复习：

Node.js模块的分类：

(1)官方提供的模块

global、util、querystring、url、Buffer、fs、http

(2)第三方模块——npm

mysql

(3)自定义模块

文件模块 user.js require('user')

目录模块 node\_modules/db/package.json require('db')

http模块的使用：

const http = require('http');

var server = http.createServer()

server.listen(8080)

server.on('request', (req, res)=>{ })

mysql模块的使用：

# npm i mysql ./node\_modules/mysql

const mysql = require('mysql')

var pool = mysql.createPool({

host: '127.0.0.1', user: 'root', password: '',

database: 'jd', connectionLimit: 10

});

pool.getConnection((err, conn)=>{

conn.query('SQL..?,?', [x, x], (err, result)=>{

conn.realease()

})

})

练习：

(1)编写SQL： tmooc.sql，数据库名为tmooc，表：

stu( sid, sname, score, schoolTime-入学时间 )

插入4行记录

(2)创建Node.js应用： app.js，创建Web服务器，监听8080端口，接收如下请求地址：

/stu/add：向客户端输出/public/stu/add.html内容

/stu/add.do：接收客户端提交的sname / score / schoolTime，保存入tmooc数据库，返回“添加成功！新记录在数据库中的编号为：5”

/stu/list：向客户端输出数据库中所有的学生信息，以JSON格式

1.Express框架

使用官方提供的http模块可以创建一个Web服务器应用，但是此模块非常底层，要处理各种情形，比较繁琐。推荐使用http模块的进一步封装简化模块 —— express —— 第三方模块，是一个基于Node.js的http模块而编写的高层模块，简化Web服务器端应用的开发。V4版本中官方的定义：Express是一个请求处理工具，用于接收客户端的请求消息，返回响应消息。

可以到npmjs.org上下载该模块。

该模块还有自己的专用官网：http://expressjs.com/

该模块中文镜像网站：http://www.expressjs.com.cn/

使用方法：

(1)下载express模块

npm i express

(2)在自己的模块中引入express模块

const express = require('express')

(3)使用http模块创建一个Web服务器对象，让express()担当请求处理函数

var app = express()

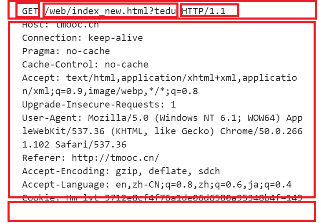
var server = require('http').createServer(app)

server.listen(8080)

午间练习：自己仿写express模块中的实例程序。找ReadMe

《HTTP编程指南》

2.HTTP协议——请求消息



**请求方法：用于标明此次请求的目的**

GET： 表客户端想“获得”指定的资源

POST： 表客户端想“邮寄/上传/添加”指定的数据给服务器，相关数据在请求主体中

PUT：表客户端想“放置/上传/更新”服务器上的指定资源，相关数据在请求主体中

DELETE：表客户端想“删除”服务器上的指定资源

|  |
| --- |
| POST和PUT的语义区别：  所有的请求方法中只有这二者有请求主体，都可以用于向服务器传数据。POST：一般表示“添加”，每次POST请求都应该在服务器上产生一条新的记录。PUT一般表示“更新”，相同的PUT请求即使多次请求也只能产生一条影响——多次PUT请求是“幂等的”。 |

|  |
| --- |
| GET /user HTTP/1.1 表客户端想获取所有用户 |
| GET /user?pno=2&psize=10 HTTP/1.1表客户端想分页获取数据 |
| GET /user/pno/2/psize/10 HTTP/1.1表客户端想分页获取数据 |
| GET /user?uid=10 HTTP/1.1 表客户端想获取10号用户 |
| GET /user/10 HTTP/1.1 表客户端想获取10号用户 |
| POST /user HTTP/1.1 表客户端想向服务器添加一条新记录  uname=tom&upwd=123 |
| PUT /user HTTP/1.1 表客户端想更新服务器上的一条记录  uname=tom&upwd=123&uid=8 |
| DELETE /user HTTP/1.1 表客户端想删除服务器上的所有用户 |
| DELETE /user?uid=9 HTTP/1.1 表客户端想删除9号用户 |
| DELETE /user/9 HTTP.1.1 表客户端想删除9号用户 |

**浏览器中如何发起GET请求：**

地址栏中输入URL、超链接/JS跳转、SRC/HREF属性、表单GET提交、AJAX-GET请求

**浏览器中如何发起POST请求：**

表单POST提交、AJAX-POST请求

**浏览器中如何发起PUT请求：**

AJAX-PUT请求

**浏览器中如何发起DELETE请求：**

AJAX-DELETE请求

|  |
| --- |
| Express路由：  //路由(Route) = Method + Path + Fn |

3.使用Express处理客户端的GET请求

app.get(url, (req, res)=>{

res.send('向客户端返回HTML字符串响应')

res.sendFile(\_\_filename+"/向客户端输出指定文件内容")

res.json( obj ) //将obj转化为JSON字符串，输出给客户端，并修改响应头

})

练习：使用Express创建一个Web服务器，可以向客户端提供一个/index响应，内容是一个HTML页面，其中还使用了一个CSS文件、图片、jquery-1.11.3.js文件

<html>

<head> <link ...href="/index.css">

<body> <img src="/logo"> <script src="/jquery">

</html>

**接收GET请求中的查询字符串：**

//GET /user?uid=10&loc=bj

app.get('/user', (req, res)=>{

//express为每个req对象添加一个query属性

console.log(req.query.uid);

console.log(req.query.loc);

})

**接收GET请求中的请求参数：**

//GET /user/jsj/60

app.get('/book/:type/:id, (req, res)=>{

//express为每个req对象添加一个params属性

console.log(req.params.type); //jsj

console.log(req.params.id); //60

})

练习：创建ExpressWeb服务器，配合mysql模块，接收如下请求地址：

GET /stu/3 向客户端输出3号学生的信息，以JSON格式

GET /stu/5 向客户端输出5号学生的信息，以JSON格式

4.使用Express处理客户端的POST请求

app.post(url, (req, res)=> {})

提示：客户端推荐使用AJAX异步请求，提交POST请求。

示例：

app.post('/user', (req, res)=>{

//接收请求主体中的数据

req.on('data', (buf)=>{

var obj = qs.parse(buf.toString());

})

});

5.使用Express处理客户端的PUT请求

app.put(url, (req, res)=> {})

提示：客户端推荐使用AJAX异步请求，提交POST请求。

示例：

app.put('/user', (req, res)=>{

//接收请求主体中的数据

req.on('data', (buf)=>{

var obj = qs.parse(buf.toString());

})

});

6.使用Express处理客户端的DELETE请求

app.delete(url, (req, res)=> {})

练习内容：

1.编写SQL：创建数据库dangdang，书籍信息表book(bid, title, price, pubDate-出版日期, intro-内容简介)，并插入4行书籍记录。

2.使用Express创建Web服务器应用，可以接收如下的请求：

至少可以向客户端提供如下静态资源：

/public/booklist.html 初始时显示一个空白的表格，页面加载完成后向服务器异步请求所有的书籍信息

/public/jquery-1.11.3.js

/public/book\_detail.html 在DIV中显示某一本书的详情

/public/book\_add.html 显示一个添加书籍的表单，点击“提交”按钮，异步提交给服务器

/public/book\_update.html 显示一个修改书籍的表单，输入域中是当前要修改的书籍信息，点击“提交”按钮，异步提交给服务器

以及如下动态资源地址——注意：下述地址都使用AJAX异步请求

1) GET /book

向客户端输出书籍表中的所有记录，以JSON格式

2) GET /book/:bid

向客户端输出指定编号的书籍全部信息，以JSON格式

3) DELETE /book/:bid 删除指定编号的书籍记录，向客户端输出

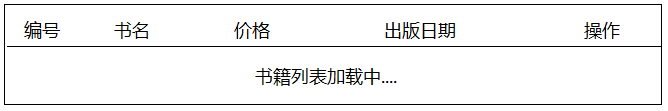
{ code: 1, bid: xx } 或者 { code: 2, msg:'指定书籍编号不存在' }

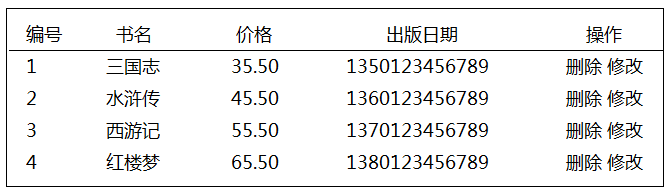
4) POST /book

接收客户端提交的请求主体数据(title/price/pubDate/intro)，执行书籍添加操作保存入数据库，向客户端输出{ code: 1, bid: xx }

5) PUT /book

接收客户端提交的请求主体数据(bid/title/price/pubDate/intro)，执行书籍更新操作修改数据库中对应的书籍，向客户端输出{ code: 1, affectedRows:1 }





www.php100.com