Server side: PHP

Ing. Cesare Monti - 20 aprile 2005

Server Side

- Siamo nel WEB:
  - Contesto eterogeneo
    - molti linguaggi per la creazione di CGI
    - nessuno standard oltre al protocollo per il trasferimento dati
  - stato si / stato no / stato forse
    - tanti meccanismi
      - ogni linguaggio ha i propri
    - pochi modelli
      - quasi nulla

Server Side

- 1991/92
  - NCSA - Mosaic
    - primi browser
    - primi server web
  - CGI
    - standard machine dependent
    - difficoltà nell'estrarre i dati dalle form
      - librerie differenti in linguaggi diversi per estrarre variabili dai vari methods delle form
      - come ogni problema trova qualcuno che ci studia sopra ...

Server Side

- nascono differenti linguaggi ognuno con meccanismi propri
  - PHP
  - Java Servlet / JSP
  - ASP
  - Phyton

PHP

Cos'è PHP

- PHP letteralmente è un acronimo ricorsivo per PHP HyperText Preprocessor
- PHP è un linguaggio di scripting general-purpose espressamente costruito per lo sviluppo di applicazioni server-side.
- È un linguaggio che permette di scrivere codice incastonato dentro HTML che interagisce con uno script CGI esterno in maniera dinamica (PHP_interpreter)

PHP:Origini e tipologie del linguaggio

- Nasce nel 1994:
  - come progetto "personale“ di Rasmus Lerdorf
    - la prima versione pubblicamente utilizzabile risale al 1995 con il nome di "Personal Home Page".
- Il resto, è storia:
  - il linguaggio si sviluppa come progetto open-source
    - nel 1996, già circa 15.000 siti web lo utilizzano;
    - alla release 3 (metà del 1999) il numero di server che utilizzano PHP si è decuplicato
    - nel 2002 release 4 si stimava intorno ai 25 milioni
    - oggi (2004) release 5 in rilascio ... non si riescono a contare

CGI e PHP: fondamentali differenze

- PHP è un linguaggio embeeded nel codice HTML delle pagine,
  - ...le cgi no
- uno script PHP, di fatto, non ha bisogno di installazione come avviene per uno script CGI:
  - ... guardate su www.cgipoint.it, guardate come ogni script debba essere caricato sul server in determinate directory con determinati permessi e via dicendo.
- con PHP non si ha più bisogno di particolari configurazioni del webserver
  - non si abilitano directory cgi-bin
  - non si abilitano l'esecuzione di determinati file con determinate estensioni
- Ogni script (o meglio, pagina con all'interno il codice dello script) potrà essere eseguito in OGNI directory esso si trovi.

Iniziamo a guardare l’ambiente in cui opera il linguaggio

- Per poter utilizzare un linguaggio occorre predisporre un ambiente di lavoro,
  - nel caso di PHP occorre abilitare il web server a ridirigere la request all’interprete e catturarne il risultato
  - Occorre insegnare al web server quale sia il tipo di file giusto (un unica estensione)
- ...sembra complicato
- ...in realtà
  - Per configurare Apache basta:
    - caricare il modulo relativo al PHP nel file di configurazione (httpd.conf)
    - definire un'appropriata estensione per le pagine PHP
      - ...il tutto in due righe di configurazione

Struttura del Linguaggio

1. Sintassi Fondamentale
2. Tipe
3. Variabili
4. Costanti
5. Espressioni
6. Operatori
7. Strutture di controllo
8. Funzioni
9. Classi e Oggetti

1) Sintassi fondamentale

1.1 Come fare eseguire codice al’interprete PHP

1.2 Separazione delle istruzioni

1.3 Commenti

1.1 Come fare eseguire codice a PHP

```
<?php ?>
<? ?>

<? echo ("questo è l'istruzione più semplice, ovvero una istruzione SGML.\n"); ?>
<?= espressione ?> Questa è un'abbreviazione per "<? echo espressione ?>"

<?php echo("se si vogliono produrre documenti XHTML o XML, si utilizzi questo modo\n"); ?>

<script language="php">
    echo ("alcuni editor (tipo FrontPage) non amano le istruzioni di elaborazione");
</script>

<% echo ("Opzionalmente puoi utilizzare tag nello stile ASP"); %>
<%= $variable; # Questo è una abbreviazione per "<%echo .." %>
```

Perché co-esistono insieme PHP e HTML ?

```
//      In blu il codice PHP,          in rosso il codice HTML

<?php
if (boolean-expression) {
?> <strong>This is true.</strong>
<?php
} else
{
?> <strong>This is false.</strong>
<?php } ?>
```

questo funziona perché PHP interpreta tutto ciò che trova tra ?> e <? come un instruzione di echo ();

1.2) Separazione delle istruzioni

- Le istruzioni sono separate come nel C o in perl o in java
  - ...ogni istruzione termina con un punto e virgola.
- Il tag di chiusura (?>) implica anche la fine di un'istruzione, perciò le seguenti sono equivalenti:

```
<?php echo "Questo ` un test"; ?>
<?php echo "Questo ` un test" ?>
```

1.3) Commenti

- Il PHP (nativo) supporta tre tipi di commenti:
  - `/* */` -> come nel linguaggio C;
  - `//` -> come nel linguaggio C++;
  - `#` -> come nello scripting di shell e nel Perl.
- Questa derivano dal fatto che originariamente il PHP (quando era Personal Home Page) era una procedura che permetteva di richiamare codice C da una pagina web con tag speciali

2) Tipi

- PHP supporta otto diversi tipi primitivi:
  - scalari:
    1. Boolean
    2. Integer
    3. Floating-point-number (float)
    4. String
  - composti:
    5. Array
    6. Object
  - speciali:
    7. Resource
    8. NULL

In PHP non è necessario attribuire un tipo ad ogni variabile, è il PHP stesso che run-time decide il tipo di ogni variabile in funzione del contesto

Booleans

- Il tipo boolean esprime la verità, può assumere due soli valori: TRUE | FALSE
- Sono considerati falsi i valori:
  - FALSE
  - L'integer 0
  - Il float 0.0
  - La stringa "" e la stringa "0"
  - Un array con 0 elementi
  - Un oggetto con 0 elementi
  - Il tipo speciale NULL
- Tutti gli altri sono considerati veri (compreso -1!)

Integers

- Un integer è un numero appartenente all'insieme di numeri interi Z={..., -2, -1, 0, 1, 2, ...}
- Un integer può essere specificato in base decimale, ottale o esadecimale

```
$a = 1234; # integer in base decimale

$a = -123; # integer negativo in base decimale

$a = 0123; # integer in base ottale (equivalente all'83 decimale)

$a = 0x1A; # integer esadecimale (equivalente a 26 decimale)
```

- Php non supporta unsigned integer
- La dimensione di un integer è platform-dependent (usually 32 bit)
- Se si specifica un integer fuori dai limiti di rappresentazione, questo viene convertito automaticamente in un float

Floating Point Number (float)

- Un float può essere definito con tre diverse sintassi:
  - `$a = 1.234;`
  - `$a = 1.2e3;  // (equivalente a 1,2 * 10^3  = 1200)`
  - `$a = 7E-10; // (equivalente a 1,2 * 10^-3 =0.0012)`
- La dimensione di un float è platform-dependent (solitamente 64 bit con 14 digit di precisione [IEEE standard float])

String

- Una stringa in PHP è una serie di caratteri
  - un carattere ha le dimensioni di un byte
  - ci sono 256 configurazioni possibili di caratteri.
  - Non c'è supporto nativo per Unicode
- Esistono tre differenti modi di specificare una stringa:
  - **Single quoted**
  - **Double quoted**
  - **Heredoc sintax**

String (single quoted)

- `echo 'questa è una stringa single quoted';`
- `echo 'Questa è una stringa single quoted con Terminazione in nuova riga';`

- Gli unici caratteri "di escape" che PHP interpreta in una single quoted sono \' e \\
- echo 'Are you sure you want to delete C:*.*?'; //produce : Are you sure you want to delete C:*.*?

String (double quoted)

Double quoted string risolvono molti più caratteri di escape

sequence	meaning
\n	linefeed (LF or 0x0A (10) in ASCII)
\r	carriage return (CR or 0x0D (13) in ASCII)
\t	horizontal tab (HT or 0x09 (9) in ASCII)
\\	backslash
\$	dollar sign
\"	double-quote
\[0-7]{1,3}	the sequence of characters matching the regular expression is a character in octal notation
\x[0-9A-Fa-f]{1,2}	the sequence of characters matching the regular expression is a character in hexadecimal notation

String (heredoc sintax)

- Heredoc sintax è un formato stringa introdotto recentemente per salvaguardare la compatibilità con sistemi operativi eterogenei
- Sono interpretate come le double quoted ma non presentano il \r\n finale tipico dei sistemi Microsoft

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

Q?: Ha senso parlare di compatibilità tra sistemi nel mondo server-side?
    --Filosofia ("Proud to be Microsoft free")
    --Sguardi al futuro ("bundle di codice mobile")
```

String as array of char

- È consentito (ma sconsigliato) utilizzare la stringa come una array di caratteri
- È possibile quindi utilizzare una stringa come un vettore indicizzato di caratteri utilizzando l'indice di array tra { }

```
/* Get the first character of a string */
$str = 'This is a test.'; $first = $str{0};

/* Get the last character of a string. */
 $str = 'This is still a test.';
$last = $str{strlen($str-1)};
```

Arrays

- Viene mappato sempre come un array di object,
- L'indicizzazione è libera,
  - di default utilizza integer,
  - … ma la mappatura degli indici utilizza stringhe
- La sintassi per l'utilizzo è nome_array['stringa_indice']
  - è indifferente scrivere:
  - `$colore[0]="red";` oppure `$colore['red']="red";`
  - `$colore[1]="blue";` oppure `$colore['blue']="blue";`

Object

- …diamo per scontato cosa sia …
- Un oggetto si definisce prima come classe e si istanzia con il costruttore new

```
class triangolo {

   function area (){
       …
   }
}
…
//Instanziamone uno
$triagolo1 = new triangolo;
//Accediamo ad un metodo (l'unico in realtà)
Triangolo1 -->area();
```

Resource

- È un tipo che referenzia una risorsa esterna,
- Di tipo resource ad esempio sono:
  - le connessioni con i data base
  - il supporto per:
    - la connessione ftp o telnet
    - per la creazione di pdf,
    - per la chiamata a classi java,
    - per le socket,
    - per i file
    - …

NULL

- Introdotta da poco tempo (PHP 4.0)
- referenzia il tipo di una variabile nulla
- o di una variabile che referenzia un oggetto distrutto
- o di una costante al valore NULL

- Q:perchè?
  - per assurdo, definendo il nulla riusciamo a sapere cosa esiste e cosa no

3) Variabili

- Generiche
- Predefinite
- Scope delle variabili
- Variabili … variabili
- Variabili dall'esterno (from outside PHP)

Generiche

- Una variabile è definita in PHP dal $ seguito dal nome della variabile
- Il nome è case sensitive
- I nomi delle variabili possono cominciare con un carattere o con underscore _ ma non con un numero
- Una variabile punta al suo valore, è possibile copiare il riferimento es:

```
$uno='uno';
$due=$uno; //viene copiato il valore della variabile
$tre=&$uno; //viene copiato solo il riferimento, un cambiamento alla variabile
   $uno si propaga anche su $tre
```

Predefinite

- … provate ad utilizzare la funzione phpinfo();
- …
- http://sd.ingce.unibo.it/info.php

Variable scope

- Una variabile definita globale è visibile in tutte le funzioni,
- Una variabile definita internamente ad una funzione ha vita solo per la funzione stessa,

```
$a = 1; /* global scope */
function Test() {
echo $a; /* reference to local scope variable */
}
Test();
Questo script non produce output in quanto $a utilizzata dentro Test non è
   definita
Per utilizzare variabili globali dal corpo di funzioni occorre utilizzare lo
   statement global prima della dichiarazione
$a = 1;
$b = 2;
function Sum() {
global $a, $b;
$b = $a + $b;
}
Sum();
echo $b;
```

Variable Variables

- È comodo avere nomi di variabili dinamici, per utilizzarli occorre qualche accorgimento:

```
$a = "hello";
$$a = "world";
echo "$a ${$a}";  // Cosa produce in output ?
```

ATTENZIONE:

```
$$a[1]      // Array style - produce ambiguità , e non viene
   interpretato
Il parser non saprebbe distinguere tra ${$a[1]}  e ${$a}[1]
```

Variables from outside PHP

- Sono le più utilizzate nel mondo PHP
- PHP permette:
  - Di recuperare valori da form html
  - Di settare cookies
  - Di recuperare valori di dimensione (altezza e larghezza) da immagini
  - Di interagire con variabili esterne (da oggetti java o com)

Variables from HTML

- ...Vi ricordate all'inizio ... recuperare valori da stdin e scriverli su stout...

Nel momento in cui una form HTML viene spedita,
PHP rende disponibile tutti i valori dei campi della form in automatico, creando variabili globali con il nome del campo form e con il valore a cui sono state settate.

Variables from HTML (es)

```
<html>
<head>
</head>
<body>
<form name="form1" action="provaAcquisizione.php"
  method="GET">
<input type="text" name="campoNome">
<input type="radio" name="radio1" value="true">
</form>
</body>
</html>
```

Variables from HTML (es) provaAcquisizione.php

...in php recuperiamo il tutto semplicemente ...

```
<?
      echo “$campoNome”;
      echo “$radio1”;
?>
```

4) Costanti

- Le costanti non richiedono il simbolo di dollaro ($) anteposto al nome;
- Le costanti si possono definire e utilizzare in qualsiasi contesto indipendentemente dalle regole dello spazio dei nomi;
- Le costanti non si possono ridefinire o eliminare una volta che siano state definite;
- Le costanti possono contenere solo valori scalari.

Costanti (es)

```
define("COSTANTE", "Ciao mondo.");
echo COSTANTE; // stampa "Ciao mondo."
echo Costante; // stampa "Costante" e genera una
  notice.
```

P.S. una notice è un tipo di errore (tra i più lievi) generato dall’interprete PHP, in questo caso viene generato in quanto l’interprete si accorge che si cerca di spampare la stringa Costante in presenza di una costante di nome COSTANTE

Costanti predefinite

- Molto utili per il debug, sono :
  - __FILE__
  - __LINE__
  - PHP_VERSION
  - PHP_OS
  - TRUE
  - FALSE
  - NULL
  - E_ERROR
  - E_WARNING
  - E_PARSE
  - E_NOTICE
  - E_ALL

Costanti predefinti

```
function comunica_errore($file, $linea, $messaggio) {
 echo "È avvenuto un errore in $file alla linea $linea: $messaggio.";
}
comunica_errore(__FILE__, __LINE__, "Qualcosa è andato storto!");
```

5) Espressioni

Estratto da “PHP Applicazioni WEB” di Tobias Ratschiller e Till Gerken: “ The simplest yet most accurate way to define an expression is "anything that has a value". “

```
$a = 5
```

è un espressione e rifacendosi al concetto sopra l’utente non è chiamato a tenere traccia del tipo

Da qui la classificazione di PHP tra i linguaggi non tipati

N.d.r. Tobias Ratschiller è insieme a Rasmus Lerdorf il creatore di PHP, nonché fondatore di Zend Tecnologies

6) Operatori – precedenza

La seguente tabella fornisce una lista della precedenza degli operatori con gli operatori a più bassa precedenza listati prima.

Associatività	Operatori
sinistra	,
sinistra	or
sinistra	xor
sinistra	and
destra	print
sinistra	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
sinistra	? :
sinistra	||
sinistra	&&
sinistra	|
sinistra	^
sinistra	&
non associativi	== != === !==
non associativi	< <= > >=
sinistra	<< >>
sinistra	+ - .
sinistra	* / %
destra	! ~ ++ -- (int) (float) (string) (array) (object) @
destra	[
non associativi	new

Operatori Aritmetici

Esempio	Nome	Risultato
$a + $b	Addizione	La somma di $a e $b.
$a - $b	Sottrazione	La differenza di $a e $b.
$a * $b	Moltiplicazione	il prodotto di $a e $b.
$a / $b	Divisione	Quoziente di $a e $b.
$a % $b	Modulo	Il resto di $a diviso da $b.

Operatori di assegnamento

- L'operatore di base dell'assegnazione è "=" (assegna il valore a)

```
$a = ($b = 4) + 5; // $a è uguale a 9 ora, e $b è stato impostato a 4.


$a = 3;
$a += 5;


$b = "Ciao ";
$b .= "tutti";
echo $b  // produce Ciao tutti
```

Operatori Bitwise

Esempio	Nome	Risultato
$a & $b	And	Sono impostati ad ON i bit che sono ON sia in $a che in $b.
$a | $b	Or	Sono impostati ad ON i bit che sono ON in $a oppure in $b.
$a ^ $b	Xor	Sono impostati ad ON i bit che sono ON in $a oppure in $b na non quelli che sono entrambi ON.
~ $a	Not	Sono impostati ad ON i bit che sono OFF in $a, e viceversa.
$a << $b	Shift left	Sposta i bit di $a a sinistra di $b passi (ogni passo significa "moltiplica per due")
$a >> $b	Shift right	Sposta i bit di $a a destra di $b passi (ogni passo significa "dividi per due")

Es:

echo 12 ^ 9; // L'output è '5'

Operatori di Confronto

Esempio	Nome	Risultato
$a == $b	Uguale	`TRUE` se $a è uguale a $b.
$a === $b	Identico	`TRUE` se $a è uguale a $b, ed essi sono dello stesso tipo. (Solo PHP 4)
$a != $b	Diversi	`TRUE` se $a è diverso da $b.
$a <> $b	Diversi	`TRUE` se $a è diverso da $b.
$a !== $b	Non identici	`TRUE` se $a è diverso da $b, o se essi non sono dello stesso tipo. (Solo PHP 4)
$a < $b	Minore	`TRUE` se $a è strettamente minore di $b.
$a > $b	Maggiore	`TRUE` se $a è strettamente maggiore di $b.
$a <= $b	Minore o uguale	`TRUE` se $a è minore o uguale a $b.
$a >= $b	Maggiore o uguale	`TRUE` se $a è maggiore o uguale a $b.

Un altro operatore condizionale è l'operatore "?:" (o trinario), che opera come in C e molti altri linguaggi.

(espressione1) ? (espressione2) : (espressione3);

Operatore di controllo di errore

- 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.

```
$value = @$array[$indice];  // se $indice non esiste si procederà ugualmente
   nell'esecuzione


$my_file = @file ('file_inesistente') or die ("Apertura del file fallita:
   l'errore è '$php_errormsg'");
// in questo caso l'esecuzione verrà interrotta ma solo in quanto è stata
   imposta col comando die()
```

!!! Attenzione all'uso poiché questo inibisce anche i messaggi d'errore dell'interprete PHP!!!

Operatori di esecuzione

- 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

```
$output = `ls -al`;
echo "<pre>$output</pre>";
```

Questo predicato può però essere inibito a livello globale nelle impostazioni dell'interprete PHP

Operatori di incremento/decremento

Esempio	Nome	Effetto
++$a	Pre-incremento	Incrementa $a di una unità, inoltre restituisce $a.
$a++	Post-incremento	Restituisce $a, inoltre incrementa $a di una unità.
--$a	Pre-decremento	Decrementa $a di una unità, inoltre restituisce $a.
$a--	Post-decremento	Restituisce $a, inoltre decrementa $a di una unità.

Operatori logici

Esempio	Nome	Risultato
$a and $b	And	**TRUE** se entrambi $a e $b sono **TRUE**.
$a or $b	Or	**TRUE** se uno tra $a o $b è **TRUE**.
$a xor $b	Xor	**TRUE** se uno tra $a o $b è **TRUE**, ma non entrambi.
! $a	Not	**TRUE** se $a non è **TRUE**.
$a && $b	And	**TRUE** se entrambi $a e $b sono **TRUE**.
$a || $b	Or	**TRUE** se uno tra $a o $b è **TRUE**.

Operatori di stringa

- Ci sono due operatori di stringa:
  - Concatenazione
    - .
  - Assegnazione concatenata
    - .=

```
$a = "Ciao ";
$b = $a . "Mondo!"; // ora $b contiene "Ciao Mondo!"


$a = "Ciao ";
$a .= "Mondo!"; // ora $a contiene "Ciao Mondo!"
```

7) Strutture di controllo

- Le stesse del C, del Java, e qualcuna del PERL
  - Esecuzione condizionata (If , else , elseif)
  - Esecuzione ciclica condizionata (while, do … while, for, foreach, …)
  - Terminazione di esecuzione (break, continue, return)
  - Controllo alternativo (switch)
  - Controllo presenza file di specifiche (require, include, require_once, include_once)

8) Funzioni

- Una funzione può essere definita usando la seguente sintassi:

```
function esempio ($arg_1, $arg_2, ..., $arg_n) {
 echo "Funzione di esempio.\n";
 …
 return $retval;
}
```

- All'interno di una funzione può apparire qualunque codice PHP valido:
  - persino altre funzioni e definizioni di classe.

Funzioni – passaggio argomenti

- Il passaggio degli argomenti può avvenire in due modi:
  - Per valore
  - Per riferimento

```
function prende_array($input) {   //per valore
echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}


function aggiungi_qualcosa(&$string) {  //per riferimento
$string .= 'e qualche altra cosa.';
}
```

Funzioni – passaggio argomenti

- È inoltre possibile e utile specificare valori di default per gli argomenti nel caso non siano stati inizializzati

```
function fare_lo_yogurt ($gusto, $tipo = "yogurt") {
 return "Fare una vaschetta di $tipo a $gusto.\n";
 }
echo fare_lo_yogurt ("fragola");
```

ATTENZIONE

```
function fare_lo_yogurt ($tipo = "yogurt", $gusto) {
return "Fare una vaschetta di $tipo a $gusto.\n";
}
echo fare_lo_yogurt ("fragola");
```

Valori restituiti

- I valori vengono restituiti usando l'istruzione opzionale return.
- Può essere restituito qualsiasi tipo, incluse liste ed oggetti.

```
function return_Array() {
return array (0, 1, 2);
}
list ($zero, $uno, $due) = return_Array();


function &restituisce_riferimento() {
 return $un_riferimento; }
$nuovo_riferimento =& restituisce_riferimento();
```

Funzioni variabili

- PHP supporta il concetto di funzioni variabili.
- Ciò significa che se un nome di variabile ha le parentesi accodate ad esso, PHP cercherà una funzione con lo stesso nome del valore della variabile, e cercherà di eseguirla.

```
function uno() {
echo "In uno()<br>\n";
}
function due($arg = '') {
echo "In due(); l'argomento era '$arg'.<br>\n";
}
$func = 'uno';
$func();
$func = 'due';
$func('test');
```

9) Classi ed oggetti

- Fino a PHP release 4.0
  - In ambito PHP non si parla di OOP
  - Una classe in PHP è una collezione di variabili e funzioni che utilizzano queste variabili
  - Risultano comode per organizzare meglio il codice e per renderlo intelligibile a terzi
  - Ogni variabile di rappresentazione interna della classe, si dichiara preceduta dal costrutto **`var`**
  - Perché una classe possa accedere alle sue variabili di rappresentazione interna, si deve utilizzare **`$this`** come riferimento a se stessa

Classi - sintassi

```
<?
class  ragazzo {
var $nome;
var $cognome;


   function set_nome($name){
        $this-->nome=$name;
   }


   function set_cognome($surname){
        $this-->nome=$surname;
   }


}
?>
```

Classi e costruttori

- In PHP 4 e 5, una funzione diventa un costruttore, quando ha lo stesso nome di una classe ed è definita all'interno della classe stessa
- In PHP 3, una funzione si trasformava in un costruttore quando aveva lo stesso nome di una classe.

```
<?
//riferita alla classe di prima
 function ragazzo (){
 $this->nome=NULL;
 $this->cognome=NULL;
 }
?>
```

Classi - estensione

- PHP fornisce supporto per l'ereditarietà ma esclusivamente in forma semplice,
  - PHP 4.0 non supporta l'ereditarietà multipla

```
<?
 class studente extends ragazzo{
 var $studio;


 function setCorso($corso){
  $this-->studio=$corso
 }
}
?>
```

Ridefinizioni di metodi

- E possibile ritrovarsi a scrivere classi con codice che si riferisce a variabili e funzion classi base. Si utilizza il comando parent per accedere ai metodi della classe base,

```
class A {
function example() {
echo "Sono A::example() e fornisco una funzionalità di base.<br>\n"; }
}
class B extends A {
function example() {
 echo "Sono B::example() e fornisco una funzionalità aggiuntiva.<br>\n";
 parent::example(); }
}
$b = new B;
```

Classi - staticità

- Può tornare comodo eseguire un metodo di una classe senza doverla istanziare.
- Ciò si può eseguire tramite l'operatore **::**

```
class A {
function example() { echo "Sono la funzione originale A::example().<br>\n"; }
}
class B extends A {
function example() { echo "Sono la funzione ridefinita B::example().<br>\n";
A::example(); }
}
A::example();
$b = new B;
$b->example();
```

!!! $this non può essere usato !!!

Perché PHP

- Perché è comodo
- Perché è intuitivo
- Non richiede troppa configurazione
- … è molto orientato al web …
- … è un linguaggio di scripting
- Ha una grossa base funzionale
- … per chè si trova molta roba già sviluppata
- Fornisce supporto per la maggioranza dei data base
- Ha meccanismi di sicurezza (le sessioni)
- Si costruisce a classi
- Funziona sulla maggioranza dei server web

PHP handling session

- Come abbiamo già detto le CGI hanno introdotto il concetto di sessione applicato al http
- In PHP le sessioni sono meccanismi per la sincronizzazione e il mantenimento dello stato passando da una pagina all'altra
- Servono per mantenere dati tra un accesso e l'altro
- Tecnicamente la sessione PHP è rappresentata da un file memorizzato sul server dove vengono memorizzate gli stati di tutte le variabili che sono attive nel contesto di una connessione
- Cambiando pagina le variabili rimangono così attive e possono esservene aggiunte altre

PHP - sessioni

- Essendo un meccanismo server side deve essere configurato l'interprete per abilitarne il supporto
- Occorre però gestirle per evitare garbage sulla macchina server
- Vengono tipicamente utilizzate per gestire ambiti in cui è richiesta sicurezza e dove non si voglia ricorrere all'emissione di Cookies (che ricordiamo essere immagazzinati client side)

PHP - Sessioni

Generazione dell'ID di sessione

Esistono vari modi, il più utilizzato è:

```
srand( (double)microtime() * 1000000 );
$sessionId = md5(uniqid(rand()));
```

La prima riga genera un numero random utilizzando l'orologio di sistema con una precisione di un double

La seconda assume il numero random generato come ID di sessione codificandolo in chiave md5

PHP - Sessioni

- Una volta creato l'ID di sessione, nel momento in cui si voglia proteggere dati sotto sessione occorre segnalarlo all'interprete utilizzando la funzione session_start();
- Per inserire una nuova variabile si usa session_register();
- Per distruggere la sessione (operazione consigliata alla fine per eliminare garbage sul server) si usa session_destroy();

PHP - Sessioni

- Nel caso si utilizzino oggetti, questi possono essere reinterpretati dalla sessione solo se l'oggetto è stato serializzato (serialize();)
- Per serializzare un oggetto occorre che la definizione di classe sia presente sia nella pagina di partenza che in quella di arrivo
- In PHP 4 la serializzazione degli oggetti passati in sessione avviene in automatico, ma è sempre buona norma includere lo stesso i file con le definizioni di classe

PHP esempi

```
function connectMy (){
//----------------------
$dbUsername = "root";
$dbPassword = "preferita";
$dbHostname = "localhost";
$dbDataName = "DatabaseName";
$msg1 = "<b>ERRORE</b>. Ci sono problemi sul server.<br>\n";
$msg2 = "<b>ERRORE</b>. Ci sono problemi nel database.<br>\n";


mysql_connect($dbHostname, $dbUsername, $dbPassword) or die ($msg1);
mysql_select_db($dbDataName) or die ($msg2);
}


$connessione=connectMy();   // Cosa ritorna questa funzione ? Come posso riutilizzarla ?
```

Php Esempi:

```
function generate_session_id (){
//--------------------------------
srand( (double)microtime() * 1000000 );
$sessionId = md5(uniqid(rand()));
session_id($sessionId);
}


//una volta chiamata questa funzione posso far partire le sessioni direttamente con
session_start();
```

HTML – PHP esempi

```
<html><head><title>Prova PHP – HTML</title></head>
<body>
<form method="POST" action="filephp.php">
<table width="80%" align="center" border="1">
    <tr>
      <td>Login</td>
<td><input type="text" name="login"></td>
      <td>Password</td>
      <td><input type="password" name="pw"></td>
      <td><input type="submit" name="Submit" value="Invia"></td>
    </tr>
</table>
</form>
</body>
</html>
```

HTML – PHP esempi

```
srand( (double)microtime() * 1000000 );
$sessionId = md5(uniqid(rand()));
session_id($sessionId);
session_start();
require ("./funzioni.php");
$sql="SELECT utenti.* FROM utenti where (utenti.username='";
$sql.=$login;
$sql.="')";
$connessione=connectMy ();
$result = mysql_query ($sql) or die ("Invalid query");
$arrayResult=mysql_fetch_array($result);
if ($arrayResult["pw"]==$pw){//accesso assicurato abilito l'utente alle funzioni }
else {session_destroy();}
mysql_close($connessione);
```