目录

[3.1:(准备工作)NodeJS 下载并且安装 1](#_Toc502304124)

[3.2:一道面试题 用户在浏览器中输入 www.taobao.com 直到看到页面之间发生了什么? 1](#_Toc502304125)

[3.3:静态网页和动态网页 1](#_Toc502304126)

[3.4:NodeJS 概述(重点&难点) 1](#_Toc502304127)

[3.5:NodeJS 运行模式 1](#_Toc502304128)

[3.6:如何自学一门语言 Node.JS 2](#_Toc502304129)

[3.7:Nodejs中特有的概念--模块(重点&难点) 3](#_Toc502304130)

[3.8:Node.js模块分类 4](#_Toc502304131)

[3.9:Node.js 预定义模块---Global 4](#_Toc502304132)

[3.10:练习:仿写Node.js提供一模块:URL 4](#_Toc502304133)

[3.1:第三方的模块(第二节课开网|重点) 5](#_Toc502304134)

[3.2:nodejs 官方模块http 5](#_Toc502304135)

[3.3:nodejs 第三方模块 mysql(此模块专用于访问与操作mysql数据库服务器) 5](#_Toc502304136)

[3.4:nodejs 第三方模块 mysql-使用步骤 6](#_Toc502304137)

[3.5:nodejs 第三方模块 mysql-网络攻击手段 6](#_Toc502304138)

[3.1:第三方模块 express (重点&难点) 7](#_Toc502304139)

[3.2:第三方模块 express "处理GET请求 7](#_Toc502304140)

[3.3:第三方模块 express "处理GET请求接收参数 7](#_Toc502304141)

[3.1:图书管理(练习) 9](#_Toc502304142)

[3.2: cors 跨域访问 9](#_Toc502304143)

[3.3:nodejs 项目(学子商城一部分功重写) 10](#_Toc502304144)

[3.4:nodejs express中间件与路由(重点&难点) 10](#_Toc502304145)

[3.1:nodejs 中间件 --(重点&难点) 11](#_Toc502304146)

[3.2:webpack 12](#_Toc502304147)

3.1:(准备工作)NodeJS 下载并且安装

3.2:一道面试题 用户在浏览器中输入 www.taobao.com 直到看到页面之间发生了什么?

(1)操作系统访问网络上的DNS服务器，把域名转换IP地址

(2)浏览器获取返回ip地址并且向ip地址发送HTTP请求

(3)web服务器接收并解析请求消息,查找指定资源,可能访问数据库,构造并返回http响应消息

(4)浏览器接收并解析响应消息

(5)浏览器缓存接收响应消息，并解析和渲染响应的内容

3.3:静态网页和动态网页

静态网页:网页内容任何人在任何时间访问都是不变的.

HTML/CSS/JS/视频...

动态网页:网页内容不同人在不同时间访问可能不同的

JSP/PHP/ASP.NET/Node.JS

3.4:NodeJS 概述(重点&难点)

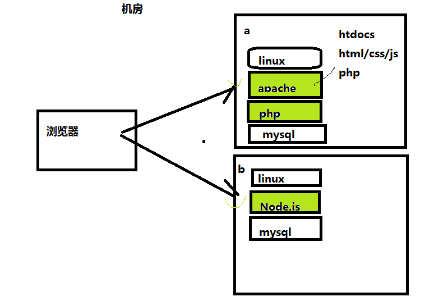
JSP=HTML+java :功能强可靠，适合大型企业级项目

PHP=HTML+PHP :简单易于，适合互联网项目

ASP.NET=HTML+C# :易用适合windows 平台

Node.JS=HTML+"JS" :性能好,适合于服务器端IO密集型项目，不适合CPU密集项目.

|  |  |
| --- | --- |
| CPU密集型项目 | 滴滴打车,天气预报 |
| IO密集型项目 | 输入输出(学子商城) |



概念:Node.JS不是JS,是一种服务器开发平台,它的竞争对手是PHP/JSP/ASP. 历史上第一次有一种语言可以通吃前后台.

官网:www.nodejs.org

版本:(16初)0.12 (16年中)4.x (16底)6.x

LTS:Long Term Support 长期稳定支持版本

Current:最新版本

安装时注意事项:安装目录(不要有中文,不要有空格)

3.5:NodeJS 运行模式

(1)交互模式--一般用于临时测试

REPL:Read Evaluate Print loop,输入一行代码执行一行

注意:交互模式自带输出功能,不必写 console.log();

node 回车 #进入交互模式

.exit #退出交互模式

(2)脚本模式--正式项目中使用的方式

把要执行的所有语句，编写在一个文本文件中(后缀任意，没有也行),一次性提交给node解释器.

node 完整路径名/x.js 回车

提示:只要安装完成Node.js重启一个WebStorm,ws可以自动检测node.js ,右键运行

练习:创建01.js文件，打印九九乘法表，以脚本模式运行,

ws中运行

3.6:如何自学一门语言 Node.JS

(1)了解背景

---百度百科 Node.JS (IO非阻塞/事件驱动)

(2)搭建开发环境

下载并安装nodejs.msi

(3)数据类型--(多)

|  |
| --- |
| 前端JS数据类型  (1)基本/原生/值类型  number/string/boolean/null/undefined  (2)引用类型  ES对象;String;Math;Date;Object;Function;RegExp;Error  *BOM:window;document;history;screen;location;event*  *DOM:Node/Element/Attr/*  用户自定义:{} |
| 后端Node.JS数据类型  (1)基本/原生/值类型  string/number/boolean/null/undefined  (2)引用/对象类型  ES:String/Number/Boolean/Math/Data/RegExp/Object..  用户自定义对象:{}  NodeJS原生第三方对象库:几百 十多万 |

(4)变量和常量

var age = 20;

const PI = 3.14;

(5)运算符

算术运算符 + - \* / % ++ --

比较运算符 < > <= >= == === !==

逻辑运算符 && || !

位运算符 << >> ^= !

三目运算符 ? :;

赋值运算符 += -= \*= /+ %=

特殊运算符 typeof .

(6)逻辑结构

循环: while do..while for(;;); for(..in..) for(..of..)

练习:03.js 声明一个保存5个学生成绩的数组,使用三种

循环依次打印出每个成绩值

选择结构:if..else switch..case

练习:创建一个变量 var path = "/path";使用二种选择结构,判断path的值为哪种(/index,/search/,/login)，调用不同的函数

(7)通用小程序

九九乘法表,100以内质数,数组排序...

练习:打印出100以内质数 14:34--14:36

(8)函数和对象

(9)常用组件,第三方工具

(10)实际小项目

3.7:Nodejs中特有的概念--模块(重点&难点)

|  |
| --- |
| Modal:模态框 Model:模型 Module:模块 |

一个web项目功能可以分为很不同的"模块":如：商品管理模块，用户管理模块，支付模块，促销模块...

Node.JS按照不同的功能，可以把函数，对象分别保存不同文件或目录下“文件或目录称为模块”

Node.js为每个模块创建独立构造函数，解释会(自动)添加到每个.js文件中.

05.js

(function (exports, require, module, \_\_filename, \_\_dirname){

exports:{} 用于声明向外部出自己成员

require:fn 用于导入其它的模块，并且创建指定模块对象

module:当前模块对象

\_\_filename

\_\_dirname

var i = 10;

console.log(document);

//自己编写程序内容

});

每个模块都可以使用自己的require()函数引入另一个模块--底层本质就是创建指定模块的一个对象实例.

var m = require("./模块文件名");

每个模块可以使用exports对象向外导出/公开一些自己的成员供其它模块使用

exports.成员名 = 成员值.

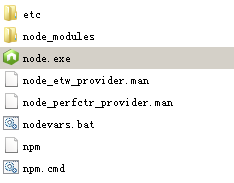
|  |
| --- |
| Node.JS中有二个对象可以向外公共数据;  exports和module.exports  Node.js底层代码: exports = module.exports;  所以：  若只是给exports对象添加新成员，则等于给module.exports添加新成员,但是若改了exports指向，则不同产生实质作用.  exports.age = 19;  module.exports.age = 19;  但是如果:  exports = {name:"tom",age:19} |

练习:创建一个文件模块:08\_circle.js 其中定义常量PI，声明一个方法getSize(r),根据传入圆形的半径，返回其面积，以及一个方法 getPerimiter(r);根据传入圆形半径，返回周长，再创建一个应用程序 09\_app.js 引入08\_circle.js模块，调用其公开的两个成员方法.

练习:创建一个模块:10\_ArrayUtil.js包含一个方法sum(arr),返回指定数组中所有数值的和，一个方法avg(arr)返回指定数组中的所有数组的平均值，使用module.exports导出上述两个成员

再写11\_app.js 模块，调用其公开的方法

3.8:Node.js模块分类



(1)Node.JS官方提供的模块--安装在解释器内器

require("模块名");

(2)第三方编写模块

(3)用户自定义编写模块

exports.x = require("./模块文件名");

3.9:Node.js 预定义模块---Global

该模块提供可以直接使用，而无需require(global);

console

setInterval();

setTimeout();

setImmediate(fb); 等价于setTimeout(0,fn);

3.10:练习:仿写Node.js提供一模块:URL

创建模块文件 MyUrl,向外导出一个方法 resolve(url);

该方法接收参数:

形如: http://www.jd.com:8080/ad/index?uname=tom&pno=3

返回对象:形如

{

protocol:"http",

server:"www.jd.com",

path:"/ad/index",

query:"uname=tom&pno=3"

}

再编写一个主模块:引入上述模块并调用

运算出错: nodejs路径不正确

(1)File->Setting->Lanaguages->Node.js and NPM

(2)Node interpreter:[c:\program Files/nodejs/node.exe][...]

作业1:授权己有拥角色/未拥有角色

作业2:url模块: query {uname:"tom",pno:3}

3:今天的目标

3.1:第三方的模块(第二节课开网|重点)

安装5个第三方模块{e:/}

npm i express

npm i body-parser

npm i mysql

npm i cookie-parser

npm i express-session

npm i cors

3.2:nodejs 官方模块http

练习：静态web服务器:创建web服务器，接收客户端请求，

一律向客户端输出一句 <h3>hello nodejs</h3>

练习:动态web服务器:创建web服务器，接收客户端请求,

若客户端请求 /index.html则向客户输出 index.html内容

若客户端请求 /其它 则向客户输出

<h1>404 您查找的数据不存在</h1>

|  |
| --- |
| Error: listen EADDRINUSE :::8080  错误:原因 端口冲突(二个程序在工作时使用同一个端口)  解决:停一个程序或者修改程序端口 |

3.3:nodejs 第三方模块 mysql(此模块专用于访问与操作mysql数据库服务器)

为了精简node.js解释器、官方没有提供任何数据库相关模块,必须使用npm工具下载第三方模块，在www.npmjs.org 上搜索关键字 mysql ,可以得到很多相关模块。

使用工具下载安装

npm i mysql

mysql数据库:查询速度比较快更新慢下

数据库存储三种方案

(1)大量数据保存 mysql数据库中

(2)NOSQL解析方案:Redis(内存级别数据库)

(3)文件存储:试卷

3.4:nodejs 第三方模块 mysql-使用步骤

(1)创建到数据库的连接(保证mysql启动成功/第三方模块加载成功)

const mysql = require("mysql");

var conn = mysql.createConnection({...})

(2)创建一条SQL语句并且发送SQL

var sql = "...";

conn.query(sql,function(err,result){});

(3)关闭连接

conn.end();

3.5:nodejs 第三方模块 mysql-网络攻击手段

(1)SQL注入

利用:sql语句中弱点，添危害数据放在sql语句中.

var fname = "';drop table mysql.user ";

var fname = "';select into c:/aa.dat ";

var uname = "' OR 1=1 #";

var upwd = "";

解决方案一:占位符

var sql = "INSERT INTO user VALUES(null,?,?)";["tom","123"]

解决方案二:项目工程分为二个组成部分:

前端业务<----->核心功能(mysql);

(2)XSS 脚本攻击

var name = "<script>document.all.disabled=''</script>";

用户名 [jerry]

解决方案:

所有用户输入的数据过滤:drop table;

练习:仿写上述代码，使用node.js向 web1708acl02/t\_user

[uid/uname/upwd] 表添加一行新的记录.

练习:仿写上述代码，使用node.js向 web1708acl02/t\_user

uid=3 upwd=123456

练习:使用http和mysql模块

\*创建动态web服务器,接收如下请求URL

\*/register.html 服务器向客户端返回register.html内容

/register.do 接收客户端提交uname/upwd

GET req.query.uname

保存mysql t\_user

/login.html 服务器向客户端返回 login.html 内容

/login.do 接收客户端提交 unname/upwd

/userlist.html

/userlist.do 接收客户请求，发送用户列表

404.html

作业1:/userlist.do

作业2:授权

3:今天的目标

3.1:第三方模块 express (重点&难点)

使用官方提的http模块可以创建一个web服务器应用，但是此模非常底层，要处理各种情形,推荐使用http模块进一步封装模块--express -第三方模块,V4版本中定义:express是一个请求处理工具，用于接收客户的请求消息，返回响应消息.

可以处npmjs.org 下载

npm i express

http://www.expressjs.com.cn

使用方式:

(1)在模块中引入express模块

const express = requre("express");

(2)创建 express对象

var app = express();

(3)创建服务器对象，绑定监听端口

var server = http.createServer(app);

server.listen(8080);

3.2:第三方模块 express "处理GET请求

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

//昨天:200;text/html;write;end

res.send("向客户端返回html字符串");

//昨天:200;text/html;fs.readFile,write.end

res.sendFile(\_\_dirname+"文件名");

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

});

3.3:第三方模块 express "处理GET请求接收参数

(1)GET /user?uid=10&loc=bj

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

//express为每一个 req对象添加query属性

req.query.uid;

req.query.loc;

});

(2)GET /user/jsj/61

app.get("/user/:type/:id",(req,res)=>{

//express为req添加属性params属性

console.log(req.params.type);

console.log(req.params.id);

});

练习: 创建express服务器,配置 mysql 模块

GET /role/3 向客户端输出3号角色信息.json

(3)POST /user

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

});

如何接收参数

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

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

});

b:单独加载一个(第三方)模块(解析post传递参数)

const bodyParser = require("body-parser");

将模块添加中间件列表

app.use(bodyParser.urlencoded({extended:false}));

在post中获取参数

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

var uname = req.body.uname;

});

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

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

练习:

(1)编写 SQL：创建数据库

dangdang book(bid,title,price,pubDate)

user(uid,uname,upwd)

添加4条记录

(2)创建express服务器,接收如下请求

/booklist.html

/login.html

/register.html

/jquery.js

public

(3) /book 获取图片列表

(4)/login 完成登录

(5)/reg 完成注册

(6)/del 完成删除图书

作业1:node js session cookie

作业2:授权

今天学习的内容

1:复习昨天的内容

2:作业

3:今天的目标 [taonetwork]

3.1:图书管理(练习)

a:添加三个中间件(session;cookie;cors)

session(会话):过程

session对象:保存数据[在当前会话需使用数据,uid,权限列表]

b:nodejs使用session

(1)express-session 第三方模块(中间件)

(2)cookie-parser 第三方模块(中间件)

c:配置

cookie: app.use(cookieParser());

session:app.use(session({

resave:false, #每次请求是否需要重新设置cookie;

saveUninitialized:true #无论是否有cookie,设置标记

connect.sid

}));

d:使用

req.session.uid=10; 向session对象添加属性

req.session.uid; 获取

完成:(坑)

1:如果用户登录成功将uid保存session.uid对象

2:获取图片列表 session

3.2: cors 跨域访问

概念:从一个域名下程序访问另一个域名程序

示例: www.a.com/1.js ---> www.b.com/1.php

跨域:

(1)域名不同

(2)端口号不同

http://127.0.0.1:80/1.js -> http://127.0.0.1:8080/login

(3)协议不同;

http://127.0.0.1/1.js -> https://127.0.0.1/login

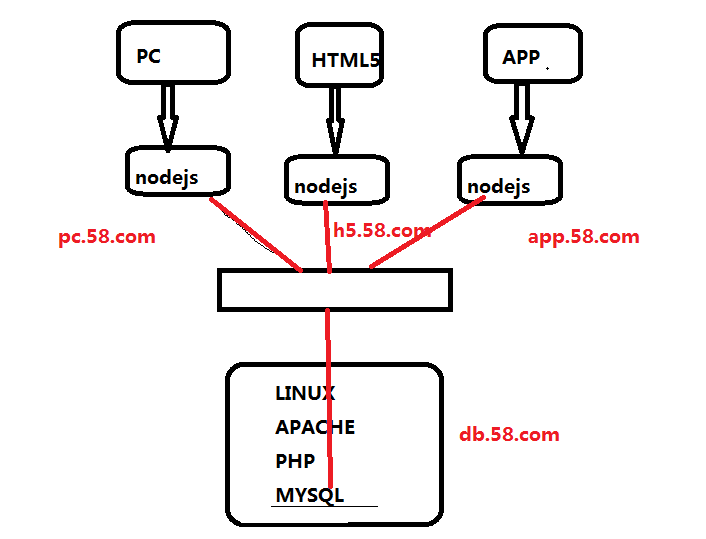
(4)ip与主机名

http://localhost/1.js -> http://127.0.0.1/login

跨域访问:

允许:js/css/img

不允许:xhr(ajax)



项目升级:[改动大]

1:将node.js/public/所有html/js/css 移动位置

apache/htdocs/book

2:nodejs /login /register /booklist

3:访问方式:apache<->node.js

http://127.0.0.1/book/login.html

-->ajax->

http://127.0.0.1:8080/login

4:重点改动程序

4.1:加载对应模块 cors

4.2:配置模块:只允许哪个域名下的资源来访问我

你访问我程序请带cookie

5:补坑:

原因:跨域cookie中保存数据丢失

添加如下的选项:指定ajax发起请求一定要带上cookie值

解决:apache/html/

$.ajax({

xhrFields:{withCredentials:true}

});

3.3:nodejs 项目(学子商城一部分功重写)

3.4:nodejs express中间件与路由(重点&难点)

今天学习的内容

1:复习近几个阶段的知识重点

2:项目

3:今天的目标

3.1:nodejs 中间件 --(重点&难点)

Express;中间件(static);

中间件(Middleware)是一个函数(session;cors;cookie)

作用:它可以访问请示对象(request object) 也可以用于访问响应对象(resonse object)和web应用请求-响应中对象.

中间件的功能包括

(1)执行任何代码

(2)修改请求和响应对象

(3)终结请求-响应循环

(4)调用下一个中间件

中间件分类

(1)应用级中间件(拦载器)

(2)路由级中间件 路由=请求路径+处理方法+处理函数

app.get("/user",(req,res)=>{});

(3)错误处理中间件

(4)内置中间件(express static)

(5)第三方中间件(session;cookie; cors)

中间件语法:

var app = express();

#req 请求对象

#res 响应对象

#next 调用下一个中间件

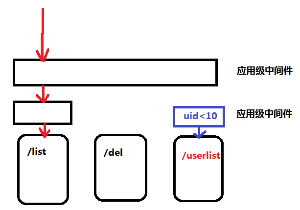
app.use(function(req,res,next){

console.log("中间任何代码");

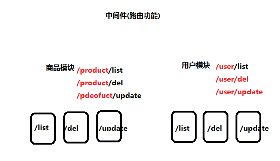
next();

});

应用级中间件



路由级中间件



//统一定义产品路由中间件

var router = express.Router();

router.get("/list",(req,res,next));

router.get("/del",(req,res,next));

router.get("/update",(req,res,next));

//统一定义用户路由中间件

var router1 = express.Router();

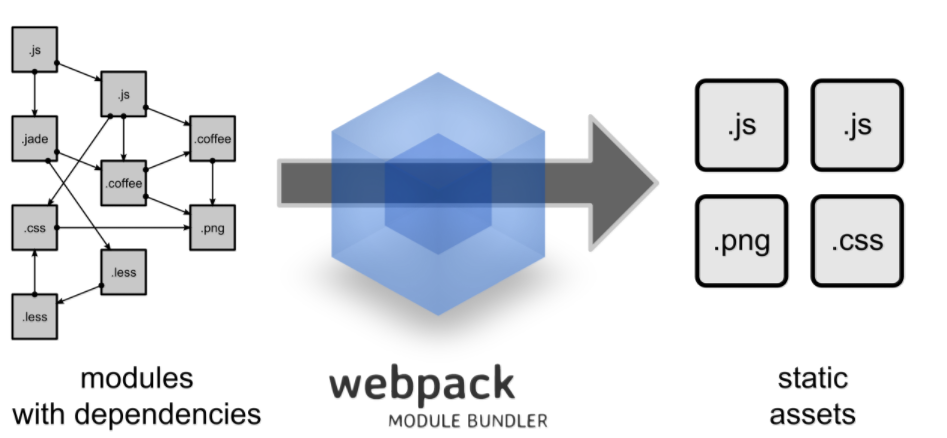
router1.get("/list",(req,res,next));

//加载应用

app.use("/product",router);

app.use("/user",router1);

3.2:webpack



webpack是一款加载器和打包工具:它能把各种资源(js/less/coffect/图片)等作为模块来使用处理并且打包.

安装和配置

1:创建测试项目 e:/webpack-test

2:进入项目目录中 e:

mkdir webpack-test

cd webpack-test

3:初始化项目 npm init

4:全局安装 npm install webpack -g

5:本地安装 npm install webpack --save-dev

示例:

1:创建js webpack-test/test.js

2:添加内容 hello

3:webpack test.js test.bundle.js

示例:

1:创建js webpack-test/word.js

2:添加

3:test.js 中添加文件 requre("./word");

4:webpack test.js test.bundle.js

示例

1:创建css 1.css

2:test.js 引入 1.css

3: webpack test.js test.bundle.js

错误:安装依装第三方模块(loader)

npm install css-loader style-loader --save-dev

并且修改过代码

示例:

(1)监控项止，内容实时修改

webpack test.js test.bundle.js --watch

(2)看打包过程

webpack test.js test.bundle.js --progress

(3)使用哪些模块

webpack test.js test.bundle.js --display-modules

示例:

(1)index.html 引入打包后文件

示例:

创建目录 e:/webpack-demo

dist #生成静态资源文件

node\_modules

src #源程序

script

styles

package.json 描述文件

webpack.config.js webpack配置文件

1:添加src/script/main.js

2:添加webpack.config.js

3:打包 webpack

示例:修改描述文件 package.json

添加入口

"webpack":" webpack --progress --colors "

再打包

npm run webpack