Programmazione Web

Il linguaggio PHP

Il nostro obiettivo

- Insegnarvi tutto il PHP? Non esattamente
- Obiettivo: insegnarvi ad interagire via web con una base dati
  - Accedere ai dati inseriti dall'utente nei form HTML
  - Interagire con un DBMS (MySQL in particolare): connettersi ad una base dati, inviare la query, memorizzare il risultato della query, …
  - Accedere alle tabelle restituite dal DBMS
  - Costruire la pagina HTML da visualizzare sul browser, costituita da istruzioni HTML e dati estratti dalla base dati

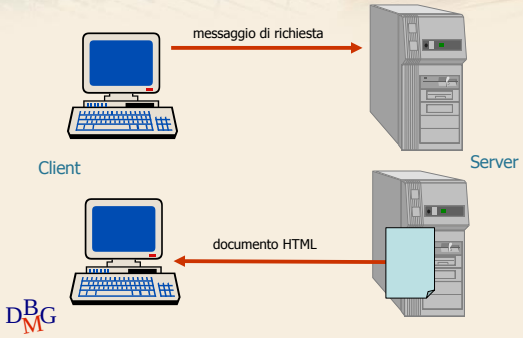
Contenuti

- Panoramica del linguaggio PHP
  - Struttura di un programma
  - Variabili e tipi (array associativi)
  - Espressioni e operatori
  - Strutture di controllo (foreach)
- Acquisizione dei parametri dai form HTML

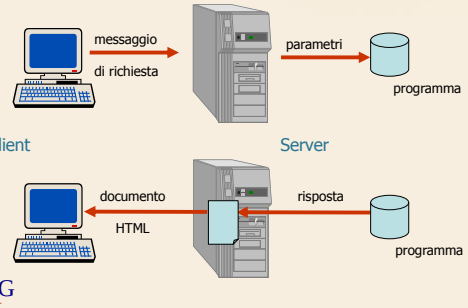
Che cos'è il PHP

- Nasce nel 1994
  - Prima: **P**ersonal **H**ome **P**age,
  - Oggi: **P**HP **H**ypertext **P**reprocessor
- Creato appositamente per lo sviluppo di pagine web dinamiche
- Moltissime risorse utili, e.g.
  - http://php.html.it/guide/
  - http://www.web-link.it/php/

Pagine Web statiche



Pagine Web dinamiche



Obiettivo principale

- Il PHP ha come obiettivo principale produrre codice HTML
  - In particolare, produrre codice HTML "condizionato" ai risultati di una elaborazione, che dipendono dall'input degli utenti, dai dati contenuti in un database, …
- Il codice PHP viene inserito all'interno del codice HTML
  - Il codice PHP viene eseguito sul server Web e il risultato (HTML e risultato dello script) viene inviato al browser

Perché usare il PHP?

- È disponibile per molte piattaforme, diverse per
  - Hardware (Intel, Sparc, Mac, ecc....)
  - Sistema operativo (Linux,Unix, Windows, ecc...)
  - Web server (Apache, IIS, IPlanet, ecc...)
- Il codice PHP è "altamente portabile"
- L'interprete PHP è Open Source
  - Gratuito, ampia disponibilità di strumenti, supporto, sviluppatori, comunità di utenti
- Abbastanza facile da imparare, molto facile se si conosce il C
- In grado di interagire con vari Database Management Systems (MySql, Postgres, Oracle, ...)

Primo esempio

- File di testo con estensione .php

```
<html>
  <head>
    <title>Ciao mondo!</title>
  </head>
  <body bgcolor="#FFFFFF">
    <?php
      // Questo è codice PHP
      echo "<h1> Ciao Mondo !</h1>";
    ?>
  </body>
</html>
```

Ciao Mondo !

Primo esempio

Ciao Mondo !

- Se visualizzo il sorgente sul browser…

```
<html>
  <head>
    <title>Ciao mondo!</title>
  </head>
  <body bgcolor="#FFFFFF">
    <h1> Ciao Mondo !</h1>   </body>
</html>
```

- Perché?
  - Il browser visualizza il risultato dell'esecuzione del file php, NON il file PHP

"Stampa" di stringhe

- Uno dei compiti più importanti (e frequenti) del codice PHP è creare codice HTML che poi verrà visualizzato dal browser
  - Costrutti echo e print

Ciao Mondo !

```
<?php

  // Producono tutti lo stesso output
  echo "<h1> Ciao mondo !</h1>";
  print "<h1> Ciao mondo !</h1>";
  echo ("<h1> Ciao mondo !</h1>");
  print ("<h1> Ciao mondo !</h1>");

?>
```

Una breve parentesi: EasyPHP

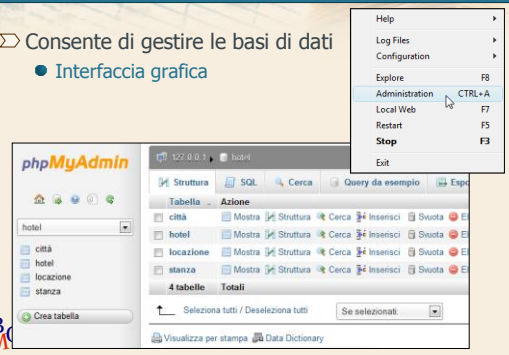
- EasyPHP è una piattaforma di sviluppo Web (ambiente di sviluppo web-database) che comprende
  - Un web server (Apache)
  - Un database management system (MySQL)
  - Un interprete di script PHP
  - Un amministratore grafico di db MySQL (phpMyAdmin)
- Permette di far funzionare localmente degli script PHP senza connettersi ad un server esterno

Una breve parentesi: EasyPHP

- EasyPHP installa tutti i software necessari per la progettazione e il funzionamento di un sito web in locale
  - Il PC diventa client e server
- Il web server Apache crea automaticamente di default un dominio virtuale (in locale) all'indirizzo di localhost (http://127.0.0.1)
  - Non è necessario essere connessi ad Internet per utilizzare EasyPHP

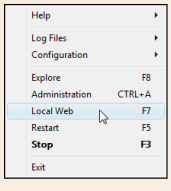
EasyPHP : amministrazione DB

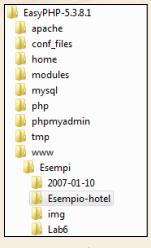
- Consente di gestire le basi di dati
  - Interfaccia grafica



EasyPHP : server Web locale

- EasyPHP crea una directory "www" nella quale si devono copiare gli script PHP





- "Local Web" consente di accedere direttamente alla pagina web http://127.0.0.1/ e di vedere i file copiati nella directory "www"

Tag per includere codice PHP

- Il codice PHP può essere posizionato in qualsiasi punto della pagina HTML
- Deve essere racchiuso fra tag

```
<?php
  // Questo è codice PHP
  echo "<h1> Ciao Mondo !</h1>";
?>
```

```
<script language="PHP">
  // Questo è codice PHP
  echo "<h1> Ciao Mondo !</h1>";
</script>
```

```
<?
  // Questo è codice PHP
  echo "<h1> Ciao Mondo !</h1>";
?>
```

Un altro esempio

- Visualizzare la data corrente
- In modo statico
  - E domani?

```
<html>
  <body>
    Today is 09/12/2011
  </body>
</html>
```

- In modo dinamico
  - Si aggiorna in tempo reale

```
<?php
  // formato GG/MM/AAAA

  $today = date("d/m/Y");
  echo $today;
?>
```

Analisi del codice

- In uno script si usano
  - Inizio fine script: <?php ?>
  - Commenti: //… **monolinea**
    /* multilinea */
  - Variabili: **$today**
  - Operatori e costrutti del linguaggio: **echo**
  - Funzioni: **date()**
  - Fine statement (obbligatorio): **;**

```
<?php
  // formato GG/MM/AAAA

  $today = date("d/m/Y");
  echo $today;
?>
```

Le variabili

- Una variabile è un simbolo o un nome che rappresenta un valore
- Una variabile può rappresentare diversi tipi di valore
  - Numero intero, numero reale, carattere, …
  - Il tipo di dato può cambiare durante l'esecuzione del programma
- Quando un programma viene eseguito le variabili sono sostituite da dati reali
  - Lo stesso programma può così elaborare diversi insiemi di dati

DBMG

Le variabili

- In PHP il nome delle variabili è preceduto dal simbolo del dollaro ('$')
- PHP non richiede che le variabili vengano dichiarate prima del loro uso
  - Maggiore flessibilità rispetto ad altri linguaggi di programmazione
- Le variabili **sono case sensitive**
- La function **isset()** controlla che la variabile argomento sia definita (**true**) o meno (**false**)

DBMG

```php
<?php
  $a = 5;
?>
```

```php
<?php
  $a = 9;
  $b = 4;
  $c = $a * $b;
  echo "Il risultato dell'operazione (9 * 4) è ";
  echo $c;
?>
```

Tipi di dato

- Il PHP supporta diversi tipi di dato
  - Boolean: vero o falso
  - Integer: numeri decimali
  - Float: numeri in virgola mobile
  - String
  - Array
  - Object
  - Resource

```php
<?php
  $a = true;
  $b = false;
  $c = 18;
  $d = -18;
  $e = 9.876;
  $f = 9.87e6;
?>
```

DBMG

Tipi di dato

- I tipi di dato non devono essere impostati dal programmatore ma sono assunti automaticamente dal compilatore PHP
  - È possibile verificare il tipo si usano le funzioni **is_int()**, **is_float()**, **is_bool()**, **is_string()**
- PHP ha un meccanismo di casting implicito ma supporta anche quello esplicito (sintassi tipo C)

```php
<?php
  $a = 56;
  $b = 12;
  $c = $a/$b;  // $c è 4.66666 (float)
  $d = (int)($a/$b);  // $d è 4 (int)
?>
```

DBMG

Gli array

- Un array (vettore) è una variabile complessa che contiene una serie di valori, ciascuno dei quali caratterizzato da una chiave (o indice) che lo identifica univocamente
- Il PHP supporta sia gli array scalari sia gli array associativi
  - Gli array scalari identificano ogni elemento del vettore con un numero determinato dalla sua posizione all'interno di una sequenza
  - Gli array associativi identificano ogni elemento del vettore con una chiave a cui l'elemento è associato in modo univoco

DBMG

Gli array

- Esempio di array scalare

```php
<?php
  $colori = array('bianco', 'nero', 'giallo', 'verde', 'rosso');

  echo $colori[1]; // stampa 'nero'
  echo $colori[4]; // stampa 'rosso'
?>
```

- Esempio di array associativo
  - La chiave può essere una stringa o un intero

```php
<?php
  $a = array(
    "nome" => "Mario",
    "cognome" => "Rossi",
    "email" => "mario@rossi.com"
  );

  echo $a["nome"];  // stampa "Mario"
  echo $a["email"]; // stampa "mario@rossi.com"
?>
```

DBMG

Gli array

- È possibile costruire array multidimensionali

```php
<?php
  $a = array(
    "primo" => array(
                 "nome" => "Mario",
                 "cognome" => "Rossi",
                 "email" => "mario@rossi.com"
               ),
    "secondo" => array(
                   "nome" => "Marco",
                   "cognome" => "Verdi",
                   "email" => "mario@verdi.com"
                 )
  );

  echo $a["secondo"]["email"];  // stampa "mario@verdi.com"
?>
```

DBMG

Gli array

- Gli elementi di un array possono anche essere disomogenei

```php
<?php
  $mix = array( 1, "ciao", 3.14, array( 1, 2, 3 ) );
?>
```

- In PHP è molto facile aggiungere o rimuovere elementi di un vettore

```php
<?php
  $x = array(25, 50);

  $x[ ] = 75;   // aggiunge un elemento nella prima posizione disponibile
  $x[4] = 125;  // aggiunge un elemento nella posizione indicata
  echo $x[3];   // Errore!!! (l'elemento non esiste)

  unset($x[1]); // rimuove l'elemento specificato
  unset($x[1]); // rimuove l'intero vettore
?>
```

DBMG

Principali funzioni per gli array

- is_array() ritorna true se il parametro è un array
- count() ritorna il numero di elementi nell'array
- sort() ordina l'array. Altri parametri opzionali specificano come ordinare.
- explode(), compact() creano un array da: stringhe (spezzettandole) o nomi di variabili (compact)
- extract() crea variabili da un array
- array_key_exists(key,array) verifica che un elemento ***key***, esista nell'array ***array***

DBMG

Espressioni e operatori

- Operatori aritmetici

```php
<?php
  $a = $x + 7; // addizione
  $b = $x - 2; // sottrazione
  $c = $x * 6; // moltiplicazione
  $d = $x / 2; // divisione
  $e = $x % 4; // modulo (resto della divisione)

  $x += 4;  // incrementa $x di 4 (equivale a $x = $x + 4)
  $x -= 3;  // decrementa $x di 3 (equivale a $x = $x - 3)
  $x /= 5;  // equivale a $x = $x / 5
  $x *= 4;  // equivale a $x = $x * 4
  $x %= 2;  // equivale a $x = $x % 2

  $a++; // incrementa di 1
  ++$a; // incrementa di 1
  $a--; // decrementa di 1
  --$a; // decrementa di 1
?>
```

DBMG

Espressioni e operatori

- Operatori logici

```php
<?php
  $x = $a && $b;  // and logico
  $x = $a || $b;  // or logico
  $x = $a xor $b; // xor logico
  $x = !$a;       // not logico
?>
```

- Operatori di confronto

```php
<?php
  if ($a == $b) ...   // uguale
  if ($a != $b) ...   // diverso
  if ($a >  $b) ...   // maggiore
  if ($a >= $b) ...   // maggiore o uguale
  if ($a <  $b) ...   // minore
  if ($a <= $b) ...   // minore o uguale
?>
```

DBMG

Espressioni e operatori

- Operatori sulle stringhe
  - Concatenazione

```php
<?php
  $x = $x . $a;   // il valore della stringa $a viene concatenato a $x
  $x .= $a;       // equivalente
?>
```

- Esempio

```php
<?php
  $Nome    = 'Mario';
  $Cognome = 'Rossi';
  echo $Nome.' '.$Cognome;       // scrive Mario Rossi
  echo "$Nome $Cognome";         // scrive Mario Rossi
  echo $Nome[0].'. '.$Cognome;   // scrive M. Rossi
  echo "$Nome[0]. $Cognome";     // scrive M. Rossi
?>
```

DBMG

Le strutture di controllo

- Permettono l'esecuzione condizionale di parti di programma
- Permettono l'esecuzione iterativa di parti di programma
- Valutano determinate condizioni
- Strutture di controllo in PHP
  - if, if..else, if..elseif
  - switch
  - while, do..while
  - for
  - foreach

Le condizioni

- Una condizione è un'espressione che genera un valore booleano (vero o falso)
  - Utilizzano gli operatori di confronto e gli operatori booleani
- Sono equivalenti a falso (false)
  - Il valore booleano false
  - Il numero intero 0 e il numero reale 0.0
  - La stringa vuota ("") e la stringa "0"
  - Un array vuoto
- Ogni altro valore è considerato vero (true)

Il costrutto if ed if… else

- Se la condizione espressa nel blocco IF è vera, il blocco di operazioni viene eseguito

```
<?php
  $a = 2;
  $b = 2;
  if ($a == $b) {
    echo "\$a è uguale a \$b e valgono $a.\n";
  }
?>
```

- Se la condizione espressa nel blocco IF è vera, il blocco di operazioni viene eseguito, altrimenti viene eseguito il ramo ELSE

```
<?php
  $a = 2;
  $b = 3;
  if ($a == $b) {
    echo "\$a è uguale a \$b e valgono $a.\n";
  } else {
    echo "\$a è diverso da \$b.\n\$a vale \"$a\" mentre \$b vale \"$b\".\n";
  }
?>
```

Il costrutto if .. elseif

- Consente di scegliere fra più opzioni

```
<?php
  $a = 2;
  $b = 3;
  if ($a == $b) {
    echo "\$a è uguale a \$b.\n";
  } elseif ($a > $b) {
    echo "\$a è maggiore di \$b.\n";
  } else {
    echo "\$a è minore di \$b.\n";
  }
?>
```

Il costrutto switch

- Permette di prevedere diversi valori possibili per un'espressione ed eseguire codice specifico in base al valore
  - Sostituisce una serie di if
  - break: forza l'uscita dal blocco switch
  - default: è opzionale

```
<?php
  switch ($nome) {
    case 'Luca':
    case 'Giorgio':
    case 'Franco':
      echo "Ciao, vecchio amico!";
      break;
    case 'Anna':
      echo "Ciao, Anna!";
      break;
    case 'Paolo':
      echo "Piacere di conoscerti, Paolo!";
      break;
    default:
      echo "Benvenuto, chiunque tu sia";
  }
?>
```

Il ciclo while

- Il blocco di istruzioni all'interno del while viene eseguito fino a che la condizione rimane vera
  - È possibile che il ciclo non venga mai eseguito, nel caso in cui la condizione sia falsa sin dall'inizio
  - In generale il blocco di istruzioni modifica i parametri usati nella condizione

```
<?php
  $mul = 1;
  while ($mul <= 10) {
    $ris = 5 * $mul;
    echo "5 * $mul = $ris<br>";
    $mul++;
  }
?>
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Il ciclo while

- Il blocco di istruzioni viene eseguito fino a che la condizione rimane vera

```php
<?php
  $mul = 1;
  while ($mul <= 10) {
    $ris = 5 * $mul;
    echo "5 * $mul = $ris<br>";
    $mul++;
  }
?>
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Il ciclo for

- Consente di ripetere un blocco di istruzioni definendo direttamente
  - Le istruzioni di inizializzazione, eseguite una sola volta all'ingresso del ciclo
  - La condizione, che deve essere vera per eseguire il blocco di istruzioni
  - L'aggiornamento, eseguito al termine di ogni iterazione
- Può essere sempre riscritto come ciclo while

```php
<?php
  for ($mul = 1; $mul <= 10; ++$mul) {
    $ris = 5 * $mul;
    echo "5 * $mul = $ris <br/>";
  }
?>
```

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Il ciclo foreach

- Ciclo creato appositamente per facilitare l'accesso agli array
  - Corrisponde ad un ciclo for sugli elementi di un array

```php
<?php
  $a = array("Andrea", "Paola", "Roberto", "Alice", "Sara");
  foreach ($a as $value) {
    echo "$value <br />";
  }
?>
```

```
Andrea
Paola
Roberto
Alice
Sara
```

Il ciclo foreach per gli array associativi

```php
<?php
  $anno = array("Gennaio" => 31,
                "Febbraio" => 28,
                "Marzo" => 31,
                "Aprile" => 30,
                "Maggio" => 31,
                "Giugno" => 30,
                "Luglio" => 31,
                "Agosto" => 31,
                "Settembre" => 30,
                "Ottobre" => 31,
                "Novembre" => 30,
                "Dicembre" => 31);

  foreach ($anno as $mese => $giorni) {
    echo "$mese ha $giorni giorni. <br />";
  }
?>
```

```
Gennaio ha 31 giorni.
Febbraio ha 28 giorni.
Marzo ha 31 giorni.
Aprile ha 30 giorni.
Maggio ha 31 giorni.
Giugno ha 30 giorni.
Luglio ha 31 giorni.
Agosto ha 31 giorni.
Settembre ha 30 giorni.
Ottobre ha 31 giorni.
Novembre ha 30 giorni.
Dicembre ha 31 giorni.
```

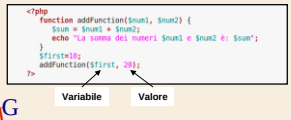
Funzioni definite dall'utente

- Formato tipo sintassi C
- Il nome della funzione è case **insensitive**
- La lista parametri è **opzionale** ed usa come separatore la *virgola*
  - La lista dei parametri sono variabili (che iniziano per $), oppure i valori direttamente
- Il valore di ritorno è **opzionale** ed è indicato dall'istruzione di *return*
- Quando chiamata, la funzione deve essere già stata processata dall'interprete PHP

Chiamate di funzione


```
La somma dei numeri 10 e 20 è: 30
```

Alternativa senza l'utilizzo dell'istruzione return




```
La somma dei numeri 10 e 20 è: 30
```

Regole di scoping

- Lo **scope** definisce dove le variabili sono visibili
- Variabili definite dentro la funzione
  - Local scope: dentro la funzione nella quale è definita (**inclusi** gli argomenti della funzione)
- Variabili definite al di fuori delle funzioni
  - Scope: tutto il codice al di fuori delle funzioni (non visibili dentro le funzioni)
- Variabili globali dentro le funzioni
  - Una variabile dichiarata dentro una funzione con il qualificatore **global** ha scope globale

DBMG

Esempio

```
<html>
   <body>
      <?php
         $acc = 0;
         function sum($x) {
            global $acc;
            $acc += $x;
         }
         sum(10);
         echo "La somma è: $acc";
         sum(10);
         echo "La somma è: $acc";
      ?>
   </body>
</html>
```

```
La somma è: 10
La somma è: 20
```

DBMG

Variabili «superglobal»

- Variabili globali predefinite
  - Sono visibili e accessibili dovunque
  - Hanno la forma di array associativi
  - Tipicamente usate per informazioni di ambiente
- Esempi:
  - $GLOBALS tutte le variabili globali correntemente definite nello script
  - $_GET tutte le variabili passate allo script via HTTP **GET**
  - $_POST tutte le variabili passate allo script via HTTP **POST**

DBMG

Passaggio parametri

- I parametri possono essere passati per **valore** o **indirizzo**
- Il default è il passaggio per valore
- I parametri passati per **indirizzo** sono preceduti da **&**

```
<html>
   <body>
      <?php
         function addExtra(&$string) {
            $string .= 'and extra.';
         }
         $str = 'A string, ';
         add_some_addExtra($str);
         echo $str;
      ?>
   </body>
</html>
```

```
A string, and extra.
```

DBMG

Risultato «by reference»

- Il risultato (return) puà essere passato per valore o per indirizzo
- Il default è il passaggio per valore
  - Se il nome della funzione è preceduto da &, la funzione ritorna un reference

```
<html>
   <body>
      <?php
         function &get_x_ref() {
            static $x = 30;
            return $x;
         }
         $y = &get_x_ref(); // $y è un alias di $x
         echo $y;
      ?>
   </body>
</html>
```

```
30
```

DBMG

Argomenti con valore di default

- Un valore di default può essere specificato per ogni argomento (come un assegnamento)
  - Se non è specificato il valore di default viene utilizzato per quell'argomento

```
<html>
   <body>
      <?php
         function conc($a, $b, $sep=', '){
            return ($a.$sep.$b."<br>");
         }
         echo conc("First", "Second");
         echo conc("First", "Second", '; ');
      ?>
   </body>
</html>
```

```
First, Second
First; Second
```

DBMG

Terminazione dello script

- Le funzioni exit() , die() terminano l'esecuzione dello script
  - Possono prendere una stringa o un intero come parametri
  - La stringa è stampata prima della terminazione
  - L'intero è ritornato
- Esempio:

```
exit("connection failed");
```

PHP e form HTML

```
<form name="datiUtenti" action="paginaRisposta.php" method="GET">
    Elementi di input
</form>
```

- Tag "form" con alcuni attributi
  - Name: nome del form
  - Action: nome del programma che elaborerà i dati del form
  - Method: modalità in cui vengono passati i parametri dal form al programma (può essere "GET" o "POST")
- All'interno del form ci sono più elementi di input

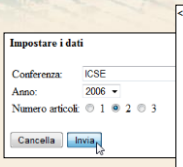
Accesso ai dati del form

- Lo script PHP di destinazione accede ai valori immessi dall'utente tramite alcune variabili speciali chiamate "**superglobali**": gli array associativi $_GET, $_POST e $_REQUEST
  - "Superglobali" perché sono accessibili anche dall'interno di eventuali funzioni
- Metodo GET
  - I valori contenuti nella query string vengono memorizzati nell'array associativo $_GET
  - Ogni parametro del form diventa un campo dell'array associativo $_GET

Accesso ai dati del form

- Metodo POST
  - Ogni parametro del form diventa un campo dell'array associativo $_POST
- L'array associativo $_REQUEST contiene $_GET, $_POST e $_COOKIE
  - Anche se non è la stessa cosa, nella pratica può essere usato con qualunque metodo, in alternativa a $_GET o a $_POST

Esempio: metodo GET




```
<form method="get" action="test.php">
  <table>
    <tr>
      <td> Conferenza: </td>
      <td> <input type="text" name="conf" size="20"> </td>
    </tr>
    <tr>
      <td> Anno: </td>
      <td>
        <select name="anno">
          <option value="2005">2005</option>
          <option value="2006">2006</option>
        </select>
      </td>
    </tr>
    <tr>
      <td> Numero articoli: </td>
      <td>
        <input type="radio" name="numero" value="1"> 1
        <input type="radio" name="numero" value="2" checked> 2
        <input type="radio" name="numero" value="3"> 3
      </td>
    </tr>
  </table>
  <br />
  <input type="reset" value="Cancella">
  <input type="submit" value="Invia">
</form>
```

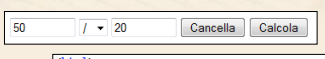
Esempio: metodo GET

- File test.php

```
<html>
<head>
<title>Risultato</title>
</head>

<body>

  <?php
    $conf = $_GET["conf"];
    $anno = $_GET["anno"];
    $num = $_GET["numero"];

    echo "Nell'anno $anno hai presentato $num articoli alla ";
    echo "conferenza $conf.";
  ?>

</body>
</html>
```

Nell'anno 2006 hai presentato 2 articoli alla conferenza ICSE.

Esempio: calcolatrice




```
<html>
 <head>
  <title>Calcolatrice</title>
 </head>
 <body>
  <form method="get" action="calculator.php">
   <input type="text" size="8" maxlength="8" name="val1" value="1">
   <select name="op">
    <option value="sum">+</option>
    <option value="sub">-</option>
    <option value="mul">*</option>
    <option value="div">/</option>
   </select>
   <input type="text" size="8" maxlength="8" name="val2" value="1">
   <input type="reset" value="Cancella">
   <input type="submit" value="Calcola">
  </form>
 </body>
</html>
```

Esempio: calcolatrice

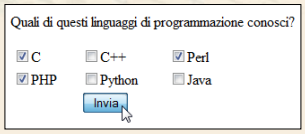
- File calculator.php

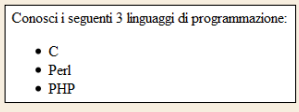
```
<html>
 <head><title>Risultato</title></head>

 <body>

  <?php

   // controllo dei dati in ingresso
   if( !is_numeric($_REQUEST["val1"]) || !is_numeric($_REQUEST["val2"]) ){
    echo '<font color="red"><h3>Non &egrave; un numero!</h3></font>';
    return;
   }

   if( $_REQUEST["op"]=="div" && $_REQUEST["val2"]==0 ){
    echo '<font color="red"><h3>Divisione per zero!</h3></font>';
    return;
   }
```

Esempio: calcolatrice

```
   // esecuzione dell'operazione richiesta
   if($_REQUEST["op"]=="sum"){
    $result = $_REQUEST["val1"] + $_REQUEST["val2"];

   } else if($_REQUEST["op"]=="sub"){
    $result = $_REQUEST["val1"] - $_REQUEST["val2"];

   } else if($_REQUEST["op"]=="mul"){
    $result = $_REQUEST["val1"] * $_REQUEST["val2"];

   } else if($_REQUEST["op"]=="div"){
    $result = $_REQUEST["val1"] / $_REQUEST["val2"];
   }

   // visualizzazione del risultato

   echo "<h3>Risultato: " . $result . "</h3>";
   ?>

  </body>
</html>
```

Esempio: scelta multipla





Esempio: scelta multipla

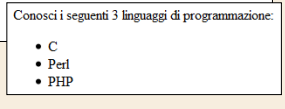
- Form HTML
  - Utilizza l'array langs[] invece di 6 variabili distinte

```
<p>Quali di questi linguaggi di programmazione conosci?</p>
<form action="select.php" method="post">
  <table width="250">
    <tr>
      <td><input type="checkbox" name="langs[]" value="C">C</td>
      <td><input type="checkbox" name="langs[]" value="C++">C++</td>
      <td><input type="checkbox" name="langs[]" value="Perl">Perl</td>
    </tr>
    <tr>
      <td><input type="checkbox" name="langs[]" value="PHP">PHP</td>
      <td><input type="checkbox" name="langs[]" value="Python">Python</td>
      <td><input type="checkbox" name="langs[]" value="Java">Java</td>
    </tr>
    <tr>
      <td></td>
      <td><input type ="submit" value="Invia"></td>
      <td></td>
    </tr>
  </table>
</form>
```

Esempio: scelta multipla

- Script PHP
  - L'array $_REQUEST ["langs"] contiene tutti i valori value selezionati (in questo caso C, Perl e PHP)

```
<?php
  $linguaggi = $_REQUEST["langs"];
  $num = count($linguaggi);
  echo "<p>Conosci i seguenti $num linguaggi di programmazione:<ul>";
  foreach ($linguaggi as $valore) {
    echo "<li>$valore </li>";
  }
  echo "</ul></p>";
?>
```



Controllo dei valori inseriti

- Prima di processare i dati forniti dall'utente conviene sempre **validarli**
  - Evita di processare dati errati
    - E.g., l'inserimento di un indirizzo email non correttamente formattato, o di un valore non previsto
  - Utile per evitare possibili attacchi informatici
    - E.g., l'inserimento di query SQL in un campo per visualizzare il contenuto del DB

Validazione dei dati

- La funzione filter_var() può essere usata per validare diversi tipi di dato. Se l'inserimento è corretto restituisce il valore true, altrimenti false
  - FILTER_VALIDATE_INT
  - FILTER_VALIDATE_FLOAT
  - FILTER_VALIDATE_BOOLEAN
  - FILTER_VALIDATE_EMAIL
- Oltre che validare la correttezza dei dati, è possibile verificare che rispettino dei vincoli
  - E.g., il limite minimo di età

Validazione dei dati

- Verificare che l'email inserita dall'utente sia corretta

```php
<?php
  function checkEmail(){
    if(array_key_exists("email",$_GET)){
      $email= $_GET["email"];
      $val = filter_var($email, FILTER_VALIDATE_EMAIL);
      if ($val == false) {
          return false;
      }
    }
    else{
      return false;
    }
    return true;
  }

  if(checkEmail()){
    print("Email valida");
  }
  else{
    print("Email non valida");
  }
?>
```

Verifica l'inserimento del campo email

Verifica la correttezza dell'email

Validazione dei dati

- Verificare che l'utente abbia almeno 14 anni

Approccio più modulare

```php
<?php
  function checkAge(){
    if(array_key_exists("age",$_GET) == False)
        return false;
    if (is_int($_GET["age"]) == False)
        return false;
    if($_GET["age"]<14){
      print("Devi avere almeno 14 anni per accedere al servizio");
      return False;
    }
    return true;
  }

  if(checkAge()){
    print("Età valida");
  }
  else{
    print("Età non valida");
  }
?>
```

Verifica l'età minima