

INTRODUCTION TO HTML

HyperText Markup Language (HTML) is a computer language used to create Web pages. Web pages are HTML documents, which consist of text and HTML tags. HTML documents can be created with a simple text editor or generated by Perl scripts. A Web browser interprets the tags in an HTML document and displays the document as a Web page.

The major benefit of using Web pages to display information is that Web pages can be displayed

on any computer that has a Web browser installed, including computers running a UNIX, Windows or Macintosh operating system. Web pages can also be displayed on devices such as handheld computers and some mobile telephones.

Since HTML documents contain only text, they are ideal for transferring information over slow networks, such as the Internet.

WEB BROWSERS

Microsoft Internet Explorer and Netscape Navigator are currently the most popular Web browsers. Microsoft Internet Explorer is included with Windows 98 and Windows 2000. You can also download the browser from the www.microsoft.com/ie Web site. Netscape Navigator is available at the www.netscape.com Web site.

WEB BROWSER SUPPORT

A Web page may not look the same when displayed in different Web browsers. Not all Web browsers support all the features of HTML and each Web browser may interpret HTML tags differently. Some companies that make Web browsers, such as Microsoft and Netscape, have also developed their own tags that Web browsers made by other companies may not be able to understand. If a Web browser does not understand a tag, the tag is usually ignored.

TAGS

Tags tell a Web browser about the structure of a Web page. Each tag gives a specific instruction and is surrounded by angle brackets <>. Most tags have an opening tag and a closing tag that affect the text between the tags. Some tags have attributes that offer options for the tag. For example, the tag has a color attribute that lets you change the color of text.

HTML VERSIONS

There are several versions of HTML. HTML specifications are constantly evolving and a new version of HTML is released about every two years. Each version offers new features to give users more control when creating Web pages. HTML version 4.01 is the latest version of HTML, although most Web pages conform to HTML 3.2. The World Wide Web Consortium (W3C) regulates the versions of HTML. You can view information about HTML and its versions at the www.w3c.org Web site.

WEB PAGE BASICS

You can use a text editor, such as Pico or Microsoft WordPad, to create a Web page. Each tag in a Web page gives a specific instruction. Most tags have an opening tag and a closing tag that affect the text between the tags. The closing tag has a forward slash (/).

You should add the <html>, <head> and <body> tags to every Web page you create. The <html> tag identifies a document as a Web page.

The <head> tag indicates the head of the Web page and can contain information such as the title of the page, the author of the page and keywords to be used by search tools to catalog the page. Information you enter between the <head> and </head> tags will not appear on the Web page.

You must enter the text you want to display on a Web page between the <body> and </body> tags. You can use alphanumeric characters and symbols to display information on your Web page.

WEB PAGE BASICS

UW PICO(tm) 3.5 File: index.html Modified

```
<html>
<head>
<title>Fruit and Flowers</title>
</head>

</html>
```

1 Start the text editor you will use to create a Web page.

2 Type <html> at the top of the page. Then press Enter twice and type </html>.

3 Type <head> directly below the <html> tag. Then press Enter.

4 Type the information you want the head to contain. Then press Enter.

5 Type </head>. Then press Enter twice.

UW PICO(tm) 3.5 File: index.html Modified

```
<html>
<head>
<title>Fruit and Flowers</title>
</head>
<body>
Fruit and Flowers, Inc.
No garden? No problem!
Our special, patented fertilizer lets you grow lush flowers and healthy fruit INDOORS!
Grow beautiful, exotic flowers and impress your friends!
Grow your own fruit and save on your grocery bills!
Our fruit selection includes:
Popular berries such as strawberries, raspberries and blueberries.
Exotic fruit such as mangos, papayas and kiwis.
Our flower selection includes:
Seasonal plants such as poinsettias, holly and mistletoe.
Tropical flowers such as orchids, birds of paradise and yellow jasmine.
</body>
</html>
```

6 Type <body> directly below the </head> tag. Then press Enter twice.

7 Type </body> directly above the </html> tag.

8 Position the cursor directly below the <body> tag and type the text you want to display on the Web page.

You can now add the HTML tags that will structure and format the text on the Web page.

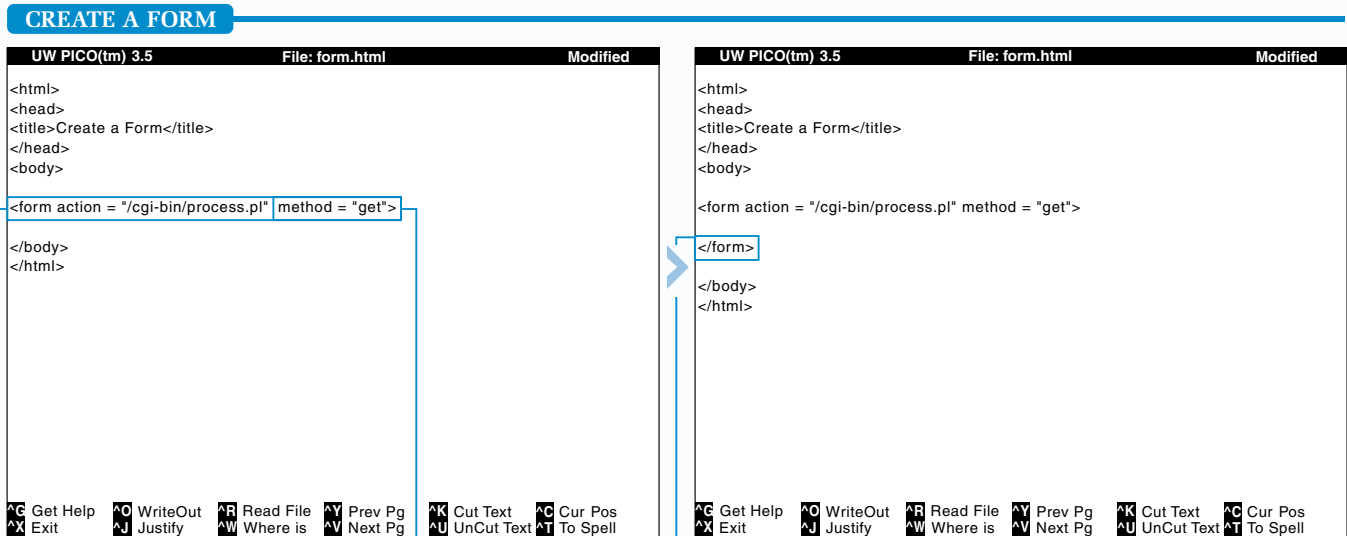
CREATE A FORM

Adding a form to a Web page allows you to gather data from users who visit the page.

A form can be placed anywhere between the <body> and </body> tags in an HTML document. The body of your Web page can include as many forms as you need.

You use the <form> tag to create a form and the action attribute to specify the directory and name of the Perl script that will process the data entered into the form. If the Perl script is not stored on the same Web server as the form, you must specify the full URL of the Perl script.

You must also specify which method the form will use to pass data to the Perl script. There are two methods the form can use—get and post. The get method sends data to the Perl script by appending the data to the URL of the page. The post method sends the data and the URL separately. The get method is faster than the post method and is suitable for small forms. The post method is suitable for large forms that will send more than 2000 characters to the Perl script. The Perl script must be capable of processing the form using the method you specify.



- 1 Position the cursor where you want to create a form and type <form action = followed by the directory and the name of the Perl script, enclosed in quotation marks, that will process the data entered into the form.
- 2 Press the Spacebar and type method = followed by the method, enclosed in quotation marks, that the form will use to pass data to the Perl script. Then type >.
- 3 Press Enter twice to leave space for the form elements and type </form> to complete the form.
- You can now add elements to the form.

ADD ELEMENTS TO A FORM

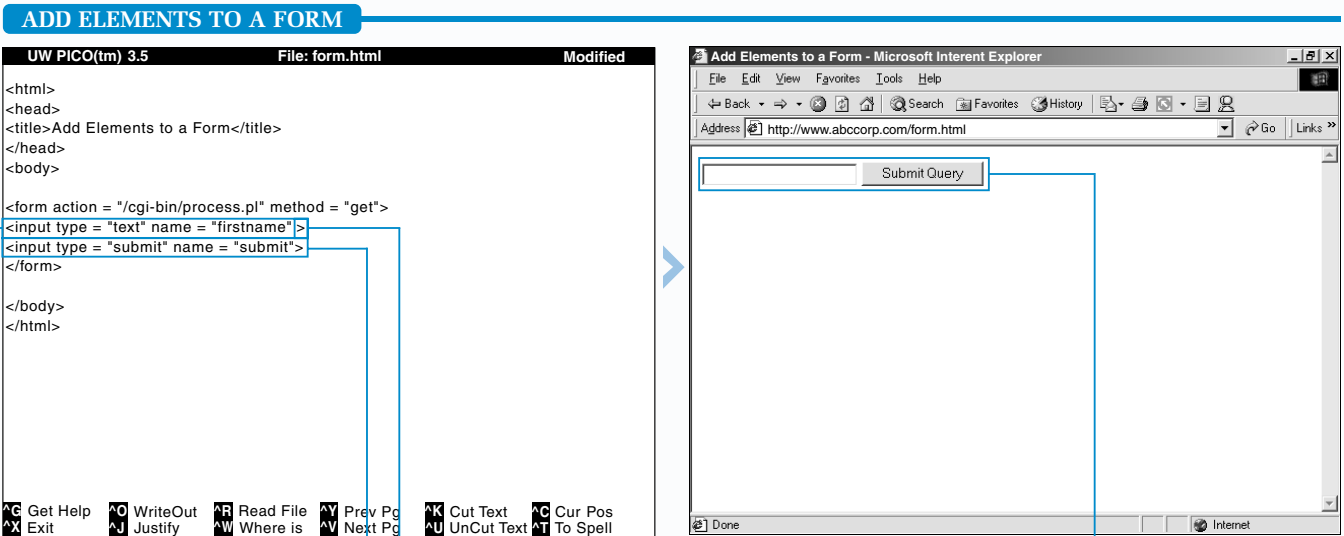
Elements are areas in a form where users can enter data and select options. The most commonly used element is a text box, which allows users to enter a single line of data into a form.

You must add elements between the <form> and </form> tags. Your form can contain as many elements as you need.

There are many different types of elements you can add to a form, including text areas, check boxes and buttons. Each element has attributes, such as type, name and size, which offer

options for the element. For information about commonly used elements and attributes, see page 198.

You must add a submit button to every form you create. The submit button allows users to send the data they entered into the form to the Web server. When the Web server receives data from a form, the server transfers the data to the Perl script that will process the data. The Perl script can then perform an action with the data, such as storing the data in a database.



- 1 Position the cursor between the <form> and </form> tags where you want to add an element.
- 2 To add a text box, type <input type = "text" name = followed by a word, enclosed in quotation marks, that describes the text box.
- 3 Type > and then press Enter.
- 4 To add a submit button, type <input type = "submit" name = followed by a word, enclosed in quotation marks, that describes the button. Then type >.
- 5 Save the Web page on the Web server and then display it in a Web browser.
- The Web browser displays the text box and submit button.

FORM ELEMENTS

An element is an area in a form where users can enter data or select options. There are several different types of elements you can add to a form. Most elements require you to specify attributes that determine how the

element will appear on a Web page. You can find more information about form elements and attributes at the www.w3.org/TR/1999/REC-html401-19991224/interact/forms Web site.

COMMONLY USED ATTRIBUTES

type The type attribute allows you to specify the kind of element you want to use.	name The name attribute allows you to specify a name for an element. Element names can contain more than one word, but should not contain spaces or special characters. The Perl script that will process data from the element uses the name attribute to identify the data.	value The value attribute allows you to specify a value for an element. If an element displays a button, you can use the value attribute to specify the text that will appear on the button.
maxlength The maxlength attribute allows you to restrict the number of characters a user can enter into an element.	size The size attribute allows you to specify the width of an element.	checked The checked attribute allows an element to display a selected option by default.

COMMONLY USED ELEMENTS

Password Box A password box allows users to enter private data. When a user types data into a password box, an asterisk (*) appears for each character, which prevents others from viewing the data on the screen. A password box does not protect the data from being accessed as it is transferred over the Internet. You must set the type attribute to password and use the name attribute to create a password box. You may also want to use the value, maxlength and size attributes. <pre>Password Please <input type = "password" name ="secretword" value = "password" maxlength = "20"></pre> Password Please <input type="password"/>	Drop Down List The select element displays a drop down list that allows users to select an option from a list of several options. For example, a drop down list can be used to allow users to select one of three shipping methods. You must use the name attribute to create a drop down list. You use the <option> tag with the value attribute to add options to the list. <pre>How would you like your products shipped? <select name = "shipMethod"> <option value = "air"> Air </option> <option value = "land"> Land </option> <option value = "sea"> Sea </option> </select></pre> How would you like your products shipped? <div><div>Air</div><div>Air</div><div>Land</div><div>Sea</div></div>
--	---

COMMONLY USED ELEMENTS (CONTINUED)

Text Box A text box allows users to enter a single line of text, such as a name or telephone number. You must set the type attribute to text and use the name attribute to create a text box. You may also want to use the maxlength and size attributes. <pre>First Name <input type = "text" name = "firstName" maxlength = "20"></pre> First Name <input type="text"/>	Text Area The textarea element displays a large text area that allows users to enter several lines or paragraphs of text. A large text area is ideal for gathering comments or questions from users. You must use the name attribute to create a text area. <pre>Questions? <textarea name = "userQuestions"> </textarea></pre> Questions? <div><div></div><div></div></div>
Check Box Check boxes allow users to select one or more options. For example, check boxes can be used to allow users to specify which states they have visited. You must set the type attribute to checkbox and use the name and value attributes to create a check box. You may also want to use the checked attribute. <pre>Which states have you visited in the past year?
 New York <input type = "checkbox" name = "states" value = "New York" checked> California <input type = "checkbox" name = "states" value = "California"> Texas <input type = "checkbox" name = "states" value = "Texas"></pre> Which states have you visited in the past year? New York <input checked="" type="checkbox"/> California <input type="checkbox"/> Texas <input type="checkbox"/>	
Radio Button Radio buttons allow users to select only one of several options. For example, radio buttons can be used to allow users to specify if they are male or female. You must set the type attribute to radio and use the name and value attributes to create a radio button. You may also want to use the checked attribute. <pre>What is your gender?
 Female <input type = "radio" name = "gender" value = "female"> Male <input type = "radio" name = "gender" value = "male" checked></pre> What is your gender? Female <input type="radio"/> Male <input checked="" type="radio"/>	
Submit Button A submit button allows users to send data in the form to the Perl script that will process the data. You must add a submit button to each form you create. You must set the type attribute to submit to create a submit button. You may also want to use the name and value attributes. <pre><input type = "submit" name = "submit" value = "Submit Now"></pre> <input type="submit" value="Submit Now"/>	Reset Button A reset button allows users to clear the data they entered into a form. Once a user has cleared the data, they cannot redisplay the data. Reset buttons are commonly used in forms that have many text boxes. You must set the type attribute to reset to create a reset button. You may also want to use the value attribute. <pre><input type = "reset" value = "Click to Reset"></pre> <input type="reset" value="Click to Reset"/>

USING PERL TO GENERATE A WEB PAGE

You can use Perl to create a script that will generate a Web page. When the Perl interpreter compiles and executes the Perl script, it will output the result as HTML code to a Web browser.

The first line of the script must include the code that points to the location of the Perl interpreter. On most UNIX systems, this location is `/usr/bin/perl`. The script must also include code that informs a Web browser that the script contains HTML code.

Use the `print` function to enter HTML code in the script, such as tags, attributes and text. If a quotation

mark (") is required in the code, which is often the case when specifying a value for an attribute in HTML code, you should use a backslash before the quotation mark (\"). For information about backslash escape sequences, see page 18.

When a Perl script is sent to a Web browser as HTML code, people can view the HTML source code for the page, but they will not be able to view the Perl script.

Extra

To create a Perl script that will generate a Web page, you must make sure you create or save the script in the correct directory. The directory used to store scripts is usually named `cgi-bin`. This directory is sometimes divided into subdirectories that store different types of scripts. Contact the system administrator or consult the documentation provided with the Web server program to ensure you are saving scripts in the correct directory. You use the `cd` command to change the current directory on a UNIX system.

Example:

```
[user]# cd /home/httpd/cgi-bin/net
```

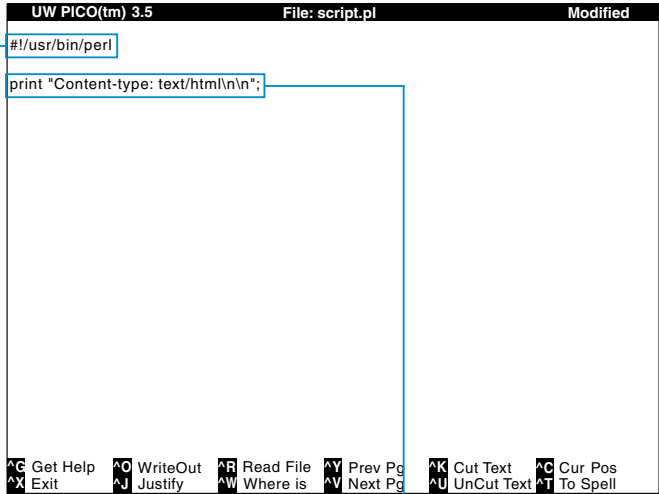
Other users require permission to read and execute a Perl script in order to view a script as a Web page. To allow other users permission to read and execute your script, enter the `chmod o+rx` command followed by the name of the script at the command prompt.

If your script is stored in a subdirectory, you also need to enter the `chmod o+x` command followed by the name of the subdirectory at the command prompt. This allows other users to access the subdirectory and its contents.

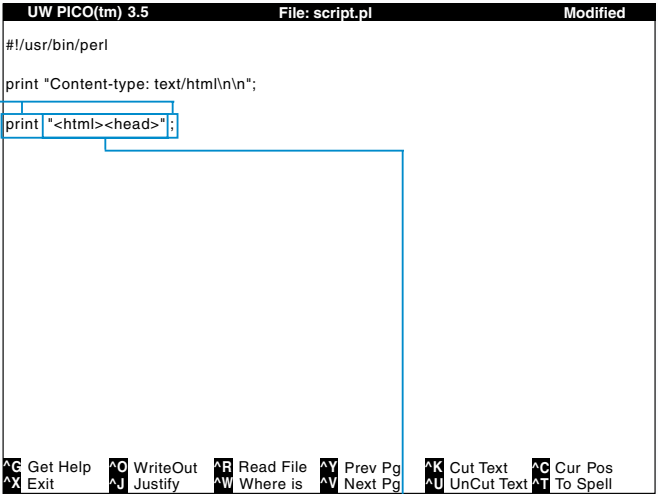
Example:

```
[cgi-bin]# chmod o+rx script.pl
[cgi-bin]# chmod o+x /home/httpd/cgi-bin/net
```

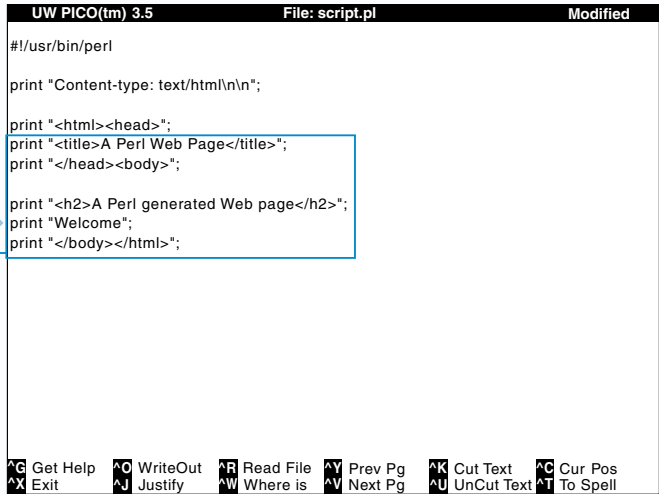
USING PERL TO GENERATE A WEB PAGE



1 To create a Perl script that will generate a Web page, type `#!` followed by the location of the Perl interpreter and then press Enter twice.



2 To inform a Web browser that the script contains HTML code, type `print "Content-type: text/html\n\n";` and then press Enter twice.



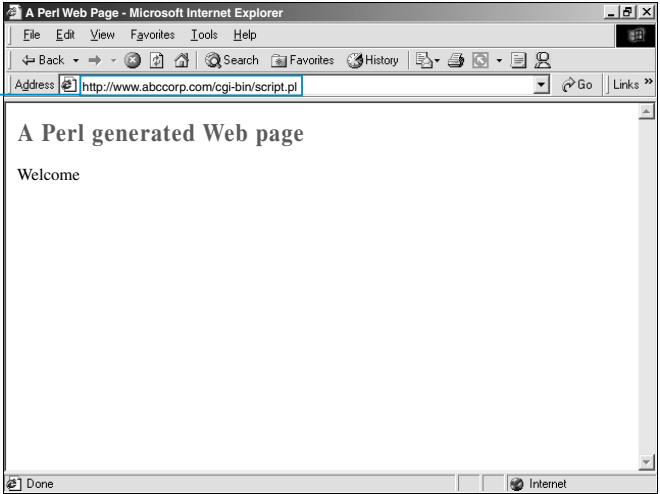
3 To enter HTML code in the script, type `print ;`.

4 Position the cursor over the semi-colon and type the HTML code you want to use enclosed in quotation marks.

5 Repeat steps 3 and 4 until you have entered all the Perl and HTML code you want to include in the Web page.

6 Save the script to your Web server.

Note: To transfer a script to a Web server, see page 14.



7 To view the Web page in a Web browser, type the URL of the page in the Web browser window.

The Web browser displays the Web page.

USING A QUERY STRING

A query string can be used to pass information to a Perl script for processing. A simple query string is composed of a name and a value separated by an equal sign, such as `email=joe@abccorp.com`.

The `QUERY_STRING` environment variable allows you to access a query string passed to a Perl script. To decode a query string, use the `split` function to split the name-value pair in the query string at the equal sign (`=`). Then assign the result of using the `split` function to an array. You can extract the value passed by the query string from the array

and assign it to a variable. This allows you to use the value in the script. For example, you can display the value or store the value in a file.

A query string is passed to a Perl script by appending the query string to the URL of the script. This can be accomplished by entering the URL and query string in the address bar of a Web browser. The query string must be separated from the URL by a question mark (`?`). A query string can also be passed to a Perl script by a form submitted to the script using the `get` method.

Extra

If the form that passes information to a Perl script uses the `post` method, the information will not be appended to the URL of the Perl script as a query string. As a result, the information cannot be accessed using the `QUERY_STRING` environment variable. To access information passed using the `post` method, use the input filehandle `STDIN`. The string of data stored in the `STDIN` filehandle can be decoded using the same method used to decode the `QUERY_STRING` variable.

Example:

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

$input = <STDIN>;
@qinfo = split(/\=/, $input);
$phone = @qinfo[1];

print "Your phone number: $phone has been entered ";
print "into our database<br><br>Thank you."
```

A query string can contain multiple names and values. Each name-value pair in a query string is separated by an ampersand (`&`), such as `name=Joe&id=134&dept=Sales`. To decode a query string containing multiple names and values, you should use the `CGI.pm` module. For information about the `CGI.pm` module, see page 228.

USING A QUERY STRING

```
UW PICO(tm) 3.5      File: script.pl      Modified
#!/usr/bin/perl

print "Content-type: text/html\n\n";

@qinfo = split(/\=/, $ENV{'QUERY_STRING'});

print "<h1>ABC Corporation Website</h1>";
```

```
UW PICO(tm) 3.5      File: script.pl      Modified
#!/usr/bin/perl

print "Content-type: text/html\n\n";

@qinfo = split(/\=/, $ENV{'QUERY_STRING'});
$name = @qinfo[1];

print "<h1>ABC Corporation Website</h1>";

print "<h2>Welcome $name!</h2>";
open(OUT, ">>name.txt");
print OUT $name, "\n";
close OUT;

print "Your name has been added to our user list.<br>";
print "Thank you.";
```

DECODE A QUERY STRING

1 To create an array that will store the name and value passed by the query string, type a name for the array followed by an equal sign (`=`).

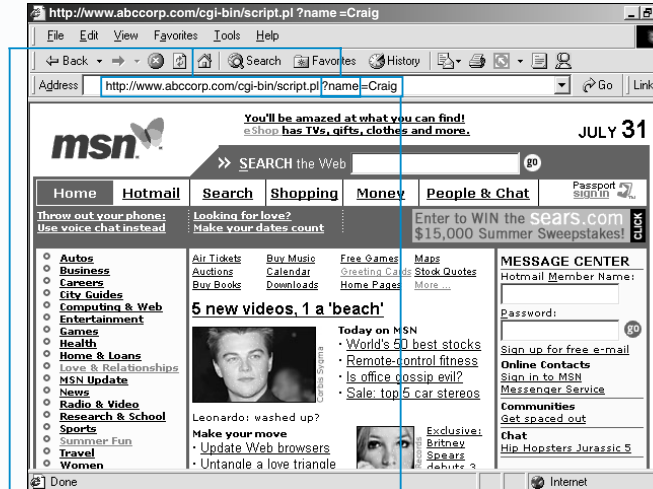
2 To separate the name and value at the equal sign (`=`) in the query string, type `split(/\=/, $ENV{'QUERY_STRING'})`.

3 Type the code that extracts the value passed by the query string from the array and assigns the value to a new variable.

4 Position the cursor where you want to type the code that uses the value passed by the query string and type the code.

5 Save the script to your Web server.

You can now enter a query string to pass a value to the script.



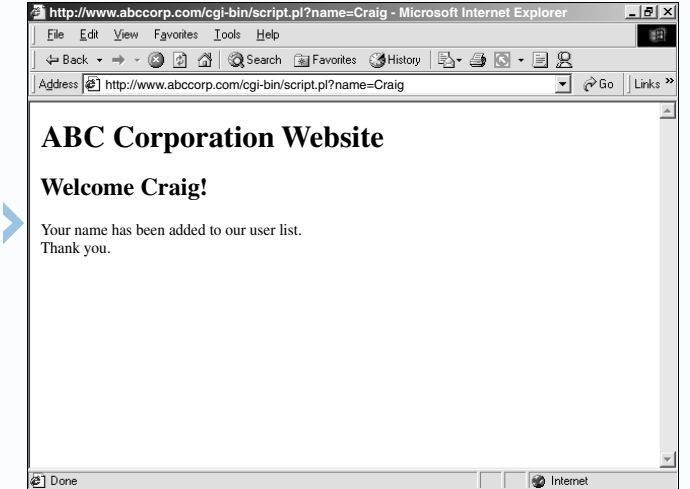
ENTER A QUERY STRING

1 In the address bar of a Web browser, type the URL of the Perl script you want to pass a value to.

2 Type `?` followed by a name for the value you want to pass.

3 Type `=` followed by the value.

4 Press Enter to pass the query string to the Perl script.



The Web browser displays the result of passing the value to the Perl script using a query string.

CREATE AND PROCESS A FORM IN ONE SCRIPT

One Perl script can be used to create a form and process the information passed by the form. This can help make your forms easier to maintain and transfer to other computers.

After typing the code that indicates the location of the Perl interpreter and informs Perl that the script contains HTML code, you can create a variable to store a query string passed to the script. You can then use an `if else` statement to determine if a query string has been passed.

When a query string is passed to the script, you can have the script decode the query string and use the value.

If a query string is not passed, you can have the script generate a Web page containing a form. A user can then enter the required information into the form and pass the information to the script for decoding and processing.

In the code that creates the form, you must ensure that the `action` attribute of the `<form>` tag specifies the name of the current Perl script.

Apply It

Regular expressions can be used to check the validity of the information entered into a form by a user. For example, you can verify that a ZIP code consists of only numeric characters or that a name contains no numeric characters. The `length` function can be used to test the length of a user's input. For example, you can check that a ZIP code consists of exactly five characters or that a name is less than 20 characters.

TYPE THIS:

```
$input = $ENV{QUERY_STRING};
if ($input)
{
    $zip = (split(/[&=]/, $input))[1];
    if (!(length($zip) == 5 and $zip =~ /^d+$ ))
    {
        print "Zip is an invalid ZIP code.";
    }
    else
    {
        print "Zip added to customer list.";
        open(OUT, ">>custinfo.txt"); print OUT "$zip\n"; close OUT;
    }
    print '<br><a href = "custinfo.pl">Go back</a>';
}
else
{
    print '<html><body>Enter your zip code';
    print '<form action = "custinfo.pl">';
    print '<input type = "text" name = "zip"><br>';
    print '<input type = "submit" value = "CLICK ME"></form></body></html>';
}
```

CREATE AND PROCESS A FORM IN ONE SCRIPT

The first screenshot shows the initial code in the UW PICO(tm) 3.5 editor. The code starts with a shebang line, sets the content type to text/html, and assigns the query string to a variable. It then uses an if statement to check if a query string was passed. The second screenshot shows the code with an else block added, which generates an HTML form with a text input for an email address and a submit button. The form's action is set to the current script's location.

SET UP A PERL SCRIPT TO CREATE AND PROCESS A FORM

1 Type the code that assigns a query string passed to the script to a variable. Then press Enter.

2 Type the code that creates an `if else` statement to test if a query string is passed to the script.

3 Position the cursor on the line directly below the opening brace ({) in the `if` statement and type the code that decodes a query string and uses its value.

4 Position the cursor on the line directly below the opening brace ({) in the `else` statement and type the code that creates a form if no query string is passed.

5 Save the script to your Web server.

The first screenshot shows the browser displaying the form at the URL http://www.abccorp.com/cgi-bin/form.pl. The form has a text input field containing 'april@abccorp.com' and a 'Submit' button. The second screenshot shows the browser displaying the result page at the URL http://www.abccorp.com/cgi-bin/form.pl?address=april@abccorp.com. The page displays 'Thank You' and 'april@abccorp.com' with a 'Go back' link.

EXECUTE THE PERL SCRIPT TO CREATE AND PROCESS A FORM

1 In the address bar of a Web browser, type the URL of the script you set up to create and process a form.

2 Enter the required information into the form that appears.

3 Click the button that passes the information in the form to the script.

4 The script decodes the information passed by the form and displays the result.

CREATE A COOKIE

You can use a Perl script to create a cookie. When a user views a Web page, the cookie is stored in a small text file on the user's computer. A cookie consists of a key, which indicates the name of the cookie, and a value, which is the information stored in the cookie.

Cookies are often used to personalize a Web page. For example, a cookie can store a user's name. The next time the user accesses the Web page, the page can use the value stored in the cookie to display the user's name.

You may want to specify when the cookie will expire. By default, a cookie will usually be deleted

as soon as the user closes the Web browser. Setting an expiry date for a cookie allows the cookie to store information for longer periods of time.

To create a cookie, you must generate a Set-cookie header in your script. This header must be placed directly under the code that indicates the location of the Perl interpreter so the header will be sent to the user's computer before any other code in the script. The Set-cookie header contains the key and value of the cookie, as well as the expiry date if one was specified.

To set up a Perl script to read a cookie you created, see page 208.

Extra

Perl allows you to delete a cookie before its expiry date. This is useful if you no longer need the information in the cookie. For example, you may want to delete a cookie that contains user registration information if the user cancels their registration to your Web site. To delete a cookie, create a new cookie with the same name and value, except specify an expiration date in the past. In this example, we erase a cookie named "userID".

Example:

```
print "Set-cookie: userID=Rev Mengle; expires=Wed 26 Jul 1999 17:59:59 GMT;";
```

Since users can disable cookies and some older Web browsers do not support the use of cookies, you should not assume that a cookie will be on a user's computer when you want to retrieve information from the cookie. You may want to warn users if a cookie is necessary to use your Web site.

You can create many cookies in your Perl script to store a user's personalized information. Each cookie on your page should not exceed 4 kilobytes (KB) in size.

CREATE A COOKIE

UW PICO(tm) 3.5File: script.plModified

```
#!/usr/bin/perl

print "Set-cookie: ";

print "Content-type: text/html\n\n";

print "<html><head>";
print "<title>ABC Corporation</title>";
print "</head><body>";

print "<center>";
print "<h1>Welcome to the ABC Corporation Web site</h1></center>";

print "</body></html>";
```

Get HelpWriteOutRead FilePrev PgCut TextCur PosExitJustifyWhere isNext PgUnCut TextTo Spell

1

Position the cursor directly below the `#!/usr/bin/perl` statement in the script and type `print "Set-cookie: ";`. Then press Enter.

UW PICO(tm) 3.5File: script.plModified

```
#!/usr/bin/perl

print "Set-cookie: visitedBefore=1;";

print "Content-type: text/html\n\n";

print "<html><head>";
print "<title>ABC Corporation</title>";
print "</head><body>";

print "<center>";
print "<h1>Welcome to the ABC Corporation Web site</h1></center>";

print "</body></html>";
```

Get HelpWriteOutRead FilePrev PgCut TextCur PosExitJustifyWhere isNext PgUnCut TextTo Spell

2

Position the cursor over the closing quotation mark and type the name of the cookie you want to create followed by `=`.

3

Type the value you want to assign to the cookie followed by `;`.

UW PICO(tm) 3.5File: script.plModified

```
#!/usr/bin/perl

print "Set-cookie: visitedBefore=1; expires=Wed 22 Sep 2004 23:59:59 GMT;";

print "Content-type: text/html\n\n";

print "<html><head>";
print "<title>ABC Corporation</title>";
print "</head><body>";

print "<center>";
print "<h1>Welcome to the ABC Corporation Web site</h1></center>";

print "</body></html>";
```

Get HelpWriteOutRead FilePrev PgCut TextCur PosExitJustifyWhere isNext PgUnCut TextTo Spell

4

To set an expiry date for the cookie, type `expires=`.

5

Type the date you want the cookie to expire followed by `;`. The date must appear in this format.

UW PICO(tm) 3.5File: script.plModified

```
#!/usr/bin/perl

print "Set-cookie: visitedBefore=1; expires=Wed 22 Sep 2004 23:59:59 GMT;\n";

print "Content-type: text/html\n\n";

print "<html><head>";
print "<title>ABC Corporation</title>";
print "</head><body>";

print "<center>";
print "<h1>Welcome to the ABC Corporation Web site</h1></center>";

print "</body></html>";
```

Get HelpWriteOutRead FilePrev PgCut TextCur PosExitJustifyWhere isNext PgUnCut TextTo Spell

6

Type `\n` to complete the Set-cookie header.

7

Save the script to your Web server.

When a user accesses the Web page, the cookie will be stored on the user's computer.

206

207

READ A COOKIE

A Perl script can read a cookie stored on a user's computer. Reading a cookie allows the script to access the information in the cookie, such as the user's name. You cannot read cookies created by other Web sites.

The CGI environment variable, HTTP_COOKIE, allows a Perl script to access the information stored in a cookie. If a cookie is available on a user's computer, the entire contents of the cookie will be stored in the \$ENV{ 'HTTP_COOKIE' } variable when the user's Web browser accesses

the Perl script. If no cookies are available on a user's computer, the \$ENV{ 'HTTP_COOKIE' } variable will be empty.

A script can search a simple cookie for a known name and value. For example, a script can search a cookie for a variable that stores information about whether the user has visited the Web site before. The script can then use the results to perform an action such as welcoming a user back to the Web site.

Apply It

To read a cookie that stores multiple names and values, a script can analyze the contents of the cookie and extract the names and values by using the split function. The names and values can then be assigned to a hash to make the information easier to work with.

When working with more complex cookies, you may prefer to use the CGI.pm module. For information about creating and reading cookies with the CGI.pm module, see pages 246 to 249.

TYPE THIS:

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

$cookie = $ENV{'HTTP_COOKIE'};

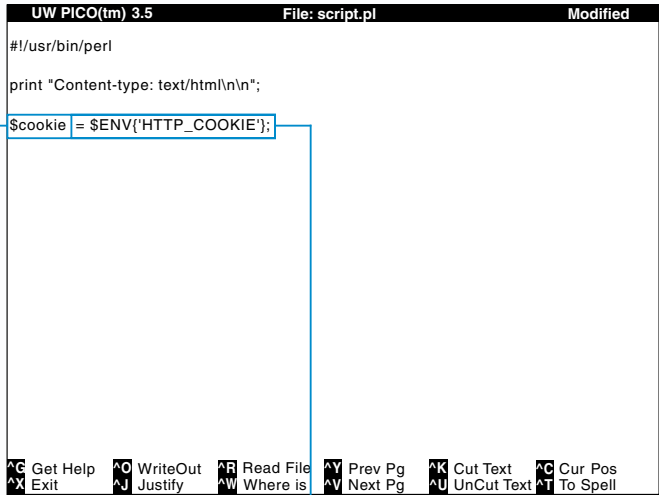
%namesvalues = split(/[,=]/, $cookie);

print "$_: <i>$namesvalues{$_}</i><br>" foreach (keys %namesvalues);
```

RESULT:

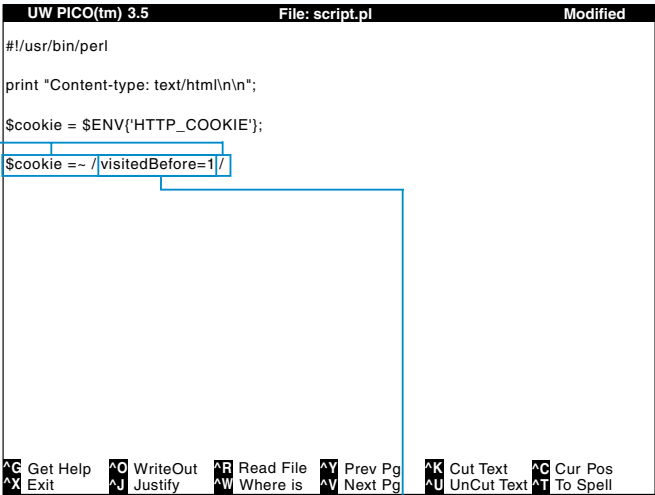
```
name: SandyR
password: blue123
```

READ A COOKIE



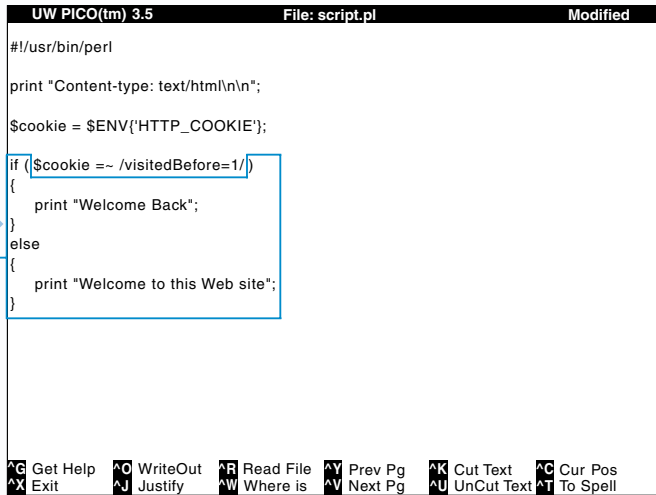
1 To create a variable to store the contents of a cookie, type the name of the variable.

2 Type = \$ENV{'HTTP_COOKIE'}; to assign the contents of a cookie to the variable. Then press Enter.



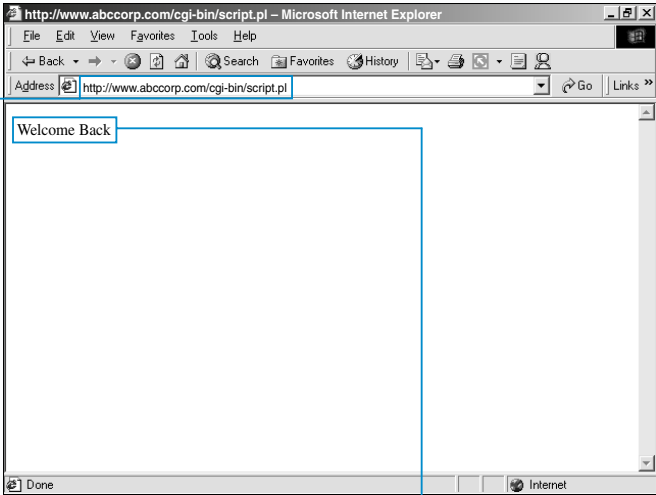
3 To determine whether a cookie contains a specific name and value, type the name of the variable that stores the cookie followed by =~ //.

4 Position the cursor over the closing pattern delimiter (/) and type the name and value you want to search for, separated by an equal sign (=).



5 Position the cursor where you want to type the code that uses the result of reading the cookie and type the code.

6 Save the script to your Web server.



7 To view the Web page in a Web browser, type the URL of the page in the Web browser window.

The Web browser displays the result of reading the cookie.

ACCESS CGI ENVIRONMENT VARIABLES

You can use a script to access Common Gateway Interface (CGI) environment variables, which store information about the Web server running your script and the user accessing the script. CGI environment variables are created by the Web server when a CGI program is started.

CGI environment variables are stored in the %ENV hash. To access the value of a CGI environment variable, you specify the name of the hash and the name of the variable you want to access. Use uppercase letters when specifying the name of the CGI environment variable.

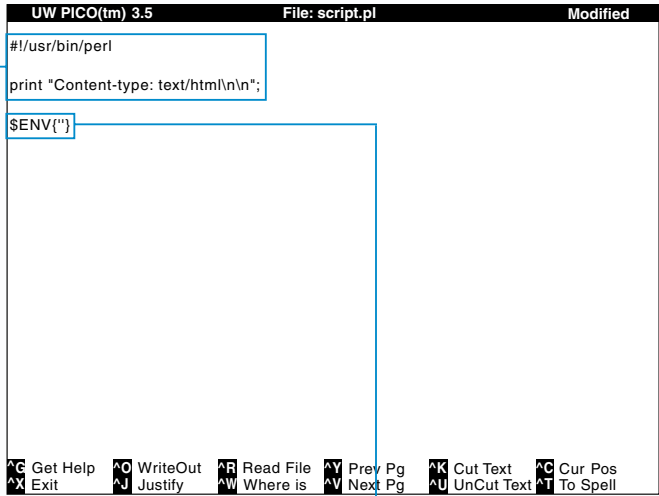
There are many CGI environment variables you can access, but two of the most useful are the REMOTE_ADDR and HTTP_USER_AGENT variables. The REMOTE_ADDR variable stores the IP number of a computer accessing your script. This is useful if you want to know where a user is located or if you want to restrict access to your script to specific IP numbers. The HTTP_USER_AGENT variable stores the type of Web browser that a visitor is using to access your script. This is useful if you want to customize your script to accommodate users with specific Web browsers.

Extra

The following CGI environment variables are commonly found on UNIX systems.

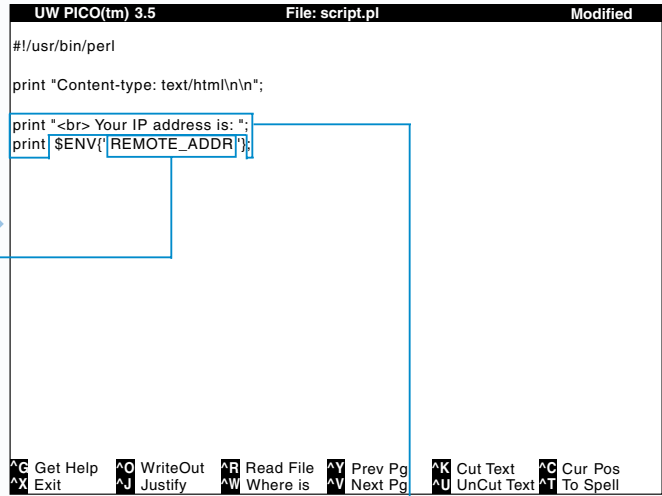
VARIABLE:	VALUE:
DOCUMENT_ROOT	The Web server directory that makes Web pages available to users.
GATEWAY_INTERFACE	The CGI version used by the Web server.
HTTP_ACCEPT	A list of media types the user's browser will accept.
HTTP_ACCEPT_ENCODING	The character encoding scheme the user's browser will accept.
HTTP_ACCEPT_LANGUAGE	The language the user's browser will accept.
HTTP_CONNECTION	The type of connection between the user's browser and the Web server.
HTTP_HOST	The IP number or host name of the Web server.
HTTP_USER_AGENT	The name and version of the user's Web browser.
QUERY_STRING	The query string information passed to the script.
REMOTE_ADDR	The IP number of the user.
REMOTE_PORT	The port number used by the user's browser to send information to the Web server.
REQUEST_METHOD	The method used to send information to the script.
REQUEST_URI	The Uniform Resource Identifier (URI) used to access the script.
SCRIPT_FILENAME	The path and filename of the script on the Web server.
SCRIPT_NAME	The virtual path of the script being run.
SERVER_ADDR	The IP number of the Web server.
SERVER_ADMIN	The e-mail address of the Web server administrator.
SERVER_NAME	The host name of the Web server.
SERVER_PORT	The port number used by the Web server to receive information.
SERVER_PROTOCOL	The version of the Web server communication protocol.
SERVER_SOFTWARE	The name and version of the Web server software.

ACCESS CGI ENVIRONMENT VARIABLES



1 Type the code that indicates the location of the Perl interpreter and informs Perl that the script contains HTML code.

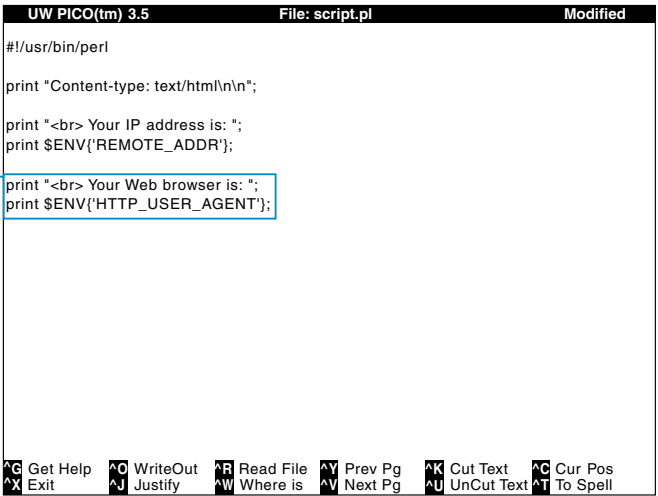
2 Position the cursor where you want to access a CGI environment variable and type \$ENV{''}.



3 Position the cursor over the closing quotation mark and type the name of the environment variable you want to access.

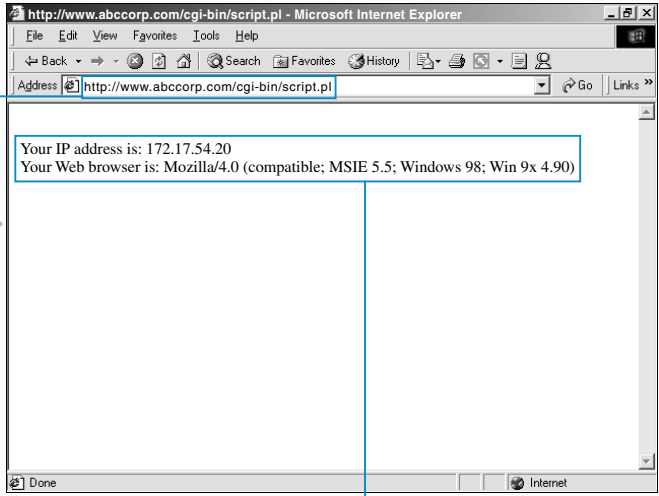
4 Position the cursor where you want to type the code that uses the CGI environment variable and type the code.

Note: You must use uppercase letters to specify the name of a CGI environment variable.



5 Repeat steps 2 to 4 for each CGI environment variable you want to access.

6 Save the script to your Web server.



7 To view the Web page in a Web browser, type the URL of the page in the Web browser window.

The Web browser displays the result of accessing the CGI environment variables.

USING SERVER-SIDE INCLUDES

Server-Side Includes (SSI) allow you to use one Perl script in several different Web pages. This can save you time when you need to include the same code, such as a header or footer, in multiple Web pages. If you change the Server-Side Include file, all the Web pages that include the file will be updated. Web pages that include a Server-Side Include may need to be saved with the .shtml extension.

You must first create the file you want to include. The file must begin with the line of code that indicates the location of the Perl interpreter.

To include the file in a Web page, you add the `#include virtual` statement to the code for

the page. The statement contains the name of the file you want to include and must be enclosed within HTML comment tags (`<!--` and `-->`).

The Web server processes the `#include` virtual statement in a Web page before executing any scripts and replaces the statement with the output generated by the specified file. The Web server then finishes processing the Web page and sends the page to a Web browser.

Server-Side Includes may not be enabled or available for your Web server. Consult the system administrator for more information.

CREATE A FILE YOU WANT TO INCLUDE

UW PICO(tm) 3.5

File: include.pl

Modified

#!/usr/bin/perl

Get Help

WriteOut

Read File

Prev Pg

Cut Text

Cur Pos

Exit

Justify

Where is

Next Pg

UnCut Text

To Spell

UW PICO(tm) 3.5

File: include.pl

Modified

#!/usr/bin/perl

print "Content-type: text/html\n\n";

print "<hr>";

print "Thank you for visiting the ABC Corporation Web site.
";

Get Help

WriteOut

Read File

Prev Pg

Cut Text

Cur Pos

Exit

Justify

Where is

Next Pg

UnCut Text

To Spell

- 1

Start the text editor you will use to create the file you want to include in several Web pages.
- 2

Type the code that tells the file where the Perl interpreter is located.
- 3

Type the code that tells the Web browser that the script contains HTML code.
- 4

Type the code you want to be able to execute in several Web pages.
- 5

Save the Perl script to your Web server.

Extra

The `#exec` command can be used in a Web page to run a Server-Side Include instead of using the `#include` virtual statement. When running a Server-Side Include using the `#exec` command, use the `cgi` attribute to specify the file you want to include. The `#exec` command can also be used with the `cmd` attribute to run a system command from the Web server. Before using the `#exec` command, make sure the file or system command you want to run will not compromise the security of your Web server or damage its contents.

TYPE THIS:

```
<!--#exec cgi = "/cgi-bin/message.pl"-->
Check out my scripts:<br>
<pre>
<!--#exec cmd = "ls"-->
</pre>
```

RESULT:

Welcome to my Web site.

Check out my scripts:

abccorp.html
script.shtml

You can include an `#echo` command in a Web page to display the value of the `DATE_GMT`, `DATE_LOCAL`, `DOCUMENT_NAME`, `DOCUMENT_URI` and `LAST_MODIFIED` environment variables, as well as all the CGI environment variables. You use the `var` attribute to specify the environment variable you want to display.

TYPE THIS:

```
Today is:<br>
<!--#echo var = "DATE_LOCAL"-->
```

RESULT:

Today is:
Monday, 31-Jul-2000 09:00:03 EDT

USE A SERVER-SIDE INCLUDE FILE

UW PICO(tm) 3.5

File: script.shtml

Modified

<html>

<head>

</head>

<body>

Enter your email address

<form action = "/cgi-bin/perlpage.pl">

<input type = "text" name = "address">

<input type = "submit">

</form>

<!--#include virtual = "/cgi-bin/include.pl"-->

</body>

</html>

File Name to write: script.shtml

Get Help

To Files

Exit

Complete

http://www.abccorp.com/script.shtml - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites History

Address http://www.abccorp.com/script.shtml

Go Links

Enter your email address

Submit Query

Thank you for visiting the ABC Corporation Web site.

Done Internet

- 1

In a text editor, display the HTML code for the Web page in which you want to include a file.
- 2

Click where you want to include the file and type `<!--#include virtual = ""-->`.
- 3

Position the cursor over the closing quotation mark and type the name of the file you want to include.
- 4

Save the Web page to your Web server.
- 5

To view the Web page in a Web browser, type the URL of the page in the Web browser window.
- The Web browser displays the result of using the Server-Side Include file.

SEND E-MAIL USING SENDMAIL

Sendmail is a popular and powerful e-mail program found on most computers running the UNIX operating system. Sendmail is often used to e-mail information that has been submitted to a Web site.

You can set up a form on your Web site to e-mail you the information users enter into the form. This provides a convenient way to keep track of the information users submit. Keep in mind that receiving form information by e-mail is only suitable for managing a small number of form submissions. For busier Web servers, it is more appropriate to store form results in a file or database.

To use the Sendmail program, you assign the program a filehandle in your script. You also need to specify the code that allows you to access the program on your Web server. Contact the system administrator or consult the documentation provided with Sendmail for information about accessing Sendmail from a Perl script.

Use the `print` function with the filehandle to specify the information you want to output to Sendmail, such as the subject of the message. When the filehandle is closed in the script, the e-mail message will be sent.

Extra

If you are using an operating system that cannot support the Sendmail program, you can add e-mail capabilities to a Perl script using a Perl module. For example, the `Net::SMTP` module, which is available from the www.cpan.org Web site, provides e-mail capabilities. Using a module may be more difficult to install and use than Sendmail but can give you more control over processing the e-mail. For example, you can process diagnostic messages from your code.

Example:

```
use Net::SMTP;
$smtp = Net::SMTP->new('mail.host.net')
or die "Couldn't create smtp object";

$smtp->mail($ENV{USER});
$smtp->to('administrator@abccorp.com');

$smtp->data();
$smtp->datasend("Dear administrator\n");
$smtp->datasend("This is the body\n");
$smtp->datasend("of an e-mail message.\n");
$smtp->dataend();

$smtp->quit;
```

If you want to learn more about the Sendmail program, you can visit the www.sendmail.org Web site. This site provides many useful resources, including links to security issues, FAQs and information about how to use the program. The Web site also offers the latest version of Sendmail that you can download for free.

SEND E-MAIL USING SENDMAIL

```
UW PICO(tm) 3.5      File: script.pl      Modified
#!/usr/bin/perl

print "Content-type: text/html\n\n";

$input = $ENV{'QUERY_STRING'};
@array = split(/=/, $input);
$address = @array[1];

open(SENDMAIL, "| /usr/lib/sendmail -f \"CGI Script\" -t");

print "$address has been added to the mailing list";

Ⓜ Get Help  Ⓜ WriteOut  Ⓜ Read File  Ⓜ Prev Pg  Ⓜ Cut Text  Ⓜ Cur Pos
Ⓜ Exit      Ⓜ Justify   Ⓜ Where is  Ⓜ Next Pg  Ⓜ UnCut Text Ⓜ To Spell
```

- 1 Type the code that processes information passed to the script using a query string.
- 2 To open the Sendmail program, type `open()`;
- 3 Position the cursor over the closing parenthesis and type a filehandle for the Sendmail program.
- 4 Type a comma followed by the code that allows you to access Sendmail on your Web server.

```
UW PICO(tm) 3.5      File: script.pl      Modified
#!/usr/bin/perl

print "Content-type: text/html\n\n";

$input = $ENV{'QUERY_STRING'};
@array = split(/=/, $input);
$address = @array[1];

open(SENDMAIL, "| /usr/lib/sendmail -f \"CGI Script\" -t");
print SENDMAIL "To: administrator@abccorp.com\n";

print "$address has been added to the mailing list";

Ⓜ Get Help  Ⓜ WriteOut  Ⓜ Read File  Ⓜ Prev Pg  Ⓜ Cut Text  Ⓜ Cur Pos
Ⓜ Exit      Ⓜ Justify   Ⓜ Where is  Ⓜ Next Pg  Ⓜ UnCut Text Ⓜ To Spell
```

- 5 Position the cursor where you want to specify the e-mail address where you want to send the message and type `print` followed by the filehandle. Then type `"To: "`.
- 6 Position the cursor over the closing quotation mark and type the e-mail address.

```
UW PICO(tm) 3.5      File: script.pl      Modified
#!/usr/bin/perl

print "Content-type: text/html\n\n";

$input = $ENV{'QUERY_STRING'};
@array = split(/=/, $input);
$address = @array[1];

open(SENDMAIL, "| /usr/lib/sendmail -f \"CGI Script\" -t");
print SENDMAIL "To: administrator@abccorp.com\n";
print SENDMAIL "Subject: Form completed\n";
print SENDMAIL "A form has been submitted containing: $address\n";

print "$address has been added to the mailing list";

Ⓜ Get Help  Ⓜ WriteOut  Ⓜ Read File  Ⓜ Prev Pg  Ⓜ Cut Text  Ⓜ Cur Pos
Ⓜ Exit      Ⓜ Justify   Ⓜ Where is  Ⓜ Next Pg  Ⓜ UnCut Text Ⓜ To Spell
```

- 7 Position the cursor where you want to specify the subject of the message and type `print` followed by the filehandle. Then type `"Subject: "`.
- 8 Position the cursor over the closing quotation mark and type the subject.
- 9 Position the cursor where you want to specify the contents of the message and type `print` followed by the filehandle. Then type the contents of the message enclosed in quotation marks.

```
UW PICO(tm) 3.5      File: script.pl      Modified
#!/usr/bin/perl

print "Content-type: text/html\n\n";

$input = $ENV{'QUERY_STRING'};
@array = split(/=/, $input);
$address = @array[1];

open(SENDMAIL, "| /usr/lib/sendmail -f \"CGI Script\" -t");
print SENDMAIL "To: administrator@abccorp.com\n";
print SENDMAIL "Subject: Form completed\n";
print SENDMAIL "A form has been submitted containing: $address\n";
close SENDMAIL;

print "$address has been added to the mailing list";

Ⓜ Get Help  Ⓜ WriteOut  Ⓜ Read File  Ⓜ Prev Pg  Ⓜ Cut Text  Ⓜ Cur Pos
Ⓜ Exit      Ⓜ Justify   Ⓜ Where is  Ⓜ Next Pg  Ⓜ UnCut Text Ⓜ To Spell
```

- 10 To close the Sendmail program, type `close` followed by the filehandle for the program.
- 11 Save the script to your Web server.
- When information is passed to the script using a query string or a form, the Sendmail program sends the message to the e-mail address you specified.