13 开发存储帖子接口

更新时间：2019-09-02 14:02:08

> 不安于小成，然后足以成大器；不诱于小利，然后可以立远功。
>
> ——方孝孺

在完成了相关的发表界面UI开发后，我们就开始发开对应的后台的接口，整个后台逻辑相对简单，基本上是Post的处理逻辑，下面就进入开发吧。

创建 Post 的 model

首先需要了解一下 Post 的表结构。

Post表结构：

字段名	类型
_id	objectID（主键）
帖子内容：content	String
图片内容：picList	Schema.Types.Mixed
创建时间：create	Date
更新时间：update	Date
发布人：user	Schema.Types.ObjectId

Post表示每条朋友圈的对象模型，我们使用mongoose来创建PostModel。

在后端项的 models 下新建 `Post.js`：

```
var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var PostSchema = new mongoose.Schema({
  content: { type: String,required:true },//朋友圈文本内容
  picList: { type: Schema.Types.Mixed },//朋友圈图片内容
  create: { type: Date, default: Date.now },//创建时间
  update: { type: Date, default: Date.now },//更新时间
  user:{ type: Schema.Types.ObjectId, ref: 'User',required:true }//朋友圈的Users外键属性，标识谁发的朋友圈
},{timestamps:{createdAt: 'create',updatedAt:'update'}});

module.exports = mongoose.model('Post', PostSchema);
```

ref 表示关联 User，相当于 Post 里有一列是 user，它的值是 User 的 ObjectId(userid)，在后面的查询时，可以通过 populate() 轻松获取到Post关联的User对象信息。

required：true 表示此列为必填项，如果在创建时没有，将会报错。

创建朋友圈post接口：

在 routes 文件夹下的 post.js 中创建了一个post方法的路由，路径是 /savepost，当浏览器请求 http://xx.xx.xx/savepost 就会进入这个方法。

```
/*
* 创建朋友圈post
*/
router.post('/savepost', async function(req, res, next) {

 //获取到当前用户的id
 var userid = req.user._id;
 //从req.body里获取content和picList
 /*{
   "content": "hello",
   "picList": [{
     "url": "//wecircle.oss-cn-beijing.aliyuncs.com/image-1559201643093.png",
     "size": {
       "width": 933,
       "height": 563,
       "type": "png"
     },
     "id": 1
   }]
 }*/
 var p = {
   content: req.body.content,
   picList: req.body.picList,
   user:userid//user和post关联
 };

 try {
   //调用PostModel的静态方法create
   //await方式返回保存后的Post对象，如果发生错误将会进入catch方法
   var result = await Post.create(p);
   res.json({
     code:0,
     data:result
   });
 }catch(e){
   res.json({
     code:1,
     data:e
   });
 }


});
```

上面这段代码的流程主要是：

1. 前端页面将content内容和picList图片内容通过post方法传到后端。
2. 在后端通过获取当前用户的id同时将数据进行组装校验。
3. 调用 Post.create() 将数据保存。

我们在创建 Post 时将当前用户的 User 的 id 放在 Post 的 user 字段上，就完成了 Post 和 User 的关联。

这样我们在查询 Post 时，使用 populate('user') (后面章节会有讲解)，就可以直接将用户数据查出来，结构如下：

```
{
  "_id": "5cf7d0c65d68df68afb51f94",
  "content": "Hello",
  "picList": [],
  "user": {
    "_id": "5cdbfba26db2f4663abd3cb0",
    "nickname": "吕小鸣的宝贝",
    "avatar": "//app.nihaoshijie.com.cn/upload/avatar/avatar6.jpg",
    "gender": "0",
    "bgurl": "//app.nihaoshijie.com.cn/upload/bg/topbg3.jpg",
    "phoneNum": "13526405082",
    "update": "2019-05-15T11:47:35.902Z",
    "create": "2019-05-15T11:44:34.642Z",
    "__v": 0
  },
  "create": "2019-06-05T14:25:10.174Z",
  "update": "2019-06-05T14:25:10.174Z",
  "__v": 0,
  "comments": [],
  "likes": [],
  "isLike": false
}
```


小节

本章主要讲解了开发创建朋友圈Post接口。

相关知识点：

1. 我们将朋友圈数据的图片数据存储在了 `picList` 字段中，这个字段采用 `Schema.Types.Mixed` 方式，可以直接存储一个js数组，方便使用。
2. 使用mongoose的 `create` 方法来存储一个Post数据。
3. 以及如何在mongoose利用 `ref` 将两个model(表)进行关联。

本章节完整源代码地址：

Github1

Github2

}

12 upload 图片上传接口

14 顶部组件 UI 开发