CREATE AND READ A COOKIE

PHP pages can be used to create and read cookies. When a user accesses a page that creates a cookie, the cookie is typically stored as a small text file on the user's computer. The cookie can later be read by a PHP page to access the information stored in the cookie. For example, a cookie can store a user's name. When the user accesses the PHP page again later, the page can use the value stored in the cookie to display the user's name.

To create a cookie, you use the `setcookie` function. The function takes several arguments, including the name you want to assign to the cookie, the value you want the cookie to store and an expiry time for the cookie. The `setcookie` function must be placed before any other code on a PHP page.

By default, a cookie will be deleted when the user closes their Web browser. Setting an expiry time for a cookie allows the cookie to store information for longer periods of time. The `time` function can be used to set the expiry time, in seconds, for a cookie.

After creating a cookie, you can have a PHP page read the cookie. When a user with a cookie stored on their computer visits a PHP page that can read the cookie, PHP automatically converts the name of the cookie to a variable and assigns the value stored in the cookie to the variable. This makes it easy to work with cookies using PHP. To read a cookie stored on a user's computer, you simply access the cookie variable by prefixing the name of the cookie with a dollar sign (`$`).

When working with cookies, keep in mind that a client may be configured to reject cookies or may be located behind a security firewall that filters out cookie information.

Extra

Instead of specifying the expiry time for a cookie in seconds, you can specify an exact date when you want the cookie to expire. This is useful for a cookie that you do not want to expire for a long period of time. To create a cookie that uses an exact date for the expiry time, you can use the `header` function to send the cookie to the client as an HTTP header.

Example:

```
header("Set-Cookie: accessedBefore=Yes; expires=Friday, 07-Mar-2003 00:00:00 GMT;");
```

To prevent unauthorized pages from reading a cookie you placed on a user's computer, you can specify a path and domain for the cookie in the `setcookie` function. Only the pages stored in the specified directory and domain will be able to read the cookie.

Example:

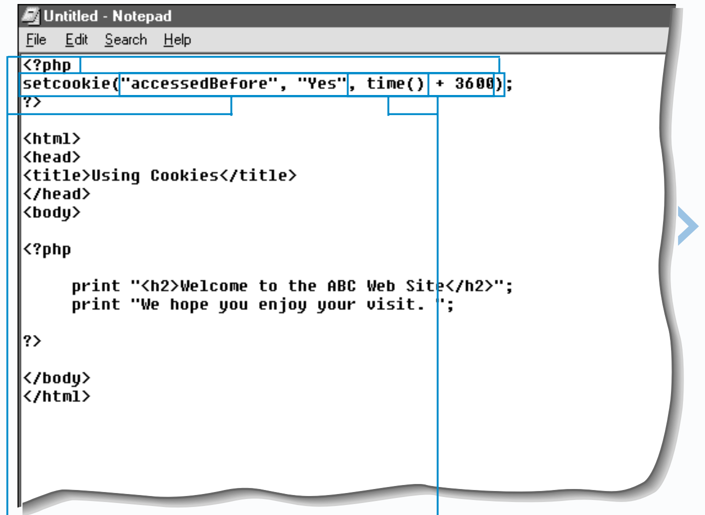
```
setcookie("accessedBefore", "Yes", time() + 3600, "/web/text", "www.test.com");
```

If a PHP page attempts to access a variable that does not exist, an error will be generated. You may want to use the `isset` function to verify that a cookie variable exists before attempting to access the variable in a PHP page.

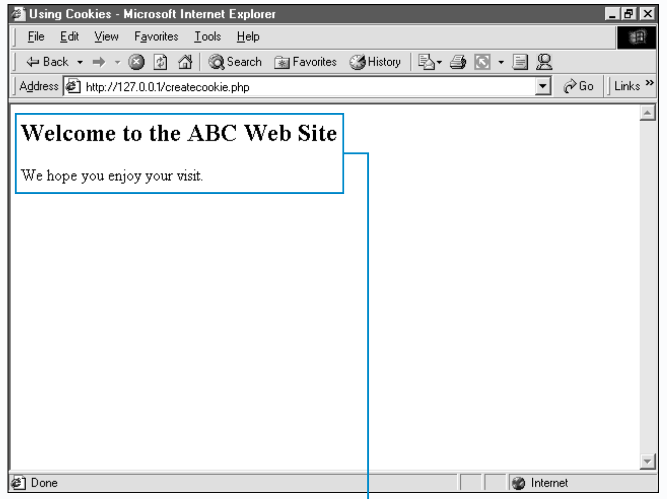
Example:

```
if (isset($cookieValue))
{
    print "A cookie exists on your computer";
}
```

CREATE A COOKIE



```
<?php
setcookie("accessedBefore", "Yes", time() + 3600);
?>

<html>
<head>
<title>Using Cookies</title>
</head>
<body>

<?php

     print "<h2>Welcome to the ABC Web Site</h2>";
     print "We hope you enjoy your visit. ";

?>

</body>
</html>
```



Using Cookies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1/createcookie.php
Go Links
Done
Internet

Welcome to the ABC Web Site

We hope you enjoy your visit.

1 To create a cookie, type **setcookie()**.

2 Between the parentheses, type a name for the cookie enclosed in quotation marks. Then type a comma followed by the value you want to assign to the cookie enclosed in quotation marks.

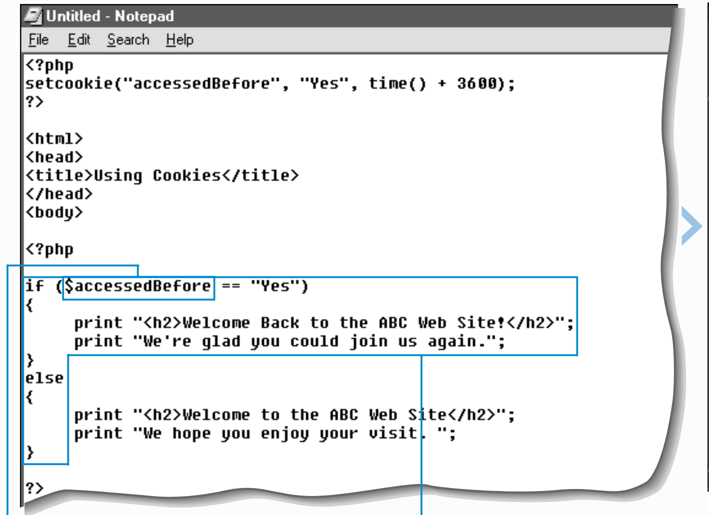
3 To specify when the cookie will expire, type a comma followed by **time()**.

4 Type **+** followed by the expiry time for the cookie in seconds.

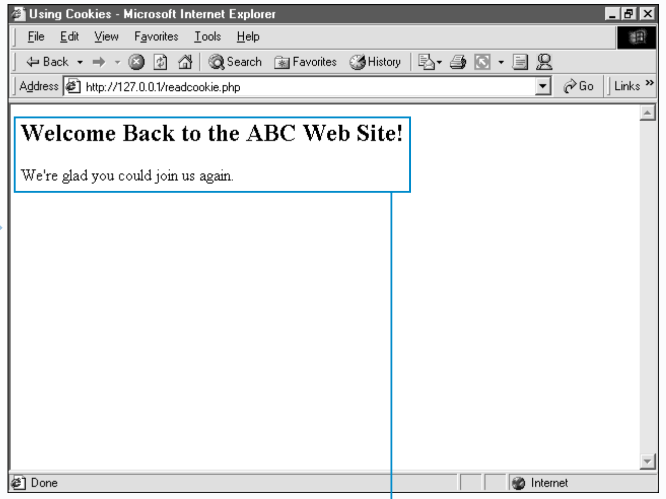
5 Display the PHP page in a Web browser.

■ The cookie is now stored on the computer.

READ A COOKIE



```
<?php
setcookie("accessedBefore", "Yes", time() + 3600);
?>

<html>
<head>
<title>Using Cookies</title>
</head>
<body>

<?php

if ($accessedBefore == "Yes")
{
     print "<h2>Welcome Back to the ABC Web Site!</h2>";
     print "We're glad you could join us again.";
}
else
{
     print "<h2>Welcome to the ABC Web Site</h2>";
     print "We hope you enjoy your visit. ";
}

?>
```



Using Cookies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1/readcookie.php
Go Links
Done
Internet

Welcome Back to the ABC Web Site!

We're glad you could join us again.

1 In the PHP page, type **$** followed by the name of the cookie you want to read.

2 Type the code that uses the value of the cookie.

3 Display the PHP page in a Web browser.

■ The Web browser displays the result of reading a cookie.

DELETE A COOKIE

PHP allows you to delete a cookie before it expires. 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.

It may also be necessary to remove cookies if you already have the maximum number of cookies allowed, but want to create more. PHP permits each domain to store up to 20 cookies. You can delete cookies you no longer need to make room for new cookies.

To remove a cookie, you use the `setcookie` function to create a new cookie that has the same name as the cookie you want to remove, but has an expiry time set as a time in the past. For example, you can use the `time` function to set the expiry time as a negative number of seconds, such as time() - 60. This will cause the cookie to expire immediately. The `setcookie` function must be placed before any other code on a PHP page.

When deleting a cookie, the value you assign to the cookie can be an empty string. If you specified a path and domain when you created the original cookie, you must specify the same path and domain when deleting the cookie. This ensures that the correct cookie is removed. For information about specifying a path and domain for a cookie, see the top of page 148.

Working with cookies is not always a simple task. Some Web servers and Web browsers work with cookies in different ways. For example, some Web servers will not allow a cookie to be removed from a client computer until the cookie reaches its original expiry time. When working with cookies, you should thoroughly test your code on all Web browsers you expect to access your PHP page.

Extra

If you want to make changes to a cookie you have previously created, you must delete the cookie and then recreate the cookie using the new information. The process of deleting and then recreating a cookie differs depending on the version of PHP you are using.

PHP Version 4

When using PHP version 4, cookies you create and delete using the `setcookie` function are processed in the order they appear in the PHP script. To delete and then recreate a cookie, you must place the `setcookie` statement that deletes the cookie before the `setcookie` statement that recreates the cookie.

Example:

```
setcookie("status", "", time() - 60);
setcookie("status", "approved", time() + 3600);
```

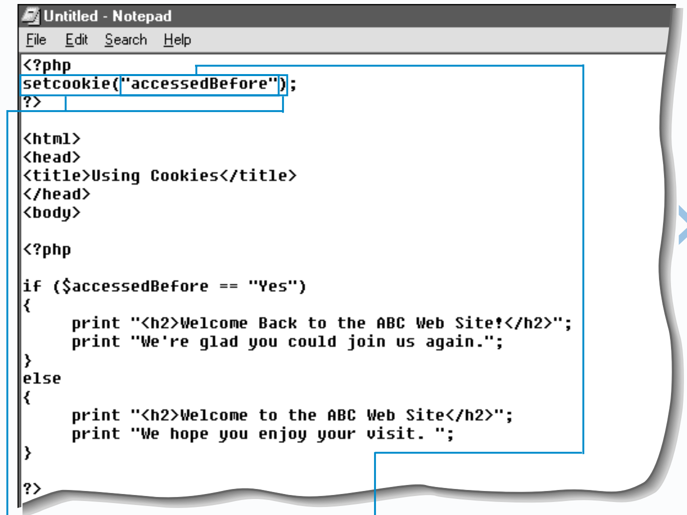
PHP Version 3

When using PHP version 3, cookies you create and delete using the `setcookie` function are processed in reverse order. To delete and then recreate a cookie, you must place the `setcookie` statement that recreates the cookie before the `setcookie` statement that deletes the cookie.

Example:

```
setcookie("status", "approved", time() + 3600);
setcookie("status", "", time() - 60);
```

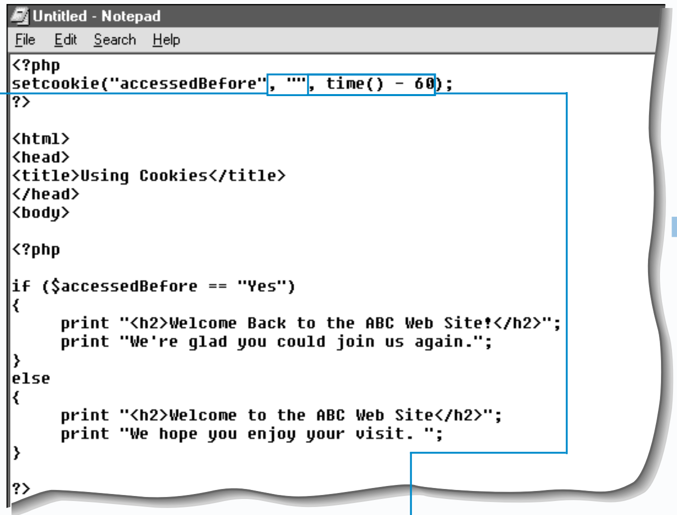
DELETE A COOKIE



```
<?php
setcookie("accessedBefore");
?>

<html>
<head>
<title>Using Cookies</title>
</head>
<body>

<?php

if ($accessedBefore == "Yes")
{
     print "<h2>Welcome Back to the ABC Web Site!</h2>";
     print "We're glad you could join us again.";
}
else
{
     print "<h2>Welcome to the ABC Web Site</h2>";
     print "We hope you enjoy your visit. ";
}

?>
```

1 To delete a cookie, type **setcookie()**.

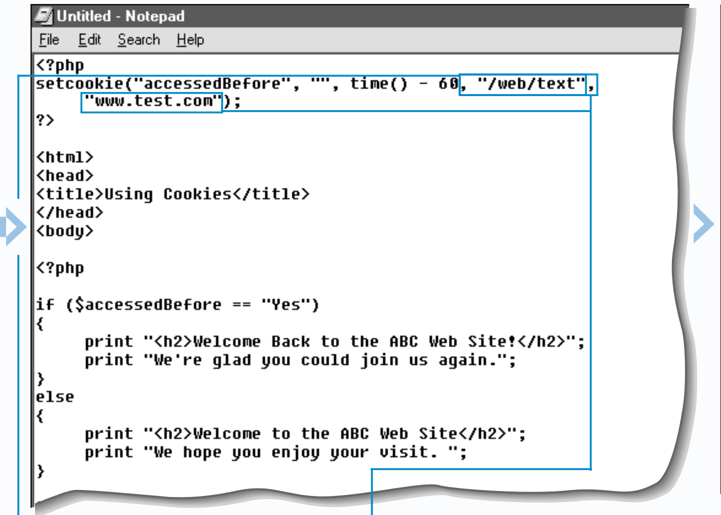
2 Between the parentheses, type the name of the cookie you want to delete enclosed in quotation marks.



```
<?php
setcookie("accessedBefore", "", time() - 60);
?>

<html>
<head>
<title>Using Cookies</title>
</head>
<body>

<?php

if ($accessedBefore == "Yes")
{
     print "<h2>Welcome Back to the ABC Web Site!</h2>";
     print "We're glad you could join us again.";
}
else
{
     print "<h2>Welcome to the ABC Web Site</h2>";
     print "We hope you enjoy your visit. ";
}

?>
```

3 Type a comma and then type "" to enter an empty string for the value of the cookie.

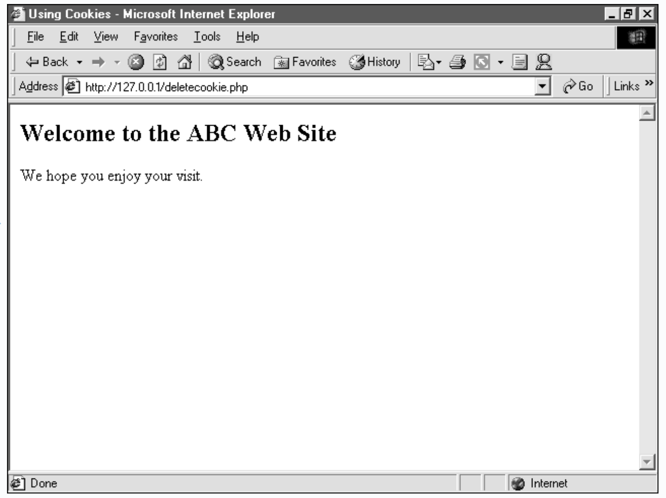
4 To specify an expiry time in the past, type a comma and then type **time() -** followed by a number of seconds.



```
<?php
setcookie("accessedBefore", "", time() - 60, "/web/text",
     "www.test.com");
?>

<html>
<head>
<title>Using Cookies</title>
</head>
<body>

<?php

if ($accessedBefore == "Yes")
{
     print "<h2>Welcome Back to the ABC Web Site!</h2>";
     print "We're glad you could join us again.";
}
else
{
     print "<h2>Welcome to the ABC Web Site</h2>";
     print "We hope you enjoy your visit. ";
}
```

5 If you specified a path for the cookie when the cookie was created, type a comma and then type the path enclosed in quotation marks.

6 If you specified a domain for the cookie when the cookie was created, type a comma and then type the domain enclosed in quotation marks.


Using Cookies - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1/deletecookie.php
Go Links
Done
Internet

Welcome to the ABC Web Site

We hope you enjoy your visit.

7 Display the PHP page in a Web browser.

■ The cookie has been deleted from the computer.

START A SESSION

A session is created for each user that requests a PHP page from your Web site. A session enables a Web server to use `cookies` to collect and use information entered by a user while the user accesses resources on the Web server. For example, if a user specifies a user name on the main page of a Web site, this user name can be used by the Web server to personalize any other Web pages the user requests during that session.

To start a session in a PHP page, you use the `session_start` function. The `session_start` function must be placed before any HTML code on your PHP page. It is also good programming practice to start a session at the beginning of your PHP page so that session information is available throughout the page. Calling the `session_start` function while an existing session is in progress will not create a new session or affect the existing session.

The Web server keeps track of each session by assigning a session ID to identify each current user. When a session is started, the Web server stores a session ID as a cookie named PHPSESSID on the user's computer. When the user requests another page from the site, the user's Web browser sends the session ID to the Web server to identify the user. To access the current session ID using a PHP page, you type $PHPSESSID.

Session IDs are randomly generated by the Web server. You should not use the session ID as a unique identifier, such as the primary key in a database, as the session ID may not always be unique. For example, if the Web server is restarted, the server may assign a user a session ID that was previously assigned to a different user.

Extra

PHP typically stores session information in a directory on the Web server. The name and location of this directory is specified by the `session.save_path` value in the php.ini configuration file. You should check to make sure the directory exists before starting a session. If the directory does not exist, you can create it. For example, if the php.ini configuration file specifies /tmp as the `session.save_path` value, you can create a directory named tmp in the main directory on the Web server.

You can change the name of the cookie used to store the session ID number by changing PHPSESSID in the `session.name` value in the php.ini configuration file on the Web server. Using a unique `session.name` value helps to increase the security of your PHP pages.

You can use the `session_id` function to specify a different value for a session ID. This is useful when you want to create unique session IDs that you can use as identifiers in your code. The session ID must be enclosed in quotation marks and placed within the parentheses following the `session_id` function. You should use only alphanumeric characters when specifying a session ID.

Example:

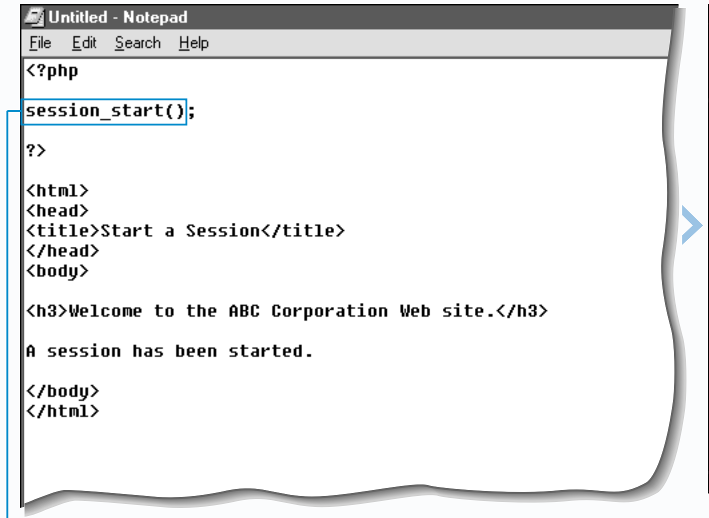
```
session_id("User593204");
```

You can also use the `session_id` function to access the session ID instead of using the PHPSESSID cookie. Before using the `session_id` function, you should call the `session_start` function to indicate that you will use session information.

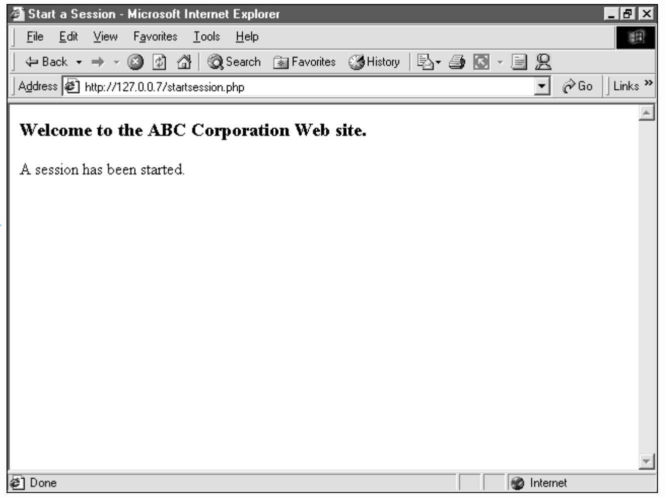
Example:

```
print session_id();
```

START A SESSION



```
<?php

session_start();

?>

<html>
<head>
<title>Start a Session</title>
</head>
<body>

<h3>Welcome to the ABC Corporation Web site.</h3>

A session has been started.

</body>
</html>
```



Welcome to the ABC Corporation Web site.

A session has been started.

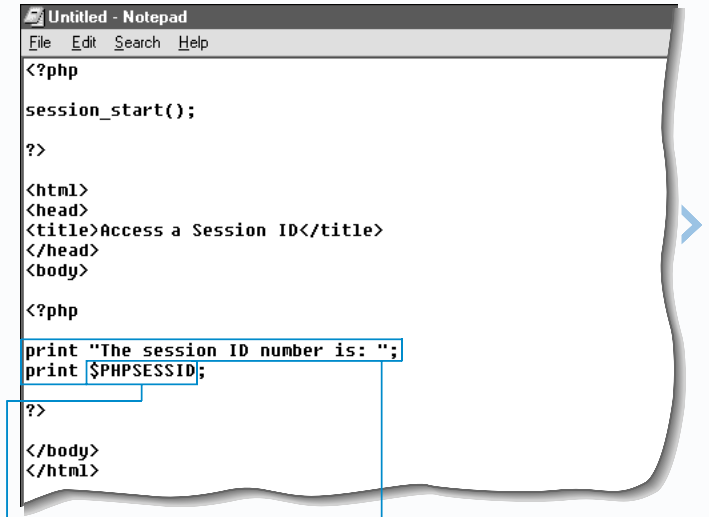
1 To start a session, type **session_start()**.

■ The `session_start` function should be placed before any HTML code on your PHP page.

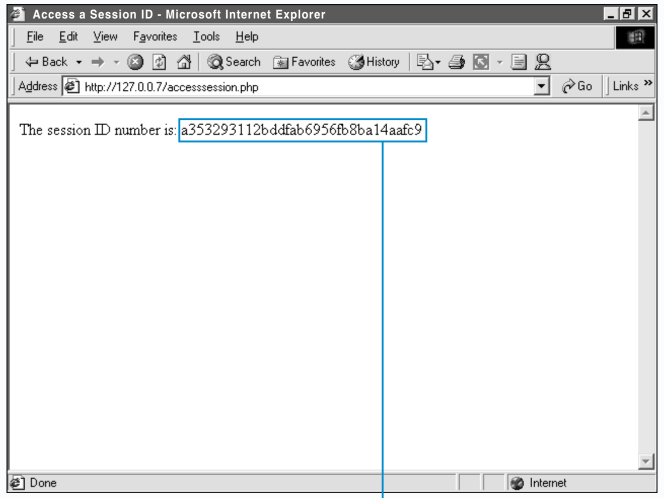
2 Display the PHP page in a Web browser.

■ The session is started.

ACCESS THE SESSION ID



```
<?php

session_start();

?>

<html>
<head>
<title>Access a Session ID</title>
</head>
<body>

<?php

print "The session ID number is: ";
print $PHPSESSID;

?>

</body>
</html>
```



The session ID number is: a353293112bddfab6956fb8ba14aafc9

1 Type **$PHPSESSID** where you want to access a session ID.

2 Type the code that will display the session ID in a Web browser.

3 Display the PHP page in a Web browser.

■ The Web browser displays the result of accessing the session ID.

CREATE AND READ A SESSION VARIABLE

As a user moves through the pages in your Web site, the user may be asked to enter information such as a user name, password or preferences to display each page. Creating session variables allows you to store this information and make the information available to all the pages viewed by the user in your Web site. This saves the user from having to repeatedly enter the same information to display each page during a session.

You use the `session_register` function to specify the name of the variable you want to create. The name of the session variable must be enclosed in quotation marks. Once the session variable has been registered, you can assign a value to the variable. The information stored in a session variable can come from sources such as forms, databases and cookies.

After creating a session variable, you can use the `session_register` function to read the information stored in the session variable. You can also change the value assigned to a session variable in any PHP page. Changes made to a variable will affect all the PHP pages that use the variable. You can use the session variable in your PHP pages as you would use any variable.

The `session_start` function should be called at the beginning of every PHP page that will use session information. Calling the `session_start` function will have no effect if a session is already in progress. If the `session_register` function is used before the `session_start` function is called, a session will be started automatically.

The use of session variables is an effective way of collecting and accessing information across multiple pages on a Web site and is more secure and easier to maintain than HTML hidden fields or cookies.

Extra

All session variables and the information stored in them will be discarded when the session ends or is terminated. If necessary, you can use cookies or a database to save the information stored in a session variable.

The `session_is_registered` function can be used to determine if a variable has been registered as a session variable. The function will return a value of true if the variable has been registered in the current session.

Example:

```
if (session_is_registered("userName"))
{
     print "The user name has been registered.";
}
else
{
     print "The user name does not exist in this Web site.";
}
```

You can use the `session_unregister` function to remove a session variable you created in the current session.

Example:

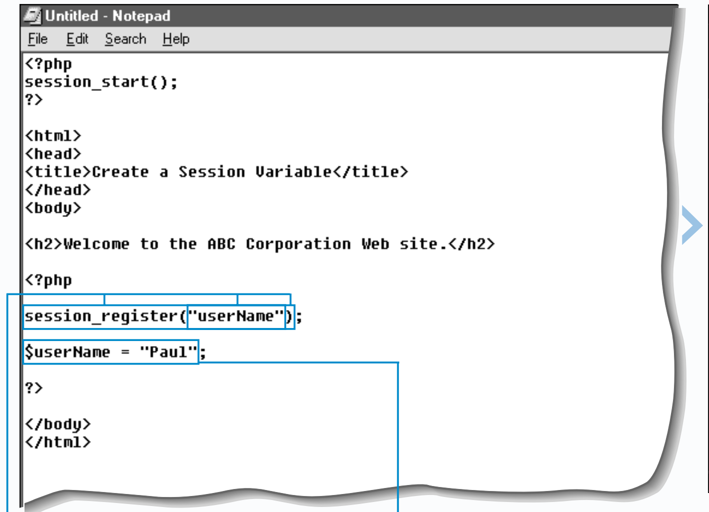
```
session_unregister("userName");
```

To remove all the session variables that have been created for the current session, you can use the `session_unset` function. The session will remain open after you use the `session_unset` function, but the variables created during the session will no longer be available.

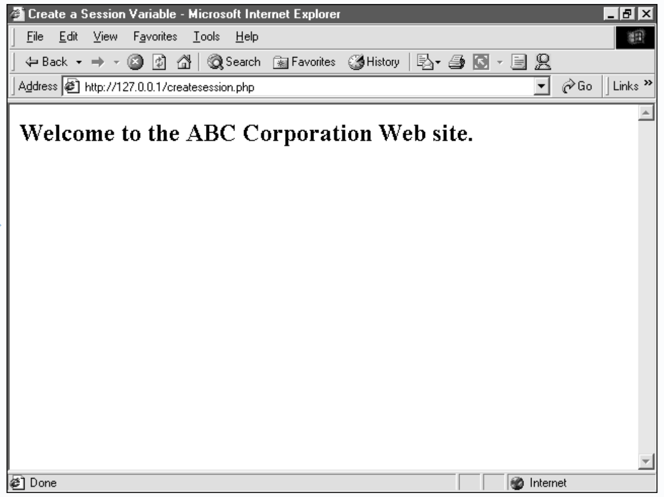
Example:

```
session_unset();
```

CREATE A SESSION VARIABLE



```
<?php
session_start();
?>

<html>
<head>
<title>Create a Session Variable</title>
</head>
<body>

<h2>Welcome to the ABC Corporation Web site.</h2>

<?php

session_register("userName");

$userName = "Paul";

?>

</body>
</html>
```


Create a Session Variable - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1/createsession.php
Go Links
Done
Internet

Welcome to the ABC Corporation Web site.

1 To create a session variable, type **session_register()**.

2 Between the parentheses, type a name for the session variable, enclosed in quotation marks.

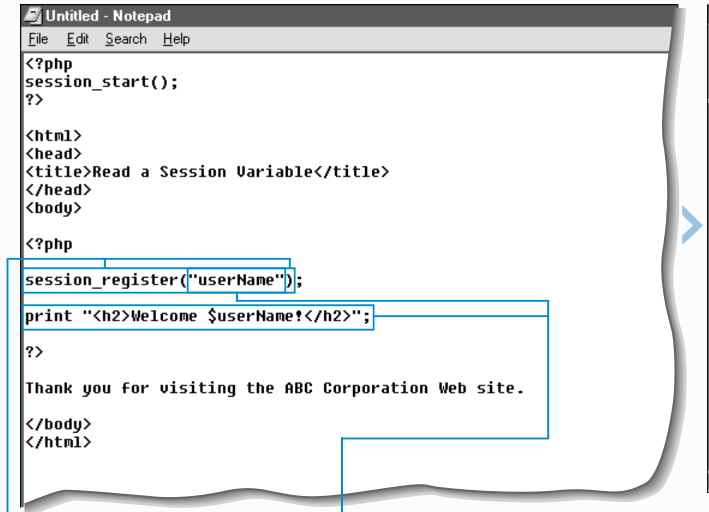
3 Type **$** followed by the name of the session variable. Then type **=** followed by the value you want to assign to the session variable.

■ String values must be enclosed in quotation marks.

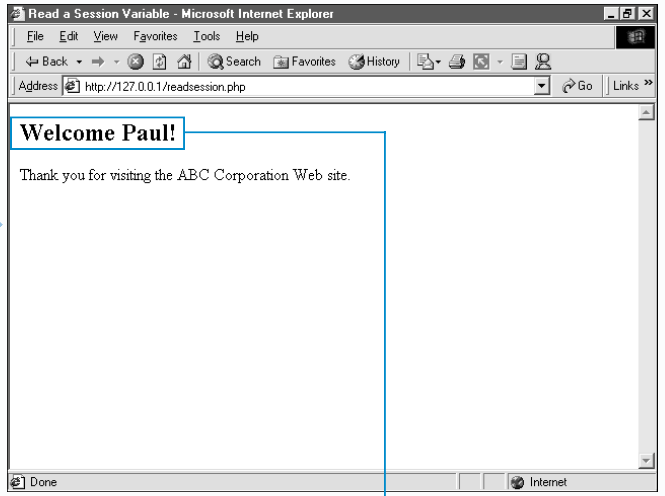
4 Display the PHP page in a Web browser.

■ The session variable is now stored on the computer.

READ A SESSION VARIABLE



```
<?php
session_start();
?>

<html>
<head>
<title>Read a Session Variable</title>
</head>
<body>

<?php

session_register("userName");

print "<h2>Welcome $userName!</h2>";

?>

Thank you for visiting the ABC Corporation Web site.

</body>
</html>
```


Read a Session Variable - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1/readsession.php
Go Links
Done
Internet

Welcome Paul!

Thank you for visiting the ABC Corporation Web site.

1 In the PHP page where you want to read information stored in a session variable, type **session_register()**.

2 Between the parentheses, type the name of the session variable you want to read, enclosed in quotation marks.

3 Type the code that uses the value of the session variable.

4 Display the PHP page in a Web browser.

■ The Web browser displays the result of reading a session variable.

SAVE SESSION INFORMATION TO A FILE

Session information you store in session variables will be available for the duration of the current session. If the session times out or is otherwise terminated by the Web server or client, any session information that has been created will be lost. If you want to be able to access session information after a session has ended, you can save the session information to a file. This is useful when you want to be able to restore the information for a client at a later time.

To save session information to a file, you use the `session_encode` function to encode the information for the session in a string. After encoding the session information, you can write the string returned by the function to a file. You must use the `fopen` function to open the file and use the `fputs` function to write the encoded session information to the file. For more information about writing data to a file, see page 124.

The encoded session information saved in a file includes session information you have stored in session variables, but does not include the session ID of the current session. This enables you to restore the session variables to a new session that has a different session ID. For example, saving session information makes it easy to transfer the information between multiple Web servers used by the same Web site. When a user accesses a different Web server on the Web site, the session information saved in a file can be passed to the new Web server with the client, maintaining continuity even though a new session has been started on the second server.

After saving session information to a file, you can have a PHP page restore the session information from the file. For information about restoring session information from a file, see page 158.

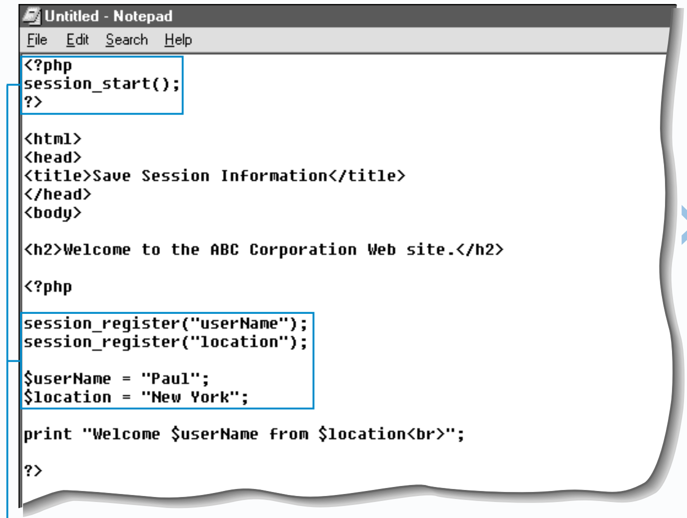
Extra

Saving session information to a file is an effective way of storing relatively small amounts of session information. If you plan to store large amounts of session information, you should consider storing the information in a database. A database provides a more efficient and versatile means of storing session information. In order to store session information in a database, your Web server must have database capabilities.

Saving session information to a file can be used in conjunction with, or as an alternative to, storing information in cookies. Saving session information to a file can be more reliable than using cookies to store information, since some users may disable cookies in their Web browsers.

If you do not want the file that stores session information to be overwritten each time a client visits the PHP page, you can append data to the file. This is useful when you want to store session information for multiple users in a single file. To access the information for a particular session later, you can search the file for a unique identifier for the information.

SAVE SESSION INFORMATION TO A FILE

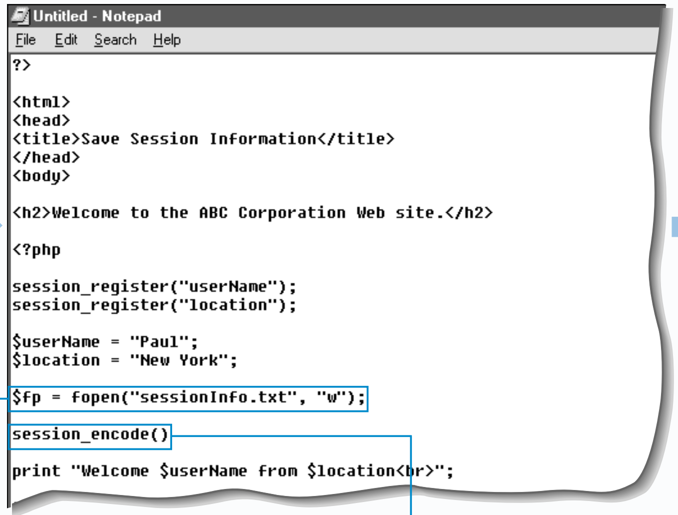


Untitled - Notepad

```
<?php
session_start();
?>

<html>
<head>
<title>Save Session Information</title>
</head>
<body>

<h2>Welcome to the ABC Corporation Web site.</h2>

<?php

session_register("userName");
session_register("location");

$userName = "Paul";
$location = "New York";

print "Welcome $userName from $location<br>";

?>
```

1 Type the code that creates the session variables you want to use to store session information.

■ For information about creating session variables, see page 154.

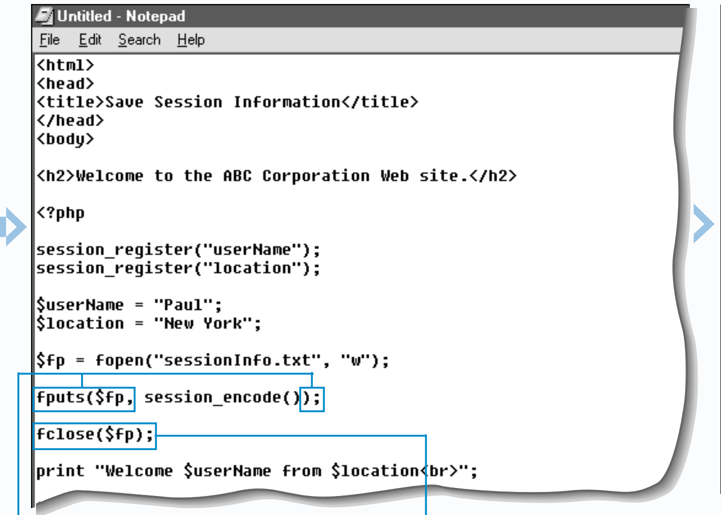


Untitled - Notepad

```
?>

<html>
<head>
<title>Save Session Information</title>
</head>
<body>

<h2>Welcome to the ABC Corporation Web site.</h2>

<?php

session_register("userName");
session_register("location");

$userName = "Paul";
$location = "New York";

$fp = fopen("sessionInfo.txt", "w");

session_encode()

print "Welcome $userName from $location<br>";
```

2 To be able to store the encoded session information in a file, type the code that opens the file for writing.

3 To encode the session information in a string, type **session_encode()**.

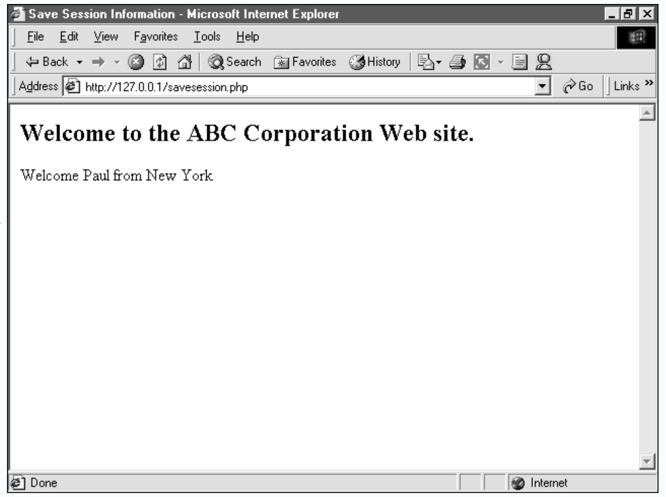


Untitled - Notepad

```
<html>
<head>
<title>Save Session Information</title>
</head>
<body>

<h2>Welcome to the ABC Corporation Web site.</h2>

<?php

session_register("userName");
session_register("location");

$userName = "Paul";
$location = "New York";

$fp = fopen("sessionInfo.txt", "w");

fputs($fp, session_encode());

fclose($fp);

print "Welcome $userName from $location<br>";
```

4 Type the code that writes the string returned by the `session_encode` function to the file.

5 Type the code that closes the file.



Save Session Information - Microsoft Internet Explorer

Address http://127.0.0.1/savesession.php

Welcome to the ABC Corporation Web site.

Welcome Paul from New York

File Edit View Favorites Tools Help
Back Search Favorites History
Go Links
Done
Internet

6 Display the PHP page in a Web browser.

■ The session information has been written to the file.

RESTORE SESSION INFORMATION FROM A FILE

You can have a PHP page restore session information you stored in a file. This allows you to make the session information from a previous session available to a new session. For example, if you saved session variables containing a user's login information to a file, you can restore the information when the user begins a new session. This saves the user from having to re-enter the information each time a new session is started. Restoring session information from a previous session does not alter the session ID for the current session.

To restore session information from a file, you first open the file for reading using the `fopen` function. For more information about opening a file for reading, see page 126. The `session_decode` function can then be used to decode and restore the session information from the file. The `session_decode` function takes a string of text containing encoded session information as its argument. You can use the `fgets` function to retrieve the string that contains the session information you want to restore from the file. For more information about the `fgets` function, see page 126.

When session information is restored from a file, the session variables stored in the file are automatically available to the PHP page. This means that you do not need to recreate the variables in the page.

After storing session information in a file, a new session must be started before the session information can be restored from the file. You should use the `session_start` function on a PHP page that restores session information to start a new session. The `session_start` function must be placed before any HTML code on the PHP page. For more information about the `session_start` function, see page 152.

Extra

If you use the `session_decode` function within another function, the scope of the session variables restored from a file will be limited to that function. If you want to be able to use the session variables outside the function, you must use the `global` keyword to give the session variables returned by the `session_decode` function global scope.

Example:

```
function getname()
{
    $fp = fopen("sessionInfo.txt", "r");
    global $userName, $location;
    session_decode(fgets($fp, 4096));
    fclose($fp);
}
```

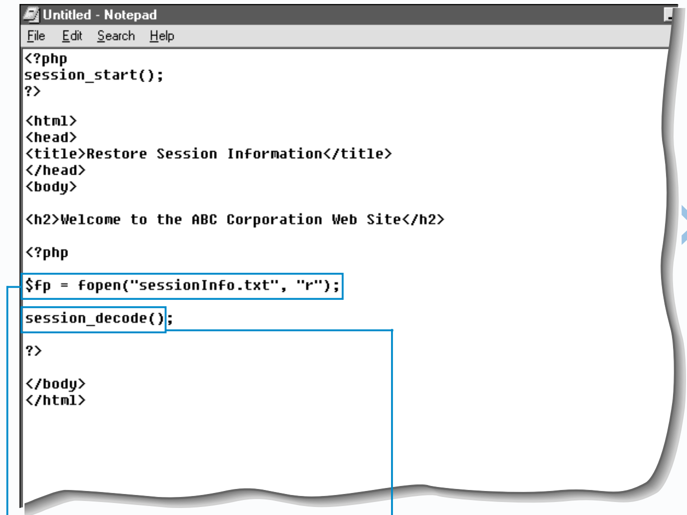
As with any file you want to access from within a PHP page, the file that stores session information you want to restore must have the appropriate file and operating system permissions in order for you to retrieve information from the file. For more information about file and operating system permissions, consult the documentation for your Web server and operating system.

Session information stored in a file is encoded in a format that is easy to read and understand. You can open the file that stores session information in a text editor or word processor to view the session information before restoring the information in a PHP page. Viewing session information can be useful for troubleshooting session-related problems.

Example:

```
userName|s:4:"Paul";location|s:8:"New York";
```

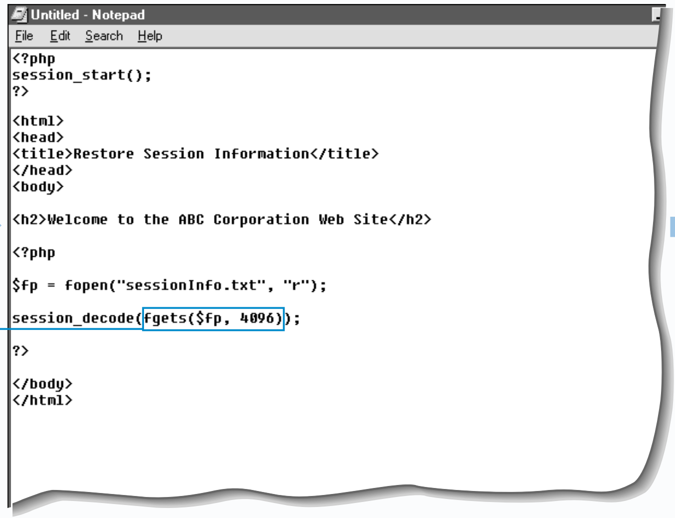
RESTORE SESSION INFORMATION FROM A FILE



```
<?php
session_start();
?>

<html>
<head>
<title>Restore Session Information</title>
</head>
<body>

<h2>Welcome to the ABC Corporation Web Site</h2>

<?php

$fp = fopen("sessionInfo.txt", "r");

session_decode();

?>

</body>
</html>
```

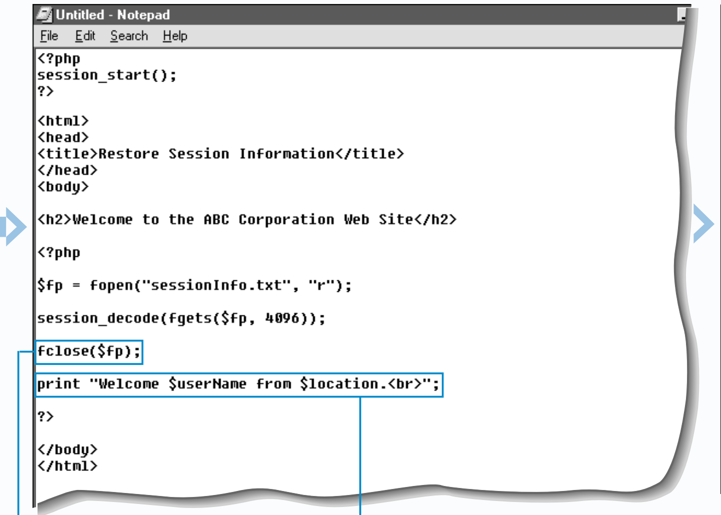
1 To open the file that contains the session information you want to restore, type the code that opens the file for reading.

2 To restore the session information stored in the file, type **session_decode()**.



```
<?php
session_start();
?>

<html>
<head>
<title>Restore Session Information</title>
</head>
<body>

<h2>Welcome to the ABC Corporation Web Site</h2>

<?php

$fp = fopen("sessionInfo.txt", "r");

session_decode(fgets($fp, 4096));

?>

</body>
</html>
```

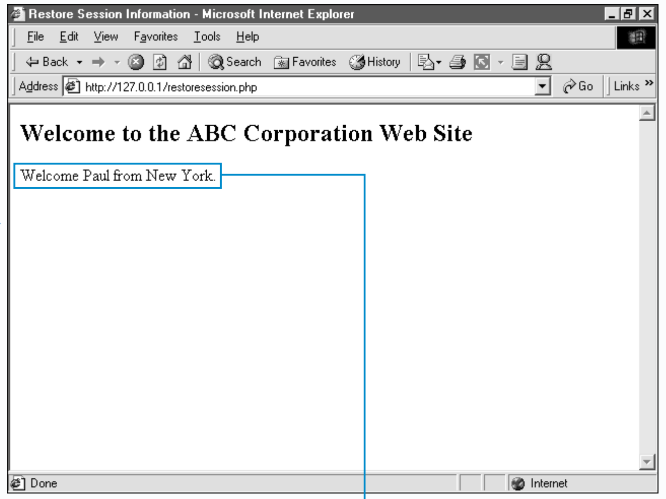
3 Between the parentheses, type the code that uses the `fgets` function to retrieve the information stored in the file.



```
<?php
session_start();
?>

<html>
<head>
<title>Restore Session Information</title>
</head>
<body>

<h2>Welcome to the ABC Corporation Web Site</h2>

<?php

$fp = fopen("sessionInfo.txt", "r");

session_decode(fgets($fp, 4096));

fclose($fp);

print "Welcome $userName from $location.<br>";

?>

</body>
</html>
```

4 Type the code that closes the file.

5 Type the code that uses the session information restored from the file.


Restore Session Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://127.0.0.1/restoresession.php
Go Links
Done
Internet

Welcome to the ABC Corporation Web Site

Welcome Paul from New York.

6 Display the PHP page in a Web browser.

■ The Web browser displays the result of restoring session information from a file.