20 首页后台接口开发

更新时间：2019-09-02 09:57:13

> 困难只能吓倒懦夫懒汉，而胜利永远属于敢于等科学高峰的人。
>
> ——茅以升

在完成了复杂的朋友圈首页的前端逻辑和 UI 开发之后，如果大家都能够理解，那么大家已经很棒了。下面我们就要进入朋友圈后台的接口开发，相关的接口包括朋友圈列表数据，点赞和评论接口，其中会涉及到使用 mongoose 来进行分页，连接查询，下面就进入开发吧。

本章节完整源代码地址，大家可以事先浏览一下：

Github-Comment.js

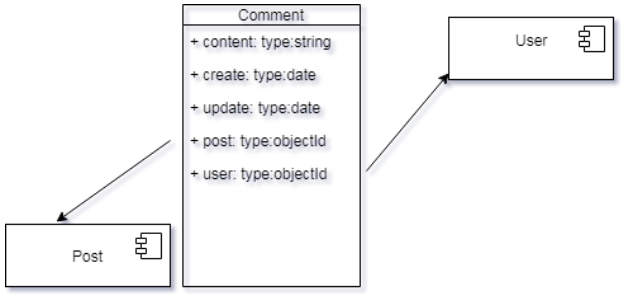
Github-Like.js

Github-post.js

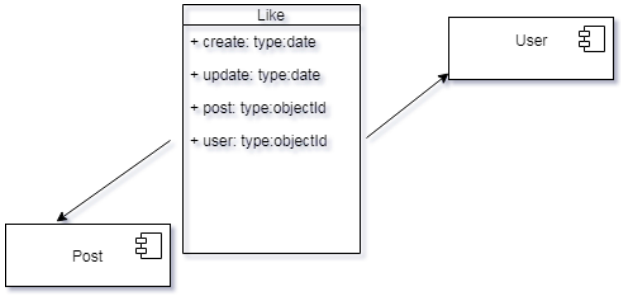
创建 Comment 和 Like 的 Model：

我们在之前发表章节中已经创建了朋友圈的表 Post，而这个表需要和点赞和评论有关联，我们需要新建 2 张表：Comment 和 Like：

Comment:



Like:



这里的关联逻辑就是，某个评论和点赞的数据必须要隶属一个 Post 表，然后在后端项目的 models 文件夹下新建两个 model 文件：

Comment:

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

var Schema = mongoose.Schema;

var CommentSchema = new mongoose.Schema({
  content: String,
  post:{ type: Schema.Types.ObjectId, ref: 'Post',required:true },
  user:{ type: Schema.Types.ObjectId, ref: 'User',required:true },
  create: { type: Date, default: Date.now },
  update: { type: Date, default: Date.now },
}, {timestamps:{createdAt: 'create',updatedAt:'update'}});



module.exports = mongoose.model('Comment', CommentSchema);
```

Like:

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

var Schema = mongoose.Schema;

var LikeSchema = new mongoose.Schema({
  post:{ type: Schema.Types.ObjectId, ref: 'Post',required:true },
  user:{ type: Schema.Types.ObjectId, ref: 'User',required:true },
  create: { type: Date, default: Date.now },
  update: { type: Date, default: Date.now },
}, {timestamps:{createdAt: 'create',updatedAt:'update'}});

LikeSchema.index({ user: 1, post: 1 }, { unique: true });

module.exports = mongoose.model('Like', LikeSchema);
```

这里采用 mongoose 的 ref 属性来给 post 键设置成外键。这样就到到了两表关联。

分页查询朋友圈 post：

然后，我们就要开发第一个接口，查询朋友圈 Post 列表数据，在后端项目的 routes 文件夹下的 post.js 文件，新增 `getcirclepost` 接口：
下面这段代码主要是创建了一个 get 方法的路由，路径是 /getcirclepost，当浏览器请求 `http://xx.xx.xx/getcirclepost` 就会进入这个方法：

```
/*
* 获取朋友圈列表数据
*/
router.get('/getcirclepost', async function(req, res, next) {

  //分页pageSize，也可以从前端传进来
  var pageSize = 5;
  //分页pageSizepageStart
  var pageStart = req.query.pageStart || 0;

  //先查出post
  var posts = await Post.find().skip(pageStart*pageSize).limit(pageSize).populate('user').sort({'create':-1}).exec();

  ...

});
```

这里解释一下我们在代码中用到的相关知识点，在 mongoose 中，可以调用 `skip` 和 `limit` 来实现分页查询：

- skip：接收一个整数参数，表示查询需要跳过的条数，通过此参数就可以设置查询的起始点下标。
- limit：接收一个整数参数，表示查询的条数。
- pageSize：表示每次查询的条数，从前端传进来也可以自己在后端写死，传到 pageSize 里面。
- pageStart：从前端传进来的参数（前端每次翻页自增），pageStart * pageSize 就可以获取每次查询的起始点下标了。
- exec ()：这个方法会返回一个 `Promise`，为了配合 `await` 使用。

通过之前章节 Post 和 user 的关联设置，我们先来看看这次通过关联两表得到的数据，
`populate()` 方法表示连接查询，参数是我们创建 PostModel 时用 `ref` 设置的外键字段，查询的结果如下：

```
{
  "_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
}
```


其中 user 关联的数据已经被查出来了，而 likes 和 comments 目前默认给一个空数组，下面就根据 post 的 id 来关联查询出 likes 和 comments 的数据。

查询 post 关联的点赞和评论：

由于朋友圈的数据是一个融合数据的集合，所以流程是先查出 post 列表数据，然后在根据列表里的数据查出每个 post 关联的点赞和评论：
还是在后端项目的 routes 文件夹下的 post.js 文件中的 `router.get('/getcirclepost')`，新增逻辑：

```
...
var result = [];

for (var i = 0 ; i < posts.length ; i++) {
  //根据post查comments
  var comments = await getCommentByPost(posts[i]);
  //根据post查likes
  var likes = await getLikeByPost(posts[i]);

  //这里对数据做一次拷贝，否则无法直接给数据添加字段
  var post = JSON.parse(JSON.stringify(posts[i]));
  //将数据组装到post列表里
  post.comments = comments || [];
  post.likes = likes || [];
  //判断是否点过赞
  post.isLike = checkPostIsLike(likes, req.user);

  result.push(post);
}
...
```

将查询关联的点赞和评论的方法抽离出来，在后端项目的 routes 文件夹下的 post.js 文件中，新增逻辑：

```
/*
* 根据post查询评论数据
*/
var getCommentByPost = async function(post){
  return Comment.find({post:post._id}).populate('user').sort({'create':1}).exec();
}
/*
* 根据post查询点赞数据
*/
var getLikeByPost = async function(post){
  return Like.find({post:post._id}).populate('user').sort({'create':1}).exec();
}
/*
* 根据post是否被当前用户点赞
*/
var checkPostIsLike = function(likes,currentUserId){

  if (!currentUserId) return false;
  var flag = false;
  for (var i = 0 ; i < likes.length ; i++) {
    if (likes[i].user._id == currentUserId._id) {
      flag = true;
      break;
    }
  }
  return flag;
}
```

sort() 方法可以通过参数指定排序的字段，并使用 1 和 - 1 来指定排序的方式，其中 1 为升序排列，而 - 1 是用于降序排列。

我们来看看最终的完整数据：


```
{
  "_id": "5d60938afc238a1d7213fa4f",
  "content": "vue真的很好用",
  "picList": [{
    "url": "//wecircle.oss-cn-beijing.aliyuncs.com/image-1566610307267.png",
    "size": {
      "width": 400,
      "height": 400,
      "type": "png"
    },
    "id": 1
  }],
  "user": {
    "desc": "",
    "params": {
      "vip": 0
    },
    "_id": "5d609353fc238a1d7213fa4e",
    "nickname": "用户1566610259776",
    "avatar": "//app.nihaoshijie.com.cn/upload/avatar/avatar4.jpg",
    "gender": "1",
    "bgurl": "//app.nihaoshijie.com.cn/upload/bg/topbg3.jpg",
    "phoneNum": "18549903397",
    "update": "2019-08-24T01:30:59.776Z",
    "create": "2019-08-24T01:30:59.776Z",
    "__v": 0
  },
  "create": "2019-08-24T01:31:54.584Z",
  "update": "2019-08-24T01:31:54.584Z",
  "__v": 0,
  "comments": [{
    "_id": "5d60b293fc238a1d7213fa56",
    "post": "5d60938afc238a1d7213fa4f",
```

```
    "user": {
      "desc": "",
      "params": {
        "vip": 0
      },
      "_id": "5d60b28afc238a1d7213fa55",
      "nickname": "用户1566618250915",
      "avatar": "//app.nihaoshijie.com.cn/upload/avatar/avatar2.jpg",
      "gender": "1",
      "bgurl": "//app.nihaoshijie.com.cn/upload/bg/topbg4.jpg",
      "phoneNum": "18779105857",
      "update": "2019-08-24T03:44:10.916Z",
      "create": "2019-08-24T03:44:10.916Z",
      "__v": 0
    },
    "content": "哈哈哈",
    "create": "2019-08-24T03:44:19.817Z",
    "update": "2019-08-24T03:44:19.817Z",
    "__v": 0
  }],
  "likes": [{
    "_id": "5d60f808fc238a1d7213fa5e",
    "post": "5d60938afc238a1d7213fa4f",
    "user": {
      "desc": "",
      "params": {
        "vip": 0
      },
      "_id": "5d60f6f4fc238a1d7213fa5a",
      "nickname": "用户1566635764572",
      "avatar": "//app.nihaoshijie.com.cn/upload/avatar/avatar4.jpg",
      "gender": "1",
      "bgurl": "//app.nihaoshijie.com.cn/upload/bg/topbg1.jpg",
      "phoneNum": "16601294856",
      "update": "2019-08-24T08:36:04.572Z",
      "create": "2019-08-24T08:36:04.572Z",
      "__v": 0
    },
    "create": "2019-08-24T08:40:40.996Z",
    "update": "2019-08-24T08:40:40.996Z",
    "__v": 0
  }],
  "isLike": false
}
```


点赞接口和评论接口：

接下来就要开始开发对应的点赞和评论接口了，在后端项目的 routes 文件夹下 post.js 新建 `likecomment.js` 路由：

```
/*
* 点赞
*/
router.post('/addlike', async (req, res, next)=> {

  var postId = req.body.postId;
  var userId = req.user._id;
  try {
    var result = await Like.create({
      post: postId,
      user: userId,
    });

    res.json({
      code:0,
      data:result
    });
  }catch(e){
```

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

});
/*
* 取消点赞
*/
router.post('/removelike', async (req, res, next)=> {

  var postId = req.body.postId;
  var userId = req.user._id;
  try {
    var result = await Like.deleteOne({
      post: postId,
      user: userId,
    });
    res.json({
      code:0,
      data:result
    });
  }catch(e){
    console.log(e);
    res.json({
      code:1,
      data:e
    });
  }

});
/*
* 添加评论
*/
router.post('/addcomment', async (req, res, next)=> {
  var postId = req.body.postId;
  var userId = req.user._id;
  var content = req.body.content;
  try {
    var result = await Comment.create({
      post: postId,
      user: userId,
      content: content,
    });
    res.json({
      code:0,
      data:result
    });
  }catch(e){
    console.log(e);
    res.json({
      code:1,
      data:e
    });
  }

});
```

这个逻辑相对来时比较简单，主要是对一个 Model 的创建和删除，很好理解，之前章节都有讲到，这里就不在赘述了，这里注意一下 `create()` 方法返回的是一个 `promise`，所以不需要调用 `exec()` 方法了。

小结

本章节主要讲解朋友圈的后台接口，包括朋友圈列表数据，点赞，评论接口。

相关技术点：

1. 利用 `mongoose` 接口开发数据的查询逻辑， `populate()` 方法表示连接查询，参数是我们创建 PostModel 时用 `ref` 设置的外键字段。
2. 利用 `mongoose` 中，可以调用 `skip` 和 `limit` 来实现分页查询。

本章节完整源代码地址，大家可以事先浏览一下：

Github-Comment.js

Github-Like.js

Github-post.js

}

19 评论输入框组件开发

21 我的页面UI及后台接口开发