

# Basi di Dati

## *PHP - parte 1*

**Prof. Mauro Conti**

Dipartimento di Matematica - Università degli studi di Padova

conti@math.unipd.it - <http://www.math.unipd.it/~conti>

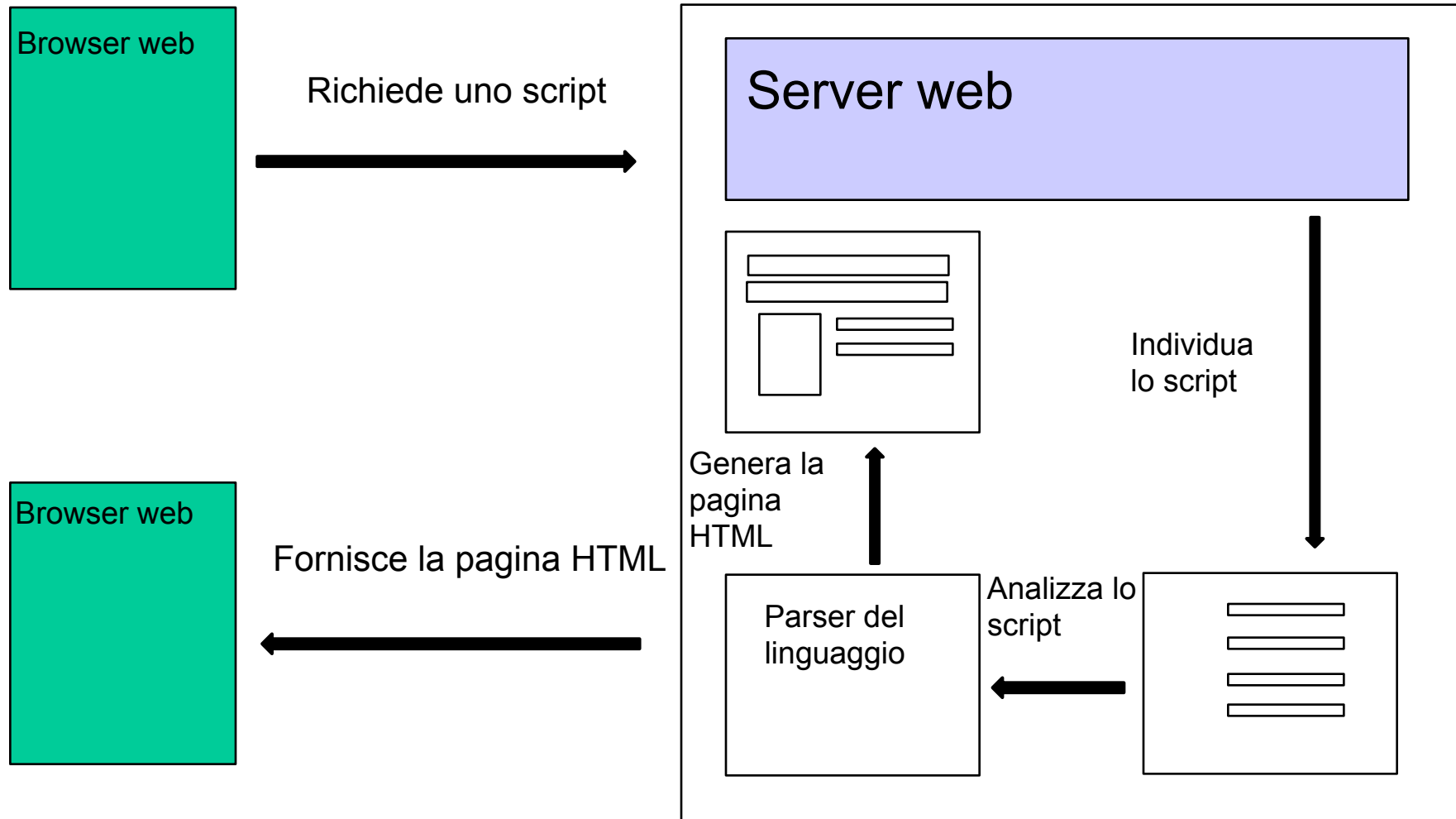


UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA

Slides credits to:  
**Paolo Baldan**

- **PHP Hypertext Preprocessor** (acronimo ricorsivo)
  - Linguaggio di scripting interpretato
  - Principalmente usato per applicazioni web lato server
- **Caratteristiche**
  - open source
  - numerose librerie (grafica, mail, ...)
  - componenti per l'interazione con vari DB (MySQL, Oracle, Postgres, ...)
  - ...

# “Esecuzione” di una pagina php



- I parametri di funzionamento di PHP sono definiti nel file **php.ini** che il server web legge ad ogni riavvio
- Tipicamente i valori di default sono ok
- Esempio:
  - **display\_errors**: impostato su 'on' mostra gli errori 'sul browser'.
  - **max\_execution\_time**: tempo concesso per l'esecuzione di uno script, dopo il quale si blocca (def. 30s). Utile se esistono loop su cicli errati.
  - **session\_save\_path**: Questo parametro indica la cartella nella quale PHP salva i files di sessione:
  - **short\_open\_tag**: default a 'on', ci permette di aprire e chiudere codice PHP con i tag `<? e ?>`.

Pagina con `estensione.php`

Parte statica  
(HTML)

Parte dinamica  
(PHP)

```
<html>
<head>
  <title>Esempio PHP</title>
</head><body> <p>Buongiorno,
mi chiamo
<?php    $username=
"<b>Nessuno</b>" ;    echo
$username;
?>
signor Polifemo</p>
</body>
</html>
```

Polifemo.php

Polifemo.html

- Il codice PHP si interviene al codice HTML, delimitato dai tag

`<?php`    ...    `?>`

- I delimitatori permettono al server web di riconoscere il codice, che viene eseguito
- L'output viene combinato con la parte statica che sta al di fuori dai tag PHP, e il tutto è inviato al browser (client)
- Per “eseguire” lo script, devo richiederlo al web-server tramite il browser

`http://localhost/miapagina.php`

- **Sintassi molto simile al C ...**

- L'istruzione di stampa di PHP è echo (o print)
  - `<?php echo "Ciao"; ?>`
- oppure
  - `<?php echo ("Ciao"); ?>`
- oppure
  - `<?php print "Ciao"; ?>`



- Tre sintassi per i commenti:
  - `/* commento a  
riga multipla */`
  - `// commento a riga singola`
  - `# commento a riga singola`

- Le variabili in PHP si denotano con una **sequenza di caratteri** preceduti dal simbolo **\$** (Maiuscole e minuscole sono diverse)

**\$nomeVariabile**

- Devono iniziare con una lettera o il carattere sottolineatura (\_) possono contenere numeri

- \$miavar, \$\_ENV, \$var45**

- Per **assegnare** un valore ad una variabile si usa '='
  - \$miavar=17;
  - \$mess="Ciao!";

- **Nota:** distinzione tra nome della variabile ed accesso alla variabile ...
- Permette di usare **variabili variabili**
  - `$a="b" ;`
  - `$b=1`
  - `echo $$a`

... produce 1
- Potente, ma pericoloso ...

- Lo **scope** di una variabile in PHP è la **pagina** stessa.
- Ogni variabile esiste solo per lo script (pagina) in cui è definita e alla fine della computazione scompare
  - Possono coesistere variabili con lo stesso nome in pagine diverse
  - Non è possibile usare il valore di una variabile in una pagina diversa da dove è stata definita
- Le uniche variabili globali permesse sono i **superglobalarray**, array globali predefiniti che sono visibili da qualsiasi pagina dell'applicazione.

- Variabili globali definite al di fuori dello script
- Variabili del server, sono definite dal server web: array `$_SERVER`
  - `$_SERVER["PHP_SELF"]`: nome dello script corrente
  - `$_SERVER["SERVER_NAME"]`: nome del server,
  - `$_SERVER["HTTP_USER_AGENT"]`: browser che ha inoltrato la richiesta
- `phpinfo()`
  - informazioni sullo stato corrente di PHP, tra cui tutte le variabili predefinite.  
(esempio: permette di vedere se Mysql è installato e visto correttamente da PHP).

- Una variabile è dichiarata **implicitamente** con un assegnamento
- Può essere riferita anche se non è stata assegnata
- Si può verificare se è stata impostata con la funzione `isset()`

**`isset($var)`**

restituisce TRUE se la variabile `$var` è stata impostata

- Una variabile può essere distrutta  
nello scope corrente con  
`unset($var)`

```
<?php
    # isset($var) FALSE;
    $var="Pippo";    # isset
($var) TRUE;
    unset($var);    # isset
($var) FALSE;
?>
```

- **Weakly-typed**: Il tipo è definito implicitamente dall'assegnamento
  - Boolean (TRUE, FALSE)
    - `$pagato = FALSE;`
  - Integer
    - `$count=1;`
  - Float, double
    - `$miavar=1.456;`
  - String
    - `$messaggio="benvenuto";`
  - Array:
    - `$lista=array("primo","secondo"); echo $lista[1];`

```
<?php
```

```
$pippo = "1";           // $pippo ha tipo string (ASCII 49)
```

```
$pippo = $pippo + 2;     // $pippo diventa integer (3)
```

```
$pippo = $pippo + 1.3;   // $pippo ora è float (4.3)
```

```
$pippo = 5 + "10 mele"; // $pippo di nuovo integer (15)
```

```
echo "20 mele" + "10 pere";
```

```
?>
```



- Si possono anche definire delle costanti con

```
define(NOME_COST, valore)
```

- Esempio

```
define(PI, 3.1456)
define(GREET, "Buon giorno")
print GREET
print ", ecco il valore di pi-greco: "
print PI;
```

- **Concatenazione** con il carattere punto (.)
  - `$nome = "Mario";`
  - `$cognome="Rossi";`
  - `$nomeintero = $nome . " " . $cognome;`
- Definite con **'stringa'** oppure **"stringa"**
- si differenziano per le regole di "espansione" (vedi prossima slide)
  - caratteri di escape: interpretazione di sequenze speciali di caratteri  
Es. `\n` new line, `\` virgolette
  - espansione delle variabili con il loro valore

- la virgoletta singola `'...'` produce un output **letterale**

```
$var="variabile";  
$myvar = 'La mia $var! \n';  
print($myvar); print "stop";
```

## Output

```
La mia $var! \nstop
```

- la virgoletta doppia `"..."` produce un output interpretato

```
$var="variabile";  
$myvar = "La mia $var! \n";  
print($myvar); print "stop";
```

## Output

```
La mia variabile!  
stop
```

- Numerosissimi, alcuni esempi ...
  - `strlen(stringa)`  
numero di caratteri della stringa
  - `trim(stringa)`  
elimina spazi all'inizio e/o alla fine della stringa
  - `substr(stringa, intero1 [,intero2])`  
sottostringa che inizia alla posizione intero1 (fino a intero1+intero2)
  - `str_replace(str1,str2,str3)`  
sostituisce tutte le occorrenze di str1 con str2 in str3.
  - `strtolower/strtoupper(stringa)`  
converte tutti i caratteri in minuscolo/maiuscolo

- Lo script

```
<?php
    $str="        pippo pluto e paperino        ";
    $str2=trim($str);
    echo $str2;
?>
```

- Restituirà la stringa  
"pippo pluto e paperino"

- **Simili a quelli del C**
  - 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
- ...

- Varie funzioni per reperire la data e ora correnti sul server.
- Il tempo viene rappresentato come un timestamp che rappresenta i secondi trascorsi dall'ora zero Unix, 1 gennaio 1970 (**Unix Epoch**)!
- Per reperire la data:
  - **getdate()**: restituisce un array contenente data e ora corrente
  - **date("formato")**: che restituisce la data nel formato definito.
- Esempio:
  - `echo date("l d/M/Y");`
  - Visualizza: Saturday 30/May/2012

- Esempio:

```
print_r(getdate())
```

```
Array(  
  [seconds] => 17  
  [minutes] => 28  
  [hours] => 12  
  [mday] => 30  
  [wday] => 3  
  [mon] => 5  
  [year] => 2012  
  [yday] => 150  
  [weekday] => Wednesday  
  [month] => May  
  [0] => 1338373697  
)
```



- `date("formato")` dove formato può contenere
  - Anno:
    - `Y` (4 cifre), `y` (2 cifre)
  - Mese:
    - `n` (numerico), `m` (numerico su due cifre)
    - `F` (testuale), `M` (testuale su tre lettere)
  - giorno del mese
    - `j` oppure `d` (su due cifre)
  - giorno della settimana
    - `w` (numerico), `l` (testuale), `D` (testuale su tre lettere)
  - ora, minuti, secondi
    - `h`, `i`, `s`

# Costrutti di Controllo

- Sono esattamente quelli del C
- `if (cond) { ... } else { ... }`
- `switch ... case ... break ... default`
- `while (...) do {...}`
- `do ... while (...)`
- `for (...) {...}`

```
• <? if (condizione) {  
    ...  
} else {  
    ...  
} ?>
```

*condizione* è un'espressione il cui risultato è un valore booleano

- variabile booleana
- operatore di confronto tra variabili
- o interpretato come tale (FALSE ~ stringa = "", numero = 0, array senza elementi, variabile NULL)

- $a == b$  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

- Se assegnamo

```
$a=12;  
$b="12 pere";
```

- Allora è vero

```
$a==$b;
```

- mentre è falso

```
$a=== $b;
```

- *exp1* **and** *exp2*,    *exp1* **or** *exp2*  
vero se entrambi/almeno uno tra *exp1* e *exp2* è vero
- **!** *exp*  
vero solo se il suo argomento è falso e viceversa
- *exp1* **xor** *exp2*  
vero solo se uno dei due argomenti (ma non entrambi) sono veri.
- *exp1* **&&** *exp2*,    *exp1* **||** *exp2*  
come **and**/**or** ma con ottimizzazione di valutazione del primo argomento

- **switch** (espressione) {  
    **case** costante\_espressione:  
        istruzione;  
        **break**;  
    **case** costante\_espressione:  
        istruzione;  
        break;  
    ...  
    default:  
        istruzione;  
}

```
<?php  
switch ($voto){  
    case "D":  
        echo "Insufficiente";  
        break;  
    case "A":  
        echo "Ottimo";  
        break;  
    default:  
        echo "Ok";  
}  
  
?>
```



- **while** (espressione) {  
    istruzione  
}
- L'istruzione viene ripetuta fino a quando l'espressione viene valutata a TRUE

```
<?php  
  
$a=1;  
  
    while ($a<10) {  
        echo $a;  
        $a++;  
    }  
  
?>
```

- **do** {  
    istruzione  
} **while** (espressione)
- L'istruzione viene eseguita prima della valutazione dell'espressione, quindi almeno una volta.

```
<?php  
  
    $a=0;  
do {  
    echo $a;  
    $a++;  
}  
while ($a<10);  
  
?>
```

- **for** (espressione1;espressione2;espressione3) {  
    istruzione  
}

```
<?php  
  
    for ($i=0;$i<=10;$i++)  
    {  
        echo $i;  
    }  
?>
```

- Realizzare una pagina PHP che scriva:
- “Buongiorno/Buonasera/Buonanotte Pippo, benvenuto nella mia prima pagina PHP”
  - la scelta tra Buongiorno/Buonasera/Buonanotte è legata all’ora attuale (8 - 12 Buongiorno, 12-20 Buonasera, 20-8 Buonanotte)
  - Il nome “Pippo” deve essere contenuto in una variabile
- La pagina dev’essere chiusa da “stai usando il browser “ completata con il tipo di browser impiegato dall’utente.

# Array

- PHP offre **array associativi**, che associano un **valore** ad una **chiave**

Chiavi	0	1	2	3
Valori	pippo	topolino	paperino	paperone

- Le chiavi possono essere anche non numeriche

Chiavi	clarabella	minnie	paperina	brigitta
Valori	pippo	topolino	paperino	paperone

- Costrutto **array()**

- `$pers = array("pippo", "topolino", "paperino", "paperone");`
- `$pers = array(1 => "topolino", 0 => "pippo",  
2 => "paperino", 3 => "paperone");`

<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>
pippo	topolino	paperino	paperone

- `$coppie = array('clarabella'=>"pippo", 'minnie'=>"topolino",  
'paperina'=>"paperino", 'brigitta'=>"paperone");`

<b>clarabella</b>	<b>minnie</b>	<b>paperina</b>	<b>brigitta</b>
orazio	topolino	paperino	paperone

- Un array può essere creato assegnando un suo elemento
  - `$pers[0] = "pippo";`  
`$pers[1] = "topolino"`  
`...`
- senza specificare l'indice si aggiungere un elemento in coda all'array (l'array è creato se non esiste)
  - `$pers[] = "pippo";`  
`$pers[] = "topolino";`  
`....`



- L'accesso agli elementi può avvenire tramite l'indice

```
$pers[1];                                # elemento "topolino"
```

- o tramite la chiave

```
$coppie['clarabella'];  # elemento "pippo"
```

- **Costrutto foreach:** permette di accedere a tutti gli elementi di un array

```
foreach($myarray as [$key => ] $val) {
```

```
....
```

```
}
```

scorre l'array \$myarray associando alla variabile \$var i valori corrispondenti.

- Mostra i personaggi contenuti in un array in una tabella HTML

```
...  
<body>  
<table border="1"> <tr><th>Personaggi</th><tr>  
<?php    $pers = array("pippo", "topolino",  
                        "paperino", "paperone");  
  
    foreach ($pers as $nome) {  
        print  
        "<tr><td>$nome</td></tr>\n";  
    };  
>  
</table>
```

Personaggi
pippo
topolino
paperino
paperone

```
...  
<table border="1"> <tr><th>Personaggio</th> <th>Partner</th><tr>  
<?php  
    $coppie = array("clarabella" =>"pippo", "minnie" => "topolino",  
        "paperina" => "paperino", "brigitta" => "paperone");  
  
    foreach ($coppie as $pers => $partner) {  
        print " <tr><td>$pers</td> <td>$partner</td> </tr>\n";  
  
    }  
>  
</table>  
...
```

Personaggio	Partner
clarabella	pippo
minnie	topolino
paperina	paperino
brigitta	paperone

- **in\_array(val, \$myarray)**

verifica se **val** è presente come valore nell'array **\$myarray**

```
$nome="pluto";  
if (in_array($nome, $pers)) {  
    echo "Il personaggio $nome" è presente nella lista"; }  
}
```

- **array\_merge(array1, ..., arrayn)**

unisce tutti gli array elencati in un unico array

```
$pers1 = array("pluto", "paperone");  
$pers2 = array("minnie", "paperino");  
$pers  = array_merge($pers1, $pers2);
```

- Contare gli elementi di un array

`count(array)`

```
if (count($pers) == 0) { echo $_SERVER["PHP_SELF"] . " :  
Errore, array \ $pers vuoto!"; }
```

- Ordinare un array

- `sort($myarray);`    # crescente
- `rsort($myarray);`    # decrescente

- Non restituiscono un valore ma ordinano l'array passato come parametro

- `explode()` e `implode()`  
convertono una stringa in un array e un array in una stringa per mezzo di un carattere separatore
- ```
$personaggi = "pippo topolino paperino paperone"  
$pers_array = explode(" ", $personaggi);
```
- ```
$personaggi_virgola = implode(",", $pers_array);
```

- Un array può contenere valori di tipo diverso:
  - `myarray = array ("nome" => "Gismondo",  
                  "eta"   => 32);`

# Funzioni



- Le funzioni possono avere argomenti in input (arg1,arg\_2, ... arg\_n) e possono restituire un valore
- Le variabili definite dentro una funzione hanno come scope la funzione stessa. Quindi al di fuori della funzione non esistono

```
function nome_funzione (arg_1,arg_2,...,arg_n) {  
    ...  
    return valore;  
}
```

- Chiamata `nome_funzione (args ...)`

```
<?php
function add($x, $y) {
    $somma = $x + $y;
    return $somma;
}

echo add(2,3);
?>
```

- esempio: per stampare una parte comune di una pagina web (es. intestazione)

```
...

<?php
function intestazione () {

print "<h1>Il mio sito</h1>\n";
print "<hr />\n";
print date("l F d, Y");
print "<hr />\n";
}

intestazione();

?>

...
```

- Stampa tutte le chiavi di un array, separate da spazi

```
function stampa_chiavi ($a) {  
    foreach ($a as $key => $dummy)  
        echo $key . ' ';;  
}
```

```
function stampa_chiavi($a) {  
    echo implode(' ', array_keys($a));  
};
```

- Le funzioni possono ritornare più valori, sotto forma di array

```
function divisori ($n) {  
    $d=2;    // si cercano i divisori di $n a partire da 2  
            // fino a $n  
    while ($d<=$n) {  
        while (! ($n % $d)) {    // quando se ne trova uno  
            $div_array[$d] = 1;    // lo si registra come chiave  
            $n= $n/$d;            // e si riduce $n  
        };  
        $d++;  
    };  
    return $div_array;  
};
```

```
echo "I divisori di 30 sono:" . stampa_chiavi(divisori(30));
```

- Per condividere e riutilizzare porzioni di script e funzioni si usa frammentare il codice in più file in modo da poterlo includere quando serve
  - `include (nomefile);`
    - L'istruzione include va posizionata all'inizio dello script php o comunque prima dei riferimenti a variabili o funzioni definiti nel file da includere
    - L'interprete PHP segnala errore se il file non viene trovato, proseguendo comunque l'esecuzione
  - `require (nomefile);`
    - Analogo a include() ma qualora il file non venga trovato genera un errore irreversibile.

- Funzione che visualizza un array come una tabella, usando gli indici come prima colonna e i valori come seconda
- Uguale ma prima ordina l'array

- Scrivere una funzione `login($log, $pwd)` che riceva in input un login e una password ed esegua un controllo di login password rispetto a due valori delle variabili `$login` e `$password` definite all'interno della funzione in modo statico (es. siano `$login="admin"` e `$password="boh"`).
  - Generare la scritta "login e password ok"/ "login sbagliato"/"password sbagliata" a seconda del risultato del check.
  - Richiamare la funzione dallo script php con valori diversi in modo da generare tutti e tre i casi
  - Modificare la funzione affinché riceva login e password come un array (ad es. `infologin[]`)