Node.js 函数 Node.js 全局对象

Node.js 路由

我们要为路由提供请求的 URL 和其他需要的 GET 及 POST 参数，随后路由需要根据这些数据来执行相应的代码。

因此，我们需要查看 HTTP 请求，从中提取出请求的 URL 以及 GET/POST 参数。这一功能应当属于路由还是服务器（甚至作为一个模块自身的功能）确实值得探讨，但这里暂定其为我们的HTTP服务器的功能。

我们需要的所有数据都会包含在 request 对象中，该对象作为 onRequest() 回调函数的第一个参数传递。但是为了解析这些数据，我们需要额外的 Node.JS 模块，它们分别是 url 和 querystring 模块。

```
                    url.parse(string).query
                                           |
            url.parse(string).pathname     |
                        |                  |
                        |                  |
                      ------ -------------------
http://localhost:8888/start?foo=bar&hello=world
                                   ---       -----
                                    |          |
                                    |          |
               querystring.parse(queryString)["foo"]    |
                                           |
                          querystring.parse(queryString)["hello"]
```

当然我们也可以用 querystring 模块来解析 POST 请求体中的参数，稍后会有演示。

现在我们来给 onRequest() 函数加上一些逻辑，用来找出浏览器请求的 URL 路径：

server.js 文件代码：

```
var http = require("http");
var url = require("url");
function start() {
function onRequest(request, response) {
var pathname = url.parse(request.url).pathname;
console.log("Request for " + pathname + " received.");
response.writeHead(200, {"Content-Type": "text/plain"});
response.write("Hello World");
response.end();
}
http.createServer(onRequest).listen(8888);
console.log("Server has started.");
}
exports.start = start;
```

好了，我们的应用现在可以通过请求的 URL 路径来区别不同请求了--这使我们得以使用路由（还未完成）来将请求以 URL 路径为基准映射到处理程序上。

在我们所要构建的应用中，这意味着来自 /start 和 /upload 的请求可以使用不同的代码来处理。稍后我们将看到这些内容是如何整合到一起的。

现在我们可以来编写路由了，建立一个名为 **router.js** 的文件，添加以下内容：

router.js 文件代码：

```
function route(pathname) {
console.log("About to route a request for " + pathname);
}
exports.route = route;
```

如你所见，这段代码什么也没干，不过对于现在来说这是应该的。在添加更多的逻辑以前，我们先来看看如何把路由和服务器整合起来。

我们的服务器应当知道路由的存在并加以有效利用。我们当然可以通过硬编码的方式将这一依赖项绑定到服务器上，但是其它语言的编程经验告诉我们这会是一件非常痛苦的事，因此我们将使用依赖注入的方式较松散地添加路由模块。

首先，我们来扩展一下服务器的 start() 函数，以便将路由函数作为参数传递过去，**server.js** 文件代码如下

server.js 文件代码：

```
var http = require("http");
var url = require("url");
function start(route) {
function onRequest(request, response) {
var pathname = url.parse(request.url).pathname;
console.log("Request for " + pathname + " received.");
route(pathname);
response.writeHead(200, {"Content-Type": "text/plain"});
response.write("Hello World");
response.end();
}
http.createServer(onRequest).listen(8888);
console.log("Server has started.");
}
exports.start = start;
```

同时，我们会相应扩展 index.js，使得路由函数可以被注入到服务器中：

index.js 文件代码：

```
var server = require("./server");
var router = require("./router");
server.start(router.route);
```

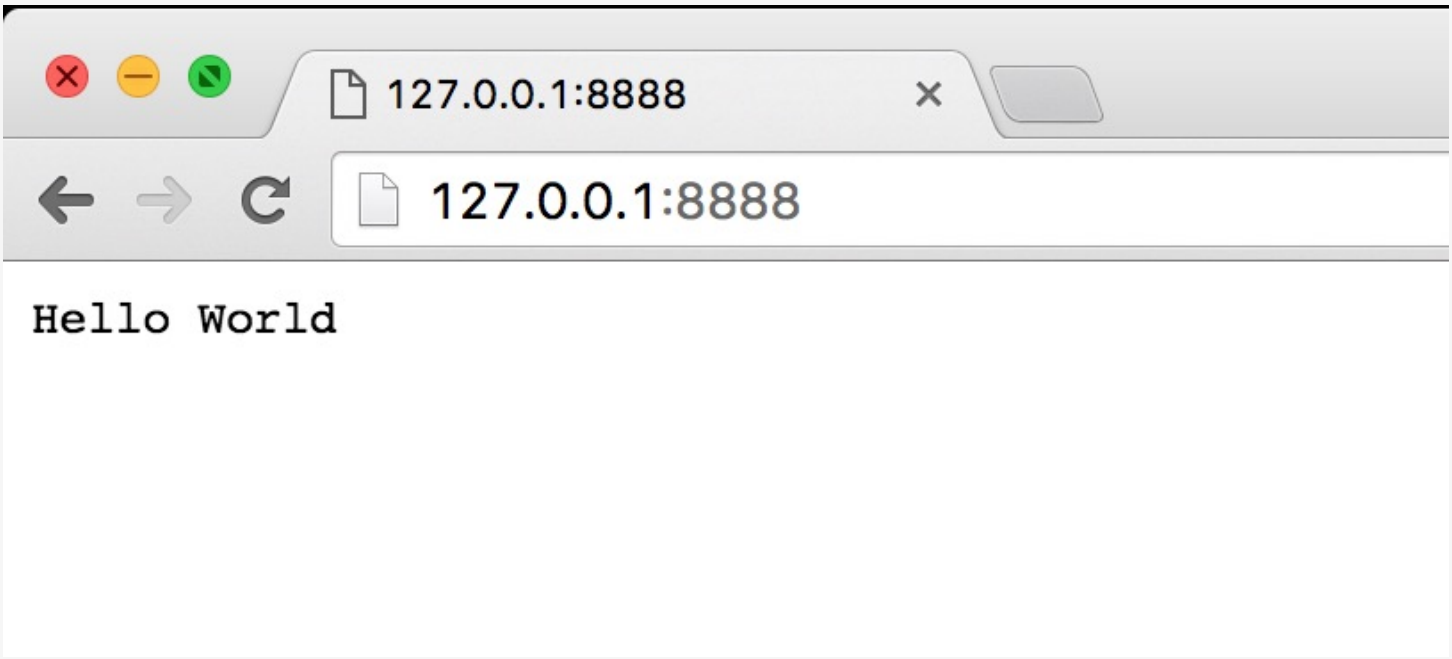
在这里，我们传递的函数依旧什么也没做。

如果现在启动应用（node index.js，始终记得这个命令行），随后请求一个URL，你将会看到应用输出相应的信息，这表明我们的HTTP服务器已经在使用路由模块了，并会将请求的路径传递给路由：

```
$ node index.js
Server has started.
```

以上输出已经去掉了比较烦人的 /favicon.ico 请求相关的部分。

浏览器访问 **http://127.0.0.1:8888/**，输出结果如下：



Node.js 函数 Node.js 全局对象

点我分享笔记