Basi di Dati-XI

Corso di Laurea in Informatica

Anno Accademico 2013/2014

Paolo Baldan

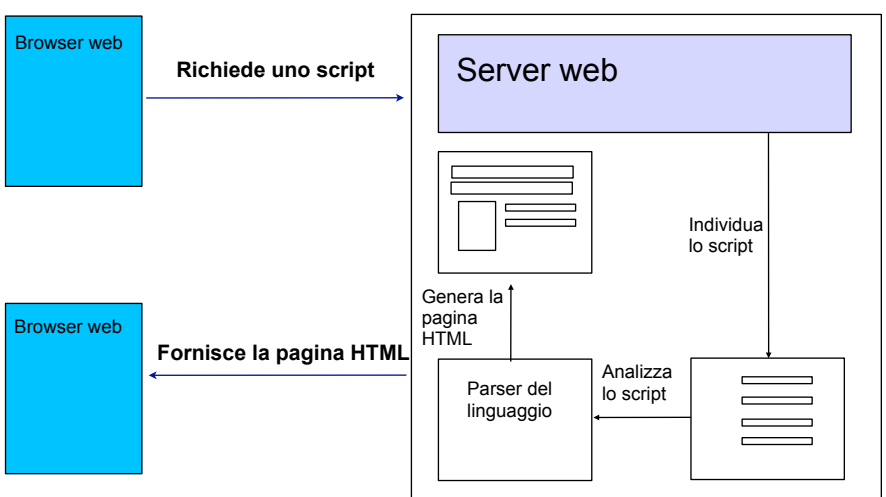
baldan@math.unipd.it

http://www.math.unipd.it/~baldan

Linguaggio PHP

- **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



Configurazione: php.ini

- 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 **`phpinfo()`**
- 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 `?>`.

Anatomia di uno script PHP

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

Delimitatori php

- Il codice PHP si intervalla 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* ...

Comando di stampa

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

Commenti

- Tre sintassi per i 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 **$** (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!";`

Variabili

- **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 ...

Scope

- 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 predefinite o superglobalarray

- 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).

phpinfo() **Superglobal.php**

Variabili e dichiarazioni

- 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;
?>
```


Tipi

- **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];`

Tipi: uso perverso


```
<?php

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


Tipi: uso perverso

15

```
<?php

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

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

Tipi: uso perverso


```
<?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)
```


Tipi: uso perverso

15

<?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)

Introduzione PHP Corso di Basi di Dati
Thursday, June 5, 2014

Tipi: uso perverso

15

<?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)

Introduzione PHP Corso di Basi di Dati
Thursday, June 5, 2014

Tipi: uso perverso

15

<?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)

Introduzione PHP Corso di Basi di Dati
Thursday, June 5, 2014

Tipi: uso perverso


```
<?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";
```


Tipi: uso perverso

15

<?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";

Introduzione PHP Corso di Basi di Dati
Thursday, June 5, 2014

Tipi: uso perverso

15

<?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";

?>

Introduzione PHP Corso di Basi di Dati
Thursday, June 5, 2014

Costanti

- 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;
```

Stringhe

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

Stringhe e quoting

- 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
```

Quoting.php

Operatori su stringhe ...

- 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

Stringhe: esempio

- Lo script

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

- Restituirà la stringa

  "pippo pluto e paperino"

Operatori numerici ...

- **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
- ...

Data e ora

- 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 contente 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

Data e ora

- 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
)
```


Costrutti di Controllo

Costrutti di controllo

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

Costrutto Condizionale


```
<? 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)

Alcuni 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

Esempio

- Se assegnamo

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

- Allora è vero

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

- mentre è falso

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

Esercizio

- Realizzare una pagina PHP che scriva:
- "Buongiorno/Buonasera/Buonanotte Paolo, benvenuta 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 "Paolo" 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

Array

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

Chiavi →

Valori →

0	1	2	3
pippo	topolino	paperino	paperone

- Le chiavi possono essere anche non numeriche

Chiavi →

Valori →

clarabella	minnie	paperina	brigitta
pippo	topolino	paperino	paperone

Creazione di un array

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

0	1	2	3
pippo	topolino	paperino	paperone

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

clarabella	minnie	paperina	brigitta
orazio	topolino	paperino	paperone

Creazione di un array

- 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”;`
    `....`

Accedere a un elemento

- 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.

Esempio

- 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

Esempio


```
...
<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

Manipolare un array

- **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);
```


Manipolare un array

- 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);** # descrescente

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

Manipolare un array

- **explode()** e **implode()**

  convertono un 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);`

Dove sono le mie struct?

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

Funzioni

Funzioni

- La 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 ...)`

Valori di ritorno


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

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


Senza valori di ritorno

- 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();
?>

...
```

Senza valori di ritorno: stampa delle chiavi

- 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));
};
```


Più valori di ritorno

- 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));
```


Inclusione di file

- 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.

Esercizio

- 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

Esercizio

- 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 risultto 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[]`)