment.vue**

在template中，加入<comment></comment>标签即可

1. **传递参数**

在标签中加入参数， 相当于给这个标签的某一个属性赋值。（先用123做测试）

<comment :mynewsid="123"></comment>

1. **在子组件中定义有哪些属性由父组件进行赋值**

在Comment.vue文件中

<script>

export default {

props:["mynewsid"]

}

</script>

在Comment.vue的template中，可以直接使用mynewsid

<template>

<div class="templateDiv">

评论组件{{mynewsid}}

</div>

</template>

123就能够传递到Comment.vue中去了,此时可以把templateDiv的样式去除.

1. **传入真实的新闻id**

给NewsInfo的data增加字段newsId，这个值在数据返回成功之后赋值

将comment标签中的mynewsid通过v-bind的方式来获取newsId的值



1.4、完成Comment.vue的静态界面

<template>

<div>

<div id="postcomment">

<h3>提交评论</h3>

<p class="p"></p>

<textarea placeholder="请输入您要评论的内容..."></textarea>

<mt-button type="primary" size="large">发表</mt-button>

</div>

</div>

</template>

配置好样式

<style scoped>

#postcomment{

padding: 5px;

}

.p{

height: 1px;

width: 100%;

border-bottom: 1px solid rgba(0,0,0,0.3);

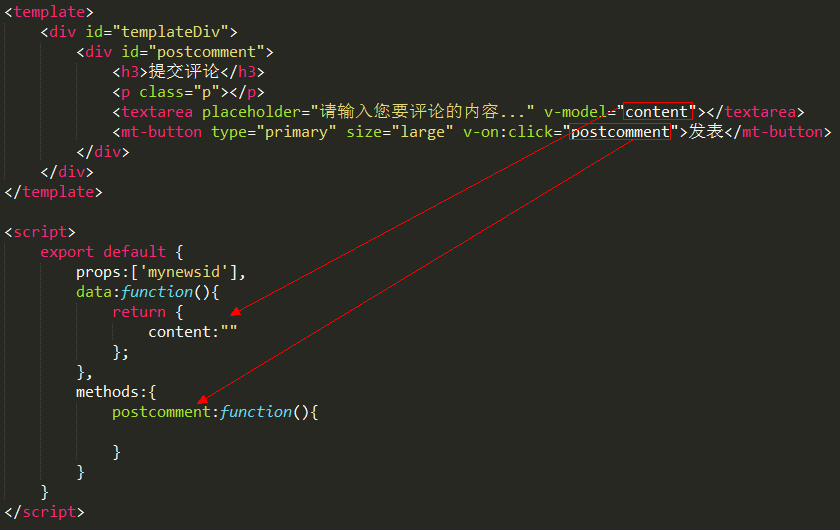
}

</style>

1.5、实现发表评论

**a、给发表按钮绑定事件**

**b、将文本域和vue的data中的属性绑定**



**c、将数据提交到服务器上**

<script>

export default {

props:['mynewsid'],

data:function(){

return {

content:""

};

},

methods:{

postcomment:function(){

var url = this.$urlPrefix + "/api/postcomment/" + this.mynewsid;

this.$http.post(url,{content:this.postcontent},{emulateJSON:true}).then(function(response){

console.log(response.body);

this.content = "";

});

}

}

}

</script>

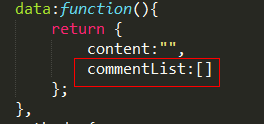
1.6、获取评论列表

**a、得到评论数据**



**b、展示评论列表数据**

在data中定义一个数组数据，由这个数据来渲染评论列表的数据



在template中遍历commentList

<template>

<div id="templateDiv">

<div id="postcomment">

<h3>提交评论</h3>

<p class="p"></p>

<textarea placeholder="请输入您要评论的内容..." v-model="content"></textarea>

<mt-button type="primary" size="large" v-on:click="postcomment">发表</mt-button>

</div>

<div id="list">

<h3>评论列表</h3>

<p class="p"></p>

<div v-for="(item,index) in commentList">

<div class="title">

<span>第{{index + 1}}楼:</span>

<span>用户：{{item.user\_name}}</span>

<span>发表时间：{{item.add\_time | dateFmt('YYYY-MM-DD HH:mm:ss')}}</span>

</div>

<ul class="mui-table-view">

<li class="mui-table-view-cell" v-text="item.content"></li>

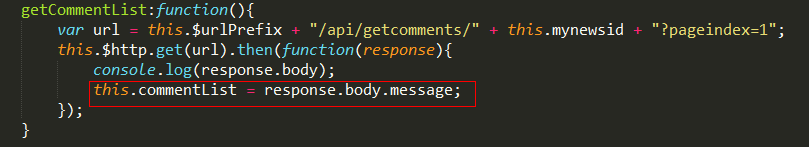
</ul>

</div>

</div>

</div>

数据获取成功之后，对commentList进行赋值



将评论列表的样式配置好

<style scoped>

#postcomment{

padding: 5px;

}

.p{

height: 1px;

width: 100%;

border-bottom: 1px solid rgba(0,0,0,0.3);

}

#list{

padding: 5px;

}

.title{

padding: 5px;

color: #6d6d72;

font-size: 15px;

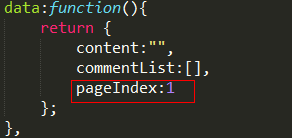
background-color: rgba(0,0,0,0.1);

}

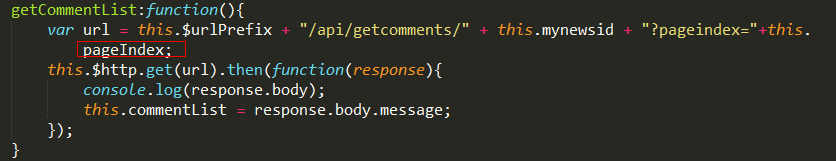
</style>

1.7、评论分页加载

**a、在data中定义一个属性pageIndex，默认值为1**



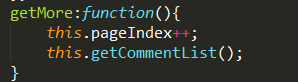
**b、在getCommentList方法中，将原来写死的pageindex=1进行修改**



**c、在template中增加一个加载更多的按钮，并绑定方法点击回调方法getMore**

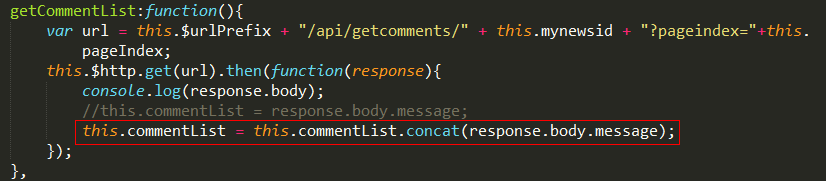
<mt-button type="danger" size="large" plain @click="getMore">加载更多</mt-button>

**d、在methods中增加getMore方法**



1. **修改getCommentList中获取到数据的逻辑**

现在并不是直接将从服务器获取到的数据赋值给commentList，而是应该将服务器获取到的数据追加到commentList中



1.8、服务启动优化

**在main.js的路由规则中，增加如下代码**

{

path:"/",

redirect:"/home"

}

# 2、图片分享的实现

* 1. 图片分类标题界面的实现

**a、标题静态效果**

<div class="category">

<ul>

<li>全部</li>

<li>家居生活</li>

<li>家居生活</li>

<li>家居生活</li>

<li>全部</li>

<li>家居生活</li>

<li>家居生活</li>

<li>家居生活</li>

<li>全部</li>

<li>家居生活</li>

<li>家居生活</li>

<li>家居生活</li>

</ul>

</div>

**b、标题样式设定**

<style scoped>

.category{

width: 100%;

overflow: auto;

}

.category ul{

margin: 0px;

padding-left: 10px;

list-style: none;

width: 2000px;

}

.category li{

cursor: pointer;

color:#0094ff;

display: inline-block;

font-size: 14px;

padding-left: 6px;

width: 60px;

}

</style>

**c、图片分类标题数据获取**

<script>

export default {

data:function(){

return {

categories:[]

}

},

methods:{

getCaterory:function(){

var url = this.$urlPrefix + '/api/getimgcategory';

this.$http.get(url).then(function(response){

this.categories = response.body.message;

});

}

},

created:function(){

this.getCaterory();

}

}

</script>

**d、使用v-for动态填充li的数量**

<ul>

<li>全部</li>

<li v-for="item in categories">{{item.title}}</li>

</ul>

**e、动态计算ul的宽度**



**f、将ulWidth的值作用到ul标签上**

<div class="category">

<ul v-bind="{style:'width:'+ulWidth+'px'}">

<li>全部</li>

<li v-for="item in categories">{{item.title}}</li>

</ul>

</div>

* 1. 图片分类列表界面的实现

1. **图片列表静态效果**

**配置标签**

<div class="imglist">

<ul>

<li>

<a href="#">

<img src="../../../static/test.jpg">

<div class="desc">

<h5>标题</h5>

<p>摘要</p>

</div>

</a>

</li>

<li>

<a href="#">

<img src="../../../static/test.jpg">

<div class="desc">

<h5>标题</h5>

<p>摘要</p>

</div>

</a>

</li>

</ul>

</div>

**配置样式**

.imglist ul {

padding-left: 0px;

list-style:none;

}

.imglist li{

position: relative ;

}

.imglist img{

width:100%;

height: 300px;

}

.desc{

width: 100%;

background-color: rgba(0,0,0,0.2);

position: absolute;

bottom: 2px;

left: 0px;

}

.desc h5{

color: #ffffff;

font-weight: bold;

}

.desc p{

color:#fff;

}

1. **图片列表数据的获取**

<script>

export default {

data:function(){

return {

categories:[],

ulWidth:0,

picList:[]

}

},

methods:{

getCaterory:function(){

var url = this.$urlPrefix + '/api/getimgcategory';

this.$http.get(url).then(function(response){

this.categories = response.body.message;

var w = 62;

var count = response.body.message.length + 1;

this.ulWidth = w \* count ;

});

},

getPicList:function(categoryId){

var url = this.$urlPrefix + '/api/getimages/' + categoryId;

this.$http.get(url).then(function(response){

console.log(response.body);

this.picList = response.body.message;

});

}

},

created:function(){

this.getCaterory();

this.getPicList(0);

}

}

</script>

**使用v-for进行数据填充**

<div class="imglist">

<ul>

<li v-for="item in picList">

<a href="#">

<img v-bind:src="item.img\_url">

<div class="desc">

<h5 v-text="item.title"></h5>

<p v-text="item.zhaiyao"></p>

</div>

</a>

</li>

</ul>

</div>

1. **使用mint-ui的懒加载实现图片列表**

**给img增加v-lazy，由于服务器返回的数据是本地路径，可以在data中增加一个tempurl的地址作为测试**

<img v-lazy="tempurl">

data:function(){

return {

categories:[],

ulWidth:0,

picList:[],

tempurl:"https://timgsa.baidu.com/timg?image&quality=80&size=b9999\_10000&sec=1503257839589&di=2c41f519b7a4cb04483e9e6f0df864b6&imgtype=0&src=http%3A%2F%2Fimg2.cache.netease.com%2Fphoto%2F0026%2F2017-04-10%2FCHML2S0K78LH0026.jpg"

}

},

**增加加载中的样式**

img[lazy=loading] {

width: 40px;

height: 300px;

margin: auto;

}

* 1. 图片详情界面的实现

1. **详情界面的入口**

**在PhotoList.vue中，将图片列表中的a标签变为router-link，并增加to的属性，值为/photo/photoinfo**

<div class="imglist">

<ul>

<li v-for="item in picList">

<router-link to="/photo/photoinfo">

<img v-lazy="tempurl">

<div class="desc">

<h5 v-text="item.title"></h5>

<p v-text="item.zhaiyao"></p>

</div>

</router-link>

</li>

</ul>

</div>

**在Photo文件夹下创建PhotoInfo.vue文件，并配置好对应的代码结构**

<template>

<div class="templateDiv">

图片详情

</div>

</template>

<script>

export default {

data:function(){

return {

}

},

methods:{

},

created:function(){

}

}

</script>

<style scoped>

</style>

**在main.js中配置路由规则**

import PhotoInfo from "./components/photo/PhotoInfo.vue";

{

path:"/photo/photoinfo",

component:PhotoInfo

}

1. **详情界面参数获取**

**在main.js中，规定/photo/photoinfo这个路径是需要参数的**

{

path:"/photo/photoinfo/:id",

component:PhotoInfo

}

**在PhotoInfo.vue中的created对参数进行获取，并复制给自身的属性id**

<script>

export default {

data:function(){

return {

id:0

}

},

methods:{

},

created:function(){

this.id = this.$route.params.id;

}

}

</script>

**在PhotoList.vue中将参数进行传递**

<li v-for="item in picList">

<router-link v-bind="{to:'/photo/photoinfo/'+item.id}">

<img v-lazy="tempurl">

<div class="desc">

<h5 v-text="item.title"></h5>

<p v-text="item.zhaiyao"></p>

</div>

</router-link>

</li>

1. **详情界面数据获取**

<script>

export default {

data:function(){

return {

id:0,

photoinfo:{}

}

},

methods:{

getinfo:function(){

var url =this.$urlPrefix +'/api/getimageInfo/'+this.id;

this.$http.get(url).then(function(response){

console.log(response.body);

this.photoinfo = response.body.message[0];

});

}

},

created:function(){

this.id = this.$route.params.id;

this.getinfo();

}

}

</script>

1. **标题界面的实现**

<template>

<div class="templateDiv">

<div class="title">

<h4>{{photoinfo.title}}</h4>

<p>{{photoinfo.add\_time | datefmt('YYYY-MM-DD HH:mm:ss')}} {{photoinfo.click}}次浏览</p>

<p class="line"></p>

</div>

<p v-html="photoinfo.content"></p>

</div>

</template>

**样式设置**

<style scoped>

.title h4{

color: #0094ff;

}

.title p{

color:rgba(0,0,0,0.4);

margin-top: 10px;

}

.title .line{

width: 100%;

height: 1px;

border-bottom: 1px solid rgba(0,0,0,0.4);

}

</style>

1. **使用Comment.vue组件**



