# express

简介 ：express简化了node的，相当于jquery。

使用步骤

1. 安装 cnpm install express
2. 导入 require('express')
3. 创建服务器 express() 相当于node的createServer
4. 监听端口
5. 处理请求地址

例子 ：

var express = require('express');

var server = express();

//相应用户的请求（当用户请求a.html时，会执行这个回调函数）

server.use('/a.html', function(req,res){

res.send('这是用express搭建的服务器');

res.end();

});

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

res.send('你要访问什么目录？');

res.end();

});

server.listen(8080);

！注意

req,res和原生的req,res不是都一模一样的。原生有的功能，express也有。只不过express里添加了一些新的功能。比如上面的send和原生的write。write有的功能send都有，并且send还能输出json、数组等信息给回客户端,而write就不行。

**express处理get、post请求**

除了上面说的server.use外，express还提供了get、post函数。用于处理get、post请求。

post请求例子

js代码

$.ajax({

url : '/logic',

data : {name : 'tom', age : 18},

type : 'POST',

success : function(data){

console.log( data );

},

error : function(error){

console.logt(error);

}

});

node代码

const express = require('express');

var querystring = require('querystring');

const server = express();

server.listen('8080');

server.post('/logic', function(req,res){

var str = '';

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

str+=data;

});

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

res.send(querystring.parse(str));

res.end();

});

});

server.use(express.static('www'));

console.log('server is running....');

**express.static**

express.static用于读取文件文件内容并显示到浏览器。相当于node的readFile。函数有一个参数，即存放html文件的文件夹名字。如 ：

server.use(express.static('./www'));

**body-parser**

上面的post请求例子中，我们需要先on一个data事件，在on一个end事件。这样实在太麻烦了。所以body-parser主要用于简化post、get请求。

我们修改下上面的post请求的例子

const express = require('express');

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

const server = express();

server.listen('8080');

server.use(bodyParser.urlencoded({

extended : true, //暂时不知道有什么用

limit : 2 \* 1024 \* 1024 //设置post能接受的最大数据长度(默认为100k)

}));

server.use('/logic', function(req,res){

var postMsg = req.body; //获取post数据

//var getMsg = req.query; //获取get数据

res.send(postMsg);

res.end();

});

server.use(express.static('www'));

console.log('server is running....');

**next函数**

next函数用于是否进行下一步。

例子：

const express = require('express');

const server = express();

server.listen('8080');

server.use('', function(req,res,next){

console.log(1);

next();

});

server.use('', function(req,res){

console.log(2);

});

console.log('server is running....');

在第一个use中，如果调用next()， 则会进入第二个ues函数。运行结果输出1、2。

**自己编写body-parse**

运用next函数的这一特性，我们可以自己来编写一个my-body-parser

my-body-parser :

const querystring = require('querystring');

module.exports = function(req, res, next){

var str = '';

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

str+=data;

});

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

req.body = querystring.parse(str);

next();

});

};

引用自己写的my-body-parser

var express = require('express');

var my\_body\_parser = require('./libs/my-body-parser');

var server = express();

server.listen(8080);

server.use(my\_body\_parser);

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

console.log(req.body);

});

！注意，上面的例子中，我们给req增加了一个body属性