






**Web programming**

The PHP language



**Our objective**

- ⊃ Teaching you everything about PHP? Not exactly
- ⊃ Goal: teach you how to interact with a database via web
  - Access data inserted by users into HTML forms
  - Interact with a DBMS (MySQL in particular): connect to a database, execute a query, store the result of the query...
  - Access the tables returned by the DBMS
  - Assemble the HTML page on the browser, composed by HTML instructions and data extracted from the database


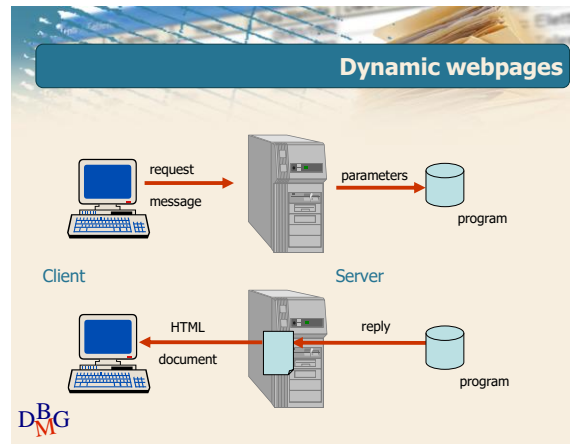
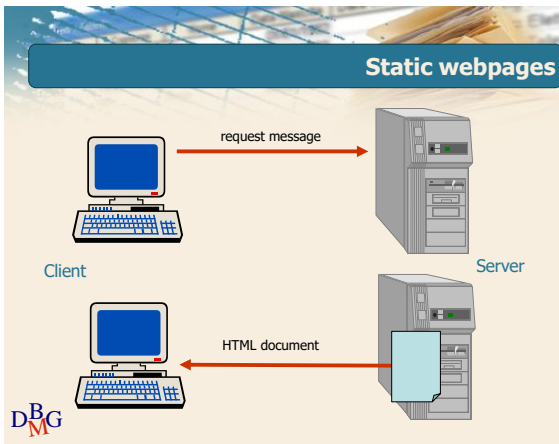
**Contents**

- ⊃ Overview of the PHP language
  - Structure of a program
  - Variables and types (*associative arrays*)
  - Expressions and operators
  - Control structures (*foreach*)
- ⊃ Parameter acquisition from HTML forms

**What is PHP**

- ⊃ Born in 1994
  - Personal Home Page, today known as PHP Hypertext Preprocessor
- ⊃ Created specifically for the development of dynamic webpages
- ⊃ Many useful resources, e.g.
  - <http://php.html.it/guide/>
  - <http://www.web-link.it/php/>

### Primary goal

- ▷ PHP's primary goal is to generate HTML code
  - In particular, generating HTML code by the results of an elaboration, that depend on the user input on the database contents, ...
- ▷ The PHP code is inserted inside the HTML code
  - The PHP code is executed on the Web server and the result (HTML and script result) is sent back to the browser



### Why using PHP?

- ▷ Available for many platforms, different in
  - Hardware (Intel, Sparc, Mac, etc....)
  - Operative system (Linux, Unix, Windows, etc...)
  - Web server (Apache, IIS, IPlanet, etc...)
- ▷ PHP code is "highly portable"
- ▷ The PHP interpreter is Open Source
  - Free, wide availability of tools, support, developers, community of users
- ▷ Pretty easy to learn, very simple if you already know C
- ▷ Able to interact with various Database Management Systems (MySQL, Postgres, Oracle, ...)



### First example

- ▷ Text file with .php extension

```
<html>
<head>
  <title>Hello world! </title>
</head>
<body bgcolor="#FFFFFF">
  <?php
    // This is PHP code
    echo "<h1> Hello world! </h1>";
  <?
</body>
</html>
```

Hello World !



### First example

Hello World !

- ▷ If I look at the source code on the browser...

```
<html>
<head>
  <title>Hello world! </title>
</head>
<body bgcolor="#FFFFFF">
  <h1> Hello world! </h1> </body>
</html>
```

- ▷ Why?

- The browser shows the result of the execution of the PHP file, NOT the PHP file



### "Printing" strings

- ▷ One of the most important (and frequent) tasks of PHP code is to create HTML code that will be displayed on the browser
  - echo and print constructs

```
<?php
// They all produce the same output
echo "<h1>Hello world! </h1>";
print "<h1>Hello world! </h1>";
echo ("<h1>Hello world! </h1>");
print ("<h1>Hello world! </h1>");
<?>
```

Hello World !




### A quick digression: XAMPP

- ▷ XAMPP is a cross-platform Apache distribution that includes:
  - A web server (Apache)
  - A database management system (MySQL/MariaDB)
  - PHP and Perl script interpreters
  - A graphical administrator of MySQL database (phpMyAdmin)
- ▷ It can be used as a web-database development environment, thus making server-side scripts (e.g., PHP) and programs (e.g., DBMS, Web server) work locally



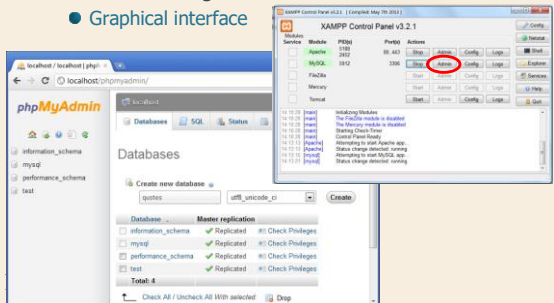

### A quick digression: XAMPP

- ▷ XAMPP installs all necessary software for the development and deployment of a local web site
  - The PC becomes client and server
- ▷ The Apache web server automatically creates a virtual domain (with local validity) at the localhost address (<http://127.0.0.1>)
  - Being connected to the Internet is not needed to use XAMPP



### XAMPP : DB administration

- ▷ Allows to manage databases
  - Graphical interface


### Tags to include PHP code

- ▷ PHP code can be inserted in any point of a HTML page
- ▷ Needs to be enclosed by tags

```
<?php
// This is PHP code
echo "<h1> Hello world! </h1>";
?>
```

```
<script language="PHP">
// This is PHP code
echo "<h1> Hello world! </h1>";
</script>
```

```
<?php
// This is PHP code
echo "<h1> Hello world! </h1>";
?>
```




### Another example

- ▷ Display the current date
- ▷ In a static way
  - And tomorrow?
- ▷ In a dynamic way
  - Updates in real time

```
<html>
<body>
Today is 09/12/2011
</body>
</html>
```


```
<?php
// dd/mm/aa format
$today = date("d/m/Y");
echo $today;
?>
```



### Code analysis


- ▷ In a script we use
  - Comments: //.....
  - Variables: \$today
  - Operators and language constructs: echo
  - Functions: date ()

```
<?php
// dd/mm/aa format
$today = date("d/m/Y");
echo $today;
?>
```



### Variables

- ▷ A variable is a symbol or a name that represents a value
- ▷ A variable can represent different types of value
  - Integer number, real number, character, ...
  - The data type can change during the execution of the program
- ▷ When a program is executed, variables are replaced by real data
  - The same program can elaborate different types of data sets in this way




### Variables

- ▷ In PHP the name of variables is preceded by the dollar symbol ('\$')
- ▷ PHP does not require that variables are declared before their usage
  - Higher flexibility with respect to other programming languages

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


```
<?php
$a = 9;
$b = 4;
$c = $a * $b;
echo "The result of the operation (9 * 4) is: ";
echo $c;
?>
```



### Data types

- ▷ PHP supports different data types
  - Boolean: true or false
  - Integer: decimal numbers
  - Float: floating point numbers
  - String
  - Array
  - Object
  - Resource
- ▷ Data types don't need to be set by the programmer but they are automatically detected by the PHP interpreter

```
<?php
$a = true;
$b = false;
$c = 18;
$d = -18;
$e = 9.876;
$f = 9.87e6;
?>
```




### Strings

- ▷ A string is a sequence of characters, with no length limitation
- ▷ Included between a couple of single or double quotes
  - If double quotes are used (""), the string content gets expanded (or, technically, "interpolated")


```
<?php
$num = 10;
$stringa = "<strong>The number is $num</strong>";
echo $stringa;
?>
```

The number is 10



### Arrays

- ▷ An array is a complex variable that contains a series of values, each of them characterized by a key (or index) that unambiguously identifies it
- ▷ PHP supports both scalar and associative arrays
  - Scalar arrays identify each element with its position number in a sequence
  - Associative arrays identify each element with a key in an univocal way




### Arrays

- ▷ Example of a scalar array

```
<?php
$colors = array('white', 'black', 'yellow', 'green', 'red');
echo $colors [1]; // prints 'black'
echo $colors [4]; // prints 'red'
?>
```

- ▷ Example of an associative array
  - The key can be a string or an integer


```
<?php
$a = array(
    "name" => "Mario",
    "surname" => "Rossi",
    "email" => "mario@rossi.com"
);
echo $a["name"]; // prints "Mario"
echo $a["email"]; // prints "mario@rossi.com"
?>
```



### Arrays

- ▷ Multidimensional arrays are possible

```
<?php
$a = array(
    "first" => array(
        "name" => "Mario",
        "surname" => "Rossi",
        "email" => "mario@rossi.com"
    ),
    "second" => array(
        "name" => "Marco",
        "surname" => "Verdi",
        "email" => "mario@verdi.com"
    )
);
echo $a["second"] ["email"]; // prints "mario@verdi.com"
?>
```



### Arrays

- Array elements can also be heterogeneous


```
<?php
$mix = array( 1, "hello" , 3.14, array( 1, 2, 3 ) );
```

- In PHP it's very easy to add or remove elements of an array

```
<?php
$x = array(25, 50);

$x[] = 75; // adds an element to the first available position
$x[4] = 125; // adds an element on the specified position
echo $x[3]; // Error!!! (The element does not exist)

unset($x[1]); // removes the specified element
unset($x); // removes the entire array
```




### Expressions and operators

- Arithmetic operators

```
<?php
$a = $x + 7; // addition
$b = $x - 2; // subtraction
$c = $x * 6; // multiplication
$d = $x / 2; // division
$e = $x % 4; // module (division remainder)

$x += 4; // increment $x by 4 (equivalent to $x = $x + 4)
$x -= 3; // decrement $x by 3 (equivalent to $x = $x - 3)
$x /= 5; // equivalent to $x = $x / 5
$x *= 4; // equivalent to $x = $x * 4
$x %= 2; // equivalent to $x = $x % 3

$a++; // increment by 1
++$a; // increment by 1
$a--; // decrement by 1
--$a; // decrement by 1
```




### Expressions and operators

- Logical operators

```
<?php
$x = $a && $b; // logical and
$x = $a || $b; // logical or
$x = $a xor $b; // logical xor
$x = !$a; // logical not
```

- Comparison operators

```
<?php
if ($a == $b) ... // equal
if ($a != $b) ... // not equal
if ($a > $b) ... // greater
if ($a >= $b) ... // greater or equal
if ($a < $b) ... // less
if ($a <= $b) ... // less or equal
```



### Expressions and operators


- String operations

  - Concatenation

```
<?php
$x = $x . $a; // the value of string $a is concatenated to string $x
$x .= $a; // equivalent
```


- Example

```
<?php
$name = 'Mario';
$surname = 'Rossi';
echo $name . ' ' . $surname; // prints Mario Rossi
echo "Name $name": // prints M. Rossi
echo $name[0] . " " . $surname; // prints M. Rossi
```




### Control structures

- Allow the conditional execution of parts of the program
- Allow the iterative execution of parts of the program
- Evaluate certain conditions
- PHP control structures
  - if, if..else, if..elseif
  - switch
  - while, do..while
  - for
  - foreach



### Conditions

- A condition is an expression that generates a boolean value (true or false)
  - They use comparison operators and Boolean operators
- The following values are equivalent to "false"
  - The Boolean value false
  - The integer number 0 and the real number 0.0
  - The empty string ("") and the "0" string
  - An empty array
- Each other value is considered true



### The IF construct

▷ If the condition expressed by the IF block is true, the piece of code is executed

```

<?php
$a = 2;
$b = 2;
if ($a == $b) {
    echo "\$a is equal to \$b, they are $a.\n";
}
    
```

### The IF .. ELSE construct

▷ If the condition expressed by the IF block is true, the sequence of operations follows the THEN branch, otherwise the ELSE branch

### The IF .. ELSE construct

▷ If the condition expressed by the IF block is true, the sequence of operations follows the THEN branch, otherwise the ELSE branch

```

<?php
$a = 2;
$b = 3;
if ($a == $b) {
    echo "\$a is equal to \$b, they are $a.\n";
} else {
    echo "\$a is different from \$b. \n\$a equals '$a' while \$b equals '$b'.";
}
    
```

### The IF .. ELSEIF construct

▷ Allows to choose among many options

```

<?php
$a = 2;
$b = 3;
if ($a == $b) {
    echo "\$a is equal to \$b.\n";
} elseif ($a > $b) {
    echo "\$a is greater than \$b.\n";
} else {
    echo "\$a is less than \$b.\n";
}
    
```

### The switch construct

▷ Allows to predict different possible values for an expression and to execute specific code according to the value

- Replaces a series of IF
- break: forces the exit from the switch block
- default: is optional

```

<?php
switch ($name) {
    case 'Luka':
    case 'George':
    case 'Frank':
        echo "Hello, my old friend!";
        break;
    case 'Anna':
        echo "Hello, Anna!";
        break;
    case 'Paolo':
        echo "Nice to meet you, Paolo";
        break;
    default:
        echo "Welcome, whoever you are";
}
    
```

### The while loop

▷ The block of instructions inside the while gets executed until the condition stays true

- It's possible that the cycle is never executed, in case the condition is false from the beginning
- In general the block of instructions modifies the parameters used in the condition

### The while loop

▷ The block of instructions inside the while gets executed until the condition stays true

```
<?php
    $smul = 1;
    while ($smul <= 10) {
        $ris = 5 * $smul;
        echo "5 * $smul = $ris<br>";
        $smul++;
    }
?>
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```



### The do .. while cycle

▷ Similar to the while, but it guarantees that the block of instructions is executed at least once

- The condition is checked after the execution of the block of instructions

```
<?php
    $smul = 11;
    do {
        $ris = 5 * $smul;
        echo "5 * $smul = $ris<br>";
        $smul++;
    } while ($smul <= 10)
?>
```

```
5 * 11 = 55
```



### The for cycle

▷ Allows to repeat a block of instructions directly defining

- The instructions of initialization, executed only once upon entering the cycle
- The condition, that, must be true to execute the block of instructions
- The update, executed at the end of iteration

▷ Can always be rewritten as a while loop

```
<?php
    for ($smul = 1; $smul <= 10; ++$smul) {
        $ris = 5 * $smul;
        echo "5 * $smul = $ris <br/>";
    }
?>
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```



### The foreach cycle

▷ Cycle created to simplify access to arrays

- Equivalent to a for cycle on the elements of an array

```
<?php
    $a = array("Andrea", "Paola", "Roberto", "Alice", "Sara");
    foreach ($a as $value) {
        echo "$value <br />";
    }
?>
```

```
Andrea
Paola
Roberto
Alice
Sara
```



### The foreach cycle on associative arrays

```
<?php
    $anno = array( "January" => 31,
                  "February" => 28,
                  "March" => 31,
                  "April" => 30,
                  "May" => 31,
                  "June" => 30,
                  "July" => 31,
                  "August" => 31,
                  "September" => 30,
                  "October" => 31,
                  "November" => 30,
                  "December" => 31);

    foreach ($anno as $month => $days) {
        echo "$month has $days days. <br />";
    }
?>
```

```
January has 31 days.
February has 28 days.
March has 31 days.
April has 30 days.
May has 31 days.
June has 30 days.
July has 31 days.
August has 31 days.
September has 30 days.
October has 31 days.
November has 30 days.
December has 31 days.
```






### PHP and HTML forms


```
<form name="userData" action="reponsePage.php" method="GET">
  Elementi di input
</form>
```

- ▷ "form" tag with some attributes
  - Name: form name
  - Action: name of the program that will elaborate form data
  - Method: how parameters will be passed from the form to the program (can be "GET" or "POST")
- ▷ Inside the form there are many input elements




### Access to form data

- ▷ The destination PHP script accesses to user inserted values through some special variables called "superglobal": the associative arrays \$\_GET, \$\_POST and \$\_REQUEST
  - "Superglobal" variables are accessible even inside some potential functions
- ▷ GET method
  - Values inside the query string are stored in the associative array \$\_GET
  - Each parameter of the form becomes a field of the associative array \$\_GET



### Access to form data

- ▷ POST method
  - Each parameter of the form becomes a field of the associative array \$\_POST
- ▷ The associative array \$\_REQUEST contains \$\_GET, \$\_POST and \$\_COOKIE
  - Even if it's not the same thing, in practice it can be used with any method, in alternative to \$\_GET or \$\_POST



### Example: GET method


**Insert data**

Conference: ICSE

Year: 2006

Articles:  1  2  3

```
<form method="get" action="test.php">
<table>
<tr>
<td> Conference: </td>
<td> <input type="text" name="conf" size="20"> </td>
</tr>
</table>
<tr>
<td> Year: </td>
<td> <select name="year">
<option value="2005">2005</option>
<option value="2006">2006</option>
</select>
</td>
</tr>
<tr>
<td> Articles: </td>
<td>
<input type="radio" name="number" value="1"> 1
<input type="radio" name="number" value="2" checked=""> 2
<input type="radio" name="number" value="3"> 3
</td>
</tr>
</table>
<br />
<input type="reset" value="Delete" />
<input type="submit" value="Send" />
</form>
```




### Example: GET method

- ▷ File test.php

```
<html>
<head>
<title> Result </title>
</head>
<body>
<?php
$conf = $_GET["conf"];
$anno = $_GET["year"];
$num = $_GET["number"];

echo "In the year $year you presented $num articles to the";
echo "$conf conference.";
?>
</body>
</html>
```

In the year 2006 you presented 2 articles to the ICSE conference.




### Example: calculator

/ 
 

Result: 2.5

```
<html>
<head>
<title> Calculator </title>
</head>
<body>
<form method="get" action="calculator.php">
<input type="text" size="8" maxlength="8" name="val1" value="1">
<select name="op">
<option value="sum"></option>
<option value="sub"></option>
<option value="mul"></option>
<option value="div"></option>
</select>
<input type="text" size="8" maxlength="8" name="val2" value="1">
<input type="reset" value="Delete">
<input type="submit" value="Calculate">
</form>
</body>
</html>
```






### Example: calculator

File calculator.php

```

<html>
<head><title> Result </title></head>
<body>
<?php
// Input data checking
if( !is_numeric($_REQUEST["val1"]) || !is_numeric($_REQUEST["val2"]) ){
echo "<font color='red'><h3> Not a number!</h3></font>";
return;
}

if( $_REQUEST["op"]=="div" && $_REQUEST["val2"]==0 ){
echo "<font color='red'><h3> Division by zero!</h3></font>";
return;
}
    
```




### Example: calculator

```

// Execution of the requested operation
if($_REQUEST["op"]=="sum"){
$result = $_REQUEST["val1"] + $_REQUEST["val2"];
} else if($_REQUEST["op"]=="sub"){
$result = $_REQUEST["val1"] - $_REQUEST["val2"];
} else if($_REQUEST["op"]=="mul"){
$result = $_REQUEST["val1"] * $_REQUEST["val2"];
} else if($_REQUEST["op"]=="div"){
$result = $_REQUEST["val1"] / $_REQUEST["val2"];
}

// Visualization of the result
echo "<h3> Result" . $result . "</h3>";
</body>
</html>
    
```




### Example: multiple choice

Which of the following programming languages do you know?

C     C++     Perl  
 PHP     Python     Java

You know the following 3 programming languages

- C
- Perl
- PHP




### Example: multiple choice

HTML form

- Uses the langs[] array instead of 6 variables

```

<?php
<form action="select.php" method="post">
<table width="250">
<tr>
<td><input type="checkbox" name="langs[]" value="C">C</td>
<td><input type="checkbox" name="langs[]" value="C++">C++</td>
<td><input type="checkbox" name="langs[]" value="Perl">Perl</td>
</tr>
<tr>
<td><input type="checkbox" name="langs[]" value="PHP">PHP</td>
<td><input type="checkbox" name="langs[]" value="Python">Python</td>
<td><input type="checkbox" name="langs[]" value="Java">Java</td>
</tr>
<tr>
<td colspan="3" style="text-align: center;">
<input type="submit" value="Send" />
</td>
</tr>
</table>
</form>
    
```



### Example: multiple choice

PHP script

- The \$\_REQUEST ["langs"] array contains all selected values (in this case C, Perl and PHP)

```

<?php
$languages = $_REQUEST["langs"];
$num = count($languages);
echo "<p>You know the following $num programming languages</p>";
foreach ($languages as $value ) {
echo "<i>$value </i>";
}
echo "</ul></p>";
    
```

You know the following 3 programming languages

- C
- Perl
- PHP

