03Node框架Express

简介

基于 Node.js 平台，快速、开放、极简的 web 开发框架。 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

安装 Express 并将其保存到依赖列表中： cnpm install express --save

新建express实例

```
const epxress = require('express');
```

```
var app = express();

app.use('/', function (req, res) {
   res.send('Hello World');
})

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)

})
```

Application常用方法

use

app.use([path,] function [, function...]) 接收用户的get请求和post请求

```
app.use('/admin', function(req, res, next) {
  // GET 'http://www.example.com/admin/new'
  console.log(req.originalUrl); // '/admin/new'
  console.log(req.baseUrl); // '/admin'
  console.log(req.path); // '/new'
  next();
});
```

get

app.get(path, callback [, callback ...]) 只接收用户的get请求

```
app.get('/', function (req, res) {
  res.send('GET request to homepage');
});
```

post

app.post(path, callback [, callback ...]) 只接收用户的post请求 获取用户传来的post数据需要使用中间件解析

```
app.post('/', function (req, res) {
  res.send('POST request to homepage');
});
```

listen

app.listen(port, [hostname], [backlog], [callback]) 让服务监听某个端口

```
app.listen(3000);
```

Request

获取get数据

req.query

```
// 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"
```

获取post数据

获取post数据需要中间件的支持

```
var app = require('express')();
var bodyParser = require('body-parser');

app.use(bodyParser.json()); // for parsing application/json
app.use(bodyParser.urlencoded({ extended: true })); // for parsing application/x-
www-form-urlencoded

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

Response

服务器响应

end

结束响应过程。这个方法实际上来自Node核心http.ServerResponse的response.end方法 用于快速结束没有任何数据的响应。 如果需要使用数据进行响应，使用res.send/res.json方法。

send

发送HTTP响应。 该body参数可以是一个Buffer对象，一个String，一个对象或一个Array。 当参数是Array 或 Object时，调用的是res.json方法

json

发送JSON响应。

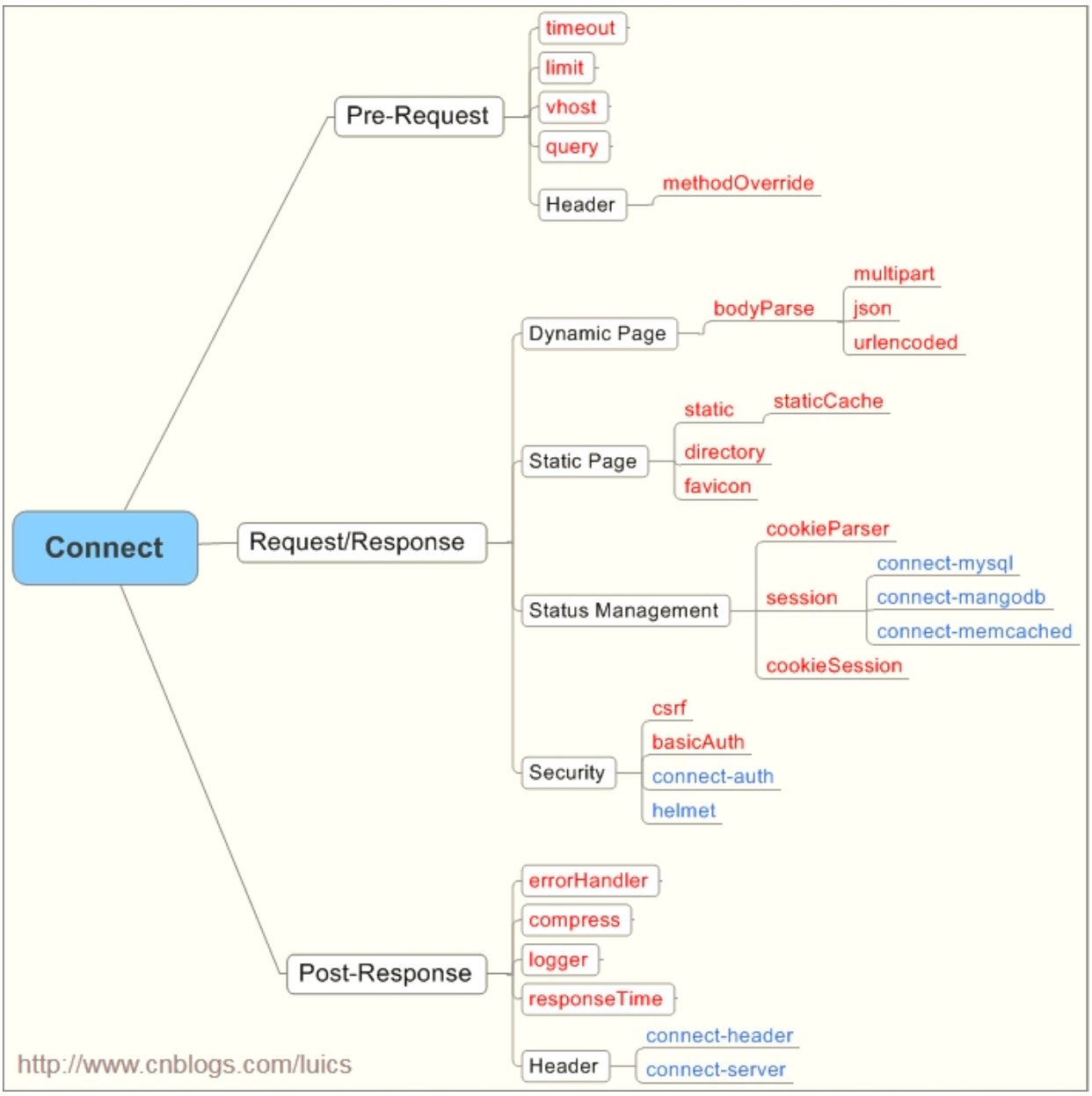
中间件

中间件是什么?

express是一个自身功能极简，完全是由**路由**和**中间件**构成的一个web开发框架，本质上说，一个express应用就是在调用各种中间件。 中间件(MiddleWare)可以理解为一个对用户请求进行过滤和预处理的东西，它一般不会直接对客户端进行响应，而是将处理之后的结果传递下去。.如果你需要这些中间件需要显示的添加。步骤如下： 1、 安装模块：npm install --save 2、 在应用中引入模块：require('module-name') 3、 按照文档的描述使用模块：app.use(...) 更多中间件请查阅官方文档: 中间件

二、中间件分类

我们可以将根据中间件在整个http处理流程的位置将Connect中间件分为三大类方便记忆。 1、 **pre-request** 通常用来该项request的原始数据 2、 **request**、**response**大部分中间件都在这里，功能各异。 3、 post-response全局异常处理，改写response数据等



三、使用中间件有什么好处?

由于我们知道在用原生node api进行`http`请求的处理时要求引入许多模块类似于`url`、`fs`等。并且在对`http`请求做处理时需要很多重复且不是逻辑上的操作，所以就会出现已经帮你封装好的处理操作，让你不再耗费时间在数据处理和异常处理上。这样能够大大减少代码量，使得逻辑更加清晰，具有便捷高效的扩展性。

四、了解原理-自己写中间件

最基本的中间件结构如下：

```
function myFunMiddleware(request, response, next) {
    // 对request和response作出相应操作
    // 操作完毕后返回next()即可转入下个中间件
    next();
}
```

接下来写两个中间件：

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

// 01
function middle01(req,res,next){
    req.name = 'luoqian'
    next();
}

//02
function middle02(){
    var accessNum = 0;
    return function(req,res,next){
        accessNum +=1;
        req.accessNum = accessNum
        next();
    }
}

app.use(middle01);
app.use(middle02());

app.get('/',function(req,res){
    res.send(req.name + '\n' + "网站访问人数" + req.accessNum);
})
app.listen(3000);
```

在这里写了两个中间件 第一个是为每一个`req`请求添加一个`name`属性 第二个它本身不是中间件实体，当执行它时，会return一个中间件函数，属于一个闭包，它的作用是每次有请求时，都会给`req`请求对象加入一个`accessNum`属性，这个属性每次都会 +1 。