一、了解node.js

1.Node.js是一个可以让 JavaScript 运行在服务器端的平台，是一个为实时Web（Real-time Web）应用开发而诞生的平台，它从诞生之初就充分考虑了在实时响应、超大规模数据要求下架构的可扩展性。这使得它摒弃了传统平台依靠多线程来实现高并发的设计思路，而采用了**单线程、异步式I/O、事件驱动**式的程序设计模型。这些特性不仅带来了巨大的性能提升，还减少了多线程程序设计的复杂性，进而提高了开发效率。

2.Node.js 有着强大而灵活的包管理器（node package manager，npm）.

3.Node.js 是一个让 JavaScript 运行在服务端的开发平台，它让 JavaScript 成为脚本语言世界的一等公民，在服务端堪与 PHP、Python、Perl、Ruby 平起平坐。

4. Node.js 可以作为服务器向用户提供服务，与 PHP、Python、Ruby on Rails 相比，它跳过了 Apache、Nginx 等 HTTP服务器，直接面向前端开发。Node.js 的许多设计理念与经典架构（如 LAMP）有着很大的不同，可提供强大的伸缩能力，以适应21世纪10年代以后规模越来越庞大的互联网环境。

5. Node.js 的 JavaScript 引擎是 V8，来自 Google Chrome 项目。V8 号称是目前世界上最快的 JavaScript 引擎，经历了数次引擎革命，它的 JIT（Just-in-time Compilation，即时编译）执行速度已经快到了接近本地代码的执行速度。Node.js 不运行在浏览器中，所以也就不存在 JavaScript 的浏览器兼容性问题，你可以放心地使用 JavaScript 语言的所有特性。

6. Node.js 最大的特点就是采用异步式 I/O 与事件驱动的架构设计.

7. CommonJS 试图定义一套普通应用程序使用的API，从而填补 JavaScript 标准库过于简单的不足. 。CommonJS 的终极目标是制定一个像 C++ 标准库一样的规范，使得基于 CommonJS API 的应用程序可以在不同的环境下运行.

ECMAscript是JS在浏览器中的标准；浏览器之外的标准由CommonJS制定，具体实现里nodejs是最出色的一个。

二、编译（window系统）

Node.js 在 Windows 下只能通过 Microsoft Visual Studio 编译，因此你需要首先安装 Visual Studio 或者免费的 Visual Studio Express。你还需要安装 Python 2（2.5以上的版本，但要小于3.0），可以在http://python.org/取得。安装完 Python 以后请确保在PATH环境变量中添加python.exe 所在的目录，如果没有则需要手动在“系统属性”中添加。

三、编程

1.运行 Node.js 程序

node script.js

node -e "console.log('hello')"；

2. REPL模式

REPL （Read-eval-print loop），即输入—求值—输出循环

在任何时候，连续按两次 Ctrl + C 即可推出Node.js 的 REPL 模式

3. 调试小技巧supervisor（英文含义：监督）

supervisor app.js 在开发 Node.js 实现的 HTTP 应用时会发现，无论你修改了代码的哪一部份，都必须终止Node.js 再重新运行才会奏效。这是因为Node.js 只有在第一次引用到某部份时才会去解析脚本文件，以后都会直接访问内存，避免重复载入

4.npm命令

npm [install/i] [package\_name]本地模式

npm [install/i] -g [package\_name]全局模式

使用全局模式安装的包不能直接通过 require 使用，但通过 npm link命令

可以打破这一限制。如，我们已经通过 npm install -g express 安装了 express，

这时在工程的目录下运行命令：

$ npm link express

./node\_modules/express -> /usr/local/lib/node\_modules/express

我们可以在 node\_modules 子目录中发现一个指向安装到全局的包的符号链接。通过这

种方法，我们就可以把全局包当本地包来使用了。

注a: npm link 命令不支持Windows。

5.包的发布

1)创建一个文件夹（一般是包名）

2)Npm init

3)Npm adduser(根据提示填写用户名、密码、邮箱，我已经有一个wudan\_pnm pnm001 [1872776189@qq.com](mailto:1872776189@qq.com))

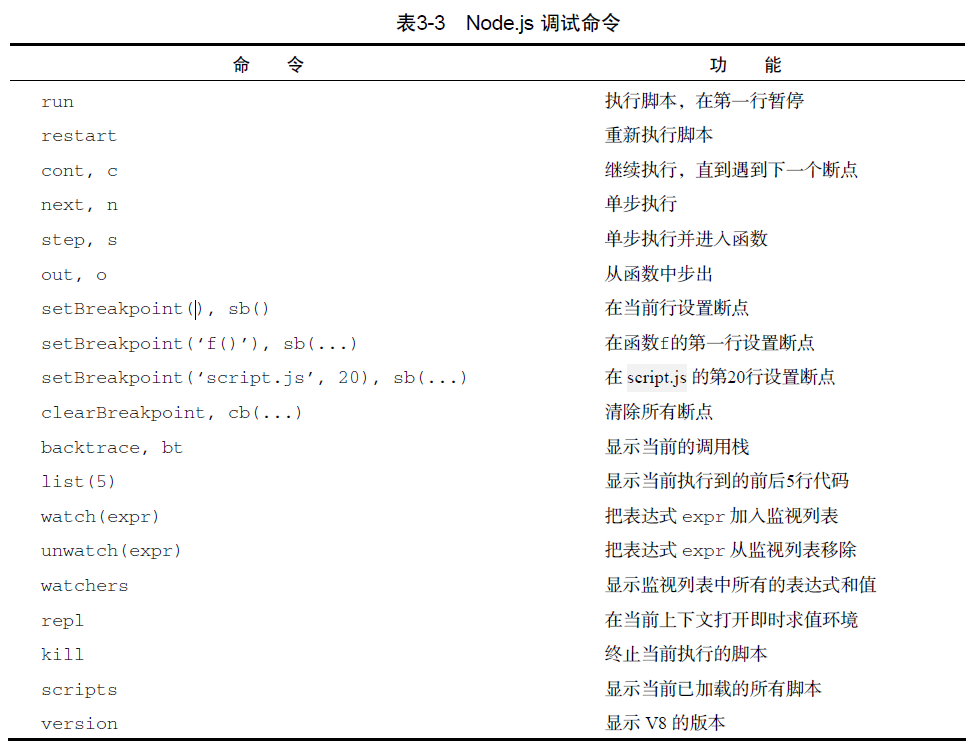
4)Npm publish

如果你的包将来有更新，只需要在 package.json 文件中修改 version 字段，然后重新使用 npm publish 命令就行了。如果你对已发布的包不满意（比如我们发布的这个毫无意义的包），可以使用 npm unpublish 命令来取消发布。

6.调试功能

在命令行下执行 node debug debug.js，将会启动调试工具，打开调试终端，下面是一些基本的命令进行调试

1)命令行调试



2）远程调试

V8 提供的调试功能是基于 TCP 协议的，因此 Node.js 可以轻松地实现远程调试。

node --debug[=port] script.js

node --debug-brk[=port] script.js

3）使用 Eclipse 调试 Node.js 程序

4）使用 node-inspector 调试 Node.js

四、核心基础

1．JavaScript 的面向对象特性是基于原型的，与常见的基于类的不同。

2.utils工具

util.inherits(constructor, superConstructor)是一个实现对象间原型继承的函数。

util.inspect(object,[showHidden],[depth],[colors])是一个将任意对象转换为字符串的方法，通常用于调试和错误输出。它至少接受一个参数 object，即要转换的对象。

四个类型测试工具：util.isArray()、util.isRegExp()、util.isDate()、util.isError()

3.事件发射器

events 模块只提供了一个对象： events.EventEmitter。EventEmitter 的核心就是事件发射与事件监听器功能的封装。

EventEmitter常用的API。

EventEmitter.on(event, listener) 为指定事件注册一个监听器，接受一个字

符串 event 和一个回调函数 listener。

EventEmitter.emit(event, [arg1], [arg2], [...]) 发射 event 事件，传

递若干可选参数到事件监听器的参数表。

EventEmitter.once(event, listener) 为指定事件注册一个单次监听器，即

监听器最多只会触发一次，触发后立刻解除该监听器。

EventEmitter.removeListener(event, listener) 移除指定事件的某个监听

器，listener 必须是该事件已经注册过的监听器。

EventEmitter.removeAllListeners([event]) 移除所有事件的所有监听器，

如果指定 event，则移除指定事件的所有监听器。

EventEmitter 定义了一个特殊的事件 error，它包含了“错误”的语义，我们在遇到

异常的时候通常会发射 error 事件。当 error 被发射时，EventEmitter 规定如果没有响应的监听器，Node.js 会把它当作异常，退出程序并打印调用栈。我们一般要为会发射 error事件的对象设置监听器，避免遇到错误后整个程序崩溃。

大多数时候我们不会直接使用 EventEmitter，而是在对象中继承它，原因有两点。首先，具有某个实体功能的对象实现事件符合语义，事件的监听和发射应该是一个对象的方法其次 JavaScript 的对象机制是基于原型的，支持部分多重继承，继承 EventEmitter 不会打乱对象原有的继承关系。

4. 文件系统 fs

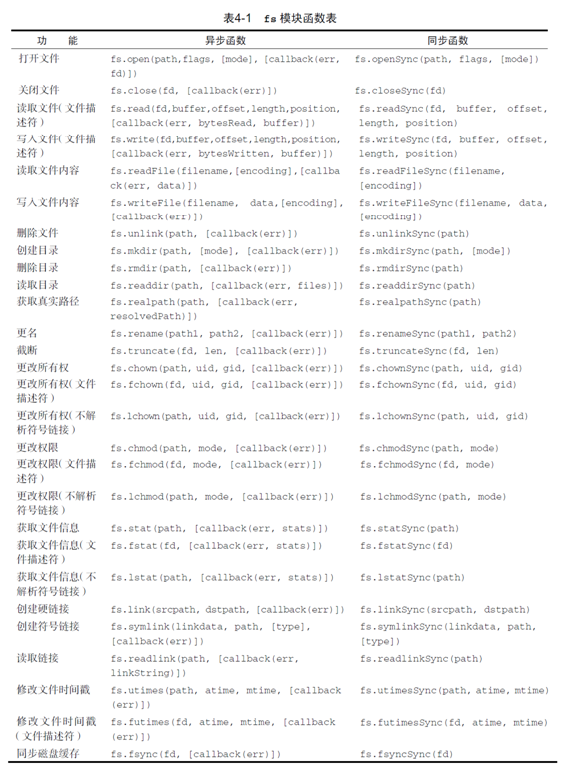
fs 模块是文件操作的封装，它提供了文件的读取、写入、更名、删除、遍历目录、链接等 POSIX 文件系统操作。与其他模块不同的是，fs 模块中所有的操作都提供了异步的和

同步的两个版本。

fs.readFile(filename,[encoding],[callback(err,data)])是最简单的读取文件的函数。（如果指定了 encoding，data 是一个解析后的字符串，否则 data 将会是以 Buffer 形式表示的二进制数据。）

fs.read的功能是从指定的文件描述符 fd 中读取数据并写入 buffer 指向的缓冲区对象.

表4-1列出了fs所有函数的定义和功能。



5.HTTP 服务器与客户端

（1）服务器端

http.Server 是 http 模块中基于事件的 HTTP 服务器对象

http.ServerRequest

http.ServerResponse有三个重要的成员函数，用于返回响应头、响应内容、结束请求。

response.writeHead(statusCode, [headers])：向请求的客户端发送响应头。

/\*response.writeHead(statusCode[, statusMessage][, headers])

这个方法的第一个参数，就是statusCode，也就是200、403之类的数字，剩下的参数是可选的。最后一个参数是headers，你可以在这里使用JSON对象表示法来写一些HTTP头部，比如：{“Content-Type”:”text/plain”,”Content-Length”:11}。第一个可选参数statusMessage用来指定一个状态描述消息，可以不填写。\*/

response.write(data, [encoding])：向请求的客户端发送响应内容。

response.end([data], [encoding])：结束响应，告知客户端所有发送已经完成。

**http.createServer([requestListener])**

**requestListener是一个方法，会与http.Server类的request事件关联起来。这样当客户端请求到达时，requestListener就会被调用。**

**requestListener有两个参数，函数原型如下：**

**function (request, response) { }**

**第一个参数request的类型是http.IncomingMessage，实现了Readable Stream接口。第二个参数的类型是http.ServerResponse，实现了Writeable Stream接口。**

var http = require('http');

http.createServer(function(req, res) {

res.writeHead(200, {'Content-Type': 'text/html'});

res.write('<h1>Node.js</h1>');

res.end('<p>Hello World</p>');

}).listen(3000);

console.log("HTTP server is listening at port 3000.");

显式的实现方法

var http = require('http');

var server = new http.Server();

server.on('request', function(req, res) {

res.writeHead(200, {'Content-Type': 'text/html'});

res.write('<h1>Node.js</h1>');

res.end('<p>Hello World</p>');

});

server.listen(3000);

console.log("HTTP server is listening at port 3000.");

（2）客户端

http 模块提供了两个函数 http.request 和 http.get，功能是作为客户端向 HTTP

服务器发起请求。

1）http.request(options, callback) 发起 HTTP 请求。不要忘了通过 req.end() 结束请求，否则服务器将不会收到信息。

2）http.get(options, callback) 它是 http.request 的简化版，唯一的区别在于http.get自动将请求方法设为了 GET 请求，同时不需要手动调用 req.end()。

五、使用node.js进行web开发

1．Express框架

它是目前最稳定、使用最广泛，而且 Node.js 官方推荐的唯一一个 Web 开发框架。

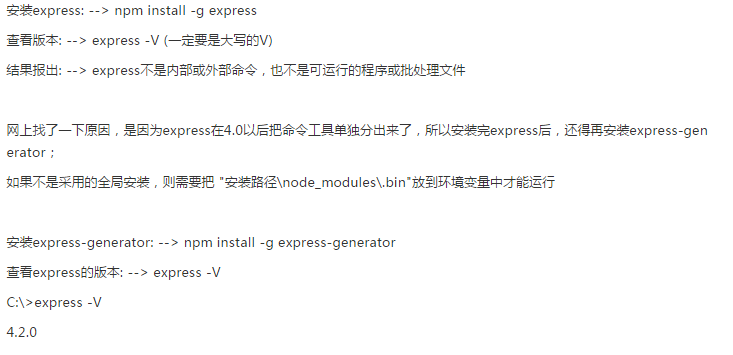
**下面用 Express 作为开发框架来开发一个网站（共三大步）：**

(1)安装express需要两步：

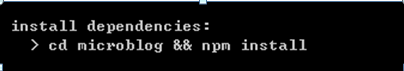
1）npm install -g express

2）npm install -g express-generator

原因如下：



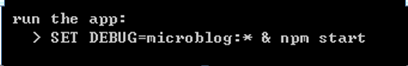
(2)建立工程express -e ejs microblog（express -t ejs microblog这种写法是老写法，现在已不适用，创建出来的view文件是jade的，因此不要用了。）



然后根据提示运行即可。

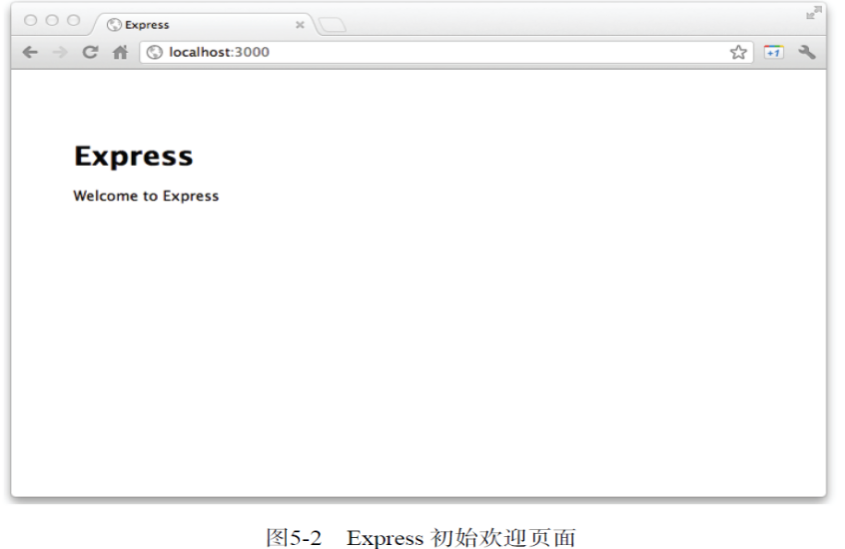
无参数的 npm install 的功能就是检查当前目录下的 package.json，并自动安装所有指定的依赖。

（3）启动服务

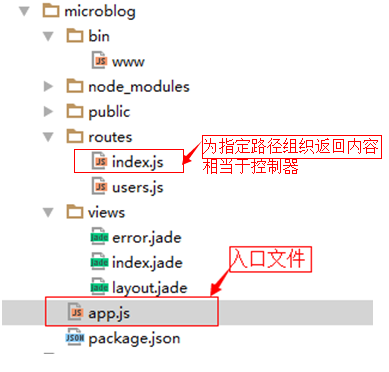


注：npm start 或者supervisor start( 通过查看package.js知道start为./bin/www，supervisor更适合调试，不用每次都关闭重启服务)

成功界面如图：

。

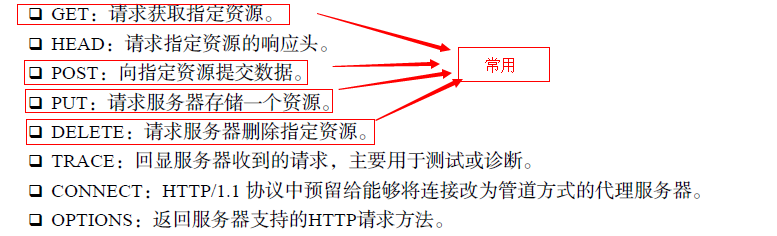
**工程结构**

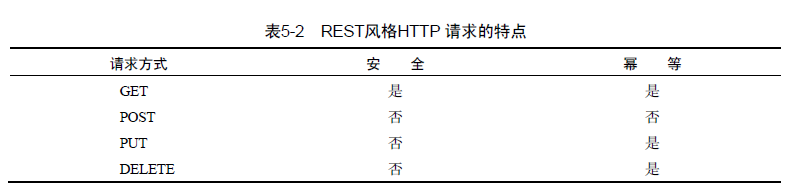


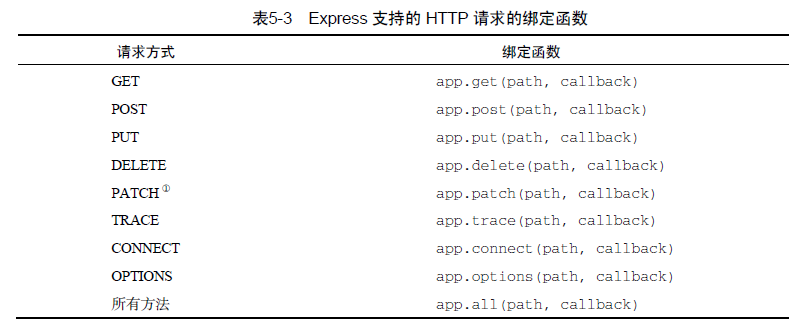
* bin目录，里面就一个www文件，其实还是个js脚本文件，使用npm start启动网站时会调用www。你可以看控制台输出来确认。
* public目录，里面放了一些静态文件，默认生成的项目，只有stylesheets下有一个style.css文件。javascripts子目录可以放js文件，images子目录放图片。
* routes目录内放的是路由模块，你可以在这里配置你网站的路由。
* views目录放的HTML模板文件。
* node\_modules目录是根据package.json生成的，package.json由express生成
* app.js，应用入口
* Package.json 对包的描述，包括包名、描述、版本、依赖、启动文件等。

2.路由选择——rest（表征状态转移）风格，一种基于HTTP协议的网络应用的接口风格（一种软件架构风格）。它并不是一个标准，而是通过表征（Representional ）来描述传输状态的一种原则。其宗旨是从资源的角度来观察整个网络，分布在各处的资源由URI确定，而客户端的应用通过URI来获取资源的表征。获得这些表征致使这些应用程序转变了其状态。随着不断获取资源的表征，客户端应用不断地在转变着其状态。

（1）HTTP定义了以下8中标准方法：







（2）控制权转移next()

Express支持统一路径绑定多个路由响应函数，但当访问任何两条同样的规则匹配到的路径时，会优先匹配先定义的路由规则，后面的相同规则被屏蔽。回调函数第三个参数即next会将控制权转移给后面的规则。举一个经典小例子。



（3）模板引擎

是一个从页面模板根据一定的规则生成HTML的工具。

其功能是将页面模板和要显示的数据结合起来生成HTML页面。

它既可以运行在服务器端又可以运行在客户端，大多数时候它都在服务器端直接被解析为HTML，解析完成后在传输给客户端，因此客户端甚至无法判断页面是否是模板引擎生成的。目前主流由服务器运行模板引擎。

res.render(‘index’,{title:’Express’});//功能是调用模板引擎，并将其产生的页面直接返回给客户端。第一个参数是模板的名称，即views目录下的模板文件名，不包含文件的扩展名；第二个参数是传递给模板的数据，用于模板翻译。

6．node.js进阶话题

1. 模块分为两大类：

核心模块：标准API中提供的模块，如fs、http、net、vm等

文件模块：存储为单独的文件（活文件夹）的模块，可能是JS代码、JSON或编译好的ｃ／ｃ＋＋代码。

1. 文件模块的加载有两种方式：

１）按路径加载模块（绝对路径、相对路径）

２）查找 node\_modules 文件夹。npm获取的包通常就是以这种方式加载。

1. 加载顺序

1) 如果some\_module 是一个核心模块，直接加载，结束。

2) 如果some\_module以“ / ”、“ ./ ”或“ ../ ”开头，按路径加载 some\_module，结束。

3) 假设当前目录为 current\_dir，按路径加载 current\_dir/node\_modules/some\_module。

 如果加载成功，结束。

 如果加载失败，令current\_dir为其父目录。

 重复这一过程，直到遇到根目录，抛出异常，结束。

Node.js核心：全局对象（变量）、util工具、事件、文件操作、HTTP服务

JS前台与后台的不同

1. 弱类型语言（变量），变量可以不声明，直接打点使用，即隐式定义。
2. JavaScript 的面向对象特性是基于原型的，与常见的基于类的不同。

**项目运行步骤：**

1）启动redis D:\redis64-latest>redis-server.exe redis.windows.conf

2）启动mongo D:\work\mongodb\bin>mongod.exe --dbpath=D:\work\jsgen\data

3）编译运行 F:\wudan\protect\jsgen>node app.js install

gulp

F:\wudan\protect\jsgen>node app.js recache

**主要看一下英文版part2中间件的概念4——7（6.7）**

**Nodejs是什么、安装部署、基本知识（基本概念【包、模块】、核心模块【全局对象、util工具、文件系统、HTTP服务】）、Express创建的网站架构。**

https://exmail.qq.com/cgi-bin/viewfile?type=logo&domain=jdwtgroup.com

**Week 2**

Mongo命令行：

Show dbs

Show collections

db.usercollection.find()

1.流控制

串行流控制

并行流控制

1. connect是Node的一个中间件框架

参考网址：https://cnodejs.org/topic/4fb79b0e06f43b56112b292c#52d657a6e20b7c8214befb38

**中间件加载**

connect的最大灵活性就在于它的中间件机制。connect的中间件加载代码定义在lib/proto.js中的app.use方法。use方法的定义如下：

app.use(route, fn)

route是中间件所使用的请求url的pattern，默认为’/’。connect会按照加载顺序，逐一执行pattern与请求url匹配的中间件处理函数。第二个参数fn即中间件处理函数，有两种定义形式。

第一种形式为：

function(req, resp, next)

第二种形式为：

function(err, req, resp, next)

第一种是正常的处理函数，第二种是异常处理函数。 req, resp为http模块的request和response对象。next是触发后续流程的回调函数，带一个err参数。通过传递给next传递一个err参数，告诉框架当前中间件处理出现异常。如果err为空，则会按顺序执行后面正常处理函数，忽略异常处理函数;相反，如果err非空，则会按顺序执行后续的异常处理函数，而忽略正常处理函数。

在connect中，请求的处理流程是一个异步的过程，fn函数的返回并不代表处理流程的结束，所以在这里需要用next回调的形式通知框架执行后续的流程。但如果某一个中间件函数认为请求的流程到它那里已经处理完毕，无需再执行后面的流程，则可以直接返回，而不用再调用next回调函数了（包括正常和异常处理函数）。如果请求遍历完中间件列表后仍在调用next函数，connect则会认为这个请求没有中间件认领。这时，如果next的err参数非空，则会给页面返回500错误，表示server出现了内部错误;如果err为空，则返回404错误，即访问的资源不存在。

依照中间件函数的定义，我们也可以编写自己的中间件函数，然后通过use方法加入到connect的处理流程中。

在这里，我们也可以看出来，各个middleware之间其实并没有直接的依赖。request和response就成为它们在connect中传递信息的唯一桥梁。前面的中间件处理完毕后，把结果附到request或response之上，后面的中间件便可以从中获取到这些数据。所以，中间件的加载顺序在connect中就显得格外重要，必须将被依赖的中间件放在依赖它的模块之前。比如说，解析cookie的中间件应该放在处理session的中间件之前，因为一般session id是通过cookie来传递的。

**3.Business-monitor项目运行步骤：**

1）安装

Npm install sails -g 安装sails框架

Npm install 安装后台依赖

Bower install 安装前台依赖

安装可视化数据库软件Navicat for mysql，根据config——xml——connection.js文件配置数据库并连接。

Grunt build

1. Sails lift 项目启动
2. **Grunt**

1)Grunt是什么

Grunt 是一个基于任务的JavaScript工程命令行构建工具。换言之，它就是一种自动化任务处理工具，一个工具框架，有很多插件扩展它的功能。

Grunt 基于 Node.js ，用 JS 开发，这样就可以借助 Node.js 实现跨系统跨平台的桌面端的操作，例如文件操作等等。此外，Grunt 以及它的插件们，都作为一个 包 ，可以用 NPM 安装进行管理。所以 NPM 生成的 package.json 项目文件，里面可以记录当前项目中用到的 Grunt 插件，而 Grunt 会调用 Gruntfile.js 这个文件，解析里面的任（task）并执行相应操作。

2）Grunt命令行（CLI）运行机制：

每次运行grunt 时，他就利用node提供的require()系统查找本地安装的Grunt。正是由于这一机制，你可以在项目的任意子目录中运行grunt 。如果找到一份本地安装的 Grunt，CLI就将其加载，并传递Gruntfile中的配置信息，然后执行你所指定的任务。

3)简单应用实例

（1）建立过程：

➀新建一个项目/文件夹，进入到项目

Npm i grunt --save-dev

npm install -g grunt-cli

➁在你的项目中添加两份文件：package.json 和 Gruntfile。

package.json文件通过npm init初始化包获得；

Gruntfile: 手动创建获得，然后里面编写代码，由四部分组成，见下。

Gruntfile是一个文件，此文件被命名为 Gruntfile.js  或  Gruntfile.coffee，用来配置或定义任务（task）并加载Grunt插件的。放在你的项目根目录中，和package.json文件在同一目录层级。

Gruntfile由以下几部分构成：

"wrapper" 函数：Grunt代码必须放在此函数内；

项目与任务配置；

加载grunt插件和任务；

自定义任务。

例：

➂安装对应的插件 npm install [插件名] --save-dev，如果项目还有其他依赖的话，还要执行npm install。

④执行grunt命令。

1. **5.查询的知识点小结**

**1）git submodule update --init --recursive**

下载的工程带有submodule

当使用git clone下来的工程中带有submodule时，初始的时候，submodule的内容并不会自动下载下来的，此时，只需执行如下命令：

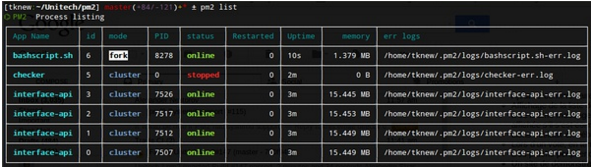
git submodule update --init --recursive

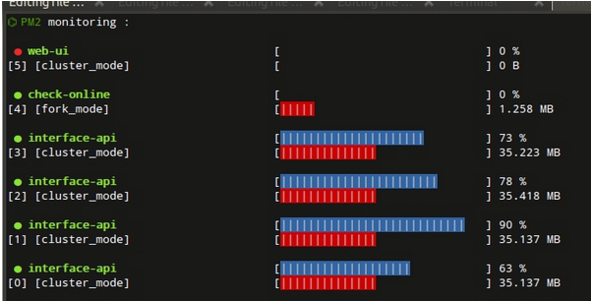
即可将子模块内容下载下来后工程才不会缺少相应的文件。

1. **git 命令 (gnu interactive tools)**功能说明：文字模式下的文件管理员。  
   语 法：git 命令  
   补充说明：git命令是用来[管理文件](https://www.baidu.com/s?wd=%E7%AE%A1%E7%90%86%E6%96%87%E4%BB%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAcsuAfkmHTzPj0YuHbY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjTzn10zPWbznWn4njD4rHnz" \t "http://zhidao.baidu.com/_blank)的程序，它十分类似DOS下的Norton Commander，具有互动式操作界面。它的操作方法和Norton Commander几乎一样，略诉如下：  
    [F1](https://www.baidu.com/s?wd=F1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAcsuAfkmHTzPj0YuHbY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjTzn10zPWbznWn4njD4rHnz" \t "http://zhidao.baidu.com/_blank) ：执行info指令，查询指令相关信息，会要求您输入欲查询的名称。  
    F2 ：执行cat指令，列出文件内容。  
    F3 ：执行gitview指令，观看文件内容。  
    [F4](https://www.baidu.com/s?wd=F4&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAcsuAfkmHTzPj0YuHbY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjTzn10zPWbznWn4njD4rHnz" \t "http://zhidao.baidu.com/_blank) ：执行vi指令，编辑文件内容。  
    F5 ：执行cp指令，复制文件或目录，会要求您输入目标文件或目录。  
    F6 ：执行mv指令，移动文件或目录，或是更改其名称，会要求您输入目标文件或目录。  
    F7 ：执行mkdir指令，建立目录。  
    F8 ：执行rm指令，删除文件或目录。  
    F9 ：执行make指令，批处理执行指令或编译程序时，会要求您输入相关命令。  
    [F1](https://www.baidu.com/s?wd=F1&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPAcsuAfkmHTzPj0YuHbY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjTzn10zPWbznWn4njD4rHnz" \t "http://zhidao.baidu.com/_blank)0 ：离开git文件管理员。
2. **PM2介绍**

pm2 是一个带有负载均衡功能的Node应用的进程管理器。当你要把你的独立代码利用全部的服务器上的所有CPU，并保证进程永远都活着，0秒的重载， PM2是完美的。

主要特性：  
  
内建负载均衡（使用Node cluster 集群模块）  
后台运行  
0秒停机重载，我理解大概意思是维护升级的时候不需要停机.  
具有Ubuntu和CentOS 的启动脚本  
停止不稳定的进程（避免无限循环）  
控制台检测  
提供 HTTP API  
远程控制和实时的接口API ( Nodejs 模块,允许和PM2进程管理器交互 )  
  
测试过Nodejs v0.11 v0.10 v0.8版本，兼容CoffeeScript,基于Linux 和MacOS.  
  
安装  
npm install -g pm2  
  
用法  
$ npm install pm2 -g # 命令行安装 pm2   
$ pm2 start app.js -i 4 #后台运行pm2，启动4个app.js   
                                # 也可以把'max' 参数传递给 start  
                                # 正确的进程数目依赖于Cpu的核心数目  
$ pm2 start app.js --name my-api # 命名进程  
$ pm2 list # 显示所有进程状态  
$ pm2 monit # 监视所有进程  
$ pm2 logs # 显示所有进程日志  
$ pm2 stop all # 停止所有进程  
$ pm2 restart all # 重启所有进程  
$ pm2 reload all # 0秒停机重载进程 (用于 NETWORKED 进程)  
$ pm2 stop 0 # 停止指定的进程  
$ pm2 restart 0 # 重启指定的进程  
$ pm2 startup # 产生 init 脚本 保持进程活着  
$ pm2 web # 运行健壮的 computer API endpoint ([http://localhost:9615](http://localhost:9615/" \t "http://www.douban.com/note/314200231/_blank))  
$ pm2 delete 0 # 杀死指定的进程  
$ pm2 delete all # 杀死全部进程  
  
运行进程的不同方式：  
$ pm2 start app.js -i max # 根据有效CPU数目启动最大进程数目  
$ pm2 start app.js -i 3 # 启动3个进程  
$ pm2 start app.js -x #用fork模式启动 app.js 而不是使用 cluster  
$ pm2 start app.js -x -- -a 23 # 用fork模式启动 app.js 并且传递参数 (-a 23)  
$ pm2 start app.js --name serverone # 启动一个进程并把它命名为 serverone  
$ pm2 stop serverone # 停止 serverone 进程  
$ pm2 start app.json # 启动进程, 在 app.json里设置选项  
$ pm2 start app.js -i max -- -a 23 #在--之后给 app.js 传递参数  
$ pm2 start app.js -i max -e err.log -o out.log # 启动 并 生成一个配置文件  
你也可以执行用其他语言编写的app ( fork 模式):  
$ pm2 start my-bash-script.sh -x --interpreter bash  
$ pm2 start my-python-script.py -x --interpreter python  
  
0秒停机重载:  
这项功能允许你重新载入代码而不用失去请求连接。  
注意：  
仅能用于web应用  
运行于Node 0.11.x版本  
运行于 cluster 模式（默认模式）  
$ pm2 reload all  
  
CoffeeScript:  
$ pm2 start my\_app.coffee #这就是全部  
  
PM2准备好为产品级服务了吗？  
只需在你的服务器上测试  
$ git clone [https://github.com/Unitech/pm2.git](https://github.com/Unitech/pm2.git" \t "http://www.douban.com/note/314200231/_blank)  
$ cd pm2  
$ npm install # 或者 npm install --dev ，如果devDependencies 没有安装  
$ npm test  
  
pm2 list  
列出由pm2管理的所有进程信息，还会显示一个进程会被启动多少次，因为没处理的异常。

  
pm2 monit  
监视每个node进程的CPU和内存的使用情况。



钱包地址：jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ

账户密匙：sn37nYrQ6KPJvTFmaBYokS3FjXUWd

信任地址：jBciDE8Q3uJjf111VeiUNM775AMKHEbBLS

钱包地址2：jGRqbsiAf2vgdbet5z31hNLEt8hdBjfPUk

address": "j4EHLd4Cxvmcx87v7DU2ZqDbRoyoRANbT3",

"secret": "snPR12gU98RQ7EpfKjHbgCZvBkAK1"

wudanjs2

address:jsZhbMn9dDrdfQQJ86qYv48Hu3rjEX4THo

Secret：snuGFniiVHUAg5cWK27Dr4r3ZR6ZY

完成查询、支付的运行过程。

Lodash:一个具有一致接口、[模块化](https://www.npmjs.com/browse/keyword/lodash-modularized)、高性能等[特性](http://lodashjs.com/" \l "features)的 JavaScript 工具库。

Respond.js 是一个快速、轻量的 polyfill，用于为 IE6-8 以及其它不支持 CSS3 Media Queries 的浏览器提供媒体查询的 min-width 和 max-width 特性，实现响应式网页设计（Responsive Web Design）。

Then():递延，能用简单优美的方式将任何同步或异步回调函数转换成then()链式调用！

Promise是CommonJS的规范之一，拥有resolve、reject、done、fail、then等方法，能够帮助我们控制代码的流程，避免函数的多层嵌套。

Bluebird的用法

Lodash的用法

2.支付

*➀getPathFind*(request, response, next) 查询支付路径V

http://localhost:3000/v1/accounts/jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ/payments/paths/jGRqbsiAf2vgdbet5z31hNLEt8hdBjfPUk/1+AAA+jGRqbsiAf2vgdbet5z31hNLEt8hdBjfPUk

*➁getAccountPayments*(request, response, next)获得支付历史V

localhost:3000/v1/accounts/jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ/payments/

*➂getPayment*(request, response, next)获得支付信息V

http://localhost:3000/v1/accounts/jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ/payments/4B05C712201739C2E20A3344BDD56889A09119B6B39D2324DB9F1B8FB88C5B05

@*submitPayment*(request, response, next)支付请求

http://localhost:3000/v1/accounts/jB7rxgh43ncbTX4WeMoeadiGMfmfqY2xLZ/payments?validated=true

1. 挂单

*getOrder*(request, response, next) 获取挂单信息

*getOrderBook*(request, response, next)获得货币对的挂单列表

*cancelOrder*(request, response, next) 取消挂单

*placeOrder*(request, response, next) 提交挂单

getOrders(request, response, next)获取用户挂单

Word accumulation

Concurrency并发

Integration整合

Extensively广泛