# Metodologie informatiche per le discipline umanistiche

## Programmazione WEB PHP

Coordinatore: Prof. Paolo Nesi
Docenti: Prof. Paolo Nesi
Dr.sa Michela Paolucci
Dr. Emanuele Bellini

1

## Php: come nasce

- Il World Wide Web è stato creato da Tim Berners-Lee nel 1991
- A metà degli anni Novanta il Web era ancora formato in gran parte da **pagine statiche**
- Con l'evoluzione di Internet, si comincia a sentire l'esigenza di rendere dinamici i contenuti, cioè di far sì che la stessa pagina sia in grado di proporre contenuti diversi, personalizzati in base alle preferenze degli utenti, oppure estratti da una base di dati (database) in continua evoluzione
- Per questo scopo nasce il PHP

2

# PHP

- PHP nasce nel 1994, ad opera di Rasmus Lerdorf, con lo scopo di facilitare ai programmatori l'amministrazione delle homepage personali
- In seguito venne ampliato fino alla realizzazione di una versione (Form Interpreter, PHP/FI) che prevedeva la possibilità di integrare il codice PHP nel codice HTML in modo da semplificare la realizzazione di pagine dinamiche





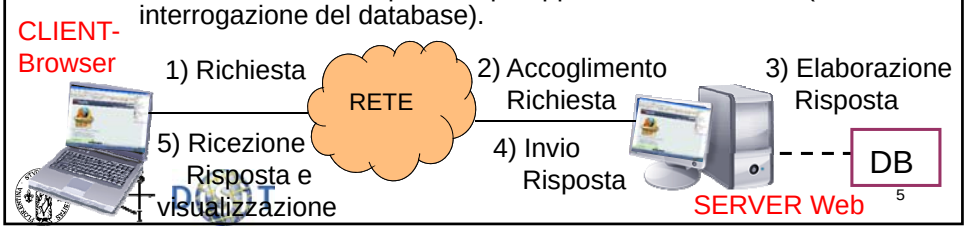
# PHP

- PHP (Hypertext PreProcessor) è un linguaggio di scripting Open Source molto utilizzato, è specialmente indicato per lo sviluppo Web e può essere integrato nell'HTML
- L'obiettivo principale del PHP è quello di permettere agli sviluppatori web di scrivere velocemente pagine web dinamiche, ma con PHP si possono fare molte altre cose
- E` integrabile in numerosi server Web (Apache HTTP Server, Internet Information Services - Microsoft -, .. )



# Ripasso: Architettura Software del Web

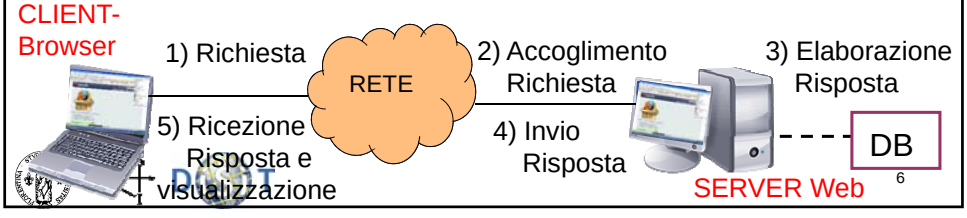
- L'architettura software del web può essere divisa in due categorie logiche:
  - Lato client: qualunque dispositivo che si connette alla rete e che comunica attraverso di essa con tutti gli altri dispositivi connessi. Nel client è sufficiente la presenza di un software per la visualizzazione delle informazioni che il server fornisce (Interne Explorer, Netscape, Mozilla, Opera, etc.)
  - Lato Server: è un dispositivo che attende le richieste del client ed invia le relative risposte dopo opportune elaborazioni (es: interrogazione del database).





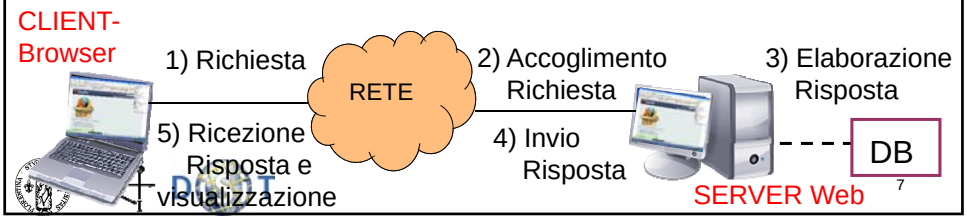

# Lato client: Browser

- Il Browser è un programma usato per navigare nel Web è serve per:
  - scaricare i file che si trovano sui server
  - visualizzare le pagine html
- Oltre ad Internet Explorer, il Browser più diffuso, ne esistono altri: Netscape Navigator, Mozilla, Opera



# Lato server: Server Web

- Il Server Web è un software che resta in ascolto delle richieste di accesso ad un sito web (inoltrate da parte di un Browser, ovvero dal client), le processa e restituisce dei dati come risposta
- I dati forniti dal Server contengono gli elementi necessari per la visualizzazione di una pagina web e vengono così analizzati ed interpretati dal Browser che li presenta all'utente nel miglior modo possibile






# PHP: funzionamento generale

- PHP può essere utilizzato inserendo codice PHP in mezzo al codice HTML che compone una pagina web
- Quando un utente (client) di un sito web (gestito dal server) arriva ad una pagina che contiene codice PHP, il server stesso lo esegue. In pratica il PHP viene eseguito localmente sul server che gestisce il sito web
- Non è possibile testare il codice PHP su un computer che non sia anche un server
- PHP è un linguaggio di scripting, ovvero non viene compilato (eseguito) prima di funzionare ma in tempo reale (mentre l'utente fa le richieste al server)
- Concludendo: I file.php possono contenere parti scritte in HTML e parti scritte in PHP delimitate da tag particolari. L'interprete php si occupa di gestire tutto il codice compreso all'interno di tali tag

# Tag di inizio e fine del codice php

- Esistono 4 set di tag che possono essere usati per delimitare blocchi di codice php. I primi due sono quelli standard:
  - **<?php echo 'Testo'; ?>**
  - <script language='php'> echo (Testo); </script>
  - <? echo 'Testo'; >
  - <% echo (Testo); %>





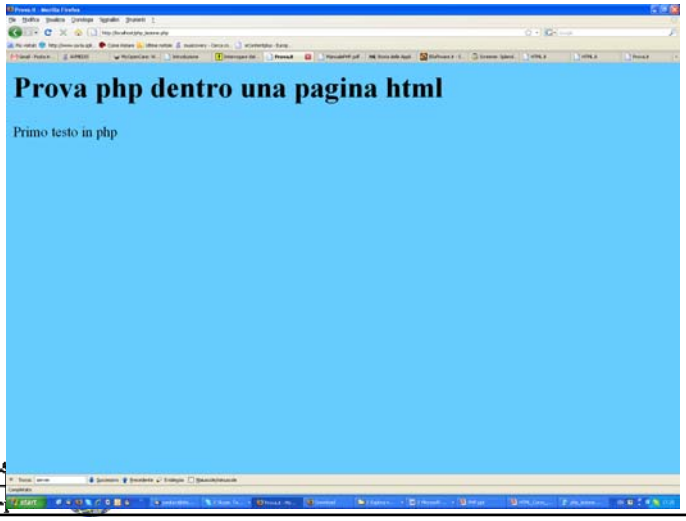
# Esempio: codice php in una pagina html

NOTA: nomefile.php contiene il seguente codice:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>Prova.it</title>
 </head>
    <body bgcolor="#66CCFF">
    <h1>
    Prova php dentro una pagina html
    </h1>
        <?php
        echo 'Primo testo in php';
        ?>
  </div>
 </body>
</html>
```



# Visualizzazione Esempio: codice php in una pagina html






# I Commenti

- Esistono più metodi:
  - Commento su una riga:
    # commento 1
    // commento 2
  - Commento su più righe:
    /* Questo è il
    commento 3
    */

# Variabili

- Le variabili PHP possono contenere array, stringhe oppure numeri
- L'assegnazione viene fatta tramite l'operatore '='
- È possibile assegnare dei valori alle variabili:
  - Per valore: il valore della variabile di origine viene copiato in quella di destinazione

    **$nomeVariabile = valore;**
  - Per riferimento: invece di creare una copia della variabile, si effettua un puntamento alla locazione di memoria della variabile. Ne consegue che modifiche sul valore di una variabile si ripercuotono sull'altra.

    **$nomeVariabile = &nomeAltraVariabile;**
- **<html>**

```
<body>
        <?php
        $myVar = "La mia prima variabile";
        echo $myVar;
        ?>
</body>
</html>
```




# Nomi delle variabili: sintassi

- I nomi delle variabili sono 'case sensitive'
- I nomi delle varibili si possono scegliere usando lettere, numeri e 'underscore' (_)
- Il primo carattere usato NON può essere un numero (si usano lettere o underscore)
- I nomi delle variabili sono precedute dal simbolo '$'


# Tipi di Dati

- Php supporta I seguenti tipi di dati:
  - boolean
  - integer
  - float
  - string
  - array
  - object
  - resource
  - NULL
  - mixed
  - number
  - callback
  - ...





# Tipi di Dati: boolean

- Nota: tutte le variabili in PHP devono iniziare con il simbolo $
- Il **boolean** è il tipo di dato più semplice, può assumere due valori: True o False

```
<?php
$variabileVero =  True;
$variabileFalso = False;
?>
```



# Tipi di Dati: integer

- Un **Integer** è un numero intero positivo o negativo di lunghezza dipendente dal sistema operativo, in genere 32 bit con segno, può essere specificato in base 10, 16, 8 eventualmente preceduto dal segno. I numeri che iniziano con una cifra diversa da 0 vengono considerati decimali, se iniziano con 0 ottali, se iniziano con 0x esadecimali.
  - <?php
    $a = 1234; # numero decimale
    $a = -123;  # numero negativo
    $a = 0123; # numero ottale (equivalente a 83 decimale)
    $a = 0x1A; # numero esadecimale (equivalente a 26 decimale)
    ?>


17

# Tipi di Dati: float

- **float**, anche in questo caso la dimensione del dato dipende dal sistema operativo, in genere 64 bit. Assume valori decimali a virgola mobile:
  - <?php
    $myDouble = 10.341;
    $myDouble = 0.14e2;
    ?>


18

# Tipi di Dati: string

- **string** contiene testo. Si hanno due modalità:
  - Tra apici (‘testo’), in questo caso se si vuole inserire un apice nella stringa, è necessario farlo precedere da backslash (\’) , il carattere backslash può essere inserito raddoppiandolo(\\)

    **<?php  echo ’testo informato \‘txt\’ ’; ?>**

    (si visualizza: testo in formato ‘txt’)
  - Tra virgolette (“testo”), in questo caso si possono usare I caratteri speciali del linguaggio di programmazione C (\n,\r,\\,`\t, … )





# Tipi di Dati: array

- Un array, contiene una serie di valori accessibili tramite un indice
- Sintassi:
  array( [key => ] value,...);
  dove key può essere un intero o una stringa e value un qualsiasi valore, compreso un altro array.

```
<?php
$arr = array("foo" => "bar", 12 => true); //con chiave
    echo $arr["foo"]; // bar
    echo $arr[12]; // 1
?>
$arr = array ('uno', 'due', 'tre', 'quattro', 'stop', 'cinque'); //senza chiave
```

- Se non specificate alcuna chiave, viene preso il massimo indice intero aumentato di 1, se specificate una chiave che esiste già il valore è sovrascritto.

```
<?php
    // Questi due array sono gli stessi ma hanno chiavi diverse
    array(5 => 43, 32, 56, "b" => 12);
    array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

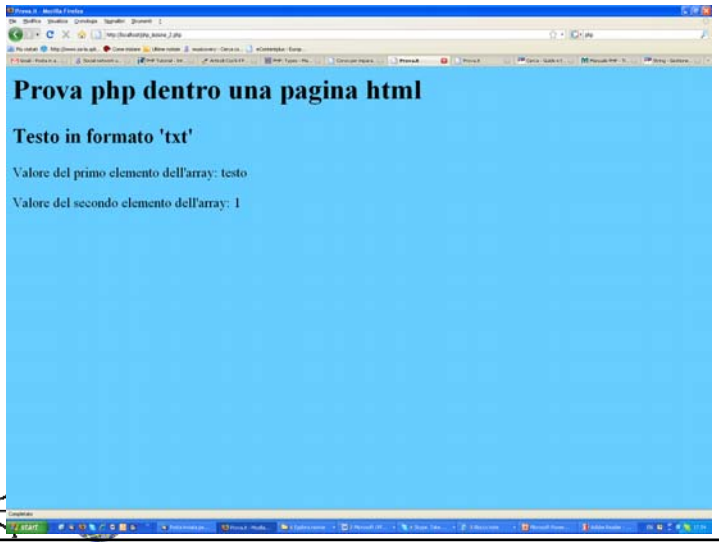


# Esempio: array

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
      <title>Prova.it</title>
 </head>
    <body bgcolor="#66CCFF">
      <h1>
      Prova php dentro una pagina html
      </h1>
      <?php
        # come trattare l'apice
        echo '<p>testo in formato \'txt\'<p>';
        // dichiarazione del'array 'arr'
        $arr = array("id1" => "testo", 12 => true);
        /*
           stampa sulla pagina dei valori dell'array
        */
        echo 'Valore del primo elemento dell\'array: ' ;
        echo $arr["id1"]; // " testo "
        echo '<p>Valore del secondo elemento dell\'array: ';
        echo $arr[12];    // 1
        echo '</p>';
      ?>
    </body>
</html>
```



# Visualizzazione array



# Tipi di Dati: object

- **object,** Il PHP consente di creare oggetti e di usarli in modo simile a C++ e Java, per esempio:

```
<?php
class oggetto {//definizione della classe
   function eseguiOggetto(){
     echo "Ho eseguito oggetto";
     }//fine funzione
}//fine classe
$istanzaoggetto = new oggetto; //creo il nuovo oggetto
$istanzaoggetto->eseguiOggetto(); // e eseguo il suo
   metodo
?>
```



# Visualizzazione: object

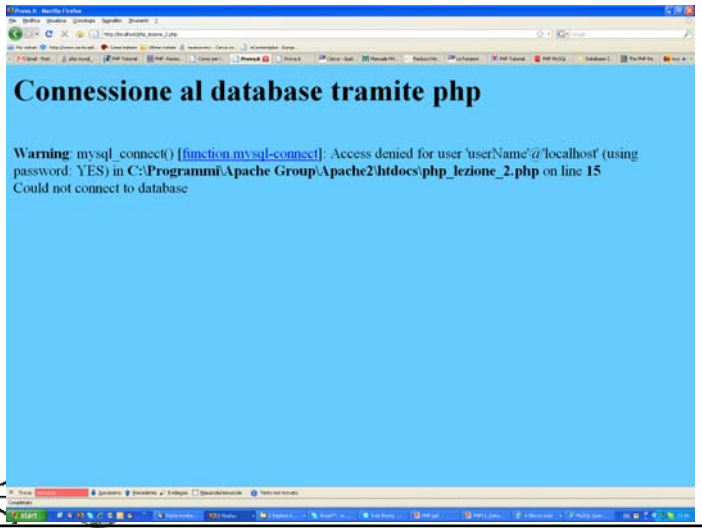

# Tipi di Dati: resource

- **resource**, si usa quando si fa riferimanto ad una risorsa esterna (un file, una connessione ad un database, etc.)
- Esempio: resource **mysql_connect**

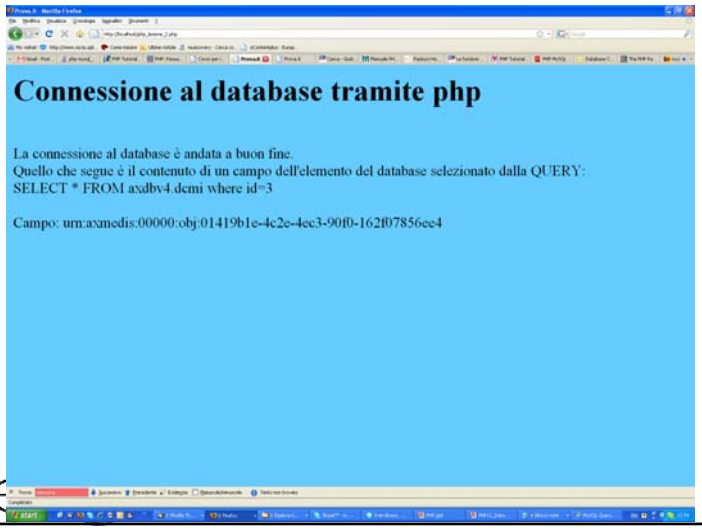
```
<body bgcolor="#66CCFF">
    <h1>
    Connessione al database tramite php
    </h1>
    <?php
            $con = mysql_connect("localhost",“UserName",“Password");
             if (!$con) {  echo 'Could not connect to database'; }
             else {
             echo '<br>','La connessione al database è andata a buon fine.','<br>',
             'Quello che segue è il contenuto di un campo dell\'elemento del database
              selezionato dalla QUERY:', '<br>', 'SELECT * FROM axdbv4.dcmi where id=3';
              $query = "SELECT * FROM dcmi where id=3;";
            $db = axdbv4;
      $mysql_result = mysql_db_query ($db, $query);
      // Display the data returned by the query
      while ($temp = mysql_fetch_row ($mysql_result)) {
            echo '<br>','<br>','Campo: ', $temp[1], '<br>';
      mysql_close($con);
      }
     }
            ?>
</body>
```



# Visualizzazione: resource errore nella connessione al DB



# Visualizzazione: resource connessione andata a buon fine




27

# Tipi di Dati: NULL

- Il valore NULL indica che ad una variabile non é stato assegnato nessun valore
- Una variabile é nulla se:
  - Gli è stata assegnata la costante NULL ($var = NULL;)
  - Non è stata mai ‘settata’ (non gli è stato assegnato nessun valore)
  - E`stata usata la funzione unset() (unset(var);)

28

# Conversioni tra tipi (esempi)

- In php è possibile trasformare una variabile da un tipo ad un altro:
  - Se abbiamo:
    - $a = 5; //intero
    - $b = '5'; //stringa
    - $c = 'f'; //stringa
    - $d = true; //boolean
  - Allora:
    - $e = $a.$b; //è uguale alla stringa '55'
    - $f = $a + $b; // è uguale al numero intero 10
    - $g = $a + $c; // è uguale al numero intero 5
    - $h = $a.$d; // è uguale alla stringa '51'
    - $i = $a+ $d; // è uguale al numero intero 6





# Conversioni tra tipi

- Si deduce che le conversioni:
  - Da intero a stringa: si ottiene il carattere corrispondente al numero
  - Da stringa a intero: si ottiene, se esiste, un numero corrispondente alla stringa altrimenti 0
  - Da intero a boolean: si ottiene false se è 0 (o minore di 0) e true se è maggiore di 0
  - Da boolean a intero: si ottiene 0 se è false e 1 se è true
  - Da stringa a boolean: si ottiene false se la stringa è vuota e true negli altri casi
  - Da boolean a stringa: si ottiene una stringa vuota se è false, il carattere '1' se è true



# Costanti

- Il PHP mette a disposizione delle costanti, ossia dei valori che si possono usare, ma non modificare (una costante rimarrà invariata per sempre)
- Le costanti non contengono il simbolo ‘$’ e sono "case-sensitive" per default





# Operatori

- Assegnazione
- Operatori aritmetici
- Incrementi
- Di stringhe
- Operatori di confronto
- Operatori Logici
- …



# Operatori di Assegnazione

- Sono usati per assegnare il valore di una variabile

| Operatore | Descrizione/Esempio |
|---|---|
| = | x=y |
| += | x+=y equivale a x=x+y |
| -= | x-=y equivale a x=x-y |
| *= | x*=y equivale a x=x*y |
| /= | x/=y equivale a x=x/y |
| %= | x%=y equivale a x=x%y |



# Operatori Aritmetici

| Operatore | Descrizione | Esempio |
|---|---|---|
| + | Somma | $a + $b |
| - | Sottrazione | $a - $b |
| * | Moltiplicazione | $a * $b |
| / | Divisione | $a / $b |
| % | Resto della divisione intera | $a % $b |
| ++ | Incremento | $a++ |
| -- | decremento | $a-- |

# Operatori di stringa

- Ci sono due operatori di stringa:
  - Il primo è l'operatore di concatenazione ('.'), che restituisce la concatenazione dei suoi argomenti a destra e a sinistra:
    - $a = "Ciao ";
      $b = $a . "Mondo!"; // ora $b contiene "Ciao Mondo!"
  - Il secondo è l'operatore di assegnazione concatenata ('.='), che aggiunge alla fine dell'argomento sul lato destro l'argomento sul lato sinistro:
    - $a = "Ciao ";
      $a .= "Mondo!"; // ora $a contiene "Ciao Mondo!"





# Operatori di confronto

| Operatore | Descrizione | Esempio |
|---|---|---|
| == | Uguale | $a == $b |
| != | Diverso | $a != $b |
| > | Maggiore di | $a < $b |
| < | Minore di | $a > $b |
| >= | Maggiore o uguale | $a >= $b |
| <= | Minore o uguale | $a <= $b |
| ?: | Operatore ternario | $ris = (esp1) ? (esp2) : (esp3);<br>$ris vale esp2 se esp1 è true, altrimenti vale esp3 |



# Operatori Logici

| Operatore | Descrizione | Esempio |
|---|---|---|
| && | and | $a && $b |
| \|\| | or | $a \|\| $b |
| ! | not | ! $a |




# Istruzioni

- Uno script PHP è costituito da una serie di istruzioni
- Una istruzione può essere un'assegnazione, una chiamata di funzione, un ciclo, …
- Le istruzioni terminano con un punto e virgola
- Le istruzioni si possono raggruppare in blocchi di istruzioni racchiudendole tra parentesi graffe
- Un gruppo di istruzioni è, a sua volta, un'istruzione


# Espressioni

- In php una espressione è una qualsiasi combinazione di funzioni (che vedremo), valori e operatori, che si risolvono in un valore
- Esempi:
  - 15 * 3; //espressione il cui valore è 45
  - Giacomo' . ' Verdi'; //espressione il cui valore è 'Giacomo Verdi'
  - $a + $b; /*espressione il cui valore è dato dalla somma dei valori delle variabili $a e $b*/





# Funzioni

- Una funzione è un blocco di codice che può richiedere uno o più parametri in ingresso e può fornire un valore di uscita
- PHP mette a disposizione numerose funzioni predefinite, di cui non si ha sottomano il codice ma risultano molto utili o talvolta indispensabili nella programmazione delle nostre applicazioni server
- In PHP la maggior parte delle funzioni restituisce un valore anche quando ciò potrebbe non essere ovvio: spesso, ad esempio, le funzioni restituiscono un valore boolean che indica l'esito della sua esecuzione



# Creare Funzioni

- function nome_funzione {
  lista di azioni;
  }
- Esempio:

```
function double($i) {
return $i*2;
}
```





# Usare le Funzioni

- Per utilizzare una funzione non bisogna fare altro che richiamarla (o invocarla)
- Se esiste ad esempio una funzione abs sarà sufficiente usarla in questa maniera:
  - $a = abs($b);
  - $c = 4 * abs(-65);
- Allostesso modo si usa una funzione creata da noi:
  - $f = double($d);



# Esercizi

- Dato un numero in ingresso (Es: $n=144), stampare il suo quadrato
- Dati due numeri in ingresso (Es: $n1=1012: $n2=250), stampare il quadrato e la media
- Dato un numero in ingresso, stampare se è pari o dispari



# Cicli: if else

- Sintassi:

```
if (condizione){
        azione1 da effettuare;
        azione2;}
else{ //se la condizione non e'verificata
        altre azioni;}
```

- Note:
  - Le parentesi graffe servono per raggruppare una serie di azioni
  - La clausola Else { } è facoltativa, va usata nel caso ci sia un'alternativa se il ciclo if non soddisfa la condizione indicata fra le parentesi tonde
- Esempio:

```
if ($a==$b){
        echo "sono uguali"; }
else{ //se la condizione non è verificata
        echo "sono diversi"; }
```

# Cicli:Elseif

- Sintassi:
  if ($a==$b){
  print "uguali";
  }
  elseif ($a==$c){
  print "uguale a c";
  }
  elseif ($a==$d){
  print "uguale a d";
  }
  ...
  else{
  print "diversi";}
- NOTE: Si tratta di un altro ciclo IF all'interno di un ciclo if. Il server controlla se il primo ciclo (IF) è vero, se è falso va sul ciclo elseif, se è falso pure questo continua con i cicli ELSEIF fino a quando non trova un'alternativa vera oppure l'istruzione finale ELSE (non obbligatoria)





# Cicli: for

- Sintassi:
  for (espressione iniziale; condizione; aggiornamento){
  lista azioni;
  }
- Esempi:
- for ($i = 1; $i <= 10; $i++) {
  echo $i;
  }
- for ($a=0; $a<=3; $a++){
  echo "ciao $a <br>";
  }
- NOTA: Se la variabile non raggiunge la condizione inserita dentro il ciclo si crea un loop.



# Cicli: While

- Sintassi:
  While(condizione){
  azione1;
  azione2;
  //azione per far variare la condizione
  }
- NOTE: Il ciclo while dura fino a quando la condizione è vera. Per far questo dobbiamo necessariamente far variare la condizione all'interno del ciclo: Esempio:
  $a = 0;
  while($a<=3){
  print "ciao $a <br>";
  $a++;
  }
- In questo caso il ciclo while continua fino a quando $a non raggiunge il valore 3




# Cicli: Do While

- È' simile al ciclo While MA mentre il ciclo WHILE può non essere eseguito, il ciclo DO WHILE si esegue sempre, almeno per una volta.
  Questo perché il ciclo Do While inserisce prima le azioni da fare e dopo la condizione. Il server esegue le prime istruzioni, poi legge la condizione e se è sempre vera esegue nuovamente le istruzioni
- Sintassi:
  do {
  azione1;
  azione2;
  //azione per far variare la condizione
  }
  while(condizione)


# Cicli: Switch

- L'istruzione Switch non è un vero e proprio ciclo. Si usa se ci sono più alternative da vagliare e non si vogliono inserire più cicli if annidati
- Sintassi:
- switch ($a) {
  case condizione;
  lista azioni;
  default:
  lista azioni; //entra qui se nessuna condizione è verificata
  }
- Supponiamo per esempio di inserire una variabile e di dover agire in maniera diversa se questa variabile corrisponde a due valori. Con l'istruzione if dovremo scrivere due cicli annidati, con switch ne basta uno
- switch ($a) {
  case 'ciao':
  print "ci vediamo presto";
  break;
  case 'addio':
  print "non torni più?";
  break;
  default:
  print "forse tornerai";
  }





# Break

- Questa funzione termina l'esecuzione di un ciclo. Accetta un argomento opzionale che definisce, nel caso di cicli annidati, il livello del ciclo che è da interrompere.
- Esempio:
- $i = 0; // Uso dell'argomento opzionale
  while (++$i) {
  switch ($i) {
  case 5:
  echo "At 5<br>\n";
  break 1; /* Interrompe solo switch. */
  case 10:
  echo "At 10; quitting<br>\n";
  break 2; /* Interrompe switch e while. */
  default:
  break;
  }
  }



# Continue

- Questo comando serve per interrompere il ciclo senza uscire, infatti ritorna all'inizio del ciclo e continua il suo lavoro. NON azzera però il valore delle variabili, si usa se uno o più valori non devono influire.
  Come la funzione 'break', accetta un argomento opzionale che definisce, nel caso di cicli annidati, il livello del ciclo che è da interrompere.
- Esempio:

```
for ($a=1; $a<=4; $a++){
        if ($a==3)
            continue;
        else
            print "ciao $a <br>";
}
```

- In questo caso otteniamo:
  ciao 1
  ciao 2
  ciao 4





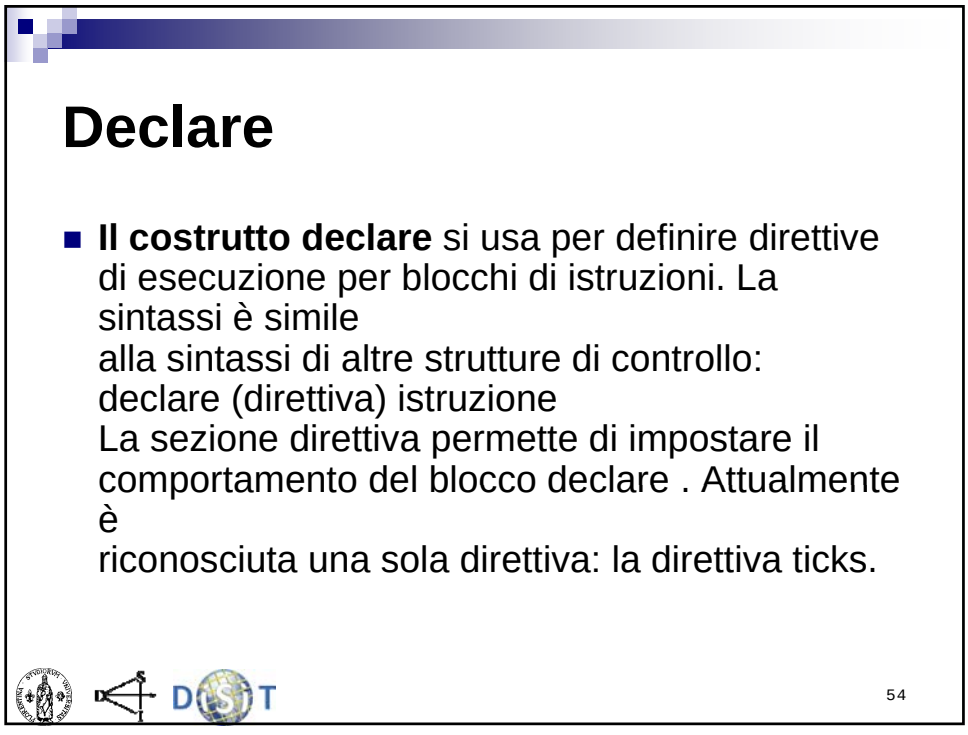
# Continue

- Si utilizza per **interrompere l'esecuzione del ciclo corrente** e continuare con l'esecuzione all'inizio del ciclo
  successivo; continue accetta un argomento numerico opzionale che definisce, nel caso di cicli annidati, il
  numero di cicli da interrompere e da cui iniziare l'esecuzione dell'iterazione successiva.



# Declare

- **Il costrutto declare** si usa per definire direttive di esecuzione per blocchi di istruzioni. La sintassi è simile
alla sintassi di altre strutture di controllo:
declare (direttiva) istruzione
La sezione direttiva permette di impostare il comportamento del blocco declare . Attualmente è
riconosciuta una sola direttiva: la direttiva ticks.



# Require(file) / Include(file)

- Includono e valutano il file specifico. Sono identiche in ogni senso eccetto per come trattano gli errori: include() produce un Warning mentre require() restituisce un Fatal Error.
- In altre parole, se si vuole che un file mancante fermi l'esecuzione della pagina È' NECESSARIO usare require(). Include() non si comporta in questo modo, lo script continuerà nonostante tutto. NOTA: Attenzione al path.
  Esempio:
- FILE 1 (nome file = 'da_includere.php' )
  <?php //file da includere,
  echo "Istruzioni contenute nel file da_includere.php </br>";
  $Lesson = "file da_includere incluso</br>";
  ?>
- FILE 2 (nome file = 'esempio.php' )
  <?php
  require 'da_includere.php'; // require (' da_includere.php '); //altro metodo di utilizzo di require()
  /*includendo il file ' da_includere.php ' l'interprete php esegue prima
  tutte le istruzioni contenute in tale file e poi continua ad eseguire le istruzioni contenute nel file 'esempio.php' */
  echo $Lesson; // adesso posso usare la variabile contenuta nel file 'da_includere.php'




# Unset()

- Questa funzione serve per eliminare una variabile
- Esempio:
  - $var_name = 5; unset($var_name);
    $colori = array(3 => 'giallo', 'verde', 'blu', 'viola');
    unset($colori[3]); /*elimina solo il primo elemento e NON cambia le chiavi, infatti:*/
    echo "<br>$colori[3]"; //non stampa niente
    echo "<br>$colori[4]<br>"; //stampa la stringa 'verde'
    unset($colori); //elimina tutto l'array


# Sort()/Rsort()

- Sort() serve per ordinare gli elementi contenuti in un array. NOTA: riordina anche le chiavi.
- Rsort() esegue l'ordinamento inverso
- Esempio: //esempio con unset e array

```
$var_name = 5; unset($var_name);
$colori = array(3 => 'giallo', 'verde', 'blu', 'viola');
unset($colori[3]); //elimina solo il primo elemento
echo "<br>$colori[3]"; //non stampa niente
echo "<br>$colori[4]<br>"; //stampa la stringa 'verde'
sort($colori);//NOTA: riordina anche le chiavi
//stampa degli elementi ordinati
for ($i=0; $i<=3; $i++){
echo "<br>$colori[$i]";
}
// quello che segue è il ciclo for per stampare gli elementi senza usare sort
/*
for ($i=0; $i<=3; $i++){
$chiave = $i+3; //per stampare gli elementi senza usare sort
echo "<br>$colori[$chiave]";//senza sort
}
*/
unset($colori); //elimina tutto l'array
```




# ESEMPIO ("form.html" )

```
<html>
 <head>
   <meta http-equiv="Content-Type" content="text/html; charset=iso-
   8859-1">
   <title>Form</title>
 </head>
    <body >
   <h1> Prova Form </h1>
     <form action="action.php" method="POST">
     Il tuo Nome: <input type="text" name="name" value="" />
     La tua e-mail: <input type="text" name="email" value ="" />
     <input type="submit">
     </form>
     </body>
</html>
```


# ESEMPIO (**“action.php”**)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <title>form</title>
 </head>
     <body >
<?php
echo 'Ciao ';
echo $_POST["name"];
echo '<br> La tua e-mail è: ';
echo $_POST["email"];
?>
 </body>
</html>
```





# ESEMPIO2 (**“form.html” )**

```
<html>
 <head> <meta http-equiv="Content-Type" content="text/html; charset=iso-
    8859-1">    <title>Form2.it</title>
 </head>
    <body >
    <h1> Prova Form con tabella per l’allineamento </h1>
    <form action="action.php" method="POST">
     <table align="left">
          <tr>
            <td> Il tuo Nome: </td>
            <td> <input  type="text" name="name" value="" />  </td>
          </tr>
            <td> La tua e-mail: </td>
            <td> <input type="text" name="email" value ="" /> </td>
          <tr>
            <td> <input type="submit"> </td>
          </tr>
     </table>
    </form>
    </body>
```


# Esempio 3 (“form.html” )

```
[…]
<form action="action.php" method="POST">
    <table align="left">
        <tr>
          <td> A quale anno di corso sei interessato? </td>
          <td>
                <select name="year">
                <option value="2004">2004</option>
                <option value="2005">2005</option>
                <option value="2006">2006</option>
                <option value="2007">2007</option>
                <option value="2008">2008</option>
                </select>
          </td>
        <tr>

     </table>
    </form>
[…]
```

NOTA: nel file action.php si deve usare la variabile $year





# ESEMPIO 4 (“form.html” )

```
[…]
<form action="action.php" method="POST">
    <table align="left">
        <tr>
          <td>Scegli un colore: </td>
          <td>
                <select name="colori[]" multiple size="5" >
                <option value="Red">Red
                <option value="Yellow">Yellow
                <option value="Blue">Blue
                <option value="Green">Green
                <option value="White">White
                <option value="Black">Black
                </select>
          </td>
        </tr>

     </table>
    </form>
[…]
```

NOTA: nel file action.php si deve usare la variabile colori



# Esempio 4 (**“action.php”**)

- Ecco come usare la variabile colori nel file **action.php**:

```
[…]

// un elenco a scelta multipla restituisce un array di valori
// se nessun valore è stato selezionato però
// $_POST['colori'] non sarà settato

if(isset($_POST['colori']))
    $valori_selezionati = implode($_POST['colori'],', ');
else
     $valori_selezionati = 'Nessun valore selezionato';

echo($valori_selezionati);
[…]
```



# ESEMPIO 5 **(“form.html” )**

```
[…]
      <p> campo di testo: <br>    <input type="text" name="textfield">
      </p>
      <p>  area di testo:<br>       <textarea name="textarea"></textarea>
      </p>
      <p>  <input type="checkbox" name="checkbox" value="checkbox">
            casella di controllo
      </p>
      <p>
        Fai una delle seguenti scelte: <br>
         <label> <input type="radio" name="radio" value="1">
                 Pulsante di scelta 1
         </label>
         <br>
         <label> <input type="radio" name="radio" value="2">
                 Pulsante di scelta 2
         </label>
          <br>
      </p>
[…]
```

NOTE: nel file action.php si devono usare le variabili textfield, textarea, checkbox, radio

# ESEMPIO 5 (**“action.php”**)

- Ecco come usare la variabili textfield e textarea nel file **action.php**:

```
[…]
// textfield
    // si convertono alcuni caratteri illeciti per l'HTML
    $testo = htmlspecialchars($_POST['textfield']);
    echo('<br>campo di testo: ' . $testo );
    echo('<br>');
// textarea
    $testo = htmlspecialchars($_POST['textarea']);
    //si convertono gli 'a capo' con dei <br>
    $testo = nl2br($testo);
    echo('area di testo: ' .$testo);
    echo('<br>');
[…]
```





# ESEMPIO 5 (**“action.php”**)

- Ecco come usare la variabili checkbox, radio nel file **action.php**:

```
// checkbox
    // se la casella non è stata selezionata
    // allora $_POST['checkbox'] non sarà settato
    $casella = isset($_POST['checkbox']) ? 'selezionata' : 'non selezionata';
    echo('casella di controllo: ' . $casella);
    echo('<br>');
// radio
    // se neanche un bottone è stato selezionato allora $_POST['radio'] non sarà
     settato
    if(isset($_POST['radio']))
       $pulsante = 'selezionato il numero ' . $_POST['radio'];
    else
       $pulsante = 'nessun pulsante selezionato';
    echo('Pulsante di scelta: ' . $pulsante);
    echo('<br>');
[…]
```



# ESEMPIO 6 **(“form_GET.html” )**

```
<html>
 <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
      <title>Form_get.it</title>
 </head>
   <body >
      <h1>
      Prova Form con Metodo get
      </h1>

      <form action="action_get.php" method="get">
                Name: <input type="text" name="name" />
                Age: <input type="text" name="age" />
                <p>
                Che Notizia vuoi?
                          <select name="id_notizia">
                          <option value="1">1</option>
                          <option value="2">2</option>
                          <option value="3">3</option>
                          </select>
                </p>
                <p>
                <input type="submit" />
                </p>
      </form>
   </body>
</html>
```

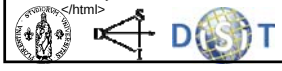


# ESEMPIO 6 (**“action_GET.php”**)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
 <head>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
      <title>form</title>
 </head>
    <body >
<?php
 if(!$_GET)
 {
    echo 'Seleziona una notizia da leggere: <br><br>';
    echo '<a href="action_get.php?id_notizia=1">Notizia 1</a> <br>';
    echo '<a href="action_get.php?id_notizia=2">Notizia 2</a> <br>';
    echo '<a href="action_get.php?id_notizia=3">Notizia 3</a> <br>';
 }
 else
 {
    $notizie = array(1=>'Questo è il testo della notizia numero 1',
                     'Questo è il testo della notizia numero 2',
                     'Questo è il testo della notizia numero 3');

    $id_notizia = $_GET['id_notizia'];

    echo $notizie[$id_notizia] . '<br><br>';
   echo '<a href="index.php">Torna all\'indice</a>';
 }
 ?>
   </body>
 </html>
```



# Metodologie informatiche per le discipline umanistiche

## Programmazione WEB PHP

Prof. Paolo Nesi
nesi@dsi.unifi.it
Dr.sa. Michela Paolucci
paolucci@dsi.unifi.it
Dr. Emanuele Bellini
ebellini@dsi.unifi.it

lab. DSI-DISIT
055 4796425