Express.js中文文档－Request对象

2016年03月13日 667 声明

`request` 对象代表HTTP请求，及请求中的查询字符串、请求体、HTTP头等。Express的 `request` 对象是对Node.js `http.IncomingMessage` (http://itbilu.com/nodejs/core/N1okQ7Eh.html#incomingMessage)对象的扩展，在 `IncomingMessage` 对象基础上增加了一些Web应用中常用的属性和方法。按照习惯，在本文档中 `request` 对象被表示为 `req` 。

1. `Request` 对象

`Request` 对象继承自Node.js的 `IncomingMessage` 对象， `IncomingMessage` 中的属性、事件、方法，在 `Request` 对象中都可以使用。它在收到用户请求时被自动创建，我们可以在路由处理器（即：回调函数）中访问这个对象。

```
app.get('/user/:id', function(request, response) {
  response.send('user ' + request.params.id);
});
```

按习惯我们可以将 `request` 简写为 `req` ：

```
app.get('/user/:id', function(req, res) {
  res.send('user ' + req.params.id);
});
```

2. 属性

2.1 `req.app`

`req.app` 是对Expressapplication (http://itbilu.com/nodejs/npm/VJ5TlyRnl.html)实例的引用，通过个属性我们可以访问 `app` 对象中的设置、属性、中间件等。

如果我们创建了一个模块，如果我们仅 `exports` 导出了中间件函数，且在主文件中 `require()` 引用了它，那么我们可以在Express的 `req.app` 实例中访问这个中间件。

如：

```
//index.js
app.get('/viewdirectory', require('./mymiddleware.js'))
```

```
//mymiddleware.js
module.exports = function (req, res) {
  res.send('The views directory is ' + req.app.get('views'));
});
```

2.2 req.baseUrl

路由实例被挂载后的URL路径。

req.baseUrl 属性类似 app 对象的 mountpath (http://itbilu.com/nodejs/npm/VJ5TlyRnl.html#app-properties-mountpath)属性，但 app.mountpath 会返回匹配到的路径模式。如：

```
var greet = express.Router();

greet.get('/jp', function (req, res) {
  console.log(req.baseUrl); // /greet
  res.send('Konichiwa!');
});

app.use('/greet', greet); // 在'/greet'上挂载路由处理器
```

即使你使用一个路径模式或一组路径模式加载路由处理器， baseUrl 仍会返回匹配的字符串而不是路由模式（s）。下面是一个加载两个路径模式的示例：

```
app.use(['/gre+t', '/hel{2}o'], greet); // 在'/gre+t' 和 '/hel{2}o'上挂载路由处理器
```

2.3 req.body

表示请求体的一个 key-value 数据对象。默认值是 undefined ， 其在 body-parser 和 multer 等 body 解析器解析后可访问。

我们可以在Express的 app.js 文件中添加了一个 body 解析器器中间件：

```
var app = require('express')();
var bodyParser = require('body-parser');
var multer = require('multer'); // v1.0.5
var upload = multer(); // 解析 multipart/form-data 类型数据

app.use(bodyParser.json()); // 解析 application/json 类型数据
app.use(bodyParser.urlencoded({ extended: true })); // 解析 application/x-www-form-urlencoded 类型数据

app.post('/profile', upload.array(), function (req, res, next) {
  console.log(req.body);
  res.json(req.body);
});
```

2.4 req.cookies

当使用 cookie-parser 中间件解析 cookie 后， req.cookies 属性是一个表示 cookie 的 key-value 对象。没有使用 cookie 时，其值为 {} 。

```
// Cookie: name=tj
req.cookies.name
// => "tj"
```

2.5 req.fresh

表示请求是"fresh"（新请求）的，与 req.stale 属性要反。

如果是 true 状态，如果 cache-control (http://itbilu.com/other/relate/EJ3fKUwUx.html#http-request-headers)请求头中没有 no-cache 指令，那么以下项都为 true ：

- if-modified-since 请求头已经指定且 last-modified 请求头等于或早于 modified 响应头
- if-none-match 请求头是 *
- if-none-match 请求头指令解析后与 etag 响应头不匹配

```
req.fresh
// => true
```

2.6 req.hostname

表示来自HTTP头信息中主机的主机名。

如果信任代理（trust proxy）的设置值不是 `false` ，这个属性会替换 `X-Forwarded-Host` 头字段的值。这个头一般由客户端或代理设置。

```
// Host: "itbilu.com:3000"
req.hostname
// => "itbilu.com"
```

2.7 req.ip

表示客户端请求的远程主机IP。

如果信任代理（trust proxy）的设置值不是 `false` ，这个属性会从 `X-Forwarded-Host` 头字段中获取。这个头一般由客户端或代理设置。

```
req.ip
// => "127.0.0.1"
```

2.8 req.ips

如果信任代理（trust proxy）的设置值不是 `false` ，此属性包含一个表示指向 `X-Forwarded-Host` 请头中IP的数组。否则，是一个空数组。这个头一般由客户端或代理设置。

如，如果 `X-Forwarded-For` 是客户端 `proxy1` 、 `proxy2` ，那么 `req.ips` 属性值为 `["client", "proxy1", "proxy2"]`

2.9 req.method

表示客户端的HTTP请求方法，如：GET、PUT、POST等。

```
var method = req.method.toLowerCase();

if (method==='get'){
  // GET请求处理
} else if (method==='put'){
  // PUT请求处理
}
```

2.10 req.originalUrl

这是一个Node.js `http` 模块的属性。这个属性类似于 `req.url` ，但是它保留原始请求的URL，它允许你重写 `req.url` 内部的路径目的地。

```
// GET /search?q=something
req.originalUrl
// => "/search?q=something"
```

2.11 req.params

这是一个表示路径参数的对象。如，我们使用 `/user/:name` 路径时，那么 `req.params.name` 表示路径中的 `:name` 属性。该对象默认值为 `{}` 。

```
// GET /user/tj
req.params.name
// => "tj"
```

2.12 req.path

表示请求URL中的路径部分。

```
// example.com/users?sort=desc
req.path
// => "/users"
```

2.13 req.protocol

表示HTTP请求的协议类型： `http` 或 `https` (http://itbilu.com/nodejs/core/4JKY7REke.html)。

如果信任代理（trust proxy）的设置值不是 `false` ，这个属性会从 `X-Forwarded-Proto` 头字段中获取。这个头一般由客户端或代理设置。

```
req.protocol
// => "http"
```

2.14 req.query

这个属性表示URL查询字符串（'?'之后的部分）的 `key-value` 对象。如果请求中不包括查询字符串，这个属性的值为 `{}` 。

```
// GET /search?q=tobi+ferret
req.query.q
// => "tobi ferret"

// GET /shoes?order=desc&shoe[color]=blue&shoe[type]=converse
req.query.order
// => "desc"

req.query.shoe.color
// => "blue"

req.query.shoe.type
// => "converse"
```

2.15 req.route

表示当前请示中匹配到的路径字符串。

```
app.get('/user/:id?', function userIdHandler(req, res) {
  console.log(req.route);
  res.send('GET');
});
```

```
{ path: '/user/:id?',
  stack:
   [ { handle: [Function: userIdHandler],
       name: 'userIdHandler',
       params: undefined,
       path: undefined,
       keys: [],
       regexp: /^\/?$/i,
       method: 'get' } ],
  methods: { get: true } }
```

2.16 req.secure

表示是否使用 TLS 连接的一个布尔值。

```
'https' == req.protocol;
```

2.17 req.signedCookies

当使用 cookie (http://itbilu.com/other/relate/4J4n8flPe.html)解析中间件时，该属性表示请求时使用的 cookie 签名，签名的作用是防止 cookie 被篡改。

```
// Cookie: user=tobi.CP7AWaXDfAKIRfH49dQzKJx7sKzzSoPq7/AcBBRVwlI3
req.signedCookies.user
// => "tobi"
```

2.18 req.stale

表示请求是"stale"（旧请求）的，与 req.fresh 属性要反。

```
req.stale
// => true
```

2.19 req.subdomains

一个表示请求中的子域名的数组。

```
// Host: "tobi.ferrets.itbilu.com"
req.subdomains
// => ["ferrets", "tobi"]
```

2.20 req.xhr

一个表示客户是否使用 Ajax 请求的布尔值，如果 X-Requested-With 请求头为 XMLHttpRequest (http://itbilu.com/javascript/js/EkIzWrt7.html)时，则为 true 。

```
req.xhr
// => true
```

3. 方法

3.1 检测内容类型： req.accepts()

```
req.accepts(types)
```

检测是否是可访问的内容类型，基于 Accept HTTP请求头检测。如果匹配失败，则返回 false ，这时应用应该响应 406 （不可接受的请求类型）。

```
// Accept: text/html
req.accepts('html');
// => "html"

// Accept: text/*, application/json
req.accepts('html');
// => "html"
req.accepts('text/html');
// => "text/html"
req.accepts(['json', 'text']);
// => "json"
req.accepts('application/json');
// => "application/json"

// Accept: text/*, application/json
req.accepts('image/png');
req.accepts('png');
// => undefined

// Accept: text/*;q=.5, application/json
req.accepts(['html', 'json']);
// => "json"
```

3.2 字符集检测： req.acceptsCharsets()

```
req.acceptsCharsets(charset [, ...])
```

返回一个可接受的字符集。该方法会根据 Accept-Charset HTTP头检测，如果指定的字符集不可用，则返回 false 。

3.3 编码方式检测： req.acceptsEncodings()

```
req.acceptsEncodings(encoding [, ...])
```

返回一个可接受的编码方式。该方法会根据 Accept-Encoding HTTP头检测，如果指定的字符集不可用，则返回 false 。

3.4 语言检测： req.acceptsLanguages()

```
req.acceptsLanguages(lang [, ...])
```

返回一个可接受的语言。该方法会根据 Accept-Language HTTP头检测，如果指定的语言不可用，则返回 false 。

3.5 获取HTTP请求头值： req.get()

```
req.get(field)
```

返回指定HTTP请求头的字段值（不区分大小写），推荐使用 Referrer 替换 Referer 字段。

```
req.get('Content-Type');
// => "text/plain"

req.get('content-type');
// => "text/plain"

req.get('Something');
// => undefined
```

3.6 MIME类型检查： req.is()

```
req.is(type)
```

检查 Content-Type HTTP头中是否是指定类型的 type MIME类型。

```
// 当 Content-Type: text/html; charset=utf-8
req.is('html');
req.is('text/html');
req.is('text/*');
// => true

//  当 Content-Type 是 application/json
req.is('json');
req.is('application/json');
req.is('application/*');
// => true

req.is('html');
// => false
```

3.7 获取参数值： req.param()

```
req.param(name [, defaultValue])
```

获取 req.params 、 req.body 或 req.query 的参数值，其获取顺序为：

- req.params
- req.body
- req.query

```
// ?name=tobi
req.param('name')
// => "tobi"

// POST name=tobi
req.param('name')
// => "tobi"

// /user/tobi for /user/:name
req.param('name')
// => "tobi"
```

下一篇：Express.js中文文档－Response对象 (/nodejs/npm/Vkp32gJpg.html)

上一篇：Express.js中文文档－Application对象 (/nodejs/npm/VJ5TlyRnl.html)

文章分类

- 基础、核心、API (/nodejs/core)
- 包、应用

阅读排行

- Sequelize 中文API文档－1. 快速入门、Seq... (/nodejs/npm/VkYIaRPz-.html) (32104)
- Sequelize 中文API文档－2. Model 的定... (/nodejs/npm/V1PExztfb.html) (30933)
- 解决类似 /usr/lib64/libstdc++.so.... (/linux/management/NymXRUieg.html) (12705)
- Sequelize 中文API文档－3. 模型（表）之间的... (/nodejs/npm/41qaV3czb.html) (9944)
- Sequelize 中文API文档－4. 查询与原始查询 (/nodejs/npm/VJIR1CjMb.html) (9071)
- HTTP请求方法：GET、HEAD、POST、PUT、DE... (/other/relate/EkwKysXIl.html) (6269)
- Linux升级安装GCC (/linux/management/V1vdnt9ll.html) (4755)
- Redis设置认证密码 Redis使用认证密码登录 在Re... (/database/redis/Ey_r7mWR.html) (4137)
- MQTT协议－MQTT协议简介及协议原理 (/other/relate/4kHBsx_Pg.html) (4114)
- ［ES6］Promise对象Promise.all()方法... (/javascript/js/41KMSZ9a.html) (3821)