**[优化网站设计（十九）：减少DOM元素的数量](http://www.cnblogs.com/chenxizhang/archive/2013/05/17/3083162.html)**

**前言**

网站设计的优化是一个很大的话题,有一些通用的原则,也有针对不同开发平台的一些建议。这方面的研究一直没有停止过，我在不同的场合也分享过这样的话题。

作为通用的原则，雅虎的工程师团队曾经给出过35个最佳实践。这个列表请参考**Best Practices for Speeding Up Your Web Site**<http://developer.yahoo.com/performance/rules.html>，同时，他们还发布了一个相应的测试工具Yslow <http://developer.yahoo.com/yslow/>

我强烈推荐所有的网站开发人员都应该学习这些最佳实践，并结合自己的实际项目情况进行应用。 接下来的一段时间，我将结合ASP.NET这个开发平台，针对这些原则，通过一个系列文章的形式，做些讲解和演绎，以帮助大家更好地理解这些原则，并且更好地使用他们。

**准备工作**

为了跟随我进行后续的学习，你需要准备如下的开发环境和工具

1. Google Chrome 或者firefox ，并且安装 Yslow这个扩展组件.请注意，这个组件是雅虎提供的，但目前没有针对IE的版本。
   1. <https://chrome.google.com/webstore/detail/yslow/ninejjcohidippngpapiilnmkgllmakh>
   2. <https://addons.mozilla.org/en-US/firefox/addon/yslow/>
   3. 你应该对这些浏览器的开发人员工具有所了解，你可以通过按下F12键调出这个工具。
2. Visaul Studio 2010 SP1 或更高版本，推荐使用Visual Studio 2012
   1. <http://www.microsoft.com/visualstudio/eng/downloads>
3. 你需要对ASP.NET的开发基本流程和核心技术有相当的了解，本系列文章很难对基础知识做普及。

**本文要讨论的话题**

这一篇我和大家讨论的是第十六条原则：[Reduce the Number of DOM Elements](http://developer.yahoo.com/performance/rules.html#min_dom) （减少DOM元素的数量）

在这个系列文章的前面部分，我们谈到的很多有关设计的高级别的知识（例如如何拆分内容，并行下载等等），并且大量讨论到了脚本、样式表、图片的一些优化设计。这一篇文章我们要来讨论的是页面本身的细节设计：我们应该尽可能地使得页面的DOM元素数量少一些，这样有助于减小页面体积，并且也降低了维护这份DOM树的成本。

**什么是DOM？**

好吧，如果你不太清楚这个概念，也没有什么大不了的。DOM的全称为：Document Object Model ，中文翻译过来叫文档对象模型。我们这里所探讨的DOM，其实有一个隐含的意思是指HTML DOM。关于它的定义，可以参考下面这个链接

<http://www.w3school.com.cn/htmldom/index.asp>

1. HTML DOM 定义了访问和操作 HTML 文档的标准方法。
2. DOM 以树结构表达 HTML 文档。

从上面的定义中，我们可以知道HTML文档的结构本身就是有一套规范的（例如可以有哪些节点，必须有哪些节点等等），而且对于HTML文档的访问也是有规范的（例如要想改变某个元素的位置，则需要修改left,或者top属性），这套规范就是DOM。这是由W3C确定，并且在所有主流浏览器中都共同遵守的一套标准。<http://www.w3.org/TR/DOM-Level-2-Core/introduction.html>

**什么是DOM树？**

实际上并不真的存在DOM树，这只是我们程序员对于DOM的一种理解方式。一个HTML文档，由于其独有的特性，它有且只能有一个根元素，所有其他元素都是根元素的子元素，然后子元素又可以有子元素。对于这种数据结构，为了便于构造以及日后的访问（包括查询、修改），我们会采用一种树形结构来表示它。DOM树从逻辑上说大致上像下面这样

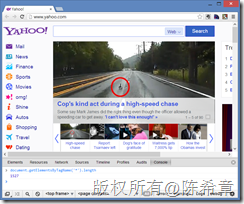
【备注】该截图来自于<http://www.w3school.com.cn/htmldom/index.asp>

如果有了上述的概念，那么对于“DOM元素应该尽量少”这条原则应该是不难理解的。问题的关键在于

1. 多少才算少
2. 如何减少

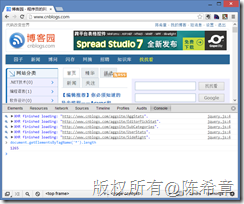
**多少才算少？**

很抱歉，这是一个没有标准答案的问题。没有谁规定我们的页面必须要少于某个数量的DOM元素。雅虎的团队当年声称他们的主页只有700个元素（对于一个门户页面来说，这个真的算很少了），但是最近我再去看这个页面，我发现目前有1527个元素。

[](http://images.cnitblog.com/blog/9072/201305/17094759-e0d1d2ec768e4f67b8af962aafcc3a65.png)  
我随意地打开另外几个门户网站（例如新浪）的主页，发现他们的元素数量就大大增加了。(而且也有很多错误）

[](http://images.cnitblog.com/blog/9072/201305/17094853-16da5ecdba1942daa8efdc86a48853bf.png)

我们也可以再来看一下博客园的主页，我发现他们的元素数量也在一个较小的级别。（1265）

[](http://images.cnitblog.com/blog/9072/201305/17094903-cff21a7e246643cab62232ef9cd04884.png)

所以，对于这个元素数量的问题，并没有什么固定的标准，应该尽可能地减小。当然，我们完全可以给自己一个小小的目标，例如1000左右？

**如何减少DOM元素的数量？**

我觉得有几个方面可以用来减少DOM元素的数量

1. 避免不正确地使用服务器控件。
2. 减少不必要的内容（并不是所有内容都必须放在页面上面的）
   * 如果数据量大，可以考虑分页，或者按需加载

**避免不正确地使用服务器控件**

这个问题被一次又一次地讨论（甚至是争论），ASP.NET给我们带来的服务器控件，从一开始诞生之日起，就充满了争议。服务器控件毫无疑问是简化了开发过程，因为通过拖拽就能实现复杂的功能。但服务器控件的代价也是相当大的（例如臃肿的代码，以及视图状态），并且从一开始就最被人诟病的是，因为服务器控件隐藏了很多细节，使得有一批网页的开发人员，只了解服务器控件，甚至连HTML的一些基础知识都不了解。

作为从ASP时代就开始做网站的人来说，包括我在内，我亲身经历了ASP.NET的整个发展过程。毋庸讳言，实际上微软也一直在改进ASP.NET。站在今天这样的时间节点，我个人给出的建议是

1. 如果能用ASP.NET MVC做的，就不要用ASP.NET Web Forms。（关于他们各自的优缺点，可以参考[这篇文章](http://msdn.microsoft.com/en-us/magazine/dd942833.aspx)。）
   * 我们不想用ASP.NET Web Forms的原因不光是不想用服务器控件，而且是希望有更好的架构，来支持大型团队和项目的开发。
2. 如果要用ASP.NET Web Forms，要慎重地使用服务器控件。尤其是一些复杂控件内部。
   * 大部分时候，我们都可以通过禁用视图状态来减小页面体积。

我们可以来看一个简单的例子。下面有一个页面，我们用一个表格来显示数据。注意，这里使用的是Repeater，而不是DataGrid或者GridView这一类更加复杂的控件。

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebApplication2.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Repeater ID="data" runat="server">

<ItemTemplate>

<tr>

<td>

<asp:Label runat="server" Text='<%# Eval("ID") %>'></asp:Label></td>

<td>

<asp:Label runat="server" Text='<%# Eval("FirstName") %>'></asp:Label></td>

<td>

<asp:Label runat="server" Text='<%# Eval("LastName") %>'></asp:Label></td>

<td>

<asp:Label runat="server" Text='<%# Eval("Company") %>'></asp:Label></td>

<td>

<asp:Label runat="server" Text='<%# Eval("Title") %>'></asp:Label></td>

</tr>

</ItemTemplate>

<HeaderTemplate>

<table border="1">

<tr>

<th>ID</th>

<th>FirstName</th>

<th>LastName</th>

<th>Company</th>

<th>Title</th>

</tr>

</HeaderTemplate>

<FooterTemplate>

</table>

</FooterTemplate>

</asp:Repeater>

</div>

</form>

</body>

</html>

后台代码很简单，我只是实例化了1000个数据，然后将其绑定而已。

using System;

using System.Linq;

namespace WebApplication2

{

public partial class Default : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack) {

//这里只是随机地绑定了1000行数据

data.DataSource = Enumerable.Range(1, 1000).Select(x =>

new

{

Id = x,

FirstName = "ares",

LastName = "chen",

Company = "microsoft",

Title = "SDE"

});

data.DataBind();

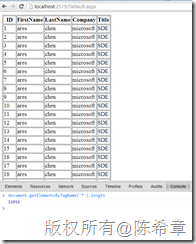
}

}

}

}

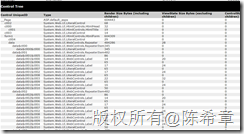
页面运行起来之后，我们可以检测到它会有11016个元素。

[](http://images.cnitblog.com/blog/9072/201305/17094938-87e5dd3a04f64afbb7f3be47364ee3ec.png)

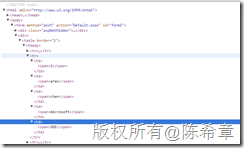
你感到诧异吗？为什么会有这么多元素呢？我们来看看页面到底是如何构造控件的吧。首先，在页面的声明语句中，加入Trace=true这个属性

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebApplication2.Default"  **Trace="true"%**>

然后在浏览器中向页面底部滚动，就可以看到一些跟踪信息

[](http://images.cnitblog.com/blog/9072/201305/17094949-3fdf0fddb7ec44fcb3cf9b0cbf698bdd.png)

我们可以很清楚地发现，为了构造得到一行数据，其实会有12个控件。其最终生成的HTML内容为

[](http://images.cnitblog.com/blog/9072/201305/17094955-c4f38c0cf63c41b09f2dcad0ad45a064.png)

那么，如何改进这一点呢？看看下面的代码

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebApplication2.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:Repeater ID="data" runat="server">

<ItemTemplate>

<tr>

<td>

<%# Eval("ID") %></td>

<td>

<%# Eval("FirstName") %></td>

<td>

<%# Eval("LastName") %></td>

<td>

<%# Eval("Company") %></td>

<td>

<%# Eval("Title") %></td>

</tr>

</ItemTemplate>

<HeaderTemplate>

<table border="1">

<tr>

<th>ID</th>

<th>FirstName</th>

<th>LastName</th>

<th>Company</th>

<th>Title</th>

</tr>

</HeaderTemplate>

<FooterTemplate>

</table>

</FooterTemplate>

</asp:Repeater>

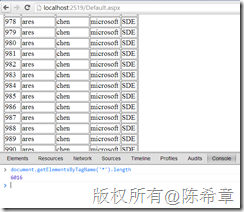
</div>

</form>

</body>

</html>

然后我们再来看页面中有多少元素呢？6016个。比刚才足足少了5000个。

[](http://images.cnitblog.com/blog/9072/201305/17094957-e4a749ea7f394355ae1c249f68963d23.png)

那么到底少了什么呢？请参考下图，对照一下前面的截图，我想你应该会明白的。

[](http://images.cnitblog.com/blog/9072/201305/17094959-da22c8c4477a42ec91cc7f16f7ca24b6.png)

现在还有6016个元素，但其实还可以进一步优化，例如将下面红色的几行去掉，并且为服务器控件禁用视图状态。（在当前这个页面中，其实只是显示数据，用不着做提交的）

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebApplication2.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

**<form id="form1" runat="server">**

**<div>**

<asp:Repeater ID="data" runat="server" EnableViewState="false">

<ItemTemplate>

<tr>

<td><%# Eval("ID") %></td>

<td><%# Eval("FirstName") %></td>

<td><%# Eval("LastName") %></td>

<td><%# Eval("Company") %></td>

<td><%# Eval("Title") %></td>

</tr>

</ItemTemplate>

<HeaderTemplate>

<table border="1">

<tr>

<th>ID</th>

<th>FirstName</th>

<th>LastName</th>

<th>Company</th>

<th>Title</th>

</tr>

</HeaderTemplate>

<FooterTemplate>

</table>

</FooterTemplate>

</asp:Repeater>

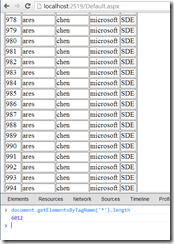
**</div>**

**</form>**

</body>

</html>

这样又可以少掉几个元素。

[](http://images.cnitblog.com/blog/9072/201305/17095007-73d8c7c2239a4934a892142dae102de2.png)

是不是跃跃欲试了呢？不要着急，我们再来谈一个问题：这个页面上的1000行数据真的有必要进行一次性的加载和显示吗？答案通常是否定的，因为浏览器的尺寸本来就是有限的，对于用户来说，并不可能一次性阅读1000行数据。所以，我们需要了解如何通过分页或者按需加载的技术，来减少页面DOM元素的数量，提高加载和维护的效率。

**使用分页加载内容**

分页就是说，虽然数据很多，但我每次只显示一部分（例如20行），用户如果想看其他的行，则通过相应的按钮来导航切换。我们可以将上面的例子稍微改动一下

<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebApplication2.Default" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<asp:Repeater ID="data" runat="server" EnableViewState="false">

<ItemTemplate>

<tr>

<td><%# Eval("ID") %></td>

<td><%# Eval("FirstName") %></td>

<td><%# Eval("LastName") %></td>

<td><%# Eval("Company") %></td>

<td><%# Eval("Title") %></td>

</tr>

</ItemTemplate>

<HeaderTemplate>

<table border="1">

<tr>

<th>ID</th>

<th>FirstName</th>

<th>LastName</th>

<th>Company</th>

<th>Title</th>

</tr>

</HeaderTemplate>

<FooterTemplate>

</table>

</FooterTemplate>

</asp:Repeater>

**<a href='default.aspx?p=<%= CurrentPageIndex+1 %>'>下一页</a>**

</body>

</html>

作为演示目的，这里只是添加了一个链接，点击可以进入下一页。服务端代码也需要稍作修改

using System;

using System.Linq;

namespace WebApplication2

{

public partial class Default : System.Web.UI.Page

{

public int CurrentPageIndex { get; set; }

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

//这里检测是否要显示特定页面

var p = Request.QueryString["p"];

var index = 0;

if (string.IsNullOrEmpty(p) || !int.TryParse(p, out index))

CurrentPageIndex = 1;

else

CurrentPageIndex = index;

//这里只是随机地绑定了20行数据

data.DataSource = Enumerable.Range((CurrentPageIndex - 1) \* 20 + 1, 20).Select(x =>

new

{

Id = x,

FirstName = "ares",

LastName = "chen",

Company = "microsoft",

Title = "SDE"

});

data.DataBind();

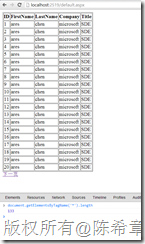
}

}

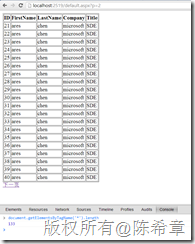
}

}

如果用户没有提供p这个参数（或者是不正确的值），则默认显示第一页。如果提供了，则显示他想要的页面。每页显示20行。这个页面显示出来，只需要133个元素。如下图所示

[](http://images.cnitblog.com/blog/9072/201305/17095009-fc7efcff18e140239c81d9fa002a8113.png)

当用户点击“下一页”的时候，实际上是一个新的请求。而且同样只需要133个元素。

[](http://images.cnitblog.com/blog/9072/201305/17095014-455d51abaf7f48ad95521a4234650f2c.png)

对于分页，还有一些细节直接研究，并且也有一些现成的插件可以使用。例如 <http://www.bing.com/search?setmkt=en-US&q=jquery+paging>

**按需加载内容**

分页可以很好地解决大数据的问题。但由于分页需要用户额外的点击操作，对于用户来说，可能不是很方便。为了进一步提高用户体验，我们是否能做到：

1. 默认显示20行
2. 当用户往下滚动的时候，根据需要再显示另外20行
   * 这是一个循坏

现实世界中，有很多这样的例子，例如本文前面提到的[雅虎主页](http://www.yahoo.com/)，目前就是这样做的。还有国内比较火的[新浪微博](http://weibo.com/)，也是这样做的。

按需加载！听起来很有点意思吧，由于讲解这个做法，相对来说篇幅较大。我希望大家可以自行参考一下下面这篇文章

Load Data From Server While Scrolling Using jQuery AJAX

<http://www.codeproject.com/Articles/239436/Load-Data-From-Server-While-Scrolling-Using-JQuery>

按需加载与分页是有根本区别的：分页之后页面的体积能够固定下来，而按需加载的做法中，页面体积是动态添加，而且也正因为是动态添加到，每次添加的内容有限，所以给用户的影响很小。

**正确地使用JQuery**

本文的最后部分，我要特别说明：我在之前的很多演示中都用到过jquery。（目前为止，它确实也是最好的一个javascript库，没有之一），但是对于jQuery，越来越多的人在学习，越来越多的人在滥用。这确实也是一个趋势。

关于如何正确地使用jQuery,国外和国内都有热心的网友做了总结，请参考

1. <http://www.cnblogs.com/huyh/archive/2009/03/30/1422976.html> （国内的，翻译文档）
2. <http://www.cnblogs.com/huyh/archive/2009/03/31/1425430.html> （国内的，翻译文档）
3. <http://net.tutsplus.com/tutorials/javascript-ajax/10-ways-to-instantly-increase-your-jquery-performance/> （国外的）