

TEMA 3. PROGRAMACIÓN DE APLICACIONES EN EL SERVIDOR

Curso 2019-2020

Programación Web

Bibliografía

- R. Nixon, «PHP, MySQL, & Javascript», O'Reilly, 2009
- J. N. Robbins, «Learning Web Design», O'Reilly, 4th Edition, 2012
- B.P. Hogan, et al. «Web Development Recipes», Pragmatic Programmers, 2012
- W. Steintmetz, B. Ward, «Wicked Cool PHP», No Starch Press, 2008

Contenido

- Aplicaciones y sitios web
- Lenguaje PHP
- Conexión con BD
- Autenticación, Cookies
- PEAR
- PHP Avanzado

APLICACIONES Y SITIOS WEB

Aplicación web

- Documento diseñado para su difusión a través de la web, accedido mediante navegadores
- Escrito en HTML o XHTML
- Contenido estático. Prefijado y almacenado en disco
- Contenido dinámico: se genera en el momento de la consulta
- **Aplicación web**: documento con capacidad interactiva
- Ingeniería del software adaptada al ciclo de vida de la web

Sitio web

- Conjunto de páginas web relacionadas y albergadas bajo un mismo dominio de Internet. Suelen compartir esquema y autoridad
- Espacio documental estructurado habitualmente dedicado a un tema específico o propósito concreto.
- Las páginas web del sitio son accesibles a través de un URL raíz, denominado “portada”.
- Organización jerárquica (árbol o DAG)

Elementos de una aplicación web

- Interfaz
- Servidor web
- Base de datos
- Lenguaje de programación
- Despliegue:
 - Servidor:
 - Hardware (*Bare-metal*)
 - Plataforma cloud (IaaS)
 - Contenedor
 - Plataforma Cloud Computing (SaaS)
 - Plataformas Fog y Edge computing

Terminología actual

Para desarrollo web

- **Back-end developer:** ámbito del servidor
- **Front-end developer:** ámbito del cliente
- **Full-stack developer:** Incluye los dos anteriores
- Relacionadas:
- **Devops:** Integra tareas de desarrollador y operador de sistemas
 - Promociona la automatización y motorización en todos los pasos de desarrollo (codificación, integración, pruebas, despliegue y administración de infraestructura).
- **Continuous-integration:** Hacer integraciones (compilación y ejecución de pruebas) automáticas de un proyecto lo más frecuentemente posible para acelerar la detección de fallos.

Aplicaciones en el ámbito del servidor

- Caracterizado porque todo el código permanece en el servidor:
 - El programa se ejecuta en el servidor. Puede invocar ejecuciones en otros servidores (RPC, SOA, ...)
 - Los programas pueden ser compilados o en código fuente, pero en ninguna de las dos versiones sale del servidor.

Tecnologías para desarrollo en el ámbito del servidor

- Common Gateway Interface:
 - programas binarios en el servidor
 - Interfaz de comunicación entre servidor web y aplicación
 - Surgieron variantes (FastCGI)
 - Aquejado de frecuentes problemas de seguridad
- **PHP**
- ASP
- Python
- Java (JSP)
- Ruby
- Node.js
- Scala
- Extensiones de lenguajes: R, Clojure, ...

Lenguajes más populares

Most popular server-side programming languages

© W3Techs.com	usage	change since 1 February 2019
1. PHP	79.0%	+0.1%
2. ASP.NET	11.5%	-0.1%
3. Java	4.0%	
4. Ruby	2.5%	+0.1%
5. static files	2.1%	

percentages of sites

<https://w3techs.com>

[80% of Web powered by PHP](#)

Roadmap to become a web developer in 2019

- [Developer Roadmap](#)

LENGUAJE PHP

Lenguaje PHP



- Lenguaje de programación de aplicaciones web en el servidor de propósito general (*PHP Hypertext Pre-Processor*)
- Diseñado para la generación dinámica de contenidos
- Código incrustado en documentos HTML e interpretado por un módulo del servidor web
- Sintaxis cercana a C y Perl
- Fácil conectividad con SGBD
- Interpretado

Script sencillo

script1.php

```
<?php  
    echo "Hola. Hoy es " . date("l") . ". ";  
?>
```

Hola. Hoy es jueves

Elementos del lenguaje PHP

- Sentencias; Comentarios
- Variables
- Operadores
- Tipos de datos; *arrays*
- Concatenadores
- Funciones
- Expresiones y control de flujo
- PDO
- Formularios

Sentencias

- El intérprete se activa para bloques delimitados por:
 <?php ...?>
- Todas las sentencias terminan con «;»

Variables en PHP

```
<?php                                     script2.php
    $contador = 1;
    $nombre = "Luis";
    $vector = array("una", "cadena", "larga");
?>
```

No hay **verificación** de tipos. Una variable es un espacio para almacenar datos

Arrays

- Tipo de dato estructurado (no necesariamente homogéneo)
- Se crean con «array()»
- Los componentes se acceden mediante índices, comenzando en 0
- Pueden ser multidimensionales:

```
$matriz = array(array(1, 2, 3),  
                array(4, 5, 6));
```

Funciones

script3.php

```
<?php
function suma($a, $b)
{
    return $a + $b;
}
$a = 3;
$b = 4;
echo "La suma de $a y $b es " . suma($a, $b);
?>
```

Extensa **biblioteca** de funciones en PHP o módulos: [PEAR](#)



Estructuración del código

- **Diseño modular:** `varios ficheros .php`
- **Sentencias:**
 - `include` (aviso)
 - `require` (Error fatal)
 - `include_once` (garantiza la inclusión una sola vez)
 - `require_once`

Entradas y salidas

No hay acceso directo a dispositivos E/S, como en lenguajes de propósito general

Entradas

- Parámetros recibidos a través de peticiones HTTP (URL): variables
- Datos en formularios
- Archivos locales

Salidas

- Código HTML que se inserta en la página que devuelve el servidor como respuesta a la petición del cliente

ARRAYS

Arrays

- Contenedores heterogéneos
- Contenido accesible mediante índices numéricos o contenedor asociativo
- Se pueden usar iteradores para recorrerlos

Arrays indizados con enteros

- Índices a partir de **0**
- Se pueden crear con el constructor (array)
- Se pueden añadir elementos:

```
$impresora[] = "laser";
```

```
$impresora[] = "de inyección de tinta";
```

```
$impresora[] = "matricial";
```

```
$impresora[] = "de margarita";
```

Mostrar contenido

script4.php

```
<? php
    print_r($impresora);

    for ($j = 0; $j < 4; ++$j)
        echo "$j: $impresora[$j]<br>";
?>
```

Arrays asociativos

- Indizados mediante clave:

script5.php

```
<? php
$fruta['naranja'] = "navel";
$fruta['melon'] = "piel de sapo";
$fruta['manzana'] = "fuji";

echo $fruta['melon'];
?>
```

Iterando sobre arrays

script6.php

- Con la estructura «for each»:

```
<? php
$fruta = array("naranja", "melon",
"manzana");

foreach ($fruta as $pieza)
    echo "$pieza<br>";
?>
```

Iterando sobre arrays asociativos

script7.php

```
<? php  
$fruta = array ("naranja" => "navel",  
                "melon" => "piel de sapo",  
                "manzana" => "fuji");  
  
foreach ($fruta as $pieza => $variedad)  
    echo "$pieza: $variedad<br>";  
?>
```

FORMULARIOS

Formularios

- Principal medio de interacción con el usuario en PHP
- Proceso:
 - Creación del formulario
 - Envío de datos al servidor
 - Interpretación y filtrado en el servidor
 - Acciones asociadas al formulario
 - Generación de contenido

Construcción de formularios

- Encerrados entre `<form> ... </form>`
- Tipo de envío: GET o POST
- Campos de entrada
- URL de envío

Ejemplo de formulario

script8.html

```
<html>
<head>
  <title>Formulario de ejemplo</title>
</head>
<body>
  <form method="post" action="formtest.php">
    Introduzca su nombre:
    <input type="text" name="nombre" />
    <input type="submit" />
  </form>
</body>
</html>
```

formtest.php

```
<?php
if (isset($_POST['nombre'])) $nombre = $_POST['nombre'];
else $nombre = "(No introducido)";

<html>
<head>
  <title>Formulario de ejemplo</title>
</head>
<body>
Tu nombre es: $nombre<br/>
<form method="post" action="formtest.php">
  Introduzca su nombre:
  <input type="text" name="nombre" />
  <input type="submit" />
</form>
</body>
</html>
?>
```

Componente «form»

- Atributo «action»: script para procesar el formulario (php, js, python, perl, ...)
- Atributo «method»:
 - POST: El navegador envía una petición independiente con la información. **Sólo la ve** el servidor
 - GET: La información enviada se incluye en el URL usando los caracteres ? y %. El envío es **público**.

Asignación a variables

- Antes \$_POST y \$_GET se asignaban a variables PHP
- Problema de seguridad por lo que se deshabilita el uso de «register_global»

Valores por defecto

```
<form method="post" action="calc.php">
<pre>
Loan Amount <input type="text"
name="principle" />
Monthly Repayment <input type="text"
name="monthly" />
Number of Years <input type="text"
name="years" value="25" />
Interest Rate <input type="text" name="rate"
value="6" />
</pre>
```

Elementos en un formulario

- Cajas de texto (*Text boxes*)
- Cajas de marca (*Checkboxes*)
- Botones de radio (*Radio buttons*)
- Campos ocultos
- Etiquetas

Campos de texto

- **Atributos:** name, maxlength, size, value
- **text:** una sola línea de texto

```
<label>City <input type="text" name="city"
id="form-city" value="Your Hometown"
maxlength="50"></label>
```
- **textarea:**

```
<textarea name="comment" rows="4" cols="45"
placeholder="Leave us a comment."></textarea>
```
- **password:**

```
<input type="password" name="pswd"
maxlength="8" id="form-pswd">
```

Campos de texto (HTML5)

- `<input type="search">`
- `<input type="email">`
- `<input type="tel">`
- `<input type="url">`

Campos de fecha y hora (HTML5)

- `<input type="date">`
- `<input type="time">`
- `<input type="datetime">`
- `<input type="datetime-local">`
- `<input type="month">`
- `<input type="week">`

Campos numéricos (HTML5)

- `<input type="number">`
- `<input type="range">`

Botones de envío y reinicio

- `<input type="submit">`
- `<input type="reset">`

Botones de radio

- `<p>¿Cual es tu edad?</p>`
- ``
- `<input type="radio" name="age" value="Menor de 24" checked> under 24`
- `<input type="radio" name="age" value="25-34"> 25 to 34`
- `<input type="radio" name="age" value="35-44"> 35 to 44`
- `<input type="radio" name="age" value="Mayor de 45"> 45+`
- ``

Botones de marcado

```
<p>¿Qué tipo de música prefieres?</p>
<ul>
<li><input type="checkbox" name="genero"
value="punk" checked> Pop Rock</li>
<li><input type="checkbox" name="genero"
value="indie" checked> Rock Folclórico</li>
<li><input type="checkbox" name="genero"
value="hiphop"> Rap</li>
<li><input type="checkbox" name="genero"
value="rockabilly"> Rock</li>
</ul>
```

Menús

```
<p>¿Cuál es tu banda favorita de los 80?  
<select name="Favorito80">  
<option>The Cure</option>  
<option>Supertramp</option>  
<option>Tears for Fears</option>  
<option>Dire Straits</option>  
<option value="EBTG">Everything but the  
Girl</option>  
<option>The Queen</option>  
</select>
```

Selección de archivos

```
<form action="/client.php" method="POST"
enctype="multipart/form-data">
<label>Envía una foto como icono
<em>(optional)</em><br>
<input type="file" name="photo" size="28"><label>
</form>
```

Campo oculto

- Enviar información que no proporciona el usuario directamente:

```
<input type="hidden" name="success-link"  
value="http://www.example.com/  
littlechair_thankyou.html">
```

- Ejemplo de uso: crawler-trap

Estilo de codificación

- Es esencial para entender y mantener el código
- Elementos importantes:
 - Indentación
 - Longitud de las líneas de código
 - Uso de espacios en blanco
 - Llaves y sentencias de control
 - Elección de identificadores
 - Documentación

ACCESO A BASES DE DATOS

Bases de Datos

- Muchas aplicaciones web se apoyan en una BD para datos
- El diseño e implementación de la BD es una **tarea esencial** en estas aplicaciones
- Manejo de las BD: operaciones CRUD
- Consultas

Recordando conceptos

- Esquema
- Base de datos
- Tabla
- Fila
- Columna
- Consultas

Acceso a bases de datos

- Uso de BD para almacenamiento de datos
- Diseño e implementación adecuadas de las BDs
- SGBD: relacionales, jerárquicos o NoSQL
- Lenguaje de consulta SQL
- SGBD *open-source*:
 - MySQL (MaríaDB)
 - PostgreSQL
 - ...

MySQL



- SGBD relacional, multihebra y multiusuario
- Soporta integridad referencial, transacciones, replicación
- Simple, ágil, versátil
- Multiplataforma: Linux, Windows, Mac OS X, ...
- Lenguajes de programación: PHP, C, C++, Pascal Delphi, Python, Java, Perl, Eiffel, Smalltalk, Lisp, ...
- La más utilizada para aplicaciones web
- Distintos motores de almacenamiento e indización

Instalación y configuración de MySQL

- Elemento esencial de sistemas LAMP y WAMP
- Instalable desde código fuente y en distintos paquetes (.exe, .rpm, .deb, ...)
- Componentes:
 - Servidor
 - Herramientas de cliente
 - Conectores

Paquetes en Fedora

- mysql
- mysql-server
- mysql-libs
- php-mysql
- mysql-test
- mysql-connector-java
- mysql-connector-odbc
- mysql-utilities

Configuración, directorios y binarios

- `/etc/my.cnf`
 - `/var/lib/mysql`
 - `/var/run/mysqld`
 - `/var/log/mysqld.log`
-
- `mysql`
 - `mysqladmin`
 - `mysqldump`

Acceso a MySQL

- Por línea de órdenes: mysql
- GUI: phpMyAdmin, MySQL Workbench
- A través de aplicaciones:
 - Conectores: ODBC, JDBC
 - Java
 - Python
 - Perl
 - PHP

Usuarios

- DBA (root)
- Usuarios: CREATE USER
- Privilegios:

```
GRANT ALL PRIVILEGES ON DB.* TO  
'usuario'@'localhost' IDENTIFIED BY 'passwd';
```

```
update user set password=PASSWORD("clave-secreta ")  
where USER= "usuario";
```

Tipos de datos

Categoría	Tipos
Numérico	INTEGER, INT, SMALLINT, TINYINT, MEDIMUINT, FLOAT, DOUBLE, BIT
Cadena de caracteres	CHAR, VARCHAR, VARBINARY
Fecha y hora	DATE, DATETIME, TIMESTAMP, TIME, YEAR
Texto	TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT
Binarios	TINYBLOG, BLOB, MEDIUMBLOB, LONGBLOB

Índices

- Permiten localizar y devolver registros de forma sencilla y rápida
- Implementación interna basada en B-trees y derivados
- PRIMARY KEY ... UNIQUE INDEX ...
- ALTER TABLE personas ADD INDEX (apellido);

Consulta SELECT

- Sentencia básica para recuperar información: SELECT
- `SELECT * FROM alumnos;`
- `SELECT nombre, apellido FROM alumnos WHERE apellido LIKE 'B%';`
- Cláusulas:
 - `GROUP BY, HAVING, ORDER BY, LIMIT`

INSERT

- Inserción de datos en tablas
- `INSERT INTO alumnos (nombre, apellido)
VALUES ('Luis', 'Escobar');`

UPDATE

- Actualización de información
- `UPDATE alumnos SET nombre = 'Juan' WHERE dni= '11223344';`

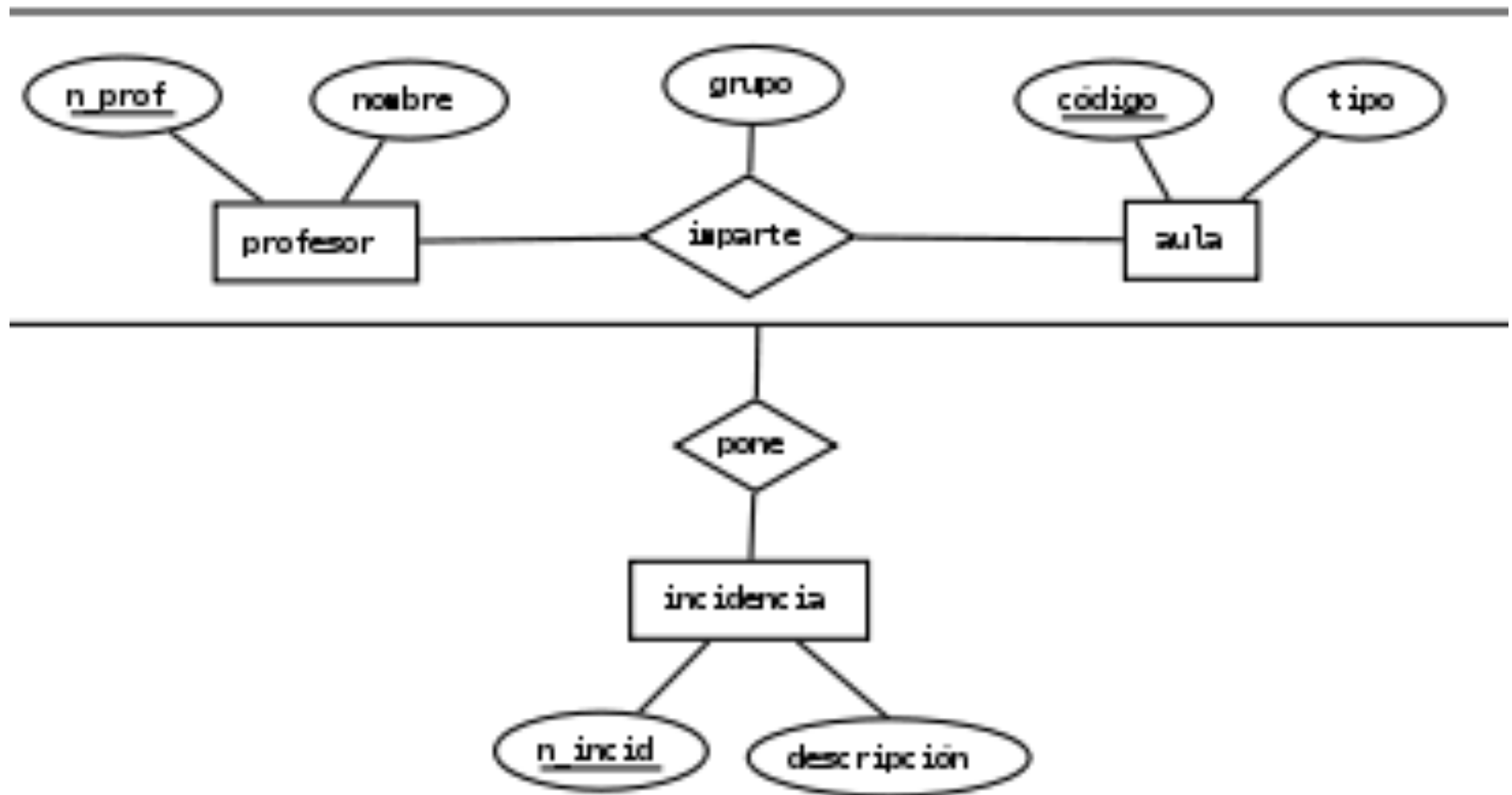
DELETE

- Eliminar filas de una tabla
- `DELETE FROM alumnos WHERE dni = '11223344';`

Diseño relacional

- Diagramas Entidad-Relación
- Esquemas
- Llaves primarias
- Llaves externas
- Reglas de integridad
- Formas normales

Diagramas entidad-relación



Relaciones

- Las BDs almacenan datos factuales y relaciones entre éstos
- Relaciones 1:1
- Relaciones 1:n
- Relaciones n:m

Transacciones

- Transacción: garantizar la ejecución íntegra y ordenada de una secuencia de consultas
- P.ej.: transferencia de fondos
- Soporte basado en el motor de almacenamiento (InnoDB)

```
BEGIN
```

```
UPDATE cuenta SET saldo = saldo+25 WHERE ID = 389;
```

```
COMMIT;
```

- Cancelación: ROLLBACK;

Copias de seguridad

- Volcado de datos en texto (también CSV)
 - `mysqldump -u user -pclave basedatos > bd.sql`
- Restauración de la BD
 - `mysql -u user -pclave < bd.sql`

Acceso a MySQL con PHP

1. Conexión
2. Seleccionar BD
3. Construir la cadena de consulta
4. Ejecutar la consulta
5. Recuperar los resultados y construir la página web
6. Repetir 3 a 5 las veces necesarias
7. Desconectar

Distintos modos de hacerlo

- MySQLi: Dirigido a Objetos
- MySQLi: Procedural
- PDO

1. Conexión

```
<?php
$servername = "localhost";
$username = "user";
$password = "clave";

$conn = new mysqli($servername, $username,
    $password);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->
connect_error);
}
echo "Connected successfully";

?>
```

2. Seleccionar la BD

```
<?php
$basedatos = "sample_database";

$conn->select_db($basedatos);

// o en el tiempo de crear la conexión

$conn = new mysqli($servername, $username,
    $password, $basedatos);

?>
```

3. Cadena de consulta

4. Ejecutar la consulta

```
<?php
```

```
$sql = "SELECT campo1, campo2 FROM  
basedatos";
```

```
$result = $conn->query($sql);
```

```
?>
```

5. Recuperar resultados y construir la página web

```
<?php
if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "campo1: " . $row["campo1"].
        "– campo2: " . $row["campo2"] . "<br>";
    }
} else {
    echo "0 results";
}

?>
```

7. Desconectar

```
<?php  
    $conn->close();  
?>
```

1. Conexión

```
<?php
$db_hostname = 'localhost';
$db_database = 'basedatos';
$db_username = 'usuario';
$db_password = 'clave';

$db_conn = mysql_connect($db_hostname,
    $db_username, $db_password);
if (!$db_conn)
    die("No puedo conectar con MySQL: " .
        mysql_error() );
?>
```

2. Seleccionar BD

```
<?php  
mysql_select_db($db_database)  
    or die("No puedo seleccionar la BD: " .  
mysql_error());  
?>
```

3. Construir la cadena de consulta

4 Ejecutar la consulta

MySQLi-Procedural

```
<?php
    $consulta = "SELECT * FROM ALUMNOS";
    $result = mysql_query($consulta);

    if (!$result)
        die ("Error en el acceso a la BD: "
            . mysql_error());
?>
```


5. Recuperar los resultados y construir la página web

MySQLi-Procedural

```
<?php
    $num_filas = mysql_num_rows($result);

    for ($j = 0; $j < $num_filas; ++$j)
    {
        echo "Nombre: " . mysql_result($result, $j,
            'nombre') . ' <br />';
        echo "Apellido: " . mysql_result($result, $j,
            'apellido') . ' <br />';
        echo "Curso: " . mysql_result($result, $j,
            'curso')
            . ' <br /><br />';
    }
?>
```

Más eficiente

```
<?php
    $num_filas = mysql_num_rows($result);

    for ($j = 0; $j < $num_filas; ++$j)
    {
        $fila = mysql_fetch_row($result);
        echo "Nombre: " . $fila[0] . ' <br />';
        echo "Apellido: " . $fila[1] . ' <br />';
        echo "Curso: " . $fila[2] . ' <br /><br
/>';
    }
?>
```

7. Desconectar

MySQLi-Procedural

```
<?php  
    mysql_close ($db_server) ;  
?>
```

PDO EN PHP

PDO

- PHP soporta el paradigma PDO
- El lenguaje fue actualizado para permitir la definición de clases, objetos, variables de instancia, métodos, herencia y polimorfismo
- También soporta sobrecarga de operadores
- Incluye definición de constructores y destructores

Ejemplo de clase

```
<?php
class phpClassName{
    var $var1;
    var $var2 = "constant string";
    function myfunc ($arg1, $arg2) {
        [...]
    }
    [...]
}
?>
```

```
<?php
    class Books {
        /* Member variables */
        var $price;
        var $title;
        /* Member functions */
        function setPrice($par){
            $this->price = $par;
        }
        function getPrice(){
            echo $this->price . "<br/>";
        }
        function setTitle($par){
            $this->title = $par;
        }
        function getTitle(){
            echo $this->title . " <br/>";
        }
    }
?>
```

```
$physics = new Books;  
$maths = new Books;  
$chemistry = new Books;
```

```
$physics->setTitle("Physics for High School");  
$chemistry->setTitle("Advanced Chemistry");  
$maths->setTitle("Algebra");
```

```
$physics->setPrice(10);  
$chemistry->setPrice(15);  
$maths->setPrice(7);
```



```
class MyClass {  
    private $car = "skoda";  
    $driver = "SRK";  
  
    function __construct($par) {  
        // Statements here run every time  
        // an instance of the class  
        // is created.  
    }  
  
    function myPublicFunction() {  
        return("I'm visible!");  
    }  
  
    private function myPrivateFunction() {  
        return("I'm not visible outside!");  
    }  
}
```

Accesso a BD con classi

```
try {  
    $conn = new  
  
    PDO('mysql:host=localhost;dbname=myDatabase',  
        $username, $password);  
    $conn->setAttribute(PDO::ATTR_ERRMODE,  
        PDO::ERRMODE_EXCEPTION);  
} catch(PDOException $e) {  
    echo 'ERROR: ' . $e->getMessage();  
}
```

Consultas

```
$name = 'Joe'; # user-supplied data

try {
    $conn = new PDO('mysql:host=localhost;dbname=myDatabase',
$username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    $data = $conn->query('SELECT * FROM myTable WHERE name =
' . $conn->quote($name));

    foreach($data as $row) {
        print_r($row);
    }
} catch(PDOException $e) {
    echo 'ERROR: ' . $e->getMessage()
}
```

Sentencias “prepare”

```
$id = 5;
try {
    $conn = new PDO('mysql:host=localhost;dbname=myDatabase',
$username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    $stmt = $conn->prepare('SELECT * FROM myTable WHERE id =
:id');
    $stmt->execute(array('id' => $id));

    while($row = $stmt->fetch()) {
        print_r($row);
    }
} catch(PDOException $e) {
    echo 'ERROR: ' . $e->getMessage();
}
```

COOKIES, SESIONES, AUTENTICACIÓN

Memoria para las interacciones HTTP

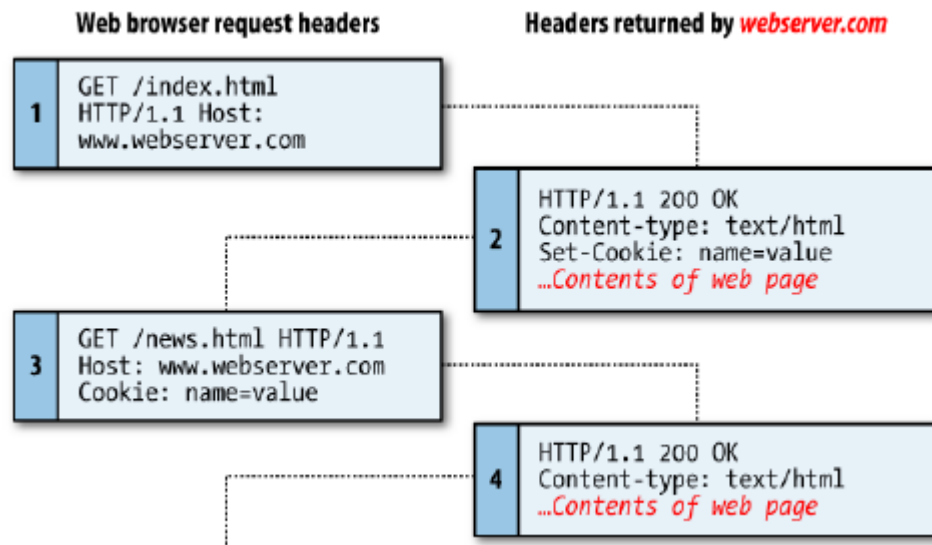
- Para hacer más fluida la relación entre un servidor web y sus clientes conviene disponer de *memoria* para anotar datos específicos de cada cliente.
- Por eso surgen las **cookies**.
- Cuando se hace necesario mantener información más organizada (estructurada), surge el concepto de **sesión**.

Cookies

- Mantener información sobre los usuarios
- *No necesariamente* nombres de usuario y claves; recuperar información para sus próximas conexiones
- **Cookie**: Dato que almacena un servidor web en el equipo local del cliente (de tamaño $\leq 4\text{KB}$). Cualquier tipo de dato no estructurado.
- Privacidad: sólo se pueden leer desde el servidor que las crea
- Autorización por parte del cliente

Sesión con cookies

- Las cookies se intercambian durante la transferencia de las cabeceras, antes del envío de la página html



Flujo de las cookies

- Tras la ejecución de `setcookie`, el servidor envía la información (nombre, valor) al cliente
- En cada petición posterior, el cliente incluye la cookie en la cabecera de la petición
- Habitualmente, `setcookie` debe aparecer en el script PHP antes de que se haya escrito ningún texto de la página html.

Crear una *cookie*

```
setcookie(name, value, expire, path, domain,  
secure, httponly);
```

```
setcookie('username', 'Hannah', time() + 60  
* 60 * 24 * 7, '/');
```

Parámetros de la cookie

- *name*: nombre
- *value*: hasta 4KB de caracteres alfanuméricos
- *expires*: fecha (Unix timestamp); cierre de navegador
- *path*: trayectoria en el servidor
- *domain*: dominio del servidor
- *secure*: ¿requiere conexión segura?
- *httponly*: ¿usar protocolo http?

Uso y eliminación de cookies

- **Acceso a una cookie:**

```
if (isset($_COOKIE['username']))  
$username = $_COOKIE['username'];
```

- **Eliminar una cookie:**

```
setcookie('username', 'Hannah', time()  
- 2592000, '/');
```

Autenticación HTTP

- Autenticación de usuarios para el acceso al servidor web. Requiere soporte del servidor (apache: mod_authz_ldpa, _kerb, _mysql, _shadow, _pam,)

```
<?php
if (isset($_SERVER['PHP_AUTH_USER']) &&
    isset($_SERVER['PHP_AUTH_PW']))
{
    echo "Usuarior: " . $_SERVER['PHP_AUTH_USER'] .
        " Clave: " . $_SERVER['PHP_AUTH_PW'];
}
else
{
    header('WWW-Authenticate: Basic realm="Restricted Section"');
    header('HTTP/1.0 401 Unauthorized');
    die("Por favor, inserte su nombre de usuario y clave");
}
?>
```

Almacenamiento de logins y claves

- Uso de funciones de un único sentido para guardar información sensible:

```
$token = md5('mypassword');
```

Valor de \$token:

```
34819d7beeabb9260a5c854bc85b3e44
```

- Salpimentar (añadir caracteres antes y después de la clave, previo a su cifrado):

```
$token = md5('hqb$tmypasswordcg*1');
```

Sesiones

- Grupos de variables/valores almacenados en el servidor, relacionados con el usuario actual.
- La relación (usuario; sesión) se fija mediante cookies.
- Las sesiones se inician llamando a **session_start**.
- Las variables de sesión se guardan en el array `$_SESSION`.
- La sesión se finaliza llamando a **session_destroy**.

FRAMEWORKS

¿Qué es un *framework*?

- Marco de trabajo: conjunto estandarizado de conceptos y procedimientos para abordar problemas de un mismo tipo
- Estructura conceptual y tecnológica para desarrollo de módulos software reutilizables para soluciones problemas similares. Incluyen programas, bibliotecas y lenguaje
- Model-View-Controller

Ventajas

- Estandarización
- Reutilización de código
- Más eficiencia
- Rapidez en el desarrollo

Inconvenientes

- Limitado a un tipo de problemas. Sólo los contemplados en el framework
- A veces, difícil de adaptar a problemas distintos

PHP Frameworks

- www.phpframeworks.com:
 - Zend
 - CakePHP
 - QPHP
 - Symfony

Zend (framework.zend.com)

- Modular, con muy bajo acoplamiento
- Seguro
- Soporte para PHPUnit
- Extensible
- Alto rendimiento

MÁS SOBRE DESARROLLO EN PHP

PEAR: PHP Extension and Application Repository



- Entorno de desarrollo y sistema de distribución de componentes de código PHP
- Catálogo extenso de bibliotecas de PHP
- Desarrolladas con PDO
- pear.php.net

Desarrollo PHP basado en patrones

- Reutilizar un esquema definido, basado en el soporte natural del lenguaje de programación. Libro «Gang of the Four»
- Programación más efectiva y segura.
- W. Sanders, «Learning PHP Design Patterns», O'Reilly, 2013

Otros lenguajes y frameworks

- Python, Django
- Ruby, Ruby on Rails
- Java, Swing
- Scala, Lift
- Clojure, Luminus
- ...