**Express 简介**

Express 是一个简洁而灵活的 node.js Web应用框架, 提供了一系列强大特性帮助你创建各种 Web 应用，和丰富的 HTTP 工具。

使用 Express 可以快速地搭建一个完整功能的网站。

Express 框架核心特性：

* 可以设置中间件来响应 HTTP 请求。
* 定义了路由表用于执行不同的 HTTP 请求动作。
* 可以通过向模板传递参数来动态渲染 HTML 页面。

**安装 Express**

安装 Express 并将其保存到依赖列表中：

$ cnpm install express --save

以上命令会将 Express 框架安装在当前目录的 **node\_modules** 目录中， **node\_modules** 目录下会自动创建 express 目录。以下几个重要的模块是需要与 express 框架一起安装的：

* **body-parser** - node.js 中间件，用于处理 JSON, Raw, Text 和 URL 编码的数据。
* **cookie-parser** - 这就是一个解析Cookie的工具。通过req.cookies可以取到传过来的cookie，并把它们转成对象。
* **multer** - node.js 中间件，用于处理 enctype="multipart/form-data"（设置表单的MIME编码）的表单数据。

$ cnpm install body-parser --save

$ cnpm install cookie-parser --save

$ cnpm install multer --save

安装完后，我们可以查看下 express 使用的版本号：

$ cnpm list express

/data/www/node

└── express@4.15.2 -> /Users/tianqixin/www/node/node\_modules/.4.15.2@express

## 请求和响应

Express 应用使用回调函数的参数： **request** 和 **response** 对象来处理请求和响应的数据。

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

// --

})

**request** 和 **response** 对象的具体介绍：

**Request 对象** - request 对象表示 HTTP 请求，包含了请求查询字符串，参数，内容，HTTP 头部等属性。常见属性有：

1. req.app：当callback为外部文件时，用req.app访问express的实例
2. req.baseUrl：获取路由当前安装的URL路径
3. req.body / req.cookies：获得「请求主体」/ Cookies
4. req.fresh / req.stale：判断请求是否还「新鲜」
5. req.hostname / req.ip：获取主机名和IP地址
6. req.originalUrl：获取原始请求URL
7. req.params：获取路由的parameters
8. req.path：获取请求路径
9. req.protocol：获取协议类型
10. req.query：获取URL的查询参数串
11. req.route：获取当前匹配的路由
12. req.subdomains：获取子域名
13. req.accepts()：检查可接受的请求的文档类型
14. req.acceptsCharsets / req.acceptsEncodings / req.acceptsLanguages：返回指定字符集的第一个可接受字符编码
15. req.get()：获取指定的HTTP请求头
16. req.is()：判断请求头Content-Type的MIME类型

**Response 对象** - response 对象表示 HTTP 响应，即在接收到请求时向客户端发送的 HTTP 响应数据。常见属性有：

1. res.app：同req.app一样
2. res.append()：追加指定HTTP头
3. res.set()在res.append()后将重置之前设置的头
4. res.cookie(name，value [，option])：设置Cookie
5. option: domain / expires / httpOnly / maxAge / path / secure / signed
6. res.clearCookie()：清除Cookie
7. res.download()：传送指定路径的文件
8. res.get()：返回指定的HTTP头
9. res.json()：传送JSON响应
10. res.jsonp()：传送JSONP响应
11. res.location()：只设置响应的Location HTTP头，不设置状态码或者close response
12. res.redirect()：设置响应的Location HTTP头，并且设置状态码302
13. res.render(view,[locals],callback)：渲染一个view，同时向callback传递渲染后的字符串，如果在渲染过程中有错误发生next(err)将会被自动调用。callback将会被传入一个可能发生的错误以及渲染后的页面，这样就不会自动输出了。
14. res.send()：传送HTTP响应
15. res.sendFile(path [，options] [，fn])：传送指定路径的文件 -会自动根据文件extension设定Content-Type
16. res.set()：设置HTTP头，传入object可以一次设置多个头
17. res.status()：设置HTTP状态码
18. res.type()：设置Content-Type的MIME类型

#### 重定向与不重定向

在使用的过程中，redirect()方法大多能重定向成功，而location()方法则不太确定，有时可以成功有时不能成功。这与我们的用法有关。

上面讲过，URL重定向是在浏览器端完成的，而URL重定向与HTTP状态码和Location头有关。浏览器首先会判断状态码，只有当状态码是：301或302时，才会根据Location头中的URL进行跳转。

所以，使用location()设置头信息，而不设置状态码或状态码不是301或302，并不会发生重定向：

res.location('http://itbilu.com');

res.sent(200);

而使用redirect()设置的状态码不是301或302也不会发生跳转：

res.redirect(200, 'http://itbilu.com');

## 路由

我们已经了解了 HTTP 请求的基本应用，而路由决定了由谁(指定脚本)去响应客户端请求。

在HTTP请求中，我们可以通过路由提取出请求的URL以及GET/POST参数。

接下来我们扩展 Hello World，添加一些功能来处理更多类型的 HTTP 请求。

创建 express\_demo2.js 文件，代码如下所示：

## express\_demo2.js 文件代码：

var express = require('express');

var app = express(); // 主页输出 "Hello World"

app.get('/', function (req, res) { console.log("主页 GET 请求");

res.send('Hello GET');

})

// POST 请求

app.post('/', function (req, res) {

console.log("主页 POST 请求");

res.send('Hello POST');

})

// /del\_user 页面响应

app.get('/del\_user', function (req, res) {

console.log("/del\_user 响应 DELETE 请求");

res.send('删除页面');

})

// /list\_user 页面 GET 请求

app.get('/list\_user', function (req, res) {

console.log("/list\_user GET 请求");

res.send('用户列表页面');

})

// 对页面 abcd, abxcd, ab123cd, 等响应 GET 请求

app.get('/ab\*cd', function(req, res) {

console.log("/ab\*cd GET 请求");

res.send('正则匹配');

})

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)

})

## 静态文件

Express 提供了内置的中间件**express.static**来设置静态文件如：图片， CSS, JavaScript 等。

你可以使用**express.static**中间件来设置静态文件路径。例如，如果你将图片， CSS, JavaScript 文件放在 public 目录下，你可以这么写：

app.use(express.static('public'));

在浏览器中访问 <http://127.0.0.1:8081/images/logo.png>.

## 比较 GET 与 POST

下面的表格比较了两种 HTTP 方法：GET 和 POST。

|  |  |  |
| --- | --- | --- |
|  | **GET** | **POST** |
| 后退按钮/刷新 | 无害 | 数据会被重新提交（浏览器应该告知用户数据会被重新提交）。 |
| 书签 | 可收藏为书签 | 不可收藏为书签 |
| 缓存 | 能被缓存 | 不能缓存 |
| 编码类型 | application/x-www-form-urlencoded | application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。 |
| 历史 | 参数保留在浏览器历史中。 | 参数不会保存在浏览器历史中。 |
| 对数据长度的限制 | 是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。 | 无限制。 |
| 对数据类型的限制 | 只允许 ASCII 字符。 | 没有限制。也允许二进制数据。 |
| 安全性 | 与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。  在发送密码或其他敏感信息时绝不要使用 GET ！ | POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。 |
| 可见性 | 数据在 URL 中对所有人都是可见的。 | 数据不会显示在 URL 中。 |

Express安装：

1. Npm install –g supervisor(安装监听代码监视器)
2. Npm install –g express(安装在全局目录)
3. 环境变量-系统变量-path加上express安装的位置D:/js\_learning
4. 重启cmd，express –version可以查看版本信息
5. Express –t ejs [project name]可以新建一个ejs框架
6. Cd到[project name]然后npm install