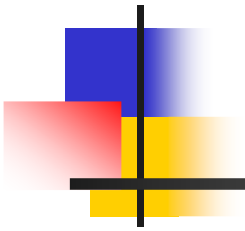
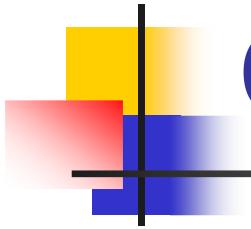
Linguaggi

Corso M-Z - Laurea in Ingegneria Informatica
A.A. 2009-2010

Alessandro Longheu
http://www.diit.unict.it/users/alongheu
alessandro.longheu@diit.unict.it

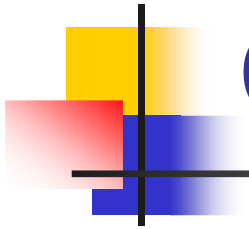


Il linguaggio PHP



Generalità sul PHP

- PHP, che significa "PHP: Hypertext Preprocessor", è un **linguaggio di scripting general-purpose Open Source** molto utilizzato, è specialmente indicato per lo sviluppo Web e può essere integrato nell'HTML.
- La sua sintassi è basata su quella di C, Java e Perl, ed è molto semplice da imparare.
- **Story:** PHP succeeds an older product, named PHP/FI. PHP/FI was created by Rasmus Lerdorf in 1995, initially as a simple set of Perl scripts for tracking accesses to his online resume. He named this set of scripts 'Personal Home Page Tools'. As more functionality was required, Rasmus wrote a much larger C implementation, which was able to communicate with databases, and enabled users to develop simple dynamic Web applications. PHP/FI, which stood for Personal Home Page / Forms Interpreter, included some of the basic functionality of PHP as we know it today. It had Perl-like variables, automatic interpretation of form variables and HTML embedded syntax.

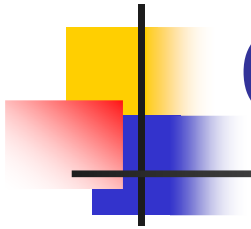


Generalità sul PHP

- Un primo **esempio** di utilizzo tipico di codice in php:

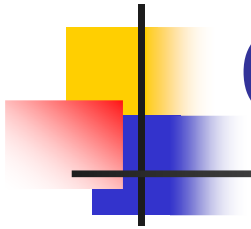
```
<html>
<head>
<title>Esempio</title>
</head>
<body>
<?php echo "Ciao, sono uno script PHP!"; ?>
</body>
</html>
```

- questo esempio è differente da uno script scritto in altri linguaggi tipo Perl o C: invece di scrivere un programma con parecchi comandi per produrre HTML, si scrive in HTML con qualche comando immerso per ottenere dei risultati (in questo semplice esempio, la visualizzazione di una frase). Il codice PHP è delimitato da speciali start ed end tag che ne indicano l'inizio e la fine e che consentono di passare dal modo HTML al modo PHP.
- Ciò che distingue PHP da altri linguaggi di scripting del tipo client-side (JavaScript) è che il codice viene eseguito nel server.



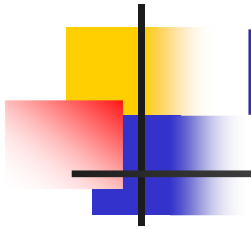
Generalità sul PHP

- Esistono **tre campi principali** in cui vengono usati gli scripts PHP:
  - Lo scripting server-side. Questo è il campo più tradizionale ed il maggiore obiettivo del PHP. Per fare questo lavoro occorrono tre cose. Il parser PHP (CGI o server module), un webserver ed un browser web. Occorre avviare il server web con un'installazione di PHP attiva. Si può accedere all'output del programma PHP con un browser web e vedere la pagina PHP tramite il server. Tutto ciò può essere attivato sul pc di casa se si desidera semplicemente provare la programmazione PHP.
  - Lo scripting su riga di comando. Si può creare uno script PHP da usare senza alcun server o browser. Per usarlo in questo modo, l'unica cosa necessaria è un parser PHP. Questo tipo di utilizzo è ideale per gli scripts eseguiti con cron (sui sistemi *nix o Linux) oppure il Task Scheduler (su Windows).
  - Scrittura di applicazioni desktop, anche se PHP non è il linguaggio più adatto per scrivere tali applicazioni.



Generalità sul PHP

- PHP può essere usato su tutti i principali sistemi operativi
- si può anche scegliere una **programmazione procedurale oppure orientata agli oggetti**, o una combinazione di entrambe.
- Con PHP non si è limitati soltanto ad un output in HTML. Le possibilità di PHP, infatti, includono l'abilità di generare immagini, files PDF e perfino filmati Flash al volo (utilizzando libswf e Ming).
- Una delle caratteristiche più importanti e significative di PHP è la possibilità di **supportare una completa gamma di database**; PHP supporta anche ODBC, lo standard di collegamento con i database.
- PHP fa anche da supporto per **dialogare con altri servizi** utilizzando i protocolli del tipo LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (in Windows) e innumerevoli altri. Si possono aprire sockets ed interagire usando qualsiasi altro protocollo.
- PHP supporta l'installazione dei JavaObjects e l'utilizzo di questi come oggetti PHP in modo trasparente. Si può anche usare l'estensione CORBA per accedere ad oggetti remoti.



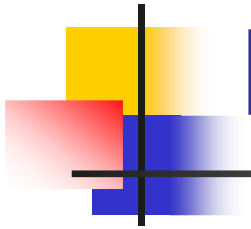
Delimitazione del PHP

- La modalità di operare del php spesso si interfaccia con Html:
- A partire dal file php, eseguito dal lato server:

```
<html>
<head>
</head>
<body>
<?php echo "Hello World!<p>"; ?>
</body>
</html>
```

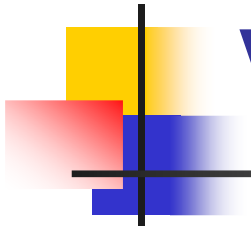
- L'output di questo script sarà il file Html, visualizzabile dal client:

```
<html>
<head>
</head>
<body>
Hello World!<p>
</body>
</html>
```



Delimitazione del PHP

- Quando il PHP inizia a esaminare un file, cerca i **tag di apertura e di chiusura**, che indicano dove iniziare e terminare l'interpretazione del codice. Questa tecnica permette al PHP di essere incorporato in tutte le tipologie di documenti, ed ogni cosa esterna a tali tag viene ignorata, ovvero quando il PHP trova il tag di chiusura ?>, inizia a visualizzare tutto ciò che incontra sino a quando non si raggiunge un'altro tag di apertura.
- Esistono **4 set di tag** che possono essere utilizzati per delimitare blocchi di codice PHP. Soltanto due di questi (<?php. . .?> e <script language="php">. . .</script>) sono sempre disponibili. Gli altri due sono i tag brevi e i tag stile ASP e possono essere attivati o disattivati tramite il file di configurazione php.ini. Sebbene i tag brevi o quelli in stile ASP possano essere pratici, questi sono meno portabili e, in generale, sconsigliati.
- il PHP richiede che le istruzioni siano chiuse dal punto e virgola al termine di ogni istruzione.
- Il PHP supporta i commenti stile C e stile PERL (con '#').



Variabili in PHP

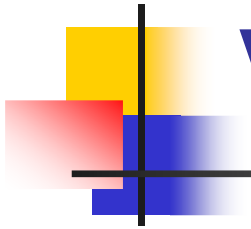
- Le **variabili** in PHP sono identificate da un nome case sensitive preceduto da '$'
- Le variabili non hanno bisogno di essere dichiarate, la dichiarazione avviene quando alla variabile viene assegnato un valore, quindi una variabile che contiene un numero può essere interpretata sia come numero che come testo. Qualche esempio:

  $variabile = "1";
  $variabile = 1;

  le variabili sono valide in entrambi i casi

  echo 234 + "145";
- In questo caso il PHP converte la stringa "145" in un numero intero, lo somma all'intero 234, converte il risultato in stringa e lo visualizza sulla pagina tramite la funzione echo().
- Il **casting** si fa direttamente o con la funzione settype():

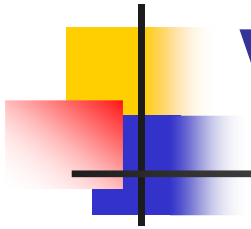
  $variabile = (string) 234;
  $variabile = 12;
  settype($variabile, "double");



Variabili in PHP

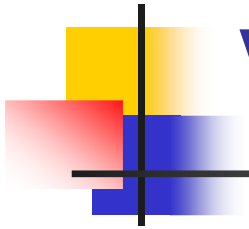
- In PHP 3, variables are always assigned by value, As of PHP 4, PHP offers **assign by reference**. This means that the new variable simply references (in other words, "becomes an alias for" or "points to") the original variable. Changes to the new variable affect the original, and vice versa. To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable).

```
<?php
$foo = 'Bob'; // Assign the value 'Bob' to $foo
$bar = &$foo; // Reference $foo via $bar.
$bar = "My name is $bar"; // Alter $bar...
echo $bar;
echo $foo; // $foo is altered too. ?
$bar = &(24 * 7); // Invalid; references an unnamed expression.
?>
```



Variabili in PHP

- PHP supports **eight primitive types**.
- Four scalar types: boolean, integer, float (aka 'double'), string
- Two compound types: array, object
- Two special types: resource, NULL
- In particolare:
- To specify a **boolean** literal, use either TRUE or FALSE. Both are case-insensitive: *<?php $foo = True; // assign the value TRUE to $foo ?>*
- **Integer** literals: *<?php $a = 1234; // decimal number $a = -123; // a negative number $a = 0123; // octal number (equivalent to 83 decimal) $a = 0x1A; // hexadecimal number (equivalent to 26 decimal) ?>*; The size of an integer is platform-dependent, although a maximum value of about two billion is the usual value (that's 32 bits signed). PHP does not support unsigned integers.
- **Floating point** numbers (AKA "floats", "doubles" or "real numbers") can be specified using any of the following syntaxes: *<?php $a = 1.234; $b = 1.2e3; $c = 7E-10; ?>*; The size of a float is platform-dependent, although a maximum of ~1.8e308 with a precision of roughly 14 decimal digits is a common value (that's 64 bit IEEE format).



Variabili in PHP

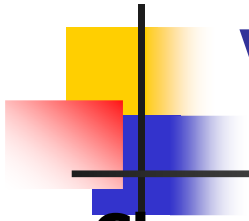
- A **string** is series of characters. In PHP, a character is the same as a byte, that is, there are exactly 256 different characters possible. This also implies that PHP has no native support of Unicode;
- It is no problem for a string to become very large. There is no practical bound to the size of strings imposed by PHP, so there is no reason at all to worry about long strings.
- A string literal can be specified in **three different ways**: single quoted, double quoted, heredoc syntax
- **Single quote** is the easiest way, but variables and escape sequences for special characters will *not* be expanded:

```
<?php echo 'this is a simple string';
echo 'Arnold once said: "I\'ll be back"';
// Outputs: Arnold once said: "I'll be back"
echo 'You deleted C:\\*.*?';
// Outputs: You deleted C:\*.*?
'This will not expand: \n a newline';
// Outputs: This will not expand: \n a newline echo
echo 'Variables do not $expand $either';
// Outputs: Variables do not $expand $either ?>
```

Variabili in PHP

- If the string is enclosed in **double-quotes** ("), PHP understands more escape sequences for special characters, as \n, \t, \\, \$, \”
- Again, if you try to escape any other character, the backslash will be printed too, but the most important feature of double-quoted strings is the fact that variable names will be expanded.
- Another way to delimit strings is by using **heredoc syntax** ("<<<"). One should provide an identifier after <<<, then the string, and then the same identifier to close the quotation. Heredoc text behaves just like a double-quoted string, without the double-quotes. This means that you do not need to escape quotes in your here docs, but you can still use the escape codes listed above. Variables are expanded

```
<?php
$str = <<<EOD
Example of string spanning multiple lines
using heredoc syntax.
EOD;
?>
```



Variabili in PHP

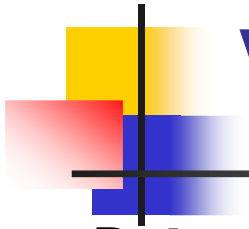
- **Characters** within strings may be accessed and modified by specifying the zero-based offset of the desired character after the string in curly braces.

```
<?php
$str = 'This is a test.';
$first = $str{0}; // Get the first character of a string
$last = $str{strlen($str)-1}; // Get the last character of a string.
$str{strlen($str)-1} = 'e'; // Modify the last character of a string ?>
```

- Ci sono due **operatori di stringa**. Il primo è l'operatore di concatenazione ('.'), che restituisce la concatenazione dei suoi argomenti di destra e di sinistra. Il secondo è l'operatore di assegnazione concatenata ('.='), che accoda l'argomento sul lato destro all'argomento sul lato sinistro.

```
<?php
$a = "Ciao "; $b = $a . "Mondo!"; // ora $b contiene "Ciao Mondo!"
$a .= "Mondo!"; // ora $a contiene "Ciao Mondo!" ?>
```

- There are a lot of useful functions, e.g. for advanced find&replacing with ER, functions for URL-strings, and to encrypt/decrypt.



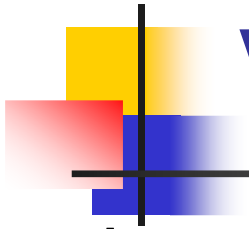
Variabili in PHP

- **Determining a variable's type:**

```
<?php
$bool = TRUE; // a boolean
$str = "foo"; // a string
$int = 12; // an integer
echo gettype($bool); // prints out "boolean"
echo gettype($str); // prints out "string"
// If this is an integer, increment it by four
if (is_int($int)) { $int += 4; }
// If $bool is a string, print it out
// (does not print out anything)
if (is_string($bool)) { echo "String: $bool"; } ?>
```

- **Automatic type conversion:**

```
<?php
$foo = "0"; // $foo is string (ASCII 48)
$foo += 2; // $foo is now an integer (2)
$foo = $foo + 1.3; // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs"; // $foo is integer (15) ?>
```



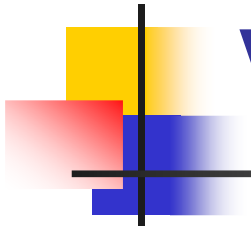
Variabili in PHP

- An **array** in PHP is actually an ordered map. A map is a type that maps values to keys. This type is optimized in several ways, so you can use it as a real array, or a list (vector), hashtable (which is an implementation of a map), dictionary, collection, stack, queue and probably more. Because you can have another PHP array as a value, you can also quite easily simulate trees.

```
<?php
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // bar
echo $arr[12]; // 1
?>
```

- A key may be an integer or a string, a value can be of any type:

```
<?php $arr = array("anarray" => array(6 => 5, 13 => 9, "a" =>
   42));
echo $arr["anarray"][6]; // 5
echo $arr["anarray"][13]; // 9
echo $arr["anarray"]["a"]; // 42
?>
```



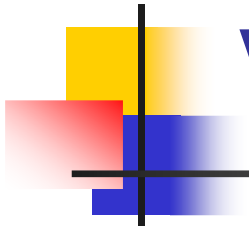
Variabili in PHP

- You can also modify an existing array by explicitly setting values

```
<?php $arr = array(5 => 1, 12 => 2);
$arr[] = 56; // same as $arr[13] = 56; at this point of the script
$arr["x"] = 42; // This adds a new element with key "x"
unset($arr[5]); // This removes the element from the array
unset($arr); // This deletes the whole array
?>
```

- The array type in PHP is very versatile:

```
$switching = array(         10, // key = 0
                    5    =>  6,
                    3    =>  7,
                    'a'  =>  4,
                             11, // key = 6 (maximum of integer-indices was 5)
                    '8'  =>  2, // key = 8 (integer!)
                    '02' => 77, // key = '02'
                    0    => 12  // the value 10 will be overwritten by 12
                  );
```

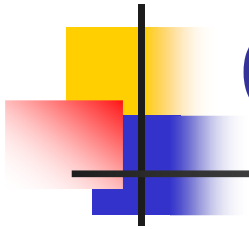


Variabili in PHP

```
<?php
$colors = array('red', 'blue', 'green', 'yellow');
foreach ($colors as $color)
{ echo "Do you like $color?\n"; }
?>
```

- Arrays are ordered. You can also change the order using various sorting functions.
- array assignment always involves value copying. You need to use the reference operator to copy an array by reference.

```
<?php
$arr1 = array(2, 3);
$arr2 = $arr1;
$arr2[] = 4; // $arr2 is changed, // $arr1 is still array(2, 3)
$arr3 = &$arr1; $arr3[] = 4; // now $arr1 and $arr3 are the
   same
?>
```



Classi ed Oggetti in PHP

- Una **classe** è una collezione di variabili e funzioni che utilizzano queste variabili. Una classe si definisce usando la seguente sintassi:

```
<?php
class Cart {
var $items; // Articoli nel carrello
// Aggiunge $num articoli di $artnr nel carrello
function add_item ($artnr, $num) { $this->items[$artnr] += $num; }
// Prende $num articoli di $artnr e li rimuove dal carrello
function remove_item ($artnr, $num) { if ($this->items[$artnr] > $num)
    {
$this->items[$artnr] -= $num; return true;
} elseif ($this->items[$artnr] == $num) {
unset($this->items[$artnr]); return true;
} else { return false; } } }
?>
```

- Il codice definisce una classe chiamata Cart composta da un array associativo che archivia gli articoli nel carrello e due funzioni per aggiungere e rimuovere gli articoli dal carrello stesso.

Classi ed Oggetti in PHP

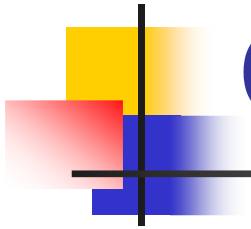
- **Costruttore** (in PHP5, __construct, prima il nome è quello della classe):

```
class Cart { var $todays_date;
var $name;
var $items = array("VCR", "TV");
function Cart() {
$this->todays_date = date("Y-m-d");
$this->name = $GLOBALS['firstname']; /* etc ... */ }
```

- Per creare una variabile oggetto si usa **l'operatore new**.

```
<?php $cart = new Cart;
$cart->add_item("10", 1);
$another_cart = new Cart;
$another_cart->add_item("0815", 3);
?>
```

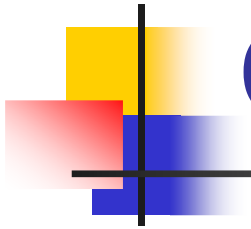
- Supporto per **ereditarietà**: una classe estesa o derivata ha tutte le variabili e le funzioni della classe di base più tutto ciò che viene aggiunto dall'estensione. Una sottoclasse può ridefinire variabili e funzioni di una classe madre. L'eredità multipla non è supportata. Le classi si estendono usando la parola chiave 'extends'.



Classi ed Oggetti in PHP

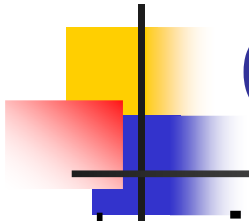
```
<?php class A {
function A() { echo "Sono il costruttore di A.<br />\n"; }
function B() { echo "Sono una normale funzione di nome
   B.<br>\n";
echo "Non sono il costruttore di A.<br>\n"; } }
class B extends A { }
// This will call B() as a constructor. $b = new B; ?>
```

- La funzione B() della classe A si **trasformerà improvvisamente in un costruttore per la classe B**, anche se questo non era previsto. PHP 4 non si preoccupa infatti se la funzione è stata definita nella classe B o se è stata ereditata.
- A volte è utile riferirsi alle funzioni ed alle variabili di classi base o **riferirsi alle funzioni di classi senza istanziarle**. L'operatore :: è usato per questi scopi, ad esempio A::example() invoca la funzione example senza che esista alcuna istanza di A
- Esiste la parola chiave **parent**, equivalente di super in Java, e la **self**, indicata anche con this nelle versioni precedenti la 5



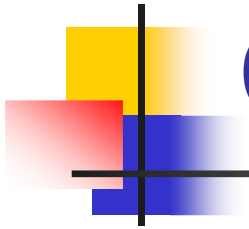
Classi ed Oggetti in PHP

- E' supportata la **serializzazione**; Se si utilizzano le sessioni e session_register() per registrare oggetti, questi oggetti vengono serializzati automaticamente alla fine di ogni pagina PHP e sono deserializzate automaticamente su ogni pagina della sessione. Ciò significa che gli oggetti possono mostrarsi in ogni pagina e che sono parte integrante della sessione.
- Si suggerisce tuttavia vivamente di includere le definizioni delle classi degli oggetti registrati su tutte le pagine, anche se le classi non sono usate su tutte le pagine. Se un oggetto viene deserializzato senza la relativa definizione della classe, perderà l'associazione ad essa e si trasformerà in in un oggetto della classe stdClass senza nessuna funzione disponibile, diventando inutile.
- Le funzioni magiche __*sleep* e __*wakeup* sono invocate la prima prima della serializzazione e la seconda dopo la deserializzazione, al fine di potere garantire consistenza all'oggetto (ad esempio chiudendo e riaprendo le connessioni ai db eventualmente presenti)
- Esistono altre funzioni magiche, ad esempio __toString



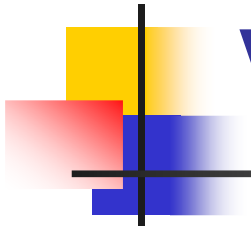
Classi ed Oggetti in PHP

- La **visibilità** di classi e attributi/metodi può essere impostata tramite private, protected, public, parole chiave con identico significato rispetto a Java
- Sono supportati metodi e variabili **statiche**
- PHP 5 introduces **abstract** classes and methods. It is not allowed to create an instance of a class that has been defined as abstract. Any class that contains at least one abstract method must also be abstract. Methods defined as abstract simply declare the method's signature they cannot define the implementation. When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child; additionally, these methods must be defined with the same (or weaker) visibility.
- Object **interfaces** allow you to create code which specifies which methods a class must implement, without any implementation. Interfaces are defined using *interface*, all methods declared in an interface must be public, and to implement it, the implements operator is used. Classes may implement more than one interface if desired by separating each interface with a comma.



Classi ed Oggetti in PHP

- In PHP 5, **object comparison** uses two comparison operator:
  - (==), where two object instances are equal if they have the same attributes and values, and are instances of the same class.
  - (===), identity operator, implies that object variables are identical if and only if they refer to the same instance of the same class.
- PHP 5 has an **exception model** similar to that of other programming languages. An exception can be thrown, try and caught within PHP. A Try block must include at least one catch block. Multiple catch blocks can be used to catch different classtypes; execution will continue after that last catch block defined in sequence. Exceptions can be thrown within catch blocks.
- When an exception is thrown, code following the statement will not be executed and PHP will attempt to find the first matching catch block. If an exception is not caught a PHP Fatal Error will be issued with an Uncaught Exception message, unless there has been a handler defined with set_exception_handler().

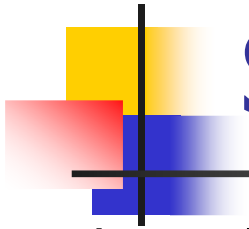


Variabili in PHP

- Oltre array ed oggetti, esistono due **tipi speciali**:
- resource, che può rappresentare una risorsa, ovvero un file bzip2, un riferimento COM, un file pdf, un link ad un database, uno stream ftp, un link ad un server POP o IMAP, una connessione LDAP ecc.
- The special NULL value represents that a variable has no value.
- PHP provides a large number of **predefined variables** to any script which it runs. Many of these variables, however, cannot be fully documented as they are dependent upon which server is running, the version and setup of the server, and other factors. Some of these variables will not be available when PHP is run on the command line.

Variabili predefinite in PHP

- $GLOBALS - Contains a reference to every variable which is currently available within the global scope of the script. The keys of this array are the names of the global variables. $GLOBALS has existed since PHP 3.
- $_SERVER - Variables set by the web server or otherwise directly related to the execution environment of the current script. Analogous to the old $HTTP_SERVER_VARS array (still available, but deprecated).
- $_GET - Variables provided to the script via URL query string. Analogous to the old $HTTP_GET_VARS array (deprecated).
- $_POST - Variables provided to the script via HTTP POST. Analogous to the old $HTTP_POST_VARS array (deprecated).
- $_COOKIE - Variables provided to the script via HTTP cookies. Analogous to the old $HTTP_COOKIE_VARS array (deprecated).
- $_FILES - Variables provided to the script via HTTP post file uploads. Analogous to the old $HTTP_POST_FILES array (deprecated).
- $_ENV - Variables provided to the script via the environment. Analogous to the old $HTTP_ENV_VARS array (deprecated).
- $_REQUEST - Variables provided to the script via the GET, POST, and COOKIE input mechanisms, and which therefore cannot be trusted.
- $_SESSION - Variables which are currently registered to a script's session. Analogous to the old $HTTP_SESSION_VARS array.



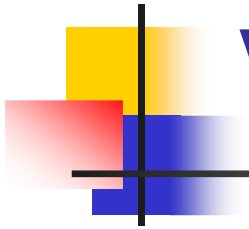
Scope variabili in PHP

- Le variabili possono essere **locali** quando definite dentro una funzione o classe, o **globali**:
  <?php $a = 1; / global scope */*
  function Test() { echo $a; / reference to local scope variable */ }*
  Test(); ?>
- This script will not produce any output because the echo statement refers to a local version of the $a variable, and it has no value within this scope.
  <?php $a = 1; $b = 2;
  function Sum() { global $a, $b; $b = $a + $b; }
  Sum();
  echo $b; ?>
- The above script will output "3". By declaring $a and $b global within the function, all references to either variable will refer to the global version.
  <?php $a = 1; $b = 2;
  function Sum() { $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b']; }
  Sum(); echo $b; ?>
- The $GLOBALS array is an associative array with the name of the global variable being the key $GLOBALS exists in any scope, this is because $GLOBALS is a superglobal

Variabili esterne in PHP

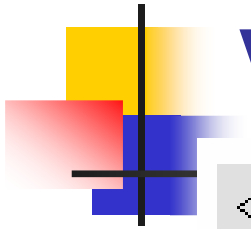
- When a **form is submitted to a PHP script**, the information from that form is automatically made available to the script. There are many ways to access this information, for example:

```
<form action="foo.php" method="post">
Name: <input type="text" name="username" /><br />
Email: <input type="text" name="email" /><br />
<input type="submit" name="submit" value="Submit me!" /> </form>

<?php // Available since PHP 4.1.0
echo $_POST['username'];
echo $_REQUEST['username'];
import_request_variables('p', 'p_');
echo $p_username;
// Available since PHP 3. As of PHP 5.0.0, these long predefined
// variables can be disabled with the register_long_arrays directive.
echo $HTTP_POST_VARS['username'];
// Available if the PHP directive register_globals = on.
As of // PHP 4.2.0 the default value of register_globals = off.
// Using/relying on this method is not preferred.
echo $username; ?>
```



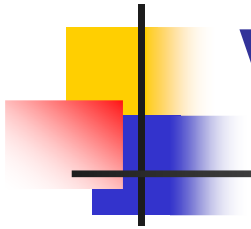
Variabili esterne in PHP

- bool **import_request_variables** (string tipo [, string prefisso])
- Imposta la visibilità delle variabili GET/POST/Cookie a globale. Tramite il parametro **tipo**, si può specificare quale variabile rendere visibile. I valori ammessi sono i caratteri 'G', 'P' e 'C' rispettivamente per GET, POST e Cookie. POST include le informazioni dei file caricati. Occorre prestare attenzione all'ordine delle lettere, ad esempio usando "gp", le variabili POST sovrascrivono le variabili GET con il medesimo nome. Il parametro **prefisso** viene utilizzato come prefisso nel nome della variabile, ovvero viene anteposto ai nomi di tutte le variabili portate a visibilità globale. Quindi, se si ha una variabile GET chiamata "userid", e si è passato il prefisso "pref_", si otterrà una variabile globale chiamata $pref_userid.
- Using a GET form is similar except you'll use the appropriate GET predefined variable instead. GET also applies to the QUERY_STRING (the information after the '?' in a URL). So, for example, http://www.example.com/test.php?id=3 contains GET data which is accessible with $_GET['id'].



Variabili esterne in PHP

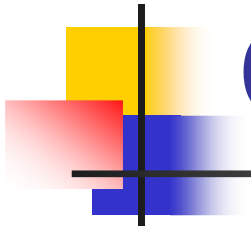
```
<?php
if (isset($_POST['action']) && $_POST['action'] == 'submitted') {
    echo '<pre>';
    print_r($_POST);
    echo '<a href="'. $_SERVER['PHP_SELF'] .'">Please try again</a>';

    echo '</pre>';
} else {
?>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
    Name:  <input type="text" name="personal[name]" /><br />
    Email: <input type="text" name="personal[email]" /><br />
    Beer: <br />
    <select multiple name="beer[]">
        <option value="warthog">Warthog</option>
        <option value="guinness">Guinness</option>
        <option value="stuttgarter">Stuttgarter Schwabenbräu</option>
    </select><br />
    <input type="hidden" name="action" value="submitted" />
    <input type="submit" name="submit" value="submit me!" />
</form>
<?php
}
?>
```



Variabili esterne in PHP

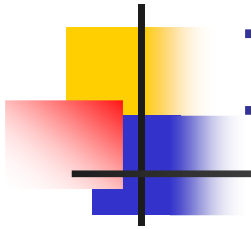
- **Cookies** are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the setcookie() function. Cookies are part of the HTTP header, so the SetCookie function must be called before any output is sent to the browser. Cookie data is then available in the appropriate cookie data arrays, such as $_COOKIE, $HTTP_COOKIE_VARS as well as in $_REQUEST.

```
<?php
if (isset($_COOKIE['count'])) {
$count = $_COOKIE['count'] + 1;
} else { $count = 1; }
setcookie('count', $count, time()+3600);
setcookie("Cart[$count]", $item, time()+3600); ?>
```



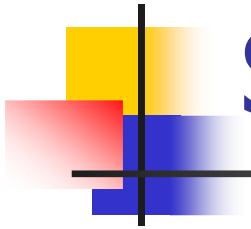
Operatori del PHP

- Oltre gli operatori standard, esistono operatori specifici:
- PHP supporta un **operatore di controllo dell'errore**: il carattere at (@). Quando prefisso ad una espressione in PHP, qualunque messaggio di errore che potesse essere generato da quella espressione sarà ignorato.
- PHP supporta un **operatore di esecuzione**: backticks (``). Notare che quelli non sono apostrofi! PHP cercherà di eseguire il contenuto dei backticks come comando di shell; sarà restituito l'output. L'uso dell'operatore backtick è identico alla funzione shell_exec().
- <?php $output = `ls -al`;
- echo "<pre>$output</pre>"; ?>
- PHP ha un unico operatore di tipo: instanceof. instanceof è utilizzato per determinare se un dato oggetto appartiene ad una data classe di oggetti



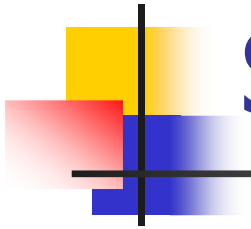
Istruzioni in PHP

- Le **strutture di controllo** in PHP sono if, while, do-while, for, foreach, switch, break, continue, tutte con la stessa sintassi del C/Java
- L'istruzione **require**() include e valuta il file specifico.
- **require**() e **include**() sono identiche in ogni senso eccetto per come esse trattano gli errori. include() produce un Warning mentre require() restituisce un Fatal Error. In altre parole, usare require() se volete che un file mancante fermi l'esecuzione della pagina. include() non si comporta in questo modo, lo script continuerà nonostante tutto.
- L'istruzione require_once() include e valuta il file specificato durante l'esecuzione dello script. È un comportamento simile all'istruzione require(), con la sola differenza che se il codice di un file è stato già incluso, esso non sarà incluso nuovamente.



Sicurezza in PHP

- A completely secure system is a virtual impossibility, so an approach often used in the security profession is one of balancing risk and usability. If every variable submitted by a user required two forms of biometric validation (such as a retinal scan and a fingerprint), you would have an extremely high level of accountability. It would also take half an hour to fill out a fairly complex form, which would tend to encourage users to find ways of bypassing the security.
- The best security is often unobtrusive enough to suit the requirements without the user being prevented from accomplishing their work, or over-burdening the code author with excessive complexity. Indeed, some security attacks are merely exploits of this kind of overly built security, which tends to erode over time.



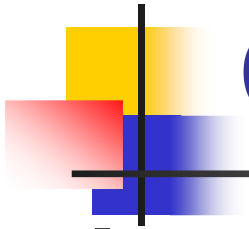
Sicurezza in PHP

- A phrase worth remembering: **A system is only as good as the weakest link in a chain.** If all transactions are heavily logged based on time, location, transaction type, etc. but the user is only verified based on a single cookie, the validity of tying the users to the transaction log is severely weakened.
- When testing, keep in mind that you will not be able to test all possibilities for even the simplest of pages. The input you may expect will be completely unrelated to the input given by a disgruntled employee, a cracker with months of time on their hands, or a housecat walking across the keyboard. This is why it's best to look at the code from a logical perspective, to discern where unexpected data can be introduced, and then follow how it is modified, reduced, or amplified.
- The Internet is filled with people trying to make a name for themselves by breaking your code, crashing your site, posting inappropriate content, and otherwise making your day interesting. It doesn't matter if you have a small or large site, you are a target by simply being online, by having a server that can be connected to. Many cracking programs do not discern by size, they simply trawl massive IP blocks looking for victims. Try not to become one.

Autenticazione HTTP in PHP

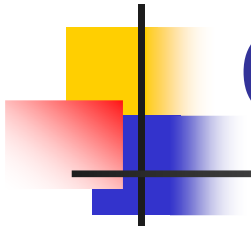
- I meccanismi di Autenticazione HTTP sono disponibili in PHP solo quando questo viene usato come un modulo di Apache. In uno script PHP modulo di Apache, è possibile usare la funzione header() per inviare un messaggio di "Authentication Required" al browser dell'utente, provocando quindi l'apertura di una finestra contenente una richiesta di Nome utente/Password. Una volta che l'utente ha compilato i campi nome utente e password, l'URL contenente lo script PHP verrà richiamato nuovamente usando le variabili predefinite, PHP_AUTH_USER, PHP_AUTH_PW e AUTH_TYPE impostate con, rispettivamente: nome, password e tipo di autenticazione. Queste variabili predefinite possono essere trovate negli array $_SERVER e $HTTP_SERVER_VARS.

```
<?php
if (!isset($_SERVER['PHP_AUTH_USER'])) {
header('WWW-Authenticate: Basic realm="Il mio realm"');
header('HTTP/1.0 401 Unauthorized');
echo 'Messaggio da inviare se si preme il tasto Cancel';
exit;
} else {
echo "<p>Ciao {$_SERVER['PHP_AUTH_USER']}.</p>";
echo "<p>La password è {$_SERVER['PHP_AUTH_PW']}.</p>"; } ?>
```



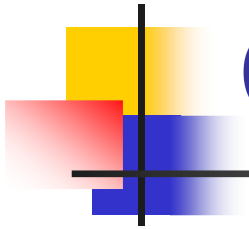
Gestione connessione in PHP

- Internamente il PHP mantiene lo stato della connessione. Si hanno 3 possibili stati:
- 0 - NORMAL
- 1 - ABORTED
- 2 - TIMEOUT
- Quendo uno script PHP viene eseguito normalmente si trova nello stato NORMAL. Se il client remoto si disconnette viene attivato il flag ABORTED. La disconnessione di un client remoto è generalmente causata dalla pressione del tasto STOP da parte dell'utente. Se invece si raggiunge il limite di tempo imposto dal PHP con set_time_limit() lo stato è TIMEOUT.
- Si può decidere se la disconnessione del client debba fare abortire lo script o meno. In certi casi è più pratico lasciare finire lo script anche se non c'è più il browser remoto a ricevere i dati. Tuttavia il comportamento di default è di fare abortire lo script quando il client remoto si disconnette. Questo comportamento può essere impostato tramite la direttiva di php.ini ignore_user_abort oppure tramite la corrispondente direttiva "php_value ignore_user_abort" del file .conf di Apache oppure con la funzione ignore_user_abort().



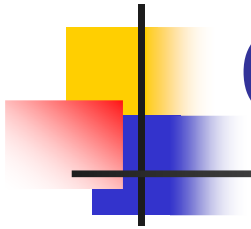
Gestione connessione in PHP

- Se non si segnala al PHP di ignorare la disconnessione dell'utente lo script sarà interrotto. Unica eccezione si ha con la registrazione di una funzione di chiusura utilizzando register_shutdown_function(). Con una funzione di shutdown, quando l'utente remoto preme il bottone di STOP, alla prima occasione in cui lo script tenterà di produrre un output, il PHP intercetterà che la connessione è interrotta e richiamerà la funzione di shutdown. Questa funzione sarà richiamata anche al normale termine dello script, pertanto per eseguire passi differenti in caso di disconnessione del client occorre utilizzare la funzione connection_aborted(). Questa funzione restituisce TRUE se la connessione è stata interrotta.



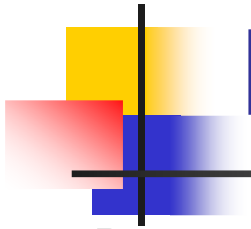
Gestione connessione in PHP

- Uno script può essere fermato dal timer incorporato nel PHP. Per default il timeout è impostato a 30 secondi. Tale limite può essere variato agendo sulla direttiva max_execution_time nel php.ini o nel corrispondente parametro php_value max_execution_time nella configurazione di Apache, oppure con la funzione set_time_limit(). Quando termina il tempo impostato lo script viene interrotto, se è stata prevista una funzione di shutdown, questa verrà eseguita. All'interno di questa funzione si può discriminare se è stata attivata per lo scadere del timeout utilizzando la funzione connection_timeout(). Questa restituisce 2 se la funzione di shutdown è stata chiamata per lo scadere del tempo a disposizione.



Gestione connessione in PHP

- Un aspetto che occorre notare sui stati ABORTED e TIMEOUT è che possono essere attivi contemporaneamente. Questo può accadere se si è impostato il PHP affinchè ignori le interruzioni da parte dell'utente. Infatti il PHP continua a tenere traccia della disconnessione dell'utente, ma, semplicemente, non viene interrotto lo script. Quindi, quando termina il tempo, lo script sarà interrotto e verrà richiamata la funzione di shutdown, se presente. In questa situazione si avrà che connection_status() restituirà 3.

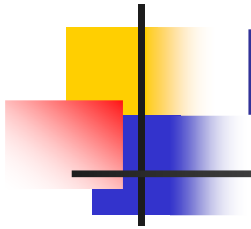


File in PHP

- Per Aprire un File in PHP si utilizza la fopen("nome_file","modalità"); le modalità sono a=apre il file in append, a+=append e lettura, se il file non c'è ne verrà creato uno nuovo, r=lettura, r+=lettura e scrittura w=scrittura, sovrascrive eventuali dati esistenti, w+=lettura e scrittura con sovrascrittura
- E' importante verificare se un file è stato aperto con un if:

```
<?php
if(@!$file=fopen("nomi.txt","r")){
echo "non posso aprire il file";
exit;
}
?>
```

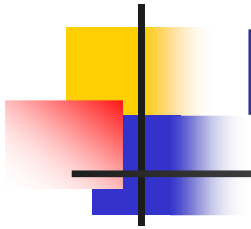
- Se non si può aprire sul browser si vedrà un messaggio d'errore. Per disabilitare i messaggi d'errore basta mettere all'inizio della stringa il carattere @



File in PHP

- Operazioni sui file:
- chiudere: fclose($file);
- scrivere: fputs($file,"testo"); (il file deve essere aperto)
- leggere: fgets($file,Byte da leggere); (il file deve essere aperto)
- Quando si scrive la fine della stringa è bene forzare a capo con \n, e per leggere una riga di file i byte sono 255
- **Esempio di scrittura su file:**

```
<?php
$linea1="E questa è la fine del file... per ora|n";
if (!$p_file = fopen("miofile.txt","w")) {
echo "Spiacente, non posso aprire miofile.txt";
exit;
}
fputs($p_file,"Ecco il file appena creato!|n");
fputs($p_file,$linea1);
fclose($p_file);
?>
```



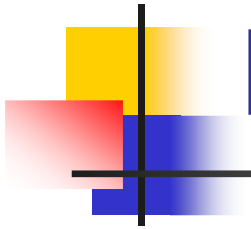
File in PHP

- **Esempio di lettura del file:**

```
<?php
if (!$p_file = fopen("miofile.txt","r")) {
echo "Spiacente, non posso aprire miofile.txt";
exit;
}
$linea= (fgets($p_file,255));
$linea2= (fgets($p_file,255));
echo "$linea<BR>";
echo "$linea2<BR>";
fclose($p_file);
?>
```

- Il risultato sarà:

Ecco il file appena creato!
E questa la fine del file...per ora



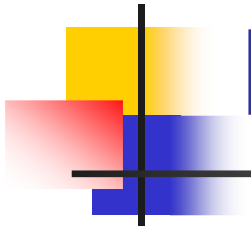
File in PHP

- **Esempio di append:**

```
<?php
$inserisci_linea="\nRiga aggiunta";
if (!$p_file = fopen("/tmp/miofile.txt","a")) {
echo "Spiacente, non posso aprire miofile.txt";
} else {
fputs($p_file,$inserisci_linea);
fclose($p_file);
}
?>
```

- I contenuti del file ora saranno:

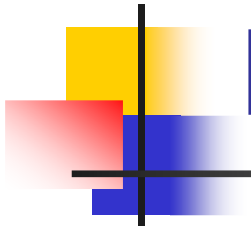
Ecco il file appena creato!
E questa la fine del file...per ora
Riga aggiunta



File in PHP

- Quando si hanno file è buona cura proteggerli in modo da limitare l'accesso. Si usa la funzione **flock**($puntatore_file, metodo), dove 1 oppure LOCK_SH condivide il file in modo che si può leggere ma non scrivere, 2 oppure LOCK_EX fa in modo che nessun altro utente può leggere o scrivere il file fino a quando il blocco non verrà rilasciato, 3 oppure LOCK_UN rilascia qualunque operazione prima impostata

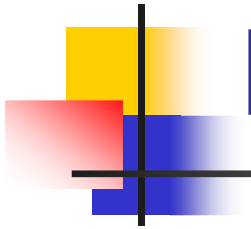
```
<?php
if (!$p_file = fopen("miofile2.txt","w")) {
echo "Spiacente, non posso aprire il file miofile2.txt";
exit;
}
flock($p_file,2);
fputs($p_file,"Sto scrivendo su questo file|n");
flock($p_file,3);
fclose($p_file);
?>
```



File in PHP

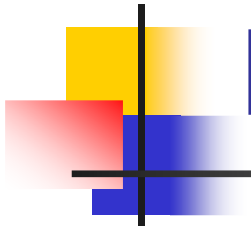
- Fare una **ricerca su un file**

```
<?php
$ricerca="stringa_da_cercare";
if (!$p_file = fopen("utenti.txt","r")) {
echo "Spiacente, non posso aprire il file utenti.txt";
} else {
while(!feof($p_file))
{
$linea = fgets($p_file, 255);
$linea=(trim($linea));
if ( $linea == $ricerca )
echo "Trovato! $linea<BR>";
}
fclose($p_file);
}
?>
```



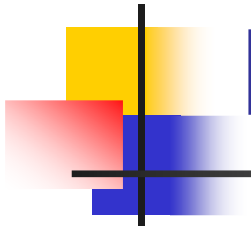
File in PHP

- Altre funzioni utili per la gestione dei file:
- Ottenere le Dimensioni di un file
- Sintassi : filesize(nome_file);
- Controllare se un file è una cartella
- Sintassi : is_dir(nome_file);
- Controllare se un file è ordinario
- Sintassi : is_file(nome_file);
- Controllare se un file è eseguibile
- Sintassi : is_executable(nome_file);
- Controllare se un file è leggibile
- Sintassi : is_readble(nome_file);
- Controllare se un file è scrivibile
- Sintassi : is_writeable(nome_file);
- Ottenere il tipo di file
- Sintassi : filetype(nome_file);
- Se è una cartella restituisce dir, se è un file restituisce file



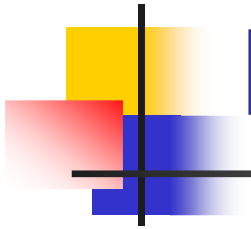
File in PHP

- Esistono funzioni per le cartelle, come chdir(nome), opendir(nome), e readdir(nome).
- **Esempio di lettura cartella:**
- <HTML><BODY>
- <TABLE WIDTH=40% ALIGN=CENTER BORDER=3>
- <TR><TH>ELENCO DEI FILE</TH></TR>
- <?php
- $cartella = opendir('.');
- #creiamo un ciclo che legga i contenuti e li metta nell'array
- while ($file = readdir($cartella)) { $file_array[] = $file; }
- foreach ($file_array as $file) {
- #tutti tranne quelli che iniziano per "." o per ".."
- if ($file == ".." || $file == ".") { continue; }
- #ogni elemento dell'array linkato al proprio nome
- echo "<TR><TD><CENTER>";
- echo "$file";
- echo "</CENTER></TD></TR>";}
- ?>
- </TABLE></BODY></HTML>



Form in PHP

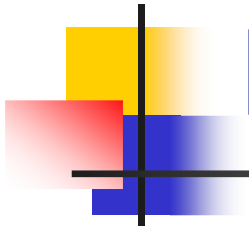
- Una form è una pagina web, realizzata in HTML tramite il tag <form>, tag che permette alla pagina di diventare interattiva, ossia di contenere elementi (caselle di testo, di spunta o bottoni) il cui stato può essere comunicato ad un codice che lo elabora.
- Il codice, specificato come attributo del form, può essere di qualsiasi natura, nel nostro caso un file php (ma poteva essere un cgi, asp, ecc)
- lo stato degli elementi di interattività può essere comunicato in due modi: GET e POST. Nel GET i valori sono passati insieme nella stringa dell'URL, nel POST i valori sono passati separatamente e quindi non sono immediatamente visibili.
- Notare che il recupero di dati passati da una form è possibile richiamando direttamente il nome della variabile (con $nome) solamente se in php.ini la variabile register_globals è impostata a on cosa che normalmente non è. In caso register_globals sia impostato a off il recupero è possibile solamente nella forma $_POST[nome_variabile] o $_GET[nome_variabile]



Form in PHP

- Esempio di form con diversi elementi di interattività:

```
<html><body>
<form method=get action="visualizza.php">
<center>
Inserisci il tuo nome:<input type="text" name="nome"
  size="30"><br>
Inserisci cognome:<input type="text" name ="cognome"
  size="30"><br>
Inserisci la tua e-mail:<input type="text" name ="email"
  size="30"><br><br><br><br>
Scegli quale sezione vorresti che ci fosse nel sito:<br>
<input type="checkbox" name="sport" value="sport">sport
<input type="checkbox" name="cinema" value="cinema">cinema
<input type="checkbox" name="donne" value="donne">donne
<input type="checkbox" name="forum" value="forum"
  checked>forum<br>
```



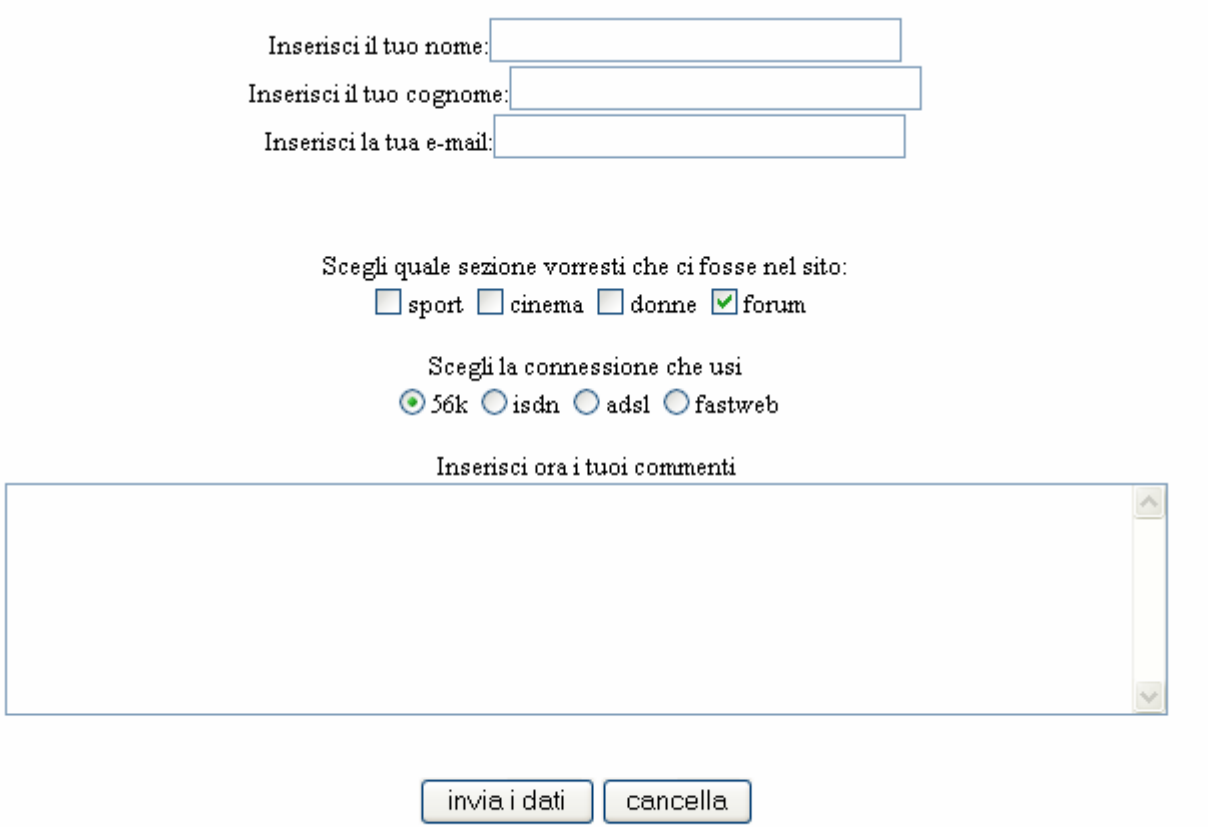
Form in PHP

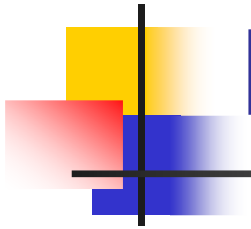
- ...continua

```
<br>Scegli la connessione che usi<br>
<input type="radio" name="linea" value="56k" checked>56k
<input type="radio" name="linea" value="isdn">isdn
<input type="radio" name="linea" value="adsl">adsl
<input type="radio" name="linea" value="56k">fastweb<br>
<br>Inserisci ora i tuoi commenti<br>
<textarea name="commenti" rows="7" cols="70"></textarea>
<br><br><br><input type="submit" value="invia i dati">
<input type="reset" value="cancella">
</form><hr></body></html>
```

Form in PHP

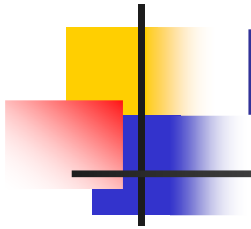
- Il risultato è il seguente:





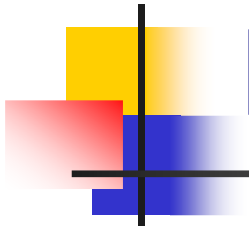
Form in PHP

- Il **metodo get** impostato nella form prevede che i nomi ed i valori da associare siano posti direttamente sul link che richiama la pagina.
  La sintassi è: <a href="nome_pagina.php?nome=valore>
- Se si vogliono inserire più nomi e valori, occorre & fra le coppie di parametri, ad esempio:
- <a href="nome_pagina.php?nome=valore&nome2=valore2>
- eventuali spazi nei valori sono convertiti in %20 o nel carattere +
- Dopo aver inserito questa riga (che per convenzione si chiama query string) abbiamo a disposizione un array di nome $_GET che contiene tutti i parametri che abbiamo inserito nella query string ed è visibile anche dalla nuova pagina.



Form in PHP

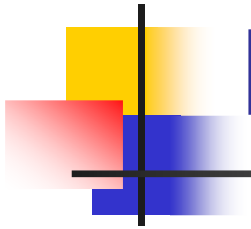
- Per chiarire, supponiamo per esempio di scrivere una riga come questa:
- <a href="index.php?ciao=2&ciao2=4">
- Quando il codice arriva al server abbiamo a disposizione le variabili:
- $ciao=2; $ciao2=4;
- E le possiamo usare per generare la pagina index.php che vedrà l'utente sul suo browser.
- Se per motivi di sicurezza il server ha impostato register_globals su OFF non possiamo usare direttamente le variabili (con il loro nome) se non dopo averle ricavate dall'array $_GET nel seguente modo (nella pagina di destinazione, nel nostro esempio index.php): $ciao=$_GET['ciao']; $ciao2=$_GET['ciao2'];
- Adesso possiamo usare le due variabili $ciao=2 e $ciao2=4



Form in PHP

- Tornando alla form, il codice di visualizza.php (lato "server") potrebbe essere:

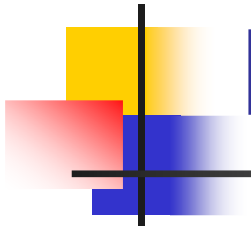
```
<html>
<body>
<center>Riepilogo</center>
<?php
echo "Ciao <b>$nome $cognome</b><br>";
echo "Questa e la tua email <b>$email</b><br>";
echo "<br> Sezioni che vorresti(puoi anche non
  sceglierne):<br>";
echo "<b>$sport $cinema $donne $forum</b><br> ";
echo"<br> Tu navighi con la seguente connessione
  :<b>$linea</b><br>";
echo "<br>ecco i tuoi commenti:<br><b>$commenti</b>";
  ?>
</body>
</html>
```



Form in PHP

- Una versione più completa potrebbe essere:

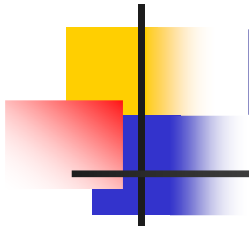
```
<HTML><BODY>
<?php
$errore=false;
if ( ($nome == "") || ($cognome == "") || ($email == "")) {
$errore=true;
echo "<BR>Spiacente, devi compilare tutti i campi<br>";
   } else {
echo "Ciao $nome $cognome<br>, la tua email è: $email<br>";}
$pattern="^[^@ ]+@[^@ ]+\.[^@ \.]+$";
if (!ereg($pattern,$email)) {
echo "<b>\"$email\"</b> non e un email valida";
}else{
echo "Questa e la tua email <b>$email</b><br>";}
if ( $errore) {
echo "<BR>campi incompleti, devi tornare al <A
   HREF=\"modulo.php\">modulo</A> ";}
?>
</BODY></HTML>
```



Form in PHP

- I form possono puntare anche a se stessi invece che puntare ad un'altra pagina:

```
<HTML>
<BODY>
<CENTER>
<?php
echo  "<FORM  METHOD=POST  ACTION=\"$PHP_SELF\">";
   ?>
<INPUT TYPE="SUBMIT" VALUE="Invia le informazioni!">
<INPUT TYPE="RESET" VALUE="Cancella!">
</FORM>
</BODY>
</HTML>
```

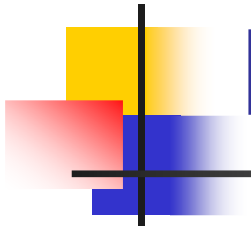


Form in PHP

- Come usare però al meglio il PHP_SELF?. Possiamo creare una pagina con un if. Cioè se il contenuto di una variabile è vuoto allora mostra il form altrimenti mostra un'altra cosa.

```
<HTML><BODY><?php
if ( (empty($nome)) || (empty($cognome))) {
# esegue questa parte perché il form non è stato ancora inviato
echo "<FORM METHOD=POST ACTION=\"$PHP_SELF\">";
echo "Nome? <BR> <INPUT TYPE=\"TEXT\" NAME=\"nome\">";
echo "Cognome? <BR><INPUT TYPE=\"TEXT\" NAME=\"cognome\">";
echo "<BR><BR><INPUT TYPE=\"SUBMIT\" NAME=\"submit\"
   VALUE=\"Invia le informazioni!\">";
echo "<INPUT TYPE=\"RESET\" VALUE=\"Cancella!\">";
echo "</FORM>";
} else {      # il form è stato inviato
echo "elaborazione...<BR>";
echo "Le informazioni sono... nome: $nome , cognome: $cognome";
}
?></BODY></HTML>
```

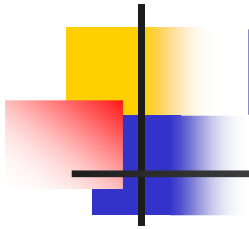
- Quindi se la variabile nome o cognome sono vuote mostra il form



Form in PHP

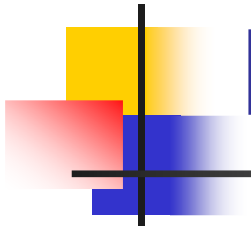
- Per verificare che un form è stato inviato si può anche usare un campo nascosto, come indicato nell'esempio seguente:

```
<HTML><BODY>
<?php
# metto tutto il form in una variabile
$form="<FORM METHOD=|"GET|" ACTION=|"$PHP_SELF|">
<CENTER><B><H3> Nuovo utente </H3></B></CENTER>
Nome utente<BR>
<INPUT TYPE=|"TEXT|" NAME=|"nome|" VALUE=|"$nome|"><BR>
Indirizzo<BR>
<INPUT TYPE=|"TEXT|" NAME=|"indirizzo|" VALUE=|"$indirizzo|"><BR>
<BR>
# ecco il campo nascosto
<INPUT TYPE=|"HIDDEN|" NAME=|"stato|" VALUE=|"inviato|">
<INPUT TYPE=|"SUBMIT|" NAME=|"submit|" VALUE=|"Invia |">
<INPUT TYPE=|"RESET|" VALUE=|"Cancella!|">
</FORM>";
```



Form in PHP

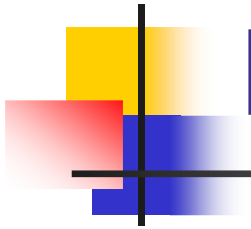
```
# inizio a controllare
$errore_nome=0;
$errore_indirizzo=0;
# il form è stato inviato?
# se il campo nascosto non c'e mostra il form
if($stato!="inviato"){                    echo "$form";
#ma se c'e controlli l'input
} else {
if (empty($nome)) {
echo "<B>Errore:</B> Manca nome <BR>"; $errore_nome=1; }
if (empty($indirizzo)) {
echo "<B>Errore:</B> Manca l'indirizzo <BR>";$errore_indirizzo=1; }
if (($errore_nome) || ($errore_indirizzo)) {   echo "$form";
} else {
echo "elaborazione...<BR>";
echo   "<B>$nome</B>   con   indirizzo   <B>$indirizzo</B>   è   stato
   aggiunto";
}}
?></BODY></HTML>
```



Form in PHP

- Si possono realizzare form a più pagine con passaggi di informazioni nei campi nascosti, ad esempio il primo form chiede nome e indirizzo, il secondo chiede altre info e nasconde nome e indirizzo e il terzo visualizza tutti i dati:

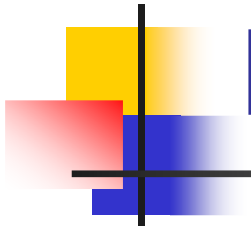
```
<HTML>
<BODY>
<FORM METHOD=GET ACTION="due.php">
<HR>
<B>Qual è il tuo nome completo?</B><BR><INPUT TYPE="text"
NAME="nome" SIZE=20>
<BR><BR>
<B>Qual è il tuo indirizzo email?</B><BR><INPUT TYPE="text"
NAME="email" SIZE=20>
<BR><BR>
<B>Qual è il tuo numero di telefono?</B><BR><INPUT
TYPE="text" NAME="telefono" SIZE=20>
<BR><BR>
```



Form in PHP

- ...continua...

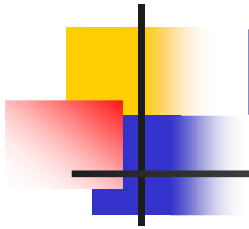
```
<B>Come desideri effettuare il pagamento?</B><BR>
<SELECT NAME="pagamento" >
<OPTION SELECTED>Carta di Credito
<OPTION>Assegno
<OPTION>Bonifico Bancario
<OPTION>Contanti
</SELECT>
<HR>
<INPUT TYPE="submit" VALUE="Pagina seguente">
<INPUT TYPE="reset" VALUE="Cancella!">
</FORM>
</BODY>
</HTML>
```



Form in PHP

- La seconda form riceve i campi della prima sull'URL con il metodo GET, e li propaga alla terza form tramite campi nascosti, cosi' saranno insieme ai campi non nascosti di questa seconda form (quelli relativi alle informazioni aggiuntive), tutti propagati con GET :

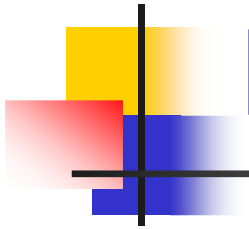
```
<HTML><BODY>
<CENTER><H2>Seleziona le opzioni di consegna</H2></CENTER>
<FORM METHOD=GET ACTION="tre.php">
<B>Tipo di servizio:</B><BR>
<SELECT NAME="tipo_consegna" >
<OPTION SELECTED>Gold Star
<OPTION>Silver Star
<OPTION>Piccione viaggiatore
</SELECT><BR><BR>
<B>Giorno di ritiro del plico:</B><BR>
<SELECT NAME="giorno">
<OPTION SELECTED>Lunedì
<OPTION>Martedì<OPTION>Mercoledì<OPTION>Giovedì
</SELECT>
```



Form in PHP

- *...continua...*

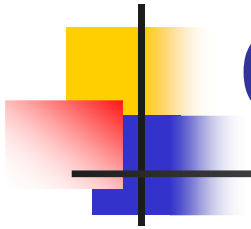
```
<?php
# innanzitutto codifica i tre valori nascosti
$nome=(urlencode($nome));
$email=(urlencode($email));
$telefono=(urlencode($telefono));
# ecco i campi nascosti all'interno del form
echo "<INPUT TYPE=\"HIDDEN\" NAME=\"nome\" value=$nome>";
echo "<INPUT TYPE=\"HIDDEN\" NAME=\"email\" value=$email>";
echo "<INPUT TYPE=\"HIDDEN\" NAME=\"telefono\" value=$telefono>";
echo "<INPUT TYPE=\"HIDDEN\" NAME=\"pagamento\"
value=$pagamento>";
?>
<HR>
<INPUT TYPE="submit" VALUE="Pagina seguente">
<INPUT TYPE="reset" VALUE="Cancella!">
</FORM></BODY></HTML>
```



Form in PHP

- infine la terza form riceve e stampa tutto:

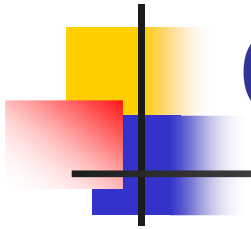
```
<HTML><BODY>
<H2>Consegne Spa ti ringrazia</B>
<TABLE BORDER=6 WIDTH=60% COLS=1>
<TR><TH>Ecco le informazioni da te indicate</TH></TR>
<TR><TD>
<?php
$nome=(urldecode($nome));
$email=(urldecode($email));
$telefono=(urldecode($telefono));
echo "Il tuo nome è: <B>$nome</B><BR>";
echo "Il tuo indirizzo email è: <B>$email</B><BR>";
echo "Il tuo numero di telefono è: <B>$telefono</B><BR>";
echo "Il metodo di pagamento è: <B>$pagamento</B><BR>";
echo "Il tipo di servizio selezionato è: <B>$tipo_consegna</B><BR>";
echo "Ritireremo il plico di: <B>$giorno</B><BR>";
echo "Il peso del plico è: <B>$peso</B><BR>";
?>
</TD></TR></TABLE></BODY></HTML>
```



Cookie in PHP

- I Cookie sono il metodo più comune per la memorizzazione dello stato nel web. Il cookie è un piccolo file che contiene alcune informazioni e viene memorizzato nella memoria del browser o sul disco. Per inviare cookie devono essere attivati dal browser del client. Quando si invia un cookie lo possiamo leggere solo quando l'utente torna sul sito. I cookie devono essere emessi prima di qualsiasi codice php o html. Questi non devono superare le dimensioni di 4 Kb. I cookie possono contenere fino a sei componenti: Nome, Valore, Scadenza, Percorso, Dominio, Sicurezza.
- La sintassi per emettere un cookie è:
  setcookie(nome, valore, tempo, percorso, dominio, sicurezza).

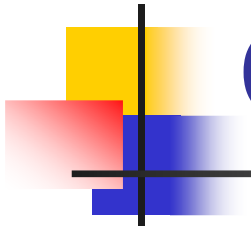
```
<?phpsetcookie ("test_cookie","niente di particolare",time()+43200,"/");
# cookie.php
echo "<HTML>";
echo "<BODY>";
if (isset($test_cookie)){
echo "Ciao cookie, i tuoi contenuti sono: $test_cookie";
} else {
echo "Non ho trovato alcun cookie con il nome test_cookie";
}     echo "</BODY>";          echo "</HTML>";          ?>
```



Cookie in PHP

- Per eliminare un cookie possiamo fare in due modi:
  setcookie ("nome del cookie");
  oppure mettere un tempo di scadenza negativo
  setcookie ("nome","","time()-43200,"/");
- Possiamo vedere tutti i cookie che corrispondono al nostro sito tramite la variabile $HTTP_COOKIE_VARS["nome"];
- I cookie si possono usare ad esempio per personalizzare la pagina:

```
<?php
#se c'e allora ciao
if (isset($nome)){
echo "Ciao $nome";
}else{
#altrimenti
echo "Non ti conosco :<a href=|"inserisci.php|">Inserisci il tuo
   nome</a>";
}?>
```



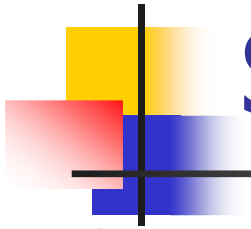
Cookie in PHP

Ecco la pagina dove si inserisce il nome:

```
<html>
<body bgcolor="blue">
<br><br><br><br>
<center><form method=get action="cookie2.php">
<input type="text" name="nome1">
<input type="submit" value="invia">
</form>
</body>
</html>
```

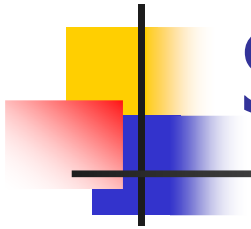
Ecco la pagina che imposta il cookie (chiamata cookie2.php):

```
<?php
setcookie ("nome",$nome1,time()+2592000,"/");
echo "cookie impostato
<a href=|"cookie1.php|"> VERIFICA</a>";
?>
```



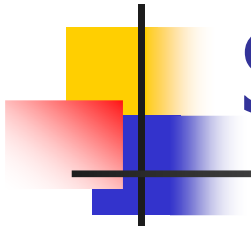
Sessioni in PHP

- La sessione è un supporto offerto dal PHP per permettere di preservare dei dati fra più pagine web, perché esse costituiscono un unico collegamento logico (sessione, appunto) effettuato da un utente/client. Il supporto si rende indispensabile perché il protocollo HTTP è stateless, ossia non conserva informazioni sullo stato precedente, per cui passare da una pagina ad un'altra, a livello di HTTP, implica una connessione nuova ogni volta.
- Il supporto delle sessioni permette di registrare numeri arbitrari di variabili che vengono preservate secondo richiesta. Quando un visitatore accede al sito, PHP controllerà automaticamente (se session.auto_start è settato a 1) o su richiesta (esplicitamente tramite session_start() o implicitamente tramite session_register()) se uno specifico id di sessione sia stato inviato con la richiesta. In questo caso, il precedente ambiente salvato viene ricreato.



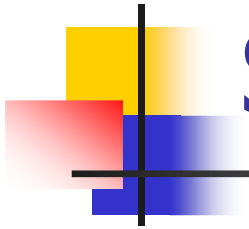
Sessioni in PHP

- Ci sono due metodi per propagare l'id di sessione:
- I Cookies
- Un parametro dell'URL
- Il modulo di sessione supporta entrambi i metodi. I cookies sono ottimi, ma dal momento che possono non essere a disposizione (i clients non sono costretti ad accettarli), non possiamo dipendere da questi. Il secondo metodo incorpora l'id di sessione direttamente negli URL, cosa che rende visibile l'id (eventuale problema di sicurezza).
- La soluzione comunemente adottata è di propagare l'id sull'URL, garantendo l'indipendenza dal cookie, e operare all'interno di HTTPS, protocollo sicuro perché cripta la comunicazione



Sessioni in PHP

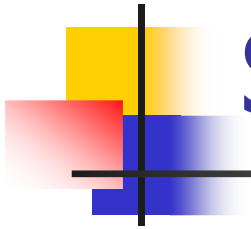
- La configurazione di track_vars e register_globals influenza come le variabili di sessione vengono memorizzate una e più volte.
- Se **track_vars è attiva e register_globals non è attiva**, solo i membri dell'array associativo globale $HTTP_SESSION_VARS possono essere registrati come variabili di sessione. Le variabili di sessione ripristinate saranno disponibili nell'array $HTTP_SESSION_VARS.
- Esempio: Registrare una variabile con track_vars attiva
  <?php if (isset($HTTP_SESSION_VARS['count'])) {
  $HTTP_SESSION_VARS['count']++; } else {
  $HTTP_SESSION_VARS['count'] = 0; } ?>
- L'uso di $_SESSION (o $HTTP_SESSION_VARS con PHP 4.0.6 o precedente) è raccomandato per sicurezza e leggibilità del codice.
- Con $_SESSION o $HTTP_SESSION_VARS, non si devono usare le funzioni session_register(), session_unregister(), e session_is_registered(). Gli utenti devono accedere alla variabile di sessione come a una variabile normale.



Sessioni in PHP

- Se **register_globals è attiva**, allora tutte le variabili globali possono essere registrate come variabili di sessione e le variabili di sessione saranno ripristinate in corrispondenza delle variabili globali. Dal momento che PHP ha bisogno di sapere quali variabili globali sono registrate come variabili di sessione , gli utenti devono registrare le variabili con la funzione session_register()
- Esempio: Registrare una variabile con register_globals attiva

```
<?php
if (!session_is_registered('count')) {
session_register("count");
$count = 0;
} else {
$count++; } ?>
```

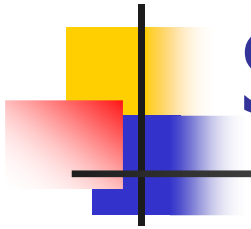


Sessioni in PHP

- L'esempio seguente dimostra come registrare una variabile e come collegare una pagina all'altra correttamente usando la variabile predefinita SID per l'id di sessione.

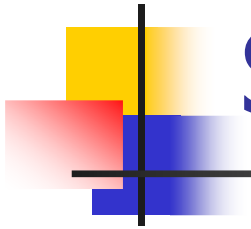
```
Esempio: Contare il numero di accessi di un singolo utente
<?php
if (!session_is_registered('count')) {
session_register('count');
$count = 1;
} else {
$count++; } ?>
Salve visitatore , hai visitato questa pagina
<?php echo $count; ?> volte.<p>;
<!-- <?php echo SID?> è necessario per preservare l'id di sessione
nel caso in cui l'utente abbia disattivato i cookies ?-->
Per continuare,
<A HREF="nextpage.php?<?php echo SID?>">clicca qui</A>
```

- Gli URL non relativi si presume che puntino a siti esterni e quindi non hanno il SID, perchè sarebbe rischioso per la sicurezza propagare il SID a un altro server.



Sessioni in PHP

- Funzioni e codici da usare nel caso che register_globals sia attiva :
- session_start(); Funzione che apre una sessione.
- session_destroy(); Funzione che distrugge una sessione.
- session_register("..."); Salva una variabile nella sessione. (Si scrive solo il nome senza $. Il valore della variabile va inserito nel codice sottostante, in pratica la macchina lo salva da sola.)
- session_unset(); Distrugge le variabile nella sessione.
- $PHPSESSID Il nome della variabile globale che contiene il codice univoco di 32 caratteri che distingue la sessione. (Se scriviamo echo $PHPSESSID; scriviamo 32 caratteri a a video se la sessione è attiva.)
- Se per motivi di protezione non è possibile utilizzare direttamente la variabile $PHPSESSID è possibile inserire la funzione session_id(); esempio $PHPSESSID=session_id();
- a seconda della versione del PHP, viene usata $SID o SPHPSESSID

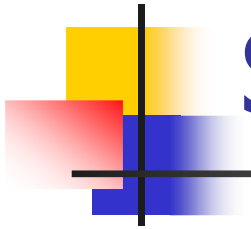


Sessioni in PHP

- Solitamente una connessione si apre così:

```
<?
session_start();//Apriamo la sessione
session_register("nomevariabile");//Salviamo una variabile
   nella sessione
...//Calcoli con la sessione...
session_register("nomevariabile2");//Salviamo una variabile
   nella sessione
$nomevariabile=...;//Definiamo la variabile
$nomevariabile2=...;//Definiamo la variabile2
...//Calcoli con la sessione...
?>
```

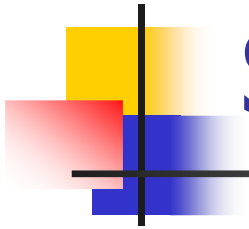
- Il codice per aprire una sessione va inserito prima del codice HTML (prima del tag <HTML>).



Sessioni in PHP

- Nel caso si voglia distruggere una sessione e riaprirne un'altra, magari perchè alcuni dati non influenzino la pagina corrente si può scrivere così:

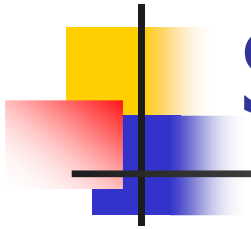
```
<?
session_start();//Distruggo la vecchia sessione
session_unset();
session_destroy();
session_start();//Apro una nuova sessione
...
```



Sessioni in PHP

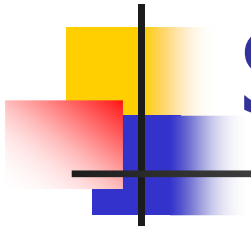
- Esempio di form con sessioni: il primo (ss1.php) crea una sessione e registra il nome e il dipartimento, il secondo (ss2.php) registra indirizzo e numero di telefono e il terzo (ss3.php) mostra le informazioni e le manda per email

```
<?php
# ss1.php
session_start();
$num_impiegato =session_id();
session_register("nome","dipartimento"); ?>
<HTML>
<BODY>
<?php
$form="<CENTER><H2>Nuovo Impiegato</H2></CENTER>
<FORM ACTION=\"ss2.php\" METHOD=\"POST\">
<B>Nome impiegato:</B><BR>
<INPUT TYPE=\"text\" NAME=\"nome\" SIZE=20>
<BR><B>Dipartimento</B><BR>
```



Sessioni in PHP

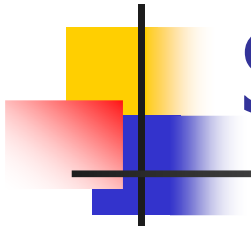
```
<SELECT NAME=\"dipartimento\" >
<OPTION SELECTED>Autista consegne
<OPTION>Magazzino
<OPTION>Vendite
<OPTION>Amministrazione
</SELECT>
<BR>
<INPUT TYPE=\"submit\" VALUE=\"Pagina seguente\">
<INPUT TYPE=\"reset\" VALUE=\"Cancella!\">
</FORM>";
?>
</BODY></HTML>
<?php
echo $form;
?>
```



Sessioni in PHP

ss2.php

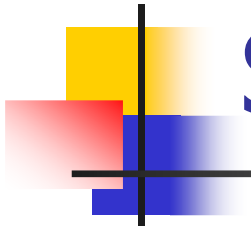
```
<?php
session_start();
session_register("indirizzo","telefono"); ?>
<HTML><BODY>
<?php
$form="<CENTER><H2>Nuovo Impiegato</H2></CENTER>
<FORM ACTION=\"ss3.php\" METHOD=\"POST\">
<B>Indirizzo impiegato:</B><BR>
<INPUT TYPE=\"text\" NAME=\"indirizzo\" SIZE=30>
<BR><BR><B>Numero telefono impiegato:</B><BR>
<INPUT TYPE=\"text\" NAME=\"telefono\" SIZE=30><BR><HR>
<INPUT TYPE=\"submit\" VALUE=\"Pagina seguente\">
<INPUT TYPE=\"reset\" VALUE=\"Cancella!\">
</FORM>";?>
</BODY></HTML>
<?php                    echo "$form";              ?>
```



Sessioni in PHP

ss3.php

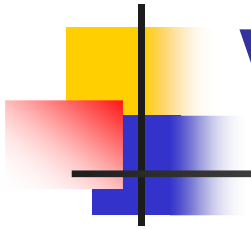
```
<?php session_start();          ?>
<HTML><BODY>
<CENTER><H3> informazioni sul nuovo impiegato</H3></CENTER>
<?php
$visualizza="<PRE>
Nome impiegato: $nome<BR>
Dipartimento: $dipartimento<BR>
Livello retributivo: $stipendio<BR>
Sede: $sede<BR>
Indirizzo casa: $indirizzo<BR>
Telefono: $telefono<BR>
ID impiegato: $num_impiegato<BR></PRE>";
?>
</body></html>
```



Sessioni in PHP

ss3.php continuo

```
<?echo "$visualizza";
$destinatario="vostraemail@nonso.it";
$header="Inserimento nuovo impiegato";
$info="Ecco le informazioni...
NOME: $nome
DIPARTIMENTO: $dipartimento
LIVELLO RETRIBUTIVO: $stipendio
INDIRIZZO CASA: $indirizzo
NUM TELEFONO: $telefono
ID IMPIEGATO: $num_impiegato";
mail($destinatario,$header,$info);
session_destroy();
?>
```

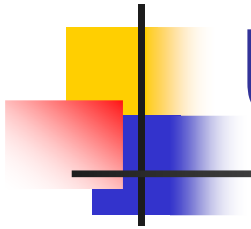


Variabili predefinite in PHP

- Le principali variabili predefinite secondo il protocollo CGI (Common Gateway Interface) sono:
- **Informazioni del server**
- SERVER_SOFTWARE, il server
- SERVER_NAME, nome del server
- SCRIPT_FILENAME, percorso e nome dello script
- SCRIPT_NAME, nome dello script
- **Informazioni del Client**
- REMOTE_HOST, nome del computer (ad esempio, dell'ISP)
- REMOTE_USER, nome dell'utente (se è presente l'autenticazione)
- REQUEST_METHOD, get o post
- REMOTE_ADDR, indirizzo del browser che fa la richiesta
- **Informazioni passate dal client al server**
- QUERY_STRING, è la parte dell'url dopo il punto interrogativo
- CONTENT_LENGTH, lunghezza dei dati inviati
- HTTP_USER_AGENT, nome del browser e sistema operativo

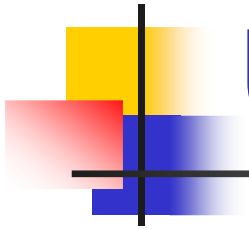
Variabili predefinite in PHP

- **Altre (utili) variabili predefinite di php**
- HTTP_AUTH_USER, nome dell'utente(se c'e autenticazione)
- HTTP_AUTH_PW, password(se c'e autenticazione)
- PHP_SELF, nome dello script
- HTTP_POST_VARS, array con coppie chiave-valore (post)
- HTTP_GET_VARS, array con coppie chiave-valore (get)
- HTTP_COOKIE_VAR["PHPSESSID"]
- Contiene il valore dell' ID di sessione
- HTTP_SERVER_VAR["HTTP_COOKIE"]
- Contiene il valore di tutti i cookie
- HTTP_SERVER_VAR["HTTP_HOST"]
- Contiene il nome dell'host su quale risiede il server web
- HTTP_SERVER_VAR["REMOTE_ADDR"]
- Contiene l'indirizzo remoto del browser
- HTTP_SERVER_VAR["SCRIPT_FILENAME"]
- Contiene il nome e il percorso completo dello script corrente
- HTTP_SERVER_VAR["SERVER_NAME"]
- Contiene il nome del server web
- HTTP_SERVER_VAR["SERVER_SOFTWARE"]
- Contiene il nome del server web



Upload di file in PHP

- L'upload di file si fa con un semplice form con una dicitura in più che permette a php di capire che si tratta di un invio di file con post. Occorre tuttavia controllare l'impostazione dei permessi sul file system
- <form method=post action="uploadit.php" ENCTYPE="multipart/form-data">
- L'input che permette di sfogliare (file dialog) è del tipo: <input type= file name="uploadfile">
- Quando si invia il file (uploadfile) php crea delle variabili utili:
- $uploadfile_name (nome e percorso)
- $uploadfile_size (dimensioni)
- $uploadfile_tupe (tipo)
- $uploadfile (nome del file creato dopo l'invio)



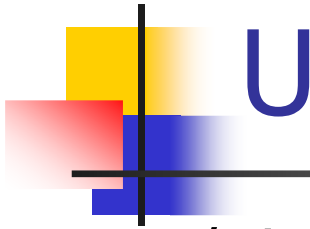
Upload di file in PHP

- il form d'invio può essere come segue:

```
<HTML><BODY>
<FORM METHOD=POST ACTION="uploadit.php
    "ENCTYPE="multipart/form-data" >
<B>Inserisci il nome del file da inviare </B>
<INPUT TYPE=FILE NAME="uploadfile"><BR><BR>
<INPUT TYPE="SUBMIT" VALUE="Invia le informazioni!">
<INPUT TYPE="RESET" VALUE="Cancella!">
</FORM></BODY></HTML>
```

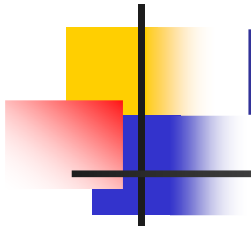
- Una volta inviato ecco come si potrebbe accedere alle variabili:

```
<HTML><BODY>
<?php
echo "nome del file locale creato dopo l'invio: $uploadfile\n";
echo "nome originale del file remoto: $uploadfile_name\n";
echo "dimensioni del file in byte: $uploadfile_size\n";
echo "tipo di file: $uploadfile_type\n";
echo "<HR>";
```



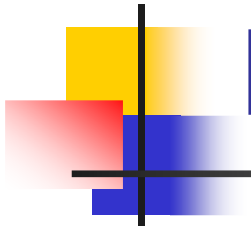
Upload di file in PHP

```
# Altri esempi di controlli…
# abbiamo veramente un file?
if ( $uploadfile == "none" ) {
echo "Non è stato inviato alcun file<BR>";
echo "<A HREF=\"upload.html\"> Ritorna al form </A>";
exit;}
# controlla innanzitutto le dimensioni del file (se è meno di 3 KB)
if ($uploadfile_size < 3072 ) {
# lo copia in una nuova posizione
if (copy($uploadfile,"/home/httpd/docs/$uploadfile_name"))
{echo "Invio del file riuscito";
# cancella il file temporaneo
unlink($uploadfile);
} else {        echo "Invio del file fallito";    }
} else {        echo "il file non deve superare 3 KB<BR>"; }
?>
Ritorna al form per inviare il <A HREF="upload.html">file</A>
</BODY></HTML>
```



Download di file in PHP

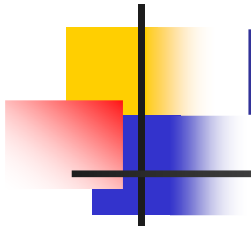
- Quando vogliamo fare il download dobbiamo mandare il contenuto direttamente agli Header specificando il contenuto MIME. Nell'header ci devono essere:
- Content-type (tipo di documento)
- Content-Disposition (destinazione del contenuto)
- Content-Description (descrizione del contenuto)
- Content-Length (lunghezza del contenuto)
- Content-type Application/octet-stream costringe il browser ad aprire la finestra di salvataggio file; Content-Disposition ha il valore “attachment”, Content-Description per esempio Download e Content-Length la lunghezza del file che stiamo per scaricare.



Download di file in PHP

- Ecco una pagina di esempio:

```
<HTML><BODY>
<b>Fai clic per scaricare i file</b>
<?
# elenco dei contenuti di una cartella con link
$cartella = opendir('docs');
while ($file = readdir($cartella)) {
$array_file[] = $file;     }
foreach ($array_file as $file) {
if ( $file == ".." || $file == ".") {      continue;      }
$dim_f=filesize("docs/".$file);
echo "<a href=\"
downloadit.php?nome_f=$file&dim_f=$dim_f\">$file</a>";
echo "<BR>";    }       ?>
</BODY></HTML>
```

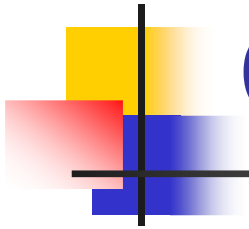


Download di file in PHP

- Quando si clicca su un file, si va su downloadit.php:

```
<?php
header("Content-type: Application/octet-stream");
header("Content-Disposition: attachment;
filename=$nome_file");
header("Content-Description: Download PHP");
header("Content-Length: $dimensioni_file");
readfile($nome_file);
?>
```

- Questa pagina non fa altro che ricevere le informazioni e fare scaricare il file



Connessione database in PHP

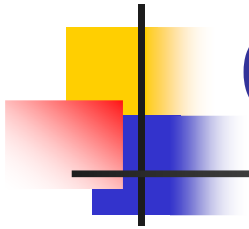
- Sovente il php opera con un database per reperire le informazioni che rendono dinamiche le pagine web
- Per utilizzare una connessione dobbiamo avere alcuni dati:

```
<?
$db_host = "localhost";
$db_user = "nostro nome";
$db_password = "nostra password";
$db_database = "nome nostro database";
?>
```

- Per connetterci ad un database MySql :

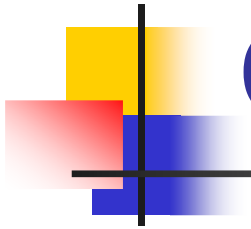
```
$xxx = mysql_connect($db_host,$db_user,$db_password);
$yyy=mysql_db_query($db_database,"query",$xxx);
```

- La prima per ottenere il valore della variabile resource, con la quale ci potremo riferire alla connessione da altre query, la seconda dove specificare il nome del database al quale ci vogliamo connettere.



Connessione database in PHP

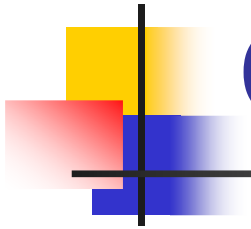
- Vediamo le query fondamentali :
- Inserire una riga: *mysql_db_query($db_database,"insert into ".$db_tabella."(nome, email) values('".$nome."', '".$email."')",$xxx);*
- La riga verrà aggiunta dopo l'ultima riga della tabella.
- Eliminare una riga: *mysql_db_query($db_database,"delete from ".$db_tabella." where id=5 and....",$xxx);*
- Modificare una riga: *mysql_db_query($db_database,"update ".$db_tabella." set nome='".$nome."', email='".$email."' where id=5 and....",$xxx);*
- Interrogare una tabella: *mysql_db_query($db_database,"select nome, email from ".$db_tabella." where id=5 ",$xxx);*
- Nel caso che all'interno di una query si utilizzi una variabile stringa si deve operare nel seguente modo:
- *mysql_db_query($db_database,"select nome, email from ".$db_tabella." where cognome=' ".$cogn."' ",$xxx);*



Connessione database in PHP

- Lavorando con PHP e MySql possiamo utilizzare alcune funzioni da abbinare con le query:
- *mysql_affected_rows();* Questa funzione restituisce il numero di righe della tabella modificate dalla query.
- *mysql_num_rows($query);* Questa funzione restituisce il numero di righe dopo una ricerca con 'select', ad esempio:

```
$query=mysql_db_query($db_database,"select nome, email
from ".$db_tabella." where id=5 ",$xxx);
$num=mysql_num_rows($query);
$abc=0;
while ( $abc<$num ){
$caso = mysql_result($query,$abc,"nome");
$caso2 = mysql_result($query,$abc,"email");
echo $caso." = ".$caso2."<br>";
$abc++;
}
```



Connessione database in PHP

- Abbiamo fatto una ricerca, poi abbiamo trovato il numero di righe dei risultati trovati, abbiamo impostato una variabile ($abc) a zero, ed infine con un ciclo while abbiamo scritto il nome e la mail dei risultati trovati.
  Il risultato potrebbe essere:
  gigi = abc@def.i
- *mysql_result(xxx,yyy,"zzz");*
  Questa funzione (come visto nell'esempio precedente) restituisce il valore di una cella della tabella conoscendo:
  xxx = La variabile associata alla query di ricerca (select).
  yyy = La variabile associata al numero di riga.
  zzz = Il nome della colonna da cui prelevare il dato.