第 3 章

规则 3——添加 Expires 头

Rule 3: Add an Expires Header

在设计 Web 页面的时候，首次访问的响应时间并不是唯一需要考虑的。如果是这样的话，我们可以将规则 1 发挥到极致，并且不在页面上放置任何图片、脚本和样式表。然而，我们都知道，图片、脚本和样式表能够加强用户体验，尽管这意味着页面需要花更长的时间进行加载。这一章介绍的规则 3 展示了如何配置组件，使其能够最大化地利用浏览器的缓存能力来改善页面的性能。

今天的 Web 页面都包含了大量的组件，并且其数量在不断增长。页面的初访者会进行很多 HTTP 请求，但通过使用一个长久的 `Expires` 头，使这些组件可以被缓存。这会在后续的页面浏览中避免不必要的 HTTP 请求。长久的 `Expires` 头最常用于图片，但应该将其用在**所有**组件上，包括脚本、样式表和 Flash。很多顶级网站现在都还没有做到这一点。在这一章里，我将指出这些网站并说明为什么他们的页面没有像应该的那样快。添加长久的 `Expires` 头会带来额外的开发成本，如“修订文件名”一节中介绍的那样。

Expires 头

Expires Header

浏览器（和代理）使用缓存来减少 HTTP 请求的数量，并减小 HTTP 响应的大小，使 Web 页面加载得更快。Web 服务器使用 `Expires` 头来告诉 Web 客户端它可以使用一个组件的当前副本，直到指定的时间为止。HTTP 规范中简要地称该头为“在这一日期/时间之后，响应将被认为是无效的”。它在 HTTP 响应中发送。

```
Expires: Thu, 15 Apr 2010 20:00:00 GMT
```

这是一个**有效期非常长久的 Expires 头**（*far future Expires header*），它告诉浏览器该响应的有效性持续到 2010 年 4 月 15 日为止。如果为页面中的一个图片返回了这个头，浏览器在后续的页面浏览中会使用缓存的图片，将 HTTP 请求的数量减少一个。请参见绪言 B，复习一下 Expires 头和 HTTP。

Max-Age 和 mod_expires
Max-Age and mod_expires

在解释缓存如何很好地改善传输性能之前，需要提及除了 Expires 头之外的另一种选择。HTTP 1.1 引入了 Cache-Control 头来克服 Expires 头的限制。因为 Expires 头使用一个特定的时间，它要求服务器和客户端的时钟严格同步。另外，过期日期需要经常检查，并且一旦未来这一天到来了，还需要在服务器配置中提供一个新的日期。

换一种方式，Cache-Control 使用 max-age 指令指定组件被缓存多久。它以秒为单位定义了一个更新窗。如果从组件被请求开始过去的秒数少于 max-age，浏览器就使用缓存的版本，这就避免了额外的 HTTP 请求。一个长久的 max-age 头可以将刷新窗设置为未来 10 年。

```
Cache-Control: max-age=315360000
```

使用带有 max-age 的 Cache-Control 可以消除 Expires 的限制，但对于不支持 HTTP 1.1 的浏览器（尽管这只占你的访问量的 1%以内），你可能仍然希望提供 Expires 头，你可以同时指定这两个响应头——Expires 和 Cache-Control max-age。如果两者同时出现，HTTP 规范规定 max-age 指令将重写 Expires 头。然而，如果你很尽职尽责，你仍然需要担心 Expires 带来的时钟同步和配置维护问题。

幸运的是，mod_expires Apache 模块（*http://httpd.apache.org/docs/2.0/mod/mod_expires.html*）使你在使用 Expires 头时能够像 max-age 那样以相对的方式设置日期。这通过 Expires-Default 指令来完成。在下面的例子里，图片、脚本和样式表的过期时间被设计为自请求开始的 10 年之后：

```
<FilesMatch "\.(gif|jpg|js|css)$">
 ExpiresDefault "access plus 10 years"
</FilesMatch>
```

时间可以以年、月、周、日、小时、分钟、秒为单位来设置。它同时向响应中发送 Expires 头和 Cache-Control max-age 头。

```
Expires: Sun, 16 Oct 2016 05:43:02 GMT
Cache-Control: max-age=315360000
```

实际的过期日期根据何时接到请求而变，但是即便如此，它永远都是 10 年以后。由于 `Cache-Control` 具有优先权并且明确指出了相对于请求时间所经过的秒数，时钟同步问题就被避免了。不用担心固定日期的更新，他在 HTTP 1.0 浏览器中也能够使用。跨浏览器改善缓存的最佳解决方案就是使用由 `ExpiresDefault` 设置的 Expires 头。

对十大网站的调查（参见表 3-1）表明其中的七个网站使用了这种方法，五个网站同时使用了 `Expires` 头和 `Cache-Control max-age` 头。一个网站只是用了 `Expires` 头，还有一个网站只是用了 `Cache-Control max-age` 头。悲哀的是，还有三个网站没使用这种方法。

表 3-1：Expires 和 max-age 的使用情况

网站	Expires	max-age
http://www.amazon.com		
http://www.aol.com	✓	✓
http://www.cnn.com		
http://www.ebay.com	✓	✓
http://www.google.com	✓	
http://www.msn.com	✓	✓
http://www.myspace.com		✓
http://www.wikipedia.org	✓	✓
http://www.yahoo.com	✓	✓
http://www.youtube.com		

空缓存 VS 完整缓存
Empty Cache vs. Primed Cache

只有在用户已经访问过你的网站之后，长久的 `Expires` 头才会对页面浏览产生影响。当用户第一次访问你的网站时，它不会对 HTTP 请求的数量产生任何影响，此时浏览器的缓存是空的。因此，其性能的改进取决于用户在访问你的页面时是否有完整缓存。你的访问流量几乎都来自于那些有完整缓存的用户。使你的组件可缓存能够改善这些用户的响应时间。

当我说“空缓存”或“完整缓存”时，我指的是与你的页面相关的浏览器缓存的状态。如果你的页面中的组件没有放在缓存中，则缓存为“空”。浏览器的缓存可能包含来自其他网站的组件，但这对你的页面没有帮助。反之，如果你的页面中的可缓存组件都在缓存中，则缓存是“完整的”。

空缓存或完整缓存页面浏览的数量取决于 Web 应用程序的本质。一个类似“每日一词”的网站对典型用户来说，每个会话可能只产生一个页面浏览。很多原因会导致当用户下一次访问网站时，“每日一词”的组件可能会不在缓存中：

- 尽管他很渴望掌握更多的词汇，他仍然可能只是每周或者每月访问该页面一次，而不是每天一次。
- 在最近一次访问之后，用户可能手动清空了他的缓存。
- 用户可能访问了太多其他的网站以致缓存已满，“每日一词”的组件被移出缓存。
- 浏览器或杀毒软件可能会在关闭浏览器时清空缓存。

由于每次会话只产生一个页面浏览，“每日一词”的组件不大可能会被缓存，因此带有完整缓存的页面浏览所占的百分比是很低的。

另一方面，在一个旅游网站或 Email 网站中可能每个用户会话能产生多次页面浏览，完整缓存的页面浏览数量就会多一些。在这种情况下，很多页面会在浏览器缓存中发现你的组件。

我们在 Yahoo!对此进行了测量，发现**拥有完整缓存的、每天至少访问一次的唯一用户数占*40%~60%***，取决于 Yahoo!的内容。同样的研究表明**带有完整缓存的页面浏览数量为*75%~85%***（注 1）。注意第一个统计测量的是“唯一用户”，而第二个测量的是“页面浏览”。拥有完整缓存的页面浏览所占的百分比要高于拥有完整缓存的唯一用户，因为很多 Yahoo!功能在每个会话中会接收到多次页面浏览。用户在一天中只会有一次使用空缓存，然后很多后续页面访问都将拥有完整缓存。

这些浏览器缓存统计数据解释了为什么优化完整缓存体验是那么的重要。我们希望 40%~60%的用户和 75%~85%的页面浏览的完整缓存能得到优化。这一百分比对你的网站来说有可能不同，但只要用户通常每个月至少访问你的网站一次，或每会话能产生多次页面浏览，这一统计值就会差不多。通过使用长久的 `Expires` 头可以增加被浏览器缓存的组件的数量，并在后续页面浏览中重用它们，而无需通过用户的 Internet 连接发送一个字节。

不仅仅是图片
More Than Just Images

为图片使用长久的 `Expires` 头非常之普遍，但这一最佳实践不应该仅限于图片。长久的 `Expires` 头应该包含任何不经常变化的组件，包括脚本、样式表和 Flash 组件。但是，

注 1：Tenni Theurer，“Performance Research, Part2: Browser Cache Usage - Exposed!”，*http://yuiblog.com/blog/2007/01/04/performance-research-part-2*。

HTML 文档不应该使用长久的 Expires 头，因为它包含动态内容，这些内容在每次用户请求时都将被更新。

理想情况下，页面中的所有组件都应该具有长久的 Expires 头，并且后续的页面浏览中只需为 HTML 文档进行一个 HTTP 请求。当文档中的所有组件都是从浏览器缓存中读取出来时，响应时间会减少 50%或更多。

我调查了美国十大 Internet 网站，并记录了有多少图片、脚本和样式表使用了 Expires 或 Cache-Control max-age 头并设置了至少 30 天以上。如表 3-2 所示，但这看起来并不好。这里统计了三类组件——图片、样式表和脚本。表 3-2 展示了可缓存 30 天以上的组件和对应的每种类型组件的总数。我们来看一下这些网站在实践缓存组件时采用的时间范围：

- 5 个网站使其大部分**图片**可缓存 30 天以上。
- 4 个网站使其大部分**样式表**可缓存 30 天以上。
- 2 个网站使其大部分**脚本**可缓存 30 天以上。

表 3-2：带有 Expires 头的组件

网站	图片	样式表	脚本	Last-Modified Δ 中值
http://www.amazon.com	0/62	0/1	0/3	114 天
http://www.aol.com	23/43	1/1	6/18	217 天
http://www.cnn.com	0/138	0/2	2/11	227 天
http://www.ebay.com	16/20	0/2	0/7	140 天
http://froogle.google.com	1/23	0/1	0/1	454 天
http://www.msn.com	32/35	1/1	3/9	34 天
http://www.myspace.com	0/18	0/2	0/2	1 天
http://www.wikipedia.org	6/8	1/1	2/3	1 天
http://www.yahoo.com	23/23	1/1	4/4	-
http://www.youtube.com	0/32	0/3	0/7	26 天

表 3-2 中的整体百分比指出，所有组件中 74.7%或者是不可缓存的，或者可缓存但时间不超过 30 天。一种可能的解释是，这些组件是不应该缓存的。例如像 *cnn.com* 这样的新闻网站，其 138 个图片 0 个可缓存，可能是有太多的新闻图片，由于需要更新，这些图片会经常刷新，而不是缓存在用户的浏览器中。如果组件是因为经常变化而不被缓存，我们期望看到很近的 Last-Modified 日期。

表 3-2 展示了所有未缓存组件的 Last-Modified 增量（当前日期和 Last-Modified 日期之间的差）中值。在这里 *cnn.com* 的 Last-Modified 增量中值是 227 天。一半的未缓存组件 227 天以来都没有更改过，因此图片的刷新并不是问题所在。

这也是 Yahoo!过去的情况。过去，Yahoo!没有任何可缓存的脚本、样式表和图片。不缓存这些组件背后的逻辑是，用户应该每次都请求它们以便获得更新，因为它们经常改变。然而，在发现实际上这些文件改变得不那么频繁之后，我们意识到将它们缓存起来可以得到更好的用户体验。尽管存在着额外的开发成本，Yahoo!选择将它们做成可缓存的，这将在下一部分介绍。

修订文件名
Revving Filenames

如果我们将组件配置为可以由浏览器代理缓存，当这些组件改变时用户如何获得更新呢？当出现了 `Expires` 头时，直到过期日期为止一直会使用缓存的版本。浏览器不会检查任何更新，直到过了过期日期。这也是为什么使用 `Expires` 头能够显著地减少响应时间——浏览器直接从硬盘上读取组件而无需生成任何 HTTP 流量。因此，即使在服务器上更新了组件，已经访问过网站的用户也不大可能获取最新的组件（因为前一个版本已经在他们的缓存中了）。

为了确保用户能获取组件的最新版本，需要在所有 HTML 页面中修改组件的文件名。Mark Nottingham 的 Web 文章“Caching Tutorial for Web Authors and Webmasters”称：

> 最有效的解决方案是修改其所有链接，这样，全新的请求将从原始服务器下载最新的内容。

取决于你如何构造 HTML 页面，这项工作可能很轻松也可能很痛苦。如果你使用 PHP、Perl 等动态语言生成 HTML 页，一种简单的解决方案就是为所有组件的文件名使用变量。使用这种方法，在页面中更新文件名只需要简单地在某个地方修改变量。在 Yahoo!我们经常将这一步作为生成过程的一部分——将版本号嵌在组件的文件名中（例如 *yahoo_2.0.6.js*），而且在全局映射中修订过的文件名会自动更新。嵌入版本号不仅可以改变文件名，还能在调试时更容易地找到准确的源代码文件。

示例
Examples

下面两个示例演示了使用长久的 `Expires` 头能够得到的性能改进。这两个示例包含相同的组件——六个图片、三个脚本和一个样式表。在第一个示例中，这些组件**没有**使用长久的 `Expires` 头，而第二个示例使用了。

无 Expires 的示例

http://stevesouders.com/hpws/expiresoff.php

长久的 Expires 的示例

http://stevesouders.com/hpws/expireson.php

添加长久的 `Expires` 头可以将后续页面浏览的响应时间从 600 毫秒降低到 260 毫秒，这是在 900Kbps 的 DSL 上测试的，**减少了** *57%*。页面中的组件越多，响应时间改善得越多。如果你的页面平均超过 6 个图片、3 个脚本和 1 个样式表，页面的速度提升就会超过这个例子中的 57%。

节省下来的这些时间究竟是从哪里来的呢？我在前面曾提到过，一个具有长久 `Expires` 头的组件将会被缓存，在后续请求时浏览器直接从硬盘上读取它，避免了一个 HTTP 请求。然而，我并没有介绍相反的情况。如果一个组件**没有**长久的 `Expires` 头，它仍然会存储在浏览器的缓存中。在后续请求中，浏览器会检查缓存并发现组件已经过期（HTTP 术语称之为“陈旧”）。为了提高效率，浏览器会向原始服务器发送一个**条件 GET 请求**（*Conditional Get Request*）。请参见绪言 B 中的示例。如果组件没有改变，原始服务器可以免于发送整个组件，而是发送一个很小的头，告诉浏览器可以使用其缓存的组件。

这些条件请求加起来，就是节省的时间。很多时候，正如我们在十大网站中看到的那样，组件并没有更改，而浏览器总是从磁盘上读取它们。通过使用 `Expires` 头来避免额外的 HTTP 请求，可以减少一半的响应时间。

为组件添加长久的 Expires 头