1. 创建node应用

引入http模块 var http = require("http"); 创建服务器

1. 小知识

Node中，一个文件就是一个模块

Node.js默认文件名为 .js

\_\_filename 表示当前正在执行的脚本的文件名。它将输出文件所在位置的绝对路径，

（如果在模块中，返回的值是模块文件的路径。）

\_\_dirname 表示当前执行脚本所在的目录

1. Exports 和 import

exports 是模块公开的接口，require 用于从外部获取一个模块的接口

Example.js

exports.world = function() {

console.log('Hello World');}

Main.js

Var example = require( ‘./example’ ); // 引入模块

Example.world(); // 调用方法

1. 新建服务器

var http = require("http");

http.createServer (function(request, response) {

response.writeHead(200, {"Content-Type": "text/plain"});

response.write("Hello World");

response.end();

} ).listen(8888);

var http = require("http");

function onRequest (request, response) {

response.writeHead(200, {"Content-Type": "text/plain"});

response.write("Hello World");

response.end();

}

http.createServer(onRequest).listen(8888);

1. 路由

查看 HTTP 请求，提取出请求的 URL 以及 GET/POST 参数

所有数据都会包含在 request 对象，该对象作为 onRequest() 回调函数的第一个参数传递

解析数据用到的模块 ：  url 和 querystring

1. 全局对象

\_\_filename 表示当前正在执行的脚本的文件名。它将输出文件所在位置的绝对路径，

（如果在模块中，返回的值是模块文件的路径。）

\_\_dirname 表示当前执行脚本所在的目录

setTimeout(cb, ms) 指定函数, 指定毫秒数

clearTimeout(t)

setInterval(cb, ms)

Console对象

Console 用于提供控制台标准输出

console.log() 向标准输出流打印字符并以换行符结束

console.info() 返回信息性消息 除谷歌，其他出现蓝色惊叹号

console.error() 输出错误消息。控制台在出现错误时会显示是红色的叉子

console.warn() 输出警告消息。黄色的惊叹号

console.dir() 对一个对象进行检查（inspect），并以易于阅读和打印的格式显示

console.time(label) 输出时间，表示计时开始

console.timeEnd(label) 结束时间，表示计时结束

console.assert( 表达式, 字符串） 判断某个表达式或变量是否为真 只有当第一为false，才会输出第二

process对象

描述进程的状态

Exit 当进程准备退出时触发

BeforeExit 当 node 清空事件循环，并且没有其他安排时触发这个事件

uncaughtException

Signal  当进程接收到信号时就触发

1. 模块
2. url 模块

方法 ：url.parse() url.format() url.resolve()

1. Url.parse ( **urlString [, boolean,boolean ]** )

第一个参数 ：url地址

第二个参数 ：true ： 返回的url对象中，query属性是对象 （默认为false ， 为字符串）

True : 

False : 

第三个参数 ：true ： 可以识别没有协议的urlString的host 和 hostname

1. url.format( urlObj ) 将传入的url对象编程一个url字符串并返回

url.format({

protocol:"http:",

host:"182.163.0:60",

port:"60"

});

结果 ： 'http://182.163.0:60'

1. url.resolve（from, to） 对传入的两个参数用"/"符号进行拼接

url.resolve("http://whitemu.com", "gulu");

返回值 ： 'http://whitemu.com/gulu'

1. util 模块

util.inspect（obj, [ showHidden ], [ depth ], [ colors ]）

将任意 对象转换为字符串 的方法， 它至少接受一个参数 object，即要转换的对象

showHidden : true 输出更多隐藏信息

Depth : 最大地柜层数

Color ： true 在终端展示效果更漂亮

util.isArray(object) 判断obj是否为数组

util.isRegExp(object) 判断obj是否为正则表达式

util.isDate(object) 判断obj是否为日期

util.isError(object) 判断obj是否为错误对象

1. 文件模块 fs

// 异步读取

fs.readFile('input.txt', function (err, data) {

if (err) { return console.error(err); }

console.log("异步读取: " + data.toString());

});

// 同步读取

var data = fs.readFileSync('input.txt');

console.log("同步读取: " + data.toString());

获取文件信息 (通过异步模式获取文件信息的语法格式)

fs.stat(path, callback) **path** - 文件路径

**callback** - 回调函数，带有两个参数如：(err, stats),  返回的**stats** 是 fs.Stats 对象

fs.stat('/Users/ode/fs.js', function (err, stats) {

console.log( stats.isFile() );         //true

})

关闭文件

fs.close(fd, callback) fd - 通过 fs.open() 方法返回的文件描述符

Callback : 没有参数

删除文件

fs.unlink(path, callback) Callback : 没有参数

创建目录

fs.mkdir(文件路径 [，目录权限], callback)

删除目录

fs.rmdir(文件路径 , callback) Callback : 没有参数

1. 获取文件信息get 和 post 请求

Get 请求 ： 参数 req.query 中

Post 请求 ： 参数 req.body 中

Get 请求解析

var http = require('http'); // 引入http模块

var url = require('url'); // 引入url模块

http.createServer(function(req, res){

res.writeHead(200, {'Content-Type': 'text/plain'});

// 解析 url 参数

var params = url.parse(req.url, true).query;

res.write("网站名：" + params.name);

res.write("网站 URL：" + params.url);

res.end(); }).listen(3000);

获取post请求内容

var http = require('http');

var querystring = require('querystring'); //

http.createServer( function ( req, res ) {

Var post = ‘’； // 定义post变量，暂存请求体信息

// 通过req的data时间监听函数， 每当接收到请求体的数据，就累加到post变量中

req.on(‘data’, function( chunk ) {

Post += chunk;

});

// 在end事件触发后， 通过querystring.parse将post解析为真正的post格式请求，然后返回给客户端

req.on(‘end, function( chunk ) {

Post += querystring.parse( post );

Res.end( util.inspect（post） );

});

} ).listen( 3000 );

九、express 模块

1. 简单实例

var express = require('express');

var app = express();

app.get('/', function (req, res) {

res.send('Hello World');

})

var server = app.listen(8081, function () {

var host = server.address().address

var port = server.address().port

console.log("应用实例，访问地址为 [http:// %s :](http://%s:) %s", host, port)

})

1. Express 使用回调函数的参数： request 和 response 对象来处理请求和响应的数据

request对象 表示http请求， 包含请求查询字符串、参数、内容、http头部等

常见属性 ：

|  |  |
| --- | --- |
| req.body | 获得请求主体 |
| Req.cookies | 获取cookies |
| Req.fresh/ req.state | 判断请求是否还新鲜 |
| Req.hostname | 获取主机名 |
| Req.ip | 获取IP |
| Req.originalUrl | 获取原始请求url |
| Req.path | 获取请求路径 |
| Req.protocol | 获取协议类型 |
| Req.query | 获取url的查询字符串 |
| Req.route | 获取当前匹配的路由 |
| Req.subdomains | 获取子域名 |
| Req.accepts | 检查可接受的请求的文档类型 |
| Req.get | 获取指定http请求头 |
| Req.is | 判断请求头Content-Type的MIME类型 |

Response对象 http响应

常见属性 ：

|  |  |
| --- | --- |
| Res.append（） | 追加指定http头 |
| Res.set（） | 在res.append后重置之前设置的头 |
| res.cookie(name，value) | 设置cookie |
| res.clearCookie() | 清除cookies |
| Res.download() | 传送指定文件 |
| Res.get() | 返回指定的HTTP头 |
| Res.json() | 传送json响应 |
| Res.jsonp() | 传送jsonp响应 |
| Res.location() | 只设置响应的location http头， 不设置状态码或close response |
| Res.redirect() |  |
| Res.send() | 传送响应 |
| Res.sendFile(‘path’) | 传送指定路径的文件 |
| res.set() | 设置HTTP头，传入object可以一次设置多个头 |
| res.status() | 设置HTTP状态码 |
| res.type() | 设置Content-Type的MIME类型 |

1. 案例

Var express = require(‘express’);

Var app = express();

// 设置路由

// get请求 主页

App.get(‘/’, function( req, res ){

console.log("主页 GET 请求"); res.send('Hello GET'); });

// post请求 主页

app.post('/', function (req, res) {

console.log("主页 POST 请求"); res.send('Hello POST'); })

// /del\_user 页面响应

app.get('/del\_user', function (req, res) {

console.log("/del\_user 响应 DELETE 请求"); res.send('删除页面'); })

// 对页面 abcd, abxcd, ab123cd, 等响应 GET 请求

app.get('/ab\*cd', function(req, res) {

console.log("/ab\*cd GET 请求"); res.send('正则匹配'); })

// 设置端口

App.listen( 3000);

1. 静态文件

Express 提供了内置的中间件 express.static 来设置静态文件如：图片， css, js等

使用 express.static 中间件来设置静态文件路径

例如，如果将图片，css, js文件放在 public 目录下，可以这么写：

App.use( express.static( ‘public’) );

静态资源放置 /public/image/logo.png

静态资源获取 /image/logo.png

1. 提交数据 和 获取数据

Index.html

<html>

<body>

<form action="http://127.0.0.1:8081/process\_get" method="GET">

Name : <input type="text" name="name"> <br>

Pass : <input type="text" name="pass">

<input type="submit" value="提交">

</form>

</body>

</html>

Server.js

var express = require('express');

var app = express();

var bodyParser = require('body-parser'); post请求使用

// 创建 application/x-www-form-urlencoded 编码解析

var urlencodedParser = bodyParser.urlencoded({ extended: false })

app.use(express.static('public'));

app.get('/index.htm', function (req, res) {

res.sendFile( \_\_dirname + "/" + "index.htm" ); })

app.get('/process\_get', function (req, res) { // 输出 JSON 格式

var response = { "name": req.query.name, // req.body.name post请求

"pass": req.query.pass }; // req.body.pass

res.end(JSON.stringify(response));

})

app.listen(3000);

1. 文件上传

使用 POST 方法，表单 enctype 属性设置为 multipart/form-data

Server.js

var fs = require("fs");

var multer = require('multer'); // 上传包

app.use(multer( { dest: '/tmp/'} ).array('image'));

App.post(‘/file\_upload’, function(req, res){

Var fileInfo = req.files[0] ;

Fs.readFile( fileInfo.path, function( err, data){

If ( err ){ console.log( err );}

Else {

Response = {

Message: ‘upload success’,

Filename: fileInfo.originalname

};

}

Res.end( JSON.stringify( response ));

});

});

App,listen( 3000 );

1. cookie管理

Express\_cookie.js

var cookieParser = require('cookie-parser');

app.use(cookieParser());

app.get('/', function(req, res) {

console.log("Cookies: ", req.cookies)

});

1. MySQL数据库

Sql.js

Var mysql = require(‘mysql’);

Var connection = mysql.createConnection( {

Host: ‘localhost’,

User: ‘root’,

Password: ‘123456’,

Database: ‘test’

} );

connection .connect();

connection.query(‘select \* from user’, function(err, result){

If (err ) {return; }

Console.log(‘result :’, result);

});