Node.js 文件系统 Node.js Buffer(缓冲区)

Node.js GET/POST请求

在很多场景中，我们的服务器都需要跟用户的浏览器打交道，如表单提交。

表单提交到服务器一般都使用 GET/POST 请求。

本章节我们将为大家介绍 Node.js GET/POST请求。

获取GET请求内容

由于GET请求直接被嵌入在路径中，URL是完整的请求路径，包括了?后面的部分，因此你可以手动解析后面的内容作为GET请求的参数。

node.js 中 url 模块中的 parse 函数提供了这个功能。

实例

```
var http = require('http');
var url = require('url');
var util = require('util');
http.createServer(function(req, res){
res.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
res.end(util.inspect(url.parse(req.url, true)));
}).listen(3000);
```

在浏览器中访问 **http://localhost:3000/user?name=菜鸟教程&url=www.runoob.com** 然后查看返回结果:


```
Url {
  protocol: null,
  slashes: null,
  auth: null,
  host: null,
  port: null,
  hostname: null,
  hash: null,
  search: '?name=%E8%8F%9C%E9%B8%9F%E6%95%99%E7%A8%8B&url=www.runoob.com',
  query: { name: '菜鸟教程', url: 'www.runoob.com' },
  pathname: '/user',
  path: '/user?name=%E8%8F%9C%E9%B8%9F%E6%95%99%E7%A8%8B&url=www.runoob.com',
  href: '/user?name=%E8%8F%9C%E9%B8%9F%E6%95%99%E7%A8%8B&url=www.runoob.com' }
```

获取 URL 的参数

我们可以使用 url.parse 方法来解析 URL 中的参数，代码如下：

实例

```
var http = require('http');
var url = require('url');
var util = require('util');
http.createServer(function(req, res){
res.writeHead(200, {'Content-Type': 'text/plain'});
// 解析 url 参数
```

```
var params = url.parse(req.url, true).query;
res.write("网站名: " + params.name);
res.write("\n");
res.write("网站 URL: " + params.url);
res.end();
}).listen(3000);
```

在浏览器中访问 **http://localhost:3000/user?name=菜鸟教程&url=www.runoob.com** 然后查看返回结果:

获取 POST 请求内容

POST 请求的内容全部的都在请求体中，http.ServerRequest 并没有一个属性内容为请求体，原因是等待请求体传输可能是一件耗时的工作。

比如上传文件，而很多时候我们可能并不需要理会请求体的内容，恶意的POST请求会大大消耗服务器的资源，所以 node.js 默认是不会解析请求体的，当你需要的时候，需要手动来做。

基本语法结构说明

```
var http = require('http');
var querystring = require('querystring');
http.createServer(function(req, res){
// 定义了一个post变量，用于暂存请求体的信息
var post = '';
// 通过req的data事件监听函数，每当接受到请求体的数据，就累加到post变量中
req.on('data', function(chunk){
post += chunk;
});
// 在end事件触发后，通过querystring.parse将post解析为真正的POST请求格式，然后向客户端返回。
req.on('end', function(){
post = querystring.parse(post);
res.end(util.inspect(post));
});
}).listen(3000);
```

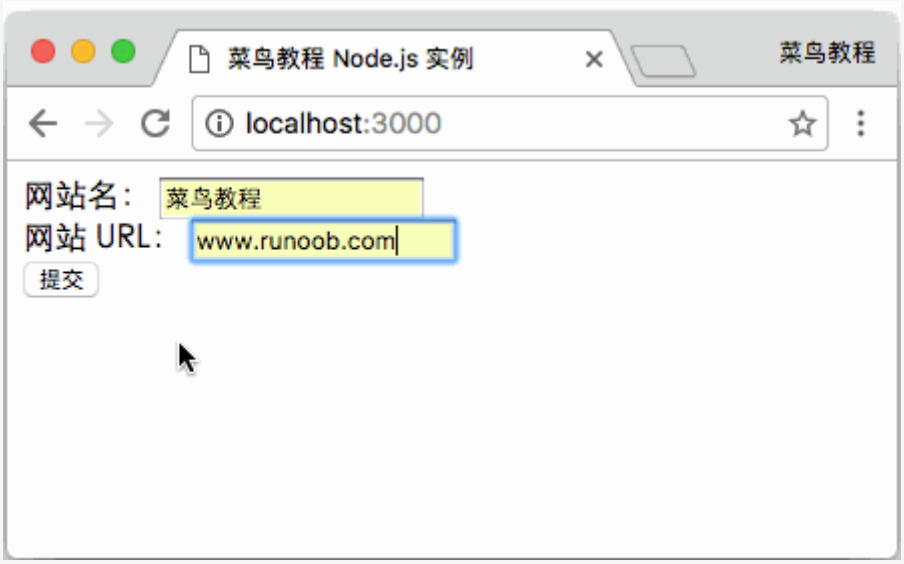
以下实例表单通过 POST 提交并输出数据：

实例

```
var http = require('http');
var querystring = require('querystring');
var postHTML =
'<html><head><meta charset="utf-8"><title>菜鸟教程 Node.js 实例</title></head>' +
'<body>' +
'<form method="post">' +
'网站名: <input name="name"><br>' +
'网站 URL: <input name="url"><br>' +
```

```
'<input type="submit">' +
'</form>' +
'</body></html>';
http.createServer(function (req, res) {
var body = "";
req.on('data', function (chunk) {
body += chunk;
});
req.on('end', function () {
// 解析参数
body = querystring.parse(body);
// 设置响应头部信息及编码
res.writeHead(200, {'Content-Type': 'text/html; charset=utf8'});
if(body.name && body.url) { // 输出提交的数据
res.write("网站名: " + body.name);
res.write("<br>");
res.write("网站 URL: " + body.url);
} else { // 输出表单
res.write(postHTML);
}
res.end();
});
}).listen(3000);
```

执行结果 Gif 演示：



← Node.js 文件系统 Node.js Buffer(缓冲区) →