为什么你应该抛弃Express的视图渲染引擎

分类 大话编程 (/home/大话编程) **关键字** Node.JS (/bbs/Node.JS) **发布** pockry (/userinfo/pockry) 2014-11-06

翻译自StrongLoop Blog (http://strongloop.com/strongblog/bypassing-express-view-rendering-for-speed-and-modularity/), 作者：Patrick Steele-Idem。你也可以到我的博客 (http://idlelife.org/archives/810)看本文。

Nodejs Express框架的一个被人们广为使用的特性是它的渲染引擎。Express视图渲染引擎允许Controller提供一个视图名称和视图模型对象给Express，然后返回由HTTP响应流输出的一些字节。基于为eBay的Nodejs技术栈提供支持所获得的经验，我们发现了这个方法的缺点并决定彻底的弃用它。我们这么做了之后，能明显看到页面加载速度的提升、更好的模块性以及开发者生产力的提高。本文将解释为什么你不应该使用Express视图渲染引擎，并提供一个推荐的替代方案。

Express视图渲染引擎

在解释Express视图渲染引擎的缺点之前，让我们先来快速的过一遍它的用法。首先你必须使用类似下面的代码来配置你的Express app：

```
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'jade');
```

第一行代码告诉Express在views目录下搜索所有模板，第二行则在这些模板上应用jade模板引擎。

配置好之后，你能在Controller中使用`res.render(viewName, viewModel)`方法来渲染视图，代码如下：

```
app.get('/', function (req, res) {
  res.render('index', { title: 'Hey', message: 'Hello there!'});
})
```

这里面的机制是，Express将视图名称对应到模板文件的路径，使用关联的模板引擎来渲染模板，将结果输出到response，最后结束响应。

不好的地方

这个方法看起来如此简单，你可能从来没想过它会出错。但实际上它还是有不少问题，当出问题时会影响web程序性能和可维护性，下面我们讲到这些。

破坏模块性

Express强制性的要求设置一个views目录，所有模板都堆放在这里。本来你可以将模板放到Controller或使用这些模板的UI组件附近，现在你不得不将它们放到一个单独的顶级目录里。一个典型的基于Express的项目，它的目录结构将会与下面类似：

```
routes/
    home.js
    login.js
    …
views/
    home.jade
    login.jade
    …
```

使用Express推荐的目录结构，你不得不将源码按照类型而非特性来分来。作为结果，紧密相关的文件将被分离到截然不同的目录树里面，这意味着如果你需要修改一个功能，你可能需要遍历整个项目的目录以修改不同的文件。在后面的替代方案里我会介绍一个更健康的、模块化的目录结构。

与Express高度耦合

当使用Express视图渲染引擎，意味着你在使用一个与Express高度耦合的特性。如果你想在客户端渲染一个模板呢？如何解析局部模板？如果你想切换到另一个web框架或者不使用框架呢？事实是，Express引入的这个解析和渲染模板的新方法，只能在服务端并且只能在Express中使用。当构建同构（isomorphic）web app时，我们希望在服务端和客户端使用更一致的渲染模板的方法。

没有Streaming

Express视图渲染引擎不支持streaming（流式传输），这对于客户端和服务端两方的性能都有负面影响。在服务端，当渲染一个HTML页面模板时，整个HTML输出都被当做一个超长的字符串存储在内存里，当整个HTML输出都构造完成之后，才开始向HTML响应里输出第一个字节。这是因为所有的视图引擎都必须使用回调来实现。而只有当整个HTML都被渲染完毕，回调才会被调用。不使用streaming来输出HTML，我们将浪费服务端的内存，以及增加客户端等待响应的时间。

使用streaming来传输响应的好处还包括，更早的发送页面的`<head>`部分，因此客户端能够更快的下载页面的CSS文件。为了感受streaming带来的好处，你需要使用支持streaming的模板引擎。此外，支持异步渲染的模板引擎（如Marko (https://github.com/raptorjs/marko)、Dust (https://github.com/linkedin/dustjs)和Nunjucks (https://github.com/mozilla/nunjucks)）甚至能够在视图模型构造完成之前就输出响应，因此能带来更多的性能提升。

集中式的配置

并不是所有的配置都不好，但不必要的配置是不好的。如果应用不同的地方需要不用的配置，集中式的配置将引起冲突。比如，如果需要多个view目录 (https://github.com/strongloop/express/pull/1186)呢？Express的sub-apps能在一定程度上解决这个问题，但最好的办法的还是避免额外的配置。

解决办法

Express将它自己描述为一个框架，那么绕过Express视图渲染引擎实际上还是很容易的。这将让开发者创造更灵活的、易理解的和更高性的应用。首先你必须理解一个重要的概念，`res`对象实际上是一个可写的HTTP响应流（尽管它已经被Express重度修改过），因此你能够直接向响应里输出：

```
app.get('/', function (req, res) {
    res.write('Hello Frank');
    res.end();
});
```

如果你想渲染一个jade模板到HTTP响应，你可以照下面做：

```
var templatePath = require.resolve('./template.jade');
var templateFn = require('jade').compileFile(templatePath);

app.get('/', function (req, res) {
    res.write(templateFn({name: 'Frank'});
    res.end();
});
```

这个方法比起`res.render()`有些啰嗦，但它很直观也更灵活。

注意：你可能注意到我们使用了require.resolve (http://nodejs.org/api/globals.html#globals_require_resolve)来获得模板的绝对路径，这个模板和我们的Controller模块放在一起。

如果你的模板引擎支持输出到一个流，代码将会更简单。比如Marko模板引擎的代码如下：

```
var templatePath = require.resolve('./template.marko');
var template = require('marko').load(templatePath);

app.get('/', function (req, res) {
    template.render({name: 'Frank'}, res);
});
```

视图解析程序

如果你觉得你的app需要使用视图解析程序（比如你需要为A/B测试使用不同的模板，或者根据用户的地区来确定模板），也有一个非常干净的解决办法。下面的代码假想了一个独立于Express的视图解析程序，它是如何根据名称和一些上下文（本例中是请求和当前目录）来解析视图模板的。

```
var myViewResolver = require('my-view-resolver');

app.get('/', function (req, res) {
    var template = myViewResolver.resolve('hello', req, __dirname);
    template.render({name: 'Frank'}, res);
});
```

使用一个明确的视图解析程序能让代码更容易理解，并提供更高的灵活度。

项目结构

现在你可以不受只能有一个views目录的限制了。让我们来看看新的项目结构：

```
pages/
    home/
        index.js
        style.css
        template.marko
    login/
        index.js
        style.css
        template.marko
```

新的项目结构让关联的文件放在相同的目录，从此之后可以模块化的开发和维护项目了。

总结

尽管Express是一个极简主义的框架，但也没有必要一定要使用它所有的特性。有些时候独立的模块能比它做得更好。Nodejs其中的一条指导原则就是“模块应该只做一件事并把它做好”，并且我认为Express将诸如视图渲染和路由这些特性分离开来会更好。我们已经看到Express 4.x里分离了不少核心的中间件，我认为还应该更进一步。也许我们根本不需要一个框架。也许我们只需要一些在一起工作良好的模块就行。

希望现在你对于绕过Express视图渲染的好处已经明了于心了，如果你想看看具体的示例，可以看这里的示例程序 (https://github.com/patrick-steele-idem/express-view-streaming)。这里还有更大型的天气示例应用 (https://github.com/raptorjs/raptor-samples/tree/master/weather)可供参考，希望能对你有所帮助。

原文地址： idlelife.org (http://idlelife.org/archives/810?utm_source=ourjs.com)

[非英文:1948, 总字符:3733]

社区评论 (Beta版)

#0 郭仿在 2017-08-13 20:22:02

？

回复

#1 杨中匆 2017-10-09 19:49:42

看看

回复