Node.js Express 框架 Node.js 多进程

Node.js RESTful API

什么是 REST？

REST即表述性状态传递（英文：Representational State Transfer，简称REST）是Roy Fielding博士在2000年他的博士论文中提出来的一种软件架构风格。

表述性状态转移是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是RESTful。需要注意的是，REST是设计风格而不是标准。REST通常基于使用HTTP，URI，和XML（标准通用标记语言下的一个子集）以及HTML（标准通用标记语言下的一个应用）这些现有的广泛流行的协议和标准。REST 通常使用 JSON 数据格式。

HTTP 方法

以下为 REST 基本架构的四个方法：

- **GET** - 用于获取数据。
- **PUT** - 用于更新或添加数据。
- **DELETE** - 用于删除数据。
- **POST** - 用于添加数据。

RESTful Web Services

Web service是一个平台独立的，低耦合的，自包含的、基于可编程的web的应用程序，可使用开放的XML（标准通用标记语言下的一个子集）标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。

基于 REST 架构的 Web Services 即是 RESTful。

由于轻量级以及通过 HTTP 直接传输数据的特性，Web 服务的 RESTful 方法已经成为最常见的替代方法。可以使用各种语言（比如 Java 程序、Perl、Ruby、Python、PHP 和 Javascript[包括 Ajax]）实现客户端。

RESTful Web 服务通常可以通过自动客户端或代表用户的应用程序访问。但是，这种服务的简便性让用户能够与之直接交互，使用它们的 Web 浏览器构建一个 GET URL 并读取返回的内容。

更多介绍，可以查看：RESTful 架构详解

创建 RESTful

首先，创建一个 json 数据资源文件 users.json，内容如下：

```
{
   "user1" : {
      "name" : "mahesh",
      "password" : "password1",
      "profession" : "teacher",
      "id": 1
```

```
    },
    "user2" : {
       "name" : "suresh",
       "password" : "password2",
       "profession" : "librarian",
       "id": 2
    },
    "user3" : {
       "name" : "ramesh",
       "password" : "password3",
       "profession" : "clerk",
       "id": 3
    }
}
```

基于以上数据，我们创建以下 RESTful API：

序号	URI	HTTP 方法	发送内容	结果
1	listUsers	GET	空	显示所有用户列表
2	addUser	POST	JSON 字符串	添加新用户
3	deleteUser	DELETE	JSON 字符串	删除用户
4	:id	GET	空	显示用户详细信息

获取用户列表：

以下代码，我们创建了 RESTful API **listUsers**，用于读取用户的信息列表， server.js 文件代码如下所示：

```
var express = require('express');
var app = express();
var fs = require("fs");

app.get('/listUsers', function (req, res) {
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       console.log( data );
       res.end( data );
   });
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port

  console.log("应用实例，访问地址为 http://%s:%s", host, port)
```

```
})
```

接下来执行以下命令：

```
$ node server.js
应用实例，访问地址为 http://0.0.0.0:8081
```

在浏览器中访问 http://127.0.0.1:8081/listUsers，结果如下所示：

```
{
   "user1" : {
      "name" : "mahesh",
      "password" : "password1",
      "profession" : "teacher",
      "id": 1
   },
   "user2" : {
      "name" : "suresh",
      "password" : "password2",
      "profession" : "librarian",
      "id": 2
   },
   "user3" : {
      "name" : "ramesh",
      "password" : "password3",
      "profession" : "clerk",
      "id": 3
   }
}
```

添加用户

以下代码，我们创建了 RESTful API **addUser**， 用于添加新的用户数据，server.js 文件代码如下所示：

```
var express = require('express');
var app = express();
var fs = require("fs");

//添加的新用户数据
var user = {
   "user4" : {
      "name" : "mohit",
      "password" : "password4",
      "profession" : "teacher",
      "id": 4
```

```
    }
}

app.get('/addUser', function (req, res) {
   // 读取已存在的数据
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       data = JSON.parse( data );
       data["user4"] = user["user4"];
       console.log( data );
       res.end( JSON.stringify(data));
   });
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port
  console.log("应用实例，访问地址为 http://%s:%s", host, port)

})
```

接下来执行以下命令：

```
$ node server.js
应用实例，访问地址为 http://0.0.0.0:8081
```

在浏览器中访问 http://127.0.0.1:8081/addUser，结果如下所示：

```
{ user1:
   { name: 'mahesh',
     password: 'password1',
     profession: 'teacher',
     id: 1 },
  user2:
   { name: 'suresh',
     password: 'password2',
     profession: 'librarian',
     id: 2 },
  user3:
   { name: 'ramesh',
     password: 'password3',
     profession: 'clerk',
     id: 3 },
  user4:
   { name: 'mohit',
     password: 'password4',
     profession: 'teacher',
```

```
      id: 4 }
}
```

显示用户详情

以下代码，我们创建了 RESTful API **:id（用户id）**， 用于读取指定用户的详细信息，server.js 文件代码如下所示：

```
var express = require('express');
var app = express();
var fs = require("fs");

app.get('/:id', function (req, res) {
   // 首先我们读取已存在的用户
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       data = JSON.parse( data );
       var user = data["user" + req.params.id]
       console.log( user );
       res.end( JSON.stringify(user));
   });
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port
  console.log("应用实例，访问地址为 http://%s:%s", host, port)

})
```

接下来执行以下命令：

```
$ node server.js
应用实例，访问地址为 http://0.0.0.0:8081
```

在浏览器中访问 http://127.0.0.1:8081/2，结果如下所示：

```
{
   "name":"suresh",
   "password":"password2",
   "profession":"librarian",
   "id":2
}
```

删除用户

以下代码，我们创建了 RESTful API **deleteUser**， 用于删除指定用户的详细信息，以下实例中，用户 id 为 2，server.js 文件代码如下所示：

```
var express = require('express');
var app = express();
var fs = require("fs");

var id = 2;

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

   // First read existing users.
   fs.readFile( __dirname + "/" + "users.json", 'utf8', function (err, data) {
       data = JSON.parse( data );
       delete data["user" + 2];

       console.log( data );
       res.end( JSON.stringify(data));
   });
})

var server = app.listen(8081, function () {

  var host = server.address().address
  var port = server.address().port
  console.log("应用实例，访问地址为 http://%s:%s", host, port)

})
```

接下来执行以下命令：

```
$ node server.js
应用实例，访问地址为 http://0.0.0.0:8081
```

在浏览器中访问 http://127.0.0.1:8081/deleteUser，结果如下所示：

```
{ user1:
   { name: 'mahesh',
     password: 'password1',
     profession: 'teacher',
     id: 1 },
  user3:
   { name: 'ramesh',
     password: 'password3',
     profession: 'clerk',
```

```
      id: 3 }
}
```

← Node.js Express 框架 Node.js 多进程 →

2 篇笔记

写笔记

如果有人和我一样遇到乱码的问题，除了不是文件读取的编码问题，可以考虑为浏览器自动为没有在 `html > head` 中解释编码的 **html** 编码为本地默认编码。可以主动添加 **head**：

```
res.setHeader('Content-Type', 'text/html; charset=utf8');
```

peng 8个月前 (07-07)

如果在同一个 server.js 里创建多个 RESTful API ， 并且 `:id` 放在前边， 那么它会拦截其他的请求， 比如：

```
var id = 2;
// 删除用户
app.get('/deleteUser', function (req, res) {
   //  读取已存在用户
   fs.readFile( __dirname + "/" + "user.json", 'utf8', function (err, data) {
       data = JSON.parse( data );
       delete data["user" + id];
       console.log( data );
       res.end( JSON.stringify(data));
   });
});

// 查询用户信息 (放到前边会拦截其他请求)
app.get('/:id', function(req, resp){
    fs.readFile(__dirname + '/user.json', 'utf-8', function(err, data){
        if(err){
            console.log(err.stack);
            return;
        }

        data = JSON.parse(data);
        var user = data['user' + req.params.id];
        console.log(user);
        resp.end(JSON.stringify(user));
    } );
});
```

在浏览器中访问 **http://127.0.0.1:8081/deleteUser**， 控制台打印信息如下：

```
应用实例，访问地址为 http://0.0.0.0:8081
undefined
undefined
```

JMH 8个月前 (07-27)