Node.js 中 Express 框架路由机制的研究

Study of Express Framework´s Routing Mechanism in Node.js

程桂花 沈 炜 何松林 张珂杰 （浙江理工大学,浙江 杭州 310018）

摘要:Express 作为 Node.js 的一个 Web 开发框架已经被工程师们所熟知。使用该框架的各种特性可以更加方便、快速地开发出一个比较完整的 Web 应用程序。和其他 Web 开发框架一样,Express 隐藏了代码背后的繁琐,开发者可以将工作的重心放在编写代码上，但 Express 更为优秀的是为开发者们提供了一种路由机制，分析了 Express 路由机制的特性,并研究该路由机制在程序开发中的应用方式。

关键词:Node.js,Express,路由机制,Web 开发框架

Abstract:Express,as one of the Web development frameworks,has been well known by engineers.Its application of various features brings convenience as well as effectiveness for building a relatively integral Web application.Like other Web development frameworks,Express has concealed the complexity of underlying codes,which contributes to the concentration of developers.Whereas,more desirably,it provides developers with a kind of routing mechanism,whose feature and applications in the process of development is explored in this paper.

Keywords:Node.js,Express,routing mechanism,Web development frameworks

1 Express 框架简述

Express 是一个基于 Node.js 的非常优秀的服务端开发框架[1]。Express 之所以可以快速地被工程师们接受,并得以广泛应用，关键就在于其具备的一系列的特性：它可以快速进行开发,拥有灵活的扩展机制,使用简单方便,此外它还有着强大的路由、多模块支持等特性[2]。下面将具体介绍其部分特性：

1)快速开发:不需要手写很多的代码,只需要一行命令,就可以生成 Express 框架的基础摸板。

2)灵活的扩展机制:Express 框架可以通过其扩展机制,很方便地加入其他的功能。

3)使用简单:Express 的 API 都非常的直观、简单,还有详细的 API 文档供查看。

如下是一个简单的 Express 示例应用程序：

```
var express = require(´express´); //引用 express 模块
var http = require(´http´);
var app = express( );
app.get (´/index.html´, function ( req, res) {
   res.writeHead (200,{´Content-Type´ : ´text/html ´});
   res.write (´<head><meta charset="utf-8"/></head>´);
   res.end (´你好\n´);
});
   app.listen(1337,"127.0.0.1");
```

2 路由机制

2.1 什么是路由机制

路由就是指如何定义应用的端点(URIs)以及如何响应客户端的请求[3]。路由是由一个 URI、HTTP 请求和若干个句柄组成(HTTP 主要有 GET、POST 等请求方法),它的具体结构为:app.method (path,[callback…],callback)。在这其中,app 是 Express 的一个实例对象,method 是 HTTP 中的一个请求方法,path 是服务器上面的路径,而 callback 是当前路由匹配时要执行的函数,URL 即统一资源定位符,是对可以从互联网上得到的资源的位置和访问方法的一种简洁的表示，简单的说就是互联网上标准资源的地址。互联网上的每一个文件都有其唯一的一个 URL;路由句柄可以是一个函数,或者是一个函数数组,还可以是函数和函数数组的混合,其作用和中间件极为相似。

如下是一个基本的路由示例：

```
var express = require(´express´);
var app = express();
app.get(´/´, function(req, res) {
 res.send(´hello world´);
});
```

2.2 路由规则

(1)* 通配 URL

通配符 * 号,可以添加在 URL 中,并可代替任意的字符,在正则表达式中也有 * 字符,但是它与通配符不同,且正则表达式与通配符只有一个适用,而不能同时使用。

(2)/:id 的占位标识符 URL

/:id 的占位标识符 URL 就是依据 id 来匹配页面的 URL,如可以用 http://loaclhost:3000/test/30,可以依据 id 为 30,得到想要的页面；而如果在加上 http://loaclhost:3000/test/30/xx 则无法继续使用。

(3) Next()权限控制转移

Express 路由控制中的 next()功能是当定义了多个路由时,next()会使匹配的 URL 按顺序执行,但是如果不使用 next()进行权限转移,那么程序只会执行第一个满足路由规则的 URL。

Express 中定义了和 HTTP 请求相对应的路由方法如:get,post、put、head、delete、options、trace、copy、lock、mkcol、move、purge、propfind、proppatch、unlock、report、mkactivity、checkout、merge、m –search、notify、subscribe、unsubscribe、patch、search 和 connect。而 app.all()是 Express 中的一个特殊的路由方法,它没有对应的 HTTP 方法,其作用就是为了给一个路径上的所有请求加载中间件。

2.3 路由

当使用 Express 的时候,可以通过如下的方法注册路由：

```
app.get('/',function(req,res){
        res.send("hello world!");
});
```

从表面看，路由的 get 方法将 URL 中的 path 与后台的处理程序关联起来[4]，下面将介绍其具体的实现方法。

从 application.js 文件的源码中可以看出路由中的 HTTP 请求方法都是存在于 methods 的数组里面的，运用方法时就是对其数组进行遍历，给 app 实例添加方法，如 app.get()、app.post()、app.put()、app.query()等。方法如下：

```
methods.forEach(function(methods){
    app[method]=function(path){
        if('get'==method&&1==arguments.length)
            return this.set(path);
            this.lazyrouter();
var route=this._router.route(path);
route[method] .apply(route,slice.call(arguments,1));
return this;
};
});
```

由上面的程序段可知：在 HTTP 的这些方法中，都是先通过调用 lazyrouter 来实例化一个 Router 对象，然后再调用 this._router.route()函数实例化一个 Route 对象，最后再调用 route[method]方法将 apply()传入的对应的处理程序完成 path 和 handler 的关联。

在使用 Express 的时候，所有的请求都是交给 app 的，通过 app.handle 来处理，在 app 中有最重要的一个属性就是_router，即 Router，简单地说 Router 就相当于一个中间件容器，存在于里面的中间件分为路由中间件和其他中间件。Router 的方法大致有以下几种：

1)use()：使用中间件，作用就是向 stack 中添加一个 layer 对象。

2)route()：创建路由中间件，即创建一个 Route 对象，并使用 Route 对象创建一个 layer 对象，然后将此 layer 对象添加到 stack 中。

3)method()：创建路由中间件，并添加处理函数(调用 route()函数)。

在 Router 的 stack 中存放着多个中间件，每一个中间件都是一个 Layer 对象，如果该中间件是一个路由中间件，则相应的 Layer 对象的 route 属性会指向一个 Route 对象，表示一条路由。Router 中的 Layer 对象都有一个 route 属性：如果是路由中间件，则该属性为一个 Route 对象，否则为 undefined。如图 1 是 Router.stack 和 Route.stack 的关联图：

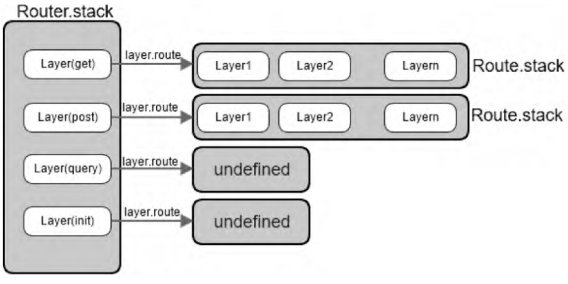

图 1 Router.stack 和 Route.stack 关联模型图

路由对于请求的处理，实质上就是 app.handle 调用 router.handle，而 router.handle 处理过程就是依次对 router.stack 中存放的中间件进行调用。如图 2，当处理一个请求时，会依次通过 router.stack 中的 Layer，即图中的 Layer1、Layer2、Layer3、Layer4、Layer5。如遇到路由中间件，图中 Layer2 和 Layer3，则会依次通过 route.stack 中的 Layer 对象。即图中 Layer2-1、Layer2-2、Layer3-1、Layer3-2、Layer3-3。然后再由路由中间件的 Layer 顺序通过 Router 的 Layer。

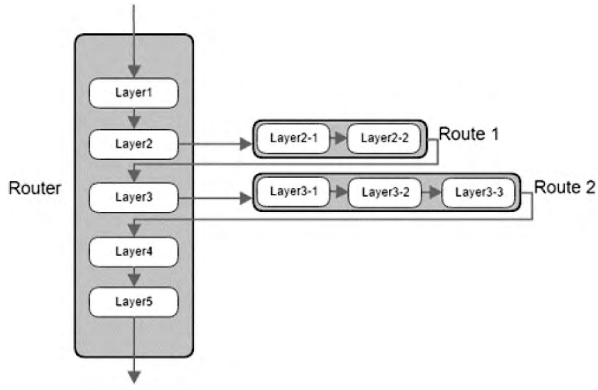

图 2 请求处理顺序图

当然，对于请求的处理是有具体的过程的，首先，对于 router.stack 中的每一个 Layer 对象，会先判断其是否匹配请求路径，如果不匹配，则跳过该 Layer，继续判断 route.stack 中接下来的 Layer；如果匹配，再判断其是不是路由中间件，如果不是路由中间件，则开始执行该中间件的函数；若是路由中间件，则继续判断该路由中间件的路由对象是否能够处理请求的 HTTP 方法，如果不能处理，则直接跳到 Router 的下一个 Layer；若能够处理，则可以对在 Route.stack 中与请求的 HTTP 方法匹配 Layer 对象依次执行。其示例图如图 3 所示。

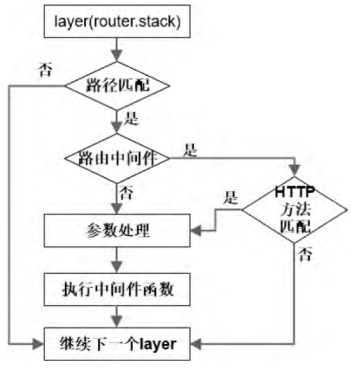

图 3 请求处理图

以上步骤就是路由机制处理请求的过程，由这些过程可以看到路由在隐藏的代码做了很多的工作，将繁琐的过程都封装起来，我们要做的就是调用其方法，这充分体现了路由机制存在的必要性。

3 结束语

通过对 Express 框架的路由机制的研究，了解到了路由内在的工作机制，在起初的路由机制中，每增加一个页面就要进行相应的配置，这一点是不太方便的，随后通过约定路由规则实现了显著改进。现在，已经出现了很多基于 Express 而建立的应用，如此更凸显了 Express 的可用性。当然这都归功于其改进的路由机制，因此对 Express 框架的路由机制的研究是很有意义的。

参考文献

[1]McManus Simon.Write Maintainable Web Apps With Experss [J]. Net, 2014 (Mar.TN.251).

[2]朱建兵.基于 Node.JS 高并发网络应用架构的研究与实现[D].北京：北京邮电大学，2014

[3]Mr_yong.Nodejs+Mongodb 系列教程之（3/5）——理解路由和中间件[EB/OL].2015-12-18.

[4]zjh-neverstop. 从 Express 源码中探析其路由机制[EB/OL].2015

[收稿日期：2016.3.25]