08 后端登录接口开发

更新时间：2019-08-06 16:35:26

> 每个人的生命都是一只小船，理想是小船的风帆。
>
> ——张海迪

本章节将要讲解登录页面相关的接口开发，主要用到了 mongoose的Schema/Model/Entity 的相关概念以及API，比如如何创建一个 Schema，如何操作 Model 。同时将会讲解到Express框架如何编写一个路由，来提供前端页面的接口。这是我们编写的第一个后端接口，大家一定要掌握好哦。

使用 mongoose 初始化数据库连接

常见的后端语言与数据库交互有大概两种方法：

1. 使用数据库的原生查询语言（例如SQL）。
2. 使用对象数据模型（Object Data Model，简称 ODM）或对象关系模型（Object Relational Model，简称 ORM）。 ODM / ORM 能将网站中的数据表示为 JavaScript 对象，然后将它们映射到底层数据库。一些 ORM 只适用某些特定数据库，还有一些是普遍适用的。

MongoDB本身是一种非关系型 NoSQL 数据库，而 mongoose 是 MongoDB 的一个对象模型工具，封装了 MongoDB 对数据的一些增删改查等常用方法，是最受欢迎的操作 MongoDB 数据库的 ODM 工具，这使得 JavaScript 程序员可以继续用 JavaScript 对象的思维而不用转向数据库语义的思维，让 Node.js 操作 MongoDB 数据库变得更加容易。

安装 mongoose：

```
npm install mongoose --save
```

在 wecircleServer 的 app.js 初始化数据库连接：

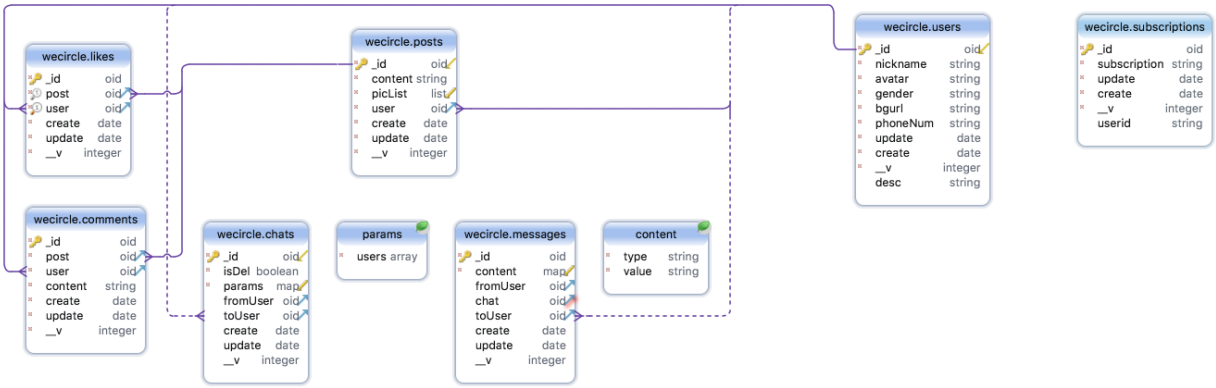
```
var mongoose = require('mongoose');

// 开启数据库连接
mongoose.connect('mongodb://127.0.0.1:27017/wecircle',{ useNewUrlParser: true ,useCreateIndex: true})
  .then(function(){
    console.log('数据库wecircle连接成功');
  })
  .catch(function(error){
    console.log('数据库wecircle连接失败：' + error);
  });
```

我们在前一章节安装了 MongoDB，并且保证已经启动了 MongoDB 服务，MongoDB 默认的端口是 27017。

指定 `useNewUrlParser: true` 和 `useCreateIndex: true` 是为了消除警告，意思是这些接口在未来会被移除，这里需要做一下兼容。

`wecircle` 是数据库的名称。

下图是我们整个项目的所有数据表结构逻辑图，在这里我们先将这张图放出，我们后面会一一讲解到。



mongoose的Schema/Model/Entity：

- **Schema**：数据库集合的结构对象，类似于关系型数据库中的表结构，它有下面几种类型：
  - String：字符串类型
  - Number：数字类型
  - Date：日期类型
  - Boolean：布尔类型
  - Buffer：二进制类型
  - ObjectId：id类型
  - Mixed：混合类型
  - Array：数组类型
- **Model**：由 Schema 构造而成，具有操作数据库的静态方法。
- **Entity**：由 Model 创建的实例，具有操作数据库的实例方法。

这里解释一下 `Mixed` ：

混合类型，是一个很灵活的类型，简单来说你可以直接将一个 json 对象传给它。我们在开发后面的帖子模块的时候会用到。

下面来看一下它的具体结构：

```
mixed:{
  foo:[1,2,3],
  boo:{
     a:1
  }
}
```

也就是说你可以给 Mixed 类型传递一个 json 对象，当你查询时，就会查出一个 json 对象，使用起来非常方便，并且扩展性强。

下面开始创建用户表：

在创建用户表之前先来看下表结构，用户表需要以下的几个字下图是字段各自的类型：

字段名	类型
_id	objectID（主键）
nickname(昵称)	string
avatar(头像url地址)	string
desc(个性签名)	string
gender(性别)	string
bgurl(朋友圈背景图片url地址)	string
phoneNum(电话号码)	string
create(创建日期)	date
update(更新日期)	date

在根目录新增 models 文件夹，同时在 models 文件夹下新建 User.js 文件:

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

var Schema = mongoose.Schema;

var UserSchema = new mongoose.Schema({
  nickname: { type: String, maxlength: 20 },
  avatar: String,
  bgurl:String,
  phoneNum: String,
  desc: { type: String, maxlength: 20 ,default:''},
  gender:String,
  update: { type: Date, default: Date.now },
  create: { type: Date, default: Date.now },
},{timestamps:{createdAt: 'create',updatedAt:'update'}});

module.exports = mongoose.model('User', UserSchema);
```

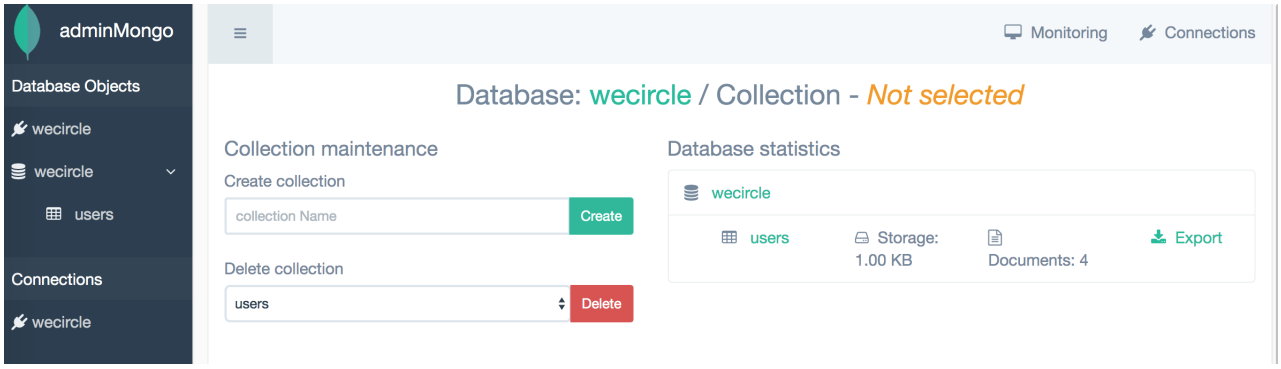
这个就是我们创建的 User 的 Schema ，最后一行我们调用 mongoose.model('User', UserSchema) 将 Schema 转换成了 Model 。其中：

- update 和 create 表示存储创建和更新的时间戳采用 Date 类型，默认值 default:Date.now 表示当前时间。
- {timestamps:{createdAt: 'create',updatedAt:'update'}} 表示添加了一个钩子，当 Model 发生创建和更新时，会自动赋值当前的时间戳到这两个字段。
- maxlength 表示最大长度，如果进行创建操作，超过这个值会报错。

这里需要注意一下在 Node.js 里存入的是 GMT 标准时间，我们在读取的时候需要转换成当前的时区时间，这里我们后续在前端进行转。

我们可以在之前讲解到的 adminMongo 里，查看 User 的数据和内容：

启动 adminMongo 之后在浏览器里打开：http://localhost:1234/



如上图，可以看到具体的数据内容。

创建users路由

在后端项目的 routes 文件夹下新建 users.js 文件(如果没有需要新建)：

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

```
/*
* 获取手机验证码
*/
router.post('/phonecode', (req, res, next)=> {

  //调用阿里云短信接口
  sms.sendSms({
    //从body里获取电话号码
    phoneNum: req.body.phoneNum,
  },function(result){
    //返回成功
    res.json({
      code:0,
      data:result
    });
  },function(result){
    //返回失败
    res.json({
      code:1,
      data:result
    });
  });

});
```

首先我们编写的 sms.sendSms 是我们封装的调用阿里云短信服务的模块，后面会讲解到，这里讲解一下 Express 框架获取路由页面参数的方法。

在 Express 框架里，获取路由参数的方法如下：

```
req.query.xxx  //get方法：从url获取参数
req.body.xxx  //post方法：从body获取参数
```

同时需要在后端项目的 app.js 里添加下面两行代码来使 post 可以接收到 json 数据：

```
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
```

在 Express 框架里，返回 json 数据：

```
req.json()  //将数据以json格式返回
```

其中 `req.json()` 等价于:

```
req.set('Content-Type', 'application/json');
req.send();
```

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

```
/*
* 注册接口
*/
router.post('/signup', async (req, res, next)=> {

  //检查验证码是否合法
  sms.checkCode({
    phoneNum: req.body.phoneNum,
    code: req.body.code,
  },async function(result){

    if (result.code === 0) {
      //是否是已经存在的用户
      var user = await checkUser(req.body);
      //没有的话创建一个新的用户
      if (!user) {
        user = await createUser(req.body);
      }
      //将用户数据设置在cookie里面
      token.setToken({user,res});
      //返回当前用户数据
      res.json({
        code:0,
        data:user
      });
    } else {
      //验证码值不合法返回错误信息
      res.json({
        code:1,
        data:result.msg
      });
    }

  }, function(result){
    //验证码值不合法返回错误信息
    res.json({
      code:1,
      data:result.msg
    });
  });

});
```

上面这段代码逻辑比较简单，从注释中就可以读懂，解释一下就是：

1. 首先判断前端的验证码是否正确。
2. 正确的话，其次判断是否是一个新用户，如果是就创建一个用户并返回前端登录成功。如果不是新用户，就查出用户信息并返回前端登录成功。
3. 失败的话，就返回登录失败。

小结

本章节主要讲解登录页面相关的接口开发，同时讲解Express框架如何编写一个路由，来提供前端页面的接口。
相关技术点：

1. 什么是mongoose，和MongoDB区别。
2. mongoose的Schema/Model/Entity的相关概念以及API。
3. Schema的type类型，特殊类型 Mixed 相关概念。
4. Express中路由获取参数的方法，包括get和post。
5. Node.js 操作数据库时，存入的是GMT标准时间，需要在前端做一次转换。

```
}
```

vuex 和 async/await 使用说明

09 阿里云短信服务