**[优化网站设计（三）：对资源添加缓存控制](http://www.cnblogs.com/chenxizhang/archive/2013/04/30/3052440.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**](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**

**缓存！这是一条多么重要的原则。**几乎所有的网站优化的书籍或者文章中都会提到这个原则，而且目前在运行的一些网站都或多或少地使用到了这个技术。这个原则的相关概念可以参考这里：<http://developer.yahoo.com/performance/rules.html#expires>

我们还是以博客园的主页为例，通过简单的监控就能发现，他们大量地使用了缓存的功能（针对不同资源，缓存的策略可能略有不同，请注意观察max-age的值，以秒为单位，有兴趣的同学可以计算一下）

[](http://images.cnitblog.com/blog/9072/201304/30212443-9e0755c54bdd4d44b4df7869fc9b7a49.png)

接下来，我会从几个方面，和大家来谈谈缓存的问题

**1.缓存的概念**

缓存是服务器与客户端（或者中间的代理服务器）之间的一种约定，利用缓存可以明显地减少重复从服务器下载内容的次数，这样就可以极大地提高吞吐量以及响应速度。在HTTP 1.1协议中，对于缓存有明确的，详细的说明：<http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html>

由于缓存如此重要，所以对于很多浏览器而言（尤其是现代浏览器），他们默认就会尝试对静态内容进行缓存。(**这里先卖一个关子，大家觉得默认情况下它会缓存多长时间呢**？）

下图可以很好地揭示缓存的作用

[](http://images.cnitblog.com/blog/9072/201304/30212449-d9076a212c474a5a8cf263705b626d81.png)

这是我两次访问一个简单的页面的情形，第一次（我用颜色选中的8个请求）的时候，因为没有缓存，所以所有的内容资源（动态的和静态的）都需要下载。浏览器会根据情况对其进行缓存（通常就是对于静态内容它会有一个缓存的策略，后面再细述）。但是第二次访问，就完全不一样了，只有第1个请求是需要下载内容的（这是一个动态页面），其他的7个内容都是无需下载的（你可以看到状态码是304，而且Body都是0）

**2.静态资源的缓存及其设置**

默认情况下，浏览器（至少绝大多数现代浏览器）都会对网站中的静态内容进行缓存。常见的静态内容包括

1. HTM,HTML文件
2. JAVASCRIPT文件
3. CSS文件
4. 图片文件

如果服务器端不做任何的设置，那么默认情况下它缓存多长时间呢？答案是：**可能很久**，这个取决于两个因素

1. 浏览器缓存中的这个文件是否被清理（可能是用户手工地进行清理，有的公司也可能会设置统一的策略自动在浏览器关闭的时候清理）
2. 服务器对应的文件是否有更新

这里就要讲到那个304的状态码了. 这个状态码的意思是：Not Modified（未更改）。为什么会产生这样的状态码呢？其实可以通过下面三个截图来理解清楚

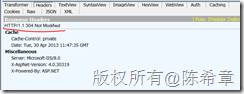
第一次请求某静态资源的时候，服务器会在返回内容（状态码为200）的同时，包含一个特殊的Header，叫做Last-Modified ,这个Header会记录在服务器端该文件最后修改的时间。如下图所示

[](http://images.cnitblog.com/blog/9072/201304/30212450-96f278d268a4402b8566e0643c941a92.png)

然后，浏览器会将此文件缓存起来。

接下来如果需要第二次访问这个文件，浏览器发起的请求中，也会包含一个特殊的Header

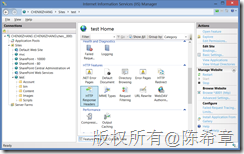
[image](http://images.cnitblog.com/blog/9072/201304/30212450-fe0d341bc1794dfdb61eff00c0931682.png)  
这个Header的意思是说，要检查从这个时间后是否有修改。如果该文件没有修改过，则服务器就会返回304这个状态码，并且不会返回任何内容。

[](http://images.cnitblog.com/blog/9072/201304/30212451-af6cba2929834d3299c7a3f3cbaa63a1.png)

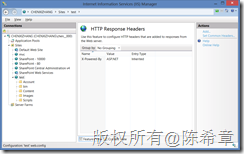
浏览器收到了304这个状态码的话，就会使用自己已经缓存的那个版本进行呈现。

我们当然也可以静态资源的缓存策略进行人工的干预，这个可以通过两个途径来实现

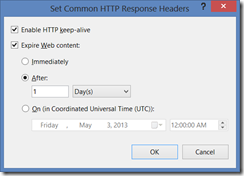
1. **通过IIS 管理界面**。选择某个站点，然后在右侧的功能面板中，选择“Http Response Header”，如下图所示

[](http://images.cnitblog.com/blog/9072/201304/30212505-1ef9fa996625474b8c0f2ca8f0f9aa19.png)

然后在功能页面中，点击右侧的“Set Common Header…” 这个Action

[](http://images.cnitblog.com/blog/9072/201304/30212510-d49cc741eb444bec9d87301dde876136.png)

然后在弹出的对话框中设置Expire Web Content的选项

[](http://images.cnitblog.com/blog/9072/201304/30212511-cb5c22facd314fbe898682a43b4bc414.png)

这里默认会有三个策略：立即过期，在一个相对时间范围内过期，在一个绝对时间后过期。你可以根据自己的需求进行设置。

【备注】这些设置仅仅影响静态资源。

1. **直接修改web.config文件**

上面提到的通过IIS管理界面对静态资源配置到期时间（其实也就是指一个缓存的时间），这个做法适合给管理员使用。作为开发人员，如果你希望自己来控制这些选项，那么可以直接在网站的配置文件（web.config）中添加相关的配置即可。

<system.webServer>

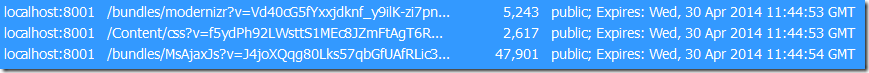
<staticContent>

<clientCache cacheControlMode="UseMaxAge" cacheControlMaxAge="1.00:00:00" />

</staticContent>

</system.webServer>

需要注意的是，在之前的截图中，有些特殊的脚本，它们的缓存时间是不受这个配置影响的。

[](http://images.cnitblog.com/blog/9072/201304/30212512-ab0a31b25bd64ae1bdcaff55d8091984.png)

他们是默认缓存**一年**的.这些特殊的脚本（或者样式表）其实是多个文件的组合，如果对这一点不太了解，请参考我之前的一篇文章

[优化网站设计（一）：减少请求数](http://www.cnblogs.com/chenxizhang/archive/2013/04/29/3050839.html)

**3.动态资源的缓存及其设置**

我们上面讨论到了静态资源的缓存及其设置，但是对于现在的大部分网站来说，光有静态资源是远远不够的，我们会有很多动态资源，典型的就是动态页面（例如aspx页面），那么对于这种动态资源，是否也有可能进行一定的缓存呢？

答案是肯定的，但对于动态资源的缓存细节相当多，恐怕要超出本文的范畴。我强烈推荐大家详细阅读MSDN中的这篇文档 ：

<http://msdn.microsoft.com/en-us/library/aa478965.aspx> （ASP.NET Caching: Techniques and Best Practices）

我这里为大家总结一下对动态资源进行缓存的几种做法

1. 页面缓存，在页面中使用[OutputCache](http://msdn.microsoft.com/en-us/library/vstudio/hdxfb6cy(v=vs.100).aspx)，适合于对整个页面所有内容做缓存
2. 片段缓存，在用户控件中使用[OutputCache](http://msdn.microsoft.com/en-us/library/vstudio/hdxfb6cy(v=vs.100).aspx)，适合于对页面的一部分（通常封装为一个用户控件）内容做缓存
   * 在MVC中，同样可以在Action上面使用[OutputCache](http://msdn.microsoft.com/en-us/library/dd492556(v=VS.108,d=hv.2).aspx)这个Attribute来定义缓存设置
3. 数据缓存，在代码中访问[Cache](http://msdn.microsoft.com/en-us/library/8bx0tk8a.aspx)对象，适合对多个页面中可能会用到的共享数据做精细的缓存控制

值得注意的是，这三种做法并非是互斥的，在实际的应用中，他们会相互结合起来使用。

另外，除了给页面或者控件设置OutputCache（为了灵活起见，建议结合CacheProfile)之外，如果确实某些缓存的设置需要动态决定，也可以采用如下的方式来实现同样的效果

Response.Cache.SetExpires(DateTime.Now.AddSeconds(60));

Response.Cache.SetCacheability(HttpCacheability.Public);

Response.Cache.SetValidUntilExpires(false);

Response.Cache.VaryByParams["Category"] = true;

if (Response.Cache.VaryByParams["Category"])

{

//...

}

关于Response.Cache的所有属性和操作，有兴趣的可以参考 <http://msdn.microsoft.com/EN-US/library/system.web.httpcachepolicy.aspx>

【备注】本文之前提到的Bundle默认设置为1年的过期时间就是通过这种方式来实现的。

**4.缓存的反面**

缓存是一种很有用的技术，几乎所有人都知道它的好处。现在的开发平台都比较强大，让我们可以有比较简单的方式来实现缓存。

但是，缓存有它的一些代价，或者说有反面的一些问题需要考虑。典型的问题在于

1. **版本控制**。
   * 版本是一个问题，不是吗？典型的问题是：如果你缓存了一个页面，但这个页面的数据其实是发生了变化，那么如果你还是以缓存的版本给用户呈现，是否有问题呢？
   * 如何解决版本问题呢？这里会有一个缓存依赖的概念需要了解，
     + <http://msdn.microsoft.com/en-us/library/system.web.caching.cachedependency.aspx>
2. **容量问题**。
   * 静态资源通常是缓存在客户端的（除非服务器端做特殊设置，IIS 7开始支持在服务器端——甚至内核模式——中缓存内容），他们通常对服务器影响不大。
   * 但动态资源的缓存，通常是缓存在服务器端的（或者客户端和服务器端各有一份），所以需要占用服务器的内存空间。
   * 如果不加以限制（或者不做周全的考虑），则很可能会因为缓存了过多的内容，而导致服务器的内存出现争用的问题。
     + 这里留一个问题，请问ASP.NET应用程序的缓存功能，默认最多可以使用多少服务器内存？
   * 这里所谓的周全考虑，是指我们需要对缓存的必要性进行评估。应该只缓存那些确实有必要缓存的内容。这个说来容易，实际上做起来却是不那么容易的。
     + 我们需要评估诸如缓存命中率这样一些性能指标来评估缓存的设置是否恰当
       - 有兴趣的同学可以参考 <http://msdn.microsoft.com/zh-cn/library/fxk122b4(v=vs.100).aspx>