编写get接口

```
router.get('/user',(req,res)=>{
    let {query} = req
    let data = ...
    res.send({
        status:200,
        msg:'请求成功',
        data,
    })
})
```

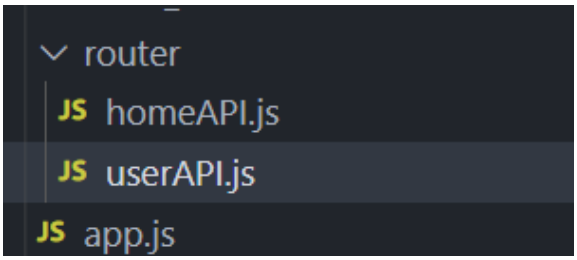
编写POST接口

```
app.use(express.urlencoded({extend:false}))
//urlencoded解析表单数据的中间件存放数据在req.body
router.post('/user',(req,res)=>{
    const body = req.body
    res.send({
        status:200,
        msg:'成功',
        data:body
    })
})
```

小接口

```
const express = require('express')
const userApi = require('./router/userAPI')
const homeApi = require('./router/homeAPI')
const app = express()

app.use(express.urlencoded({extended:false}))
app.use('/user',userApi)
app.use('/home',homeApi)
app.listen(80,()=>{
   console.log('serve is runing....');
})
```

```
const express = require('express')
const mysql = require('mysql')
const router = express.Router()
var cnect = mysql.createConnection({
    host:'localhost',
    port:"3306",
    user:"root",
    password:"...",
    database:'...'   //连接的数据库 没有就不连
})
cnect.connect((err)=>{
console.log(err)
})
                            // 数据库连接
const select_all = "select * from user"
function success(res,data) {
    res.send({
            status:200,
            msg:"请求成功",
            data:data
    })              //懒狗封装的请求成功函数
}
router.get('/userInfo',async (req,res)=>{
    cnect.query(select_all,(err,ret,fields)=>{
            success(res,ret)

    })

})
router.post('/add',(req,res)=>{
    let {body} = req
    success(res,body)

})
module.exports = router
```



跨域的行为： （来自百度）

1. 协议不同： http://
2. 子域名不同：www
3. 主域名不同：baidu.com
4. 端口号不同：8080
5. ip地址和网址不同：www.baidu.com

CORS 跨域资源共享

- 接口跨域的问题 自己编写的get和post**不支持跨域请求**

```
<html>
    <body>
            <buttonn></button>
    </body>
    <script>
            //以jquery的ajax为实例
            $(function() {
                    $('button').on('click',function(){
                            $ajax({
                                    type:"GET",
                                    url:"http://127.0.0.1/api/get",
                                    data:{name:"zcs",age:18,},
                                    success:function(res){
                                            console.log(res)
                                    }
                            })
                    })
            })
    </script>
</html>
```

解决方式:

1. **CORS(主流) √**
2. **JSONP(有缺陷,只支持get)**

```
//cors中间件解决跨域问题
npm install cors
```

```
const cors = require('cors') //导入中间件
app.use(cors())
```

什么是CORS?

A:跨域资源共享(Cross-Origin Resource Sharing) 由一系列HTTP响应头组成,这些http响应头决定浏览器是否阻止前端JS代码跨域获取资源。

CORS的注意事项

- **CORS主要在服务器端进行配置。浏览器端无需做任何额外的配置，即可请求开启CORS的接口**
- **在浏览器中有兼容性**

CORS响应头部

Access-Control-Allow-Origin

```
            Access-Control-Allow-Origin:<origin>(url) | *    //*表示任意
res.setHeader('Access-Control-Allow-Origin','http://ymtx.cn')
```

Access-Control-Allow-Headers

默认情况下，CORS 仅支持客户端向服务器发送如下的 9 个请求头：

Accept、Accept-Language、Content-Language、DPR、Downlink、Save-Data、Viewport-Width、Width 、Content-Type （值仅限于 text/plain、multipart/form-data、application/x-www-form-urlencoded 三者之一）

超过9个会失败

客户端发送额外的请求需要在服务端额外声明 否则请求失败

```
res.setHeader('Access-Control-Allow-Origin','Content-Type,X-Custom-Header')
```

Access-Control-Allow-Methods

默认情况下,CORS仅支持客户端发起的GET POST HEAD请求

如果客户端希望通过PUT,DELETE等方式请求服务器的资源，则需要在服务端，通过Access-Control-Allow-Methods，来指明实际请求所需使用的HTTP方式

```
res.setHeader('Access-Control-Allow-Methods','*')
```

预检请求

- **请求方式为GET，POST，HEAD之外的请求Method类型**
- **请求头中包含自定义头部字段**
- **向服务器发送了application/json格式的数据**

预检请求:在浏览器与服务器正式通信之前,浏览器会先**发送OPTION请求进行预检,已获知服务器是否允许该实际请求,**所以这一次的OPTION请求称为"预检请求"。**服务器成功响应请求预检后,才会发送真正的请求,并现代真实数据。**

简单请求和预检请求的区别：1次与两次

JSONP

1. 回顾 JSONP 的概念与特点

概念：浏览器端通过 <script> 标签的 src 属性，请求服务器上的数据，同时，服务器返回一个函数的调用。这种请求数据的方式叫做 JSONP。

特点：

① JSONP 不属于真正的 Ajax 请求，因为它没有使用 XMLHttpRequest 这个对象。

② JSONP 仅支持 GET 请求，不支持 POST、PUT、DELETE 等请求。