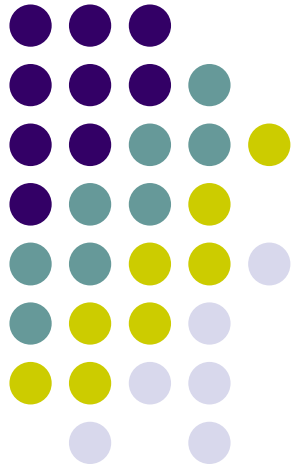
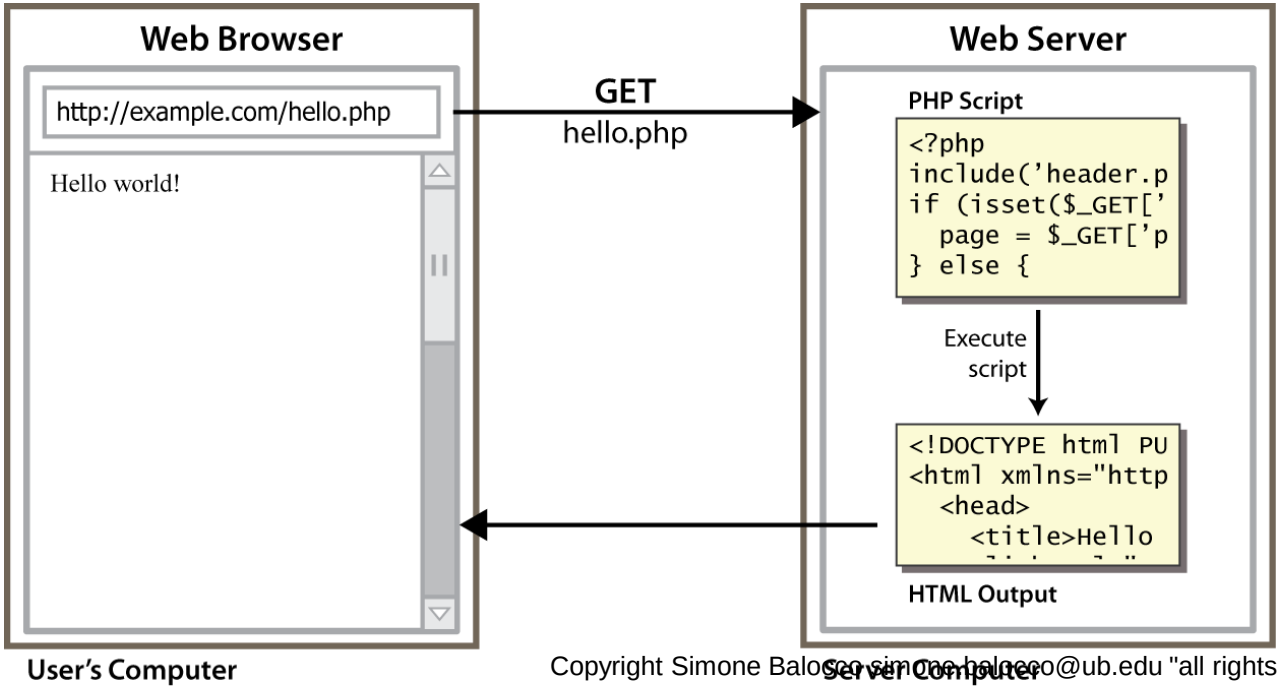
PHP programación servidor



Programacion servidor

- En general cuando se especifica un URL en el browser:
  - el ordenador busca el IP del servidor usando el DNS
  - el browser se conecta a ese IP y pide el fichero
  - el programa del web server (ej. Apache) recupera el fichero

  desde el sistema de ficheros locales y lo envía al browser



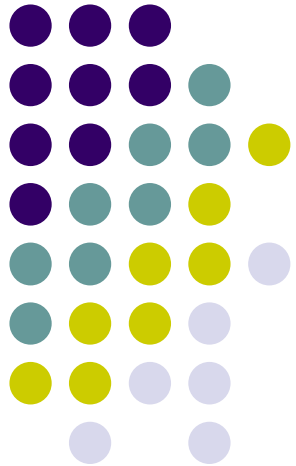
Lenguajes servidor

- Las paginas del servidor pueden estar escritas en muchos lenguajes web
  - ejemplos: PHP, Java/JSP, Ruby on Rails, ASP.NET, Python, Perl

Porque elejimos PHP?

- Libre y open source. Cualquier persona puede usar un servidor PHP gratuitamente
- Compatible: Soportado por la mayoria de los servidores web
- Sencillo: muchas funcionalidades
- Bien documentado: escribiendo php.net/functionName se obiene ayuda

Repaso formularios y envío datos cliente



Ejemplo de form (HTML)

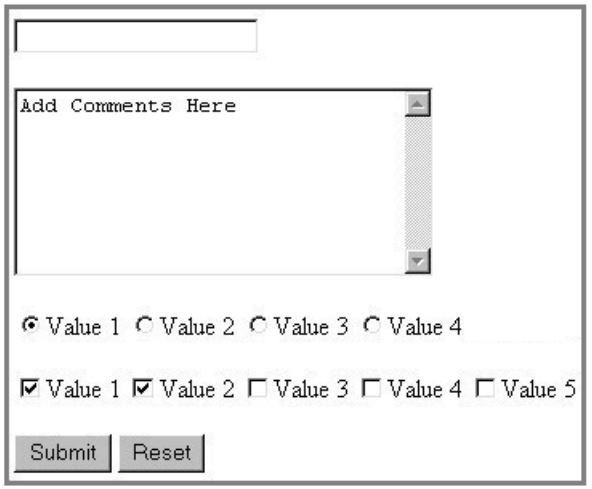
- input crea los elementos principales del form
- name especifica el nombre del parámetro de la query que pasamos al servidor
- type puede ser ***button, checkbox, file, hidden, password, radio, reset, submit, text, ...***
- value especifica el texto inicial

```
<form action="http://www.google.com/search">
    <div>
        Let's search Google:
        <input name="q" />
        <input type="submit" />
    </div>
</form>
```

Parametros de un web Query

- Form es una interficie interactiva en la cual se recogen informaciones introducidas por el usuario (cliente) y se envían al servidor web

- El código PHP produce una salida diferente en función de parámetros pasados por el usuario

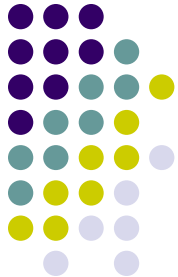


URL?name1=value1&name2=value2...

http://www.google.com/search?q=Obama

http://example.com/student_login.php?username=stepp&id=1234567

Input (HTML)



- **input** : disabled, maxlength, readonly, size, value
- **Size**: controla la dimensión del texto en pantalla
- **Maxlength:** limita cuantos caracteres se pueden escribir en un campo

Ejemplos:

```
<input type="text" size="10" maxlength="8" /> ID <br />
<input type="password" size="16" /> Password
<input type="submit" value="Log In" />
```

Otros Input y botones (HTML)

- Textarea

```
<textarea rows="4" cols="20">
Type your comments here.
</textarea>
```

- Otros input (checkbox, radio, reset)

```
<input type="checkbox" name="pickles" checked="checked" />
```

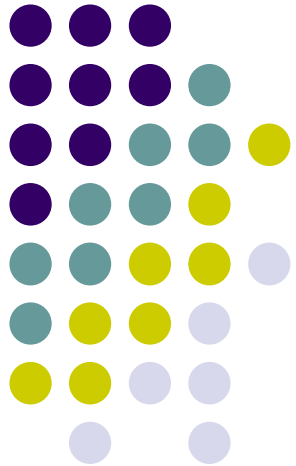
  - Se envía al servidor

http://web....php? pickles=on

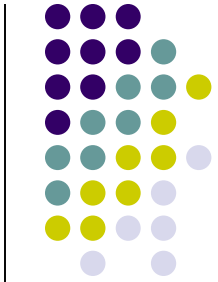
- Dropbown lists

```
<select name="favoritecharacter">
        <option>Jerry</option>
        <option selected="selected">Anna</option>
</select>
```

Introducción PHP



Los comentarios



- Existen tres formas de comentar el código en PHP

```
# single-line comment

// single-line comment

/*
multi-line comment
*/
```

Primer ejemplo en php

Hello_world.php

```
<?php
echo “Hello world in PHP ";
?>
```

- <?php ?> son tags de inicio y fin. Alternativamente: <? ?>
- Para ejecutarse el fichero hello_world.php tiene que encontrarse en un repertorio del servidor.
  - Por ejemplo si se encuentra en c:\xampp\htdocs\COW\
  - Se ejecuta escribiendo **"http://localhost:8080/COW/Hello_world.php**
    Atención! El servidor Apache tiene que esta activo

Incrustar código HTML en PHP

```
<?php
Codigo....
?><br><?php //Opcion 1

echo "<html><br></html>"; //Opcion 2
Codigo....
?>
```

-
 es código HTML

Input cliente – output servidor

- Cliente: (*.HTML)
- El form creado a la pagina index.html genera el siguiente output

https://localhost/test_page.php?base=3&exponent=4

- Servidor: (*.PHP)

```
<?php
    $base = $_GET["base"];
    $exp = $_GET["exponent"];
    $result = pow($base, $exp);
    echo "$base ^ $exp = $result";
?>
```

Output a pantalla:

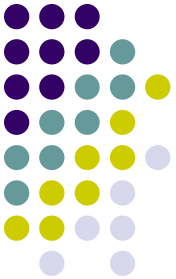
- 3 ^ 4 = 81

IMPORTANTE:
Los ficheros PHP no se ejecutan en el navegador como los ficheros HTML,
se crean en un repertorio del servidor (en el caso de XAMPP en “C:\xampp\htdocs”)

Incluir otro fichero

- Para insertar el contenido de un PHP en otro fichero antes que el servidor lo ejecute.
- Opciones:
  - `include()` genera un warning si no encuentra el fichero, pero el script continua la ejecución.
  - `require()` genera un fatal error si no encuentra el fichero, y el script se para.

Ejemplo include



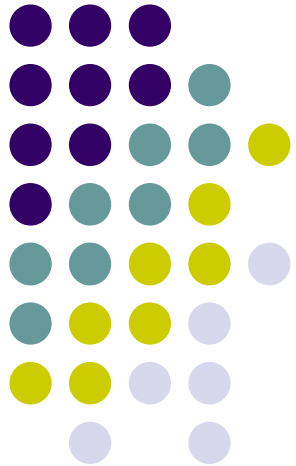
-Menu.php-

```
<a href="/default.php">Home</a>
<a href="/tutorials.php">Tutorials</a>
<a href="/references.php">References</a>
<a href="/examples.php">Examples</a>
<a href="/contact.php">Contact Us</a>
```

-main.php-

```
<html>
<body>
<div class="leftmenu">
<?php include("menu.php"); ?>
</div>
<h1>Welcome to my home page.</h1>
</body>
</html>
```

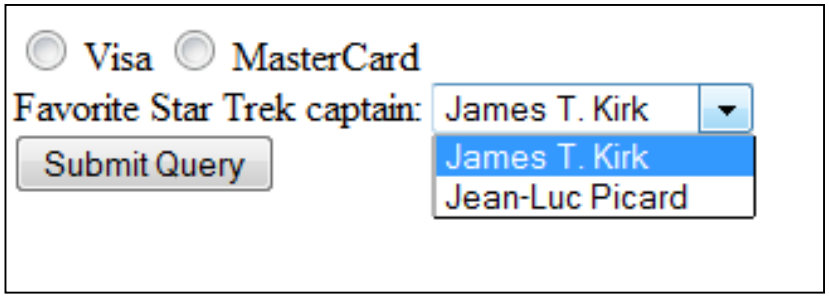
Envio datos con formularios en PHP



Ejemplo de envío datos cliente

Index.html

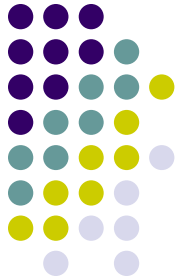
```
<form action="https://localhost/test_page.php">
    <div>
      <label><input type="radio" name="creditcard" value="visa" /> Visa</label>
      <label><input type="radio" name="creditcard" value="mastercard" />
    MasterCard</label> <br />
      Favorite Star Trek captain:
      <select name="startrek">
          <option value="kirk">James T. Kirk</option>
          <option value="picard">Jean-Luc Picard</option>
       </select> <br />
    <input type="submit" />
    </div>
</form>
```



- Query:

https://localhost/test_page.php?creditcard=mastercard&startrek=picard

Peticiones HTTP Get vs Post



- **GET** : pide al servidor una pagina o datos
  - Si el pedido tiene parámetros, ellos se envían por medio del URL como un string query
- **POST** : envía los datos a un web server y recupera la respuesta del servidor
  - Si el pedido tiene parámetros, ellos se incluyen en el paquete HTTP pedido, y no en el URL

CUAL ELEGIR?

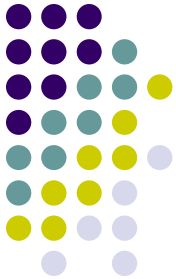
- Al fin de enviar datos, un POST es mas apropiado que un GET
  - GET necesita incrustar los parámetros en el URL
  - Los URL están limitados a una longitud (~ 1024 caracteres)
  - Los URL no pueden contener caracteres especiales de codificado
    - (Se soluciona con el comando htmlspecialchars)
  - Los datos privados en un URL pueden ser modificados por otros usuarios

Comandos para peticiones

- $_GET, $_POST
- $_SERVER, $_ENV
- $_FILES
- $_COOKIE
- $_SESSION,

- Para recuperar parámetros pasados en peticiones GET y POST
- Informaciones sobre el servidor web
- Ficheros cargados con la petición web
- "cookies" usados para identificar el usuario
- Control de sesión

Ejemplo Post



Cliente.html

```
<form
    action="http://localhost:8080/COW06/servi
    dor.php" method="POST">
<div>
name: <input type="text" name="var1" />
password: <input type="text" name="var2" />
<input type="submit" />
</div>
</form>
```

Servidor.php

```
<?php
$var1= $_POST[" var1"];
$var2 = $_POST[" var2"];
echo "Hello $var1  $var2 ";
?>
```

GET or POST?

```
if ($_SERVER["REQUEST_METHOD"] == "GET") {
….}
elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
….}
```

Redirección Post/Get al mismo fichero

- Un form puede enviar datos a él mismo
  - Ventaja, reducir el numero de paginas

```
if ($_SERVER["REQUEST_METHOD"] == "GET")
    {
    # pedido GET normal;
    ?>
    <form action="" method="post">...</form>
    <?php
}
elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
    # pedido POST; el usuario envia el form aqui y lo processa
    $var1 = $_POST["param1"];
    ...
}
```

Ejemplo post al mismo fichero

```
<html><body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
   Name: <input type="text" name="fname">
   <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
     // collect value of input field
     $name = $_POST['fname'];
     if (empty($name)) {
         echo "Name is empty";
     } else {
         echo $name;
     }
}?>
</body></html>
```

Name: Submit Query

Name: Submit Query
hola

Subir ficheros al servidor

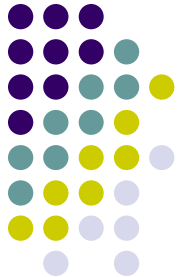
```
<form action="http://webster.cs.washington.edu/params.php"
      method="post" enctype="multipart/form-data">
      Upload an image as your avatar:
      <input type="file" name="avatar" />
      <input type="submit" />
</form>
```

El Servidor recibe global array $_FILES, y no $_POST

Cada elemento de $_FILES es un array asociativo y contiene:

- name : el nombre de fichero local cargado
- type : el tipo de data que ha sido cargado (image/jpeg)
- size : el tamaño del fichero en bites.
- tmp_name : un nombre del fichero adonde el PHP ha grabado de forma temporánea en el fichero.
  - CUIDADO: para cargar permanentemente el fichero, se tiene que mover a otro lugar.

Subir ficheros



- Por ejemplo si se carga un fichero llamado “borat.jpg” con un parámetro llamado avatar

$_FILES["avatar"]["name"] será "borat.jpg"
$_FILES["avatar"]["type"] será "image/jpeg"
$_FILES["avatar"]["tmp_name"] será "/var/tmp/phpZtR4TI"

- Habrá que procesar los ficheros:

```
if (is_uploaded_file($_FILES["avatar"]["tmp_name"])) {
    move_uploaded_file($_FILES["avatar"]["tmp_name"],
    "$username/avatar.jpg");
    print "Saved uploaded file as $username/avatar.jpg\n";
} else {
    print "Error: required file not uploaded";
}
```

Comandos del servidor

- $_SERVER["SERVER_NAME"]
- $_SERVER["SERVER_ADDR"]
- $_SERVER["REMOTE_HOST"]
- $_SERVER["REMOTE_ADDR"]
- $_SERVER["HTTP_USER_AGENT"]
- $_SERVER["HTTP_REFERER"]
- $_SERVER["REQUEST_METHOD"]

- nombre de este web server "webster.cs.washington.edu"
- direccion IP address del web server "128.208.179.154"
- dominio del usuario "hsd1.wa.comcast.net"
- direccion IP del usuario "57.170.55.93“
- web browser del usuario "Mozilla/5.0 (Windows; ...“
- Que pagina el usuario ha visitado antes de esta pagina http://www.google.com/
- HTTP metodo usado para contactar el servidor "GET" or "POST"

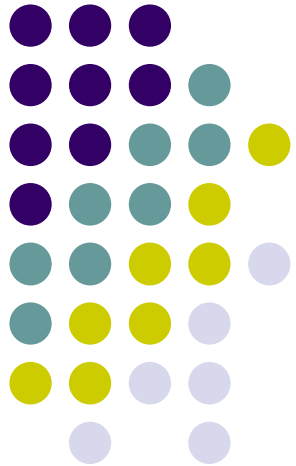
Lista completa metodos de $_Server :
http://php.net/manual/en/reserved.variables.server.php

Valores de los errores

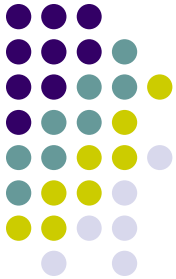
- Codigo HTTP
  - 200
  - 301-303
  - 400
  - 403
  - 404
  - 500
- Significado
  - OK
  - La pagina ha sido desplazada
  - Pedido illegal
  - No tienes aceso a esa pagina
  - Pagina no encontrada
  - Problema interno del servidor

Lista completa

Sintaxis y arrays en PHP



String interpretadas



- Los strings entre " " son interpretados

```
$age = 16;
print "You are " . $age . " years old.\n";
print "You are $age years old.\n";
```

- Los strings entre ' ' no son interpretados:

```
print 'You are $age years old.\n';
```

Arrays

- Crear array

```
$name = array();
$name = array(value0, value1, ..., valueN);
```

- Recuperar un valor

```
$name[index]
```

- Definir un valor

```
$name[index] = value;
```

- Añadir un valor

```
$name[] = value;
```

- Array vacio (tamaño 0)

```
$a = array();
```

- Asigna 23 al index 0

```
$a[0] = 23;
```

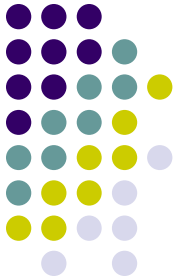
- Añade un element al final de un array

```
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";
```

Funciones con arrays

Nombre función	descripción
count	Cuenta el numero de elementos en un array
print_r	Imprime el contenido de un array
array_pop, array_push, array_shift, array_unshift	Usa el array como una pila/cola
in_array, array_search, array_reverse, sort, rsort, shuffle	Busca y re-ordena
array_fill, array_merge, array_intersect, array_diff, array_slice, range	crea rellena, filtra
array_sum, array_product, array_unique , array_filter, array_reduce	Procesa un elemento

Loop foreach



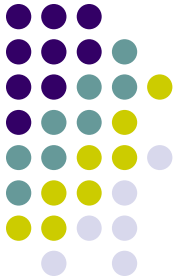
- Syntaxis

```
foreach ($array as $variableName)
```

- Ejemplo:

```
$nombres = array("Larry", "Moe", "Curly", "Shemp");
for ($i = 0; $i < count($ nombres ); $i++) {
  print "Moe slaps {$ nombres [$i]}\n";
}
foreach ($nombres as $nombre) {
  print "Moe slaps $nombre\n";
}
```

Arrays asociativos



- Sugerencia: para transmitir varias informaciones al mismo tiempo:

```
$blackbook = array();
$blackbook["marty"] = "206-685-2181";
$blackbook["stuart"] = "206-685-9138";
...
print "Marty's number is " . $blackbook["marty"]
    . ".\n";
```

Ejemplos:

- Creación

```
$name = array();
$name["key"] = value;
...
$name["key"] = value;

$name = array(key => value, ..., key =>
    value);
```

- ejemplo

```
$blackbook =
        array("marty"  => “607685218",
            "stuart" => "607685913",
            "jenny"  => "607867530");
```

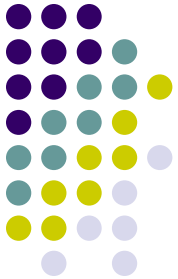
- Impresión

```
print_r($blackbook);
Array
(
    [jenny] => 607867530
    [stuart] => 607685913
    [marty] => 607685218
)
```

- alternativamente

```
foreach ($blackbook as $key =>
    $value) {
    print "$key's phone number is
    $value\n";
}
```

Mas ejemplos:



- creación:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

- Alternativamente:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

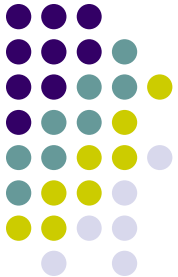
- Impresión

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

Resultado:

Peter is 35 years old.

Mas ejemplos:



- Loop entre elementos

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value)
    {
        echo "Key=" . $x . ", Value=" . $x_value;
        echo "<br>";
    }
    ?>
```

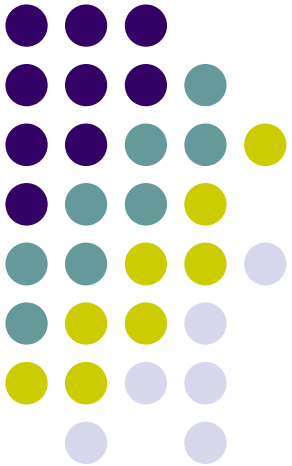
- Resultado

Key=Peter, Value=35

Key=Ben, Value=37

Key=Joe, Value=43

Clases en PHP



Funciones con PHP

```php
<?php
function cube($x)
   {
   return $x * $x * $x;
   }
echo "The cube of 4 is :
   ".cube(4)."<br />";
?>
```

- Pasaje de parámetros por valor y devolución de resultado

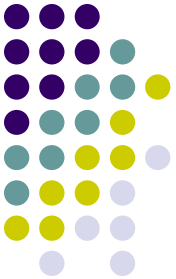
Programación a objetos con PHP

```php
<?php
class Myclass
{
 public $font_size =10;
}
$f = new MyClass;
echo $f->font_size;
?>
```

- Se puede instanciar una clase
- Error típico:

NO-> $f->$font_size

Programación a objetos con PHP



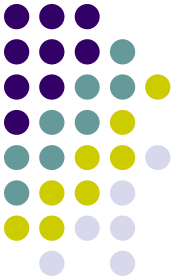
```php
<?php
class Myclass
{
  public $font_size ="18px";
  public $font_color = "blue";
  public $string_name = "w3resource";
  public function customize_print()
   {
   echo "<p style=font-  size:".$this-
   >font_size.";color:".$this-
   >font_color.";>".$this-
   >string_name."</p>";
   }
}
$f = new MyClass;
echo $f->customize_print();
?>
```

- El objeto se ha instanciado
- Se ha creado el método customize_print()
- Se usa

`.$this->font_size.`

Como referencia a los atributos de la clase

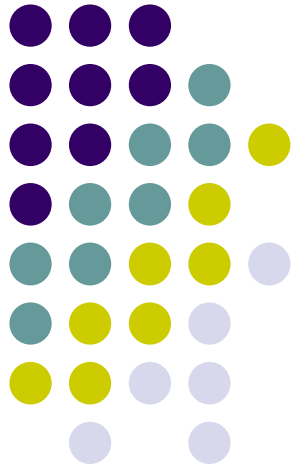
Programación a objetos con PHP



```php
<?php
class Myclass
 {
 private $font_size;
 private $font_color;
 private $string_value;
 function
     __construct($font_size,$font_color,$string_val
     ue)
 {
 $this->font_size = $font_size;
 $this->font_color = $font_color;
 $this->string_value = $string_value;
 $this->customize_print();
 }
 function customize_print()
 {
 echo "<p style=font-size:".$this-
     >font_size.";color:".$this-
     >font_color.";>".$this->string_value."</p>";
 }
 }
$f = new MyClass('20px','red','Object Oriented
     Programming');
?>
```

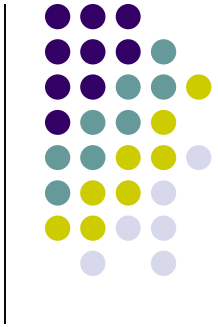
- Declara tres variables private
- Declara un constructor que acepta tres parámetros y el método customized_print()
- Crea un nuevo objeto y pasa tres parámetros

PHP funciones Input/Output



Nombre función	categoría
file, **file_get_contents**, **file_put_contents**	Lectura escritura ficheros
basename, file_exists, filesize, fileperms, filemtime, is_dir, is_readable, is_writable, disk_free_space	Informaciones ficheros
copy, rename, unlink, chmod, chgrp, chown, mkdir, rmdir	Gestión ficheros y repertorios
glob, **scandir**	Lectura repertorios

Lectura/escritura



contents of prueba.txt	file(“prueba.txt")	file_get_contents(“prueba.txt")
Hello how r u? I'm fine	array("Hello\n", # 0 "how r u?\n", # 1 "\n", # 2 "I'm fine\n" # 3)	"Hello\n how r u?\n # un unico \n # string I'm fine\n"

- file devuelve las líneas del fichero como en un array (\n al final de linea)
- `file_get_contents` devuelve el contenido de un fichero como un unico string
- `file_put_contents` escribe el string en un fichero

ejemplo

```
$lines = file(“pruebat.txt");
foreach ($lines as $line) {
   print $line;
}
```

- Problema: ese codigo imprime tambien el \n final.

- Solucion:

```
$lines = file("todolist.txt",
   FILE_IGNORE_NEW_LINES);
```

Separar string

- `$array = explode(delimiter, string);`
- `$string = implode(delimiter, array);`
- explode y implode se usan para convertir entre string y arrays

- Ejemplo:

```
<?php foreach (file("names.txt") as
    $name)
     {
     $tokens = explode(" ", $name);
     ?>
     <p> author: <?= $tokens[2] ?>,
    <?= $tokens[0] ?> </p>
     <?php
     }
?>
```

- Nombres.txt

```
Marty D Stepp
Jessica K Miller
Victoria R Kirst
```

- Resultado:

```
author: Stepp, Marty
author: Miller, Jessica
author: Kirst, Victoria
```

Separar string en variables distintas

- Definición:

```
list($var1, ..., $varN) = array;
```

- Ejemplo:

```
“Agenda.txt”
Allison Obourn
(206) 685 2181
570-86-7326
```

```
list($name, $phone, $ssn) = file("agenda.txt");

list($area_code, $prefix, $suffix) = explode(" ",
  $phone);
```

Lectura repertorios

function	description
glob	Devuelve un array con todos los nombres que corresponden a un patrón definido. (Por ejemplo “prueba/bar/myfile.txt")
scandir	Devuelve un array de todos los nombres de los ficheros de una directory especificada (por ejemplo "myfile.txt")

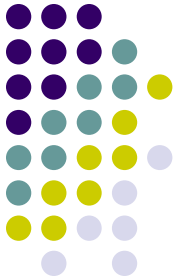
- `glob(“prueba/bar/*.doc")` devuelve todos los ficheros.doc en el repertorio “prueba/bar”
- `glob("food*")` devuelve todos los ficheros que empiezan por "food"

Lectura repertorios (glob)

- Ejemplo: Invierte el texto de todos los poemas contenidos en un repertorio

```
$poems = glob("poetry/poem*.dat");
foreach ($poems as $poemfile) {
  $text = file_get_contents($poemfile);
  file_put_contents($poemfile, strrev($text));
  print "I just reversed " . basename($poemfile) . "\n";
}
```

Lectura repertorios (scandir)



```
<ul>
  <?php foreach (scandir("taxes/old") as $filename)
  { ?>
    <li>I found a file: <?= $filename ?></li>
  <?php } ?>
</ul>
```

- Resultado:

```
I found a file:.
I found a file: ..
I found a file:  2007_w2.pdf
I found a file: 2006_1099.doc
```

Añadir contenido a un fichero existente

```
$new_text = "P.S. ILY, GTG TTYL!~";
file_put_contents("poem.txt", $new_text, FILE_APPEND);
```

Contenido viejo	Contenido nuevo
Roses are red, Violets are blue. All my base, Are belong to you.	Roses are red, Violets are blue. All my base, Are belong to you. P.S. ILY, GTG TTYL!~