<http://www.expressjs.com.cn/starter/installing.html>

见素材express

**安**装

安装 Express 并将其保存到依赖列表中

npm install express --save

**Hello** world

var express = require('express');

var app = express();

app.get('/路由', function (req, res) {

});

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

});

启动应用：node 文件名

Express **应**用生成器

安装

npm install express-generator -g

在当前目录下创建命名为 myapp 的应用

express myapp

在myapp目录下安装所有依赖包

npm install

启动应用

DEBUG=myapp:\* npm start

**基**本路由

**每**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，处理器函数将被执行

路由的**定**义由如下结构组成

express实例.method(path,handler)

method是http请求方式。有些method名不是合规的javascript变量名，此时使用['method名']。all对应所有http请求方式。

path是路由。可以是字符串或者正则表达式或者字符串模式（?和+和\*和()是正则表达式的子集，-和.按照字面值解释。）

handler是处理器函数，可为单个，单个形式的多个，数组形式的多个，以上的混合。如果在处理器函数中以下响应方法一个也不调用，http请求会一直挂起。

方法 描述

res.download() 提示下载文件。

res.end() 终结响应处理流程。

res.json() 发送一个 JSON 格式的响应。

res.jsonp() 发送一个支持 JSONP 的 JSON 格式的响应。

res.redirect() 重定向请求。

res.render() 渲染视图模板。

res.send() 发送各种类型的响应。

res.sendFile 以八位字节流的形式发送文件。

res.sendStatus() 设置响应状态代码，并将其以字符串形式作为响应体

的一部分发送。

可使用 app.route() 创建路由路径的**链**式路由句柄

app.route('/路由')

.get(function(req, res) {

})

.post(function(req, res) {

})

**静**态文件

Express**内**置的express.static中间件可以托管静态资源文件

将静态资源文件所在的目录作为参数传递给express.static中间件就可以提供静态资源文件的访问了。可以多次调用 express.static 中间件。访问静态资源文件时，express.static 中间件会根据目录添加的顺序查找所需的文件。

app.use(express.static('静态资源文件所在的目录'));

如果希望所有通过express.static访问的静态资源文件都存放在一个**虚**拟目录下面可以

app.use('/虚拟目录', express.static('静态资源文件所在的目录'));

常见问题

如何处理 **4**04 ？

在 Express 中，404 并不是一个错误。错误处理器中间件并不捕获 404。因为 404 只是意味着某些功能没有实现。所需要做的就是在所有其他中间件的后面添加一个处理 404 的中间件。如下：

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

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

});

如何设置一个**错**误处理器？

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

});

路由

**中**间件

中间件是一个函数，可以访问请求对象（req）, 响应对象（res）, 和 web 应用中处于请求-响应循环流程中的中间件，一般被命名为 next 的变量。

中间件的**功**能包括

执行代码。

修改请求和响应对象

终结请求-响应循环。

调用堆栈中的下一个中间件

**如**果当前中间件没有终结请求-响应循环，则必须调用 next() 方法将控制权交给下一个中间件，否则请求就会挂起。

Express 应用可使用如下几种中间件：

**应**用级中间件

**路**由级中间件

**错**误处理中间件

**内**置中间件

**第**三方中间件

应用级中间件

express实例.method(path,handler)

express实例.use(function (req, res, next) {

next();

});// 没有挂载路径的中间件，应用的每个请求都会执行该中间件

express实例.use('/a/:b', function (req, res, next) {

next();

});// 挂载至 /a/:b 的中间件，任何指向 /a/:b 的请求都会执行它

路由级中间件

var express实例 = express();

var router = express.Router();

将express实例换成router，其他相同

express实例.use('/路由', router);

错误处理中间件

与应用级中间件和路由级中间件类似，但是参数为 (err, req, res, next)

应最后定义错误处理中间件

如果向 next() 传入参数（除了 ‘route’ 字符串），Express 会认为有错误，因此跳过后续其他非错误处理中间件

Express内置了一个错误处理中间件，它可以捕获应用中可能出现的任意错误。这个缺省的错误处理中间件将被添加到中间件堆栈的底部。

内置中间件

express.static(root, [options])是 Express 唯一内置的中间件

root 指静态资源文件所在的目录

options参数有如下属性。

属性 描述

dotfiles 是否对外输出文件名以**点**（.）开头的文件。可选值为

“allow”、“deny” 和 “ignore”

etag 是否启用 etag（**Entity**Tag）生成

extensions 设置文件**扩**展名备份选项

index 发送目录**索**引文件，设置为 false 禁用目录索引。

**last**Modified 设置 Last-Modified 头为文件在操作系统上的最后修改

日期。可能值为 true 或 false。

**max**Age 以毫秒或者其字符串格式设置 Cache-Control 头的

max-age 属性。

redirect 当路径为目录时，**重**定向至 “/”。

setHeaders **设**置 HTTP 头以提供文件的函数

第三方中间件

模板引擎

错误处理

调试

为 Express 设置代理

迁移到 Express 4

**数**据库集成

MySQL

npm install mysql

var mysql = require('mysql');

var connection = mysql.createConnection({

host : 'localhost',

user : 'user',

password : 'password'

});

connection.connect();

connection.query('', function(err, rows, fields) {

if (err) throw err;

});

connection.end();

安哈应

基每定链

静内虚

4错

中功如应路错 内点实扩索上大重设 第

数

安哈应基每定链 静内虚4错

中功如应路错 内点实扩索上大重设第数