二、express框架

1.express是什么？

Express 是一个基于 Node.js 平台的极简、灵活的 web 应用开发框架，它提供一系列强大的特性，帮助你创建各种 Web 和移动设备应用。

2.安装

mkdir myapp

cd myapp

npm init

npm install express –save

3.HelloWorld

在myapp目录下，创建app.js,复制以下代码：

var express = require('express');

var app = express();

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

res.send('Hello World!');

});

var server = app.listen(3000, function () {

var host = server.address().address;

var port = server.address().port;

console.log('Example app listening at http://%s:%s', host, port);

});

启动：node app.js

在浏览器中查看：<http://localhost:3000>

4.express脚手架

npm install express-generator –g

express myapp2

cd myapp2

npm install

启动自动生成的这个应用：

debug=myapp npm start

在浏览器中输入：

<http://localhost:3000/> 就可以看到这个应用的效果了。

5.基本路由

每一个路由都可以有一个或者多个处理器函数，当匹配到路由时，这个/些函数将被执行。

路由的定义由如下结构组成：app.METHOD(PATH, HANDLER)

在myapp文件夹下的app.js中输入如下代码：

/ 对网站首页的访问返回 "Hello World!" 字样

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

res.send('Hello World!');

});

// 网站首页接受 POST 请求

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

res.send('Got a POST request');

});

// /user 节点接受 PUT 请求

app.put('/user', function (req, res) {

res.send('Got a PUT request at /user');

});

// /user 节点接受 DELETE 请求

app.delete('/user', function (req, res) {

res.send('Got a DELETE request at /user');

});

分别访问不同的网址，查看效果。

可以引入正则：

// 匹配 abcd、abbcd、abbbcd等

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

res.send('ab+cd');

});

当处理逻辑较为复杂，需要多个函数时，使用回调函数处理路由：

var cb0 = function (req, res, next) {

console.log('CB0');

next();

}

var cb1 = function (req, res, next) {

console.log('CB1');

next();

}

var cb2 = function (req, res) {

res.send('Hello from C!');

}

app.get('/example/c', [cb0, cb1, cb2]);

另外一种更好的写法：

app.route('/book')

.get(function(req, res) {

res.send('Get a random book');

})

.post(function(req, res) {

res.send('Add a book');

})

.put(function(req, res) {

res.send('Update the book');

});

**链式编程，jquery当年特色之一。**

web最纯朴的形式：请求->处理->响应

结束处理：



当项目变大，路由变多，代码变得冗长，难以维护----🡪模块化

express.Router

新建一个js文件（birds.js）

var express = require('express');

var router = express.Router();

// 该路由使用的中间件

router.use(function timeLog(req, res, next) {

console.log('Time: ', Date.now());

next();

});

// 定义网站主页的路由

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

res.send('Birds home page');

});

// 定义 about 页面的路由

router.get('/about', function(req, res) {

res.send('About birds');

});

module.exports = router;

在最初要写路由代码的文件中：

var birds = require('./birds');

...

app.use('/birds', birds);

6.利用express托管静态文件

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

http://localhost:3000/images/kitten.jpg

http://localhost:3000/css/style.css

http://localhost:3000/js/app.js

http://localhost:3000/images/bg.png

http://localhost:3000/hello.html

如果静态文件在多个目录下，多次调用express.static

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

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

如果你希望所有通过 express.static 访问的文件都存放在一个“虚拟（virtual）”目录（即目录根本不存在）下面，可以通过为静态资源目录[指定一个挂载路径](http://www.expressjs.com.cn/4x/api.html" \l "app.use)的方式来实现，如下所示

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

http://localhost:3000/static/images/kitten.jpg

http://localhost:3000/static/css/style.css

http://localhost:3000/static/js/app.js

http://localhost:3000/static/images/bg.png

http://localhost:3000/static/hello.html

7.处理404,其他错误处理

404 只是意味着某些功能没有实现

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

res.status(404).send('Sorry cant find that!');

});

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

//res.status(404).send('Sorry cant find that!');

//res.status(404).sendfile('public/html/2.html');

res.redirect('/html/2.html')

});

其他错误处理：

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

console.error(err.stack);

res.status(500).send('Something broke!');

});