

Informatica **U**manistica

# Introduzione a PHP

Laboratorio di Progettazione Web

*AA 2009/2010*

*Chiara Renso*

*ISTI- CNR - [c.renso@isti.cnr.it](mailto:c.renso@isti.cnr.it)*

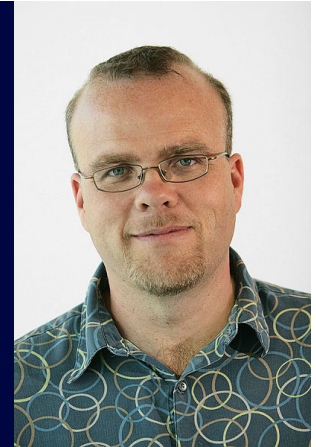


UNIVERSITÀ DI PISA

# Il linguaggio PHP

- ◆ Il linguaggio PHP (PHP Hypertext Preprocessor) è un linguaggio di script lato server, viene cioè interpretato da una componente aggiuntiva del server web. E' generalmente impiegato per applicazioni web.
- ◆ E' un linguaggio free open source, liberamente scaricabile da [www.php.net](http://www.php.net) e supportato da numerose comunità online.
- ◆ E' indipendente dalla piattaforma
- ◆ Permette di interagire con vari database, tipicamente MySQL ma anche Oracle, Postgres e molti altri.
- ◆ E' integrabile con numerose librerie esterne (grafica, mail, pdf etc)
- ◆ Per l'installazione possiamo usare un kit, come Wamp (windows), XAMMP (Linux o Windows) , MAMP (MacOS) o EasyPHP (Windows). Per una guida ai kit di installazione su Windows <http://php.html.it/articoli/leggi/2568/php-apache-e-mysql-su-windows-in-un-clic/1/>

# PHP



Pagine facili da creare e non necessitano di compilazione.

PHP è un linguaggio di scripting server side free opensource

E' stato introdotto da Lerdorf nel '94

Le pagine PHP sono completamente integrate con l'HTML.

Il linguaggio di scripting è meno strutturato di un linguaggio di programmazione classico.

# Caratteristiche di PHP

- 1) Indipendente dalla piattaforma (Windows, MacOS, Linux)**
- 2) Necessita di un Webserver (Apache, IIS, ...)**
- 3) Possibilità di connessione a molti database (Oracle, MySQL, Postgres, Access,....)**
- 4) La versione attuale è la 5**
- 5) E' uno dei linguaggi lato server più usati al mondo, è installato su più di 20 milioni di websites (Wikipedia)**

# Installare PHP-MYSQL

- ◆ Per poter utilizzare PHP con il database MySQL (configurazione tipica per le applicazioni web) occorre installare un server web, PHP e MYSQL.
- ◆ L'installazione integrata di queste tre componenti non è sempre agevole, per questo motivo sono stati resi disponibili in rete dei **kit di installazione integrati** (as es. EasyPHP per Windows, MAMP per MACOS e XAMMP per Linux)
- ◆ [http://www.hotscripts.com/PHP/Software\\_and\\_Servers/Installation\\_Kits/](http://www.hotscripts.com/PHP/Software_and_Servers/Installation_Kits/)
- ◆ L'installazione tipica di PHP è LAMP, che sta per Linux, Apache, MySQL, PHP

# Una pagina PHP

Una pagina PHP è un file di testo con estensione **.php** e contiene:

- ◆ Testo
- ◆ Marcatori HTML
- ◆ Comandi script

# Una pagina PHP

In PHP la parte di script può apparire in qualsiasi punto della pagina html e viene delimitato dai *delimitatori di script*:

◆ `<?php e ?>`

Ad esempio il comando:

◆ `<?php $x="ciao" ?>`

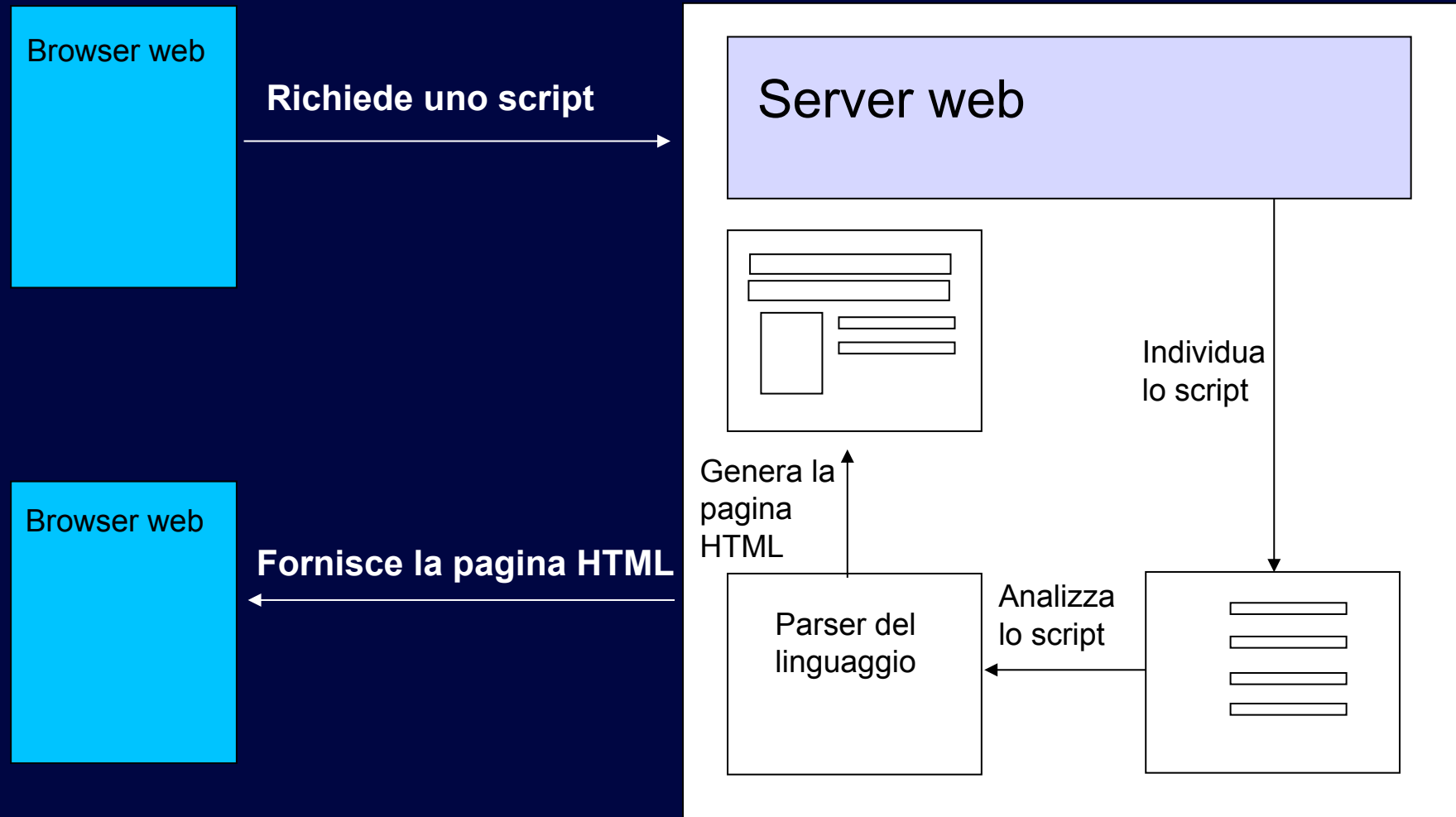
assegna alla variabile x il valore "ciao".

## Una pagina PHP

- ◆ Quando il web server riceve la richiesta di una pagina con estensione **.php** sa che la pagina deve essere processata dal modulo PHP. Il risultato della computazione viene tornato al web server che lo restituisce al client.
- ◆ Gli elementi HTML sono passati direttamente al server web mentre le parti di script vengono interpretate dal motore di scripting PHP
- ◆ In un file PHP le parti di script si alternano e si integrano al codice HTML



# Funzionamento di PHP



# Pagine PHP sul server

- ◆ Le pagine php, come le HTML, risiedono sul web server. Apache ha una cartella predefinita per le pagine HTML/PHP che è `htdocs`, nella cartella di installazione di apache.
- ◆ Se vogliamo eseguire la pagina *miapagina.php* dobbiamo quindi copiarla nella cartella `htdocs` e poi visualizzarla digitando dal browser la URL

`http://localhost/miapagina.php`

Nome del server web  
dove risiede la pagina

# File di configurazione di PHP: php.ini

- ◆ I parametri di funzionamento di PHP sono definiti in un apposito file, denominato **php.ini** che il server web legge ad ogni riavvio
- ◆ In questo file sono definiti alcuni parametri con i valori di default
- ◆ Non è necessario modificare il file per il corretto funzionamento di PHP, i parametri predefiniti generalmente sono sufficienti
- ◆ I parametri riguardano molti degli aspetti di PHP, ad esempio path dei file, uso della sessioni e dei cookie...

# Anatomia di uno script PHP

La pagina deve essere salvata con estensione .php

HTML – parte statica della pagina

```
<HTML><HEAD><TITLE>PHP</TITLE></HEAD>
```

```
<BODY>
```

Il mio nome è:

```
<?php
```

```
$username="chiara";
```

```
echo $username; ?>
```

```
</BODY></HTML>
```

Delimitatore di istruzione

PHP – parte dinamica della pagina

# Delimitatori PHP

- ◆ Il codice PHP si intervalla al codice HTML.
- ◆ I delimitatori permettono al server web di distinguerlo dall'HTML
- ◆ I delimitatori di PHP sono i caratteri:  
`<?php` per denotare l'inizio (analogo a `<?`)  
`?>` per delimitare la fine
- ◆ Per visualizzare l'output di uno script PHP occorre visualizzare la pagina dal browser digitando la URL del server (ad es. <http://localhost/miapagina.php>), non aprendo il file direttamente dal browser

# Esempio

```
<HTML><HEAD><TITLE>Esempio PHP</TITLE></HEAD>
```

```
<BODY>
```

Questo testo è in HTML

```
<P>
```

```
<?php echo "questo testo è in PHP"; ?>
```

```
<P>
```

questo testo è ancora in HTML

```
<?php echo "questo testo è ancora in PHP"; ?>
```

```
</BODY></HTML>
```

# Scrivere sul browser

L'output di uno script viene scritto sulla finestra del browser

L'istruzione di stampa di PHP è `echo` (o `print`)

```
<? echo "Ciao"; ?>
```

oppure

```
<? echo("Ciao"); ?>
```

oppure

```
<? print "Ciao"; ?>
```

# HTML e PHP

- ◆ Un testo puo' essere scritto sul browser sia da PHP che direttmanete in HTML
- ◆ Ad esempio, questi due pezzi di codice sono equivalenti:

```
<B> <?php echo "ciao!"; ?> </B>
```

e

```
<?php echo "<B>ciao!</B>"; ?>
```



# Commenti

I commenti sono utili per spiegare il codice scritto (USARLI SPESSO!!!) ma anche per disabilitare temporaneamente una o più istruzioni.

Esistono tre tipi di commenti:

- ◆ /\* commento a riga multipla \*/
- ◆ // commento a riga singola
- ◆ # commento a riga singola

# Variabili

- ◆ Le variabili in PHP si denotano con una sequenza di caratteri preceduti dal simbolo \$
- ◆ **Lettere MAIUSCOLE e minuscole sono diverse**
- ◆ Devono iniziare con una lettera o il carattere sottolineatura (\_) possono contenere numeri

\$miavar

\$\_ENV

\$var45

# Assegnare e visualizzare le variabili

Per assegnare un valore ad una variabile si usa il carattere '='

```
$miavar=17;
```

```
$mess="Ciao!";
```

Le variabili vengono visualizzate dall'istruzione *echo*

```
echo $miavar;
```

# Tipi delle variabili

Un tipo è la descrizione del formato della variabile

## ◆ Boolean (TRUE, FALSE)

- `$pagato = FALSE;`

## ◆ Integer

- `$count=1;`

## ◆ Float, double

- `$miavar=1.456;`

## ◆ String

- `$messaggio="benvenuto";`

## ◆ Array

- `$lista=array("primo","secondo");`
- `$lista[1];`

## Lo *scope* di una variabile

- ◆ L'ambito o *scope* di una variabile in PHP è la pagina stessa: ogni variabile esiste solo per lo script dove è definita, alla fine del processamento della pagina scompare.
- ◆ Questo significa che possono coesistere variabili con lo stesso nome se definite in pagine diverse e che non è possibile usare il valore di una variabile in uno script diverso da dove è stata definita.
- ◆ Le uniche variabili globali permesse sono i *superglobalarray*, array globali predefiniti che sono visibili da qualsiasi pagina dell'applicazione
- ◆ Le vecchie versioni di PHP permettevano la definizione esplicita di variabili globali. Nelle versioni attuali (5) le globali sono disattivate di default nel file `php.ini` con la direttiva `register_globals = Off`.

# Variabili predefinite

- ◆ Negli script PHP sono disponibili variabili globali definite al di fuori dello script, chiamate variabili predefinite (o superglobalarray)
- ◆ **Variabili del server**, sono definite dal server web e quindi variano a seconda del server usato. Sono definite come l'array `$_SERVER`

`$_SERVER["PHP_SELF"]` nome dello script corrente,  
`$_SERVER["SERVER_NAME"]` indica il nome del server,  
`$_SERVER["HTTP_USER_AGENT"]` indica il browser che ha inoltrato la richiesta

- ◆ **phpinfo()** fornisce informazioni sullo stato corrente di PHP, tra cui tutte le variabili predefinite. È utile ad esempio, per vedere se MySQL è installato e viene visto correttamente da PHP

## Costrutto isset()

- ◆ Questa funzione permette di verificare se una variabile è impostata o meno:
- ◆ `isset($var)`; restituisce *true* se la variabile `$var` è settata altrimenti *false*

```
<? $var="Pippo";
```

```
$settata=isset($var);
```

```
?>
```

# Stringhe

Le stringhe sono sequenza di caratteri alfanumerici.

Possono essere definite con i caratteri ' oppure “

L'unione di stringhe si effettua con il carattere punto (.)

```
$nome = "Mario";
```

```
$cognome="Rossi";
```

```
$nomeintero=$nome.$cognome;
```



# Stringhe

- ◆ Possono essere specificate con virgolette singole ‘ oppure doppie “
- ◆ Si differenziano:
  - per i caratteri di *escape*: sequenze speciali di caratteri che hanno una specifica interpretazione, ad esempio \n per new line, \' per virgoletta singola)
  - per l'interpretazione delle variabili.

# Stringhe

- ◆ La virgoletta singola ' produce un output letterale

```
$var="variabile";
```

```
$myvar = 'La mia $var! \n';
```

```
print($myvar);
```

produce come output

```
La mia $var!\n
```

# Stringhe

- ◆ **La virgoletta doppia “ produce un **output processato**:**
  - I caratteri che seguono il backslash vengono tradotti
  - Le variabili vengono valutate

```
$var="variabile";
```

```
$myvar = "La mia $var! \n";
```

```
print($myvar);
```

produce come output

```
La mia variabile!
```

# Stringhe

- ◆ Le stringhe che contengono un numero nella parte iniziale possono essere convertite in numero

## Esempio:

```
$stringa="45 anni";
```

```
$num=23;
```

```
$add = $num + $stringa;
```

# Operatori su stringhe

Vediamo alcuni degli operatori di manipolazione di stringhe. Numerosi altri sono disponibili sulla documentazione del linguaggio

- ◆ **strlen(stringa)** restituisce il numero di caratteri della stringa
- ◆ **trim/ltrim/rtrim(stringa)**. Trim elimina spazi all'inizio e alla fine della stringa, ltrim a sinistra rtrim a destra
- ◆ **substr(stringa, intero1 [,intero2])**. Restituisce la sottostringa che inizia alla posizione intero1 eventualmente fino a intero1+intero2
- ◆ **str\_replace(str1,str2,str3)** restituisce una nuova stringa dove sostituisce tutte le occorrenze di str1 con str2 in str3.
- ◆ **strtolower/strtoupper(stringa)** converte tutti i caratteri in minuscolo/maiuscolo
- ◆ Il confronto tra stringhe si effettua con gli usuali operatori di confronto ==, <, >

# Esempio manipolazione di stringhe

```
<? $str="pippo pluto e paperino";
```

```
$n=strlen($str);
```

```
echo $n;
```

```
?>
```

Restituirà la lunghezza in caratteri della stringa

# Operatori numerici

- ◆ PHP supporta cinque operatore numerici

Addizione  $a + b$ ;

Sottrazione  $a - b$ ;

Moltiplicazione  $a * b$ ;

Divisione  $a / b$ ;

Modulo  $a \% b$ ;

- ◆ Incremento  $i++$  incrementa di 1
- ◆ Decremento  $i--$  decrementa di 1

# Data e ora

- ◆ Sono disponibili varie funzioni per reperire la data e ora correnti sul server. Il tempo viene rappresentato come un *timestamp* che rappresenta i secondi trascorso dall' "ora zero" Unix, 1 gennaio 1970!
- ◆ In PHP abbiamo due funzioni per reperire la data: **getdate()** che restituisce un array contenente data e ora corrente e **date("formato")** che restituisce la data nel formato definito.

```
$dataoggi=date("j/M/Y");
```

```
echo $dataoggi;
```

Visualizzerà

21/Apr/2010



# Istruzione date()

**date("formato")** dove *formato* può contenere

Y anno su 4 cifre

y anno su 2 cifre

n mese numerico

m mese numerico su due cifre

F mese testuale

M mese testuale su tre lettere

d giorno del mese su due cifre

j giorno del mese

w giorno della settimana

l giorno della settimana testuale

D giorno della settimana su tre lettere

H ora su due cifre

G ora

i minuti

s secondi

**ESEMPIO: date("j/M/Y")**

# Costrutto condizionale

```
<? if (condizione) {
```

*istruzioni da eseguire se la condizione è vera*

```
} else {
```

*istruzioni da eseguire se la condizione è falsa*

```
} ?>
```

**Il risultato di condizione deve essere un valore booleano, quindi una variabile, se essa è booleana, oppure un operatore di confronto tra variabili**

# Operatori di confronto

$a == b$  uguale

$a === b$  identico (uguale anche il tipo)

$a != b$  non uguale

$a !== b$  non identico

$a > b$  maggiore

$a < b$  minore

$a >= b$  maggiore uguale

$a <= b$  minore uguale

# Operatori logici

- ◆ `and` è vero se e solo se entrambi gli argomenti sono veri.
- ◆ `or` è vero solo se uno (o entrambi) degli argomenti è vero.
- ◆ `!` Negazione. E' vero solo se il suo argomento è falso e viceversa
- ◆ `xor` è vero solo se uno dei due argomenti (ma non entrambi) sono veri.
- ◆ `&&` come `and` ma con ottimizzazione di valutazione del primo argomento
- ◆ `||` come `or` con ottimizzazione di valutazione del primo argomento

# Istruzione switch

**switch (espressione)**

```
{ case costante_espressione: istruzione; break;
```

```
  case costante_espressione: istruzione; break;
```

```
  ....
```

```
  default: istruzione;
```

```
}
```

**Rappresenta una serie di if annidati**

## Switch - esempio

```
<?
```

```
switch ($miavar)
```

```
{ case 5: echo "Insufficiente"; break;
```

```
  case 10: echo "10 e lode!!"; break;
```

```
  default: echo "sufficiente";
```

```
}
```

```
?>
```

# Cicli

```
while (espressione) { istruzione }
```

L'istruzione viene ripetuta fino a quando l'espressione viene valutata a TRUE

```
<? $a=1;
```

```
while ($a<10) {
```

```
echo $a;
```

```
$a++;
```

```
} ?>
```

# Cicli

```
do { istruzione } while (espressione);
```

L'istruzione viene eseguita prima della valutazione dell'espressione, quindi almeno una volta.

```
<? $a=0;
```

```
do {
```

```
echo "ciclo do-while questo è a: $a";
```

```
} while ($a > 0) ; ?>
```



# Cicli

```
for (espressione1;espressione2;espressione3)
```

```
{ istruzione }
```

**Esempio:**

```
<? for ($i=0;$i<=10;$i++)
```

```
{ echo $i;
```

```
} ?>
```