

b/s架构开发的发展历程




成钞公司印钞管理部 李宾

2018-01-09

自我介绍

李宾，07年北京印刷学院印刷工程专业，07年进入成钞公司，12年四川大学软件工程在职研究生，13年6月进入技术质量部。

 www.github.com/realeve

 41192163

质量信息管理系统代码片断，2014年，asp，第一次涉足B/S架构开发。

```
1 <table style="width:720px;margin-left:24px;table-layout:fixed;align:center;cellp
2 <tbody>
3 <%
4 strMachineTemp=""
5 nColor=2
6 Do Until rsg.EOF
7     if strMachineTemp <> rsg("机台") then
8         nColor=nColor+1
9     end if
10    if nColor mod 2 =1 then
11        HColor="#F4F4F4"
12    else
13        HColor="#FFFFFF"
14    end if%>
15    <tr>
16        <td class="FirstRow" bgcolor="<%=HColor%>"> <%=rsg("品种")%></td>
17        <td bgcolor="<%=HColor%>"> <%=rsg("正反面")%></td>
18        <td bgcolor="<%=HColor%>"> <%=rsg("机长")%></td>
19        <td bgcolor="<%=HColor%>"> <%=rsg("总张数_技质")%></td>
20        <td bgcolor="<%=HColor%>"> <%=rsg("平均得分_技质")%></td>
21        <td bgcolor="<%=HColor%>"> <%=rsg("总张数_质检员")%></td>
22        <td bgcolor="<%=HColor%>"> <%=rsg("总张数_机台")%></td>
23        <td class="LastRow" bgcolor="<%=HColor%>">
24            <%if rsg("标准偏差_机台")<1 then%>0<%end if%>
25            <%=rsg("标准偏差_机台")%>
26        </td>
```

常见问题 —— 一锅粥

1. 模块化程度低

菜单：页面间菜单的复用

报表：所有报表需要手动定义thead,tbody

2. 逻辑与页面视图高度耦合，后期维护困难

3. 样式直接写在html中，未使用css管理样式

4. 部分页面使用table定位布局，对于当时已兴起的html5技术尚未涉及

在过去10年，以asp 快速实现功能 的开发模式成为了许多初学者的首选，在后续的深入开发中可能都会遇到这样的一些问题，此时具代表性的开发软件主要是dreamware.

“PHP是世界上最好的语言”

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <title>Document</title>
8     <!-- css 文件引用 -->
9     <?php include "templates/style.php";?>
10 </head>
11 <body>
12     <!-- 头部 -->
13     <?php include "templates/header.php";?>
14
15     <!-- 侧边菜单 -->
16     <?php include "templates/sidebar_menu.php";?>
17
18     <!-- 页面主体 -->
19     <?php include "templates/content.php";?>
20
21     <!-- 底部，同时包含必要的js文件 -->
22     <?php include "templates/footer.php";?>
23 </body>
24 </html>
```

php

PHP

2016年开始重构成钞公司质量信息平台，以php为服务端，解决了模块化的问题。

目前PHP是应用最为广泛的服务端语言，因为简单易学成为了许多开发者的首选语言。在应用较广的PHP框架中主要有ThinkPHP(对国内主流虚拟云主机支持较好：阿里云，新浪云)、CodeIgniter(轻量、简单)、Larvel(社区目前发展最活跃)等。

在用php开发的过程中，逐渐对mvc/mvp有了学习和认识，更多技术细节推荐阅读这篇文章：[MVC，MVP 和 MVVM 的图示](#)

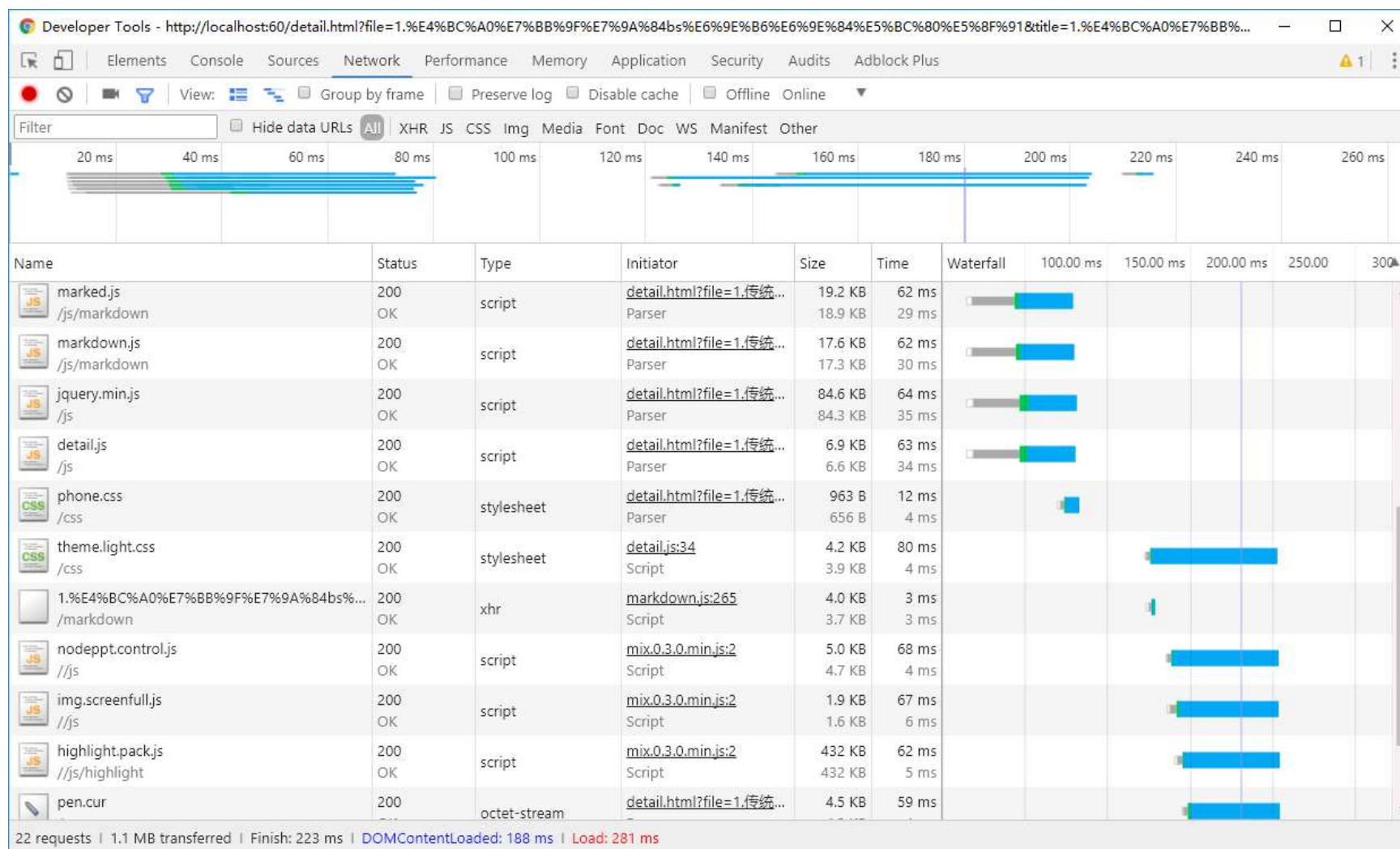
Server Side Rendering 服务端渲染 (SSR)

不管是用asp/php/java作为服务端，我们把这种 通过服务器直接生成html文本 输出至浏览器的方案统称为服务端渲染。其最大的特点是查看网页源码时能看到页面所有的dom元素的html文本。

在印制行业中有许多应用的功能是以数据或报表为主，点击 查询 按钮后，整个系统将经历以下过程：

1. 浏览器端/客户端/前端发起查询请求，http Request;
2. 服务端根据前端请求，连接数据库，发起SQL请求并等待数据库返回结果；
3. 服务端对返回的数据拼凑 Table 的 HTML 文本返回前台；
4. 前台读取html内容，并请求css/js/font/img等资源，同时将结果渲染。

网页渲染过程中的网络请求



系统为什么这么慢？

同样以行业中的各类报表系统为例，要保证系统运行速度，需要在以下几个方面做优化：

1. 数据库优化，部分系统甚至未建议表单索引（钞纸机检系统本地数据存储）；
2. 服务端缓存，几乎都没做；
3. 减少资源请求频次，但SSR中的每一次请求都需要重新重新请求资源；
4. 浏览器渲染时间的消耗。比如8000行数据的报表，如何快速渲染(采用数据分页分增加数据请求次数)；

前端渲染

为了减少页面渲染中(前后台)的不必要时间消耗，前端渲染的方案开始应用，这也得益于 **ajax** 的出现。

! ajax实现了在页面 **不刷新** 的情况下前台页面同后端获取数据的能力。

延伸阅读：**JSON** 、 **ajax**

页面局部刷新

```
1 $.ajax({
2     url: './yourAPI',
3     type: 'GET',
4     dataType: 'json'
5 })
6 .done(function(data) {
7     var tbodyStr = data.map(tdData => {
8         var tdStr = tdData.map(item => `<td>${item}</td>`).join('');
9         return `<tr>${tdStr}</tr>`;
10    }).join('');
11    $('#yourTable tbody').html(tbodyStr);
12 });
```

jQuery的出现为开发者解决了页面兼容性的问题，在用户交互事件中，系统从后台获取数据并实现 **页面局部刷新**，极大提升了用户体验。此时后端在以往的基础上还需要提供 **API数据接口** 功能。

前后端分离

虽然页面局部刷新为用户的使用体验提供了一大步，但开发者的体验却未得到明显的改善，这里仅是将后端的一些职责转移到了前端，开发者需要同时维护前后两端的代码。

此后出现了后端只负责数据接口，前端负责页面样式、逻辑、数据请求等所有工作，这也让开发效率和后期的维护效率得到了改善。

前端页面的渲染——字符串模板

```
1 var html = '<p>你所想渲染的html内容</p>';  
2 $('#container').html(html);
```

js

前面通过拼凑字符串的方式又让前端又遇到了最初asp年代同样的问题——**html片断和js代码混合编排**。

问题的解决同后端解决一样，引入字符串模板。

延伸阅读：**[常见的字符串模板速度测试](#)**、**[百度eTpl](#)**

字符串模板

```
1  <!-- target: table -->
2      <table class="table table-hover table-light">
3          <thead>
4              <!-- import: thead -->
5          </thead>
6          <tbody>
7              <!-- import: tbody -->
8          </tbody>
9      </table>
10
11 <!-- target: thead -->
12     <tr class="uppercase">
13         <th>序号</th>
14         <!-- for: ${header} as ${head} -->
15         <th>${head.title}</th>
16     <!-- /for -->
17 </tr>
18
19 <!-- target: tbody -->
20 <!-- if: ${rows} == 0 -->
21 <!-- var: colspan = ${cols}+1 -->
22     <tr><td colspan=${colspan} class="text-center">未检索到 ${title} 相关数据</td></tr>
23 <!-- /if -->
24 <!-- for: ${data} as ${item}, ${index} -->
25     <!-- var: idx = ${index}+1 -->
26     <tr>
27         <td> ${idx} </td>
28         <!-- for: ${item} as ${td} -->
```

前端三大流行框架的产生

技术的进步是为了更好的解决现有的问题。

问题1：对性能的极致追求。

在字符串模板/字符串拼接的过程中，两次数据请求结果只有部分数据修改时，以前的处理方式是更新dom结点的所有数据，页面需要重新计算并渲染。

延伸阅读：[react的dom diff算法](#)、[如何理解虚拟DOM\(visual dom\)](#)

前端三大流行框架的产生

问题2：数据驱动视图

```
1 <p>数据录入测试: </p>
2 <p id="result"></p>
3 <input type="text" id="text">
```

html

```
1 $('#text').on('keyup', function() {
2     $('#result').html($(this).val());
3 });
```

js

在传统的开发中，我们需要绑定input按钮的事件。

Vue.js

```
1 <div id="app">
2   <p>{{message}}</p>
3   <input type="text" v-model="message">
4 </div>
```

html

```
1 var app = new Vue({
2   el: '#app',
3   data: {
4     message: 'hello'
5   }
6 })
```

js

延伸阅读：[Vue.js官方文档](#)

在 Vue.js 中不需要用户绑定事件，当数据变化时，视图将同步变化，这种数据驱动（data driven）的模式将开发效率和页面运行效率提高到了极致。同Vue.js 一起的框架还有 Facebook 的 react 以及 google 的 angular.js。

SPA

前面讲到每次页面刷新时，系统将重新加载所有信息，但在html中有一个叫锚点的技术，可以实现页面 **在不刷新的情况下加载路由**。

伴随着Vue.js等的发展，出现了 **SPA** (single page application,单页应用程序)技术，它的标志就是整个系统仅有index.html一个页面，所有请求均在其中完成，这样对于资源请求的优化已经做到了极致。

关于SPA及PWA(渐进式应用开发)等技术的细节，此处暂时不展开说明，接下来我们来讲述关于搜索引擎优化（SEO，Search Engine Optimization）的一些问题。

延伸阅读：**前端：将网站打造成单页面应用SPA（一）**

SEO

随着前后端分离的发展，前端页面的可见源码变得越来越少，以下是成钞信息网(包含前后台管理)在查看网页源码时html部分的内容。

```
1
2 <body>
3   <div id="app">数据加载中...</div>
4   <!-- Todo: only include in production -->
5   <script>// This service worker file is effectively a 'no-op' that will reset
6 // previous service worker registered for the same host:port combination.
7 // In the production build, this file is replaced with an actual service worker
8 // file that will precache your site's local assets.
9 // See https://github.com/facebookincubator/create-react-app/issues/2272#issuecom
10
11 self.addEventListener('install', () => self.skipWaiting());
12
13 self.addEventListener('activate', () => {
14   self.clients.matchAll({ type: 'window' }).then(windowClients => {
15     for (let windowClient of windowClients) {
```

对于搜索引擎爬虫来讲，无法读取页面的内容，这对于一些新闻网站来讲是不可接受的。

重新回到SSR

为了解决SEO的问题，react/vue.js等提出了用node.js做SSR渲染的方案，通过对原来前端代码的简单改造实现node.js的服务端渲染。

对于google,bing等搜索引擎，目前已支持SPA单页应用的爬虫抓取，百度也在计划中。

Q & A