# Node.js学习笔记

# 一．安装

npm install;安装modules

查看安装的所有包：npm list -g --depth 0

Node -v

Npm -v 安装好node.js后查看一下版本

安装express框架：

npm install -g express-generator

express（有一个你确定安装吗？键入y）

npm install（安装express及依赖）

express --version，如下图：

IMG_256

express -e jizhi ;在此目录下创建一个jizhi

之后开始npm

Npm init 初始化包，自动生成package.json。

# 二．Fs文件模块

## 读文件

//readFile(文件名，回调函数)

const fs=require("fs");

fs.readFile("a.txt",function(err,data){

if(err){

console.log("读取失败！");

}else{

console.log(data.toString());

}

});

## 写文件

//writeFile(文件名，内容，回调函数)

const fs=require("fs");

fs.writeFile("a.txt","awda wdo",function(err){

console.log(err);

});

# 三．http数据解析GET/POST

## Get(数据在URL中)

<form action="http://localhost:8080/" method="get">

Get数据解析：

使用Url：

const http=require("http");

const urlLib=require("url");

http.createServer(function(req,res){

console.log("服务启动");

var obj=urlLib.parse(req.url,true);

var url=obj.pathname;

var GET=obj.query;

console.log(url,GET);

res.write("成功了");

res.end();

}).listen(8080);

## post(数据不在URL中)

POST数据比GET数据大得多。所以需要分段发送。

var str = '';

// 数据量大的时候可能会执行很多次

req.on('data',function(data){

// 结果类似xxx=xxx&&yyy=yyy

str += data;

});

req.on('end',function(){

// 解析输出

var POST = querystring.parse(str);

console.log(POST);

});

res.end();

# 对文件/接口的访问

## 文件（其余均是文件）

## 接口（访问uesr的）

# 前台往后台发送数据

## form表单

表单传的时候必须要由name，即

用户：<input type="text" name="user" id="user" value="" />

## Ajax(常用)

我用jQuery.ajax({

Url：，

Data：，

Type：，

Success：，

Error:

});

# 实例（登录注册）

## 1.前台：

<body>

用户：<input type="text" name="user" id="user" value="" /><br /><br />

密码：<input type="password" name="pass" id="pass" value="" /><br /><br />

<input type="button" value="登陆" id="login\_btn" />

<input type="button" value="注册" id="reg\_btn" />

</body>

<script src="http://apps.bdimg.com/libs/jquery/2.1.4/jquery.min.js"></script>

<script type="text/javascript">

window.onload=function(){

var oTxtUser=document.getElementById("user");

var oTxtPass=document.getElementById("pass");

var oBtnLogin=document.getElementById("login\_btn");

var oBtnReg=document.getElementById("reg\_btn");

oBtnReg.onclick=function(){//注册

jQuery.ajax({

url:"/user",

data:{act:'reg',user: oTxtUser.value,pass: oTxtPass.value},

type:"get",

success:function(str){

var json=eval('('+str+')');

if(json.ok){

alert("注册成功");

}else{

alert("注册失败:"+json.msg);

}

},

error:function(){

alert("通信错误");

}

});

};

oBtnLogin.onclick=function(){//登陆

jQuery.ajax({

url:"/user",

data:{act:'login',user: oTxtUser.value,pass: oTxtPass.value},

type:"get",

success:function(str){

var json=eval('('+str+')');

if(json.ok){

alert("登陆成功");

}else{

alert("登陆失败:"+json.msg);

}

},

error:function(){

alert("通信错误");

}

});

};

};

</script>

## 2.后台：

const http = require("http");

const urlLib = require("url"); //get

const querystring = require("querystring"); //post

const fs = require("fs");

var users={};//存放数据

const server = http.createServer(function(req, res) {

//解析文件

var str = "";

req.on("data", function(data) {

str = str + data;

});

req.on("end", function() {

var obj = urlLib.parse(req.url, true);

var url = obj.pathname;

var GET = obj.query;

var POST = querystring.parse(str);

//区分接口和文件

if (url=="/user") {

switch (GET.act){

case "reg":

if (users[GET.user]) {

res.write('{"ok":false,"msg":"用户名已存在！"}');

} else{

users[GET.user]=GET.pass;

res.write('{"ok":true,"msg":"注册成功！"}');

}

break;

case "login":

if (users[GET.user]==null) {

res.write('{"ok":false,"msg":"用户名不存在！"}');

} else if(users[GET.user]!=GET.pass){

res.write('{"ok":false,"msg":"用户名或密码错误！"}');

}else{

res.write('{"ok":true,"msg":"登陆成功！"}');

}

break;

default:

res.write('{"ok":false,"msg":"未知的act"}');

}

res.end();

} else{

var filename = "./www" + url;

fs.readFile(filename, function(err, data) {

if(err) {

res.write("404");

} else {

res.write(data);

}

res.end();

});

}

});

});

server.listen(8083);

# 模块化

## 系统模块

Crypto 加密

Events 事件

Net 网络操作

Os 操作系统

Path 处理文件路径

Stream 流操作

Timers 定时器

ZLIB 压缩

## 自定义模块

Require：引入模块（模块名前加./或者直接./+模块名可省略js，因为nodejs都是js,当然当你写的模块放在node\_modules里时可以不加./）

1.var test4=require("./test4.js");

2.var test4=require("./test4");

3.var test4=require("test4");

console.log(test4.a);

引入顺序：引入模块时先从当前目录下找，找不到再在系统模块找，再找不到从node\_modules找。故今后自己写的模块全放在node\_modules里。

Module：批量输出

module.exports={a:12,b:3,c:24};

Exports：单个输出

exports.a=5;

Npm:nodejs package manager

优点：统一下载途径，自动下载依赖

eg：npm install mysql 安装

npm uninstall mysql 卸载

## 包管理器

Npm publish(上传)/unpublish（删除）/update（更新）/init（初始）

# 八．Express框架

## 安装

Npm install express

## 配置

## 接收请求

有三种方式接收用户请求，get,post,use

Get: .get(“/”,function(req,res){});

Post: .post(“/”,function(req,res){});

Use: .use(“/”,function(req,res){}); //通吃，get,post都行

eg:

var express=require("express");

var server=express();//创建服务

server.use("/a.html",function(req,res){ //处理请求

res.send("abc");

res.end();

});

server.use("/b.html",function(req,res){

res.send("123");

res.end();

});

server.listen(8080); //监听

## 响应

Express保留了原生的功能(write，end)，添加了一些方法，比如send就可以输出json而write就不行。

## 实例（登陆注册，改进六实例）

后台：（与六的后台比较）

注意：res.send('{"ok":false,"msg":"用户名不存在！"}');json里的都必须严格按照json规则书写，不可省略“”。

const express=require("express");

const expressStatic=require("express-static");

var server=express();

server.listen(8090);

console.log("监听成功");

var users={"zhangsan":'123',

"lisi":'456',

"wangmazi":'789'};//相当于数据库存东西

server.get("/login",function(req,res){//登陆接口

var user=req.query["user"];

var pass=req.query["pass"];

if (users[user]==null) {

res.send('{"ok":false,"msg":"用户名不存在！"}');

res.end();

} else if(users[user]!=pass){

res.send('{"ok":false,"msg":"密码错误，请重输！"}');

res.end();

}else{

res.send('{"ok":true,"msg":"登陆成功！"}');

res.end();

}

});

//读文件

server.use(expressStatic("./www"));

## 中间件（比如express-static）

1.使用前先安装:npm install express-static

好处：引入文件时可以直接

Server.use(expressStatic(“文件名”))；省去了原来的很多功夫。可回看二.1进行比较，这就是使用express框架得优势，更快。

Eg：

const express=require("express");

const expressStatic=require("express-static");

var server=express();

server.listen(9090);

server.use(expressStatic("1.html"));//引入文件

2.express框架获取get和post。

获取get：直接req.query;

获取post:借助中间件body-parser;

Req.body来输出post;

Eg：

server.use(bodyParser.urlencoded({

Extended:true//扩展

Limit：2\*1024//大小限制

}));

server.use("/",function(req,res){

console.log(req.body);

});

## 链式操作

Eg：

server.use("/",function(req,res,next){

console.log("a");

next();

});

server.use("/",function(req,res,next){

console.log("b");

});

## cookie和session

Cookie：保存在浏览器，不太安全，大小有限（4k）

Session：保存在服务器上，很安全，无限（服务器大小）

Session是基于cookie实现的。Cookie中会有一个session的id，服务器利用sessionid来找到session文件读取，写入。

隐患：session-id会被session劫持。

Cookie：（目前不能放入汉字）

1.读取：cookie-parser(使用中间件)

server.use(cookieParser());

server.use("/",function(req,res){

console.log(req.cookies);

res.send("ok");

});

2.发送：res.cookie();

server.use("/",function(req,res){

res.cookie("jizhi","40",{maxAge:1000\*3600\*24\*365});

res.send("ok");

});

3.删除

res.clearCookie("lisi");

4.签名:（cookie不安全，但这个可以起到一定效果）：

server.use("/",function(req,res){

req.secret="fdsf234234sfsf";

res.cookie("hehe","12431",{path:"/",maxAge:2\*24\*3600\*1000,signed:true});

res.send("cookie写入成功");

});

5.解析签名：

server.use(cookieParser("ad1234234dghdfhfh"));

server.use("/",function(req,res){

req.secret="ad1234234dghdfhfh";

res.cookie("wangmazi","dgd",{maxAge:2\*1000\*3600\*24,signed:true});

console.log("无签名:",req.cookies);//未签名得cookie

console.log("签名:",req.signedCookies);//签名得cookie

res.send("ok");

res.end();

});

1. 加密cookie（不需要了解太深，没多大意义）

Session：(通过cookie-session中间件使用)

1. 写入：
2. 读取：

# 九．模板引擎

## 1.jade

## 2.ejs

非侵入，非破坏的。

<%=name%>也可以输出JSON

<%-name%>是不转译输出，原来是什么就输出什么。