Programación del Servidor

Desarrollo de Aplicaciones Web.

Departamento Informática y Sistemas.
Universidad de Murcia.

Curso 2014/15

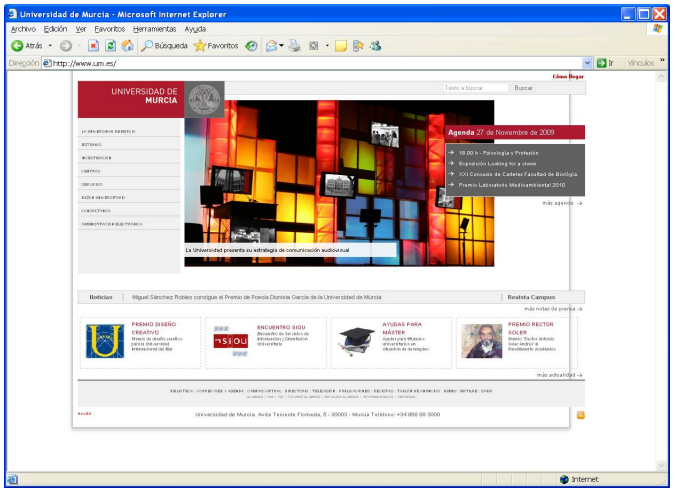
Contenido 1/2

- Concepto de aplicación web.
- Protocolo HTTP.
- Software de servidor.
- PHP básico:
  - Creación de una aplicación PHP.
  - Bloques de código PHP.
  - Variables.
  - Funciones.
  - Cadenas.
  - Operadores y control de flujo.

Contenido 2/2

- PHP básico (continúa):
  - Arrays.
  - Parámetros en páginas PHP.
  - Bases de datos MySQL.
  - Constantes y URLs.
  - Función include.
- Aplicaciones web de gestión.
- Sesión web.
- Cookies.
- Cabeceras HTTP.
- AJAX.

La Web



Navegador

Petición

http://www.um.es/index.html

HTTP

Respuesta

Servidor Web

HTML

Documento /index.html

Sistema Ficheros

Sitio Web frente Aplicación Web

- Aplicación web
  - "Sitio web donde las entradas del usuario (navegación y entrada de datos) afectan al **estado de un sistema de información**"

Protocolo HTTP

- Protocolo de comunicación entre navegador y servidor web.
- Modelo **Petición/Respuesta**
- Es un protocolo **sin estado**:
  - Cada ciclo petición/respuesta es independiente.
- **Tipos de peticiones**:
  - **GET**: petición de un recurso
  - **POST**: envío de datos a un recurso
  - HEAD: petición cabeceras de un recurso
  - ...

Petición HTTP

GET /index.html HTTP/1.1

Accept: image/gif, image/jpg,
application/msword, */*

Accept-Language: es

Accept-encoding: gzip

User-Agent: Mozilla/4.72

Host: dis.um.es

Connection: Keep-Alive

(línea en blanco)

CUERPO

PETICIÓN

CABECERAS

CUERPO

Respuesta HTTP

HTTP/1.1 200 OK
Date: Mon, 23 oct 2006 12:24:23 GMT
Connection: close
Server: Apache/1.3.24 (Linux)
Content-Length: 2032
Last-Modified: Sun, 22 oct 2006 10:03:11 GMT
Content-Type: text/html
Content-Language: es
Expires: Sun, 22 oct 2006 12:24:23 GMT

<html>
<head> ...

Código de Estado

Cabeceras

Cabecera de Entidad

Cuerpo

Software de Servidor

- **Servidor web**:
  - Aplicación que aloja los recursos web.
  - Sirve recursos solicitados con el **protocolo HTTP**.
  - Servidores más populares:
    - **Apache** (software libre)
    - IIS de Microsoft.
- **Base de datos**:
  - Aplicaciones empresariales: *Oracle*, *MS SQL Server*.
  - De propósito general: **MySQL**, PosgreSQL
- **Páginas de Servidor**:
  - **PHP.**
  - Otros: JSP, ASP, ASP.NET, etc.

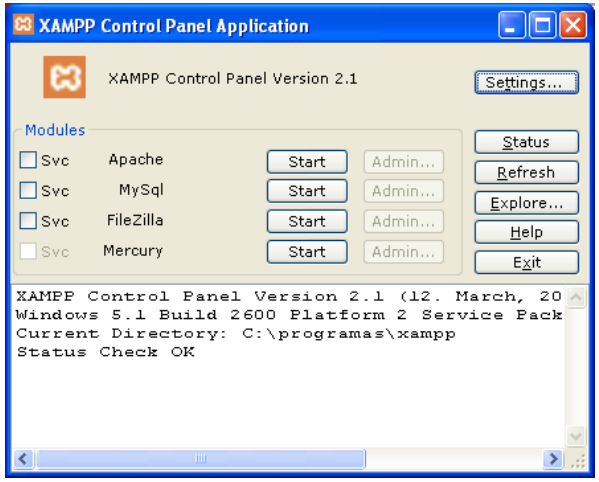
Software de Servidor

- Software adicional:
  - **Servidor FTP** para mantener los sitios web.
  - **Servidor de correo**.
  - …
- ➔ La instalación y configuración de todo el software es compleja

- Solución: **XAMPP**
  - Distribución que incluye: *Apache*, *MySQL*, *PHP*, servidores FTP y correo, etc.

Software de Servidor - Instalación

- **XAMPP**:
  - Instala todo el software en una sola carpeta.
  - Durante el proceso nos pregunta si queremos que los servidores se instalen como *servicios de Windows*.
    - Esta opción sólo interesa para un servidor en producción.
    - **NO instalaremos el software como servicios**.
  - Se administran los servidores a través de la aplicación ***XAMPP Control Panel***



Software de Servidor - Instalación

- Arrancamos el servidor *Apache* y *MySQL.*

- Probamos la correcta instalación escribiendo en el navegador las direcciones:
  - **Apache**: http://localhost/xampp/
  - **MySQL**: http://localhost/phpmyadmin/
- La primera dirección es la utilidad de **administración web de XAMPP.**
- La correcta instalación de **PHP** es comprobada accediendo a la herramienta PHP de administración de la base de datos: *PHPMyAdmin.*

Base de datos MySQL

- El sistema gestor de bases de datos ***MySQL*** **está incluido en *XAMPP.***
- Hay numerosas aplicaciones de **administración de MySQL**.
- La más popular es ***PHPMyAdmin***: aplicación web que funciona en PHP.
- Accedemos a *PHPMyAdmin* en la **dirección**: http://localhost/phpmyadmin/

PHPMyAdmin

- Desde la página de inicio podemos **crear una nueva base de datos:**
  - Crearemos una de ejemplo con nombre “ejemplo”.

- Para trabajar con la base de datos es conveniente **definir un nuevo usuario**: enlace “*privilegios*”
  - “*Agregar nuevo usuario*”
  - *Nombre y clave*: web/web
  - *Servidor*: **local** → las aplicaciones web y la base de datos residen en el mismo ordenador.
  - *Privilegios*: todos los de *datos* y *estructura*.
- **Reiniciamos el servidor** para que se establezca el usuario.

PHPMyAdmin

- **Accedemos a la base de datos** (menú izquierdo)
- Aunque se pueden crear las **tablas** con la aplicación, es conveniente definirlas **en SQL**: enlace “SQL”.
- **Inserción de datos** también a través del enlace “SQL”
- Podemos **exportar el contenido** de una BD en SQL: enlace “*Exportar*”
  - Seleccionamos las tablas.
  - Marcamos “Añada IF NOT EXISTS”
  - Marcamos “Enviar” para generar un archivo SQL.
- **Importar el contenido**:
  - Seleccionamos la base de datos y cargamos el archivo *.sql* en el enlace “SQL”.
- También podemos **exportar todas las bases de datos**.

Estructura de una aplicación web

- Una **aplicación web** consiste en una **carpeta en el servidor web** que contiene:
  - **Recursos web**: páginas web, hojas de estilo, código *JavaScript*, etc.
  - **Páginas de servidor** (PHP, JSP, ASP).

- La ubicación de las carpetas en un servidor web depende de la configuración.

- En **XAMPP**, las aplicaciones web se crean dentro de la **carpeta `htdocs`**.

PHP

- PHP es una tecnología para el desarrollo de páginas de servidor.
- Una **página de servidor** es una **plantilla** de página web:
  - Es una página web que contiene **bloques de código** (PHP) que generan dinámicamente código HTML cada vez que la página es solicitada.
  - En general, una página de servidor puede generar dinámicamente cualquier recurso textual, aunque suele emplearse para páginas web.

Bloques de código

- Bloques de código entre <?php y ?>

```
<html>
<head> <title>Ejemplo PHP</title> </head>

<body>
<p>Esto es contenido en HTML</p>

<?php

print "<p>Este contenido ha sido generado en PHP</p>";

?>
</body>
</html>
```

Variables

- Las variables y parámetros son declarados sin especificar el tipo (lenguaje tipado dinámicamente).
- No obstante, el lenguaje tiene **tipos**: `integer`, `double`, `boolean`, `string`, etc.
- Los **identificadores** de variables comienzan por **$**

```php
<?php

$nombre = "Juan";

print "<p>Valor de la variable: $nombre</p>";

?>
```

Funciones

```php
<?php

function suma($a, $b) {
        return $a + $b;
}

$resultado = suma(3, 2);
print "<p>SUMA: $resultado</p>";

?>
```

- Declaración similar a *JavaScript*: no se especifica el tipo de los parámetros ni el valor de retorno.

Funciones – variables globales

- Dentro de una función hay que declarar el uso de una variable global con **`global`**.

```php
<?php
$factor = 0.1;

function suma($a, $b) {
        global $factor;
        return ($a + $b) * $factor;
}

$resultado = suma(3, 2);

print "<p>SUMA: $resultado</p>";
?>
```

Cadenas

- La **declaración** de una cadena admite tres alternativas según su uso:
  - **Comillas simples**: sólo sirven para declarar cadenas de texto.
  - **Comillas dobles**: cadenas que **evalúan variables**. Dentro de la cadena, las comillas dobles se introducen con el carácter de escape: \”
  - **Bloque de texto** (*heredoc*): evalúa variables y evita tener que utilizar el carácter de escape para comillas dobles.

Cadenas

```php
<?php

$cadena = 'Esto es un ejemplo';
$cadena2 = "Variable \"cadena\": $cadena";

print <<<END
<p align="center">$cadena2</p>
END;

?>
```

Cadenas

- Para **fragmentos HTML** es recomendable declarar un bloque de texto (cadena *heredoc*), ya que en HTML las comillas dobles son habituales.
  - La declaración comienza por **<<<** y va seguida de una cadena delimitador y finaliza con el mismo delimitador.
  - El **delimitador** puede ser cualquier cadena que no aparezca en el texto.
  - El **cierre del bloque** debe estar situado al comienzo de una línea.

Cadenas – Funciones

- **Concatenación**: operador “.”
- **Longitud**: función **`strlen`**
- Obtener una **subcadena**: función **`substr`**
- **Búsqueda** de una subcadena: función **`strpos`**
- **Acceso a los caracteres** como un array:
  - Ejemplo: `$cadena[3]`

Cadenas – Funciones

```php
<?php
$simple = 'Esto es un ejemplo';
$cadena = "Variable" . "\"cadena\": " . $simple;

// Longitud 36
print "Longitud cadena: " . strlen($cadena) . "<br />";
// Subcadena hasta el final: "un ejemplo"
print "Subcadena : " . substr($simple, 8) . "<br />";
// Subcadena indicando un tamaño: "un"
print "Subcadena : " . substr($simple, 8, 2) . "<br />";
// Posición 8
print "Posición : " . strpos($simple, "un") . "<br />";
// Acceso al carácter 8: "u"
print "Carácter 8 : " . $simple[8] . "<br />";
?>
```

Operadores y control de flujo

- PHP incluye los mismos **operadores** que C y añade:
  - Operador de identidad (`===`)
  - Operadores lógicos: `and` y `or`.

- **Estructuras de control** de C: `if`, `switch`, `while`, `for`, etc. Características destacadas:
  - `switch` permite evaluar cadenas.
  - Recorrido de arrays con **`foreach`**.

- **Comentarios** igual que en C.

Arrays

- **Construcción**: función **`array`**
- **Acceso a los elementos** por índice a partir de 0
- **Tamaño** del array: función **`count`**
- **Añadir elementos por el final** asignando a la variable acabada en `[]`
- **Recorrido** utilizando la construcción **`foreach`**.

Arrays

```php
<?php
// Declaración
$colores = array ("rojo", "verde", "blanco");
// Acceso por índice
print $colores[0] . "<br />";
// Inserción por el final
$colores[] = "azul";
// Tamaño 4
print count($colores) . "<br />";
// Recorrido
foreach ($colores as $color)
        print "<p>Color: $color </p>";
?>
```

Arrays

- **Tablas asociativas**: establecer como clave del array una cadena
  - Ejemplo: `$tabla["murcia"] = "30";`

- **Arrays multidimensionales**: añadir como elemento de un array otro array.

- **Depuración**: función `print_r` muestra todo el contenido de un array.

Arrays

```php
<?php
// Declaración tabla asociativa
$tabla = array("barcelona" => "08", "madrid" => "28");

$tabla["murcia"] = "30";

print $tabla["barcelona"] . "<br />";

// Array multidimensional
$tabla["colores"] = $colores;

print $tabla["colores"][0] . "<br />";

// Muestra el contenido
print_r($tabla);
?>
```

Arrays

- Recorrido de una tabla asociativa:

```php
<?php

$tabla = array("barcelona" => "08", "madrid" => "28");

foreach ($tabla as $ciudad => $codigo) {

        print "<p>Ciudad $ciudad y código $codigo</p>";
}

?>
```

Arrays y cadenas

- Conversión **cadena → array**: función **`explode`**
- Conversión **array → cadena**: función **`join`**

```php
<?php
$cadena = "Uno, dos, tres";

$valores = explode(", ", $cadena);

print $valores[0]; // Muestra "Uno"

$cadena2 = join(" - ", $valores);

print $cadena2; // Muestra "Uno - dos - tres"
?>
```

Variables inicializadas

- La función **isset** se utiliza para consultar si una variable o elemento de un array ha sido inicializado.
- Eliminar un elemento de un array: función **unset**

```php
<?php

$tabla = array("barcelona" => "08", "madrid" => "28");

isset($tabla); // true
isset($tabla["barcelona"]); // true
isset($tabla["murcia"]); // false

unset($tabla["barcelona"]);
isset($tabla["barcelona"]); // false
?>
```

Conversión de tipos

- Se puede realizar conversión de tipos realizando un **casting**.
- Los operadores aritméticos convierten cadenas en números automáticamente.
- El operador de concatenación de cadenas convierte números en cadenas automáticamente.

```php
<?php

$cadena = "100";

$entero = (integer) $cadena;

?>
```

Parámetros en páginas PHP

- La **ejecución** de una página PHP puede estar **parametrizada**.

- **Ejemplo**: página PHP que reciba como parámetro el código de un registro de la base de datos y que visualice sus datos.

- La recepción de parámetros es clave para el **procesamiento de formularios en el servidor**:
  - Los campos de un formulario se pasan como parámetros a una página PHP.

Parámetros en páginas PHP

- Hay dos modos de **envío de parámetros**:
  - Enviados en el **cuerpo de la petición** HTTP utilizando el método **POST** (enviados por formularios).
  - **Establecidos en la URL** que invoca a la página PHP:
    - **Ejemplo:** **`pagina.php?codigo=1&nombre=Juan`**
    - **Parámetros:** `codigo` con valor "1" y `nombre` con valor "Juan"

Parámetros en páginas PHP

- Los parámetros son accesibles a través del **array** `$_REQUEST`
- **Ejemplo:** acceso al parámetro “codigo”:
  `$_REQUEST["codigo"]`
- Los parámetros son **cadenas de texto**.
- **Parámetros multivaluados**:
  - En lugar de una cadena se obtiene un array de cadenas.
  - El nombre del **control del formulario** debe acabar en `[]`.
  - Ejemplo: `aficiones[]`
  - Al procesarlo en la página PHP se obtiene sin utilizar `[]`

Bases de datos MySQL

- **Establecer una conexión** a la base de datos:

  `mysql_connect`

  - Parámetros: URL del servidor, usuario y clave
- Seleccionar la **base de datos de trabajo** en el servidor:

  `mysql_select_db`

  - Parámetro: nombre de la base de datos.
- **Realizar una consulta**: `mysql_query`
  - Parámetro: cadena con la consulta en SQL.
  - Retorna un **identificador de consulta**. Utilizando el identificador podemos recuperar los datos.

Bases de datos MySQL

- Determinar el **número de registros de la consulta**:

  `mysql_num_rows`

  - Parámetro: identificador de la consulta

- Determinar el número de **registros afectados por una consulta de actualización**:

  `mysql_affected_rows`

  - Útil para consultas de actualización como las inserciones.

Bases de datos MySQL

- Obtener los **resultados de la consulta como objetos**
  `mysql_fetch_object`
  - Retorna un **objeto** que tiene como propiedades las columnas de la consulta.
  - Ejemplo: `$libro->titulo`
- Obtener los **resultados de la consulta como arrays**:
  `mysql_fetch_array`
  - Retorna un array cuyos valores son las columnas de la consulta. Son accesibles por nombre.
  - Ejemplo: `$libro["titulo"]`

- **Cerrar una conexión**: **`mysql_close`**

Bases de datos MySQL

```php
<?php
mysql_connect("localhost", "web", "web");
mysql_select_db("libros");

$consulta = mysql_query("SELECT autor, titulo FROM Libro");

if (mysql_num_rows($consulta) == 0)
   print "<p>No hay resultados</p>";
else {
   while ($libro = mysql_fetch_object($consulta)) {

       print "<p>\"$libro->titulo\" de $libro->autor</p>";
   }
}
mysql_close();
?>
```

Bases de datos MySQL

```php
<?php

mysql_connect("localhost", "web", "web");

mysql_select_db("libros");

mysql_query("UPDATE Libro "
        . " SET autor = 'Juan' WHERE codigo = 1");

if (mysql_affected_rows() == 1)
        print "<p>Un registro actualizado</p>";

mysql_close();

?>
```

Bases de datos MySQL

- Las llamadas a las funciones de MySQL pueden producir **errores**.
- La función **mysql_error** devuelve una cadena con el error.

```
<?php
mysql_connect("localhost", "web", "web");
mysql_select_db("libros");
$consulta = mysql_query("Consulta error");

if (mysql_error())
        print "<p>Error: " . mysql_error() . "</p>";

mysql_close();
?>
```

Bases de datos MySQL

- Evitar código malicioso del usuario en consultas SQL: función **mysql_escape_string**

```
<?php
$codigo = mysql_escape_string($_REQUEST["codigo"]);

$consulta = "SELECT * FROM Libro WHERE codigo =
$codigo";
?>
```

- Limitar los resultados de una consulta SELECT con LIMIT:
  → útil para **listados paginados**
  - "SELECT autor, titulo FROM Libro **LIMIT 0, 10**"
  - LIMIT índice del primer registro, número de registros

Constantes y URLs

- Declaración de **constantes**: función `define`

```php
<?php
        define ('BD', 'localhost');
        print BD;
?>
```

- Codificación de parámetros en **URLs**:

```php
<?php
$cadena = 'hola qué tal';
$codificada = urlencode($cadena); // "hola+qu%E9+tal"

$url = "http://dis.um.es/saludo.php?msg=$codificada";
?>
```

Función include

- La función **include** permite incluir el resultado del procesamiento de una página PHP o HTML dentro de otra página PHP.

```
<div id="menu">
        <?php include ("menu.php"); ?>
</div>
<div id="contenido">
        <?php include ("contenido.php"); ?>
</div>
<div id="pie">
        <?php include ("pie.html"); ?>
</div>
```

- Muy útil en el desarrollo web.

Función include

- No es posible establecer parámetros en el nombre de los ficheros PHP con la función `include`.
- Para pasar un parámetro a la página PHP hay que utilizar el array `$_REQUEST`.

```php
<?php

$_REQUEST["codigo"] = "1";
include ("contenido.php");

?>
```

Aplicaciones web de gestión

- Una aplicación web de gestión proporciona acceso a un **sistema de información** almacenado en una base de datos.

- Los elementos que constituyen la aplicación son:
  - Formularios para el **alta** de los datos.
  - Formularios de **consulta**.
  - **Listados**.
  - Páginas para la **edición** de datos.
  - Operaciones para la **eliminación** de datos.

Procesamiento de formularios

- Los campos de un formulario son enviados a una **página PHP** para su procesamiento.
- Habitualmente, los campos de un formulario son utilizados para el **acceso a una base de datos**: consulta, inserción, edición, etc.
- La página PHP que procesa los datos de un formulario debe:
  - Procesar los datos.
  - Mostrar el resultado del procesamiento.

Listados

- Los listados son construidos a partir de una **consulta a la base de datos**.
- Un listado puede ser paginado (varias páginas de resultados) y ordenado según algún campo.
- **Listado paginado**:
  - Hay que establecer un **parámetro** en la página PHP para el **índice del primer registro de la página**.
  - Hay que generar los **índices de todas las páginas** y los enlaces “Anterior” y “Siguiente”.
- **Listado ordenado**:
  - Se añaden dos parámetros: uno que indique el **campo del orden** y otro para el **criterio** (ascendente, descendente).

Edición de registros

- La edición de información de la base de datos se realiza a través de **dos páginas PHP**:
  - Página para la **generación de un formulario** para la edición de los datos con los campos rellenos.
  - Página para el **procesamiento de los datos actualizados** del formulario.
- **Inicialización de los campos del formulario**:
  - Depende de su tipo: cajas de texto, botones de radio, listas, etc.
  - **Ejemplo**: campos de tipo `input` se establece el valor en el atributo `value`.
- Las dos páginas están ligadas a través de un **campo oculto** establecido con el **código de registro**.

Eliminación de registros

- **Página PHP** que tiene como objetivo **eliminar un registro de la base de datos**.
- Tiene como **parámetro el código** del registro a eliminar.
- Habitualmente esta página es **enlazada en los listados** de registros.
- Si la operación se completa con éxito suele dar como resultado el listado de registros actualizado.

Sesión web

- **Problema**:
  - **HTTP es un protocolo sin estado**: cada petición de un recurso al servidor web es independiente.
  - Sin embargo, las aplicaciones web necesitan **conocer** qué **usuario** (navegador) hace una petición y **asociar información** al usuario.
  - **Ejemplos**: confeccionar un carro de la compra, recordar el identificador de un usuario.

- **Solución**:
  - Los servidores web implementan el concepto de **sesión web**.

Sesión web y cookies

- Tradicionalmente, se ha podido controlar el estado de la navegación utilizando ***cookies***.
- Una cookie es un **fragmento de texto** que una aplicación web envía a un navegador con el compromiso de que lo devuelva en futuras peticiones de recursos a la aplicación.

- Hay dos **tipos de cookies**:
  - Las que tienen un tiempo de vida definido: horas o días.
  - Las que están vinculadas a la ejecución del navegador (**cookies de sesión**).

Sesión web

- Las tecnologías de soporte web como PHP, JSP o ASP utilizan **cookies de sesión para crear el concepto de sesión web**.

- Para cada nuevo navegador (usuario) que accede a una aplicación web, se crea una cookie de sesión con un **identificador único** que se envía al navegador.

- Cada vez que el navegador solicita una página devuelve el identificador en la cookie al servidor web.

Sesión web

- El servidor web maneja una **estructura de datos** que permite a las páginas de servidor almacenar información que se mantiene durante la sesión de navegación.

- PHP ofrece el array `$_SESSION` para consultar y almacenar la información de la sesión.

Sesión web

- **Activación** de la sesión web para almacenar y acceder a la información:
  - Se sitúa el siguiente fragmento de código justo al comienzo de la página

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

- **Eliminación** de la sesión web:

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

Sesión web

- Ejemplo de uso de la sesión web: registra en la sesión el parámetro “codigo”

ejemplo1.php

```php
<?php
$codigo = $_REQUEST["codigo"];
// Registro
$_SESSION["codigo"] = $codigo;
?>
```

ejemplo2.php

```php
<?php
// Consulta
print $_SESSION["codigo"];
// Eliminación
unset($_SESSION["codigo"]);
?>
```

Cookies

- La información almacenada en la sesión se pierde cuando el usuario cierra el navegador.
- Para **almacenar información que persista la ejecución del navegador** hay que usar cookies.
- La función `setcookie` permite enviar una cookie al navegador.
- El **envío** de cookies debe realizarse **al comienzo de la página**.
- Si no se indica el **tiempo de vida**, se envía una **cookie de sesión**.

Cookies

- **Ejemplo**:
  - Crea dos cookies con el valor de un parámetro.
  - El código debe situarse al comienzo de la página.

```php
<?php
$codigo = $_REQUEST["codigo"];
// Envía una cookie de sesión
setcookie("codigo", $codigo);
// Envía una cookie que dura una hora
// El tiempo de vida se expresa en segundos
setcookie("codigo2", $codigo, time() + 3600);
?>
```

Cookies

- Acceso a las cookies en cualquier página de la aplicación utilizando el array **$_COOKIE**:

```php
<?php

print "Cookie codigo: " . $_COOKIE["codigo"];

?>
```

Cabeceras HTTP

- Es posible enviar cabeceras del protocolo HTTP con la función **`header`**.
- Utilidad: **cabecera `Location`**
  - Si una página PHP envía la cabecera **`Location`** al navegador, éste interpreta que debe solicitar la página (HTML o PHP) indicada en la cabecera.
- El envío de cabeceras debe realizarse al comienzo de una página PHP.

```php
<?php
        header ("Location: http://dis.um.es");
?>
```

AJAX

- **Motivación**:
  - En una página de alta, comprobar si existe el identificador de usuario antes de enviar los datos al servidor.
  - En general, **obtener datos del servidor sin necesidad de recargar la página**.

- AJAX es la tecnología que permite el acceso al servidor de forma **asíncrona** en respuesta a eventos de la programación dinámica en HTML.

- Supone un **punto de inflexión en el desarrollo de aplicaciones web**.

AJAX

- **AJAX**: ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*

- **Inicialmente** AJAX representaba el soporte de los navegadores para acceder, de modo asíncrono, a datos en el servidor utilizando *JavaScript*.

- **Actualmente** *también* se entiende por AJAX las **bibliotecas de componentes gráficos** programados con HTML Dinámico.

AJAX

- Las peticiones AJAX utilizan un objeto **XMLHttpRequest**.
- Permite realizar con *JavaScript* una petición HTTP al servidor y esperar a que se reciban los datos.
- La clave de la programación con AJAX es la **petición asíncrona de datos al servidor**:
  - Se solicitan datos al servidor, pero el navegador no queda bloqueado a la espera.
  - Cuando llegan los datos, una función *JavaScript* se encarga de procesarlos.

AJAX

- La **función** que recibe los datos del servidor se conoce como **“callback”.**

- La función *callback* es llamada cada vez que cambia el estado de la petición HTTP:
  - Cuando el **estado de la petición** es *completo* (4) y el **código** de respuesta de **HTTP** es *correcto* (200) se pueden obtener los datos de la petición en texto normal o estructurado en un árbol DOM.

AJAX – Petición HTTP

- Los navegadores antiguos no soportan los objetos `XMLHttpRequest`.
- **Solución**:

```
if (typeof XMLHttpRequest != "undefined") {
  req = new XMLHttpRequest();
} else if (window.ActiveXObject) {
  req = new ActiveXObject("Microsoft.XMLHTTP");
}
```

- La **variable** "req" es **global**. Almacena el objeto de la petición y permite a la función callback obtener los datos.

AJAX – Petición HTTP

- El segundo paso es **componer la URL del recurso** que procesa la petición. En nuestro contexto, una **página PHP**.
- Habitualmente se pasan **parámetros** a la página PHP en la URL.
- Sin embargo, hay que tener en cuenta que una URL no admite cualquier carácter, por lo que hay que **codificar los parámetros** con la función **encodeURIComponent**

```
var url = "validaUsuario.php?id="
          + encodeURIComponent("García");
```

AJAX – Petición HTTP

- El siguiente paso es registrar la URL en el objeto `XMLHttpRequest`:

```
req.open("GET", url, true);
```

- Los **parámetros** son:
  - El **tipo de petición HTTP**: habitualmente GET.
  - La **URL del recurso** que procesa la petición.
  - El **modo de petición**, **asíncrono** (true) o *síncrono* (false).
- Si el modo es asíncrono, se registra la **función** de ***callback*** que es llamada cuando cambie el estado de la petición:

```
req.onreadystatechange = callback;
```

AJAX – Petición HTTP

- Por último, se **realiza la petición**:

  ```
  req.send(null);
  ```

  - El parámetro es el *contenido* de la petición.
  - Habitualmente es nulo debido a que los datos se establecen en la URL.

- **Importante**: si la llamada es asíncrona, al invocar `send` **el código no espera**. En general, esto es fundamental para que el navegador no se quede bloqueado por la latencia de la comunicación de red.

AJAX – Respuesta

- El objeto `XMLHttpRequest` controla todo el proceso de petición.
- Conforme **cambia el estado**, notifica a la función de ***callback***.
- La **petición** es **completa** cuando se llega al estado 4.
- Una vez completa, el estado de la **respuesta HTTP** debe ser 200 (**OK**) para que el proceso haya finalizado correctamente.
- En general, el patrón de la **función *callback*** es:

```
if (req.readyState == 4 &&
      req.status == 200) {
         … → procesar la repuesta
```

AJAX – Respuesta

- El objeto `XMLHttpRequest` mantiene el **resultado** de la respuesta de dos modos:
  - Como **texto**, habitualmente formateado en XHTML.

```
req.responseText;
```

  - Como **árbol DOM**:

```
req.responseXML;
```

- El **procesamiento como texto** es el más cómodo para asignar el contenido de la respuesta a un nodo de la página utilizando la propiedad **innerHTML**.