

PHP - Interacción con el cliente

Cookies

Cookies

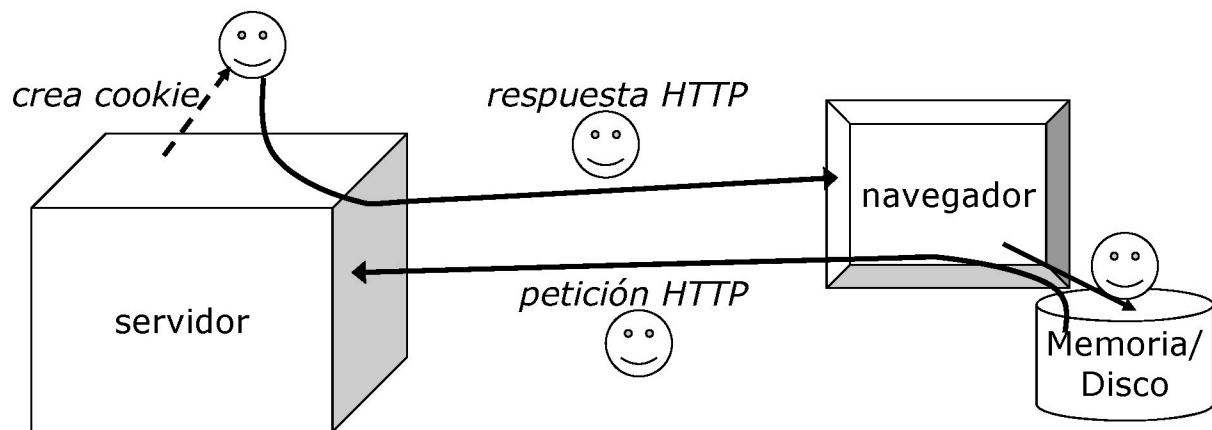
- HTTP es un protocolo SIN ESTADO
 - No se guarda información de la sesión/historia pasada
 - (Esto simplifica el protocolo)
- Uso de *cookies*
 - Un *cookie* es un *string* que se pasa en una cabecera HTTP y que el navegador puede guardar en un pequeño fichero de texto
 - En archivos temporales del navegador correspondiente
 - El *cookie* se reenvía luego al servidor HTTP con cada petición del cliente a ese servidor
 - Los *cookies* **no** pueden capturar información del cliente
 - Sólo recuerdan información proporcionada por el usuario al servidor (es el servidor quien los crea)
 - Usos
 - Guardar las preferencias del usuario
 - Reconocimiento de usuarios
 - El cookie puede guardar un identificador que permite al servidor acceder a todos los datos almacenados en su base de datos
 - Gestión de sesiones

Cookies

- Atributos
 - Par (Nombre, Valor)
 - Comentario (se puede presentar al usuario)
 - P.ej. para explicar para qué se usa el cookie (política del sitio web)
 - Especificación de las páginas y dominios a los que se puede enviar el *cookie*
 - Fecha y hora de expiración
 - Permite controlar por ejemplo el tiempo máximo de una sesión antes de volver a pedir login
 - Requiere o no una página segura
 - Versión
- Tamaño máximo: 4Kbytes (Normalmente ocupan alrededor de 100 bytes)
- Seguridad
 - Los *cookies* sólo pueden ir al dominio especificado
 - No conviene poner información sensible en el *cookie*, mejor utilizar un identificador en el *cookie* que sirva de clave de acceso en la base de datos del servidor

Cookies

- Funcionamiento del mecanismo de cookies



Programación de cookies con PHP

- Creación y envío de un cookie: **setcookie()**
 - El cookie se envía en la cabecera del mensaje de respuesta HTTP
 - El método se tiene que llamar antes de que la página PHP genere cualquier resultado (antes de cualquier sentencia echo o print)
- El navegador recordará el nombre y valor del cookie y lo enviará al servidor en peticiones posteriores
 - Los cookies recibidos con la solicitud del cliente se pueden consultar en **\$_COOKIE[]**

```
<?php
$cookie1="nombre";
$valor1="Juan";
$expira1=time()+3600*24; // expira en 24 horas
setcookie($cookie1, $valor1, $expira1);
?>
<html>
<head><title>Hola Cookie</title></head>
<body>
<?php
echo "<h1>Hola $_COOKIE[$cookie1]</h1>";
?>
</body>
</html>
```

Programación de cookies con PHP

- **setcookie(\$nombre, \$valor, \$tvida, \$ruta, \$dominio, \$seguridad)**
 - Las cookies tienen un \$nombre y un \$valor
 - El nombre no debe coincidir con el de un control de formulario porque en \$_REQUEST se guardaría solo el valor del cookie, no el del control
 - Se puede indicar un tiempo de vida del cookie
 - Si no se indica, el cookie se elimina al cerrar el navegador
 - El tiempo se indica como tiempo Unix, esto es, el número de segundos desde el 1 de Enero de 1970
 - La función **time()** devuelve el número de segundos que han pasado desde esa fecha
 - Se indicará como \$tvida=time()+\$numeroSegundos;
 - \$ruta y \$dominio determinan páginas y dominios a los que se puede enviar el cookie
 - \$seguridad indica si se mandará el cookie únicamente en conexiones seguras https (TRUE) o indistintamente (FALSE)

```
$cookie1="nombre";
$valor1="Juan";
$tvida=time()+3600*24; // expira en 24 horas
setcookie($cookie1, $valor1, $tvida, ".dominio.com");
```

Programación de cookies con PHP

- Modificación del valor de un cookie
 - Basta con crear nuevamente el cookie con otro valor
- Borrado de un cookie
 - Se consigue creando el cookie con un tiempo de expiración del pasado
`setcookie("nombre", "valor", time()-60);`
- Uso de un cookie
 - Consultando su existencia en la superglobal `$_COOKIE`
 - Conviene comprobar antes que se haya recibido

```
if (isset($_COOKIE["nombre"])) {  
    echo "<p>El valor del cookie nombre es $_COOKIE[nombre]</p>";  
} else {  
    echo "<p>No se ha recibido el cookie nombre.</p>";  
}
```

PHP - Interacción con el cliente

Sesiones

Sesiones de usuario

- Una sesión determina un contexto que relaciona las acciones del cliente sobre un sitio web
 - Normalmente las variables son destruidas cuando acaba la ejecución de una página PHP
 - A veces es necesario guardar cierta información entre una página y otra durante la navegación de un cliente
- Las sesiones tienen un ciclo de vida
 - Inicio de sesión
 - Login de usuario
 - Actividad del usuario
 - Flujo lógico de operaciones de consulta/modificación de información
 - Cierre de sesión
 - Explícito por el usuario
 - Por expiración de un tiempo de inactividad

Mecanismos para implementar sesiones

- Para gestionar las sesiones sobre HTTP (protocolo sin estado) se podrían usar varios mecanismos
 - Un cookie: PHPSESSID
 - Cuando se inicia una sesión en una página, el intérprete PHP comprueba la presencia de este cookie y la establece si no existe
 - El identificador de sesión en la cookie PHPSESSID permite identificar únicamente ese cliente en el servidor
 - Variables de identificación de sesión
 - Normalmente el usuario navega de una página a otra del mismo sitio
 - Se podría crear un identificador único al visitar la primera página si no existiera y pasarlo en las siguientes páginas
 - Como un argumento en cada GET

```
<a href="siguiente.php?sesion=<?php echo $_GET['id_sesion'];?>">Siguiente página</a>
```

- En formularios, como un argumento oculto en el POST

```
<form action=siguiente.php method=post>  
Campos del formulario  
<input type=hidden name=sesion value="<?php echo $_GET['id_sesion'];?>" >  
</form>
```

Sesiones en PHP

- PHP ofrece un mecanismo de gestión de sesiones que abstrae al programador de cuál de esos mecanismos se utilice
 - Normalmente usa cookies si el navegador lo permite, y si no el identificador de sesión en GET y POST
 - Las variables de la sesión se guardan en un fichero en el servidor con el nombre del identificador
- Gestión de sesiones en PHP
 1. Iniciar una nueva sesión: **session_start();**
 - Se tiene que invocar antes de escribir cualquier cosa con echo o print
 - Porque el identificador de la sesión se envía en la cabecera de respuesta HTTP
 2. Uso de la variable superglobal **\$_SESSION**
 - Todas las variables de la sesión se incluirán y se pueden acceder, entre página y página de una misma sesión, en el array **\$_SESSION**
 - Siempre se tiene que haber invocado antes **session_start()** al principio de la página (así PHP prepara las variables correspondientes a la sesión)
 3. Cerrar sesión: **session_destroy();**

Ejemplo de sesión PHP

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

<html>
<head><title>Ejemplo de sesiones PHP</title></head>
<body>
<h1>Ejemplo de sesiones con PHP</h1>

<?php
if (!isset($SESSION['contador'])) {
    echo "<p>Bienvenido por primera vez</p>";
    $SESSION['contador']=1;
}
else {
    $SESSION['contador']++;
    echo "<p>Ya nos has visitado ". $SESSION['contador'] ." veces.</p>";
}
?>

</body>
</html>
```

Ejercicio – Sesiones

- Probar a crear dos páginas distintas para una misma sesión
 - En la primera crear la sesión

```
<?php  
session_start();  
$_SESSION["nombre"] = "Juan";  
print "<p>Se ha guardado tu nombre.</p>";  
?>
```

- En la segunda usar alguna variable de la sesión creada

```
<?php  
session_start();  
print "<p>Hola $_SESSION[nombre], vemos que sigues por aquí</p>";  
?>
```

Sesiones

- Una sesión se puede destruir con la función **session_destroy()**
 - Pero las variables correspondientes pueden seguir usándose en esa ejecución del script
- Los datos de una sesión en `$_SESSION` se guardan durante un tiempo predeterminado de 24 minutos
 - La directiva de configuración **session.gc_maxlifetime** permite configurar este valor por defecto
 - `ini_set(string varConfig, valor)` permite modificar ese valor
 - Tiene que invocarse antes de `session_start()`
- Los valores de `$_SESSION` se pueden borrar también como en cualquier otra matriz mediante la función **unset()**

Otras funciones para gestión de sesiones

- **session_register("variable")**
 - Registra la variable en la sesión
 - Se debe especificar el nombre de la variable sin \$
 - Las asignaciones a esa variable se mantendrán en futuras invocaciones dentro de la sesión
 - Si no se hubiera invocado session_start(), esta función lo hace
- **session_unregister("variable")**
 - Elimina la variable en la sesión
- **session_is_registered("variable")**
 - Comprueba si la variable está registrada en la sesión
- **session_id()**
 - Devuelve el identificador de la sesión