- **PHPRATICO**
- L'alternativa open source per lo scripting server-side. Porzioni di codice per la programmazione web oriented.
- di: Claudio Garau

PHP: di cosa parliamo

- Il termine PHP nacque come acronimo di "Personal Home Page"; ora significa PHP: Hypertext Preprocessor, infatti PHP "pre-processa" attraverso il suo motore (Zend) istruzioni contenute in ipertesti.
- E' un linguaggio di scripting server side, cioè un codice che attraverso istruzioni determina comportamenti.
- Il campo di applicazione più importante è il Web, grazie a PHP è possibile produrre dinamicamente codice HTML leggibile tramite browser.
- Con PHP, creiamo scripts da inserire nei siti internet: guestbook, forum di discussione, chat, webmail, CMS, gestori di mailing, e-commerce.
- PHP automatizza i processi di interazione tra le pagine web, il Webmaster e i visitatori.

Un po' di storia: le origini

- 1994: Rasmus Lerdorf, creò PHP per controllare gli accessi al un suo C.V. on-line. Nacque “PHP Tools”, presto sostituito dal “PHP/FI” (Personal Home Page/Form Interpreter), dotato di CGI per interpretare pagine HTML.
- Lerdorf riscrisse il codice PHP in modo che potesse interagire col database miniSQL.
- PHP venne distribuito su licenza Open Source, chiunque poteva partecipare al suo sviluppo; una vasta comunità di programmatori si unì a Lerdorf
- 1998: nasce PHP3 dalla collaborazione tra Zeev Suraski e Andi Gutmans. Venne implementato per interfacciarsi ad Apache, il Web server più diffuso della Rete, e per interagire col database server MySQL.

Un po' di storia: dal 2000 ad oggi

- Nel 2000 venne rilasciato PHP4, un linguaggio dotato di un nuovo motore, "Zend".
- Con PHP4 si possono produrre scripts in grado di interfacciarsi su svariati tipi di database come: MySQL, Access, Postgres, Sql, Oracle e di lavorare su tutti gli SO più diffusi come: Linux, Unix, Windows, Solaris e Mac.
- 2004: nasce la versione 5 di PHP. Integra un nuovo motore, “Zend 2”; un'architettura maggiormente predisposta alla programmazione orientata agli oggetti (OOP) e un mini-db chiamato SQLlite.

Perché utilizzare codice PHP

- E' facile da imparare, da scrivere e da modificare.
- Permette di essere operativi in poco tempo.
- E' HTML - embedded
- E' supportato da una comunità di sviluppatori composta da migliaia di persone.
- Ampia disponibilità di tutorials gratuiti su Internet.
- Milioni di scripts pronti all'uso scaricabili gratuitamente dalla rete.
- PHP è multipiattaforma
- E' Open Source.

Client side e Server side

Un Web server è un software destinato ad elaborare pagine web contenenti scripts server side.

Client Side: html

- Il Web server riceve la richiesta di una pagina .html da un client.
- Interpreta e riconosce la richiesta.
- Cerca e trova la pagina all'interno del server.
- Invia la pagina al browser del client.

Server Side: PHP

- Il motore PHP si occupa della creazione della pagina richiesta.
- Il Web server riconosce la richiesta del client.
- Cerca e trova la pagina all'interno del server.
- Esegue le istruzioni all'interno del codice.
- Invia la pagina al browser del client.

Cosa ci serve per lavorare

- Un comune PC casalingo
- Un SO (ad es.: Linux)
- Un editor di testo
- Un Web server (ad es.:Apache);
- Una versione di PHP;
- Il database server MySQL.

Sintassi fondamentale

- Il codice PHP deve essere contenuto in una pagina con estensione “.php” (esempio: “info.php”), in caso contrario il Web server non sarebbe in grado di interpretarlo.
- Le istruzioni PHP devono essere contenute all'interno dei tags “<?php" e "?>”, delimitatori, grazie ad essi diremo al Web server: “Attenzione! All'interno di questi tags vi sono delle istruzioni PHP”.
- I delimitatori possono seguire una sintassi che ricorda quella del JAVASCRIPT: “<script language="php">codice</script>”.
- Ogni istruzione PHP deve essere seguita da un punto e virgola (;).

Sintassi comune

```
<?php
phpinfo()
?>
```

“Short tags” abbreviati

```
<?
phpinfo();
?>
```

“Short tags” in stile Asp

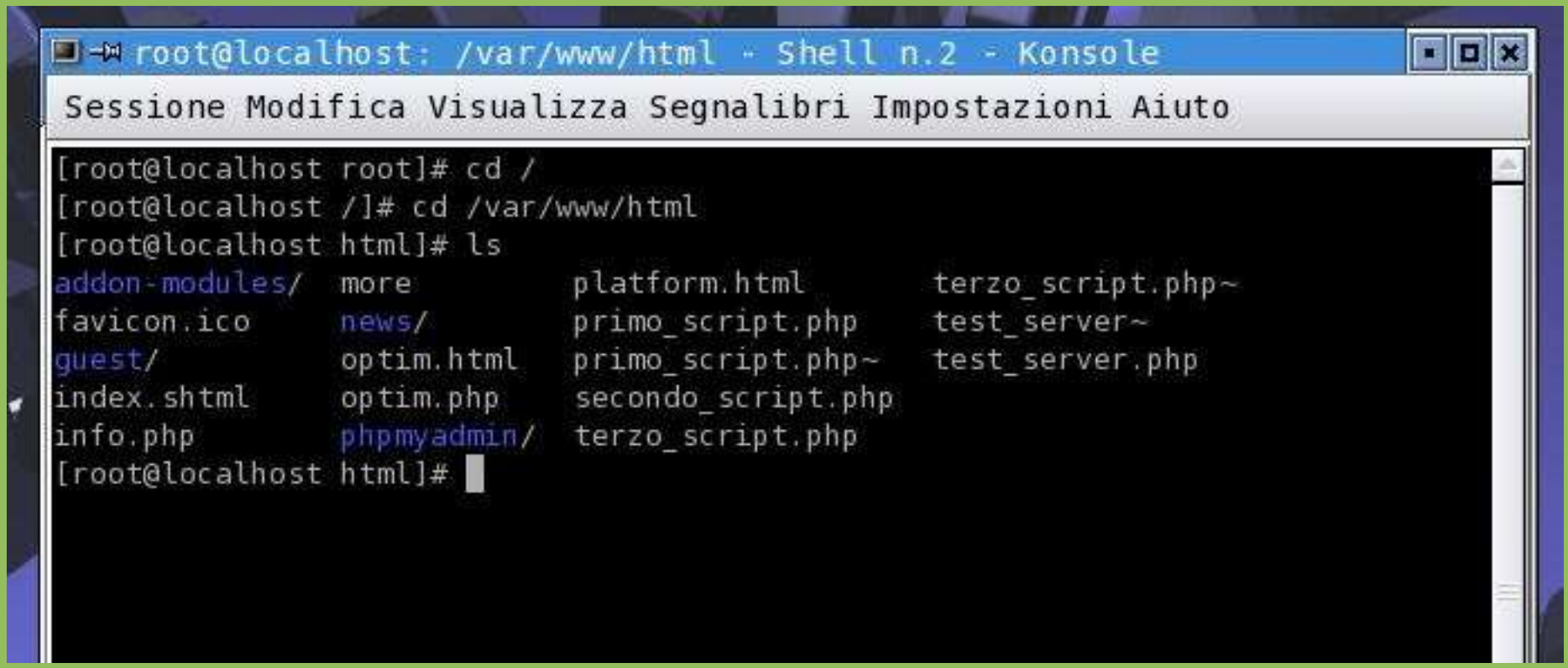
```
<%
phpinfo();
%>
```

PHP info

PHP Version 4.3.4

System	Linux localhost 2.6.3-7mdk #1 Wed Mar 17 15:56:42 CET 2004 i686
Build Date	Mar 22 2004 21:23:38
Configure Command	'./configure' '--prefix=/usr' '--exec-prefix=/usr' '--bindir=/usr/bin' '--sbindir=/usr/sbin' '--sysconfdir=/etc' '--datadir=/usr/share' '--includedir=/usr/include' '--libdir=/usr/lib' '--libexecdir=/usr/lib' '--localstatedir=/var/lib' '--sharedstatedir=/usr/com' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--enable-discard-path' '--disable-force-cgi-redirect' '--enable-shared' '--disable-static' '--disable-debug' '--disable-rpath' '--enable-pic' '--enable-inline-optimization' '--enable-memory-limit' '--with-config-file-path=/etc' '--with-config-file-scan-dir=/etc/php' '--with-pear=/usr/share/pear' '--enable-magic-quotes' '--enable-debugger' '--enable-track-vars' '--with-exec-dir=/usr/bin' '--with-versioning' '--with-mod_charset' '--with-regex=php' '--enable-track-vars' '--enable-trans-sid' '--enable-safe-mode' '--enable-ctype' '--enable-ftp' '--with-gettext=/usr' '--enable-posix' '--enable-session' '--enable-sysvsem' '--enable-sysvshm' '--enable-yp' '--with-openssl=/usr' '--without-kerberos' '--with-ttf' '--with-freetype-dir=/usr' '--with-zlib=/usr' '--with-zlib=/usr' '--with-zlib-dir=/usr' '--without-pear'
Extensions listed here are (or will be soon) available as external modules. To install one or all of these, use "urpmi" php-EXTENSION_NAME	mysql pgsql sqlite gd imap ldap bcmath bz2 calendar cpdf crack curl cyrus db dba dba_bundle dbase dbx dio domxml exif fbsql fdf filepro fribidi gmp hwapi hyperwave iconv imagick informix ingres_ii interbase ircg java mbstring mcal mcrypt mcve mhash mime_magic ming mnogosearch msession msql mssql ncurses notes oci8 odbc oracle overload ovrimos pam_auth pcntl pdf pfpro pspell qtdom readline recode rrdtool shmop snmp smbauth sockets swf sybase sybase_ct sysvmsg tokenizer wddx xml

Dove lavoriamo



Con cosa lavoriamo: le variabili

- Una variabile è una zona di memoria destinata a contenere delle informazioni.
- Le variabili sono dei "contenitori" nei quale possiamo inserire dei dati: testo, valori numerici, riferimenti a file e altro.

```
<?php
//diamo alla variabile $var il valore di un numero intero
$var=1000;
//istruzione
print $var."<br />";
//ora riassegnamo alla variabile $var il valore di una stringa di testo
$var="Zola";
//istruzione
echo $var;
?>
```

Gli array

- Gli **array** (**vettori**), sono un particolare tipo di variabili a cui è associato più di un valore. Se alla variabile $n associamo "pari" abbiamo dichiarato il valore di una variabile comune, se invece a $n associamo "pari" e "intero", abbiamo dichiarato i valori di un array.
- PHP è in grado di associare ad ogni valore assegnato un **indice numerico** a partire da 0 (numeri indice).

```
<?php
$nano = array("Brontolo", "Cucciolo", "Dotto", "Eolo",
"Gongolo", "Mammolo", "Pisolo");
{ print "Biancaneve si sposa con $nano[3]."; }
?>
```

```
<?php
$nano = array(1=>"Brontolo", "Cucciolo", "Dotto",
"Eolo", "Gongolo", "Mammolo", "Pisolo", "amica" => "Biancaneve");
{ print "$nano[amica] si sposa con $nano[6]"; }
 ?>
```

I tipi di dato

- Numeri **interi** (integer)
- Numeri in **virgola mobile** (*Floting point*).
- **Stringhe**.
- Valori **logici** o **booleani**.
- Valori di tipo **NULL**.
- **Array** (variabile contenente più valori)
- Gli **oggetti**: elementi appartenenti ad una **classe**.
- Le **risorse**: files o connessioni a database.

Gli operatori

- Gli operatori sono “**costrutti nativi**”, forniti dal PHP stesso. Ogni operatore lavora con uno o più tipi di dati.
- Operatori **aritmetici**
- Operatori di **assegnamento**
- Operatori di **confronto**
- Operatori **logici**

Esempio di operatore aritmetico

L'operatore di **modulo**, simboleggiato dal **segno percentuale** (“**%**”), effettua una divisione tra i valori delle variabili calcolando **il resto** della divisione.

```
<?php
$dividendo = 100;
$divisore = 3;
$resto = $dividendo  %  $divisore;
echo $resto;
/*il risultato sarà uguale a 1*/
?>
```

Esempio di operatore di assegnamento

Con l'operatore “**.=**” viene concatenata la variabile a sinistra con la variabile alla destra del simbolo.

```
<?php
$var = "Linux";
$iabile = "Day";
$var . =  $iabile;
print $var;
/*il valore della stringa $iabile viene concatenato a $var,
verrà stampato a video il termine "LinuxDay"*/
 ?>
```

Esempio di operatore di confronto

L'operatore di uguaglianza “==”, restituisce TRUE quando l'elemento alla destra del simbolo ha un valore uguale a quello di sinistra.

SE

$mele = 9; $pere = 10;

→

allora

$mele = = $pere

restituisce FALSE

SE

$mele = 9; $pere = 9;

→

allora

$mele = = $pere

restituisce TRUE

Esempio di operatore logico

L'operatore **AND** ha come simboli “And” o “&&” restituisce TRUE quando entrambi gli argomenti sono TRUE.

SE

530 > 2 And 6 < 600

allora

restituisce **TRUE**, entrambi gli argomenti sono veri.

Istruzioni condizionali: “ if ”

- Le istruzioni condizionali permettono di controllare l'esecuzione di uno script
- E' possibile determinare il numero di volte che queste operazioni andranno ripetute e i casi in cui esse dovranno essere eseguite.

```
<?php
$SO_gratuito = “nonwindows”;
$linux = $SO_gratuito;
//introduciamo "if" e una condizione
if ($linux  == “nonwindows”)
//procediamo con le istruzioni tra parentesi graffe
{ print  “<h3>” . “Installo Linux” . “</h3>” . “<br />”;
print  “<h2>” . “Almeno è gratis” . “</h2>” . “<br />”; }
//questa istruzione verrà eseguita comunque
print  “<h1>” . “Meglio un SO gratuito che funziona di uno a pagamento che non funziona.”  .  “</h1>”;
?>
```

Istruzioni condizionali: “ if ” ed “ else ”

Utilizzare di "**if**" ed "**else**" in uno stesso script, permette di indicare le istruzioni che desideriamo eseguire quando la condizione è vera e di imporre un'alternativa se la condizione si dimostra falsa.

```
<?php
$prezzo_windows = 200;
//stabiliamo la condizione
 if ($prezzo_windows<=2)
{ print "<h1>" . "Compro" . "</h1>"; }
//introduciamo un'alternativa
 else {
 print "<h1>" . "Installo Linux." . "</h1>"; }
 ?>
```

I cicli: “for”

- Grazie ai cicli possiamo eseguire ripetutamente determinate operazioni.
- Consistono nel ripetere dei comandi per un determinato numero di volte.

```
<?php
for ($tabellina =1; $tabellina <= 10; $tabellina++)
{
$stampa = 2* $tabellina;
print("2*$tabellina = $stampa<br />");
}
?>
```

I cicli: "while"

Il ciclo **while** ha una funzione simile a quella di una "if ripetuta.

```
<?php
$n_maialetti=2;
$posso_mangiarne=10;
//introduciamo un ciclo "while" seguito da una  condizione
 while($n_maialetti <= $posso_mangiarne)
//istruzione
{ print ("se mangio $n_maialetti maialetti, posso mangiarne
ancora " . ($posso_mangiarne - $n_maialetti) . "<br />");
//stabiliamo l'incremento da ripetere ciclicamente
$n_maialetti++; }
?>
```

I cicli: il " Loop"

- Un particolare effetto dei cicli mal concepiti è il **loop**.
- Si verifica quando un ciclo non trova una via d'uscita e riesegue iterazioni all'infinito.

```
<?php
//sbagliamo  il nome della variabile nell'incremento
for ($stringhe=1; $stringhe<8; $maialetti++)
{ print "Questa è la riga numero $stringhe"; }
 ?>
```

Funzioni

- Sono delle **istruzioni** con il compito di eseguire determinate operazioni.
- **Funzioni native: c**ostrutti elementari incorporati e predefiniti nel linguaggio PHP.
- **Funzioni personalizzate** (o **proprie**): definite dal programmatore in sede di scrittura del codice.
- Con esse è possibile compiere più volte uno stesso tipo di operazione senza dover riscrivere ogni volta il codice corrispondente.
- Una volta introdotta una funzione, è sufficiente richiamarla fornendole dei **parametri** (dati) da elaborare.

Funzioni: gestione delle variabili

Esempio: **isset()** restituisce TRUE o FALSE dopo aver verificato se una variabile è stata definita o meno.

```
<?php
$n =  10;
if (isset($n));
{ print "La variabile è stata definita."; }
 ?>
```

La funzione "**isset()**" verificherà che la variabile $n sia definita e restituirà TRUE. Infatti, abbiamo definito la variabile prima di introdurre la funzione assegnandole il valore numerico "10".

Funzioni: gestione delle stringhe

Esempio: **strrev()** inverte l'ordine dei caratteri di una stringa.

```
<?php
$var="LinuxDay";
$risultato=strrev($var);
{ print $risultato; }
 /*verrà stampata la striga "yaDxuniL"*/
?>
```

Funzioni: gestione di array

Esempio: **array_values()** restituisce tutti i valori di un array e li indicizza numericamente.

```
<?php
$alieno = array("disco" => "volante", "pelle" =>
"verde");
print_r (array_values ($alieno));
/*restituisce Array ( [0] => volante [1] => verde )*/
?>
```

Funzioni: gestione delle date

Le funzioni predefinite destinate alla gestione delle date si basano sul **timestamp**, un numero intero in uso sui sistemi operativi UNIX (o basati su UNIX come Linux), che rappresenta il numero di secondi trascorsi a partire dal 1° gennaio del 1970.

```
<?php
//mostriamo ora e data odierna
 print "Sono le ore " . date("H:i:s") . " del giorno " . date("d/m/Y");
?>
```

Inviare e-mail con PHP

PHP utilizza la funzione: **mail()**, che raccoglie i dati (i valori delle variabili interessate) e li spedisce via SMTP (**S**IMPLE **M**AIL **T**RANSFER **P**ROTOCOL).

```
<?php
$messaggio = "Saluti dal LinuxDay 2004";
$destinatario = "suamail@suamail.it";
$oggetto = "Mail in PHP";
$intestazione = "Inviata da: $mittente";
if (mail ($destinatario, $oggetto, $messaggio, $intestazione));
{ print "Mail inviata correttamente."; }
?>
```

MySQL: connessione

- MySQL è un software per gestire db (**DBMS**: **D**ata**B**ase **M**anagement **S**ystem).
- Appartiene alla famiglia dei db **SQL server** (**S**tructured **Q**uery **L**anguage).
- Caratteristiche: "multi - processo", "multi - utente", "veloce" e "robusto".

```
//assegnamo i valori alle variabili
$host = 'localhost';
$user = 'admin';
$pwd = 'password';
 //stabiliamo la connessione
$connessione = mysql_connect ($host, $user, $pwd)
or die ("Impossibile connettersi al server host.");
```

MySql: creazione del db

- Comando SQL **CREATE DATABASE**.
- Funzione **"mysql_query()"**.

```
//stabiliamo la connessione con MySQL
$connessione = mysql_connect($host, $user, $pwd)
or die ("Impossibile connettersi al server host");
//formuliamo la query e passiamo il risultato  alla funzione
$query = "CREATE DATABASE linuxday";
mysql_query($query, $connessione)
```

MySql: selezione del db

Per poter lavorare con MySql, una volta connessi dobbiamo selezionare il db desiderato.

```
$host = 'localhost';
$user = 'admin';
$pwd = 'password';
$db = 'linuxday';
//connessione
$connessione = mysql_connect($host, $user, $pwd)
or die ("Impossibile connettersi al server host.");
//selezioniamo il database
mysql_select_db ($db, $connessione)
or die ("Impossibile connettersi al database $database");
```

La programmazione ad oggetti in PHP (OOP)

- **Programmare ad oggetti:** suddividere un problema in parti indipendenti; creare **oggetti** con cui poter operare separatamente.
- Oggetto: **entità logica con identità univoca** contenente sia le informazioni sia il codice destinato a manipolare le informazioni stesse.
- Davanti ad un nuovo problema è sufficiente intervenire sugli oggetti coinvolti nel cambiamento.
- La OOP, si differenzia dalla "**metodologia procedurale**" che adatta gli applicativi mano a mano che si presentano nuovi problemi.

Caratteristiche di un oggetto

- Ogni oggetto è "univoco" in quanto ha un nome che è unico all'interno dell'applicativo di cui fa parte.
- Oggetti dotati di caratteristiche simili possono essere raggruppati in **classi**. Le **classi** sono dei tipi di dato dei quali fanno parte oggetti della stessa natura.
- Le classi definiscono: gli **attributi** (**proprietà**), dati salvati nello spazio di memoria che gli oggetti contengono.
- Le classi definiscono: i **metodi**, le funzioni eseguibili da un oggetto sugli attributi.
- Gli oggetti sono le **istanze** di una classe. Tra un oggetto è una classe intercorre la stessa relazione che c'è tra una variabile ed il suo tipo di dato.

Concetti basilari dell'OOP

- **Incapsulazione**: rende "invisibili" dettagli sull'implementazione di una classe; viene impedito l'accesso ad essi da parte di frammenti di codice esterni. In sede di scrittura del codice non è necessario sapere in che modo un oggetto sia stato realizzato, ma è sufficiente conoscerne metodi e attributi.
- **Ereditarità**: consente di creare **gerarchie** nelle classi. Data una classe generale, **classe base** (**superclasse**), che definisce le peculiarità di un deteminato insieme, essa potrà essere **ereditata** da **sottoclassi** (**derivate**), che aggiungeranno alla classe base i loro elementi specifici.
- **Polimorfismo:** possiamo dar vita a comportamenti differenti utilizzando gli stessi metodi su oggetti diversi. Si basa sulla possibilità di ridefinire in sottoclassi il comportamento dovuto a uno o più metodi ereditati da una superclasse. Avremo metodi capaci di agire su oggetti di diversa natura, lasciando all'ambiente in cui "gira" l'applicativo la decisione di utilizzare metodi diversi a seconda delle operazioni da svolgere.

Risorse per PHP

http://www.php.net/manual/it/ (manuale ufficiale)
http://www.php.net/downloads.php (versioni PHP)
http://www.zend.org (tutorials, scripts)
http://www.hotscripts.com/PHP/index.html (Scripts e articoli)
http://www.phpfreaks.com (PHP Help Center)
http://www.apache.org/ (Web server)
http://www.mysql.com/ (DBMS)
http://www.claudiogarau.it (Scusate lo spam :-))