**PHP COURSE**

 **By. Buttolo Marco**

**PHP** stands for **Hypertext Preprocessor** and is a powerful scripting language widely used on the web. The scripts are written in PHP file. 'Php'. PHP is a language with a syntax type C-like, and therefore anyone who knows C is very advantaged in learning PHP. Anyway it is not a particularly complicated language and as you will see shortly is very powerful. PHP is a server side language and PHP scripts reside on a web server and are called by clients (HTML pages) when the user needs to do some special processing. For example, if the user views an HTML page on your browser recording your personal information, once these data are entered, they are sent to a PHP script residing on the server which carries out the registration of such data to database on server. Graphically there is a thing of the following type:

WEB SERVER CON SCRIPT PHP.

CLIENT (WEB BROWSER)

 request

 response

 Picture 1

A web server on which to experiment, and freely downloadable, is the Wamp server (for those who have Windows) Lamp (for those with Linux). The server includes Apache (web server probably the most famous), the PHP interpreter, and MySQL. Therefore, Wamp contains everything you need to begin. Let's see a first illustrative example.

<?php

 echo "test!!";

 ?>

This file, that I have called test.php, is inserted under the document root of the server Wamp (eg: c: \ wamp \ www). So before starting the example you have to start the wamp server by clicking on the small icon present on the Desktop (which has been done by the installer).



 Picture 2

The final result is the following:



 Picture 3

The result is nothing special, but it's still a PHP script working perfectly. Usually the PHP code snippets are inserted directly into the HTML code, as shown in <html><head>:

 <title>

 Test

 </title>

</head>

<body>

 <?php

 echo "Test!";

 ?>

</body>

</html>

As you can easily notice the PHP code begins with an opening tag <? Php and a closing tag?>.

Let's take another example:

<?php

 $x=2;

 echo $x;

 ?>



 Picture 4

The echo statement displays a video message enclosed in double quotes. Each statement in PHP ends with a semicolon. Variables in PHP are used with the $ symbol before the name. PHP is untyped, and is case-sensitive, the variable named x is different from the variable named X.

We see another example:

<?php

 $x=2;

 $y=3;

 echo $x, "<br>";

 echo $y;

 ?>

The HTML tag <BR> allows the carriage return.



 Picture 5

We see another example more complicated:

<?php

 $a = 10;

 $b = 11;

 $result = $a + $b;

 echo $result;

 ?>

The code does not need to comment further. About comments, comments in PHP are placed in the following way:

//this is a rem.

We see another example more complicated:

<?php

 $numeri = array('1', '2', '3', '4', '5');

 //vector scan cycle

 for($i=0;$i<count($numeri);$i++)

 {

 echo $numeri[$i];

 }

 ?>

This example illustrates the use of the classical loop. This cycle is structured as:

 for(begin condition; quit condition; counter increment)

 istruction list.

Note the use of braces. If within a loop is a single statement the use of braces is not mandatory but is recommended as it makes the code more readable. If the instructions are more numerous than the use of braces is required. In PHP the declaration of a vector is via the keyword array, as is clearly shown in the example. You should pay particular attention to the fact that the PHP scripts have a limit of time for their execution, after which they stop. This limit is normally 30 seconds, and it is still set via the php.ini file (the initialization file). Next we see a more complex example:

<?php

 $x=2;

 if ($x==2)

 echo $x;

 else

 echo "x is not 2";

 ?>

This simple example of PHP code shows the use of the if construct. In essence, if the condition inside the if is true, it executes the statement contained in it (echo $ x), otherwise it is false if you run another statement (echo "x is not worth 2"). Therefore, in general, it has:

 if(condition)

 action 1

 else

 action2

1 action is performed only if the condition is met, otherwise the action is performed 2.

<?php

 $x = 1;

 while ($x <= 10)

 {

 $ris = 5 \* $x;

 print("5 \* $x = $ris<br>");

 $x++;

 }

 ?>

This example illustrates the use of the while loop. As long as the loop condition is verified statements are executed inside. The increase of one unit of the value of a variable is obtained by + +. Similarly we have for the **do while** loop.

<?php

 $x = 4;

 do

 {

 $ris = 5 \* $x;

 print("5 \* $x = $ris<br>");

 $x++;

 } while ($x <= 10)

 ?>

Note the use of the **print** which allows you to view video messages and variable values ​​(such as the echo statement). The difference between echo and print is that print allows you to format the variables included in the string, while the echo statement not. Suppose now to have two strings. Let's see how it performs string concatenation:

<?php

 $str1="Marco";

 $str2=" Buttolo";

 $str = $str1.$str2;

 print($str);

 ?>

The result is the following:



 Picture 6

Therefore, the operator '.' Allows the concatenation of two strings. In the previous examples we have seen other operators as in for example the sum (+). There are also clearly the difference (-), multiplication (\*), division (/), and so on. To increase the value of a variable of a unit you can follow two paths:

 $variabile=$variabile+1;

Or:

 $variabile++;

Similarly, it has for the subtraction of a unit:

 $variabile=$variabile-1;

 $variabile--;

To check if the value of two variables is equal using the == operator, and to verify if these values ​​are different you use it! =. Note the syntactic similarity that exists between the PHP language and the language C. In fact, PHP is unlinguaggio C-like.

Even in PHP there are logical operators. Let us look briefly:

* AND (&&)
* OR (||)
* NOT (!)

We see an example:

<?php

 $x=1;

 $y=2;

 if ($x==1 && $y==2)

 echo "Correct values!!";

 else if ($x==1 || $y==2)

 echo "A value not correct!!!";

 ?>

As shown in the example is run the first echo only if both conditions are true (the AND works well), while the second echo is performed only if at least one condition is met.

Suppose now that we have a loop, and you want to get out of this cycle to the fifth iteration. The instruction to do this is the break statement, as shown below:

<?php

 for($index=0;$index<10;$index++)

 {

 if($index==4)

 break;

 }

 ?>

The instruction instead allows you to jump to the next iteration is the continue statement that is shown in the following example:

<?php

 for($index=0;$index<10;$index++)

 {

 if($index==4)

 continue;

 }

 ?>

The PHP5, like most other languages ​​is procedural, ie you can use the functions, appropriately called, which perform some specific tasks. These functions can be built into the language (the default) or can be created. To invoke a particular function, just write the name with any parameters:

 function\_name (par1, par2,…)

Let us briefly discuss some elementary functions default. We analyze the empty function which checks whether a variable is empty or not (if èdefinita or not or if it contains a 0 ...). Let's see the following example:

<?php

 $val2 = empty($val1); /\*$val1 non è ancora definita, quindi $val2 sarà vero\*/

 echo $val2;

 ?>

Another important function is is\_int () which returns a boolean (true if the parameter that is passed is an integer, false otherwise). Similarly you have to check for real, a string, an array, a numeric field and so on. Let's see about the following example:

<?php

 /\*I cotnrol if variable is numeric\*/

 $x=is\_numeric(3);

 echo $x;

 /\*I control if variable is float\*/

 $y=is\_float(3.3);

 echo $y;

 /\*I control if variable is string\*/

 $z=is\_string("ss");

 echo $z;

 /\*I control if variable is integer\*/

 $w=is\_int("a");

 echo $w;

 /\*I control if variable is array\*/

 $a=is\_array(1);

 echo $a;

 ?>

Finally, you can also get to fly the type possessed by a variable, as shown below, using the **gettype ()**:

<?php

 $x=3;

 echo gettype($x);

 ?>

Even PHP5, like other types of languages, it is able to work with strings. And 'well known that a string is a sequence of characters. Examples of a string is the following:

 “light”

PHP5 provides the programmer functions for string manipulation. We see a short Carello of the most used. For example if we want to display the length of a string use strlen as shown below:

<?php

 $str="light";

 echo strlen($str);

 ?>

Appears on the screen the number 4 (4 characters of the string). Now suppose we want to extract a substring from main string. For example, suppose we have available the following string:

 str=”light of my eyes”;

and want to extract the substring "light". To do this you use the substr function as an argument the original string, which character to start the extrapolation, and how many characters you want to extract.

<?php

 $str="light of my eyes";

 $str2=substr($str,0,4);

 echo $str2;

 ?>

It 'important to note that the characters are counted starting from scratch. If you instead want to replace parts of string using str\_replace function as shown below:

<?php

 $str="light of my eyes";

 $str2=str\_replace("light","dark",$str);

 echo $str2;

 ?>

As is easily seen, the substring "light" will be replaced with the substring "dark." If you now want to get the position at which the substring "my" lies in the original string is simply use the function **strpos ()** as shown below:

<?php

 $str="light of my eyes";

 $str2=str\_replace("light","dark",$str);

 $pos=strpos($str, "my");

 echo $pos;

 ?>

Suppose now that we have available the following string:

 str=”hello\_world”

we want to break up the string into two different strings "hello" and "world" according to the special character **underscore** (\_). This operation is called a split in computer science. Usually there is a function to allow the **splitting** of a string. This function is the function **split()**. Even in PHP function that was present but has been deprecated in PHP5 and replaced by the explode function as the following example shows:

<?php

 $str="hello\_world";

 $vett= explode('\_', $str);

 echo $vett[0];

 echo $vett[1];

 ?>

As the example shows, the result is a vector of strings. There are other functions for working with strings. For further information the reader is referred to the various manuals more approfoniti available on the internet. Anyway some functions very much used, in addition to those just mentioned, are:

* **trim()** which can remove the white spaces in the string.
* **Ltrim()** that allows you to remove whitespace at the beginning of the string.
* **Rtim()** which eliminates the white space at the end of the string.

**Strtolower()** which allows you to put in all lowercase characters in the string.

* **Strtoupper()** which allows you to put in all uppercase characters in the string.

Similarly to what we saw with strings, esisotno functions to work on arrays. Let's see some cool features through examples:

<?php

 $vett = array('x', 'y', 'z');

 echo count($vett);

 ?>

This simple example shows how to use the count function to count the number of elements in the array.

<?php

 $vett=array('a', 'b', 'c');

 $arr1 = array\_reverse($vett);

 for($i=0; $i<count($vett);$i++)

 echo $arr1[$i];

 ?>

The array\_reverse function allows you to invert the order of the elements of the vector. For example, if you have the following vector:

 $x=[‘a’, ‘b’, ‘c’];

with this function you get a new vector of the following type:

 $y=[‘c’, ‘b’, ‘a’];

To order a carrier utilizzainvece the sort function as shown below:

<?php

 $vett=array(3,2,1);

 $vett1=sort($vett);

 for($i=0; $i<count($vett1);$i++)

 echo $vett1[$i];

 ?>

To search for a particular element in an array using **in\_array ()** function, as shown below:

<?php

 $vett=array(3,2,1);

 $trovato=in\_array(1, $vett);

 echo $trovato;

 ?>

Please pay attention that the function returns 1 (true) if it finds the key, otherwise it returns 0 (false). If you instead want to merge the values ​​in an array on a single string, you must perform the opposite operation of splitting that is, you have to use the opposite function dell'explode. This function is the implodes and is used in the following way:

<?php

 $vett=array(3,2,1);

 $str=implode(' ', $vett);

 echo $str;

 ?>

As you can easily notice, the character "small space" ('') is used to separate the various substrings (array elements) of the string. Other useful functions are:

* **Array\_pop**, that allows you to extract the last element of the array.
* **Array\_push**, that allows you to queue at the bottom of the new elements in the array.
* **Array\_merge**, that allows you to merge the elements of two arrays.
* **Array\_shift**, that allows you to extract the first element of the array, and then all the other shiftware.

Let us now conclude this brief overview of the main features offered by PHP with the management of the dates. It 'important to remember that the management of the dates in PHP is through the use of the classic Unix timestamp. The timestamp is an integer that represents the number of seconds since January 1, 1970. The function that allows you to extract the timestamp is the current time function, which is shown below:

<?php

 $time=time();

 echo $time;

 ?>

Since the timestamp, to get today's date is sufficient to use the **date** function:

<?php

 $time=time();

 $data = date('d M y - H:i', $time);

 echo $data;

 ?>

This function has some of the parameters described below:

**Y** year on 4 chipers

**y** year on 2 chipers

**N** numeric month (1-12)

 **m** numeric month on 2 chipers (01-12)

**F string month** ('January' - 'December')

 **M** mese testuale su 3 lettere ('Jan' - 'Dec')

**d** day of the month on 2 chipers (01-31)

**J day of the month** (1-31)

**W day of the week, numeric** (0=dom, 6=sab)

**l day of the week, string** ('Sunday' - 'Saturday' )

 **D day of the week on 3 chipers** ('Sun' - 'Sat')

**H Hour on 2 chipers** (00-23) **G**ora (0-23)

**i minutes on 2 chipers** (00-59)

**s seconds on 2 chipers** (00-59)

**To carry out an audit on the date given is appropriate to use the checkdate.**

**Another basic construct is the require() statement. This statement includes and evaluates the specified file. Another instruction is include identical, with the only difference that require stops the execution of the page if the specified file is missing and includes not allow it. Let's see an example:**

<?php

 require 'file\_require.php';

 echo $x;

 ?>

**file\_require.php and contains the following code fragment:**

**<?php**

 **$x=34;**

 **?>**

**The constructs and the instructions set out so far are those of the basic language. Clearly, there are more detailed guides on the market which make reference safely. Let us now briefly describe the concept of function. Consider the following example:**

function fun ($arg\_1, $arg\_2, ..., $arg\_n)

{

echo "Example of function...\n";

return $retval;

}

The above example shows a generic function named function that has as input parameters ' n ' arguments (' n ' variables). A function is something that has a certain number of inputs, processes them , and produces certain outputs . This way of programming (procedural programming) typical of modern programming languages ​​makes it possible to make the code readable and slender as it allows to split the operations performed by the script among the most called functions appropriately within the portion of the main code of the program. By default the variables are passed to functions by **value**. This means that, once the execution of the function , the changed value from the function of the variable is lost . Moving on the variable by **reference** you can not lose that value. The passage of the variable reference occurs before yielding to the variable name the special character ' & ' , as shown below:

Function fun(&$string)

{

$string .= ’xxx.’;

}

$str = ’stringa, ’;

fun($str);

echo $str; // outputs ’stringa, xxx.’

**At this point we can make a step forward and begin to get into the object-oriented programming since PHP is an object oriented language. We begin by defining the basic construct present in the object-oriented programming: the class. A class is a collection of variables and methods to operate on such variables. An object is an instance of the class. In order to work with a class then you must first instantiate it and then you can make using the methods of the variables with the usual dot notation. Let's see the following example:**

<?php

class test

{

function f1()

{

echo "Test";

}

}

$obj = new test;

$obj->f1();

?>

**As this example shows, we have created a class named test in which there is a function (method) called f1. This function prints the string Test Subsequently, an object is created ($ obj) which is used to perform the function f1. If you test in the class had been present for example a variable named $ var\_test, then via Object $ obj you could access the contents of this variable in the following way:**

 **$obj.$var\_test**

**Often, especially for those who have worked hard in the field of object-oriented programming, you are forced to build classes derived from other classes. Inheritance is an important feature of object-oriented programming. For example, it is possible to create a generic class named vehicle that contains all the features of a generic vehicle, and then create a class that derives naturally car class vehicle (seen that the car is a particular type of vehicle) and which contains the generic properties of an automobile. In PHP the eredity is achieved by the extends keyword. See the following example about:**

**<?php**

 **class car extends vehicle**

 **{**

 **var $motore;**

 **var $targa;**

 **var $colore;**

 **}**

 **?>**

**In this example the car class derives from the class vehicle. Closed the section on classes analyze now the most important part of PHP. Database management and interaction with HTML files. Let's start with the latter.**

**An HTML form is basically a set of input elements (textbox, checkbox, radiobox, ...). In a nutshell in the form allows the user to enter certain information. Think, for example, the registration screens for the online purchase of a particular product. Once you have entered the data, and after clicking on the "Submit" button is sent to an application on a server (usually a script) that processes them, and returns a result. The URL (address) of this script is specified in the action attribute of the HTML tag form. For more assistance, I invite the reader to read the HTML manual free download on my own site. See the following example about:**

<form action="elabora.php" method="get">

<input type="text" name="campione">

<input type="submit" name="bInvia" value="Send values">

</form>

**The PHP script is called elabora.php and the data is sent using the GET method. For the transmission of data to the script there are basically two methods:**

1. **method post**
2. **method get**

**The POST method allows you to send the form data to the standard input of the application addressed and are therefore not visible in the URL. Very important thing: there are no limits on the amount of data to send. With the GET method, the data is appended to the address and, therefore, are clearly visible in the same. For this reason the amount of data that can be sent with the GET method is limited to 256 characters of the URL itself. For the PHP interpreter to use a method rather than the other makes no difference. Interprets and stores them in global variables within the script.**

**The following examples show, the PHP side, the management of variable with the GET method and the POST method.**

**HTML SIDE:**

<form action="lettura.php" method="post">

<input type="text" name="nome">

<input type="checkbox" name="nuovo" value="si">

<input type="submit" name="submit" value="invia">

</form>

**PHP SIDE:**

**<?php**

 **Var $test = $\_POST[‘nome’];**

 **echo $test;**

**?>**

**$ \_POST Should be seen as an array.**

**Another important concept in PHP sessions. And ' well known that the protocol widely used for browsing on the internet is the HTTP protocol. This protocol , however, were less that is stateless . This means that when the user leaves the web page transactions that have occurred will be lost. To solve the problem have been invented cookies , or normal text file in which to save the session information . The working group of PHP, to solve this problem and simplify the issue without resorting to cookies too, has introduced an intelligent session management. In a nutshell, when the function is called start\_session or session\_register the session data is stored in the memory of the session. The path where resides the session memory is specified in the php.ini directive session.save\_path in. When a session is started it is assigned a session ID . Therefore the support of PHP sessions is to preserve certain data across subsequent accesses. This concept is also very important to speed up access to pages. A visitor accessing to our site will be assigned a unique id that is the id of the session.**

**As of version 4.1.0, PHP $ \_SESSION is available as a global variable just like $ \_GET and $ \_POST . Because the $ \_SESSION is used need to edit the php.ini file in the WINDOWS directory by setting = 1 session.auto\_start and creating , if there is a temp directory where the session variables are recorded . We see the following illustrative example**:

<?php

session\_start();

if (!session\_is\_registered('contatore'))

 {

 session\_register('contatore');

 $\_SESSION[contatore] = 1;

}

else {

 $\_SESSION[contatore]++;

}

?>

<html>

 <head>

 <title>Sessioni</title>

 </head>

 <body>

 <?php

 echo "<h2>Page visited $\_SESSION[contatore] times.</h2>

 <A HREF=\"session\_cookie.php\">clicca qui</A>";

 ?>

 </body>

</html>

**The other important concept is the concept of database. The ability to interact with databases is one of the most interesting features of PHP. Relational databases are the means by definition used to store data of any size. PHP provides us with the opportunity to connect with a large number of databases (MySQL, Oracle, ....) First you need to connect to the database in question. This in PHP is very simple and is shown below:**

 **mysql\_connect(server, user, password);**

The connection is established through the mysql\_connect function. Which requires the following parameters:

* + **server** : l’URL of database
	+ **utente** : access name
	+ **password** : password of user

**Once the connection is established, you can make the selection of the database on which we intend to work. For this we use the mysql\_select\_db function that takes the following parameters**:

* + **nomedb**: database name
	+ **connessione**: ID of connection (that is what we got from mysql\_connect)

The sintax is the following:

 mysql\_select\_db(nomedb,connection);

This function returns a Boolean value based on the outcome of the transaction. At this point you can perform queries (queries) on the selected db. This operation can carry it through mysql\_query whose parameters are

* + **query**:query to execute
	+ **connessione**: id of connection

This function also returns a value, but for which we have to distinguish two possibilities with respect to the type of query that we launched:

* + **Query selection** (SELECT, SHOW, EXPLAIN, DESCRIBE), the function returns an identifier of the result (for example another variable of type resource), which will be useful later, if the query is successful, but if MySQL has detected errors, the function returns FALSE
	+ **Update** query (INSERT, UPDATE, DELETE), the function returns a Boolean value in each case, to indicate whether the execution is successful or not

At this point it is clear that it is also necessary to verify the correct execution of the query. If you want to know how many rows were returned by a read operation can, for example, use **mysql\_num\_rows ()**. If we run an update query (INSERT, UPDATE, DELETE) and want to know how many rows were changed, we can use **mysql\_affected\_rows** (connection), that returns the number of rows that have changed since the last update query.

After completion of the various operations on the db you can close the connection to the same. This operation is performed by the function **mysql\_close ()** that has as its only parameter the id of the connection. But be careful: this function is used very little as PHP is concerned only by closing the connection to the database once it has finished executing the script. The following script shows a complete example of access to a database in PHP.

<?php

 $dbUtente="root";

 $passwd="";

 $dbNome="test";

 $dbHost="localhost";

 //open connection

 if(!($conn=mysql\_connect($dbHost, $dbUtente, $passwd)))

 {

 echo "Error connection!!";

 }

 //select current db

 if(!mysql\_select\_db($dbNome, $conn))

 {

 echo "Error db!!";

 }

 //execute query

 $query="select \* from Test";

 $result=mysql\_query($query, $conn);

 echo $result;

 //close connection

 mysql\_close($conn);

 ?>