Universidad
Politécnica
de Cartagena

Internet applications

Programming languages on the server side: PHP

Introduction

Today, most web pages are dynamically generated on the server before being sent to the client, even if the server is not processing any application

You need to develop applications that run on the server and generate the HTML response.

The CGI standard is insufficient for current needs: it's too computationally expensive to run an application on a context (process) different for each request.

Programming languages modularly integrated on the web server itself are used.

- When the server detects that the URL addresses a **registered application extension** the requested document is handled by the appropriate module.
- Ex. If th local paths is /procesar.php, the file procesar.php is opened and processed by the PHP interpreter (PHP preprocessor).

PHP features

Simple. Quick development

- Interpreted language (*script*): No need to compile.
- No need to declare data types.

Built upon the best features of previous languages: Perl, C, etc.

Lots of utility libraries: DB, image processing, XML, etc.

Widespread use. Lots of additional libraries and frameworks.

Usually directly integrated in the HTTP server.

Basic use

- PHP can be interleaved in a HTML document.
- A file with PHP code is identified with the .php extension. The server recognizes the extension and runs the PHP interpreter.
- The PHP preprocessor interprets as PHP code everything that is between the tags ***<?php*** *and* ***?>***, everything else is simply sent back to the client.

```
<html>
<body>
        <H1>
        <?php threw out
'Hello World';
        ?>
        </ H1>
</body>
</html>
```

Variables and types

- Start with $: $ i = 0;
- No need to declarem the type: PHP performs **type conversions automatically.**
- Declared in the body of a file, variables are global to that file and the files included by it.
- Declared in a function, they are local to that function.

```
$mivar = 123;
echo $mivar; // converted to string
```

```
$mivar= '3'; // Cast to integer
$mivar = 2 + $mivar; //to  perform the sum
```

Type of data

- **Integer, decimal, octal or hexadecimal**: $I = 123;
- **floating point**: $I = 1.3e4;
- **strings**: $C = "text string \ n";
- **Objects**: $obj= new My class();
- **arrays**: $V[2] = 123;
- **associative arrays** : $V['Name'] = "Homer";
  - The index is a string.

```
$MyArray[0] = 1;
$MyArray[1] = "Hello!! ";
$MyArray[] = 3;
echo $MyArray[2]; //Print 3
$MyArray[ "first name"] =" Homer ";
echo $MyArray[0]; // 1
echo $MyArray[ "first name"]; // "Homer"
```

Strings

- Delimited by double (“) quotation marks to expand variables:

```
$a = Hello;
echo "$a" // Print "Hello"
```

- Delimited by single (') quotes, variables not expanded:

```
$a = Hello;
echo '$a' // Print "$a"
```

- Concatenation: operator .

```
$a = "Hello ";
$b = "and Goodbye";
$c = $a.$b;
print "$c"; //Prints "Hello and good bye"
```

Arithmetic operators.

Operator	Name	Result
$a + $b	Addition	addition de $a y $b.
$a - $b	Substraction	difference between $a y $b.
$a * $b	Multiplication	Product $a y $b.
$a / $b	Division	Quotient $a y $b.
$a % $b	Modulus	Remainder of $a/$b.

Auto-increment and decrement

Operation	Name	Result
++$a	Pre-increment	Incrementa $a en 1, y devuelve $a (incrementado).
$a++	Post-increment	Devuelve $a, y después lo incrementa en 1.
--$a	Pre-decrement	Decrementa $a en 1, y después lo devuelve.
$a--	Post-decrement	Devuelve $a, y después lo incrementa en 1.

Bit operators.

Operación	Nombre	Resultado
$a & $b	Y	Se ponen a 1 los bits que están a 1 en $a y $b.
$a | $b	O	Se ponen a 1 los bits que están a 1 en $a o $b.
$a ^ $b	O Exclusivo	Se ponen a 1 los bits que están a 1 en $a o $b, pero no en ambos.
~ $a	No	Se invierten los bits (se cambian 1 por 0 y viceversa.)
$a << $b	Desp. Izq.	Desplaza $b posiciones a la izquierda todos los bits de $a.
$a >> $b	Desp. Drch.	Desplaza $b posiciones a la derecha todos los bits de $a.

Logical operators.

Operación	Nombre	Resultado
$a and $b	Logical AND	Cierto si $a y $b son ciertos.
$a or $b	Logical OR	Cierto si $a o $b es cierto.
$a xor $b	Exclusive OR.	Cierto si $a o $b es cierto, p ambos.
!$a	Negation	Cierto si $a es falso.
$a && $b	Logical AND	Cierto si $a y $b son ciertos.
$a || $b	Logical OR	Cierto si $a o $b es cierto.

Allocation, equality and identity.

Operación	Nombre	Resultado
$a = $b	Assignment	Asigns the value of $b to $a.
$a == $b	Equality	Compares if the value is the same
$a === $b	Identity	Compares if the value is the same AND the type is equal.

Comparisons.

Operación	Nombre	Resultado
$a != $b	No igual	Cierto si el valor de $a no es igual al de $b.
$a !== $b	No idéntico	Cierto si $a no es igual a $b, o si no tienen el mismo tipo.
$a < $b	Menor que	Cierto si $a es estrictamente menor que $b.
$a > $b	Mayor que	Cierto si $a es estrictamente mayor que $b.
$a <= $b	Menor o igual que	Cierto si $a es menor o igual que $b.
$a <=> $b	Spaceship	-1 , 0, 1 si $a es menor, igual o mayor que $b respectivamente

Control Structures

- if (exp) {...} elseif (exp) {..} else {...}
- while (exp) {...}
- do {...} while (exp)
- switch (var) {Case v1: v2 case ..: ... default: ...}
- for (Expression1; expression2; expression3) {...}

```
$Factorial = 1;
for ($i= 2; $i<= 5; $i++)
{
    $Factorial*=$i;
}
```

```
switch ($i)
{
    case 1:
        threw out "Code 1 ";
    case 2:
        threw out "Code 2";
}
```

Control Structures

- *Foreach* loop: iterates through all the elements of a vector without using explicity its size.
- foreach (array_expression as $ value) {...}
- foreach (array_expression as $ key => $ value) {...}

```
$ A = array (1, 2, 3, 17);
foreach ($A as $v)
{
    print "Present value of element is: $v \
n";
}
```

```
$ A['first name'] = "Pepe";
$ A['dni'] = 123412;
foreach ($A  as $k => $v)
{
    print "Value of $k = $v \ n";
}
```

True or false

- numerical values

```
$ X = 1; // $ x
if ($ x) // is evaluates to true

$ X = 0; // $ xdefined as he whole 0
if ($ x) // is evaluates to false
```

- Strings

```
$x = "hello"; //assigned a string to $X
if ($x) // is evaluated to true

$x = ""; // empty string
if ($x) // evaluates to false

                // Exception:
$x= "0"; // zero string
if ($x) // evaluates to false
                // (cast to integer)
```

True or false

- arrays

```
$x = array (); // $ x is an empty array
if ($x) // evaluates to false

$x = array ( "a", "b", "c");
if ($x) // evaluates to true
```

functions

- Declaration:

```
function name ($ arg_1, $ arg_2, ..., $ arg_n)
{
    commands
    return $output;
}
```

- Example:

```
factorial function ($value) {
    if ($ value <0) {
        return -1; // Error
    }

    if ($value == 0) {
        return 1;
    }

    if ($ value == 1 == 2 || $ value) {
        return $ value;
    }

    $ ret = 1;
    for ($i= 2; $i <= $value; $i++) {
        $ret *= $i;
    }
    return $ret;
}

$F = factorial (5);
```

Integration with the server

- PHP is integrated with the server so that the scritps have access to context information about the requests
- **superglobals**: Are predefined variables which the interpreter automatically fills with information about the request, information sent by the user, cookies and others.
  - $ GLOBALS
  - $ _SERVER
  - $ _GET
  - $ _POST
  - $ _FILES
  - $ _COOKIE
  - $ _SESSION
  - $ _REQUEST
  - $ _ENV

Integration with the server

- The $_GET variable is an associative array whose indices correspond to the variables sent in a GET request.
- The variable $_POST is an associative array whose indices correspond to the variables sent in a POST request.

```
<form action = "action.php" method = "POST">
Your first name: <input type =“text”
name=“firstname”> <br>
Your age: <Input type =“text” name =“age”>
<br>
<input type = submit>
</form>
```

```
Hello <? echo $_POST[‘first name’]>.
You are <? echo $_POST[‘age’]> years old.
```

Cookies

To send a cookie use *setcookie()*:

```
int setcookie (string first name [, String
value
                [, int end [, string path
                [, String domain
                ]]]]])
```

```
setcookie ( "PruebaCookie"
           "I shall expire within an hour“,
           time () + 3600);
```

Used the superglobal $_COOKIE to process them:

- An associative array indexed with the names of the received cookies.

```
Received cookie <? echo $_COOKIE[‘visits']?>
```

Bibliography

- All documentation www.php.net
- Examples and tutorials on many other sites,
- To practice without the HTTP server: http://phpcodepad.com/