

使用秒滴短信验证接口实现注册或登录功能

需要的模块: **npm i moment md5 axios -S** 安装方法

生成时间戳, 时间格式化 moment

加密 md5

网络请求 axios

get参数编码 querystring

使用:

```
const moment = require('moment');//生成时间戳使用
```

```
const md5 = require('md5');//生成时间戳使用
```

```
const axios = require('axios');//发送http请求
```

```
const qs = require('querystring');//参数编码 自带的
```

格式化时间:

```
const now = moment();
```

```
const timestamp = now.format('YYYYMMDDHHmmss');
```

加密:

```
const sig = md5(accountSid + authToken + timestamp)
```

网络请求:

```
//参数1: url, 参数2: 参数, 参数3: config
```

```
//返回Promise: 可用于await, 也可以then()中处理结果
```

```
const resp = axios.post(url,  
  qs.stringify({to, accountSid, templateid, param, timestamp, sig}),  
  {headers: {'Content-Type': 'application/x-www-form-urlencoded'}}  
);
```

## session处理

安装: **npm i -S express-session**

配置: app.js的中间件处注册session, 注册前先要注册cookie

**its a secret**: 就是对cookie数据的加密，可以是任意值

//cookie解析

```
app.use(cookieParser('its a secret'));
```

// 配置session，需要在cookie下面

```
const session = require('express-session');
```

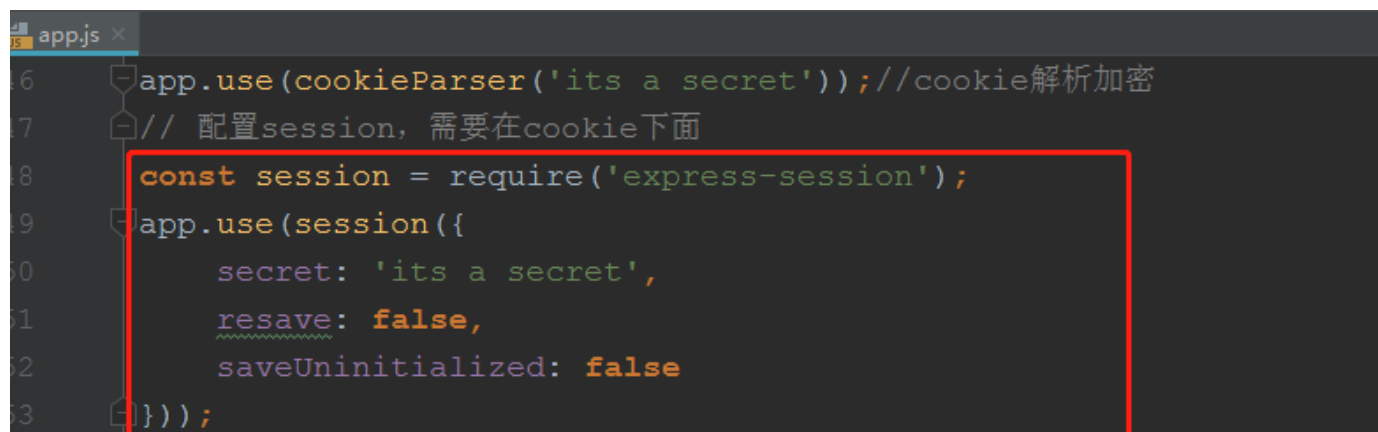
```
app.use(session({
```

```
  secret: 'its a secret',    //加密
```

```
  resave: false,            //强制保存session会话数据到内存中
```

```
  saveUninitialized: false  //保存未初始化的session数据到内存中
```

```
}))
```




```
app.js x
6 app.use(cookieParser('its a secret'));//cookie解析加密
7 // 配置session，需要在cookie下面
8 const session = require('express-session');
9 app.use(session({
10   secret: 'its a secret',
11   resave: false,
12   saveUninitialized: false
13 }));
```

保存数据到session域:

// 登录成功需要在会话中保存登录状态

```
req.session.user = user;
```



```
const u = result[0];
delete u.password;
// 登录成功需要在会话中保存登录状态
req.session.user = user;
res.json({success:true,data:u});
```

获取session域存的数据:

使用

- 赋值 req.session.xx = xx;
- 获取 req.session.xx

- 删除 delete req.session.xx

## 将session信息存入数据库

(在使req.session.xx = xx;时候会自动在数据库建表并将信息保存到数据库中)

1、安装: npm i -S express-mysql-session

2、在app.js中引入

```
// 配置session, 需要在cookie下面
const session = require('express-session');
const Store = require('express-mysql-session')(session);
const {pool} = require('./models/mysql_ljc'); //引入数据库连接池对象
const store = new Store(null, pool);
app.use(session({
  store, // 设置session存储为mysql, 注意当前数据库用户需要表创建权限
  secret: 'its a secret',
  resave: false,
  saveUninitialized: false
})));
```

// 连接池 注意上面的pool是多数据库连接池中导出的

```
const pool = mysql.createPool(cfg);
```

```
module.exports = {
  query: function (sql, value) {
    return new Promise((resolve, reject) => {
      // pool.getConnection((err, conn) => {
      //
      //   conn.release();
      // })

      // resolve函数在异步操作成功时执行
      // reject函数在异步操作失败时执行
      pool.query(sql, value, (err, results) => {
        if (err) reject(err);
        else resolve(results);
      })
    })
  },
  pool
}
```



# 图形验证码提供

1、用到的库：**npm i -S trek-captcha**

2、使用：当前端发起网络请求要获取图形验证码时，后端在路由中通过库生后一个4位的图形验证码响应给前端

```
// token:是数字字母表示形式
// buffer:是图片数据
// size 是生成验证码的长度或者说是个数

const captcha = require('trek-captcha');//图形验证码的库
const moment = require('moment');//生成时间戳使用
router.get('/codeimg', async (req, res) => {
  try {
    // token:是数字字母表示形式
    // buffer:是图片数据
    // size 是生成验证码的长度或者说是个数
    const {token,buffer} = await captcha({size:6});
    // 生成的有效时间
    const to2Date = moment().add(2, 'minutes').toDate();
    // 图片验证 和 有效时间的对象数组
    const dateToken = [{ 'Date':to2Date,'token':token}];
    // 将数组对象保存到session中
    req.session.imgCode = dateToken;
    res.json({
      success:true,
      data:buffer.toString('base64')
    });
  } catch (err) {
    res.json({
      success:false,
      data:''
    });
  }
});
```

