第23学时　服务器推送和访问次数计数器

在本学时中，我们将要讲述两个常用的CGI编程方法。你可以使用这些方法制作更加有趣的Web页，使之具备初步的动画功能，或者让人们竞相使用。

在本学时中你要学习：

- 使用服务器推送方法来刷新Web页。
- 访问次数计数器。
- 代理和缓存。

23.1　什么是服务器推送

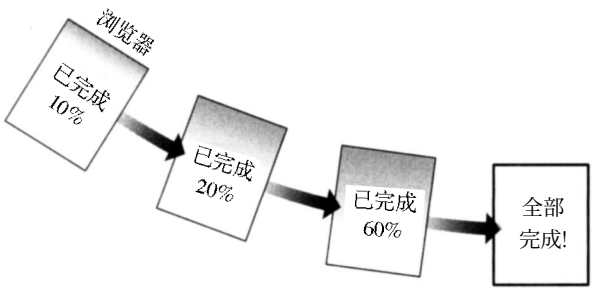
在传统的Web页中，加载速度慢或者不断增大的文档是很难使用的，因为Web页只能一次看一页。比如，运行CGI程序的Web页需要花费很长的时间来运行。

首先，浏览器为了等待CGI程序结束运行，可能会超过原定的时间。浏览器为了等待程序运行的结果，通常等90秒钟左右，然后显示了一条消息，声称无法访问该站点。

其次，CGI程序有时会输出一条消息说：“I’m still working, 20%complete (我仍在运行，已完成10%)”，过一会儿又说：“I’m still working ,20% complete(我仍在运行，已完成20%)”，等等。输出这些消息是好的，问题是这些消息并不按固定间隔出现（因为缓存的缘故），当你完成程序的运行时，会有一个很长很长的Web页。

你希望的是浏览器显示如图23-1所示的信息。

图23-1　浏览器显示进度递增的信息



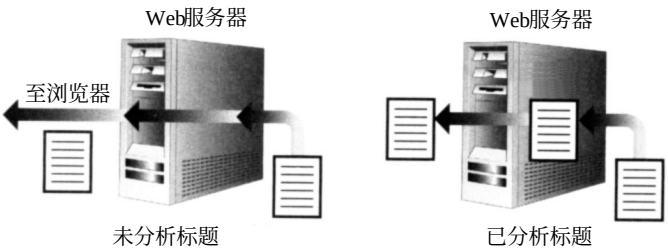
服务器推送技术利用了这样一个特性，即浏览器能够按各个部分来接收Web页，然后依次重新显示这些Web页，就像你是依次取出不同的Web页一样。

在撰写本书时，Microsoft的Internet Explorer并不支持执行服务器推送技术所需的协议。这是很遗憾的，因为使用服务器推送技术是使Web页的内容动起来的一种简便方法。对于需要支持Internet Explorer或者不支持这个特性的其他浏览器的Web页来说，你应该使用客户机拖拉之类的其他技术。

23.1.1 激活服务器推送特性

你的Web服务器必须正确地安装，以便激活服务器的推送特性。为此，必须将 CGI程序作为未分析标题(nonparsed header)CGI程序来运行。当你使用未分析标题 CGI程序时，服务器并不要求输出CGI标题，数据应该按原始状态直接发送给浏览器。通常来说， Web服务器要检查来自CGI程序的输出，以确保它们的正确性，因此，当 CGI程序运行失败时，便出现错误 500。未分析标题CGI程序将它们的输出直接发送到浏览器，如图 23-2所示。

图23-2 未分析数据不经过检查就通过了服务器



你如何运行CGI程序以及服务器如何对标题不进行检查，这取决于 Web服务器本身。例如，如果是Apache Web服务器，那么在CGI程序的文件名前面加上前缀 nph-，就能使程序作为未分析标题程序来运行。例如， push.cgi是个已分析标题CGI程序，nph-push.cgi便是未分析标题CGI程序。但是Web服务器管理员可以修改这个命名规则的运行方式。

在Microsoft的Internet信息服务器（Internet Information Server, IIS）下，所有CGI程序都是作为未分析标题程序来运行的。 CGI模块的header函数通常向你隐藏了这个情况，因此，在IIS下不必为服务器推送特性作任何修改。

如果你不清楚如何运行带有未分析标题的 CGI程序，可以查看Web服务器的文档资料，或者求助于你的系统管理员。

23.1.2 一个小例子：更新Web页上的时钟

服务器推送技术的第一个例子是：编写一个简单的程序，以便更新 Web页上的时钟。该时钟的运行方式是：让 Web服务器每隔5秒钟左右推送出一个 Web页，而Web页上将显示新的时间。Web服务器将不断推送出新的时间，直到浏览器删除该页，或者用户点击浏览器的 Stop按钮，停止加载该页为止。

该CGI模块包含一组函数，目的是使服务器的推送操作比较容易一些。服务器推送的 Web页也称为多部分文档。

程序清单23-1包含了HTML时钟的源代码。你必须键入该代码，然后用一个名字保存该代码，使Web服务器能够像上一节介绍的那样，将该程序作为未分析标题 CGI程序来运行。

程序清单23-1 HTML时钟的源代码

```
1:   #!/usr/bin/perl -w
2:
3:   use strict;
4:   use CGI qw(:push -nph);
5:
6:   $|=1;    # Enable automatic buffer flushing
7:
```

```
8:   print multipart_init;
9:   while(1) {
10:      print multipart_start;
11:      print "The time is <H1>", scalar(localtime), "</H1>\n";
12:      print multipart_end;
13:      sleep 5;
14:  }
```

第4行：当CGI模块加载时，你必须指明正在执行服务器推送操作，因此这一行代码将命令:push赋予该CGI模块。另外，当你编写未分析标题脚本程序时，必须使用 -nph将这个情况通知CGI模块。

第8行：multipart_init负责告诉浏览器，它后随的是个多部分Web页。它输出的是multipart_init，而不是通常用在普通Web页上的header函数。

第9行：while(1)能够有效地创建一个while循环，该循环将永远重复运行下去，这种循环称为无限循环。

第10行：multipart_start给要刷新的Web页的开始做上标号。如果一个Web页已经显示，本行代码将使浏览器清除该Web页，并等待接收新的内容。

第11行：这一行代码用于指明该页的内容。第 4学时我们曾经讲过，标量上下文中的localtime用于输出格式为“ Sun Sep 5 15:15:30 1999 ”的时间。

第12行：multipart_end用于给要刷新的Web页的结尾做上标号。本行代码只应该后随另一个multipart_start或者程序的结尾。

请注意while循环如何给multipart_init和multipart_end函数加上方括号。该循环能够有效地一次又一次重复显示同一个Web页，只有Web页上的时间是变化的。

23.1.3 另一个例子：动画

在程序清单23-2中显示的下一个例子（与上一个例子非常相似）中，显示了来自目录/images的一连串图形。这些图形使用服务器推送方法每次显示一个图形。目录中的每个文件被读取，并且作为一连串推送的Web页显示在浏览器中。

程序清单23-2 用服务器推送方法实现的动画

```
1:   #!/usr/bin/perl -w
2:
3:   use strict;
4:   use CGI qw(:push -nph);
5:   my($imagedir, @JPEGS);
6:   # Change the directory name below to suit your needs
7:   $imagedir="/web/Clinton_Test_Area/images";
8:   opendir(ID, $imagedir) || die "Cannot open $imagedir: $!";
9:   @JPEGS=sort grep(/\.jpg$/, readdir ID);
10:  closedir(ID);
11:
12:  $|=1;    # Enable automatic buffer flushing
13:
14:  print multipart_init;
15:  foreach my $image (@JPEGS) {
16:      print multipart_start(-type => 'image/jpeg');
17:
18:      open(IMAGE, "$imagedir/$image") || die "Cannot open $image: $!";
```

```
19:       binmode(STDOUT); binmode(IMAGE);  # Windows NT/95/98 only
20:       print <IMAGE>;
21:       close(IMAGE);
22:       print multipart_end;
23:       sleep 5;
24:   }
```

程序清单23-2中的程序，大部分都与第20学时中介绍的“当日图形”和程序清单23-1的例子非常相似。

需要介绍的重要代码是第16行，即multipart_start(-type=>‘image/jpeg’)，它用于指明CGI程序并不在连续的Web页上输出普通文本或HTML输出信息，而是输出JPG图形。为了执行动画操作，该程序既可以直接输出JPEG，也可以输出包含＜IMG SRC＞标号的HTML。

23.1.4 客户机拖拉技术

使Web页依次加载的另一种技术称为客户机拖拉。使用客户机拖拉技术时， HTML中嵌入了一些标记，以便告诉浏览器在一个间隔时间之后重新加载 Web页（或另一个URL）。例如，下面这个在Web页的＜HEAD＞节中的HTML

```
<META HTTP-EQUIV="refresh" CONTENT="6;http://foo.bar.com">
```

将使浏览器在6秒钟后加载Web页http://foo.bar.com。CGI模块直接支持客户机拖拉命令。

当Web页的标题输出时，你可以设定应该重新加载的 Web页，或者加载另一个Web页取代它的位置，方法是使用CGI模块的header函数的-Refresh选项，如下所示：

```
print header(-Refresh => '6;URL=http://foo.bar.com');
```

客户机拖拉方法实际上用于加载两次“刷新”之间的一个完整新页。这意味着如果你的Web页需要依次显示，比如像幻灯片那样来显示，就必须使用 URL中嵌入的cookie或参数来跟踪下面显示哪个 Web页。在两次刷新之间，必须在服务器与客户机之间建立一个新的连接，并且Web服务器必须为每次刷新启动一个新的 Perl程序。这意味着不能太频繁地进行 Web页的刷新。

使用客户机拖拉方法和服务器推送方法时遇到的主要问题是：

- 有些浏览器（比如Internet Explorer）不支持服务器推送技术。
- 有些浏览器不支持客户机拖拉方法。

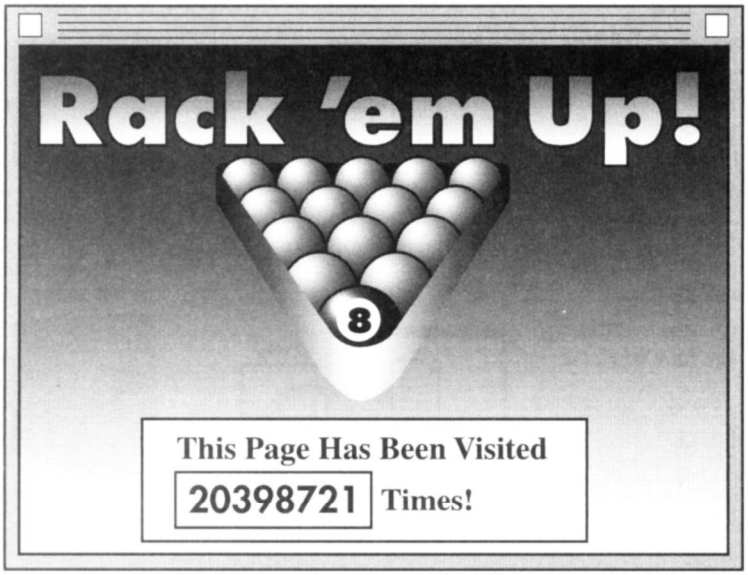
无论你用何种方法来编写你的程序，一种浏览器与另一种浏览器之间有些特性是互不兼容的。你必须决定你将容许存在哪些错误，并且根据情况编写代码。

23.2 访问次数计数器

在Web页上，你常常会看到一种称为访问次数计数器即访问者数量指示器的东西。可以想像，它表示Web页已被人们访问了多少次。图23－3显示了一个计数器的例子。

访问次数计数器有许多问题值得注意。首要的问题是计数器中的数字表示什么意思。“访问次数”的数目很大，表示访问这个 Web页的人很多。如果访问的人很多，是否表示这是个很好的Web页呢？不一定。如果你访问一个 Web页，这个Web页中可能有你需要的信息，也可能没有你要的信息。 Web页的质量好坏在于它是否含有对你有价值的信息，而不在于对其他人是否有价值。

图23-3 访问次数计数器的举例



实际上访问次数计数器是一种瞎子当裁判的选美比赛。计数器中的数字不一定是访问你的Web页的人数，它最多只不过是一个很不准确的估计数字。为什么这些计数器如此不准确呢？我想有下列几个原因。

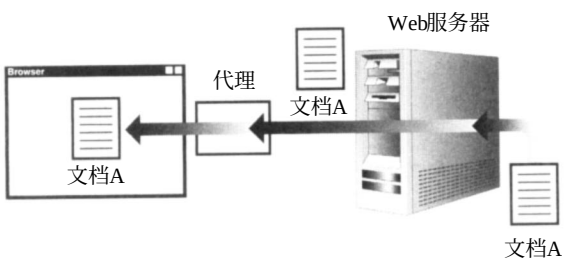
首先，没有一条规定说，访问次数计数器必须从 0开始计数。当给你发放最后一本支票簿时，第一张支票是1号支票吗？当你给支票排序时，完全可以选择你的起始序号。如果你很聪明，你会选择一个大号码，这样，你看上去在银行中开立了一个长期帐户。如果支票号码很小，那么商店服务员一定会对你的 ID号码多看两眼，并且也许根本不会考虑接收你的支票。Web站点操作员常常在开始时将访问次数计数器的数字设置得比较大，使他们的 Web点看上去比实际上更“受欢迎”。

访问次数计数器存在的第二个问题是 Web机器人程序，也叫做 Web蜘蛛、Web爬虫等。这些自动化进程能够搜索Web上的数据，有时只是为了查看一组特定的数据，有时为了建立感兴趣的Web站点的索引。你是否想过为什么 AltaVista、Google或HotBot要建立它们的索引呢？它们搜索Web，检索Web页，最后访问次数计数器的数字升高到比它们的实际访问次数高。

第三个问题是Web浏览器上的Refresh（刷新）按钮。每次在你的 Web页被刷新时，访问次数计数器就会升高一格。如果有人点击重新加载按钮，实际上你并没有计算你的 Web站点的“访问者”数目，是不是？

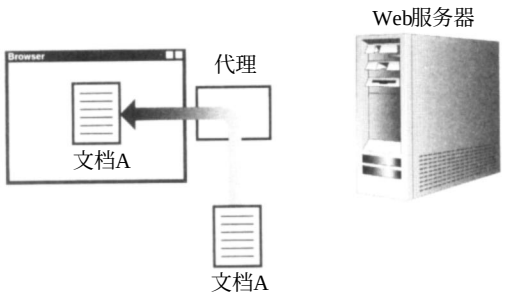
最后也是最重要的一个问题是缓存问题。在第 17学时中，我们介绍了浏览器如何与 Web服务器进行通信的方块图。它展示了一个重要细节，如图 23-4所示。

图23-4 代理服务器为浏览器检索Web页



如果Web浏览器位于大型ISP如aol.com或home.com等的域中，这些ISP拥有数百万个用户，那么这些ISP通常要使用缓存代理。缓存代理位于你的 Web浏览器与Web服务器之间。当你检索一个Web页时，检索请求先送到缓存代理那里，然后由缓存代理为你在 Internet上取出该Web页，再将该Web页发送给你的浏览器，此前它要为自己将该 Web页的拷贝存储起来（见图23-5）。如果同一个域中的另一个人想要检索该 Web页，缓存代理就不必到Internet上去检索这一页，它可以使用保存在缓存中的拷贝。

图23-5 代理服务器从它的缓存中检索Web页



存储Web页拷贝的代理服务器人为地降低了访问次数计数器指示的访问次数。奇怪的是，它还使remote_host值多次重复，因为该Web页被许多人所检索。

大公司和大学中的 Web冲浪者常常位于作为缓存代理的防火墙的后面。从这些站点之一检索的每一页都有可能没有计入访问次数计数器，因为它被缓存代理挡住了。

23.2.1 编写一个访问次数计数器程序

阅读了上一节内容后，如果你接着读下去，一定有兴趣为你的 Web页编写一个访问次数计数器程序。访问次数计数器有两个基本类型，一种是简单的文本计数器，另一种是图形计数器。下面介绍的第一个计数器例子是文本计数器，第二个计数器是图形计数器，并且我们还讲述制作非常出色的访问次数计数器的一些思路。

若要使用该访问次数计数器，请将它用作服务器端的包含程序的一部分，我们在第 20学时中已经讲过这种包含程序。如果你调用访问次数计数器 CGI程序hits.cgi，可以使用下面这个SSI，将它纳入任何Web页：

```
<!--#exec cgi="/cgi-bin/hits.cgi"-->
```

程序清单23-3显示了该访问次数计数器的源代码。

程序清单23-3 访问次数计数器程序

```
1:  #!/usr/bin/perl -w
2:
3:  use strict;
4:  use Fcntl qw(:flock);
5:  use CGI qw(:all);
6:
7:  my $semaphore_file='/tmp/webcount_lock';
8:  my $counterfile='/web/httpd/countfile';
9:  sub get_lock {
```

```
10:      open(SEM, ">$semaphore_file")
11:          || die "Cannot create semaphore: $!";
12:      flock(SEM, LOCK_EX) || die "Lock failed: $!";
13:  }
14:  # Function to unlock
15:  sub release_lock {
16:     close(SEM);
17:  }
18:  get_lock();  # Get a lock, and wait for it.
19:  my $hits=0;
20:  if ( open(CF, $counterfile) ) {
21:      $hits=<CF>;
22:      close(CF);
23:  }
24:  $hits++;  # Increase the hits by 1.
25:  print header;
26:  print "You have had $hits visitors";
27:
28:  open(CF, ">$counterfile") || die "Cannot open $counterfile: $!";
29:  print CF $hits;
30:  close(CF);
31:
32:  release_lock();  # Release the lock
```

第18行：这里需要一个锁，因为访问次数计数器文件可能被许多进程同时读取和写入。

第20~23行：$counterfile中的文件内容被读取。它是迄今为止的访问次数。

第28~30行：访问次数计数器的内容被重新写回到 $counterfile中的文件。

第32行：最后，锁被释放。

程序清单23-3中的大部分代码你并不会感到有什么特殊。但是请注意，它使用了文件锁，并且这个示例程序遵循第15学时中介绍的文件锁定原则。

当两个人几乎同时加载Web页时，就有必要对文件加锁。如果对 Web访问次数计数器文件的读取和写入操作稍稍失去同步，那么计数器的数字就会增加得太快或太慢，也可能会产生一个受到破坏的文件。这些结果将会进一步降低计数器的准确性。

23.2.2 图形访问次数计数器

若要改进访问次数计数器，可以采取 3种不同的方法。首先，可以制作一个图形，代表计数器的每个可能的值，并且根据需要来显示该图形。如果你接到多个访问者对 Web站点的访问请求，那么这种办法比较费时。

第二种方法是让一个 Perl CGI程序生成必要的图形，以便显示访问次数计数器本身。CPAN中的GD模块可以用于以Perl程序来创建图形，因此你可以将它用于这个目的。不过关于GD模块的具体特性不在本书讲解的范围之内。

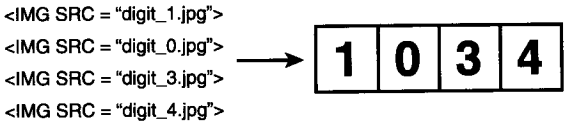
最容易的方法是创建 10个图形，分别代表 0至9的10个数字。然后，当计数器中的数字递增时，你的程序只需输出带有＜ IMG＞标记的HTML，将数字放入正确的位置（见图 23-6）。当然，必须创建代表数字的图形。程序清单 23-4中的Perl CGI程序将图形命名为 digit_0.jpg，digit_1.jpg，直至digit_9.jpg。

若要使用图形访问次数计数器，可以将它用作服务端的包含程序的一部分，如第 20学时中描述的那样。如果你调用访问次数计数器的 CGI程序graphical_hits.cgi，你就可以将它纳入任何Web页，如下所示：

```
<!--#exec cgi="/cgi-bin/graphical_hits.cgi"-->
```

程序清单23-4显示了该图形访问次数计数器程序的源代码。

图23-6 图形访问次数计数器的输出



程序清单23-4 图形访问次数计数器的程序

```
1:   #!/usr/bin/perl -w
2:
3:   use strict;
4:   use Fcntl qw(:flock);
5:   use CGI qw(:all);
6:
7:   my $lockfile='/tmp/webcount_lock';
8:   my $counterfile='/web/httpd/countfile';
9:   my $image_url='http://www.server.com/images';
10:
11: sub get_lock {
12:    open(SEM, ">$lockfile")
13:        || die "Cannot create semaphore: $!";
14:    flock(SEM, LOCK_EX) || die "Lock failed: $!";
15: }
16: sub release_lock {
17:    close(SEM);
18: }
19:  get_lock();  # Get a lock, and wait for it.
20:  my $hits=0;
21:  if ( open(CF, $counterfile) ) {
22:      $hits=<CF>;
23:      close(CF);
24:  }
25:  $hits++;
26:
27:  open(CF, ">$counterfile") || die "Cannot open $counterfile: $!";
28:  print CF $hits;
29:  close(CF);
30:  release_lock();  # Release the lock
31:
32:  # Now, create the <IMG> tags.
33:  print header;
34:  foreach my $digit (split(//, $hits)) {
35:      print "<IMG SRC=$image_url/digit_$digit.jpg>";
36:  }
```

程序清单23-4实际上与程序清单23-3相同，只有某些很小的修改。

第9行：这行代码在$image_url中包含了构成数字的各个图形的基本URL。请记住，它必须是浏览器加载图形时查看的URL，而不是到达本地磁盘上的图形的路径。

第34~35行：访问次数计数器中的数字 $hits对每个字符进行分割，再赋予 $digit，每次赋予一个数字。然后为每个数字输出＜IMG＞标号。

23.3 课时小结

在本学时中，我们讲述了在 Web页上实现动画的两种方法。可以使用服务器推送技术，

迫使浏览器连续更新 Web页。如果这种方法不行，或者在某种浏览器上无法实现，可以使用客户机拖拉技术，获得类似的效果。然后介绍了访问次数计数器，并且说明了计数器为什么不那么准确。

23.4 课外作业

23.4.1 专家答疑

问题：服务器推送技术不起作用，为什么？

解答：许多因素会导致服务器推送技术不起作用。首先，浏览器必须支持服务器推送技术；第二，Web服务器必须使用未分析标题；最后，你的 CGI程序必须正确地运行。如果从命令行提示符处以交互方式运行CGI程序，应该确保它的输出按固定时间间隔产生，并且输出必须正确。

问题：如果访问次数计数器很不准确，有没有别的办法可以用来测定对 Web站点的访问次数？

解答：几乎没有。查看服务器日志与使用访问次数计数器同样不可靠。测定访问站点次数的方法之一是使用实现重定向（参见第 20学时）的点击链接，另一种方法是让访问者填写一个窗体。使用HTML窗体中的POST方法，是避免你的Web页被缓存代理进行缓存的惟一完全可靠的方法，它可以确保POST方法提交的窗体不能被任何代理进行缓存。

23.4.2 思考题

1) 为了执行服务器推送操作，需要使用 CGI模块中的哪些函数？
   a.multipart_start和multipart_end
   b.multipart_init,multipart_start和multipart_end
   c.push_start和push_end
2) 所有浏览器都支持客户机拖拉技术，因为它是 HTML标准的组成部分。
   a.是。
   b.否。
3) 缓存代理能够保证不对何种类型的 Web页进行缓存？
   a.来自使用POST方法的HTML窗体的答复页。
   b.服务器推送的内容。
   c.CGI程序的任何输出。

23.4.3 解答

1) 答案是b。multipart_init用于使浏览器准备接收多部分构成的 Web页。multipart_start和multipart_end用于给每个Web页做上开始和结束标记。

2) 答案是b。完全不是。＜META＞标记可以被浏览器忽略，这是 HTML标准规定的。另外，使用header函数中的-Refresh选项并不能保证浏览器按需要重新加载 Web页，浏览器也有一个该命令的选项。

3) 答案是a。原因已经在“ 专家答疑 ”这一节中作了解释。

23.4.4 实习

- 修改程序清单23-3（或程序清单23-4）中的访问次数计数器程序，为不同类型的浏览器保存不同的计数。为此，你必须为你要跟踪的每种浏览器建立不同的文件。不要忘记为你不能识别其身份的浏览器保留一个额外的文件。