CHAPTER 20

The Guestbook

OVERVIEW

Our guestbook script, based on the freeware guestbook written by Matt Wright called **guestbook.pl**, allows users to dynamically manipulate a guestbook HTML file by adding their own entries to the document. Thus, you can create a virtual guestbook that visitors can sign, leaving their contact information and perhaps comments about your pages.

The guestbook is configurable so that you can specify what your guestbook file looks like and how the script-generated responses are displayed. Most of the configuration takes little more than a knowledge of HTML, so it is fairly easy to use.

If configured to do so, the guestbook application will mail the guestbook administrator the text of new entries as well as add them to the guestbook. The script will also respond to new entrants with a configurable "Thank you" message. Thus, there is no need to continually monitor the page.

Finally, the script comes with the capability of "four letter word" filtering for a child-safe guestbook. You can censor words by adding them to a list of "bad words." If a guest attempts to use one of the excluded words in a guestbook entry, the word will be censored.

INSTALLATION AND USAGE

The application should expand into your local CGI-executable directory in the root directory **Guestbook**, as diagrammed in Figure 20.1.

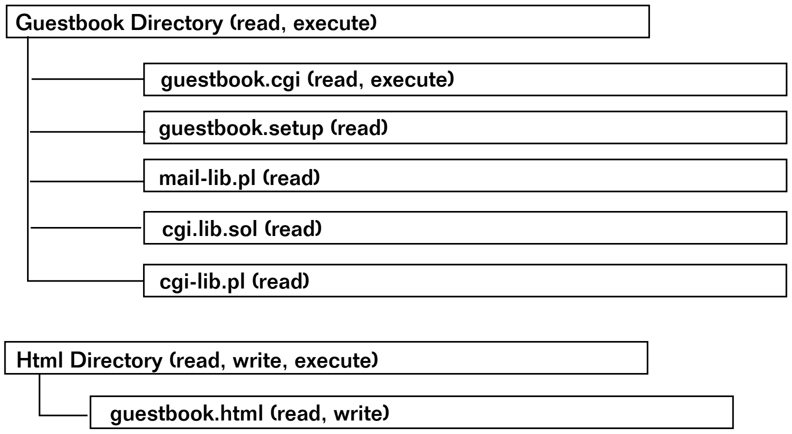

Guestbook Directory (read, execute)
guestbook.cgi (read, execute)
guestbook.setup (read)
mail-lib.pl (read)
cgi.lib.sol (read)
cgi-lib.pl (read)
Html Directory (read, write, execute)
guestbook.html (read, write)

Figure 20.1 *Guestbook directory structure.*

Guestbook is the application's root directory and must be readable and executable by the Web server. It contains five files: **cgi-lib.pl**, **guestbook.cgi**, **guestbook.setup**, **guestbook.html**, and **mail-lib.pl**.

guestbook.cgi is the main body of the application and must be readable and executable by the Web server. We will discuss this file more thoroughly in the "Design Discussion" section.

guestbook.setup is the file that contains the configurable option variables and the variables that must be set to the specifics of your system. This file must be readable by the Web server and will be discussed in the "Server-Specific Setup and Options" section.

cgi-lib.pl, **cgi-lib.sol**, and **mail-lib.pl** are library files described in Part Two. **cgi-lib.pl** is used to parse form data, **mail-lib.pl** is used to send notifications by E-mail, and **cgi-lib.sol** is used for its lock file routines. Each library file must be readable by the Web server.

guestbook.html is the HTML guestbook file, which clients can read and add to if they desire. In Figure 20.1, **guestbook.html** is contained in a directory called **Html**.

Html, however, is not included on the accompanying CD-ROM. It represents a generic HTML directory on your system. We diagram it here because you will need to move the distribution copy of **guestbook.html** (which we will discuss soon) to an HTML directory elsewhere on your system; most servers are configured to disallow HTML in the **cgi-bin** directory. **guestbook.html** must be readable and writable by the Web server and will be discussed in greater depth in the "Server-Specific Setup and Options" section. The directory containing **guestbook.html** must be readable, writable, and executable by the Web server. You cannot use the **Guestbook** directory, which must remain read and execute only, for this purpose.

Server-Specific Setup and Options

The Setup File

The primary functions of the setup file are to define server-specific variables and to define options. The setup file contains the following variables.

`$guestbookurl` is the URL of **guestbook.html**.

`$guestbookreal` is the location and name of **guestbook.html** on your server.

`$cgiurl` is the URL of this script.

`$cgi_lib_location`, `$cgi_sol_location`, and `$mail_lib_location` are the locations on the server of the three library files that accompany this script.

`@bad_words` is a list of words that you want to censor from your guestbook. Any word in this list will be removed before the guestbook is modified. If you do not want to censor anyone, set `@bad_words` to nothing as follows:

```
@bad_words = ();
```

`$mail` determines whether to E-mail the guestbook administrator when a new entry has been made. If `$mail` is set to 1, the guestbook administrator will be notified; if it is set to zero, no E-mail will be sent.

`$recipient` is the E-mail address of the guestbook administrator who should receive the E-mail notification, and `$email_subject` is the subject of that message.

`$linkmail` determines whether you want E-mail addresses in your guestbook to be clickable. If you want them to be, set this equal to 1; otherwise, set it to zero.

`$remote_mail` determines whether a thank you note is sent to the guest who signed the guestbook. If you set this equal to 1, the guest will receive a thank you note. Set it to zero and the note will not be sent.

`$allow_html` is set to allow or disallow the use of HTML tags in guestbook entries. If you set this to 1, guests will be able to use HTML. If you set it to zero, they will not be able to use HTML.

`@required_fields` is the list of fields that the guest MUST submit to add the guestbook entry.

As an example, the complete text of **guestbook.setup** is shown next:

```
$guestbookurl = "http://www.foobar.com/Guestbook/guestbook.html";
$guestbookreal = "/Guestbook/guestbook.html";
$cgiurl = "guestbook.cgi";
$cgi_lib_location = "./cgi-lib.pl";
$cgi_sol_location = "./cgi-lib.sol";
$mail_lib_location = "./mail-lib.pl";
@bad_words = ("darn", "fudge", "frack", "poopoo", "Bob Dole");
$mail = 1;
$recipient = 'selena@foobar.com';
$email_subject = "Entry to Guestbook";
$linkmail = 1;
$remote_mail = 1;
$allow_html = 1;
@required_fields = ("realname", "comments");
```

The Guestbook File

Because it must be read as an HTML file, **guestbook.html** must be placed in a directory from which the Web server is allowed to read HTML files.

Thus, if your **cgi-bin** directory is configured to disallow HTML files, you must move **guestbook.html** to an HTML-friendly directory. If your server displays HTML files from the directory containing the CGI executables, you must take care to put the HTML file in a subdirectory that is writable and not in the root **Guestbook** directory.

As time goes by, this file will expand with guestbook entries. At first, however, the file will look like the following HTML page with a hyperlink reference to **guestbook.cgi** for additions:

```
<HTML>
<HEAD>
<TITLE>Selena Sol's Guestbook</TITLE>
</HEAD>
<BODY BGCOLOR = "FFFFFF" TEXT = "000000">
<CENTER><H1>Selena Sol's Guestbook</H1>
</CENTER>
Thank you for visiting my homepage.  Feel free to
<A HREF =
"http://www.foobar.com/cgi-bin/Guestbook/guestbook.cgi">Add</a>
to my guestbook! <HR WIDTH = "75%">
<!—begin—>

</BODY>
</HTML>
```

NOTE

You also must change the hyperlink reference to guestbook.cgi to reflect your local directory structure. When clients click on the hyperlink to Add an Entry, they should be taken to the script that will handle things from there.

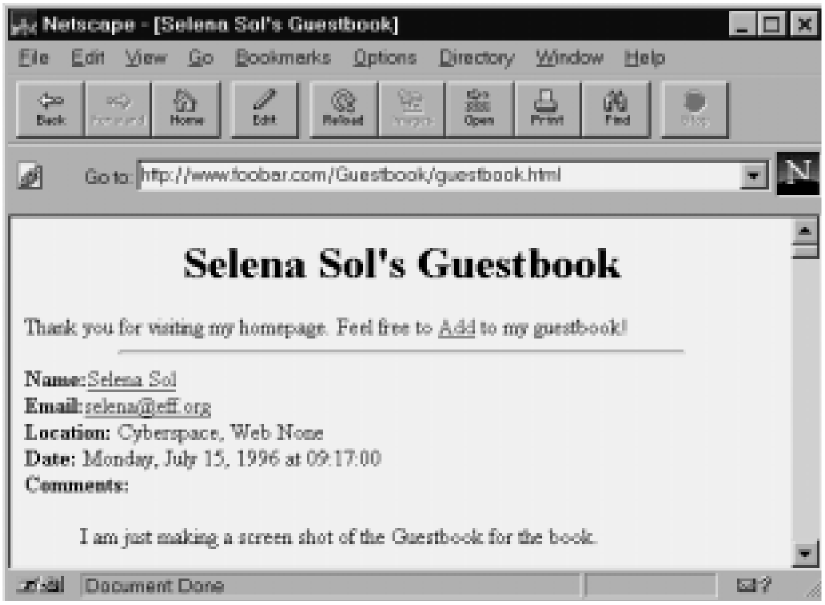
The initial guestbook is a simple, expected HTML file except for the following line:

```
<!—begin—>
```

This tag line tells the script where to add entries. We will discuss this later. For now, you should know what will happen when someone submits a guestbook entry, so the following listing shows one guestbook entry. Notice that each new entry is inserted after the `<!—begin—>` line.

```
<!—begin—>
<B>Name:</B><A HREF = "http://www.foobar.com/~erict">Selena
Sol</A><BR>
<B>Email:</B><A HREF =
"mailto:selena@foobar.com">selena@foobar.com</A><BR>
<B>Location:</B> Arlington, VA 22201<BR>
<B>Date:</B> Saturday, June 8, 1996 at 09:10:07<BR>
<b>Comments:</B><BLOCKQUOTE>This is a test</BLOCKQUOTE>
</BODY>
</HTML>
```

Figure 20.2 shows the **guestbook.html** interface on the Web.


Netscape - [Selena Sol's Guestbook]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Go to: http://www.foobar.com/Guestbook/guestbook.html
Document Done

Selena Sol's Guestbook

Thank you for visiting my homepage. Feel free to Add to my guestbook!

Name: Selena Sol
Email:selena@eff.org
Location: Cyberspace, Web None
Date: Monday, July 15, 1996 at 09:17:00
Comments:

I am just making a screen shot of the Guestbook for the book.

Figure 20.2 *The guestbook.html interface.*

RUNNING THE SCRIPT

Once you have configured the setup file, set all the appropriate permissions, and prepared your **guestbook.html** file, you can create a link to the guestbook so that clients can begin signing. The initial link should point to **guestbook.html** as follows:

```
<A HREF = "http://www.foobar.com/Guestbook/guestbook.html">The
Guestbook</A>
```

Clients can then read through guestbook entries. If they choose, they can click on the **Add an entry** link, which accesses the main script.

Design Discussion

Once you have configured the setup file, your guestbook is ready to go. The logic of the script is depicted in Figure 20.3.

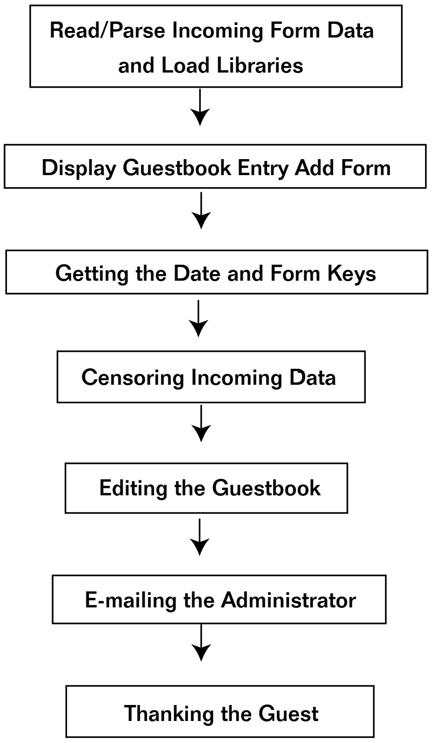

Read/Parse Incoming Form Data and Load Libraries
Display Guestbook Entry Add Form
Getting the Date and Form Keys
Censoring Incoming Data
Editing the Guestbook
E-mailing the Administrator
Thanking the Guest

Figure 20.3 *Script logic.*

The script begins by starting the Perl interpreter and printing the HTTP header.

```
#!/usr/local/bin/perl
print "Content-type: text/html\n\n";
```

Loading Supporting Files and Reading Incoming Form Data

Next, the script loads the supporting files and uses **cgi-lib.pl** to read the incoming form data. It sends `*form_data` as a parameter to the subroutine `ReadParse` in **cgi-lib.pl** so that the associative array of form keys and values comes back with a descriptive name rather than `%in`.

```
require "./guestbook.setup";
require "./$cgi_sol_location";
require "$cgi_lib_location";
require "$mail_lib_location";
&ReadParse(*form_data);
```

Displaying the Guestbook Entry Add Form

Next, the script determines what the client wants. If `$form_data{'action'}` equals `add` (client clicked on a button somewhere) or if (`||`) `$ENV{'REQUEST_METHOD'}` equals `GET` (client is accessing this script for the first time as a link and not as a submit button), then the script knows that the client is asking to see the form to add an item to the guestbook.

```
if ($form_data{'action'} eq "add" ||
    $ENV{'REQUEST_METHOD'} eq "GET")
  {
```

First, it prints the form's header using the `HERE DOCUMENT` method.

```
print <<"    end_of_html";
<HTML>
```

```
<HEAD>
<TITLE>Selena Sol's Guestbook (Add Form)</TITLE>
</HEAD>
<BODY>
<CENTER>
<H2>Add to my Guestbook</H2>
</CENTER>
Please fill in the blanks below to add to my guestbook.  The only
blanks that you have to fill in are the comments and name section.
Thanks!
<P><HR>
end_of_html
```

Next, the script uses the subroutine `output_add_form` at the end of this form to print the form. It then quits and lets the client submit data.

```
 &output_add_form;
 exit;
}
```

Figure 20.4 shows the add form on the Web.

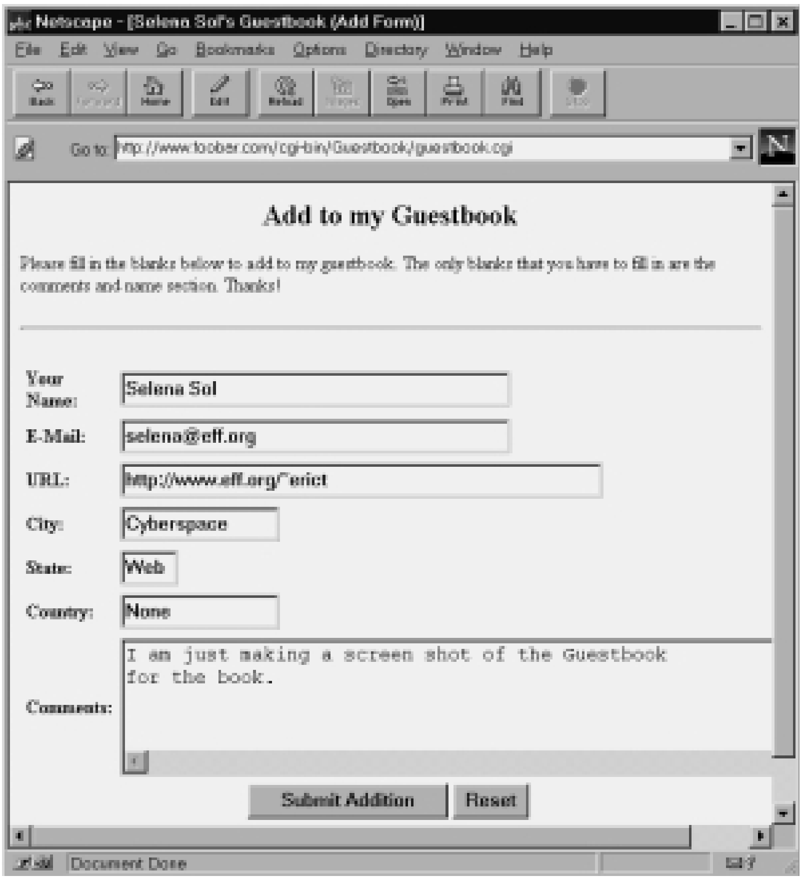
Getting Date and Form Keys

If the script gets to this point, it means that the client has filled out the add form and is submitting a new guestbook entry. The script uses the `get_date` subroutine at the end of this script to get the current date and time so that it can use it in the output.

```
$date = &get_date;
```

Then it creates an array of form variables by accessing the `keys` of the associative array `%form_data` given to it by **cgi-lib.pl**.

```
@form_variables = keys (%form_data);
```


Netscape - [Selena Sol's Guestbook (Add Form)]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Go to: http://www.foober.com/cgi-bin/Guestbook/guestbook.cgi
Document Done

Add to my Guestbook

Please fill in the blanks below to add to my guestbook. The only blanks that you have to fill in are the comments and name section. Thanks!

Your Name:	Selena Sol
E-Mail:	selena@eff.org
URL:	http://www.eff.org/~erict
City:	Cyberspace
State:	Web
Country:	None
Comments:	I am just making a screen shot of the Guestbook for the book.

Submit Addition | Reset

Figure 20.4 *Guestbook add form.*

Censoring Incoming Form Data

Next, the script checks to see whether it was asked to censor any particular words. For every variable sent to it from the form and for each word in the list of bad words, the script replaces (`=~ s/`), with case insensitivity off (`/gi`), every occurrence of the bad word (`$word`) with the word "censored." `$form_data{$variable}` should be equal to what the client filled in using the input boxes.

```
foreach $variable (@form_variables)
 {
 foreach $word (@bad_words)
   {
   $form_data{$variable} =~ s/\b$word\b/censored/gi;
   }
```

If the guestbook administrator has set `$allow_html` to zero (`!= 1`), it means that the administrator does not want the users to be able to use HTML tags. So the script deletes them.

```
 if ($allow_html != 1)
   {
   $form_data{$variable} =~ s/<([^>]|\n)*>//g;
   }
 }
```

Checking Required Fields

For every field that was defined in the list of required fields, the script then checks the form data to see whether that variable has an empty value. If it is empty, the script jumps to the subroutine `missing_required_field_data` at the end of this script, passing as a parameter the name of the field that was not filled out.

```
foreach $field (@required_fields)
  {
  if ($form_data{$field} eq "" )
    {
    &missing_required_field_data($field);
    }
  }
```

Editing the Guestbook

Now the script opens the guestbook HTML file and reads each of the lines into an array called `@LINES`. Thus, `@LINES` will contain every line of the guestbook file so that each array element will correspond to a line in the file. Then the script closes the guestbook file. Finally, it sets the variable `$SIZE` to the number of elements in the array, which is the same number of lines in the guestbook file.

```
open (FILE,"$guestbookreal") ||
die "Can't Open $guestbookreal: $!\n";
@LINES=<FILE>;
close(FILE);
$SIZE=@LINES;
```

The script opens the guestbook file again, but this time it opens the file for writing. In fact, it overwrites the existing guestbook file with new data using `>`. It also uses the `GetFileLock` subroutine in **cgi-lib.sol** to make sure that no one else writes to the guestbook at the same time.

```
&GetFileLock ("$guestbookreal.lock");
open (GUEST,">$guestbookreal") || &CgiDie ("Can't Open $guestbookreal");
```

Then the script goes through the `@LINES` array adding lines "back" to the guestbook file one by one and inserting the new entry along the way. For every line in the guestbook file (remember that `$SIZE` equals the number of lines), the script assigns the value of the line (`$LINES[$i]`) to the variable `$_`. It begins with the first line in the array (`$i=0`) and ends when it has gone through all the lines (`$i<=$SIZE`), counting by 1 all along (`$i++`). Then the script references the array in the standard form `$arrayname[$element_number]`.

```
for ($i=0;$i<=$SIZE;$i++)
  {
  $_=$LINES[$i];
```

If the line happens to include the `<!—begin—>` tag, the script knows that it must add a new entry. This is why it is essential that your **guestbook.html** have that line all on its own somewhere in the body when you initialize the guestbook. If the line is not there, the new entry will not be added; if there is other information on the same line, that information will be deleted.

```
if (/<!--begin-->/)
  {
```

The script then adds the entry. It first prints the line `<!–begin–>` so that it will be able to find the top of the guestbook again next time.

```
print GUEST "<!--begin-->\n";
```

Then it begins adding the guest's information. First, it prints the name of the guest. If the guest submitted a URL, it makes the name clickable to the URL. If the guest did not submit a URL, however, it just prints the name.

```
if ($form_data{'url'})
  {
  print GUEST "<B>Name:</B>";
  print GUEST "<A HREF = \"$form_data{'url'}\">$form_data{'real-
name'}</A>";
  print GUEST "<BR>\n";
  }
else
  {
  print GUEST "<B>Name:</B>
               $form_data{'realname'}<BR>\n";
 }
```

Next, the script prints the E-mail address of the guest, and, if you have set `$linkmail` to 1 in the setup file, it makes the E-mail link clickable.

```
if ( $form_data{'email'} )
  {
  if ($linkmail eq '1')
    {
    print GUEST "<B>Email:</B>";
    print GUEST "<A HREF = \"mailto:$form_data{'email'}\">";
    print GUEST "$form_data{'email'}</A><BR>\n";
    }
 else
    {
    print GUEST " $form_data{'email'}<BR>\n";
    }
  }
```

The script then prints the guest's address if he or she submitted the values.

```
if ( $form_data{'city'} )
  {
  print GUEST "<B>Location:</B> $form_data{'city'},";
  }
if ( $form_data{'state'} )
  {
  print GUEST " $form_data{'state'}";
  }
if ( $form_data{'country'} )
  {
  print GUEST " $form_data{'country'}<BR>\n";
  }
```

Finally, the script prints the date and the guest's comments.

```
 print GUEST "<B>Date:</B> $date<BR>\n";
 print GUEST "<b>Comments:</B>

 <BLOCKQUOTE>$form_data{'comments'}";

 print GUEST "</BLOCKQUOTE><HR>\n\n";

}
```

If the line was not `<!--begin-->`, however, the script makes sure to print the line so that it retains all of the HTML that was in the guestbook before a new entry was added. Thus, the long `for` loop goes through each line, prints the header, and gets all the way down through whatever HTML you've written until it gets to the guestbook entries, which begin with the `<!--begin--`" tag. Then the script prints the new entry as well as all the old entries. When the script gets to the end of the file (`@LINES`), it's over.

```
else
  {
  print GUEST $_;
  }
}
```

The script closes the guestbook and releases the lock file so that others can access the guestbook.

```
close (GUEST);
&ReleaseFileLock ("$guestbookreal.lock");
```

E-Mailing to the Guestbook Administrator

Now the script prepares to E-mail a note to the guestbook administrator. First, it renames `$form_data{'email'}` to `$email_of_sender`.

```
$email_of_guest = "$form_data{'email'}";
```

If the guestbook administrator has set `$mail` to 1 (guestbook administrator wants to be notified when someone enters a guestbook entry), the script begins creating the body of the E-mail. It stores the body in the variable `$email_body` and continually appends this variable by using ".=".

```
if ($mail eq '1')
  {
  $email_body .= "You have a new entry in your guestbook:\n\n";
  $email_body .= "-------------------------\n";
  $email_body .= "Name: $form_data{'realname'}\n";
```

If the guest submitted values, the script adds them, too.

```
if ($form_data{'email'} ne "")
  {
  $email_body .="Email: <$form_data{'email'}>\n";
  }

if ($form_data{'url'} ne "")
  {
  $email_body .="URL: <$form_data{'url'}>\n";
  }

if ($form_data{'city'} ne "")
  {
  $email_body .= "Location: $form_data{'city'},";
  }

if ($form_data{'state'} ne "")
  {
  $email_body .= " $form_data{'state'}";
  }
```

```
if ($form_data{'country'} ne "")
  {
  $email_body .= " $form_data{'country'}\n";
  }
```

Next, the script finishes the message body.

```
$email_body .= "Time: $date\n\n";
$email_body .= "Comments: $form_data{'comments'}\n";
$email_body .= "-------------------------\n";
```

Finally, the script uses the `send_mail` subroutine in the **mail-lib.pl** library file to send the E-mail to the guestbook administrator.

```
&send_mail("$email_of_guest", "$recipient",
           "$email_subject", "$email_body");
}
```

The guestbook administrator will receive the following E-mail summarizing the guestbook entry:

```
Date: Sat, 8 Jun 1996 13:33:41 -0700
From: selena@foobar.com
To: selena@foobar.com
Subject: Entry to Guestbook

You have a new entry in your guestbook:

--------------------------------------
Name: Selena Sol
Email: <selena@foobar.com>
URL: <http://www.foobar.com/~erict>
Location: Arlington, VA USA
Time: Saturday, June 8, 1996 at 13:33:40

Comments: Testing
--------------------------------------
```

Thanking the Guest through E-Mail

If the guestbook administrator has set `$remote_mail` to 1 and (`&&`) if the guest has submitted an E-mail, the script sends a "Thank you" E-mail to the guest. The process is identical to the previous one for generating and sending E-mail.

```
if ($remote_mail eq '1' &&
$form_data{'email'} ne "")
  {
  $email_body = "";
  $email_body .= <<"    end_of_message_to_guest";
  Thanks very much for stopping by my site, and a
  double thanks to you for taking the time to sign my
  guestbook. I hope you found something useful, and
  please let other netizens know of the existence of my
  little corner of the net.
  end_of_message_to_guest
  $email_body .= "\n";
  $email_body .= "    By the way, you wrote...\n\n";
  $email_body .= "    Name: $form_data{'realname'}\n";
  if ($form_data{'email'} ne "")
    {
    $email_body .="    Email: <$form_data{'email'}>\n";
    }

  if ($form_data{'url'} ne "")
    {
    $email_body .="    URL: <$form_data{'url'}>\n";
    }

  if ($form_data{'city'} ne "")
    {
    $email_body .= "    Location: $form_data{'city'},";
    }

 if ($form_data{'state'} ne "")
    {
    $email_body .= " $form_data{'state'}";
    }

 if ($form_data{'country'} ne "")
    {
    $email_body .= " $form_data{'country'}\n";
    }

 $email_body .= "    Time: $date\n\n";
 $email_body .= "    Comments:
                     $form_data{'comments'}\n";
 &send_mail("$recipient", "$email_of_guest",
            "$email_subject", "$email_body");
 }
```

The client who filled out the guestbook form will get an E-mail message something like the following:

```
Date: Sat, 8 Jun 1996 13:33:41 -0700
From: selena@foobar.com
To: gunther@foobar.com
Subject: Entry to Guestbook

Thanks very much for stopping by my site and a
double thanks to you for taking the time to sign my guestbook. I hope
you found something useful, and please let other netizens know of the
existence of my little corner of the net.

By the way, you wrote...

Name: Gunther Birznieks
Email: <gunther@foobar.com>
URL: <http://www.foobar.com/~gunther>
Location: Anytown, Anycity USA
Time: Saturday, June 8, 1996 at 13:33:40
Comments: Testing
```

Displaying HTML-Based Thank You

Then the script sends guests a thank you note on the Web and provides them with a way to get back to where they were before.

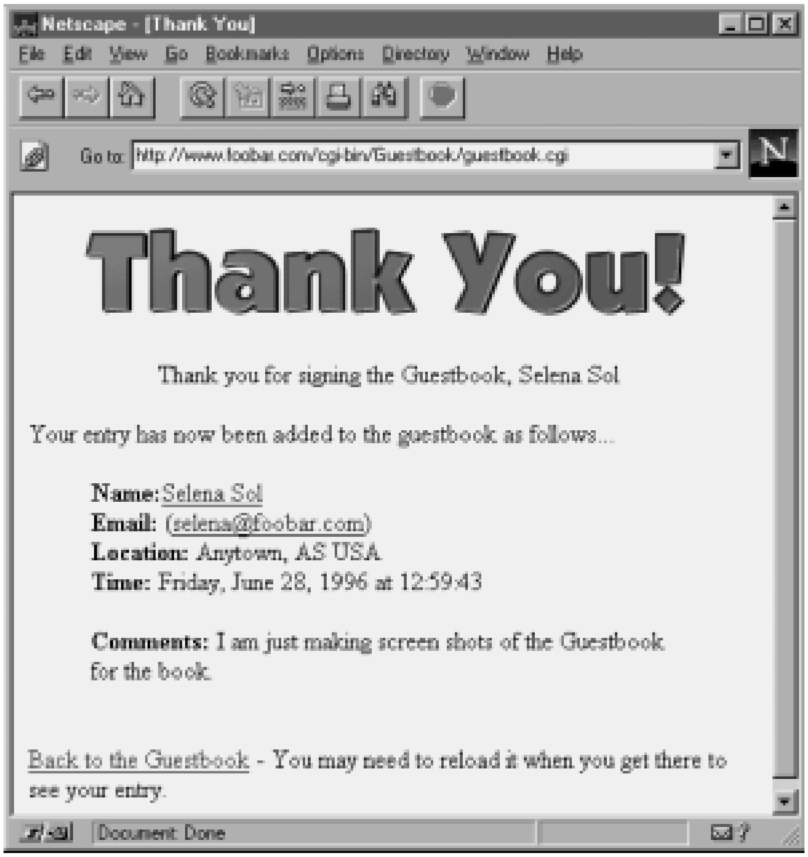
```
print <<"  end_of_html";
<HTML><HEAD><TITLE>Thank You</TITLE></HEAD>
<BODY>
<CENTER>
<IMG SRC = "http://www.foobar.com/Images/thankyou.gif">
<P>
Thank you for signing the Guestbook, $form_data{'realname'}
</CENTER>
<P>
Your entry has now been added to the guestbook as follows...<BLOCK-
QUOTE>
end_of_html
```

As a check for the client, the script prints a copy of the submissions.

```
if ($form_data{'url'} ne "")
  {
  print "<B>Name:</B>";
  print "<A HREF =
\"$form_data{'url'}\">$form_data{'realname'}</A><BR>";
  }
else
  {
  print "<B>Name:</B> $form_data{'realname'}<BR>";
  }
if ( $form_data{'email'} )
  {
  if ($linkmail eq '1')
    {
    print "<B>Email:</B>
           (<a href=\"mailto:$form_data{'email'}\">";
    print "$form_data{'email'}</a>)<BR>";
    }
  else
    {
    print "<B>Email:</B> ($form_data{'email'})<BR>";
    }
 }

 print "<B>Location:</B> ";

 if ( $form_data{'city'} )
   {
   print "$form_data{'city'},";
   }

 if ( $form_data{'state'} )
   {
   print " $form_data{'state'}";
   }

 if ( $form_data{'country'} ){
    print " $form_data{'country'}";
 }

 print "<BR><B>Time:</B> $date<P>";
 print "<B>Comments:</B> $form_data{'comments'}<BR>\n";
 print "</BLOCKQUOTE>";
 print "<a href=\"$guestbookurl\">Back to the
```

```
       Guestbook</a>\n";
print "- You may need to reload it when you get there
       to see your\n";
print "entry.\n";
print "</body></html>\n";
exit;
```

The client's response will look something like Figure 20.5.


Netscape - [Thank You]
File Edit View Go Bookmarks Options Directory Window Help
Go to: http://www.foobar.com/cgi-bin/Guestbook/guestbook.cgi
Document: Done

Thank You!

Thank you for signing the Guestbook, Selena Sol

Your entry has now been added to the guestbook as follows...

Name: Selena Sol
Email: (selena@foobar.com)
Location: Anytown, AS USA
Time: Friday, June 28, 1996 at 12:59:43

Comments: I am just making screen shots of the Guestbook for the book.

Back to the Guestbook - You may need to reload it when you get there to see your entry.

Figure 20.5 *Guestbook thank you response.*

The missing_required_field_data Subroutine

`missing_required_field_data` is used to warn clients that they failed to submit values for any or all of the required fields.

```
sub missing_required_field_data
  {
```

First, the passed parameter is set to the local variable `$field`.

```
local($field) = @_;
```

Then the script sends the client an informative error message.

```
print <<"    end_of_html";
<HTML><HEAD><TITLE>Data Entry Error</TITLE></HEAD>
<BODY>
<BLOCKQUOTE>Woopsy, You forgot to fill out $field and I am not allowed
to add your guestbook entry without it.  Would you please type some-
thing in below...</BLOCKQUOTE>
end_of_html
```

The script then reprints the add form using the subroutine `output_add_form` at the end of this script and exits.

```
 &output_add_form;
 exit;
}
```

The output_add_form Subroutine

`output_add_form` prints the form that clients use to add their entries.

```
sub output_add_form
  {
```

This subroutine is straightforward printing using the `HERE DOCUMENT` method.

```
print <<"    end_of_html";
<FORM METHOD = "POST" ACTION = "guestbook.cgi">
<TABLE>
<TR>
<TH ALIGN = "left">Your Name:</TH>
<TD><INPUT TYPE = "text" NAME = "realname" SIZE = "40"
           VALUE = "$form_data{'realname'}"></TD>
</TR><TR>
<TH ALIGN = "left">E-Mail:</TH>
<TD><INPUT TYPE = "text" NAME = "email" SIZE = "40"
           VALUE = "$form_data{'email'}"></TD>
</TR><TR>
<TH ALIGN = "left">URL:</TH>
<TD><INPUT TYPE = "text" NAME = "url" SIZE = "50"
           VALUE = "$form_data{'url'}"></TD>
</TR><TR>
<TH ALIGN = "left">City:</TH>
<TD><INPUT TYPE = "text" NAME = "city" SIZE = "15"
           VALUE = "$form_data{'city'}"></TD>
</TR><TR>
<TH ALIGN = "left">State:</TH>
<TD><INPUT TYPE = "text" NAME = "state" SIZE = "4"
           VALUE = "$form_data{'state'}"></TD>
</TR><TR>
<TH ALIGN = "left">Country:</TH>
<TD><INPUT TYPE = "text" NAME = "country" SIZE = "15"
           VALUE = "$form_data{'country'}"></TD>
</TR><TR>
<TH ALIGN = "left">Comments:</TH>
<TD><TEXTAREA NAME = "comments" COLS = "60" ROWS = "4">
$form_data{'comments'}
</TEXTAREA></TD>
</TR></TABLE>
<CENTER>
<INPUT TYPE = "submit" VALUE = "Submit Addition">
<INPUT TYPE = "reset">
</FORM>
<P>
<A HREF = "$guestbookurl">Back to the Guestbook Entries</A><BR>
</CENTER>
</BODY>
</HTML>
end_of_html
}
```

The get_date Subroutine

`get_date` is a subroutine written by Matt Wright that is used to get the current date to add to guestbook entries.

```
sub get_date
  {
  @days = ('Sunday', 'Monday', 'Tuesday', 'Wednesday',
           'Thursday', 'Friday', 'Saturday');
  @months = ('January', 'February', 'March', 'April',
             'May', 'June', 'July', 'August',
             'September', 'October', 'November',
             'December');
```

`get_date` uses the `localtime` command to get the current time. It also splits it into variables.

```
($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
localtime(time);
```

It then formats the variables and assigns them to the final `$date` variable.

```
if ($hour < 10) { $hour = "0$hour"; }
if ($min < 10) { $min = "0$min"; }
if ($sec < 10) { $sec = "0$sec"; }
$date = "$days[$wday], $months[$mon] $mday, 19$year at
$hour\:$min\:$sec";
}
```