《大型管理系统前端架构设计》

百度 EFE

王迎然：大家好，我是来自去哪儿网的王迎然，今天其实分享也是我一个处女作。我先做个自我介绍王迎然，来自去哪儿网的前端架构设计，喜欢摄影、旅行做家常菜。这边是我们之前团队出去普吉岛拍的一张照片，重点大家希望知道吗？其实我已经结婚了。下面跟大家互动一下，在场的男生有时间约妹子举举手？没人举手吗？一个都没有。为什么没有时间约妹子，其实我也没有时间，再说我也不敢去。主要是我们太苦了。下面我介绍一下什么是管理系统？我在百度的百科上，然后摘了一段话。以企业管理需求为基础，以 IT 技术为支撑，为企业提供数据信息的综合管理办法。摘自百度百科。其实说白了就是为企业的需求去开发他们的管理软件。既然是管理系统那它有什么样的特点呢？特点我总结了一下首先是固定用户，管理系统因为是商户在使用他可能每年都在使用，它是一个固定的用户群体。重业务轻视觉，一般管理系统的界面，可能比较单一。然后它主要是为了解决商户的功能性的需求，所以它的业务比较强。

比如说我们酒店管理系统里面比如说去开房，比如说去对用户的一些信息修改之类的。可能操作非常多。还有它的一个浏览器相对宽松什么是比较宽松呢？就是商户的浏览器可能是我们在实施安装的时候或者建议他们高版本的浏览器，这个跟普通的版本就不一样。可能更偏向于高的版本，所以我们前端在真正开发的时候，可能它的空间也是比较大的。

下面我来讲一下，就是管理系统，一般我们采用什么样的一个架构方式呢？然后这一是张比较简洁的架构图。通过浏览器的访问，然后到控制层。到业务逻辑层处理，再到数据的服务层，最终它再返回到前端的浏览器，这是一个很常规的 BS 的一个系统架构的简单图。那我们前端基本做哪些事情呢？我们前后端。我们前端做了一个 VIEWS 层，后端做的系 CONTROLLERS 和 MODELS。这种实现还是比较常见的，那未来我们会实现一个什么样的方式呢？未来我们自己去做 VIEWS 和 CONTROLLERS。那 CONTROLLERS 怎么实现呢？会有哪些优点呢？使用 CONTROLLERS 自己控制一些接口的展现层的处理，前端的 GS 不用写的太复杂，就是数据端的处理，我们通过 CONTROLLERS 处理就好了。

我们现在的传统开发方式 CONTROLLERS 是在后端处理的情况下，我们会有一个什么样的问题呢？就我们的苦恼。首先学习成本比较高，因为我们 VIEWS。因为前端需要学习（英文）的知识，那你自己本地还要启动一个（英文）之前我们有一个新人来了我们部门之后，说前端怎么去本地开发呢？还要去（英文）。搞了半天他也没有把这事情弄出来，最后找后

端去问，然后各种去求后端，后端那一帮人也没时间，搞了一天本地环境也没搭建好，这样的学习成本是比较高的。

还有它的可维护性，一般我们一个上午开发时间久了，然后如果说没有一个好的前端架构规范的话，那它的可维护性其实是非常差的，时间越来越久的情况下，我们代码非常臃肿。还有如果说学习成本又高、可维护性又差，那我们的开发效率是非常低的。还有就是我们本地开发，我们本地开发过程中，然后怎么去调试那些假的数据、因为我们前后端分离的。我开发的过程中怎么样让我们本地的工程跑起来了。还有我们线上调试，比如线上除了一个（英文）我怎么样快速的定位、怎么样快速的解决、发布。这样一系列的问题我们是不是苦恼，这是多么痛的领悟。

那我们怎么去解决这些痛呢？我们前端这一块，做的解决方案大概是四个方案，从四个点，一个方案做了四个点，首先从工程的组织结构来开始。还有 MOCK 数据，还有多环境的链条，还有一个线上发布的流程。从这几个方案我们来解决我们的痛。首先我们从工程的组织结构来去讲一下，我们怎么去组织前端的工程概览呢？这是一个工程概览的大家看一下，前面的 AURA 大的名称。VM 还有一个 SRC 是我们前端资源的目录，还有一个 MOCK 目录，我们的规范是三个目录。那我们先从 VM 目录来看一下。一般 URL 设计，比如是我们一个页面，我们 URLACCOUNTLOGIN.THM，这样一般的模板目录是怎么划分的呢？VM 模板的结构 VM 有一个配置词，然后建一个（英文）目录。这样对应的结构，其实这样对应的结构关系其实有一定规范。按照我们一个功能的模块进行划分，再具体到页面的级别。页面的访问跟它的 VM 的位置是一一对应。其实在我们去开发的过程中，只要知道你的 URL 地址其实很容易定位到你的模块目录。后续 GS 目录跟（ 英文）目录也是有一定的关系。

那我们模板目录是什么因的呢？VM 目录下我们去划分了两个文件夹一个 PAGES、WIDGET。WIDGET 有 INCLUDE 代码的路段。比如说（英文）我们会把一些代码片段抽离出来，这样改动的时候也是比较方便。比较印象深刻的是之前有一个代码改动，比如说需要改一下静态资源服务，是要改一下域名，其实只要改一下通用的代码片断，所有的模板就生效。按照刚才的 URL 设计然后是一个文件夹的形式。其他的页面也是遵循这样结构的划分去存放就 OK 了。

这是代码的事例，这代码非常简单，我们管理系统是采用通过引入（英文）GS，还有业务配置的 CSS，CSS 我们有两个一个是基础一个是业务的。GS 也是，我们会引入一个（英文）还有业务文件的。其实中间还看到（英文）DRV 的标签，其实整个我们前端架构（英文）的框架。我不知道大家有没有了解过（英文）有使用过的吗？其实我们整个前端的架构方式是

采用浏览器端去渲染。因为我们现在页面已经采用了一个标签，其他的一些展示代码，就展示的样子全是通过 GS 渲染出来。那可能同学们会问，那你这渲染可能会白屏，GS 可能会很大或者怎么样，那为什么这样设计呢？因为我们是管理系统。管理系统其实它关注的几个点，就是首先它的业务逻辑比较复杂。然后它还需要控制一些权限，可能就是操作的一些功能模块比较多，可能他们之间的功能会有一些依赖，所以说我们采用（英文）架构的话，会使用双向绑定还有一些模块间的通讯来去解决模块间的依赖。

这样的话通过权衡的考虑，这些问题跟业务逻辑还有开发的效率来讲，我觉得这其实有一定的权衡，一个架构好赖，要跟实际的业务去出发。这样的话在符合你开发的比较合适架构。

下面我讲一下我们的 SCRIPTS 的组织架构。我们整体采用的是一个组件化开发方式，从这个图可以看出除，我们这一个页面其实还有 Sidebar、moduleA、moduleB 等。那我们在开发中你很容易理解依赖在哪里？会把一个很复杂的页面拆分成各个模块间的依赖关系去组织成一个页面。

那 Scripts 分了几个目录，（英文）为什么还会用（英文）可能在有一些其他的开发项目中，也有一些（英文）满足不了我们的需求，然后（英文）有一个划分（英文）。因为管理系统它的界面是比较规整，可能每一个液面长的非常像，比如说它的头部导航还有一些它的侧边栏，可能是比较一样，所以我们把它抽象出来。（英文）其实是它的整个布局的代码。然后 module 里面然后会放一些组件，我们会把整个的页面弄成组件的形式。然后我们把（英文）的布局会拆分一些 module 的形式。比如说头部的导航区，还有它的功能区，还有一些侧边栏，我们都把它拆分出来，拆分出来（英文）企事业是按照（英文）去存放。这样的话我们每个模块都有自己的模板，都有自己的 Scripts 的逻辑。它配置里面其实跟（英文）也是类似，配置这里面我们会以页面的形式去区分它的目录，比如说（英文）是我们的叶面，我们也会有一个入口，入口的名字叫（英文）。然后还有一个（英文）文件对当前的文件做一些基础上的配置，还有一些业务的需求。

基础模块跟业务模块两个 Scripts 它的引入是我们 LIBS，还有 LIBSAVALON.JS，还有一个 JS 文件，里面可能会有一些全局性的配置，网站的一些配置，可能还有一些全局变量，可能是供页面中的数据交换。还有一个 HOOK.JS 去做一个引入。

业务模块里面 DEMO 首先会引入 ENTRYJS，就配置一些网站的信息，还会引入 LAYOUT.JS，下面业务层的组件，就是 module 的解释。它会把所有的模块初始化，那我们的页面就渲染好了。

那刚才其实我们整个的页面是采用了组件化的开发方式，那可能会想，这样的模块全不拆分了怎么进行通讯呢？我现在画了这样一个小例子，里面有两个按纽，还有展示的模块。现在有这样的需求，就是点击新增按纽，可能出现一个弹出窗口，然后弹出窗口做一些新增的项目，提交之后我们展示模块刷新，将我们新增的数据展示出来。这样的话我们会这样去处理，我们点击新增的时候，去打开一个新增的模块，新增的模块操作完之后，我们新增模块再去调用我们展示模块的一个刷新的方法。这样的话就是我们每一个模块之间都会提供出对外开放的通讯借口，那在（英文）其实观察模式里面的注册事件。

那修改其实同样也是这样的，就是调用修改的模块，然后修改模块操作完之后，再去调用我们展示模块刷新的方法。那这样的通讯模式代码会怎么去写呢？这里面写一个简单的例子，然后大概给介绍一下。我们会先进来两个模块，最上边是引入的一个新增模块，然后还有一个修改的模块。其次第三行代码是引入的模板。其实我们还要继续定义它的 VM。在下面有 DATE 的其实跟我们模块绑定好了。下面有一个方法叫（英文）其实里面写的（英文）DATE 其实请求一个数据，对我们的 VM 进行一个复制，这个数据就显示到模块中。

两个事件里面点击事件其实这两个事件是跟刚才上一张图里面新增按纽，跟修改按纽其实是绑定的。就是点击的时候我们会去触发一个新增的窗口，会将我们的弹出层打开，修改的时候也是。点击修改按纽打开我们修改的窗口。新增完之后就会去调用我们，在新增的窗口里面去调用我们一个（英文），我调了一下（英文）这样几行代码，其他的结构就能实现我们上一篇讲的新增修改，还有页面区去展示，其实这三个功能，如果说采用一个常规的方式去开发的话，可能会写的很多。其实按照我们这样一个结构去设计的话。那我们区区几行代码就把这功能搞定了，是不是很爽，前端开发的时候也是很快速。

其实 VM 跟 Scripts 讲过了。其实再讲一下 STYLES 一样。其实 Scripts 跟 STYLES 一样。其实我觉得架构设计这东西，你怎样去设计你们团队里面怎样去用的比较顺手就 OK 了。其实放在什么位置重要性也不是很大。我们倾向就把 Scripts 跟 STYLES 区分开。这里面四个目录，一个 COMMOE 目录还有 LIBS 还有一个 LAYOUT 还有 PAGES。

比如说（英文）样式还有导航的，就各种的通用的。LIBS 会放一些常用的组件，比如说 reset.css 等等。其他的像比如说需要使用一个开元的 css 框架的话，其实也可以放到 libs，libs 也可以分为 indes，module 下面分了几项一个 header 还有 footer，其实把它的框架拆成了几个块，在不同的地方去维护这样的话定位也是比较好定位。

最后一个是 pages，是每一个页面的文件。拿一个 demo 为例子也是 entry.css，组件的放到 module 里面，然后是 indesscss，其实就是我们 sars 后缀的方式。就是我们 scss 是采用

sars 管理。如果你想去做成相应模式的话，其实你可以去拆分，把 Scripts 的模块拆分把 STYLES 的模块拆分其实目录是一样。

OK 其实 CSS 这一块也没有什么太多。下面看一下加载方式也是加载基础模块还有业务模块。基础模块里面我们进入了它的一些组件（英文）。其他我依据需求去加。业务模块引入了一个（英文），就是框架跟业务的。按照这样的一个规范来讲的话，那我们每一个业务页面都是这样的划分模式，就是它的规范性提高了。

下面我来节杀一下 Mock，在场的同学知道 Mock 这个词吗？我觉得应该也知道。Mock 其实在后端叫的比较多在前端这边其实可能说白一点就是造一个假数据，我们在前端开发的时候需要写一个假的数据供前端去开发的时候去使用。在最初我们采用什么方式，进来一个人就自己去开发，我们当时采用的一个软件 Charles 是什么样的东西，能抓大网络请求之后进行一个修改。我们当时拿这软件去为 Mock 数据。可能有几百个上千个借口。拿这去管理极方便，可能需要花大量的时间去管理会耽误很多时间。那后来我们又采用了一种其他的方式 MockJSON 是开元的方案，在本地的开发环境中，我们本地是基于 Mock 的开发环境做了一套方案，开始我们是表达去匹配请求的地址，再将我们本地做一个映射。其实看样子是简单一点，假如上千个接口我们还得一点一点去写。但是它的好处不依赖与第三方软件。我们在自己去加一个配置软件配置一下就好了。但是还是满足不了我们的需求。比如说几个同事在开发一个项目，可能是三四个人并行在开发，那每个人都在写这些东西可能产生一些冲突，那我们产生冲突要解决，那我们耽误很多时间，后来我们就去想方法去解决，还是用这个。我们用的 MockJS 的方案，去匹配 URL。将我们匹配的 URL 解析去把它拼出来，去匹配当中的 JSON 文件，将里面的数据进行读取在对外进行输出，这样方案就解决了我们的问题。我们 Mock 就不需要一条一套去写，只需要根据 URL 格式去定义文件，只要放到相应的目录里面就 OKL。按照这样的方案，我们前端去 Mock 的目录怎么去设计呢？假如一个接口，其实是互相对应的，按照接口我们是建一个文件，其他的接口是按照一个 URL 的格式去划分，我们只需要将每一个接口就可以了，其实是节省了很多时间。

那其实我们本地的开发，Mock 还有我们的工程的目录去都讲一下，我们怎么去连调呢？我们在开发的过程。本地的专业 Mock 目录还有 JS、CSS 目录，通过这样的方式就完全解决了，我们四套环境的连调，你再回想一下之前模块里面，其实是采用的 JS 去渲染我们的页面，这样的话我们在连调的过程中，其实只要将你的 HOSTS 去调试，那我们直接把环境线上的代码， 那我很快就调完了就发布上线，这过程可能也就十几分钟、二十分钟很快就搞定。这样的方案我其实是符合我们团队的预期。

但是我们看到前端的过程中，那你会想到模块前端了，那你怎么去访问呢？有人会疑问，那其实我们把，会把（英文）模块发布到我们的 Maven，最终发布的时候只需要将这些资源进行整合，这些资源的整合是通过我们内部的发布系统去直接自动化脚本去发布，我们也不用关心。

我介绍一下我们项目的案例，我最近在做一个项目，就是酒店版的系统，从这样来看的话，其实它的功能也是很复杂，其实里面会涉及到很多逻辑。因为我们是做智能门锁的系统。智能门锁会涉及到智能的门锁，会通过下发一个密码，通过手机 APP 开门，这样的系统很复杂，我们之前开发的时候就很苦恼，后来我们就是对这种前端的架构去改进去设计，最终去开发这项目。这还有一个客人的页面，还有客人的信息可以进行修改，上面还有管家密码的展示，其实这一套，它这系统里面还会跟我们的硬件做一些关联。

那我们这项目其实做了一些改进，因为预估的工作量前端大概是两百，后端是五百多一千，我们感觉很惊讶，我估了两百多，但是我们通过前端的架构，最终开发的项目，我们其实老大对我们也很惊讶，我们整体的效率提升到 30%，可能这个数字做更多的时间，我们学习一些更多的基础，这样的话我们还会有一些提升，有的提升我们还可以有时间去约妹子，一起去喝咖啡。

我希望的是前端提高一些自己的效率，陪陪老婆，陪陪孩子，希望大家都有一个比较美满的生活，不要很苦的让人一想到很苦。其实我们都是努力在向这方向发展，其实我们目前还是很苦的，但是我们正在向这方向在发展，有约妹子。我们老大让我们打一个广告，如果有希望前端想加入我们团队的，可以简历发送到我们老大的邮箱，之前我们老大也是百度的。我大概就讲这到里。大家有问题可以提一下。

提问：您好，我看这东西我最关心是你们路由？

王迎然：是这样，路游，按照页面的纬度，其实是模块，前端是通过通用的（英文），后端请求之后就是直接映射到的模块，如果映射不到 404。

提问：我就是说页面自然加上了，有时候我只需要这页面，我其他的页面？

王迎然：我明白的意思你的意思按需的加载。

提问：是的。

王迎然：按需加载我也考虑过，可能存在各种考虑吧。我们就是调研了一下商户，因为在管理系统中，商户对于速度上还有一些并不是太关心，可能满足他们业务的需求就 OK，但是我们在这样处理的时候，我们是将前端代码整个加载到本地去执行的。至于说包的大小，我们也会通过前端的手段去控制，但是说你说的按需加载之前考虑过，但是后来采用过目前的方案但是这种可能更直接一点。

主持人：因为限于时间关系，可能后续圆桌讨论跟嘉宾来一个头脑风暴。还有一次提问的机会，哪位同学？

提问：你好你讲的大部分都是规范目录，我想优化这一块压缩网络传输这一块，你们做了一些工作吗？

王迎然：其实压缩这一块，其实在我们整个去哪儿里面有通用的发布策略，还有压缩的这些工具，比如说百度有自己的，我们其实去哪儿有，还有本地开发的工具，其实我们介绍本地开发的时候，我们其实通过 LOT 层，但是后面还是通过（英文）去渲染。我们时间点也不是很充裕。

提问：好谢谢。