今天学习的内容

1：复习昨天重点知识

nodejs模块分类

1)官方提供模块:

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

const http = require("http");

var server = http.createServer();

server.listen(80);

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

mysql模块的使用

npm i mysql ./node\_modules/mysql

#加载模块

const mysql = require("mysql");

#创建连接 发送sql

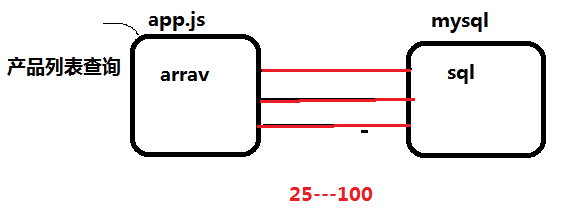
var conn = mysql.createConnection({});

conn.query("INSERT INTO t\_user values(?,?)",['tom',1],function(err,result){});

#关闭连接

conn.end();

2:为了提升数据库操作性能，nodejs提供完善功能连接池.



1:创建连接池

var pool = mysql.createPool({...})

2:从连接池中获取连接

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

3: 释放连接，回到连接池

conn.release();

});

10:25---10:35

练习:动态 web 服务器, 接收客户端请求，若客户请求地址是/register,则向客户端输出 public/register.html中的内容.

<form action="register.do">

用户名<input type="text" name="uname" />

密码<input type="password" name="upwd" />

提交<input type="submit"/>

</form>

若客户端请求URL 是register.do,则解析URL中查询字符串，把客户端提交uname和upwd追加写入jd\_user表中，并向客户端输入"注册成功".

3:Express框架

使用官方提供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http 模块进一步封装简化模块--express.

第三方模块，是一个基于nodejs的http模块而编写高层模块,简化web服务端应的开发,express是一个请求处理工具，用于接客户端请求消息，返回响应消息.

可以到 npmjs.org上下载该模块

该模块还有自己专用官网: http://expressjs.com

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 = http.createServer(app);

server.listen(8080);

4:HTTP协议---

[http协议作用:传送网页]

请求消息(客户端浏览器发送服务器)

响应消息(服务器发送客户端浏览器)

请求消息：第一行{请求消息启始行}



请求方式:用于标明此次请求的目的

GET: 表示客户端想"获取"服务器端指定资源

POST:表示客户端想“上传/发送/添加”数据给指定服务器,

相关数据请求主体中.

PUT:表示客户端想"上传/更新"服务上指定资源，

相关数据请求主体中.

DELETE:表示客户端想"删除"服务器指定资源

#浏览器如何发起GET请求

地址栏输入URL回车/超链接/js跳转/src/href/表单GET提交

/AJAX GET 请求.

#浏览器如何发起POST请求

表单POST提交/AJAX-POST请求

#浏览器如何发起PUT请求

AJAX-PUT请求

#浏览器如何发DELETE请求

AJAX-DELETE请求

5:使用Express处理客户端GET请求

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

res.send("向客户端返回html字符串响应");

res.sendFile(绝对路径\_\_dirname/客户端输出指定文件内容);

res.json(obj);//将obj转换json字符串,输出客户端,并修改响

//应头

});

练习一:使用express创建一个web服务器,可以向客户提供一个/index响应，内容是一个index.html，其中还使用一个css文件,

一个图片jquery.js文件

<html>

<head>

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

</head>

<body>

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

</body>

</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("/user/: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);

});

练习:创建express web服务器,配合mysql模块,

接收如下格式请求地址

GET /user/3 向客户端出3号用户的信息,以json格式

GET /user/5 向客户端出5号用户的信息,以json格式

6: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)=>{

//接收请求主体数据

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

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

})

})

7: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put请求

示例:

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

//put 接收请求主体中的数据

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

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

});

})

8:使用 express处理客户端的delete请求

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

练习内容: 17:30---17:45

使用express创建web服务器应用，可以接收如下请求

/ 获取/public/users.html 初始化显示空表格,页面加载完成后向服务器异步请求(/list.do)所有用户信息.

/jquery 获取jquery文件/public/jquery.js

/list.do 获取用户列表(jd/jd\_user)json

/add 获取/public/user\_add.html 显示添加用户表单

点击"提交"按钮

/add.do 将用户信息保存数据库并且显示添加成功