Web开发人员需知的Web缓存知识_Web开发_酷勤网

什么是Web缓存，为什么要使用它？

Web缓存游走于服务器和客户端之间。这个服务器可能是源服务器（资源所驻留的服务器[Add]），数量可能是1个或多个；这个客户端也可能是1个或多个。Web缓存就在服务器-客户端之间搞监控，监控请求，并且把请求输出的内容（例如html页面、 图片和文件）（统称为副本）另存一份；然后，如果下一个请求是相同的URL，则直接请求保存的副本，而不是再次麻烦源服务器。

使用缓存的2个主要原因：

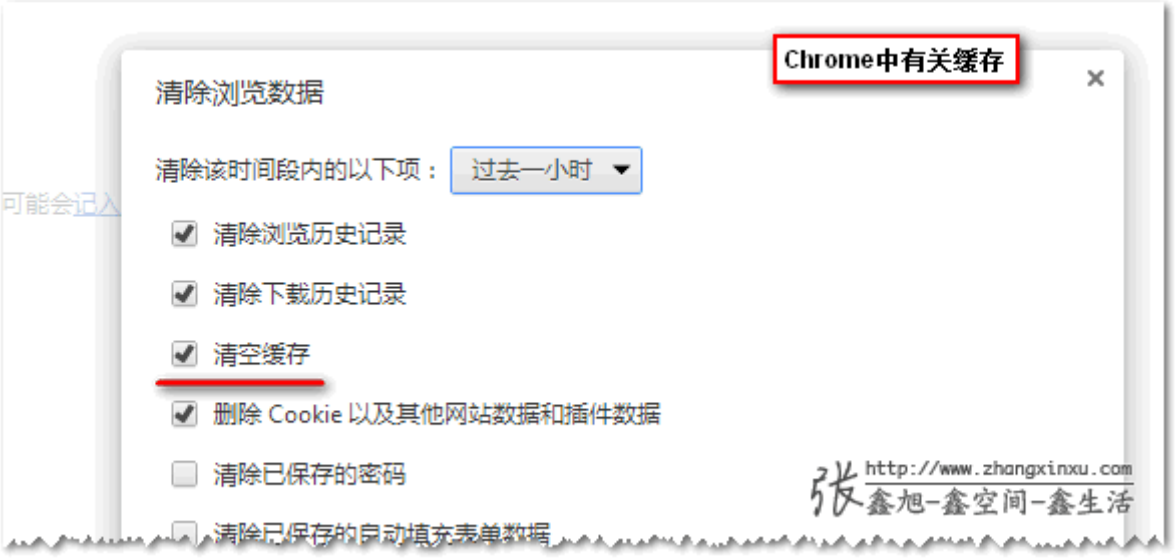
- 降低延迟：缓存离客户端更近，因此，从缓存请求内容比从源服务器所用时间更少，呈现速度更快，网站就显得更灵敏。

- 降低网络传输：副本被重复使用，大大降低了用户的带宽使用，其实也是一种变相的省钱（如果流量要付费的话），同时保证了带宽请求在一个低水平上，更容易维护了。

Web缓存的类型

1. 浏览器缓存

在任何现代浏览器上(如IE, FireFox, Chrome)折腾清除隐私数据（//zxx: 原文说的是首选项，显然out了，这里有改动）的对话框，你很可能会注意到“缓存”这个设置项。



浏览器会在你的硬盘上专门开辟一个空间专门为你存储资源副本。浏览器缓存的工作规则很简单：检查以确保副本是最新的，通常只要一次会话（就是当前浏览器调用的这次[N]）。

浏览器缓存在用户触发“后退”操作或点击一个之前看过的链接的时候很管用。同样，如果你在网站上访问同一张图片，该图片可以从浏览器缓存中调出并几乎立即显现出来。

2. 代理服务器缓存

Web代理服务器使用同样的缓存原理，只是规模更大。代理以同样的方式服务千万用户，大公司和ISP(Internet Server Provider, Internet服务提供商[Add])经常在他们的防火墙或者单独的设备（也被称为中介

(intermediaries)）上架设代理缓存。

由于代理服务器缓存并非客户端或者源服务器的一部分，而是处于网络中，请求需要以某种方式路由到它们。一种方法是手动设置，告诉浏览器的你常用的代理服务器(//zxx：翻墙的时候常用的)，另外就是使用拦截。拦截代理(Interception proxies)把Web请求根据自己的底层网络重定向，因此，客户端无需配置，甚至都不需要知道它们。//zxx：维基百科上提供的几种检测拦截代理服务器存在的方法[add]，您若有兴趣，可以点击这里查看。

代理缓存属于一种共享缓存；往往有大量的用户使用，因此，其在降低延时和网络流量上很有用，毕竟每个副本都被大量重用。//zxx：这里我有疑问：就算是放在代理服务器上，每次获取还是要通过网络的啊，如何降低了网络流量呢？希望谁可以帮忙解惑下。

3. 网关缓存
也被称为“反向代理缓存”或“替代缓存”。网关缓存同样是起中介作用的，不过不是（素不相识、不曾谋面的[Add]）网络管理员部署的，而多半是网站管理员（公司专门的运维工程师、或UED或程序组某人[Add]）他们自己部署，这样更容易扩展与维护。

可以有多种方法把请求路由到网关缓存，但通常使用某种形式的负载均衡器①，使它们中的一个或多个看起来像是源服务器。内容分发网络②(CDNs)为整个网络（或部分）分配网关缓存，然后把这些缓存卖给需要的网站。Speedera③和Akamai④就是代表性的网络内容发布商。

> ①负载均衡器：是一种采用各种分配算法把网络请求分散到一个服务器集群中的可用服务器上去，通过管理进入的Web数据流量和增加有效的网络带宽，从而使网络访问者获得尽可能最佳的联网体验的硬件设备。
>
> ②内容分发网络：即CDN，基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定。通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络，CDN系统能够实时地根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。其目的是使用户可就近取得所需内容，解决 Internet 网络拥挤的状况，提高用户访问网站的响应速度。
>
> ③Speedera：是一家全球性的内容服务提供商，它与北美、欧洲以及亚太地区的1000多家大型运营商都有联系，并为那些不想在自己服务器上寄存内容的公司提供软件下载、媒体及其它服务管理等业务。05年的时候被下面要介绍的Akamai以$130m的价格给收购了。
>
> ④Akamai：美国Akamai是国际上最大的CDN服务商，它巨大的网络分发能力在峰值时可达到15Tbps。Akamai公司是为数不多的旨在消除Internet瓶颈和提高下载速度的几家新公司之一，是一个致力于网络交通提速的”内容发布”公司，是波士顿高技术区最卓越的新兴企业之一。Akamai公司向全球企业提供发送互联网内容，汇流媒体和应用程序的服务（目前，该公司为15个国家的企业管理着8000多台服务器）。1998年，丹尼尔。L和麻省理工学院的一些研究人员一起创立了这家公司，他在麻省理工学院的硕士论文构成了Akamai公司最初的”自由流”（Freeflow）技术的核心。

本教程重点在浏览器和代理缓存，尽管有些信息对网关缓存感兴趣的人也适用。

Web缓存无害吗？为什么要鼓励缓存？

Web缓存是互联网中最容易被误解的技术之一。网站管理员特别希望知道网站的一举一动，比方说多少人访问啦，访问时间啊什么的，而缓存会“隐藏”他们的用户，他们就无从得知到底谁访问了这个站点。

捡了芝麻丢西瓜，自认为放弃缓存可以精确跟踪用户，实际上，互联网中有太多的变数，想精确得到一张用户查看网站的图片？没那么简单的，亲！如果你很重视这个问题，恭喜你，本文正好提供了解决之道，即保证缓存友

好，同时又能获得统计。

另外需要注意的是，缓存的内容都是旧的过时的。因此，如何准确更新就成了一个问题。不过不要担心，本文会向你展示如何配置服务器，让缓存就像你的女仆——随便调教。

CDN算是个挺有意思的技术，不同于代理缓存，CDN的网关缓存和被缓存的Web站点的利益是一致的，因此，上面提到的问题对于CDN而言是没有的。不过，即使你使用了CDN，你仍要顾虑下游的代理和浏览器缓存。

以上为缓存可能的“糟粕”，那他好的地方呢？缓存可以让你的Web站点加载更快，让你的服务器和互联网链接间负担更小。这种差异会导致一些类似质的变化，一个网站要几秒钟才能加载出来，而另外一个充分发挥缓存的优势，几乎瞬间显示。用户自然更喜欢那个加载迅速的站点，访问也更多。

再说个现实示例，许多大型互联网公司花费了数百万??美元，在世界各地设立服务器集群来复制他们的内容，以使其尽可能快被他们的用户访问。缓存为你做同样的事情，而且他们更接近最终用户。最重要的是，你不要花银子。

实际上呢，无论你喜欢与否，代理和浏览器缓存都会被使用。如果你站点的缓存配置不正确，你只能听天由命了。

Web缓存如何工作

所以的缓存都有一套自己的规则，可以用来决定何时跟缓存暧昧往来。其中部分规则设定在协议中(HTTP 1.0 以及 1.1)，部分由缓存管理员⑤设置。

> ⑤缓存管理员：如果指的是浏览器缓存，则有可能就是我们服务器专家同事，在服务器上配置一些缓存规则；如果是代理缓存，则指的就是处理代理服务器这块的管理人员。

一般而言有如下常用规则[N]：

1. 响应头明确说明，偶不想被缓存，则不会被缓存；
2. 如果请求信息是需要认证或者安全加密的(如, HTTPS)，相应内容也不会被缓存；
3. 缓存如果有以下表现，则认为是fresh新鲜的（无需检查源服务器，直接发送给客户端）：
   - 含有完整的过期时间和寿命控制头信息，并且内容仍在保鲜期内，或者
   - 缓存最近已展现，并且在不久前修改。

   则内容缓存直取，绕过源服务器。
4. 若内容陈旧，则会要求源服务器做验证 validate ，或者告诉缓存其拷贝副本是否是OK的。
5. 特定情况下——例如，断网了，之前有过的响应缓存直取而不检查源服务器。

响应如果没有类似ETag或Last-Modified头这样的校验器，也没有明确的更新信息，通常（并不绝对）认为是不可缓存的。

总而言之，新鲜度freshness和校验validation是确定缓存内容是否可用的最重要途径。如果要展示的足够新，直接缓存取；如果检测发现展示内容并未变化，则不会再来一次完整的传输。

如何控制缓存和不缓存

有很多工具可以帮助设计师和网站管理员调整服务器缓存网站的方式，这也许需要你亲自动手对服务器的配置进行一些调整，但绝对值得。了解如何使用这些工具请参考本文后面的章节。

HTML Meta标签 vs. HTTP头信息

HTML重构人员可以在文档的`<head>`中添加标签进行描述。这些meta标签通常用来标记不可缓存或过期时间。

Meta标签使用简单，但效果一般。因为只被少数几个浏览器宠幸，而代理缓存基本上就不访问HTML文档。尽管我们可以在页面上试图添加`no-cache` meta标签让页面一直是最新的，但其实没必要。

如果你的网站托管在ISP或者主机托管商那里，并且他们没有赋予您任意设置HTTP头信息的能力(比如Expires和Cache-Control)，你要投诉争取，因为在你的工作中这些是必须的。

另外一方面： HTTP头信息可以让你对浏览器和代理服务器如何处理你的副本进行更多的控制。他们在HTML代码中是看不见的，一般由Web服务器自动生成。但是，根据你使用的服务器，你可以在某种程度上进行控制。在下文中：你将看到一些有趣的HTTP头信息，以及如何在你的站点上应用部署这些特性。

HTTP头信息发送在HTML代码之前，只能被浏览器和一些中间缓存能看到，一个典型的HTTP 1.1协议返回的头信息看上去像这样：

```
HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
```

头信息空一行后是HTML代码的输出，关于如何设置HTTP头信息请参考对应章节。

Pragma HTTP头信息（以及为什么不起作用）

很多人认为在HTTP头信息中设置了`Pragma: no-cache`后会让内容无法被缓存。但事实并非如此：HTTP的规范中，响应型头信息没有任何关于Pragma属性的说明，只说明了请求头信息（浏览器发送给服务器的头信息）中的Pragma属性。虽然有少部分缓存会买账，但大部分无视，使用Pragma没作用。若要使用，试试下面的头信息。

使用Expires HTTP头信息控制不过期

Expires HTTP头是控制缓存的基本手段，Expires的中文意思是“有效期”，显然，就是告诉浏览器缓存的有效期。如果过期，缓存会检查源服务器以确定文件是否改变了。Expires头几乎每个缓存都支持。

大部分的服务器允许你以多种方式设置Expires响应头。通常，他们允许设置一个绝对过期时间，然后对比最后一次访问的时候或者最后一次文档修改的时候决定客户端内容的获取方式。

对于静态图片（如导航或按钮的图片）而言，Expires头信息是相当有用的，因为图片不怎么修改，您可以给图片设置一个相当长的过期时间，这回让你的用户感觉网站变快了。Expires对于控制有改变规律的网页也很有用，例如：你有一个新闻聚合页面，每天早上6点钟准时更新，您可以设置缓存的过期时间也是这个点，于是缓

存就可以很聪明地知道什么时候该去重载新的内容，什么时候睡大觉。

Expires头唯一的有效值是HTTP时间，其他值都会被认为是“前男友前女友”之类，不会去缓存的。注意：时间是格林威治时间（GMT），而不是本地时间。如下所示：

```
Expires: Fri, 30 Oct 1998 14:19:41 GMT
```

显然，如果你要使用Expires头，确保你的Web服务器时间的准备就非常重要了。使用网络时间协议(Network Time Protocol – NTP)不失为一个号方法。如果你的身边有本地系统管理员，可以向他咨询，或者查看下面的百科[Add ⑥]。

尽管Expires头很有用，但它有一定的局限性。首先，因为牵扯到时间，Web服务器端的时钟必须和缓存的同步，否则很可能实现不了预期的结果——缓存把前女友当初现女友，把现女友当作过去式——那就悲剧了。

另外一个问题是，你很容易忘记给某内容设置了一个特定时间，如果返回内容的时候没有更新这个过期时间，则每个请求都是上访到服务器，反而增加了负载和响应时间。

> ⑥网络时间协议(NTP)：以封包交换把两台电脑的时钟同步化的网络协议。NTP使用UDP端口123作为传输层。它是用作抵销可变延迟的影响。NTP是仍在使用中的最古老的网络协议之一（在1985年前开始）。NTP最初由德拉瓦州大学的Dave Mills设计，他与一群志愿者仍在维护NTP。

Cache-Control（缓存控制）HTTP头信息
HTTP 1.1引入了新的头信息：`Cache-Control`响应头信息，让网站的发布者可以更全面的控制他们的内容，更好地处理Expires的些限制。`Cache-Control`有用的响应头包括：

- max-age=[秒]：表示在这个时间范围内缓存是新鲜的无需更新。类似Expires时间，不过这个时间是相对的，而不是绝对的。也就是某次请求成功后多少秒内缓存是新鲜的。
- s-maxage=[秒]：类似`max-age`，除了仅应用于共享缓存（如代理）。
- public：标记认证的响应才能够被缓存。一般而言，需要认证HTTP请求内容会自动私有化（不会被缓存[Add]）。
- private[N]：允许缓存专门为某一个用户存储响应，比方说在浏览器中；共享缓存一般不会，例如在代理中。
- no-cache：每次在释放缓存副本之前都强制发送请求给源服务器进行验证，这在确保认证有效性上很管用（和`public`结合使用）或者保证内容必须是即时的，不得无视缓存的所有优点，如国内的微博、twitter等的刷新显示[Add]。
- no-store：强制缓存在任何情况下都不要保留任何副本。
- must-revalidate：告诉缓存，我给你准备了一些关于新鲜度的信息，在表现的时候要严格遵循之。HTTP允许缓存在某些特定情况下返回过期数据，指定了这个属性，相对于告诉缓存，你丫必须严格遵循我的规则。
- proxy-revalidate：类似`must-revalidate`，除了只能应用于代理缓存。

举个板栗：

```
Cache-Control: max-age=3600, must-revalidate
```

如果`Cache-Control`和`Expires`同时存在，`Cache-Control`说了算[N]。如果你打算使用`Cache-Control`头，你应该好好看看”HTTP 1.1 规范“，详见参考文章以及拓展阅读。

验证器和验证

在缓存如何工作这段译文中，我们说过，服务器以及缓存通过验证来判断内容是否改变，在不确定内容是否过期的时候，可以避免本地已经存在副本的时候下载整个内容。

验证器是很重要的，如果一个都没有，同时没有可用的新鲜度信息(`Expires`或`Cache-Control`)，缓存一点儿都不会存储内容。

最常见的验证是通过`Last-Modified`头信息通信确定文档最后的修改时间，如果缓存有内容存储，会包含`Last-Modified`信息的，辅助`If-Modified-Since`请求，我们可以询问服务器内容是否改变了。

HTTP 1.1引入了一个新的验证器，称为`Etag`⑦. `Etag`是每次展现内容改变时候由服务器生成的唯一标识符，由于服务器控制`ETag`如何生成，当缓存发起`If-None-Match`请求的时候，如果`Etag`匹配，就可以确定展示内容其实是一样的。

> ⑦Etag: HTTP协议规格说明定义ETag为”被请求变量的实体值”。另一种说法是，ETag是一个可以与Web资源关联的记号（token）。典型的Web资源可以一个Web页，但也可能是JSON或XML文档。服务器单独负责判断记号是什么及其含义，并在HTTP响应头中将其传送到客户端，以下是服务器端返回的格式：
> ETag:”50b1c1d4f775c61:df3″客户端的查询更新格式是这样的：If-None-Match : W /“50b1c1d4f775c61:df3″如果ETag没改变，则返回状态304然后不返回，这也和Last-Modified一样。测试Etag主要在断点下载时比较有用。

几乎所有的缓存使用`Last-Modified`时间作为验证器，`Etag`验证也开始变得流行。

所有新一代的Web服务器都对静态内容（如：文件）自动生成`ETag`和`Last-Modified`头信息，而你不必做任何设置。但是，服务器对于动态内容（例如：CGI, ASP或数据库生成的网站）并不知道如何生成这些信息，参考一下编写支持缓存的脚本章节；

创建支持缓存网站的小技巧

除了使用新鲜度信息以及验证，还有其他一些技巧可以让你网站的缓存更加友好：

- 保持URL稳定：这是缓存的金科玉律，如果你为不同页面，不同用户或不同网站提供相同的内容，他们应该使用相同的URL. 这是简单却非常行之有效的方法。例如，你的HTML中的某个引用地址是`"/index.html"`, 则要一直使用这个地址。
- 不同地方的图片和其他元素使用同一库。
- 对于不经常改变的图片/页面启用缓存，通过将`Cache-Control: max-age`头信息的值设大一点。
- 对于定期更新的内容通过指定`max-age`或过期时间实现缓存。
- 如果资源改变了（尤其下载文件），改变其名字。由于一般这种资源会有很长的过期时间，而服务器上一直是正确的版本；因此，链接这个下载资源的页面需要要比较短的过期时间（//zxx: 我司页面5分钟过期）。否则，会出现服务器的资源是新的，但页面被缓存了，其中的链接地址还是旧的，就会出现新旧版本冲突的可能Add。
- 万不得已不要变动文件：否则你要设置一个新的`Last-Modified`值。另外，当你更新站点的时候，只要上传改动的那些文件，而不要把整个站点都覆盖过去。
- Cookie能不用就不用：Cookie难以被缓存，且大多情境下是没有必要的。如果你非得使用Cookie，建议用在动态页面上。
- 减少SSL⑧的使用：因为共享缓存不能存储认证页面，只在必要的时候使用，并且在SSL页面上减少图片的使用。
- 使用REDbot⑨检查你的网站：可以帮助你应用本文所介绍的一些概念。

> ⑧ SSL：全称Secure Socket Layer – 安全套接层，为Netscape所研发，用以保障在Internet上数据传输之安全，利用数据加密(Encryption)技术，可确保数据在网络上之传输过程中不会被截取及窃听。目前一般通用之规格为40 bit之安全标准，美国则已推出128 bit之更高安全标准，但限制出境。只要3.0版本以上之I.E.或Netscape浏览器即可支持SSL。
>
> ⑨ REDbot：REDbot = RED + robot，是个机器人，检查HTTP资源，看他们如何会表现，指出常见的问题，并提出改进建议。虽然它属于HTTP一致性测试仪，但却可以找到不少HTTP相关问题。

编写支持缓存的脚本

默认情况下，大多数的脚本不会返回验证器（`Last-Modified`或`Etag`响应头）或新鲜度信息（`Expires`或`Cache-Control`）。尽管有些脚本的确是动态的（意味着每次请求都有不同的响应），还是有很多（如搜索引擎或数据库驱动的）网站可以从缓存中受益。

一般来讲，对于同一个请求（无论是几分钟还是几天之后），如果脚本产生的内容是可重复的，则可以缓存。脚本内容的改变仅仅依赖于URL，则可以缓存。如果是依赖于Cookie，认证信息或其他外部条件，很可能不缓存。

- 最利于缓存的脚本就是在内容改变时导出成静态文件，服务器会想对待其他Web一样对待它的，生成以及使用验证器，于是你可以好好地喝杯咖啡了。记住，只有文件更改的时候才写入，这样Last-Modified时间就会被保存下来。
- 另外的脚本缓存之道就是使用`age`相关的头部，相比`Expires`，`Cache-Control: max-age`更容易些，因为是相对时间，每次新请求完成后重新设置，时间到了，再重新请求，再设置新的相对过期时间。
- 如果上面的做法你搞不定，你还可以试试通过脚本生成一个校验器，然后回应`If-Modified-Since`和/或`If-None-Match`请求。通过分析HTTP头信息，在适合的时候回应`304 Not Modified`. 不幸的是，这不是个打打酱油就能搞定的任务。

其他一些技巧

- 不要使用POST：若是获取数据，尽量不使用POST模式，因为POST方式返回内容大部分不会被缓存，相对的，通过GET以路径和查询发送的信息被缓存存储下来供后续使用。
- URL地址中不要嵌入特定的用户信息，除非生成的内容对于用户而言是唯一的。
- 不要指望同一用户的所有请求来自同一主机，因为缓存经常协同工作。//zxx:　嘛意思？
- 生成Content-Length[⑩]头信息。实现不难，可让你的脚本以持久连接(persistent connection)形式响应。这允许客户端在一个TCP/IP请求上请求多个内容，而不是为每次请求单独建立连接，这样你的网站相应会快很多。

> ⑩Content-Length：指明实体正文的长度，以字节方式存储的十进制数字来表示。在数据下行的过程中，Content-Length的方式要预先在服务器中缓存所有数据，然后所有数据再一股脑儿地发给客户端。

常见问题解答

缓存可用的最重要事情是？
其中一个不错的策略是找出常用的、规模较大的内容（尤其图片），然后优先处理之。

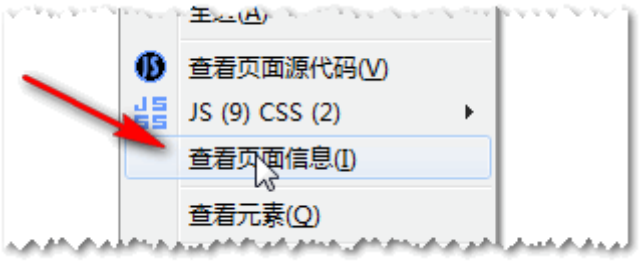
我该如何利用缓存让我的页面尽可能的快？
最应该缓存的内容设置一个较长的过期时间。验证有助于减少查看内容的时间，不过缓存仍会连接源服务器查看是不是过期了。如果缓存已经知道内容是新鲜的，直接返回。

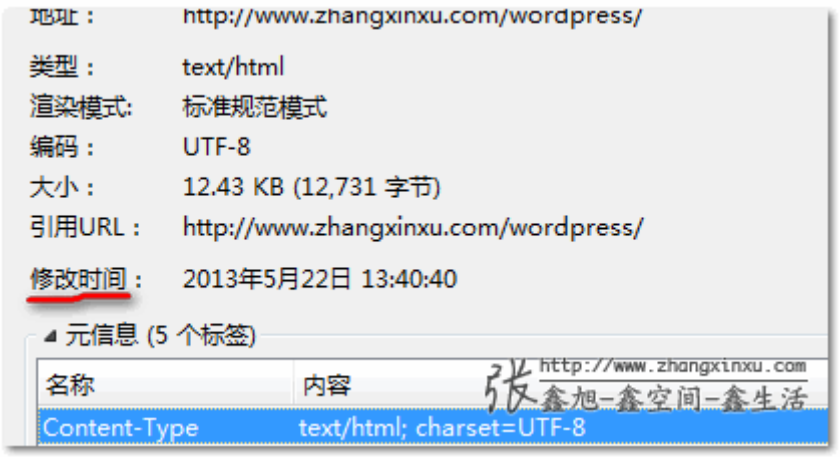
我知道缓存是个好东西，但是我想随时知道多少人访问了我的网页！

如果你必须知道每一次页面被访问的情况，可以选择页面上的一个小元素(或页面本身)，然后给这个元素一个适当的头信息使它是不可缓存。比如，你可以在每一个页面上引用一个1像素×1像素的不可缓存（如scr地址后面加个随机数[Add]）的透明图片。Referer头信息将会包含调用它的页面信息。

请注意，即使这样也不能给出你用户的精确统计，并且对通过互联网访问的用户也不是很友好：产生不必要的流量，并强迫用户等待未被缓存的内容从网络上下载回来。更多的信息可参见拓展阅读中的“解读访问统计”对应内容。

我该如何查看HTTP头？
许多浏览器可以查看`Expires`和`Last-Modified`头信息，如右键→查看页面信息或类似面板。例如，在Firefox浏览器下[Add]：





表示要看到完整的头，您可以使用Telnet[2]客户端手动连接到Web服务器上。

为此，你可能需要用一个字段指定端口（默认是80），或者连接到`www.example.com:80`或者`www.example.com 80`(注意是空格)，更多设置请参考一下telnet客户端的文档。

一旦连接到该网站，输入请求。比如，你想查看`http://www.example.com/foo.html`的头信息，首先连接到`www.example.com`，使用80端口，并输入：

```
GET /foo.html HTTP/1.1 [return]
Host: www.example.com [return][return]
```

`[return]`等同敲回车键，最后输入两次确认。这样就会输出头信息，然后跟着实际内容。如果只想看到头信息，使用`HEAD`来替换`GET`.

> ?Telnet：Telnet协议是TCP/IP协议族中的一员，是Internet远程登陆服务的标准协议和主要方式。它为用户提供了在本地计算机上完成远程主机工作的能力。在终端使用者的电脑上使用telnet程序，用它连接到服务器。终端使用者可以在telnet程序中输入命令，这些命令会在服务器上运行，就像直接在服务器的控制台上输入一样。可以在本地就能控制服务器。要开始一个telnet会话，必须输入用户名和密码来登录服务器。

> Telnet是常用的远程控制Web服务器的方法。

我的页面是密码保护的，代理缓存是怎么处理的？
默认情况下，HTTP验证保护的页面是私有的，共享缓存是不能保存的。然而，你可以通过`Cache-Control: public`头的设置使其公有。HTTP 1.1标准兼容的缓存服务器可以使之缓存。

如果你希望这些缓存的页面在用户查看之前还要验证一下，可以组合使用`Cache-Control: public`和`no-cache`头，这相对于告诉缓存器它从缓存中送出内容前必须递交客户端的验证给原始服务器。这个头信息如下所示：

```
Cache-Control: public, no-cache
```

不管怎么，这是最小化验证最好的方法；例如，你的图片不敏感，你可以把它放在分离的目录中，并配置你的服务对它们不做强制验证。这样，那些图片就会很自然的被缓存了。

如果人们通过缓存访问我的网站，我应该担心安全吗？
SSL页面不会被代理服务器缓存，所以这个你不需要担心。但是，代理服务器就好非SSL页面请求以及URL抓取这口，你懂的，这是不安全的。无良的管理员可能就会收集网站用户的信息，尤其在URL中。

事实上，任何网络管理员都可以收集你的客户端和服务器端之间的这类信息。CGI[?]脚本有个漏洞，会把用户名和密码放在自身的URL地址中，这很容易让其他人发现用户的登陆信息。

如果你懂得互联网安全的些基本机制，就不会对代理缓存感到任何惊讶。

> ?CGI：通用网关接口(Common Gateway Interface). 用于初始化软件服务的服务器方接口。这套接口描述了Web服务器与同一计算机上的软件的通信方式。
>
> 通用网关接口，它是一段程序，运行在服务器上，提供同客户端HTML页面的接口，通俗的讲CGI就像是一座桥，把网页和WEB服务器中的执行程序连接起来，它把HTML接收的指令传递给服务器，再把服务器执行的结果返还给HTML页；用CGI可以实现处理表格，数据库查询，发送电子邮件等许多操作，最常见的CGI程序就是计数器。CGI使网页变得不是静态的，而是交互式的。

我在寻找一个集成的Web发布解决方案。哪些是可缓存的？
这个是不确定的。一般来说，越复杂的系统越难缓存。最差的情况就是所有的内容都是动态生成，并且不提供校验器，与缓存压根无缘。你可以和你供应商的技术人员沟通获取更多信息，并参考下面实现注意事项。

我的图片缓存一个月后才到期，我现在就想变动！
`Expires`头是绕不过去的，除非缓存（浏览器或者代理）空间不足才会删除副本，缓存副本会一直使用。

最有效的方法是修改链接，这样会从源服务器获取完整的新内容。请记住，调用图片的这个页面也会被缓存的，正因如此，我们需要让图片以及其他类似的静态资源易缓存，而页面呢可以随着自身的改变（例如改变了一个图片的URL地址[Add]）即时更新。

如果你想摆脱特定缓存，重载内容，可以试试强制刷新（在FireFox中，shift键+reload按钮等同于处理`Pragma: no-cache`请求头）或者让缓存管理员使用某些接口删除内容。

我运行一个Web Hosting服务。我怎样才能让我的用户发布缓存友好的网页？
如果你使用apahe，可以考虑允许他们使用`.htaccess`文件并提供相应的文档。

否则你需要在每一个虚拟主机上为各种缓存属性建立预定的区域。比如：你可以指定一个叫`/cache-1m`的目录用

来放读取后要缓存一个月的内容，然后再建一个`/no-cache`的目录，并在头信息中指定这么目录中的内容不被缓存。

不管上面你做的如何，总之最好优先给用户量大的客户做缓存处理。大部分服务器节约的流量以及负载都是来自高容量的网站。

我明明告诉网页要好好缓存，但它老是去请求，怎么破？
缓存服务器并不总是要求内容要保持并重用，某些条件下，他们是不保存不重用的，所有的缓存服务器都回基于文件的大小、类型（图片、页面…），或者服务器空间的剩余来确定如何缓存。如果你的文件比较大或很热门，可能就不会被缓存。有些缓存服务器允许管理员决定哪些内容要存储，有些缓存服务器允许内容长存缓存中，所以，它们总是可用的。

实现需注意的：Web服务器端

一般说来，应该选择最新版本的Web服务器程序来部署。不仅因为它们包含更多利于缓存的功能，新版本往往在性能和安全性方面都有很多的改善。

Apache HTTP服务器
Apache使用可选模块包含头信息，头信息`Expires`和`Cache-Control`一并包含。这些模块在1.2版本以上都支持。

这些模块需要编译到Apache中，虽然包含，但是默认并未开启。为了确定相应模块已经被启用，找到httpd[?]程序，运行`httpd -l`，它会列出可用的模块（注意，仅有内部编译的模块列表才会显示，在较新版本的Apache中，使用`httpd -M`可以包含动态加载的模块[N]），我们需要关注的是expires模块(`expires_module`)和headers模块(`headers_module`)。

> ?httpd：httpd是Apache超文本传输协议(HTTP)服务器的主程序。被设计为一个独立运行的后台进程，它会建立一个处理请求的子进程或线程的池。

- 如果这些模块不可用，你需要联系管理员，重新编译以包含这些模块。这些模块可以通过取消配置文件中的注释掉启用，或者在编译的时候增加`-enable -module=expires`和`-enable-module=headers`参数(apache 1.3+)．参开Apache中的INSTALL文件。

一旦你的Apache有了相应的模块，你可以使用`mod_expires`指定过期的时间，要么在`.htaccess`文件，要么在服务器的`access.conf`文件。你可以设置过期时间是从访问时间开始还是文件修改时间开始，并应用到特定类型文件上或设为默认配置。查看官方该模块文档获得更多信息，或者遇到问题的时候向你身边的apache专家讨教。

为应用`Cache-Control`头，你需要使用`mod_headers`模块，其允许你为资源指定任意的头信息。可参考mod_headers官方文档。

下面是`.htaccess`文件展示了如何使用头信息：

- `.htaccess`文件允许Web发布者使用配置文件中的指定。可以影响目录以及子目录内容。和你的服务器管理员沟通下，看看它们是否可用。

```
### activate mod_expires
ExpiresActive On
### Expire .gif's 1 month from when they're accessed
ExpiresByType image/gif A2592000
### Expire everything else 1 day from when it's last modified
```

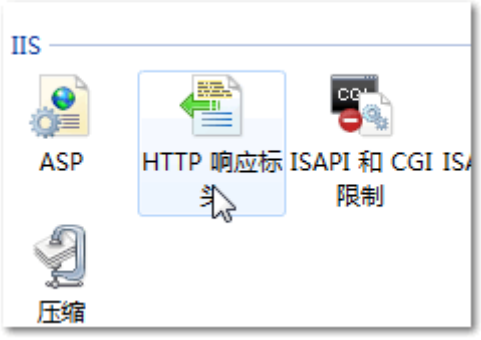
```
### (this uses the Alternative syntax)
ExpiresDefault "modification plus 1 day"
### Apply a Cache-Control header to index.html
<Files index.html>
Header append Cache-Control "public, must-revalidate"
</Files>
```

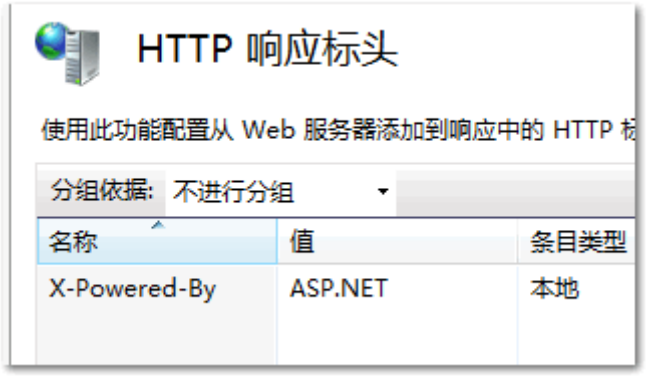
- 注意，在有些情况下，`mod_expires`会自动计算并插入`Cache-Control:max-age`头信息。

Apache 2′ s的配置和1.3类似，更多信息可以参考2.2[N]的mod_expires和mod_headers文档。

微软IIS
微软的IIS有一些灵活的方式可以很容易得设置头信息，不过似乎只针对IIS 4.0服务器，并且只能在NT服务器上运行。

为了给网站某区域指定头信息，需要进入`Administration`(管理员)工具面板，然后再设置属性。选择`HTTP Headers`选项卡后，你会看到两个有意思的区域：`Enable Content Expiration`和`Custom HTTP headers`，第一个含义一目了然，第二个用来应用`Cache-Control`头。//zxx: 此处的操作描述很过时了，看看window7下，操作界面早就大变样了！





设置ASP页面(Active Server Pages)的头信息可以参考后面的ASP章节，也可以通过ISAPI模块设置头信息，细节请参考MSDN。

Netscape/iPlanet企业服务器
3.6版本以后，企业版服务器已经不能以任何方式设置`Expires`头信息了。然而，其从3.0版本开始支持HTTP 1.1的功能。这意味着HTTP 1.1的缓存（代理服务器/浏览器）利用你对`Cache-Control`的设置来获得。

为了使用`Cache-Control`头，在管理员服务器中选择`Content Management | Cache Control Directives`（内容管理|缓存控制指令）。然后，使用资源选择器(Resource Picker)，选择你希望设置头信息的目录。设置完头信息后，点击”确定”。更多信息请参考NES手册。

实现需注意的：服务端脚本

时刻谨记，在Web服务器上设置HTTP要比通过脚本设置轻松些。你可以两者都试试。

因为服务器端的脚本主要是为了动态内容，所以即使实际上内容可以被缓存的，其也不会生成缓存很强的页面。如果你的页面内容经常变动，但不是每个页面都中枪，可以考虑设置`Cache-Control: max-age`头信息，大部分用户是在相对端的时间内再次访问这个页面。例如：用户点击“后退”按钮，如果没有任何验证或新鲜度信息，他们将不得不等待，直到从服务器页面重新下载才能看到它。

CGI

CGI脚本是生成内容最常用的技术之一。你可以轻轻松松在请求发送给主体之前添加HTTP请求信息。大部分CGI实现都需要添加`Content-Type`头信息，例如这个Perl脚本：//zxx: 还是挺好懂的

```
<#!/usr/bin/perl
print "Content-type: text/html\n";
print "Expires: Thu, 29 Oct 1998 17:04:19 GMT\n";
print "\n";
### the content body follows.../pre>
```

由于都是文本，你可以很容易通过内置函数生成`Expires`和其他日期相关的头信息。如果你使用`Cache-Control: max-age`会更简单：

```
print "Cache-Control: max-age=600\n";
```

上面脚本可以让请求完成后缓存10分钟，因此，当用户点击“后退”按钮的时候，就不会重新涂胶请求了。

CGI的规范同时也允许在脚本环境中，客户端发送请求头信息，每个头信息都有一个'HTTP_'的前缀。于是乎，如果一个客户端发送一个`If-Modified-Since`请求，就是这样的：

```
HTTP_IF_MODIFIED_SINCE
```

可观摩cgi buffer库，其可以自动实现`Etag`生成和验证，`Content-Length`生成及gzip内容，而所有这些实现，只需要一行include，就可以为Perl和Python写CGI脚本。Python版本还可以包装任意的CGI脚本。

服务器端包含

SSI（扩展名通常是`.shtml`）最早可以生成动态内容的网站发布方案。通过在页面中使用特定的标签，有一定限制的内HTML脚本就可以使用了。大部分的SSI实现不设置验证器，故无法缓存。不过Apache服务器允许通过设置让SSI文件可缓存，通过适当的文件并结合`XbitHack full`指令设置组执行权限。欲了解更多信息，请参阅mod_include文档。

PHP

PHP为服务器端脚本语言，在服务器内置的时候，可以在HTML页面中内嵌使用，很像SSL，不过有更多的可选项。PHP可以在任何Web服务器(Unix或Windows)或Apache模块上作为CGI使用。

默认情况下，PHP生成的内容没有分配验证器，因此，不能缓存。不过，开发人员可以通过`Header()`函数设置HTTP头信息。例如，创建`Cache-Control`头，过期时间为3天：

```
<?php
 Header("Cache-Control: must-revalidate");

 $offset = 60 * 60 * 24 * 3;
 $ExpStr = "Expires: " . gmdate("D, d M Y H:i:s", time() + $offset) . " GMT";
 Header($ExpStr);
?>
```

记住`Header()`需要在所有的输出之前。

正如你看到的，你可以手工创建HTTP日期。PHP没有专门的函数（新版本已改进，请参考PHP的日期相关函数文档）。当然，最简单的还是设置`Cache-Control: max-age`头信息，适用于大部分情况。

更多内容，请参阅header手册。

还是cgi buffer库，只要一行包含，就能以PHP脚本形式自动实现`Etag`生成和验证，`Content-Length`生成及gzip内容。

Cold Fusion

Cold Fusion是Macromedia的商业服务器端脚本引擎，并且支持多种Windows平台，Linux平台和多种Unix平台。//zxx: 看到Macromedia心亮了半截，几百年前就被收购的公司……此文未免太过时了点了~~大家这段可以跳过了，几乎没有任何价值……

Cold Fusion通过CFHEADER标记设置HTTP头信息相对容易。可惜的是，以下的Expires头信息的设置有些容易误导：

```
<CFHEADER NAME="Expires" VALUE="#Now()#">
```

它并不像你想像的那样工作，因为时间（本例中为请求发起的时间）并不会被转换成一个符合HTTP时间，而且打印出副本的Cold fusion的日期/时间对象，大部分客户端会忽略或者将其转换成1970年1月1日。

但是：Cold Fusion另外提供了一套日期格式化函数-GetHttpTimeSTring. 结合DateAdd函数，就很容易设置过期时间了，这里我们设置一个头信息，声明内容在1个月以后过期：

```
<cfheader name="Expires"
  value="#GetHttpTimeString(DateAdd('m', 1, Now()))#">
```

你也可以使用`CFHEADER`标签设置`Cache-Control: max-age`以及其他头信息。

记住，Web服务器也会将头信息设置转给Cold Fusion(做为CGI运行的时候)，检查你的服务器设置并确定你是否可以利用服务器设置代替Cold Fusion.

ASP和ASP.NET

在asp中设置HTTP头信息时，确保`Response`方法调用在HTML内容输出之前，或者使用`Response.Buffer`暂存输出。同时，注意某些版本的IIS默认设置会输出`Cache-Control: private`头信息，必须声明成`public`才能被共享缓存服务器缓存。

ASP(Active Server Pages)，IIS内置，也可用于其他Web服务器，同样允许你设置HTTP头。例如设置过期时间，

你可以使用`Response`自带属性：

```
<% Response.Expires=1440 %>
```

指定内容过期的分钟数。`Cache-Control`头添加如下：

```
<% Response.CacheControl="public" %>
```

在ASP.NET中，`Response.Expires`已经不推荐使用了，正确的方法是通过`Response.Cache`设置缓存相关的头信息，如下：

```
Response.Cache.SetExpires ( DateTime.Now.AddMinutes ( 60 ) ) ;
Response.Cache.SetCacheability ( HttpCacheability.Public ) ;
```

参考文档和拓展阅读

HTTP 1.1规范
HTTP 1.1的规范对页面缓存以及权威的接口实现指南有了大量的扩展，参考章节：13, 14.9, 14.21以及14.25 .

Web-Caching.com
对缓存概念有很好的介绍，并且有很多其他在线资源的链接。

解读访问统计
Jeff Goldberg这篇内容丰富叙述会告诉你为什么不应该过度依赖访问统计和计数器。//zxx 上世纪的复古页面...

REDbot
检查HTTP资源，以确定它们如何与Web缓存交互，以及通常如何使用该协议。

cgi buffer库
只要包含一行Perl CGI, Python CGI以及PHP脚本，就能自动实现`Etag`生成以及验证，`Content-Length`生成以及Gzip内容的正确编码。Python版本还可以包装任意的CGI脚本。

关于本文档

内容在发布时是可以确保其正确性，但是，随着时间推移，就不能保证正确无误了。因此，如有链接404，描述错误或其他需要纠正的问题请尽快告知作者。

本文最新版本可以从http://www.mnot.net/cache_docs/获得

文档说明：含有上标N的表示与前辈翻译时候相比新增的；上标Add表示作为译者的我为了便于理解自己添加的；上标数字(①-?)是对一些可能不熟悉的名词的百科解释。

虽然原作语言不生动，教科书般一板一眼[?]；有些可能过时了。不过，还是学到了很多东西。还是很值的！欢迎分享，欢迎传播，以后面试之前来这里看看，可能会有帮助哦！

> ？如果我介绍缓存，我可能就这么讲：缓存是什么？顾名思意，就是缓慢的存钱。为什么要缓慢的存钱，因为工资卡都上交老婆了，为了攒点零花钱又不能被老婆发现，只能慢慢存了。那缓存有什么用呢？你想啊，自己有点小钱，做事情就方便快捷了，比方说我想买个鱼竿，就不要去向老婆要（给不给先不谈），自己从自己这边取，大大提高了执行的速度。
>
> 那什么时候可以存什么时候不能存呢？老婆给零花钱的时候，可能会有过期时间头，例如，周一甩了100块钱，拿去，这是一周的伙食！这个一周就是过期时间头(Expires Header)，在这个时间内，你的钱可以从缓存，也就是自己这里取……

> 相关主题：