S. S Jain Subodh P.G. (Autonomous) College

SUBJECT - PHP

TITLE - Cookies and Session Management

Cookies and Session Management

-by Praveen Choudhary

Contents

Cookies

- **A cookie** is a small text file that contains a small amount of information about a user visiting your site and is stored on the site visitor's computer by their browser.
- Because the cookie is stored on the user’s computer, it does not require any server space no matter how many users you have.
- You can use cookies to save user preferences, customize data, remember the last visit, or to keep track of items in an order while a user browses.

Limitation

- The cookie specification introduced by Netscape also places limits on cookies. These limits are:
  - 310 total cookies.
  - 4 kilobytes per cookie
  - 20 cookies per server or domain.

Privacy

- Cookies can only be read by the site that created them, or a site 'underneath' the site that created them. This prevents other websites from stealing cookies.

Types of cookies

There are three different types of cookies.

- **First Party Cookies** are written by your site and can only be read by your site.
- **Third Party Cookies** are created by advertising in your page that is loaded from a third party site. These can only be read by the advertising code on any site displaying the same ads.
- **Session Cookies** are not actually written to a file but are stored in the browser itself. These cookies only last as long as the browser is open.

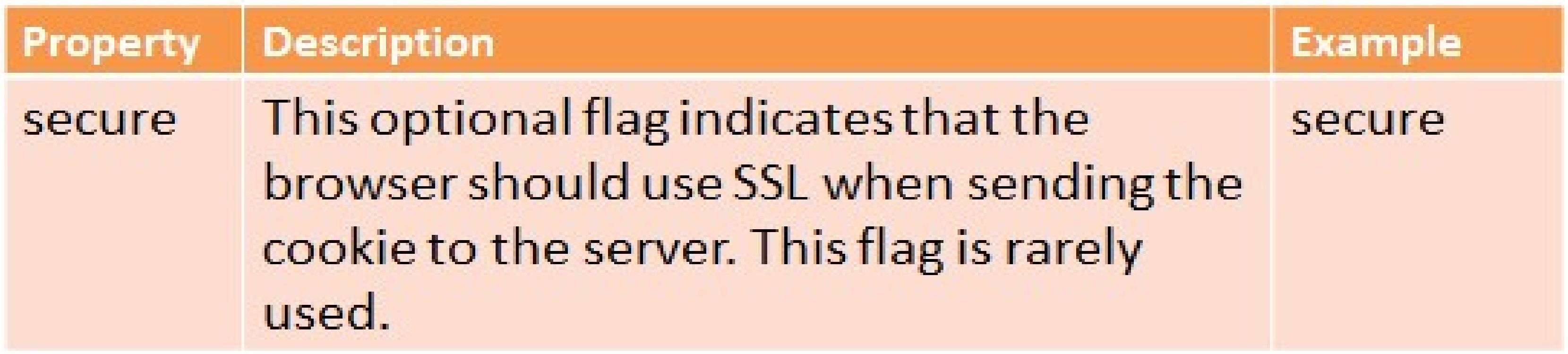
Setting a cookie

Property	Description	Example
name=value	This sets both the cookie's name and its value.	username=matt
expires=*date*	This optional value sets the date that the cookie will expire on. The date should be in the format returned by the **toGMTString() or toUTCString()** methods of the **Date** object. If the expires value is not given, the cookie will be destroyed the moment the browser is closed.	expires= 13/06/2003 00:00:00
path=*path*	The *path* gives you the chance to specify a directory where the cookie is active. So if you want the cookie to be only sent to pages in the directory cgi-bin, set the path to /cgi-bin. Usually the path is set to /, which means the cookie is valid throughout the entire domain.	path=/tutorials/

Setting a cookie

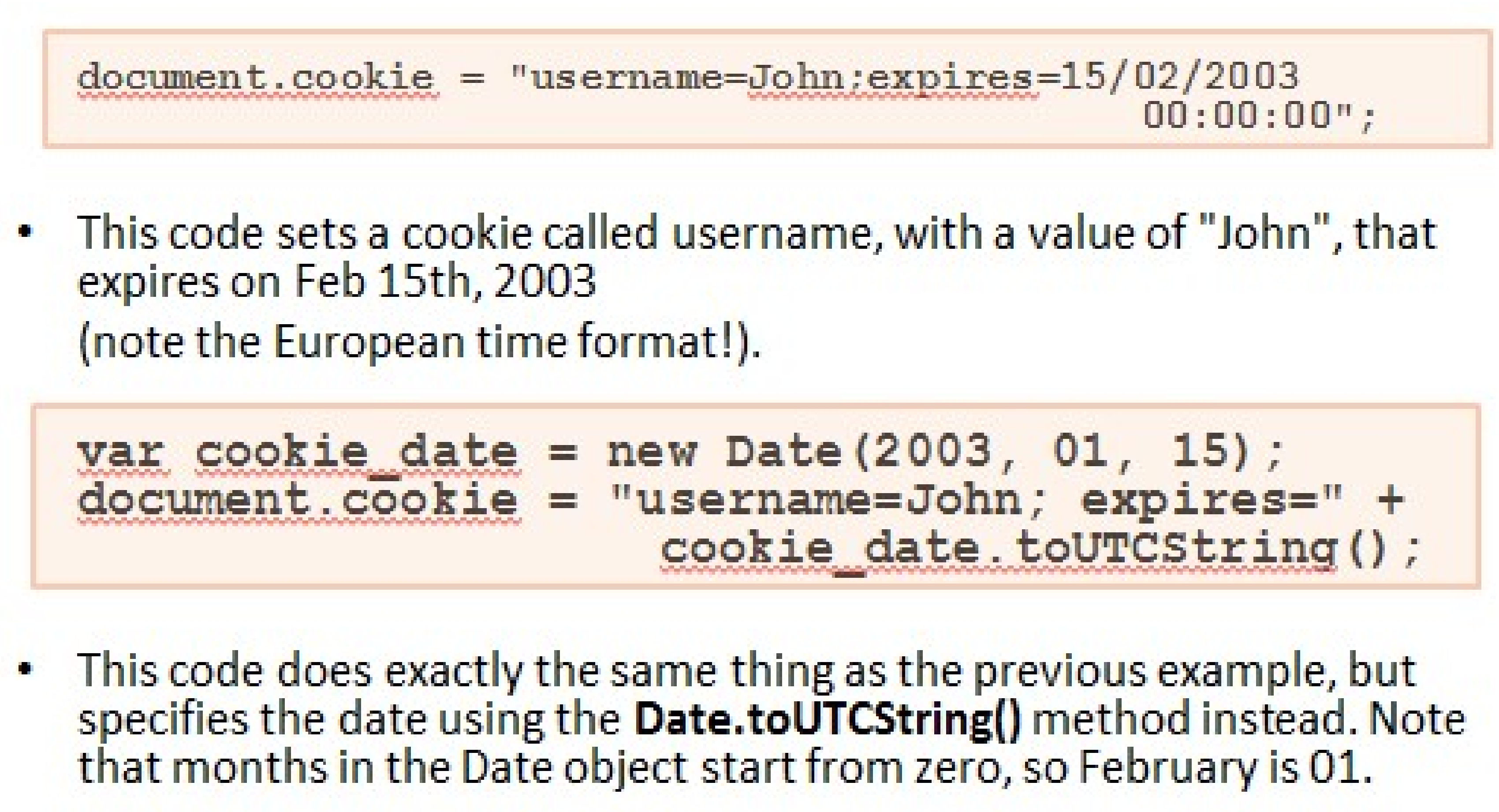
Property	Description	Example
name=value	This sets both the cookie's name and its value.	username=matt
expires=*date*	This optional value sets the date that the cookie will expire on. The date should be in the format returned by the **toGMTString() or toUTCString()** methods of the **Date** object. If the expires value is not given, the cookie will be destroyed the moment the browser is closed.	expires= 13/06/2003 00:00:00
path=*path*	The *path* gives you the chance to specify a directory where the cookie is active. So if you want the cookie to be only sent to pages in the directory cgi-bin, set the path to /cgi-bin. Usually the path is set to /, which means the cookie is valid throughout the entire domain.	path=/tutorials/

Setting a cookie



Property	Description	Example
secure	This optional flag indicates that the browser should use SSL when sending the cookie to the server. This flag is rarely used.	secure

A few examples of cookie setting



```
document.cookie = "username=John;expires=15/02/2003
                                              00:00:00";
```

- This code sets a cookie called username, with a value of "John", that expires on Feb 15th, 2003
  (note the European time format!).

```
var cookie_date = new Date(2003, 01, 15);
document.cookie = "username=John; expires=" +
                    cookie_date.toUTCString();
```

- This code does exactly the same thing as the previous example, but specifies the date using the **Date.toUTCString()** method instead. Note that months in the Date object start from zero, so February is 01.

Examples of cookie setting

```
document.cookie = "logged_in=yes";
```

- This code sets a cookie called logged_in, with a value of "yes". As the expires attribute has not been set, the cookie will expire when the browser is closed down.

```
var cookie_date = new Date(); //current date & time
cookie_date.setTime(cookie_date.getTime() - 1);
document.cookie = "logged_in=; expires=" +
                        cookie_date.toUTCString();
```

- This code sets the logged_in cookie to have an expiry date one second before the current time - this instantly expires the cookie. A handy way to delete cookies!

Examples of cookie setting

- Strictly speaking, we should be ***escaping*** our cookie values - encoding non-alphanumeric characters such as spaces and semicolons. This is to ensure that our browser can interpret the values properly. Fortunately this is easy to do with JavaScript's **escape()** function.

  For example:

```
document.cookie = "username=" + escape("John
            Smith") + "; expires=15/02/2003
                                   00:00:00";
```

A function to set a cookie

Setting cookies will be a lot easier if we can write a simple function to do stuff like escape the cookie values and build the document.cookie string.

```
function set_cookie(name, value, exp_y, exp_m, exp_d, path,
                                                          domain,
   secure) {
    var cookie_string = name + "=" + escape(value);
    if (exp_y) {
      var expires = new Date(exp_y, exp_m, exp_d);
      cookie_string += "; expires=" + expires.toGMTString();
    }
    if (path)   cookie_string += "; path=" + escape(path);
    if (domain) cookie_string += "; domain=" + escape(domain);
    if (secure) cookie_string += "; secure";
    document.cookie = cookie_string;
}
```

A function to set a cookie

- For example, to use this function to set a cookie with no expiry date:

```
set_cookie("username", "John Smith”);
```

- To set a cookie with an expiry date of 15 Feb 2003:

```
set_cookie("username", "John Smith",
2003,                              01,
15);
```

- To set a secure cookie with an expiry date and a domain of elated.com, but no path:

```
set_cookie("username", "John Smith",
2003, 01, 15, "", "elated.com",
"secure”);
```

A function to delete a cookie

Another useful cookie-handling function is provided below. This function will "delete" the supplied cookie from the browser by setting the cookie's expiry date to one second in the past

```
function delete_cookie(cookie_name)
{
  var cookie_date = new Date();
  cookie_date.setTime(cookie_date.getTime(
  ) - 1); document.cookie = cookie_name +=
  "=; expires="                          +
  cookie_date.toUTCString();
}
```

A function to delete a cookie

- To use this function, just pass in the name of the cookie you would like to delete - for example:

```
delete_cookie("username”);
```

Retrieving cookies

- To retrieve all previously set cookies for the current document, you again use the **document.cookie** property:

  var x = document.cookie;

- This returns a string comprising a list of name/value pairs, separated by semi-colons, for *all* the cookies that are valid for the current document. For example:

  "username=John; password=abc123"

- In this example, 2 cookies have been previously set: username, with a value of "John", and password, with a value of "abc123".

A function to retrieve a cookie

Usually we only want to read the value of one cookie at a time, so a string containing all our cookies is not that helpful. So here's another useful function that parses the document.cookies string, and returns just the cookie we're interested in:

```
function get_cookie(cookie_name)
{
  var results = document.cookie.match('(^|;) ?' +
                    cookie_name + '=([^;]*)(;|$)’);
  if (results)
      return (unescape(results[2]));
  else return null;
}
```

A function to retrieve a cookie

- Using the function is easy. For example, to retrieve the value of the username cookie:

```
var x = get_cookie
("username");
```

Sessions

- Sessions are a combination of a server-side cookie and a client-side cookie.

- Client-side cookie simply holds a value (session token) that uniquely identifies the client to the server, and corresponds to a data file on the server.

- Thus, when the user visits the site, their browser sends the reference code to the server, which loads the corresponding data.

HTTP session token

- A **session token** is a unique identifier that is generated and sent from a server to a client to identify the current interaction session.

- The client usually stores and sends the token as an HTTP cookie and/or sends it as a parameter in GET or POST queries.

HTTP session token

- The reason to use session tokens is that the client only has to handle the identifier—all session data is stored on the server (usually in a database, to which the client does not have direct access) linked to that identifier.

- Examples of the names that some programming languages use when naming their HTTP cookie include
  JSESSIONID (JSP),
  PHPSESSID (PHP),
  ASPSESSIONID (ASP).

Sessions advantages

- Your server-side cookie can contain very large amounts of data with no hassle - client-side cookies are limited in size
- Your client-side cookie contains nothing other than a small reference code - as this cookie is passed each time someone visits a page on your site, you are saving a lot of bandwidth by not transferring large client-side cookies around
- Session data is much more secure - only you are able to manipulate it, as opposed to client-side cookies which are editable by all