**前后端分离**

随着不同终端(PC/Pad/Mobile)的兴起，对开发人员的要求越来越高，纯浏览器端的响应式已经不能满足用户体验的高要求，我们往往需要针对不同的终端开发定制的版本。为了提升开发效率，前后端分离的需求越来越被重视，后端负责业务/数据接口，前端负责展现/交互逻辑，同一份数据接口，我们可以定制开发多个版本。

**一、什么是前后端分离？**

大家一致认同的前后端分离的例子就是SPA(Single-page application)，所有用到的展现数据都是后端通过异步接口(AJAX/JSONP)的方式提供的，前端只管展现。  
从某种意义上来说，SPA确实做到了前后端分离，但这种方式存在两个问题：

* WEB服务中，SPA类占的比例很少。很多场景下还有同步/同步+异步混合的模式，SPA不能作为一种通用的解决方案。
* 现阶段的SPA开发模式，接口通常是按照展现逻辑来提供的，有时候为了提高效率，后端会帮我们处理一些展现逻辑，这就意味着后端还是涉足了View层的工作，不是真正的前后端分离。

SPA式的前后端分离，是从物理层做区分（认为只要是客户端的就是前端，服务器端的就是后端），这种分法已经无法满足我们前后端分离的需求，我们认为从职责上划分才能满足目前我们的使用场景：

前端：负责View和Controller层。

后端：只负责Model层，业务处理/数据等。

**二、为什么要前后端分离？**

**2.1 现有开发模式的适用场景**

* 几种开发模式，各有各的适用场景，没有哪一种完全取代另外一种。
* 比如后端为主的MVC，做一些同步展现的业务效率很高，但是遇到同步异步结合的页面，与后端开发沟通起来就会比较麻烦。
* Ajax为主SPA型开发模式，比较适合开发APP类型的场景，但是只适合做APP，因为SEO等问题不好解决，对于很多类型的系统，这种开发方式也过重。

**2.2 前后端职责不清**

在业务逻辑复杂的系统里，我们最怕维护前后端混杂在一起的代码，因为没有约束，M-V-C每一层都可能出现别的层的代码，日积月累，完全没有维护性可言。  
虽然前后端分离没办法完全解决这种问题，但是可以大大缓解。因为从物理层次上保证了你不可能这么做。

**2.3 开发效率问题**

淘宝的Web基本上都是基于MVC框架webx，架构决定了前端只能依赖后端。  
所以我们的开发模式依然是，前端写好静态demo，后端翻译成VM模版，这种模式的问题就不说了，被吐槽了很久。  
直接基于后端环境开发也很痛苦，配置安装使用都很麻烦。为了解决这个问题，我们发明了各种工具，比如[VMarket](https://www.npmjs.org/package/vmarket)，但是前端还是要写VM，而且依赖后端数据，效率依然不高。  
另外，后端也没法摆脱对展现的强关注，从而专心于业务逻辑层的开发。

**2.4 对前端发挥的局限**

性能优化如果只在前端做空间非常有限，于是我们经常需要后端合作才能碰撞出火花，但由于后端框架限制，我们很难使用Comet、Bigpipe等技术方案来优化性能。

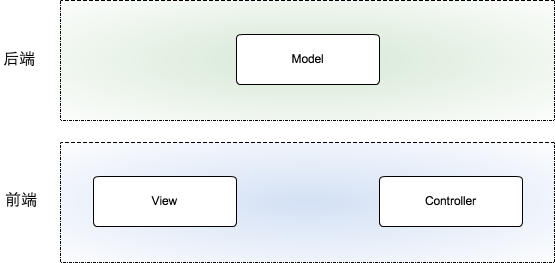
为了解决以上提到的一些问题，我们进行了很多尝试，开发了各种工具，但始终没有太多起色，主要是因为我们只能在后端给我们划分的那一小块空间去发挥。只有真正做到前后端分离，我们才能彻底解决以上问题。

**三、怎么做前后端分离？**

真正的前后端分离：

前端：负责View和Controller层。

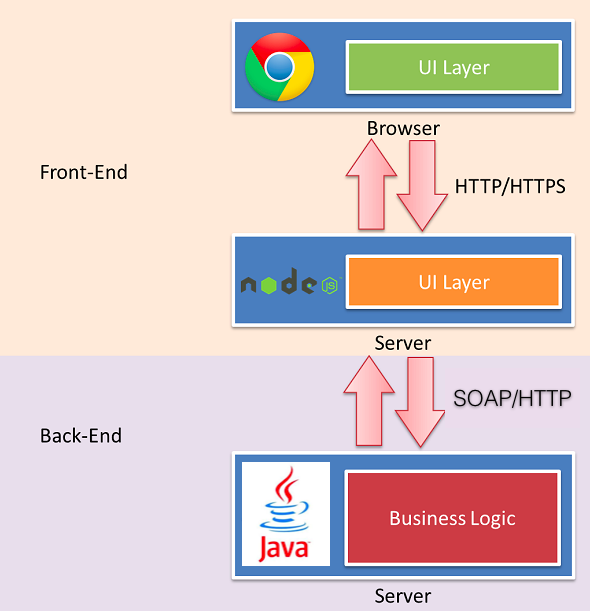
后端：负责Model层，业务处理/数据等。



还是根据view层数据输出json数据，我们还可以根据表现层需求很容易的做Bigpipe,Comet,Socket等等，完全是需求决定使用方式。

**3.1 基于NodeJS“全栈”式开发**

如果想实现上图的分层，就必然需要一种web服务帮我们实现以前后端做的事情，于是就有了标题提到的“基于NodeJS的全栈式开发”



这张图看起来简单而且很好理解，但没尝试过，会有很多疑问。

SPA模式中，后端已供了所需的数据接口，view前端已经可以控制，为什么要多加NodeJS这一层？

多加一层，性能怎么样？

多加一层，前端的工作量是不是增加了？

多加一层就多一层风险，怎么破？

NodeJS什么都能做，为什么还要JAVA？

这些问题要说清楚不容易，下面说下我的认识过程。

**3.2 为什么要增加一层NodeJS？**

现阶段我们主要以后端MVC的模式进行开发，这种模式严重阻碍了前端开发效率，也让后端不能专注于业务开发。  
解决方案是让前端能控制Controller层，但是如果在现有技术体系下很难做到，因为不可能让所有前端都学java，安装后端的开发环境，写VM。  
NodeJS就能很好的解决这个问题，我们无需学习一门新的语言，就能做到以前开发帮我们做的事情，一切都显得那么自然。

**3.3 性能问题**

分层就涉及每层之间的通讯，肯定会有一定的性能损耗。但是合理的分层能让职责清晰、也方便协作，会大大提高开发效率。分层带来的损失，一定能在其他方面的收益弥补回来。  
另外，一旦决定分层，我们可以通过优化通讯方式、通讯协议，尽可能把损耗降到最低。

**举个例子：**  
淘宝宝贝详情页静态化之后，还是有不少需要实时获取的信息，比如物流、促销等等，因为这些信息在不同业务系统中，所以需要前端发送5，6个异步请求来回填这些内容。  
有了NodeJS之后，前端可以在NodeJS中去代理这5个异步请求，还能很容易的做Bigpipe,这块的优化能让整个渲染效率提升很多。  
可能在PC上你觉得发5,6个异步请求也没什么，但是在无线端，在客户手机上建立一个HTTP请求开销很大，有了这个优化，性能一下提升好几倍。

淘宝详情基于NodeJS的优化我们正在进行中，上线之后我会分享一下优化的过程。

**3.4 前端的工作量是否增加了？**

相对于只切页面/做demo，肯定是增加了一点，但是当前模式下有联调、沟通环节，这个过程非常花时间，也容易出bug，还很难维护。  
所以，虽然工作量会增加一点，但是总体开发效率会提升很多。

另外，测试成本可以节省很多。以前开发的接口都是针对表现层的，很难写测试用例。如果做了前后端分离，甚至测试都可以分开，一拨人专门测试接口，一拨人专注测试UI（这部分工作甚至可以用工具代替）。

**3.5 增加Node层带来的风险怎么控制？**

随着Node大规模使用，系统/运维/安全部门的同学也一定会加入到基础建设中，他们会帮助我们去完善各个环节可能出现的问题，保障系的稳定性。

**3.6 Node什么都能做，为什么还要JAVA？**

我们的初衷是做前后端分离，如果考虑这个问题就有点违背我们的初衷了。即使用Node替代Java，我们也没办法保证不出现今天遇到的种种问题，比如职责不清。我们的目的是分层开发，专业的人，专注做专业的事。基于JAVA的基础架构已经非常强大而且稳定，而且更适合做现在架构的事情。

**数据联调**

**问题重现：**

小张同学的项目采用的是前后端分离架构，现在前后端程序猿已经碰过面，协商好了一些前端需要的数据接口，于是前后端程序猿同时开工码代码。  
这时候小张同学某个页面需要后端数据，但是后端叔叔还没有将该数据的借口开发出来，此时小张只能做虚拟数据（专业点叫**mock数据**），但是小张的ajax的接口写的是与后端叔叔商量好的绝对路径（域名+请求路径+请求参数，跨域问题已解决），因为这是以后真正的请求路径，所以小张又不想先写本地相对路径，后期再来修改（万一后台叔叔开发的慢了，鬼知道有多少接口要修改呀）。于是他就迷茫了。。。

仔细看看这其实就是前后端分离中的mock数据和联调的问题，就现在来说能解决的方式有很多种。先说mock数据，gulp，webpack，[fekit](http://ued.qunar.com/fekit/)（去哪儿网的一款前端自动化构建工具，据说历史比webpack和gulp都要久远）等等自动化构建工具都有mock数据的功能，这不是问题；再说绝对路径的问题，其实只需要做一个host的映射就行了。

# 什么是前后联调

我的理解是前后端联调是一种真实业务数据和本地mock数据之间来回切换以达到前后端分离架构下的不同开发速度时数据交换的一种方式方法。

# 为什么要前后端联调 回到上面小张的问题，假设小张现在ajax中的url写的是相对路径（如：./mock/user.json），小张现在面临的是后端接口没有数据，需要本地mock数据，但是当后端接口开发完成，并且有了测试数据后，小张是不是需要切换到后端数据呢？答案是肯定的，但是小张需要在本地mock数据和后端接口数据之间来回切换，为什么呢？很明显吗，本地的mock数据是小张自己写的，肯定符合前端需求，但是后端接口首先需要测试通不通？还需要测试数据格式对不对？有可能前两部都整好了，但是后端还没有填写足够的数据（比如，列表页，小张想分页，后端叔叔就写了两条测试数据，你咋整？），所以呀，小张需要根据后端叔叔对接口的调整不断的来回切换url，这样岂不是还在受后端的影响，还谈毛线前后端分离呀！

所以当我们在前后端分离架构中，不断来回切换本地mock数据和后端接口数据时就遇到了联调的问题。

# 实现前后端联调

首先我们已经知道，在现在流行的**“前后端完全分离”**架构中，前后端联调是一个不可能避免的问题，就我目前对前后端联调的理解来说，我认为需要联调的有以下两个方面的资源：

**css、js、图片等前端静态资源文件。**

**ajax获取的后端数据。**

而前后端分离的架构的应用可能也分为两种情况：

**前后端完全分离，前后端分别拥有自己的域名和服务器。**

**前后端开发分离，但是部署时是一个域名和一台服务器。**

还有就是需要联调的场景也是两种情况：

**开发阶段**

**调试阶段**

**\*\*\*\*先解释一下 ‘222’**

**2种资源：静态资源 | Ajax调取的数据**

为什么我将资源分为两种呢？首先来看css、js文件和图片是通过link、scripts、img标签来引入的（样式中的图片也是一个道理）， 它们是不存在跨域问题的，你在哪里引入都无所谓。

而ajax获取后端数据是有跨域问题的，虽然说可以解决，但是有可能是该数据（或者后端部署环境）和上面的css、js、图片等静态资源是不在同一台服务器上的，也可能是同一台服务器不同域名下的，如果都在同一域名同一服务器下，开发阶段肯定也不在一起吧！

**2种模式：两台服务器对应两个域名 | 一台服务器和一个域名**

虽然架构可以采用前后端分离，但是部署有可能就不一样了，这和项目的大小，公司的情况等等都有关系了，一个百八十人用的小系统，还得两台服务器两个域名，你不觉着浪费吗？

2种不同的部署情况直接导致了前期在设计联调方案的时候就不同了，下面会详细介绍。

**2个阶段：开发阶段 | 调试阶段**

这不用多说了，虽然现在主张前后端分离，但也不能不和后端叔叔亲近亲近了呀，不然谁给你数据区，所以说开发阶段也遇到联调的问题。

**\*\*\*\*联调开始**

在讲解联调之前，你需要了解主流自动化构建工具本地mock数据的方法，本例以gulp来讲解，不了解的小伙伴可以看看[Gulp和Webpack对比](http://chping.website/2016/10/18/gulp%E5%92%8Cwebpack%E5%AF%B9%E6%AF%94/)的mock数据部分。

**\*\*\*开发阶段**

\*\*\***两台服务器对应两个域名**

这是最nice的方案，也是联调最舒服的方式。

在这种方式下，前后和后端各有各的服务器和域名。后端会把他们开发的后端业务代码，实时的上传到服务器上（也可能不是，如放在后端叔叔自己的电脑上，实现是一样的，只不过域名变成了叔叔的ip地址）；而我们前端的代码会实时的上传到我们前端的服务器上（同理也可能在自己的电脑上，开发完一块上传）。

为了后面方面讲解，前端服务器域名为“chping.website”，后端服务器域名为"chping.site"。

**css、js和图片等静态文件**

此时，我们的静态文件的请求路径建议使用我们前端服务器的绝对路径，为什么呢？因为如果我们写本地的相对路径，等把我们的入口文件交给后端后，路径还是绝对路径，岂不是成了后端服务器的路径了，那就坏菜了，还得一个一个修改成我们前端的绝对路径（如：http://chping.website/css/reset.css ）。

那可能有小伙伴说，这样的话，我在开发阶段，我们的服务器上还没有自己的静态文件呢，去请求的话，会报404的！

对的，这时候就该我们的host的出场了，我们知道当进行域名解析的时候，第一步是在浏览器缓存中寻找，如果没找到，第二步在我们系统的缓存中寻找（就是本地的hosts文件），然后....（想详细了解请看[网络请求过程扫盲](http://chping.website/2016/09/11/netword/)的DNS解析部分）就没有然后了。既然域名解析需要通过hosts文件这一步，那么我们就修改一下它，让访问http://chping.website这个域名的请求，全都访问我们本地的mock数据就好了呀！

在hosts文件中添加下面这一条

127.0.0.1 chping.website

好的，搞定，从此以后所有访问域名http://chping.website的请求都会被映射到本地了，然后再结合gulp的mock数据，就完成了你写的是域名+绝对路径（如：http://chping.website/css/reset.css ），但是请求的却是你本地mock的数据（上面的请求会被映射到http://localhost/css/reser.css）。这样你就不需要来回修改请求路径了。

**ajax后端数据**

开发工程中，静态文件的联调问题可能还是比较少的，特征也不明显。但是通过ajax获取后端数据使用联调的情况可就多了去了，前面**为什么要前后端联调**一节中已经通过小张的例子说明了，就不在赘述了。

其实在介绍完**css、js、图片等静态文件**一节，相信你也能猜到了怎么去前后端联调了。很简单，ajax中的请求还是写绝对路径（如：http://chping.site/userlist，获取用户列表），此时后端接口如果没有开发好，和上一节一样通过hosts文件的修改可以实现chping.site域名映射到本地来。如果后端像让你给他测试一下接口获取的数据对不对，你可以在修改一下hosts文件就可以了。

可能还有小伙伴问了，这样还不是改来改去，麻烦死了。首先来说，相比较你修改多个url地址和修改一个hosts文件，孰轻孰重，一目了然；其次，怎么可能让你去每次都去修改hosts文件呢，麻烦死了。现在向大家推荐一款小插件，在google商店里搜索一下**HostAdmin**就找到它了，下载安装就好了，然后简单配置一下你的hosts地址，就可以愉快地切换hosts的映射了（HostAdmin的使用自行google，干这行岂能不会用软件）。

好了，两台服务器对应两个不同域名的场景在开发阶段就这样愉快地结束了。但是在实际开发中，中小公司很少给你两台服务器和两个域名的，一般正常情况就是一台服务器一个域名的。

**\*\*\*一台服务器一个域名**

我仔细想一下，此时这个服务器会给谁管理呢？

想都不用想肯定不是前端，后端叔叔比你对服务器熟悉多了，交给你去管理那是出鬼了（我还真遇到过，不过也是给后端首先使用，最后才把前端的代码放到服务器上），所以说在这种情况下，你就老老实实在本地开发就行了。

**css、js和图片等静态文件**

这时候的静态文件再开发阶段不需要任何考虑，按照你喜欢的相对路径或者相对于项目的根路径的形式写就行了，因为早晚还得交给后端。但是，需要注意：

1、如果你采用**相对项目根路径的书写方式来写你的静态文件路径**时，一定要先和后端叔叔商量好，将来项目部署的时候他会把你的前端整个项目放在哪里？如果不是根目录下，你就挂了。比如：你的reset.css的路径时/exports/styles/common/reset.css，后端叔叔把你前端项目放在了根目录下的frontEnd文件夹下，reset.css文件就报404了。

2、如果后端采用的java，你需要特别注意的是，**tomcat的根目录**并不是webapps文件，而后端项目默认是部署在webapps/ROOT文件下的，所以你如果使用了相对项目根路径的书写方式来写你的静态文件路径时，对不起又是404了。

**ajax后端数据**

因为现在唯一的一台服务器还是在后端叔叔那里，所以此时你还是可以写绝对路径（域名+请求路径），利用hosts文件来改变域名映射实现联调。

**开发阶段小结**

其实在开发阶段做联调的情况还是很少的，一般都在自己的机器上开发，后端会自己做单元测试，根本不用前端来给他做测试的（先别骂，完事一块，哈哈）；而前端更不需要了，都在自己机器上开发。及时有两台服务器和域名，有几个每开发一个功能就上传服务器呀（此服务器指项目上线时部署的服务器），有病吧！所以说，在这个阶段只需要做好hosts映射就行了，不会让你来回切换域名映射的。

但是接下来的调试阶段就不一样了，项目前后端大体都已经开发完成了，Beta版本已经部署了，这时候如果出现bug就需要频繁的线上线下（真实数据和mock数据）切换了。

**\*\*\*\*调试阶段**

调试阶段的情况说明刚才提了一下，现在再详细的阐述一下：

首先，已发布的Beta版本在测试的时候，发现前端页面上有个地方有个小bug，需要你修改一下，于是你访问线上的页面，看了一下这个bug，发现是某个css样式文件里面出错了，于是你希望在本地修改，然后线上版本可以直接查看，这样就可以实时看到你的修改在线上的真实样子了。

#### \*\*\*两台服务器两个域名

这里就不在分静态文件和后端数据分开介绍了，其实道理都一样的，另外此时后端数据都是真实的了，也不需要你管后端了，此时你本地的mock数据已经下岗了。

首先你只需要打开前面你安装的**HostAdmin**，通过它配置一下你的hosts文件（如果年前面的做了，此时你已经配置好了），然后chping.website这个域名你可以在本地和线上之间来回切换了。

再具体一点就是：

1、启动你本地的项目

将chping.website域名通过HostAdmin设置为

127.0.0.1 chping.website

# 202.201.112.232 chping.website //假设202.201.112.232是线上服务器的ip

2、打开浏览器，并清除一下浏览器缓存，重新打开你们开发的网站。此时网站是获取到的静态资源就是从你本地获取的了，但是ajax获取的数据确实后端返回的真是数据。

3、然后你开始修改bug，这时候线上的版本可以实时查看你修改的前端内容，修改完bug，QA说了OK，你就可以把修改的文件替换掉服务器上的版本就行了。

#### \*\*\*一台服务器一个域名

这种情况就复杂了，此时只有一个域名，如果你开发阶段是按照我前面说的，写的是相对路径，那么此时你是没有办法实现两台服务器两个域名那种线下修改，线上查看的高大上联调的。那该怎么办呢？

第一种方式就是，你在本地修改，本地查看，测试好了以后上传到服务器，看看线上环境可不可以，OK当然好；不行就本地接着改，然后在上传。

第二种方式就是，在开发阶段你给静态文件前面加上一个假的域名，然后像两个域名一样进行假的域名与本地映射配置，然后像两个域名一样开发。等部署的时候，在服务器上也暂时做一个本地映射，你就可以实现同两个域名一样的线下修改，线上查看了。但是，最后真正上线的时候，你就需要收到，把假域名全部删掉。所以说，一台服务器一个域名的时候，前端静态文件做联调是很不方便的。

#### 小结

总结一下，**前后端联调**是一种**真实业务数据**和**本地mock数据**之间来回切换以达到前后端分离架构下的不同开发速度时数据交换的一种方式方法，而实现就是利用自动化构建工具的mock数据功能和修改域名解析文件hosts来实现的。

如果你的项目是两台服务器和两个域名，随便你折腾。但是如果你是一台服务器和一个域名的话，就有些麻烦了，而这个麻烦仅对于你前端静态文件联调有影响，需要你好好思考一下怎么解决了，反正我目前还不知道去怎么