

# PHP – Interacción con el cliente

---

## Aplicaciones Web/Sistemas Web



**Juan Pavón Mestras**  
**Dep. Ingeniería del Software e Inteligencia Artificial**  
**Facultad de Informática**  
**Universidad Complutense Madrid**

*Material bajo licencia Creative Commons*



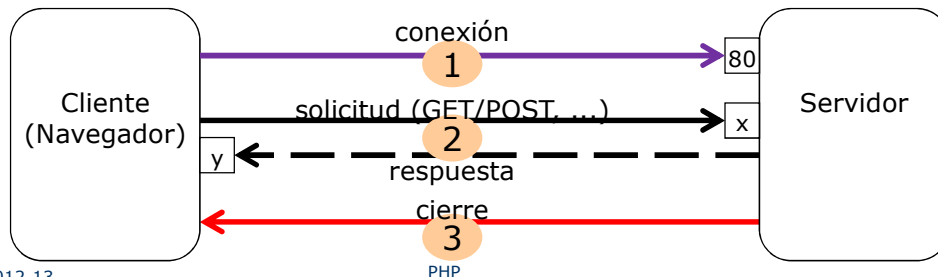
# PHP - Interacción con el cliente

---

Formularios

## Protocolo HTTP

- El navegador (cliente, *user agent*) solicita un recurso (página HTML, imagen, video, etc.) a un servidor
  - Solicitud: método que se utiliza GET, POST, PUT, HEAD, etc.
  - Campos de cabecera
  - Línea en blanco
  - Cuerpo del mensaje (texto): puede llevar parámetros del formulario
- El servidor responde enviando el recurso o con un mensaje de error
  - Línea de estado: código del estado (OK, Error) y texto asociado
  - Campos de cabecera
  - Línea en blanco
  - Cuerpo del mensaje: el recurso solicitado



Juan Pavón - UCM 2012-13

PHP

3

## Paso de parámetros

- La petición del cliente puede llevar varios parámetros
    - Normalmente se obtienen de un formulario
  - Cómo se pasan depende de la acción indicada en el formulario HTML en el que se recogen los datos
    - **GET:** petición de información (operación idempotente)
      - GET `consultatelefono.php?cliente=empresa1`
      - Los parámetros se pasan como pares nombre=valor
      - Se pueden pasar varios parámetros seguidos con &
    - **POST:** peticiones que cambian el estado del servidor
      - Guardar o actualizar datos
      - Enviar email
      - Ordenar datos
- POST `modifica.php?cliente=empresa1&telefono=917892893`

Juan Pavón - UCM 2012-13

PHP

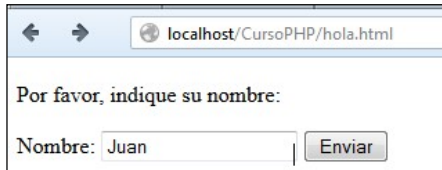
4

## Escenario típico de interacción (con GET)

<http://localhost/CursoPHP/hola.html>

*hola.html*

```
<p>Por favor, indique su nombre:
<form method="get" action="procesaform.php">
Nombre:
<input type="text" name="cliente" />
<input type="submit" value="Enviar">
</form>
</p>
```

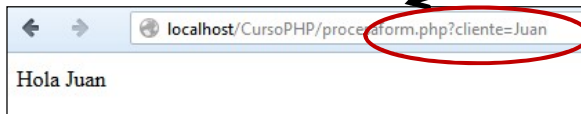


Por favor, indique su nombre:

Nombre:

*procesaform.php*

```
<?php
$cliente=$_REQUEST["cliente"];
echo "Hola $cliente";
?>
```



localhost/CursoPHP/procesaform.php?cliente=Juan

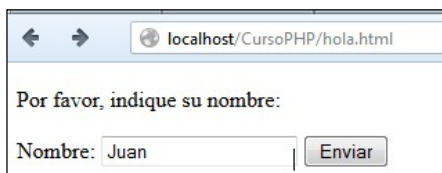
Hola Juan

## Escenario típico de interacción (con POST)

<http://localhost/CursoPHP/hola.html>

*hola.html*

```
<p>Por favor, indique su nombre:
<form method="post" action="procesaform.php">
Nombre:
<input type="text" name="cliente" />
<input type="submit" value="Enviar">
</form>
</p>
```

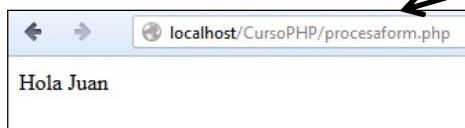


Por favor, indique su nombre:

Nombre:

*procesaform.php*

```
<?php
$cliente=$_REQUEST["cliente"];
echo "Hola $cliente";
?>
```



localhost/CursoPHP/procesaform.php

Hola Juan

## Formularios

- **<form>**
- Conjunto de controles que permiten al usuario interactuar
  - Generalmente para introducir datos y enviarlos al servidor web
  - El navegador envía únicamente los datos de los controles contenidos en el formulario
  - En una misma página puede haber varios formularios que envíen datos al mismo o a diferentes agentes

```
<form action="http://www.miweb.com/procesaform.php" method="post">  
Escribe tu nombre:  
<input type="text" name="nombre" value="" />  
<br/>  
<input type="submit" value="Enviar" />  
</form>
```

### Formulario muy sencillo

Escribe tu nombre:

*POST "/procesaform.php"*  
*nombre="valor"*

## Formularios

- Dentro de un formulario puede haber:
  - Cualquier elemento típico de una página web
    - Párrafos, imágenes, divisiones, listas, tablas, etc.
  - Controles de formularios
    - <input />
    - <button>
    - <select>
    - <optgroup>
    - <option>
    - <textarea>
  - Estructura de formularios
    - <fieldset>
    - <legend>
  - Información para accesibilidad
    - <label> permite mejorar la accesibilidad de los controles

## Formularios

---

- Atributos de <form>
  - **action="URL"**: aplicación del servidor que procesará los datos remitidos (por ejemplo, un script de PHP)
  - **method**: método HTTP para enviar los datos al servidor
    - **GET**: como añadido a la dirección indicada en el atributo action
      - Limitado a 500 bytes
      - Los datos enviados se añaden al final de la URL de la página y por tanto se ven en la barra del navegador
      - Se suele usar cuando se envía información que no modifica el servidor (por ejemplo, términos para una búsqueda)
      - Si no se especifica, los navegadores suelen hacer GET
    - **POST**: en forma separada
      - Puede enviar más información
      - Permite enviar ficheros adjuntos
      - Los datos enviados no se ven en la barra del navegador
      - Se suele usar cuando se envía información que puede modificar el servidor
  - **enctype**: Tipo de codificación al enviar el formulario al servidor
    - "application/x-www-form-urlencoded" o "multipart/form-data"
    - Sólo se indica cuando se adjuntan archivos

## Formularios

---

- **<input />**
  - **type** = "text | password | checkbox | radio | submit | reset | file | hidden | image | button" - Indica el tipo de control que se incluye en el formulario
  - **name** = "texto" - Nombre del control (para que el servidor pueda procesar el formulario)
  - **value** = "texto" - Valor inicial del control
  - **size** - Tamaño inicial del control (en píxeles, salvo para campos de texto y de password que se refiere al número de caracteres)
  - **maxlength** = "numero" - Máximo tamaño de texto y de password
  - **checked** = "checked" - Opción preseleccionada para los controles checkbox y radiobutton
  - **disabled** = "disabled" - El control aparece deshabilitado y su valor no se envía al servidor junto con el resto de datos
  - **readonly** = "readonly" - El contenido del control no se puede modificar
  - **src** = "url" - Para el control que permite crear botones con imágenes, indica la URL de la imagen que se emplea como botón de formulario
  - **alt** = "texto" - Descripción del control

## Formularios

---

### ■ Cuadro de texto

Nombre <br/>

```
<input type="text" name="nombre" value="" />
```

Nombre

- Se enviará al servidor cuando se pulse un botón de enviar
- El nombre asignado en *name* tiene que concordar con el que se use en la aplicación en el servidor
  - No se deben utilizar caracteres problemáticos en programación (espacios en blanco, acentos y caracteres como ñ o ç)
- *value* permite establecer un valor inicial en el cuadro de texto

### ■ Contraseñas

Contraseña <br/>

```
<input type="password" name="contrasena" value="" />
```

Contraseña

- Igual que el cuadro de texto por el valor introducido no se ve

## Formularios

---

### ■ Cuadro de texto de varias líneas

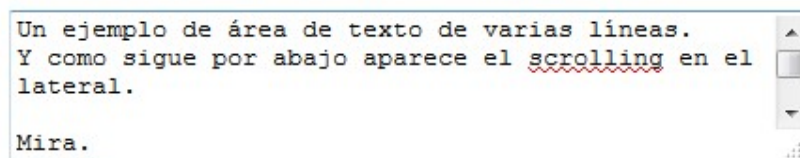
Nombre <br/>

```
<textarea name="nombre" rows="4" cols="50">
```

Contenido inicial del cuadro de texto

```
</textarea>
```

- filas: número de filas visibles (sale una barra de desplazamiento si se hay más)
- columnas: anchura en caracteres

A screenshot of a web browser showing a multi-line text area. The text area contains the text: "Un ejemplo de área de texto de varias líneas. Y como sigue por abajo aparece el scrolling en el lateral. Mira." The text is wrapped, and a vertical scrollbar is visible on the right side of the text area, indicating that the content is longer than the visible area.

## Formularios

---

- Botón de envío de formulario

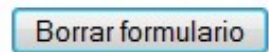
`<input type="submit" name="enviar" value="Enviar" />`



- El navegador se encarga de enviar automáticamente los datos cuando el usuario pincha el botón

- Botón de reseteo de formulario

`<input type="reset" name="borrar" value="Borrar formulario" />`



- El navegador borra toda la información introducida y muestra el formulario en su estado original

## Formularios

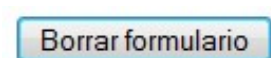
---

- Botones en general: **<button>**

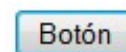
`<button type="submit">Enviar</button>`



`<button type="reset">Borrar formulario</button>`



`<button type="button">Botón</button>`



- El navegador se encarga de enviar automáticamente los datos cuando el usuario pincha el botón

## Formularios

### ■ Casillas de verificación (*checkbox*)

Lenguajes de programación: <br/>

<input name="java" **type="checkbox"** value="on"/> Java

<input name="cplusplus" type="checkbox" value="on"/> C++

<input name="csharp" type="checkbox" value="on"/> C#

<input name="otros" type="checkbox" value="on"/> Otros

- value indica el tipo de casilla: on/off, yes/no, true/false

Lenguajes de programación:

☐ Java ☐ C++ ☐ C# ☐ Otros

### ■ Radiobutton

Sexo <br/>

<input **type="radio"** name="sexo" value="hombre" checked="checked" />

Hombre

<input type="radio" name="sexo" value="mujer" /> Mujer

Sexo  
☒ Hombre  
☐ Mujer

## Ejercicio

### ■ Crear una página PHP que genere un formulario con los siguientes campos:

- Un campo de texto para preguntar el nombre
- Un campo radio button para seleccionar el sexo
- Un campo checkbox para seleccionar lenguajes de programación

Al hacer submit se envían los datos al servidor con POST y el servidor devuelve una página que muestra los datos recopilados.

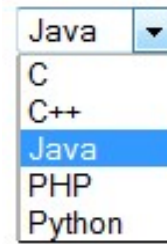
- Para probar lo que se envía al servidor, usar las herramientas de desarrollador del navegador
- También se puede probar con GET y se verán los parámetros en la URL resultante al hacer submit



## Formularios

### ■ Listas de selección

```
<form action="">
<select name="lenguajes">
  <option value="c">C</option>
  <option value="cplusplus">C++</option>
  <option value="java" selected>Java</option>
  <option value="php">PHP</option>
  <option value="python">Python</option>
</select>
</form>
```



#### ■ Atributos de option:

- *value* determina el valor que se envía al servidor
- *selected* permite definir la opción por defecto

## Formularios

### ■ Agrupación de elementos

- Permite ver mejor las partes de un formulario agrupando elementos relacionados
- `<legend>` es el título que se visualiza con el grupo

```
<form action="">
<fieldset>
  <legend>Información personal:</legend>
  Nombre: <input type="text" size="50"><br>
  E-mail: <input type="text" size="50"><br>
  Ciudad: <input type="text" size="20">
</fieldset>
</form>
```

Información personal:

Nombre:	<input size="50" type="text"/>
E-mail:	<input size="50" type="text"/>
Ciudad:	<input size="20" type="text"/>

## Información recibida con la solicitud del cliente

---

- El valor de los parámetros se guarda en **\$\_REQUEST**
  - **\$\_REQUEST** ["nombre-parámetro"]
    - nombre-parámetro es el que en el formulario se indica con el atributo name

<p>Nombre: <input type="text" name="nombre" /></p>
  - Si se quiere depurar se puede ver toda la información recibida con `print_r($_REQUEST);`
- Se pueden usar igualmente las siguientes variables superglobales
  - **\$\_GET** ["nombre-parámetro"]
  - **\$\_POST** ["nombre-parámetro"]
    - Pero **\$\_REQUEST** vale para ambos tipos de solicitudes

## Ficheros en formularios

---

- Incluir un fichero
  - El atributo `enctype` en la etiqueta `<form>` del formulario tiene que ser `multipart/form-data`

```
<form name="fichero" action="procesa_fichero.php" method="post"
enctype="multipart/form-data">
Fichero: <input type="file" name="archivo" />
<input type="submit" value="Enviar">
</form>
```



- Los ficheros recibidos se pueden acceder con **\$\_FILE[]**
  - **\$\_FILE['campoFile']['name']** Nombre del fichero en el cliente
  - **\$\_FILE['campoFile']['type']** Tipo MIME del fichero
  - **\$\_FILE['campoFile']['size']** Tamaño, en bytes, del fichero

## Validación de la información recibida

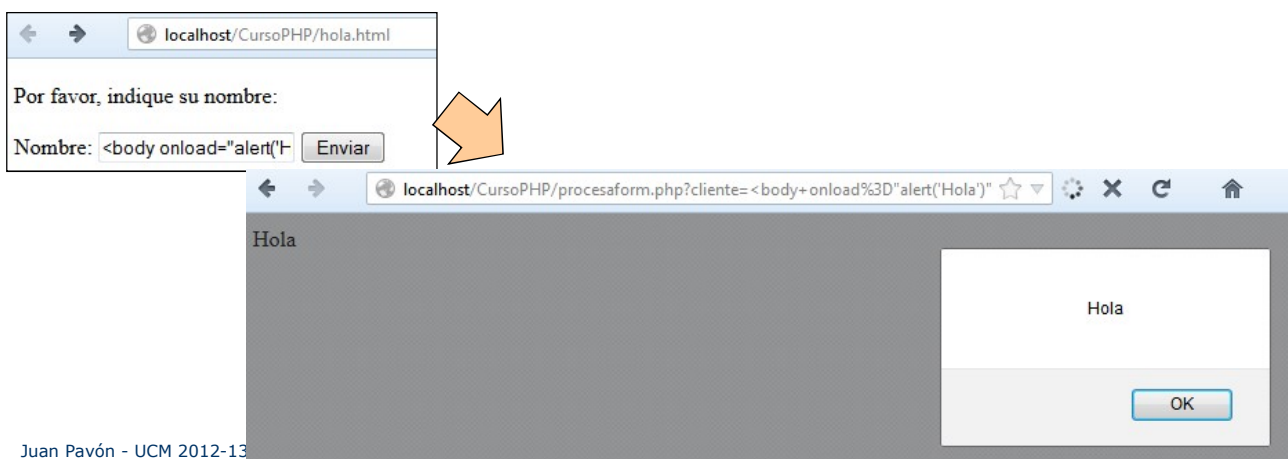
- Los campos de texto de los formularios siempre se reciben
  - Conviene comprobar que no estén vacíos
- Las casillas de verificación y los botones radio solamente están definidos en `$_REQUEST` si se han marcado en el formulario
  - Conviene comprobar que están definidos
- SIEMPRE hay que validar los datos recibidos
  - Texto correcto
    - No vacío (`strlen() > 0`)
    - Eliminar caracteres en blanco (`trim()`)
    - Cuidado con caracteres especiales
  - Números
    - Bien formados
      - Enteros: `intval()`
      - Reales: `floatval()`
    - Rango de valores
  - Dirección de correo electrónico  
`Preg_match('/^[^@\s]+@([a-z0-9]+\.)+[a-z]{2,}$/i', $_POST['email'])`

## Seguridad en las entradas

- Conviene comprobar que no llegue código con `<` y `>`



- Podría ocasionar efectos inesperados



## Seguridad en las entradas

---

- Para evitarlo se usa una función que elimine < y >
  - **strip\_tags(string)**
    - Retira las etiquetas HTML y PHP de un string
  - **htmlspecialchars(string)**
    - Convierte caracteres especiales en entidades HTML
      - & → &amp;
      - " (comillas dobles) → &quot;
      - ' (comilla simple) → &#039;
      - < → &lt;
      - > → &gt;
- También conviene quitar los espacios al principio
  - **trim(string)**
    - Elimina los espacios en blanco iniciales y finales del string
- En resumen, se debería hacer algo así:  
`$cliente=htmlspecialchars(trim(strip_tags($_REQUEST["cliente"])));`

## Codificación de caracteres especiales

---

Carácter	Código
/	%2F
:	%3A
=	%3D
"	%22
'	%27
(espacio)	%20
?	%3F
@	%40
&	%26
\	%5C
~	%7E
	%7C

(también como +)

## Funciones útiles para tratar strings

---

- **substr**(string, posición, [longitud])
  - Devuelve una subcadena de caracteres, a partir de la posición indicada y de longitud la especificada (o hasta el final si no se especifica)
- **strpos**(string1, string2, [posición])
  - Buscan en string1 la primera aparición de string2
  - Si se especifica, se empieza a buscar a partir de la posición indicada
- **htmlspecialchars**(string)
- Reemplaza en el string aquellos caracteres que no son válidos en HTML y los convierte en sus equivalentes válidos (con &)
  - & → &amp;    " → &quot;    < → &lt;    > → &gt;
- **nl2br**(string)
  - Cambia los saltos de línea '\n' por <br>

## Ejercicio

---

- Crear una página con un formulario que recoja información de un nuevo cliente, la valide y la almacene en la base de datos *tienda* como nuevo registro de la tabla *clientes*
  - Si todos los datos son correctos y se almacena bien en la base de datos se mostrará una página indicando que la operación se ha realizado con éxito, mostrando los campos del registro que se han guardado
  - Si hubiera campos con datos incorrectos, volver a mostrar el formulario resaltando dichos campos. Los datos que fueran correctos aparecerán en sus respectivos campos para que el usuario no tenga que volver a introducirlos

# PHP - Interacción con el cliente

---

## Cookies

## Cookies

---

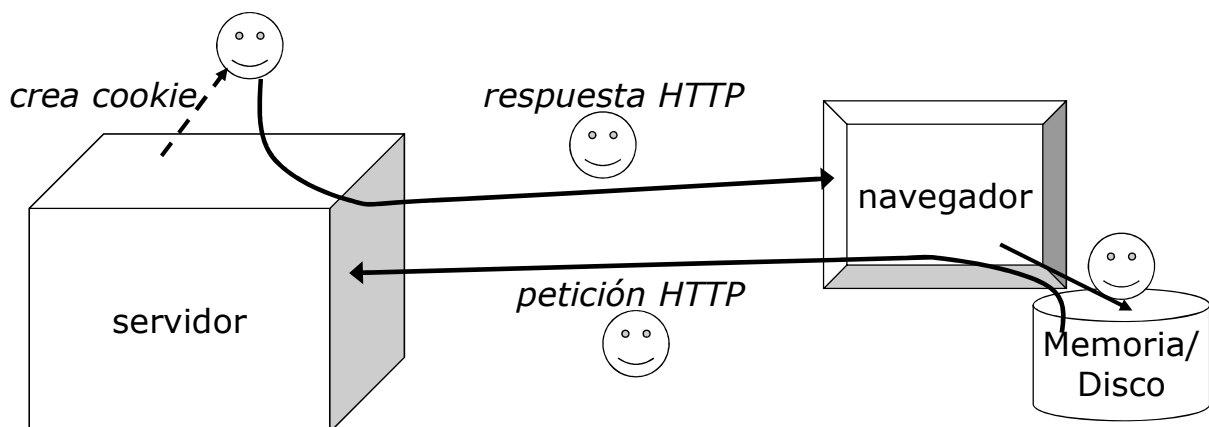
- HTTP es un protocolo SIN ESTADO
  - No se guarda información de la sesión/historia pasada
    - (Esto simplifica el protocolo)
- Uso de *cookies*
  - Un *cookie* es un *string* que se pasa en una cabecera HTTP y que el navegador puede guardar en un pequeño fichero de texto
    - En archivos temporales 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>";  
}
```

## Ejercicios – Cookies

---

- Crea una página PHP que solicite al usuario un nombre y el número de segundos de vida para el cookie asociado al nombre. Comprueba su funcionamiento
- Desarrolla una página PHP que recuerde el color de fondo preferido por un cliente
- Desarrolla una página PHP que compruebe si el navegador permite crear cookies y devuelva un mensaje indicando si los admite o no
- Soluciones a ejercicios similares:  
<http://www.mclibre.org/consultar/php/ejercicios/cookies/cookies.html>

# 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 unívocamente 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'];?>">Siguiendo 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

## Ejercicios – Sesiones

---

- Prueba a contar el número de accesos de un cliente a una página durante una sesión. La página tendrá un botón para iniciar la sesión y otro para cerrarla. También visualizará en cada momento el número de accesos que se han producido a la página durante la sesión
  - Es bastante similar al ejemplo anterior
- Crea una secuencia de páginas que soliciten información sobre un usuario. En la primera página su nombre, en la segunda su número de teléfono y en la tercera su email. En la cuarta se mostrarán los datos recibidos
  - Prueba a acceder a la vez desde dos navegadores distintos para comprobar que se pueden gestionar dos sesiones diferentes a la vez

*Solución a un problema similar en:*

[http://www.mclibre.org/consultar/php/ejercicios/nivel\\_medio/sesiones/sesiones.html](http://www.mclibre.org/consultar/php/ejercicios/nivel_medio/sesiones/sesiones.html)

## Ejercicio: Autenticación de usuarios

---

- Crear un sistema de autenticación de usuarios que guarde los passwords codificados en una base de datos
  - Habrá que cifrar el password
  - Se recomienda usar https
  - ¿Qué información habrá que guardar en la base de datos?
  - Ejemplo:

[http://www.mclibre.org/consultar/php/ejercicios/nivel\\_medio/registro\\_usuarios.html](http://www.mclibre.org/consultar/php/ejercicios/nivel_medio/registro_usuarios.html)

- Con **LDAP**:

[http://docstore.mik.ua/oreilly/webprog/pcook/ch17\\_09.htm](http://docstore.mik.ua/oreilly/webprog/pcook/ch17_09.htm)

<http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=494>