## 针对不同的请求，Express提供了use方法的一些别名。比如，上面代码也可以用别名的形式来写。

var express = require("express");

var http = require("http");

var app = express();

app.all("\*", function(request, response, next) {

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

next();

});

app.get("/", function(request, response) {

response.end("Welcome to the homepage!");

});

app.get("/about", function(request, response) {

response.end("Welcome to the about page!");

});

app.get("\*", function(request, response) {

response.end("404!");

});

http.createServer(app).listen(1337);

上面代码的all方法表示，所有请求都必须通过该中间件，参数中的“\*”表示对所有路径有效。get方法则是只有GET动词的HTTP请求通过该中间件，它的第一个参数是请求的路径。由于get方法的回调函数没有调用next方法，所以只要有一个中间件被调用了，后面的中间件就不会再被调用了。

除了get方法以外，Express还提供post、put、delete方法，即HTTP动词都是Express的方法。

这些方法的第一个参数，都是请求的路径。除了绝对匹配以外，Express允许模式匹配。

app.get("/hello/:who", function(req, res) {

res.end("Hello, " + req.params.who + ".");

});

上面代码将匹配“/hello/alice”网址，网址中的alice将被捕获，作为req.params.who属性的值。需要注意的是，捕获后需要对网址进行检查，过滤不安全字符，上面的写法只是为了演示，生产中不应这样直接使用用户提供的值。

如果在模式参数后面加上问号，表示该参数可选。

app.get('/hello/:who?',function(req,res) {

if(req.params.id) {

res.end("Hello, " + req.params.who + ".");

}

else {

res.send("Hello, Guest.");

}

});

下面是一些更复杂的模式匹配的例子。

app.get('/forum/:fid/thread/:tid', middleware)

// 匹配/commits/71dbb9c

// 或/commits/71dbb9c..4c084f9这样的git格式的网址

app.get(/^\/commits\/(\w+)(?:\.\.(\w+))?$/, function(req, res){

var from = req.params[0];

var to = req.params[1] || 'HEAD';

res.send('commit range ' + from + '..' + to);

});

## express的all方法可以合并相同路径的不同请求方法，根据request对象的method来获取对应的方法进行对应的处理

|  |
| --- |
|  |