第18学时　基 本 窗 体

当你浏览Web时，肯定会填写几个HTML窗体。HTML窗体，比如电子邮件Web窗体、购物窗体、宾客留言簿、在线拍卖窗体、邮件列表和订单窗体等，可以用于搜集信息，如 Web浏览器用户的登录信息和Web站点的首选设置等。

当用户点击这些窗体上的Submit（提交）按钮时，将会出现什么情况呢？几乎在所有情况下，该窗体的数据都会传递给一个CGI程序。在本学时中，我们将要介绍如何从窗体中取出数据，如何在你的CGI程序中对数据进行操作。

在本学时中，你将要学习：

- 如何处理Perl CGI程序中的基本窗体。
- 如何调试CGI窗体。
- 如何编写更加安全的CGI程序。

18.1　窗体是如何运行的

你肯定使用过Web上的窗体，甚至知道窗体是如何布局的，以及它们是如何运行的。但是，如果要确保你处理的是同一个Web页，那么就必须了解HTML窗体的基本知识。

18.1.1　HTML窗体元素概述

在你开始学习窗体如何运行之前，首先应该了解 HTML是如何展示窗体的，以及窗体中的所有元素起着什么样的作用。

本书中介绍的HTML不应该被视为最典型的HTML。本书中讲述的HTML足以展示必要的CGI特性，但是没有涉及到太多其他的东西。在本书的代码例子中展示的 HTML中没有使用 <HEAD>或<BODY>标记，也没有使用<DOCTYPE>标记。此外，屏幕都非常简单朴素，你可以添加自己的 HTML，使屏幕更加生动和完整。

HTML窗体是HTML文档的一个部分，用于接收用户的输入。当浏览器加载包含窗体的HTML文档时，各个不同的HTML标号便在Web页上建立各个用户输入区域。用户的输入被放入各个窗体元素中，比如复选框、单选按钮、选项菜单和文本输入项元素。当用户使用 Web浏览器对输入元素的操作完成后，窗体通常被提交给 CGI程序，以便进行处理。

程序清单18-1显示了一个创建的典型HTML窗体。

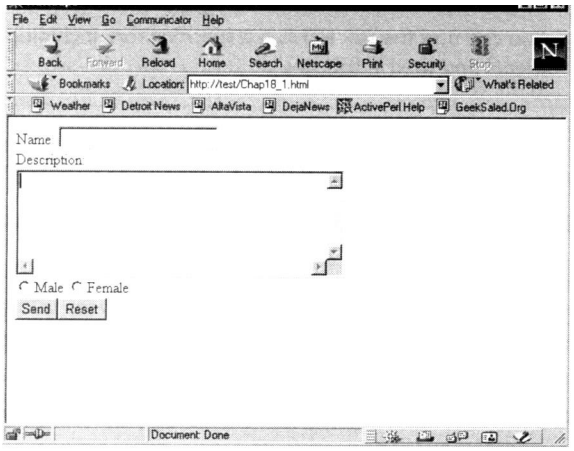
程序清单18-1　一个小型HTML窗体

```
1: <FORM action="http://www.server.com/cgi-bin/submit.cgi" method="get">
2: <INPUT TYPE="text" name="name">
3: <TEXTAREA name="description" rows=5 cols=40>
4: </TEXTAREA>
5: <INPUT type="radio" name="sex" value="male">Male
6: <INPUT type="radio" name="sex" value="female">Female
```

```
7: <BR>
8: <INPUT type="submit" value="Send"><INPUT type="reset">
9: </FORM>
```

图18-1给出了：Netscape浏览器中显示的程序清单18-1的窗体。

图18-1 Netscape中显示的程序清单18-1的窗体



<FORM>标记用于设定完整的HTML文档中的窗体的开始。method属性用于设定该窗体是使用GET还是POST方法来提交窗体。如果这个属性没有设定，浏览器将使用GET方法将窗体提交给CGI程序。GET与POST方法的差异将在后面说明。action属性用于设定接收窗体数据的CGI程序的URL。

<INPUT>标记用于为用户提供一个输入域，在这里，它是一个空白文本框。该文本框被赋予一个名字，它恰好是“ name ”。

<TEXTAREA>标记用于使浏览器显示一个多行文本框，以便接收输入的数据。值得注意的重要属性是name属性，在这里，该域的名字是 description。HTML窗体中的每个元素都必须拥有不同的name属性。当CGI程序被赋予该窗体以便进行处理时， name属性用来区分各个域。

“ 每个属性都有它自己的名字 ”这一原则有一个例外，那就是单选按钮。单选按钮按小组放在一起。单选按钮组中每次只能选定一个按钮。每个单选按钮组都有它自己的 name属性。

最后显示submit（提交）按钮。当用户单击该按钮时，窗体的值就被传递给 CGI程序，以便进行处理，这将在下一节中介绍。

HTML 4.0的技术规范包含了很少的不同窗体元素类型，因此，我们不想在本书中将它们全部完整地加以介绍。例如，许多窗体元素包含了一些属性，以便安装窗体元素的某些特性，比如前面介绍的窗体中的 TEXTAREA中的rows和cols。在本书的其他学时中，每当使用HTML窗体元素时，都只使用最基本的属性。

在网址http://www.w3c.org上，你可以找到 HTML 4.0完整的技术规范，包括有效的窗体及其属性。

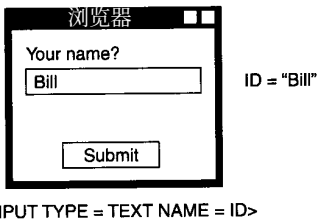
18.1.2 单击submit时出现的情况

当用户在他的Web浏览器上填写窗体信息时，将会发生一连串的事件：

- Web浏览器接收窗体上的数据，放入名字与值对中（见图 18-2）。例如，在这个示例窗体中，名字为 body的域接收了文本输入域的值。名字为 sex的域将接收单选按钮的值。

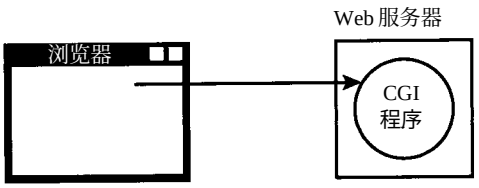
Web浏览器在发生任何情况之前执行所有这些操作。

图18-2 浏览器对数据与域的名字进行匹配



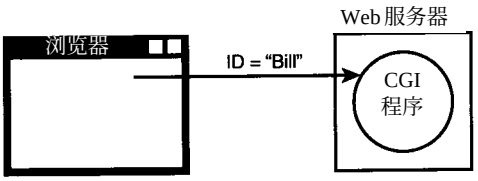
- 对窗体域的action部分的URL进行访问。这是CGI程序的URL（见图18-3）。

图18-3 浏览器与服务器进行联系



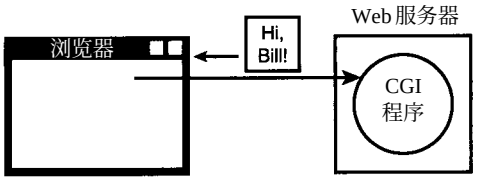
- 使用CGI方法GET或POST之一，窗体上的域的名字和值被传送到CGI程序（见图18-4）。你不必过分担心这个传输机制。

图18-4 数据被送往服务器



- CGI程序接收这些值，并生成一个应答，并将应答送回给浏览器（见图 18-5）。该应答可以是个HTML页，或者包含另一个窗体的 HTML页，也可以是转到另一个 URL的HTML或者是CGI程序能够生成的任何其他东西。

图18-5 Web服务器的CGI程序作出的应答



18.2 将信息传递给你的CGI程序

当由于窗体的提交而使CGI程序运行时，从窗体传递过来的域的名字和值（称为参数）必须由CGI程序来处理。这是使用param函数来处理的。

如果没有任何参数，那么 param函数返回传递到CGI程序中的域的名字。如果 CGI程序接收到程序清单18-1中的窗体，param函数将返回body、sex、name和submit。

如果带有参数，param函数返回该参数的值。例如，param(‘sex’)将根据你的选择，返回单选按钮的值male或female。

程序清单18-2包含了用于输出这些参数的简短的 CGI程序。

程序清单18-2 用于输出参数的CGI程序

```
1: #!/usr/bin/perl -w
```

```
2: use strict;
3: use CGI qw(:standard);
4:
5: print header;
6: print "The name was", param('name'), "<BR>";
7: print "The sex selected: ", param('sex'), "<BR>";
8: print "The description was:<BR>", param('description'),
9:       "<P>";
```

如果param函数指定的参数没有用于该窗体，则param返回undef。

GET与POST方法

在程序清单18-1中的窗体内，<FORM>标号拥有一个属性，称为method。Method属性用于设定Web浏览器应该如何将数据传送到Web服务器。目前可以使用的方法有两个。

第一个方法称为GET，如果你在<FORM>标号中没有设定方法，那么这就是默认的方法。使用GET方法，通过在URL中对窗体值进行编码，就可以将这些值传递给CGI程序。当你在Web上冲浪时，可能看到下面这样的URL：

```
http://www.server.com/cgi-bin/sample.pl?name=foo&desc=Basic%20Forms
```

CGI程序运行时，能够将URL的剩余部分解码，使之分解成域和值。当你调用param函数时，它实际上也是进行这样的操作。你不应该试图自己对这些值进行解码。param函数能够全面地进行这项操作，你没有理由使用别的方法来提取这些值。

另一个方法是POST，它产生的结果完全相同，但使用的手段不同。不是将所有的窗体值编码后放入URL，而是通过访问Web服务器，然后将HTML窗体值传送给CGI程序，作为其输入数据。另外，现在你不必清楚地了解这个过程究竟是如何运行的，CGI模块会为你处理好这个问题。只要调用param函数，就可以读取这些值，对它们解码，然后将它们传递给你的程序。

> 你可能已经从Internet下载了一些CGI程序，或者在别的书中见过一些例子，通过对环境变量QUERY_STRING进行解码，或者使用变量REQUEST_METHOD来确定窗体是使用GET还是POST方法。这些程序试图重复进行在标准CGI模块中已经做的工作，但是也可能没有这样做。你应该避免自己执行这项操作。

那么究竟你应该选择哪一种方法呢？每种方法都有它的优点和缺点。GET方法使得Web浏览器能给生成Web页的特定URL做上书签。例如下面的URL

```
http://www.server.com/cgi-bin/sample.pl?name=foo&desc=Basic%20Forms
```

可以做上书签，并且总是由浏览器返回。从CGI程序sample.pl的角度来看，它不知道你刚刚是否查看了该窗体。它像往常一样接收通常的CGI参数。如果能够使用GET方法的URL编码值来反复调用一个CGI程序，这称为幂等性。

但是你可能不是特别想使浏览器能在你的站点中做上书签，以便直接运行你的CGI程序，坦率地说，用于以GET方法启动CGI程序的URL是很讨厌的。

POST方法根本不对URL中的窗体数据进行编码，当它为Web页进行处理时，它依靠浏览器发送数据。但是，由于数据并没有被编码后放入URL，因此你无法使用POST方法给CGI程序生成的Web页做上书签。

18.3 Web安全性

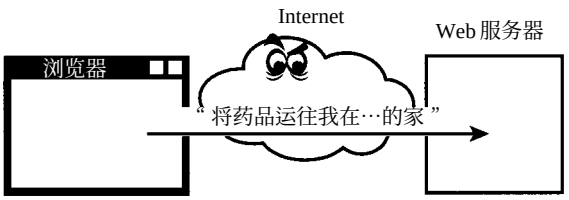
在你将CGI程序放到World Wide Web上去之前，必须了解下面几个问题。通过将CGI程序放在Web页上，你就为远程用户（使用Web浏览器）赋予对你的系统的有限访问权。使用普通HTML文档，他们只能从你的Web站点检索静态文档。但是，使用CGI程序，他们就能在你的Web服务器上运行程序。

懂得如何编写安全而保密的CGI程序后，你与你的Web服务器管理员一定会感到更加高兴。编写这样的程序并不难，只需要掌握几条简单的注意事项。

18.3.1 建立传输明码文本的连接

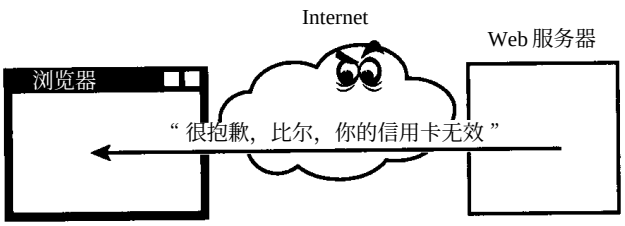
当Web浏览器从Web服务器中检索Web页时，HTML是通过一个明码文本信道来发送的（见图18-6）。这意味着当数据一路通过Internet时，它并不进行加密、编码，否则就无法被对方理解。

图18-6 明码文本被传送到服务器



用户填入窗体然后提交给你的CGI程序的数据，传输时所用的协议与初始Web页使用的协议相同。任何人只要访问窗体，就可以查看它的所有域（见图18-7）。

图18-7 服务器用明码文本作出应答



用明码传送数据时存在的问题确实是你应该担心的问题。Internet不是一个安全的地方，在Web浏览器与Web服务器之间的线路上的任何人都能够窃听线路上来回传送的信息。

应该记住，决不应该以普通CGI窗体来发送下列几种类型的数据：

- 任何形式的口令。
- 个人信息（社会保险号，电话号码）。
- 财务信息（帐号，个人身份识别号，信用卡号码）。

请记住这些基本原则。决不要在Internet上发送你不会写在明信片上任何信息。

你会说：“等一等，我曾经在Internet上看到一些窗体，要求查询所有上述信息，并且说这是安全的。”使用某些辅助工具，可以在Web上执行相当安全的事务处理。若要执行安全的Web事务处理，实际上必须对浏览器／服务器之间的全部会话进行加密。这是通过运用http协议的安全版本https来实现的。

18.3.2 注意不安全数据

在编写安全的CGI程序时需要考虑的另一个问题是：你编写的程序将根据 Web页提供给你的输入来执行Perl命令。Internet或者你的Intranet（专用网）上有许多人属于不良之徒，他们以损害你的Web服务器为乐，并因此而感到自己了不起。还有一些并无恶意的用户可能不小心将无效数据发送给你的CGI程序。

请看程序清单18-3中的HTML窗体和程序清单18-4中的CGI程序。

程序清单18-3 目录清单Web窗体

```
1: <FORM action="/cgi-bin/directory.cgi">
2: What directory to list?
3: <INPUT TYPE=text NAME=dirname>
4: <INPUT TYPE=submit name=submit value="Run This">
5: </FORM>
```

程序清单18-4 名字为directory.cgi的不安全CGI程序

```
1: #!/usr/bin/perl -w
2: # Do NOT use this CGI program, it's very insecure
3: use strict;
4: use CGI qw(:all);
5:
6: print header;
7: my $directory=param('dirname');
8: print `ls -l $directory`;  # Do a directory listing
```

程序清单18-3为用户提供了一个简短的窗体，用于接收一个目录名，再将它传送给称为directory.cgi的CGI程序。在程序清单 18-3中，directory.cgi程序接收该目录，并为 DOS/windows用户对它执行ls -1命令，它与dir等价，为用户提供一个目录列表。

这种类型的程序使得远程Web冲浪者能查看你的整个目录结构。CGI程序并不检查该目录名是什么，如果浏览器想要查看你的敏感数据，它就可以查看。

更重要的一个问题是：$directory可能根本不包含任何目录。如果 Web浏览器为dirname发回了值/home；cat ／etc/passwd，这时，CGI程序运行的命令将类似下面的形式：

```
ls -l /home; cat /etc/passwd
```

这个命令将能有效地将系统的口令文件拷贝发回给 Web浏览器。实际上，所有UNIX shell命令或MS-DOS命令都可以这样运行。如果你的Web服务器尚未正确地安装，那么任何用户都可以上Internet。

Perl拥有一个机制，可以帮助你避免做这样的傻事。＃！行上的 -T开关可以激活数据受感染特性。当数据从外部信息源（如文件句柄、网络套接字、命令行等）接收过来时，它就被做上“ tainted(受感染) ”标记。受感染的数据不能用在反引号、系统函数调用（如 open函数）、系统命令或可能破坏安全性的其他地方。

当受感染检查正在进行时，不能将 open函数、system函或反引号用于你的Perl程序，除非首先明确设置PATH环境变量。

程序清单18-5显示了这个程序的更加安全的版本。

程序清单18-5 directory.cgi程序的更安全版本

```
1: #!/usr/bin/perl -wT
```

```
2: # tainting is enabled!
3: use strict;
4: use CGI qw(:all);
5:
6: print header;
7: # Explicitly set the path to something reasonable
8: $ENV{PATH}='/bin:/usr/bin';
9: my $dir=param('dirname');
10: # Only allow directory listings under /home/projects
11: if ($dir=~m,^(/home/projects/[\w/]+)$, ) {
12:     $dir=$1;    # This "untaints" the data, see "perldoc perlsec"
13:     print 'ls -l $dir';
14: }
```

若要了解关于受感染的数据、如何消除数据的感染以及如何编写安全的Perl程序的详细信息，请参见Perl产品包含的Perlsec手册页。

18.3.3 从事无法执行的操作

HTML／CGI窗体也可能遭到另一种情况的损害。请看程序清单18-6中的HTML窗体。

程序清单18-6 一个简单的窗体

```
1: <FORM action="/cgi-bin/doit.cgi">
2: Please type in your favorite color:
3: <INPUT TYPE=text length=15 name=color>
4: <INPUT TYPE=submit value="Submit color">
5: </FORM>
```

在这个窗体中，color域允许的最大宽度是15。这对吗？大概差不多。HTML技术规范规定，文本域的length最多允许这么多的字符。但是，浏览器可能发生故障，有人可能故意在这个域中放入15个以上的字符，方法是不使用你的窗体，或者创建一个新窗体。

如果你希望某个域拥有一个特定值，请不要依赖HTML、Java或JavaScript来保证这个值的正确性。例如，如果color域的绝对限值应该是15，那么Perl程序可以像下面这样对它进行处理：

```
my $color=param('color');       # Get the original field value
$color=substr($color, 0, 15);   # Get just the first 15 characters...
```

18.3.4 拒绝服务

通过拒绝服务，任何Web服务器的性能都会受到削弱。由于Web服务器是代表远程用户处理访问请求的，因此，如果远程用户发出的访问请求太多，Web服务器就会不堪重负。这种做法有时是恶意的，而且常常是恶意的。许多时候，一些公司为Web提供了许多服务，结果为了响应用户的访问请求，负担太重，因此不得不关闭这些服务，重新考虑自己的做法。

静态HTML页或CGI程序也会出现拒绝服务的情况。

为了防止拒绝服务的问题，你有时会感到无能为力，除非手头拥有足够的服务系统，能够处理浏览器的负荷。如果你的CGI程序花费很长时间来执行或使用相当一部分系统资源（如文件访问的频率，CPU使用的密度），以便使程序能够运行，服务器将很容易受到拒绝服务的攻击。你应该设法尽量缩小你运行的CGI程序的规模，并使之更快地运行。

18.4 宾客留言簿

这个例子使你能够为Web站点编写定制的宾客留言簿。宾客留言簿是个 HTML窗体，在这个窗体中，用户可以指明来宾的名字并配有一些说明。宾客留言簿可以用来收集关于某个问题的反馈信息，作为一个简单的消息板，或者将问题提交给帮助桌。数据保存在一个文件中，并且可以在窗体信息填满后显示出来。它也可以在它自己的 Web页上显示。

程序清单18-7提供了一个简短的HTML代码段，它展示了一个用于虚构帮助桌的宾客留言簿窗体。你当然可以修改这个窗体，使之适合你自己的需要。

程序清单18-7 帮助桌窗体

```
1:  <FORM action="/cgi-bin/helpdesk.cgi" name="helpdesk">
2:  Problem type:
3:  <INPUT TYPE=radio name=probtype value=hardware>Hardware
4:  <INPUT TYPE=radio name=probtype value=software>Software
5:  <BR>
6:  <TEXTAREA name=problem rows=10 cols=40>
7:  Describe your problem.
8:  </TEXTAREA>
9:  <BR>
10: Your name:
11: <INPUT TYPE=text width=40 name=name><BR>
12: <INPUT TYPE=submit name=submit value="Submit Problem">
13: </FORM>
```

这个帮助桌窗体需要运行一个名叫 /cgi-bin/helpdesk.cgi的CGI程序。程序清单18-8显示了这个CGI程序。如果你想要将该CGI程序放在另外某个位置，或者将它称为另一个名字，请务必将正确的URL输入程序清单18-7的帮助桌窗体中。

程序清单18-8 帮助桌CGI程序

```
1: #!/usr/bin/perl -wT
2: use strict;
3: use CGI qw(:all);
4: use Fcntl qw(:flock);
5:
6: # Location of the guestbook log file.  Change this to suit your needs
7: my $gbdata="c:/temp/guestbook";
8: # Any file name will do for semaphore.
9: my $semaphore_file="/tmp/helpdesk.sem";
10:
11: # Function to lock (waits indefinitely)
12: sub get_lock {
13:     open(SEM, ">$semaphore_file")
14:         || die "Cannot create semaphore: $!";
15:     flock SEM, LOCK_EX;
16: }
17: # Function to unlock
18: sub release_lock {
19:     close(SEM);
20: }
21:
22: # This function saves a passed-in help desk HTML form to a file
23: sub save {
24:       get_lock();
25:       open(GB, ">>$gbdata") || die "Cannot open $gbdata: $!";
```

```
26:       print GB "name: ", param('name'), "\n";
27:       print GB "type: ", param('probtype'), "\n";
28:       print GB "problem: ", param('problem'), "\n";
29:       close(GB);
30:       release_lock();
31: }
32: # This function displays the contents of the help desk log file as HTML,
33: # with minimal formatting.
34: sub display {
35:       open(GB, $gbdata) || die "Cannot open $gbdata: $!";
36:       while(<GB>){
37:            print "<B>$_</B><P>";  # The name
38:            my($type,$prob);
39:            $type=<GB>;
40:            $prob=<GB>;
41:            print "$type<P>";
42:            print "$prob<BR><HR>";
43:       }
44:       close(GB);
45: }
46:
47: print header;
48: # The parameter 'submit' is only passed if this CGI program was
49: # executed by pressing the 'submit' button in the form in listing 18.7
50: if (defined param('submit')) {
51:       save;
52:       display;
53: } else {
54:       display;
55: }
```

程序清单18-8中的大部分代码是你已经熟悉的Perl程序，不过请特别注意下列几个问题：

- get_lock()和release_lock()这两个函数对于这个窗体是绝对必要的。对于任何一个CGI程序来说，你始终必须假设任何时候都可能有CGI程序的多个实例正在运行。写入帮助桌日志文件的helpdesk.cgi的多个实例将会出错，因此在将信息写入文件之前，该文件应被锁定。在读取文件之前，它不锁定，因为一边读取日志文件，又一边写入文件，那将是很糟糕的。
- 这个CGI程序有两个目的。当作为程序清单 18-7中的窗体的目标操作来调用时，它将新项目写入日志文件。当不使用该窗体来调用该程序时，它只显示日志文件的内容。

18.5 课时小结

在本学时中，我们介绍了 HTML窗体与CGI程序如何进行交互操作，如何使用 CGI模块的param函数使你的CGI程序能够转换窗体的内容。另外，还介绍了怎样才能使你的 CGI程序更加安全，如何处理受感染的数据。我们还介绍了一个简单的 CGI宾客留言簿应用程序，你可以对它定制和修改，以便适应你自己的需要。

18.6 课外作业

18.6.1 专家答疑

问题：我无法使用窗体的提交功能，老是出错，怎么办？

解答：请使用第17学时中介绍的CGI调试指南，找出存在的问题。仅仅因为它是个窗体，

并不意味着调试该窗体与调试普通CGI程序有什么不同。

问题：我在Internet上看到了这个出色的程序，但是我不懂得为什么它试图使用$ENV{QUERY_STRING}来获得窗体参数。为什么？

解答：因为该程序的开发人员决定放弃该CGI模块的窗体处理功能。这个情况说明它可能是该CGI模块以前的一个非常老的Perl程序，也可能程序开发人员决定使用他自己的窗体处理代码。不管属于哪种情况，这表明你应该用警惕的目光观察这个程序，并且小心地使用它。

问题：我通过命令行提示符运行程序，其#！行上有一个选项 -T，我得到一条出错消息Too late for -T option（运行-T选项太晚了），然后程序停止运行了。为什么？

解答：你应该尽快将-T选项赋予Perl程序，这样它就知道要去寻找受感染的数据。当你的程序中的#！行被处理时，这就太晚了， Perl已经处理了你的没有感染的命令行选项。若要从命令行提示符来运行Perl程序，例如在调试程序中运行，你也必须在命令行提示符上设定 -T：

```
Perl -T -d foo.cgi
```

问题：Perl的数据受感染功能是否能使我避免在 CGI程序中犯一些愚蠢的错误？它们现在是否能够确保安全？

解答：没有一个CGI程序是绝对安全的。 Perl的数据受感染功能在很大程度上可使你不犯愚蠢的错误，不过它们无法保证你编写出安全的程序。

18.6.2 思考题

1) 在数组上下文中，不带参数的param函数将返回
   a. undef。
   b. 窗体元素的数目。
   c. 窗体元素名的列表。
2) 如果你使用CGI模块，POST与GET方法之间的差别是清楚的。
   a. 是。
   b. 否。
3) HTML窗体上的password域的输入类型是安全的，因为它在发送前会对口令进行加密。
   a. 是。
   b. 否。

18.6.3 解答

1) 答案是c。如果不带参数，param将返回来自提交的窗体的元素名列表。
2) 答案是a。
3) 答案是a。在普通HTTP和CGI程序中，所有窗体域都是以明码文本传送的，因此是不保密的。口令域输入类型只在你键入口令时将该域隐藏起来而已。

18.6.4 实习

- 对帮助桌窗体稍作修改。将时间戳添加给每个项目，并且给输出添加某些颜色。
- 问题：display()函数从最老的项目开始输出帮助桌窗体中的各个项目。请修改 display()函数，使之首先输出最新的项目。