框架设计

**Express框架概述**

1. **什么是Express**

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。

1.什么是路由？

路由是指接收用户请求，处理用户数据，返回结果给用户的一套程序，可以理解为：生成动态网页的程序；

后端路由的核心：URL

2.express的路由：

express对象自带一个Router类，可以实例化出路由对象，可以在该对象上挂载非常多的路由节点；

3.路由的写法：

router.请求方式（‘请求地址’，function(req,res){

res,send('数据');

});

4,创建一个路由模块：

//需求：创建一个VIP路由模块，接收VIP目录下的所有请求，响应数据

1）创建一个VIP路由模块，

2）编写路由模块的代码

a.引入express模块

b.实例化路由对象

c.编写路由线路，挂载到路由线路上

d.暴露对象

3）将编写好的路由模块引入到主模块中，由主模块分配对应的请求到该模块去处理；

var vip = require(引入vip路由模块）；

app.use('/vip',vip);//分配VIP目录下的请求给VIP模块去处理;

4.路由的区别：

大路由（总路由）：app.js 负责接收所有请求，对请求进行分配。

Express 是一个基于 Node.js 平台的极简、灵活的 web 应用开发框架，它提供一系列强大的特性，帮助你创建各种 Web 和移动设备应用。

Express 是一个简洁而灵活的 node.js Web应用框架, 提供一系列强大特性帮助你创建各种Web应用。Express 不对 node.js 已有的特性进行二次抽象，我们只是在它之上扩展了Web应用所需的功能。丰富的HTTP工具以及来自Connect框架的中间件随取随用，创建强健、友好的API变得快速又简单。

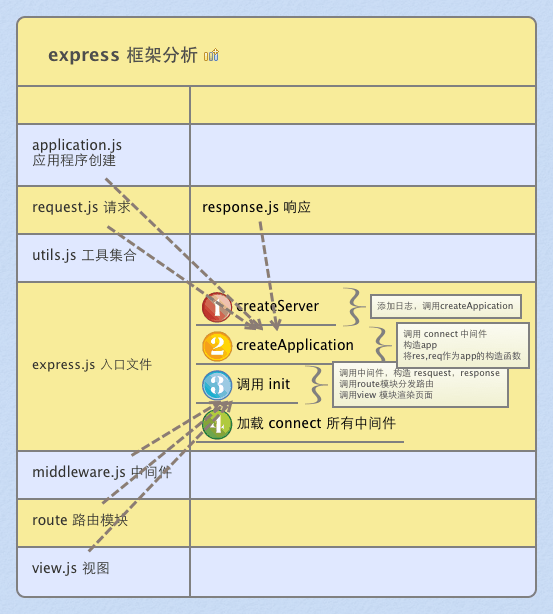
1. **Express的优点**

1、易上手：nodejs最初就是为了开发高性能web服务器而被设计出来的，然而相对底层的API会让不少新手望而却步。express对web开发相关的模块进行了适度的封装，屏蔽了大量复杂繁琐的技术细节，让开发者只需要专注于业务逻辑的开发，极大的降低了入门和学习的成本。

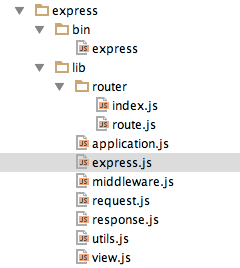
2、高性能：Express仅在web应用相关的nodejs模块上进行了适度的封装和扩展，较大程度避免了过度封装导致的性能损耗。

3、扩展性强：基于中间件的开发模式，使得express应用的扩展、模块拆分非常简单，既灵活，扩展性又强。

**3. Express 框架调用关系**



**4.Express 代码结构**



* bin/express 是在命令行下的生成express 框架目录文件用的
* lib/express 是框架的入口文件
* lib/router 是路由模块，主要是进行路由分发，比对，执行callback
* lib/middleware 是中间件模块，主要是对response，request进行改写
* lib/request 是请求
* lib/response 是响应
* lib/utils 是工具集函数，是对connect模块的一个补充，比如地址处理正则
* lib/view 是处理视图

1. **Express中间件**

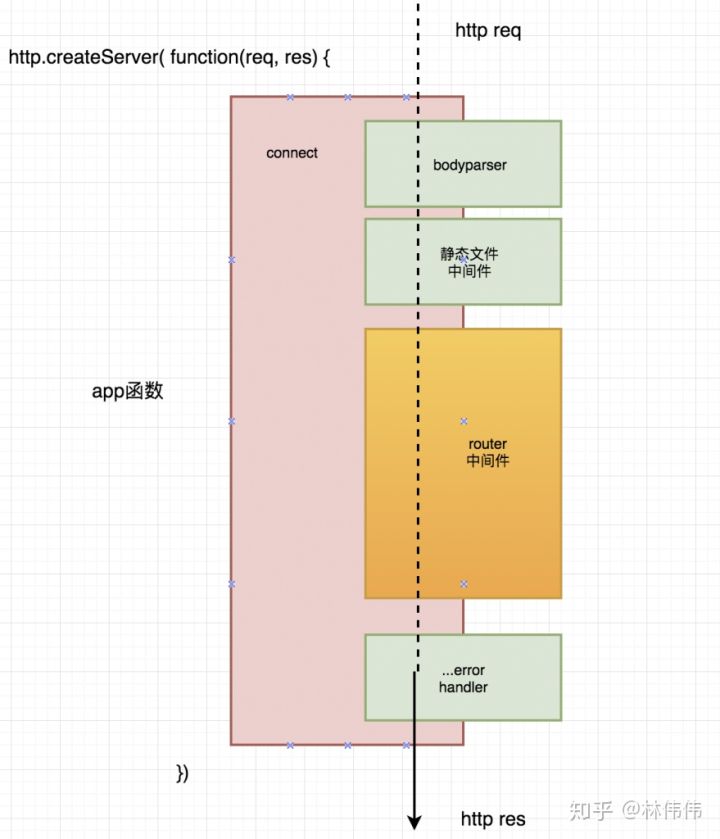
1.可以将处理接口数据的逻辑提出来，让组件更纯粹；  
 2.方便处理异步逻辑；  
 3.不同的要求使用不同的中间件处理，方便解耦；  
 4.设置请求；  
 5.路由分级处理；  
 6.前后端分离职责更清晰；  
 7.前端开发自己处理请求响应数据，不依赖于后台；

1. **express框架核心原理**

6.1 express构造的是Http.createServer的回调函数

实际上express是一个函数，运行后可以构造出上面代码中http.createServer的回调函数，express做的一切文章都是在这个回调函数上。来看下express3.x的源码express.js

connect.js的具体内容先不关心，后面会重点介绍。可以看出connect是一个函数，运行返回一个app，app是一个形如function(req, res , next){ ... } 的函数。express的createApplication返回即是此app，用于http.createServer的回调。并在这个函数上混入的许多能力，如req、res的处理、模板引擎、静态文件服务、router的能力。

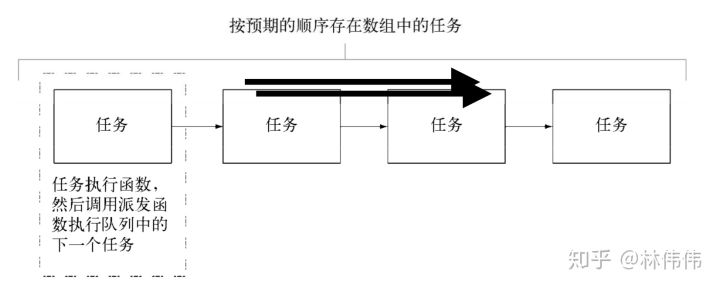


## 6.2 中间件框架

app.use中间件时，只是把它放入一个数组中。当http请求时，app会从数组中逐个取出，进行匹配过滤，逐个运行。遍历完成后，运行finalhandler，结束一个http请求。可以从http请求的角度思考，一次请求它经历经历了多少东西。express的这个中间件架构就是负责管理与调用这些注册的中间件。中间件顺序执行，通过next来继续下一个，一旦没有继续next，则流程结束。

## 6.3 异步串行流程控制

为了用串行化流程控制让几个异步任务按顺序执行，需要先把这些任务按预期的执行顺序放 到一个数组中。如图，所示，这个数组将起到队列的作用:完成一个任务后按顺序从数组中取 出下一个

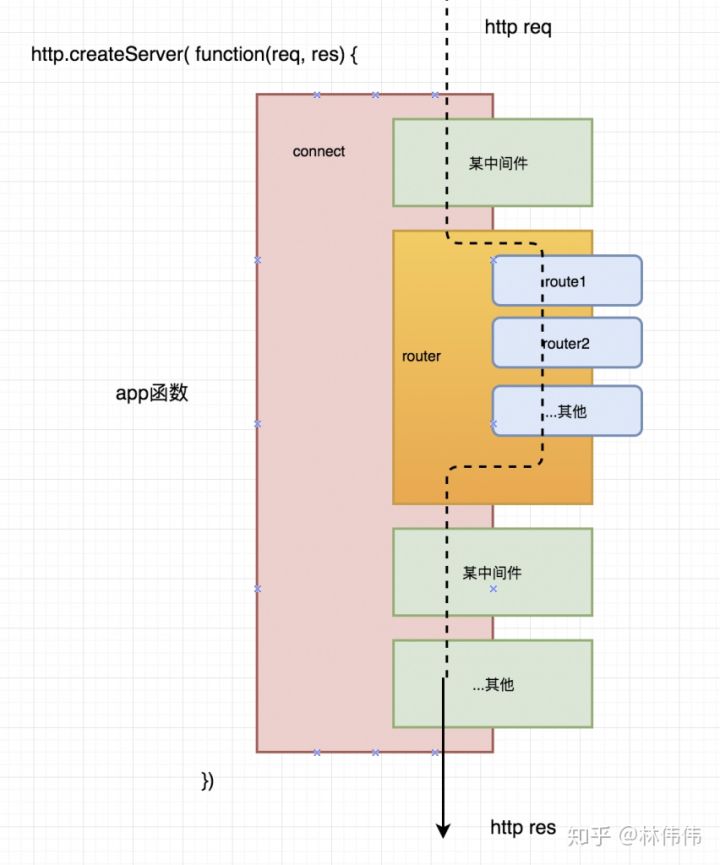


数组中的每个任务都是一个函数。任务完成后应该调用一个处理器函数，告诉它错误状态和 结果。如果有错误，处理器函数会终止执行;如果没有错误，处理器就从队列中取出下一个任务 执行它异步串行控制方案除了上面的这种以外，还可以用es6的promise的then链、async/await、yeild、社区工具等；

可以看到代码确实谈不上高级 ，串行导致的性能谈不上优秀，但是得益于此它足够简单易用。到此可以发现express的中间件架构就是一个中间件的的管理与数组遍历运行，这个方案就让社区形形色色各种各样的中间件很好的添加express能力，这点很简单也很重要，因为后续的路由、静态文件服务、代理等都是中间件，都在这个框架内运行。

## 6.4 Router

router跟connect非常类似，理解了connect，router就很清晰了



usedRouter是个开关，未开启则不加入router中间件，因为应用理论上也是可能不用到router的。当app[method] 如app.get('/user', fn)调用后，则触发this.use(this.router) 使用router中间件，同时把usedRouter设置为true。之后往router对象中加入fn回调函数。

实际上router还有细分，某个router还是可以继续做类似的串行流程控制；与中间件相同，每个router一旦停止了next，流程就结束了。

request经过router可以请求一个数据，或者一个网页；网页的话是怎么返回的呢，接下来看下view的render；

## 6.5 视图-模板引擎

## 模板引擎是根据对模板结合data进行运行处理，生产real html；这跟React、Vue、模板引擎是类似的。模板引擎不是express 实现的，实际上express仅仅只是做了调用；这里有个通用的支持各种模板引擎的模块[consolidate.js](https://github.com/tj/consolidate.js" \t "https://zhuanlan.zhihu.com/p/_blank)。

## 为了性能考虑还做了cache；关于模板引擎，实际上很简单，读者可以自定一个模板引擎规则。

## 6.6静态文件服务

## 静态文件服务也是一个中间件，express做的事情也仅仅是引用。require一个serve-static，内置在app函数上。

app.**static** **=** **function** (dir) {

**this**.use(serveStatic(process.cwd() **+** '/' **+** dir), {});};