# PHP

*breve excursus su uno dei linguaggi più utilizzati del web*

PHP, che significa "Hypertext Preprocessor", è un linguaggio di scripting server-side, Open Source, molto utilizzato, è specialmente indicato per lo sviluppo Web e può essere integrato nell'HTML. La sua sintassi è basata su quella di C, Java e Perl, ed è molto semplice da imparare ma nonostante ciò e nonostante l'obiettivo principale del linguaggio sia quello di permettere agli sviluppatori web di scrivere velocemente pagine web dinamiche, è estremamente potente e versatile; non a caso è molto diffuso nella comunità internet nonostante sia un linguaggio che ha più di 25 anni.

Essendo un linguaggio basato sull'azione del server, esattamente sull'interpretazione da parte del server degli script immersi nell'HTML (che viene ignorato dal parsing lato server), ha bisogno di particolari daemon in esecuzione per funzionare, fra questi apache è la miglior scelta possibile come server http dato che la risposta dell'elaborazione da parte dell'interprete viene trasportata sotto forma di semplice HTML al client che ne ha fatto richiesta.

Il linguaggio è nativo di ambienti linux ma viene anche lavorato in ambienti windows con software come WAMP o XAMPP che adattano un ambiente non esattamente confortevole per il php al suo utilizzo.

La "cartella madre" (http document root) nella quale il server apache cercherà i file da trasmettere, eventualmente ramificati in normalissime directory e subdirectory, e quindi dove operare o trasportare i file per poter essere interpretati e trasmessi in ambiente linux è ***/var/www/*** in alcune versione accompagnata da ***/html/***, mentre per attivare la trasmissione si deve navigare il file tramite semplice browser, descrivendo un opportuno URL come descritto in seguito.

Cosa vuol dire che può essere integrato in html? Ecco un esempio:

**Esempio 1-1. Un esempio introduttivo**

```
<html>
    <head>
        <title>Esempio</title>
    </head>
    <body>

        <?php
        echo "Ciao, sono uno script PHP!";
        ?>

    </body>
</html>
```

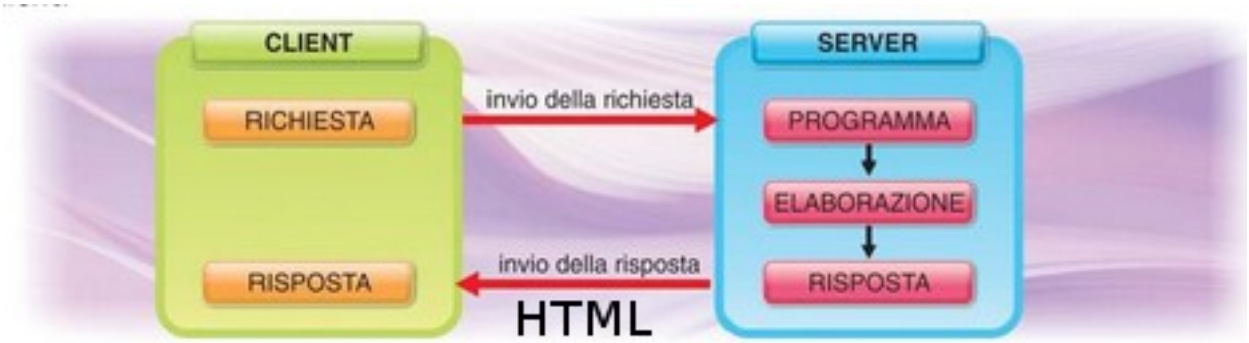
Si noti come questo esempio è differente da uno script scritto in altri linguaggi, tipo Perl o C; invece di scrivere un programma con parecchi comandi per produrre HTML, si scrive in HTML con qualche comando immerso per ottenere dei risultati (in questo semplice esempio, la visualizzazione di una frase). Il codice PHP è delimitato da speciali tag di start ***<?php*** e di end ***?>*** che ne indicano l'inizio e la fine e che consentono di passare dal modo HTML al modo PHP.

Ciò che distingue PHP da altri linguaggi di scripting del tipo client-side (ad esempio javascript) è che il codice viene eseguito nel server, quindi il computer riceve la pagina dove è presente lo script già elaborata dal server e non deve elaborarla durante la navigazione; grazie a ciò è possibile

ottenere un alleggerimento dei tempi di navigazione della pagina ed inoltre non c'è modo per gli utenti di conoscere quale sia l'elaborazione fatta dal web-server essendo loro fornito il "prodotto finito" e non il metodo/logica di sviluppo della pagina, come ad esempio avviene negli javascript.

In merito a ciò si ricorda che la dinamica (semplificata) di navigazione di una pagina html è la seguente: un web-server è un software reperibile presso un host ad un indirizzo IP ben noto (e traducibile dalla stringa di navigazione del browser tramite DNS: esempio www.google.it → 209.85.135.99) che attende su un certo "canale" (detto porta, ad esempio la porta 80 è per il protocollo http) che il client (computer che desidera ottenere un servizio) richieda una pagina; una volta ricevuta la richiesta il server avvia il processo legato al canale ed interpreta la pagina (scripting lato-server) inviando l'HTML risultante all'indirizzo IP richiedente che potrà disporne su un browser qualsiasi per poterla navigare.

Il tutto può essere fatto su una sola macchina operando in localhost (127.0.0.1), soprattutto a fini di sviluppo; in tal caso si consiglia di attivare la visualizzazione degli errori per favorire il debug o con apposite funzioni di pagina come error_reporting() oppure operando su file di configurazione php.ini andando a settare opportunamente la direttiva display_errors.




Ovviamente per operare l'elaborazione del php il server deve avere degli opportuni moduli che possano processare gli script ed attuarne le azioni richieste; in poche parole un computer che non ha installato un server e che non abbia installato l'interprete php non riuscirà a "capire" il codice e non eseguirà lo scripting lato server trasmettendo lo stesso in formato testuale.

# Sintassi fondamentale

Quando il PHP inizia a esaminare un file, cerca i tag di apertura e di chiusura, che indicano dove iniziare e terminare l'interpretazione del codice. Questa tecnica permette al PHP di essere incorporato in tutte le tipologie di documenti, poichè ogni cosa esterna ai tag di apertura e di chiusura viene ignorata dall'analizzatore di PHP. Il più delle volte si vedrà codice PHP racchiuso in documenti HTML, come nel seguente esempio.

```
<p>Questo sarà ignorato.</p>
<?php echo 'Questo, invece, sarà analizzato.'; ?>
<p>Anche questo sarà ignorato.</p>
```

Si possono usare anche strutture più avanzate, iniziando ad imparare che le variabili in PHP sono autotipizzanti (capiscono il tipo dal valore loro fornito e possono anche cambiar tipo di valore accolto durante l'elaborazione) e precedute dal simbolo del dollaro:

```
<?php
if ($espressione=="si") {
    ?>
    <strong>Questo  è  vero.</strong>
    <?php
} else {
    ?>
    <strong>Questo è falso.</strong>
    <?php
}
?>
```

Questo codice funziona come atteso, perchè quando il PHP trova il tag di chiusura ?>, inizia a visualizzare tutto ciò che incontra sino a quando non si raggiunge un'altro tag di apertura. L'esempio è semplificato, ovviamente, ma nella visualizzazzione di grossi blocchi di testo uscire dalla modalità di parsing del codice PHP è generalmente più efficiente che inviare il tutto tramite **echo** o **print()** (le due funzioni predefinite di stampa esterna al php), cosa che si potrebbe fare per evitare aperture e chiusure di php; cosa vivamente sconsigliata, anche se possibile, poiché estremamente confusionaria.

Come in C od in Perl, il PHP richiede che le istruzioni siano chiuse dal punto e virgola al termine di ogni istruzione. I tag di chiusura di un blocco di codice PHP implicano in automatico il punto e virgola; non occorre, pertanto, inserire il punto e virgola per chiudere l'ultima riga di un blocco PHP. Il tag di chiusura del blocco include il newline (a capo) immediatamente seguente, se presente.

Il PHP supporta i commenti dei linguaggi 'C', 'C++' e stile shell (stile Perl) di Unix. Per esempio:

```
<?php
    echo 'Questo &grave; un test'; // Questo è un commento su linea stile c++
    /* Questo è un commento su più linee
       ed infatti questa è ancora un'altra linea di commento */
    echo 'Questo &egrave; un altro test';
    echo 'Un ultimo test'; # Questo &egrave; un commento stile shell Unix
?>
```

Le strutture di controllo e le funzioni possono essere implementate all'interno del PHP come nel C o C++, esempi di if, while, for, switch  e funzioni con passaggio di parametri. Si invita a provare gli script facendo copia-incolla sul proprio editor e quindi navigare la pagina da browser e vedere qual è il codice sorgente trasmesso in HTML al client.

Per la piena comprensione dei precedenti esempi, ove non fosse chiaro il funzionamento e la logica sottesa, si rimanda alla conoscenze maturate negli studi del terzo e quarto anno.

Nota: *si ponga attenzione nel fare copia-incolla ad eventuali spazi codifica in maniera "strana" dall'editor di testo, che potrebbero dare errori o funzionamenti inattesi.*

```php
<?php
$a=4;//si invita cambiare valore di volta in volta
if ($a == 5){
    echo  "a è uguale a 5";
    echo  "...";
}
elseif ($a == 6){
    echo  "a è uguale a 6";
    echo  "!!!";
}
else echo  "a non è nè 5 nè 6";
?>
```

*(I due esempi seguenti sono identici: entrambi visualizzano i numeri da 1 a 10 utilizzando strutture di controllo diverse)*

```php
<?php
/* esempio 1 */

$i = 1;
while ($i <= 10) {
    echo $i++;  /* Il valore visualizzato è il valore della
                   variabile $i prima dell'incremento
                   (post-incremento) */
}
?>
```

```php
<?php
/* esempio 2 */

for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
?>
```

*(Si sottolinea come nell'esempio seguente sia importante il "break" per uscire dalla multicondizionalità dell'espressione switch ed eseguire UN SOLO caso, poiché in assenza di questo ed essendo posta a "true" la condizione del case saranno fatti anche tutti i precedenti, azione che in tale caso non è opportuna)*

```php
$i=1;//si invita a cambiare valore di volta in volta
switch ($i) {
    case 1:
        echo "i è uguale a 1";
        break;
    case 2:
        echo "i è uguale a 2";
```

```
        break;
    case 3:
        echo "i è uguale a 3";
        break;
    default:
        echo "i non è fra i primi tre numeri positivi";
        break;
}
?>
```

```
<?php
function quadrato_decimale($num,$tipo="i")
{
 if ($tipo==”d”)//ritorno del quadrato decimale
  return (float)$num*$num;
 else //ritorno del quadrato in formati intero (azione di default)
  return (int)$num*$num;
}
echo quadrato_decimale(4);//Output 16
echo quadrato_decimale(2,”d”);//Output 4.0
?>
```

Per quel che riguarda i parametri formali delle funzioni, anch’essi sono autotipizzati dalla chiamata dei parametri attuali, a meno di casting espliciti come quello appositamente utilizzato nei due ritorni della funzione per ricordare tale possibilità. E’ possibile anche utilizzare dei parametri di default che interverranno con valori predefiniti se non fosse dichiarato il parametro attuale.

I vettori mono o multidimensionali in PHP non hanno alcun bisogno di dichiarazione o inizializzazione, nascono già “pronti all’uso” grazie alla già citata autotipizzazione oppure si autotipizzano al momento dell’inserimento del valore. L’indice è di default un numero intero, ma si può utilizzare anche un carattere oppure si possono generare array associativi con indici stringa, cosa che avverrà implicitamente nella successiva spiegazione della trasmissione valori da form con i metodi get/post.

```
<?php
for ($i=0;$i<5;$i++){
 $vett[$i]=$i*2;//vettore di 5 interi contenente il doppio dell’indice
}
$vett2=[];
$vett2[3][4]="ciao";//le precedenti 19 caselle sono inizializzate a NULL

$vett3=[8,4.0,6.26,’a’];//ogni casella del vettore ha tipi diversi coesistenti

$parola="Benvenuti";//la stringa è un vettore di char

echo $vett[3];//Output 6
echo $vett2[2][5];//Output un carattere nullo
echo $vett3[0];//Output 8
echo $vett3[3];//Output a
echo $parola[6];//Output u
?>
```

La gestione dei file in php è estremamente simile a quella in C ed è quindi molto semplice; poiché tale gestione ha spesso a che fare con la shell del sistema operativo si cita nei seguenti esempi la funzione “system()”, in grado di passare il parametro alla shell di sistema per operatività diretta sul server; ragion per cui si raccomanda di usarla con attenzione.

```
<?php
$interrogati=["tizio","caio","sempronio","pinco pallino"];
shuffle($interrogati);//mix casuale delle posizione del vettore
system("cd /tmp");//posizionamento su directory obiettivo

//scrittura della sequenza casuale su file di testo
$fp = fopen('sequenza.txt', 'w');//apertura file e puntatore a file
for($i=0;$i<4;$i++)
 fwrite($fp,$interrogati[$i]."\n");
fclose($fp);//chiusura handle e marcatura dell’EOF
?>
```

Il file di testo generato nella directory tmp sulla root del sistema conterrà al suo interno la sequenza casuale degli interrogati.

E’ possibile aprire il file in scrittura (parametro “w”) e quindi cancellare totalmente il contenuto all’interno degli stessi prima di scrivere oppure in continuità (parametro “a”) che riprenderà dal precedente End Of File senza cancellare il contenuto già presente. Esistono possibilità “plus” di apertura in scrittura da inizio file troncato e lettura (w+) oppure in scrittura da EOF e lettura (a+).

L’accesso ai file è sequenziale ed avviene, dopo l’apertura e la restituzione del puntatore del file, tramite la funzione fread(); la lettura avviene dall’inizio del file stesso o tramite spostamenti di un certo numero di byte (tramite la funzione fseek). Segue un semplice esempio.

```
<?php
system("cd /tmp");//posizionamento su directory obiettivo
//lettura del file con accesso sequenziale
$fp = fopen('sequenza.txt', 'r');//apertura file e puntatore a file
fseek($fp, 5); //posizionamento al quinto carattere
$test = fread($fp, 10);//lettura di 10 caratteri partendo dal quinto
echo $test;
fclose($fp);//chiusura dell’handle del file e reset del puntatore
?>
```

E’ possibile aprire un file in lettura e scrittura (r+) oppure in lettura e scrittura con reset del puntatore e del contenuto(w+) o anche in scrittura dalla fine del file e lettura (a+).

Altre funzioni utili per l’interazione con i file e per le quali si consiglia un approfondimento sul manuale php sono fgets (per lettura di singole stringhe), feof (da usare in combinazione con while per lettura di file sino al termine), fscanf (lettura con specificatore di formato), explode (per passare su vettore una lettura da file con specifici separatori) e count (per sapere quanti elementi contiene in totale un vettore), filesize (per conoscere le dimensioni di un file).

Non si confonda la gestione dei files con la variabile di tipo $_FILES, che è generata da un form OBBLIGATORIAMENTE di tipo POST con l’attributo enctype="multipart/form-data" che consente la trasmissione tramite body del messaggio post http di uno streaming di bit utilizzato per upload sul server di un file.

Nell'esempio seguente si mostra come sia possibile con un input di tipo "file" generare la variabile suddetta che conterrà un file selezionato tramite apposito dialog-box del sistema operativo sul client per il trasporto su server.

```
  <form method="post" enctype="multipart/form-data">
   Seleziona il file da caricare
   <input type="file" name="mioupload"><br/>
   <input type="submit" name="carica" value="Carica un file">
  </form>
<?php
if ($_POST[carica]=="Carica un file")//esecuzione script successiva a submit
{//Output di alcune caratteristiche del file caricato
 echo "Nome file: ".$_FILES[mioupload][name]."<br/>";
 echo "Dimensione file: ".$_FILES[mioupload][size]."<br/>";
 echo "Tipo file: ".$_FILES[mioupload][type]."<br/>";
 if ($_FILES[mioupload][type]=="image/jpeg")
  {//funzionalità specifiche di un file di tipo immagine
   $dimensioni=getimagesize($_FILES[mioupload][tmp_name]);
   echo "Larghezza foto ".$dimensioni[0]."<br/>";
   echo "Altezza foto ".$dimensioni[1]."<br/>";
  }
 echo "Nome momentaneo su server: ".$_FILES[mioupload][tmp_name]."<br/>";
 //trasferimento del file dall'area temporanea alla cartella di destinazione
 $trovapunto=strrpos($_FILES[mioupload][name],".");//posizione punto finale
 $estensione=substr($_FILES[mioupload][name],$trovapunto);//stringa post-punto
 $nome_file=time().$estensione;//nome file in base al momento di upload
 //spostamento da locazione temporanea a definitiva
 move_uploaded_file($_FILES[mioupload][tmp_name],"./caricamenti/".$nome_file);
}
?>
```

Si noti come il vettore associativo stavolta abbia due livelli di indicizzazione, uno riguardante il nome della variabile e l'altro riguardanti caratteristiche specifiche della variabile citata, come nome del file presso il client (name), dimensione in byte (size), tipo del file rilevato dall'intestazione dello stesso secondo gli standard noti (type) e nome che viene temporaneamente assegnato al file durante il trasporto sull'area momentanea del server (directory /tmp per i server linux).

Il trasporto dall'area temporanea alla cartella di destinazione, presso la quale si prevede di aver settata opportunamente i permessi di scrittura per l'utente www-data, avviene tramite il comando move_uploaded_file che non effettua l'upload, già effettuato dal momento del submit tramite metodo post, ma uno spostamento su server dall'area momentanea di immagazzinamento a quella definitiva; durante tale spostamento avviene anche la rinomina del file con il suo nome definitivo su server, in merito al quale si consiglia sempre una politica che possa evitare sovrascritture e smarrimenti.

**Nota importante**: *PHP, come java, C e C++ è "case sensitive", quindi attenzione alla differenza fra maiuscole e minuscole, e ricordate che il PHP è un linguaggio molto versatile e potente ma* ***"grandi poteri comportano grandi responsabilità"****, quindi è sempre bene farne uso ordinato e disciplinato per evitare problemi in fase di parsing e soprattutto di runtime. Non ci vuole molto a bloccare un server con operazioni di elaborazione poco corrette!*

# Come far interagire le variabili con l'HTML

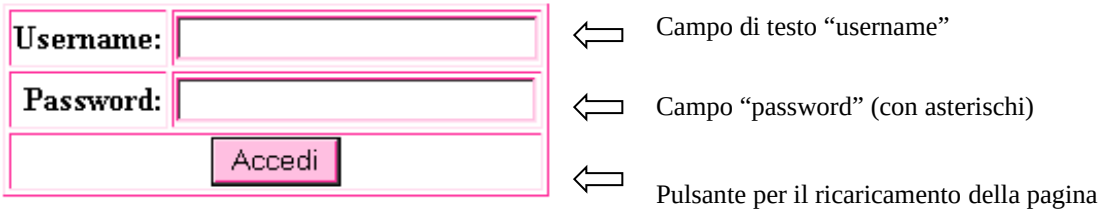
Vista la gestione delle variabili nella propria essenza e nella logica delle strutture di controllo, vediamo come si fa a dare valori in ingresso alle stesse… quel che nel C o C++ è delegato rispettivamente alle funzioni "scanf" e "cin".

Partendo dal concetto che HTML NON HA STATO, e che quindi non può gestire in memoria variabili, deve essere chiaro che non ci sono variabili html ma soltanto php e che esse "durano" lo spazio vitale di una pagina (dall'apertura alla chiusura della stessa). Allora come si fa a "passare" una variabile da una pagina all'altra o, ancor di più, come si fa a passare un valore fra una navigazione di una pagina e la successiva navigazione per aggiornamento della stessa?

La risposta sta nel tag "form" dell'html che è capace, attraverso appositi oggetti "input" al suo interno, di trasmettere variabili nel ricaricamento della pagina stessa (metodo POST) e nella possibilità di passare valori alle variabili direttamente nel chiamare la pagina tramite il proprio indirizzo (metodo GET). Se la action del form rimane vuota la pagina richiama se stessa, altrimenti la "spedizione" dei valori avverrà verso la pagina dichiarata nella action; professionalmente si preferisce utilizzare una sola pagina per favorire sviluppo e manutenzione del software.

***Esempio di metodo POST:***

Il seguente codice realizza un controllo d'ingresso user/password sfruttando il "passaggio" delle variabili da un caricamento della pagina all'altro; tutta la spiegazione del codice è nei commenti dello stesso mentre immediatamente seguente a questo preambolo vediamo la grafica della pagina "login.php".




```
<html>

<head> <title> Pagina per l'autenticazione </title></head>

<body>

<?php

if (isset($_POST['username']))  //controlla se il campo "user" della pagina è stato compilato e SOLO in tal caso agisce

{

 $username=$_POST['username']; //conservazione in una variabile momentanea del valore passato da username

 $password=$_POST['password']; //conservazione in una variabile momentanea del valore passato da password

 if ($username==”prova” and $password==”test”) // verifica dei valori immessi
```

```
 header("Location: benvenuto.php"); //funzione di reindirizzamento alla pagina di apertura del sito
else //azione da eseguire se i valori immessi non corrispondono a quelli desiderati
   header("Location: login_fallito.php"); //funzione di reindirizzamento alla pagina che comunica l'errore nella user o password
}?>
<center>
<form action="" method="POST" name="formLogin" id="formLogin">
  <table border="1" align="center">
    <tr>
      <td><div align="right">Username:</div></td>
      <td><div align="center">
          <input name="username" type="text" id="username">
        </div></td>
    </tr>
    <tr>
      <td><div align="right">Password:</div></td
      <td><div align="center">
          <input name="password" type="password" id="password">
        </div></td>
    </tr>
    <tr>
      <td colspan="2" align="center"><div align="center">
          <input type="submit" name="Submit" value="Accedi">
        </div></td>
    </tr>
  </table>
</form>
</center>
</body>
</html>
```

Come è possibile vedere dal semplice esempio proposto i valori delle variabili entrano tramite i campi di testo, fra i quali si sottolinea che il campo “password” non sarà visibili ma sarà “asteriscato” poiché è stato dato nell’html il valore password e non text per criptarlo alla scrittura (ragioni di sicurezza).

Tutto ciò che è dentro il form ed ha un nome ed un id viene passato al ricaricamento della pagina e viene incanalato nelle variabili precedute dalla scritta $_POST, queste conterranno i valori immessi negli appositi campi e quindi saranno trattabili all’interno dell’html sino al prossimo ricaricamento della pagina.

***Esempio di metodo GET:***

Un’altra maniera per passare valori è quella data direttamente dalla possibilità di scrivere il valore della variabile nel richiamare la pagina con il proprio indirizzo.

Immettiamo nella barra degli indirizzi del browser il seguente URL:

www.miosito.it/php/login.php?username=prova&password=test

Si noterà che dopo l’indirizzo esatto della pagina (www.miliziano.it/php/prova.php) sono stati aggiunti un punto interrogativo dei nomi e degli altri simboli; tutto ciò che segue il punto interrogativo è da passare al metodo GET, ciò che è a sinistra dell’uguale è il nome della variabile mentre ciò che è a destra è il valore della variabile stessa; ogni variabile, dato che è possibile passarne più d’una, è separata dalla successiva con il carattere &.

Riadattando il precedente script del metodo POST (senza più avere, però, il form e quindi la grafica) si ha che lo script in grado di interpretare la nostra stringa URL ed eventualmente smistarci alla pagina corretta è il seguente:

```
<?php

if (isset($_GET['username']))  //controlla se il campo “user” della stringa URL passata è stato compilato e SOLO in tal caso agisce

{

 $username=$_GET['username']; //conservazione in una variabile momentanea del valore passato da username nell’URL

 $password=$_GET['password']; //conservazione in una variabile momentanea del valore passato da password nell’URL

if ($username==”prova” e $password==”test”) // verifica dei valori immessi

 header("Location: benvenuto.php"); //funzione di reindirizzamento alla pagina di apertura del sito

else //azione da eseguire se i valori immessi non corrispondono a quelli desiderati

   header("Location: login_fallito.php"); //funzione di reindirizzamento alla pagina che comunica l’errore nella user o password

?>
```

Come visibile non c’è bisogno, <u>in questo specifico caso</u>, di html a supporto perché il php opererà prima dell’apertura della pagina stessa in base alle variabili passate dalla stringa URL! Per tale specifico obiettivo si sconsiglia vivamente l’uso di un semplice GET poiché le variabili sono in chiaro nell’URL!!!

***Autorizzazione all'utilizzo di una pagina:***

Spesso capita di dover proteggere una pagina (o una parte di essa) dall'utilizzo non autorizzato; in tal caso si deve far ricorso a qualcosa che possa essere riconosciuto in tutte le pagine finchè non si esce dal sito… una specie di "pass"; per questo scopo sono state create le "variabili di sessione", particolarissime variabili che vengono riconosciute in tutte le pagine di un sito finchè questo non si abbandona e che non perdono valore nel ricaricamento o nel cambiamento di pagina all'interno del sito stesso, come tutte le altre perché salvate dal server sul client, tramite il browser, in appositi cookies, file di testo che contengono valori con cui ogni pagina si può interfacciare.

Per attivare la lettura/scrittura dei cookies si deve dichiarare prima di ogni cosa la funzione session_start(), mentre se si vuole cancellare la memoria dei cookie e "chiudere la sessione" si deve attivare la session_destroy(); successivamente alla session_start();

Queste variabili vengono precedute dalla dicitura $_SESSION ed il loro nome è messo fra parentesi quadre. Rifacendoci all'esempio del metodo "POST" possiamo modificare il riconoscimento positivo della user/password con il seguente stralcio:

```
……..

$username=$_POST['username']; //conservazione in una variabile momentanea del valore passato da username

 $password=$_POST['password']; //conservazione in una variabile momentanea del valore passato da password

if ($username==”prova” && $password==”test”) // verifica dei valori immessi

{

 $_SESSION[‘autorizzato’]=true; //Assegnazione al valore “vero” della variabile di sessione “autorizzato”

 header("Location: benvenuto.php"); //funzione di reindirizzamento alla pagina di apertura del sito

}

else //azione da eseguire se i valori immessi non corrispondono a quelli desiderati

{

……..
```

In questa maniera in tutte le pagine la variabile di sessione (booleana per autotipizzazione) varrà true (vero) e quindi possiamo "incapsulare" la nostra pagina in maniera che sia visibile solo se questa variabile è vera!

Esempio nella pagina "benvenuto.php" cui si tenta lo smistamento dopo il login user/pw:

```
<html>

 <head> <title> Pagina di benvenuto </title></head>

 <body>

  <div align=”center”>

   <?php
```

```
   if ($_SESSION['autorizzato'])  //controlla se la variabile di sessione è posta al valore desiderato

   {?>

    <b>Benvenuto nel sito!</b>

   <?php

   }

  else

   {?>

    INGRESSO NON AUTORIZZATO!<br>

    EFFETTUARE IL LOGIN cliccando <a href=”login.php”>qui</a>

   }?>

  </div>

 </body>

</html>
```

Come è possibile vedere tutto ciò che è compreso fra le parentesi graffe (quindi il codice html) viene eseguito soltanto in presenza di una delle due possibilità: riconoscimento o no dell’utente in base al valore passato dalla variabile di sessione, che viene “marcata” nella pagina di login e tale resterà sino alla fine della sessione (disconnessione dal sito); una variabile di sessione non inizializzata ad alcun valore è nulla, quindi non è possibile che alla fine della sessione si conservino valori errati in essa.

Quindi si otterrà una pagina dal titolo “pagina di benvenuto” e, se si è stati riconosciuti come autorizzati alla navigazione, al centro della pagina ci sarà la scritta (in grassetto grazie al tag <b>):

Benvenuto nel sito!

Altrimenti si troverà la scritta:

INGRESSO NON AUTORIZZATO!

EFFETTUARE IL LOGIN cliccando qui

Dove il clic su “qui” sarà un link che porterà alla pagina di login per permettere l’identificazione.

Per quel che riguarda approfondimenti nell’uso dei vari campi di input (tipo password, radio, checkbox, select, textarea) si rimanda ai corsi di HTML e/o javascript degli anni precedenti. Di contro, si approfondisce di seguito l’uso di particolari input che non accolgono semplici valori ma sequenze di bit appartenenti a file.

Tali input sono di tipo “file” ed hanno OBBLIGATORIAMENTE bisogno di particolari form POST detti “multipart”, la cui dichiarazione è:

*<form method="post" enctype="multipart/form-data">*

Tali form caricheranno la sequenza di bit contenente qualsiasi tipo di file in appositi “stream” di bit che si potranno manipolare opportunamente per estrarre dati o per trasferire il file sul server o come attach di una mail da form.

***Esempio di caricamento di file sul server***

Si premette che la directory che accoglierà i file dovrà avere gli appositi permessi di scrittura, per quel che riguarda server seri, ovvero linux, i permessi di scrittura dovranno essere dati al gruppo “altri” poiché l’utente che scrive sulla directory è l’utente apache, normalmente diverso dall’utente proprietario della /var/www

```
<html>
 <head>
  <title>Prova caricamento file su server</title>
 </head>
 <body>
  <form method="post" enctype="multipart/form-data">
   <input type="file" name="allegato">
   <input type="submit" name="upload" value="Carica file">
</form>
<?php
$tutto_ok=false; //flag di controllo del caricamento
if(isset($_POST[‘upload’]))
 {
  if((strlen($_FILES['allegato']['name'])>=3) and ($_FILES['allegato']['size']<1048576))
  {//controlla che il file abbia almeno nome ed estensione minore di 1MB
   $pos_punto_finale=strrpos($_FILES['allegato']['name'], "."); //posizione punto
   $estensione=substr($_FILES['allegato']['name'],$pos_punto_finale+1); //estrazione sottostringa con tipo
   //spostamento da posizione temporanea a posizione definitiva sul server e rinomina del file con marcatura data
   move_uploaded_file($_FILES['allegato']['tmp_name'],"../files/allegato_".date(“dmyhis”)."."$estensione);
   if($_FILES['allegato']['error']==0) $tutto_ok=true;
  }
```

```
if($tutto_ok)

 {//caricamento riuscito

  echo”<br/><br/>caricamento del file avvenuto con successo!!”;

 }

else

 {//caricamento non riuscito

  echo”<br/><br/>Problemi nel caricamento del file, riprova!”;

 }

}?>
```

E’ ora possibile verificare che nella directory designata sia presente il file caricato in upload con il nuovo nome; si sottolinea come il proprietario del file dovrebbe essere differente da quello della directory su server linux, ma il file è comunque accessibile al proprietario della directory.

Si lascia come approfondimento ai discenti la gestione dei parametri del file in caso di immagini, come ad esempio il tipo MIME o le dimensioni di lunghezza e larghezza.

# PHP e SQL

Il php è spesso utilizzato per interfacciare il web con i database, specialmente “vincente” è l’accoppiata php-mysql che ormai trova vastissimi utilizzi privati ed aziendali per efficienza, velocità, leggerezza e costi nel reperire o modificare informazioni via web la cui struttura è stata creata tramite classiche istruzioni sql.

Ma come si interfaccia php con mysql?

Fondamentalmente si deve “connettere” il server php con il server mysql (entità distinte e separate) e di ciò si occupa la funzione php *mysql_connect*; questa passa dei paramentri la web-server che interfaccerà i due programmi facendoli lavorare in simbiosi con incredibili risultati.

Vediamo varie tecniche per far cio, partendo da quella più antiquata (php 5 version).

Per ottenere questo si dovranno passare nell’ordine alla funzione l’indirizzo del server che ospita il database (quasi sempre lo stesso che ospita il web-server con php, quindi il valore è quasi sempre “localhost”), il nome dell’utente autorizzato ad entrare nel server mysql e la password assegnatagli per riconoscerlo. Quindi se ad esempio dovessimo entrare mettere in contatto un server mysql che funziona sul computer che ospita il sito del della scuola e sullo stesso sito abbiamo la possibilità di interpretare pagine php, ed inoltre la combinazione user/pw fosse “ciccio”/”luca” è la seguente funzione che permette di interfacciare i due server e di avere nella variabile di ritorno che chiameremo “contatto” il valore che tiene uniti i due server ed al quale dobbiamo fare riferimento quando usiamo l’interfacciamento: $contatto=mysql_connect (“localhost”,“ciccio”,“luca”);

Una volta che ci si è messi correttamente “in contatto” con il server mysql bisogna selezionare il database che si vuole utilizzare con la funzione *mysql_select_db*; questa funzione ha bisogno di aver passati come parametri i valori del “punto di contatto” fra i server (restituito da mysql_connect) ed il nome del database. Quindi se in precedenza avevo creato in mysql il database “scuola” con l’istruzione SQL “create database scuola”) ed ora voglio connettermi via web con php a questo database devo utilizzare la seguente sintassi: mysql_select_db(“scuola”,$contatto);

Una volta che mi sono messo “in comunicazione” con il mio database posso utilizzare le query per estrarre, inserire, aggiornare e cancellare dati dal mio database, tramite la funzione *mysql_query* cui si dovrà passare come parametro la query da eseguire. Quindi per poter eseguire la query che estrae, ad esempio, il nome ed il cognome di tutti gli alunni maggiorenni dovrà utilizzare il seguente comando che pone i risultati della query nella variabile “ricerca” di cui si discuterà nel prossimo paragrafo: $ricerca=mysql_query(“SELECT nome,cognome FROM alunni WHERE eta>17”);

Una volta eseguita la query i dati estratti sono contenuti in un oggetto “astratto” di dimensioni non note a priori, cui si può accedere in diverse maniere per estrarre le informazioni richieste; uno dei metodi più semplici è l’utilizzo della funzione *mysql_fetch_row* che vuole come parametro i dati estratti dalla query (quindi, nel nostro caso, la variabile “ricerca”) e che darà una riga (in inglese, appunto, row) alla volta per ogni chiamata della funzione, in formato vettoriale, con la seguente sintassi che assegna il vettore alla variabile “dato”: $dato= mysql_fetch_row($ricerca); il vettore è indicizzato in maniera classica (numerica), se lo si volesse indicizzare in maniera associativa tramite i nomi dei campi mysql si dovrà similmente usare $dato= mysql_fetch_array($ricerca);

Esempio chiarificatore con la sequenza esatta delle azioni e commento a margine:

**$contatto=mysql_connect (“localhost”,“ciccio”,“luca”);** //interfacciamento con il database

**mysql_select_db(“scuola”,$contatto);** //selezione del database “scuola” sul server mysql

Dati in tabella “alunni”

| ID | Nome | Cognome | Eta |
|---|---|---|---|
| 1 | Claudio | Rossi | 19 |
| 2 | Andrea | Neri | 15 |
| 3 | Miriam | Verdi | 17 |
| 4 | Giusto | Bianchi | 18 |

**$ricerca=mysql_query(“SELECT nome,cognome FROM alunni WHERE eta>17”);** //query

Dati estratti e presenti nella “tabella astratta” cui ci riferiamo con la variabile “ricerca”

| Nome | Cognome |
|---|---|
| Claudio | Rossi |
| Giusto | Bianchi |

**$dato= mysql_fetch_row($ricerca);** // lettura della prima riga della “tabella astratta”

**echo $dato[0];** //stampa su html del valore “Claudio”

**echo $dato[1];** //stampa su html del valore “Rossi”

**$dato= mysql_fetch_row($ricerca);** // lettura della successiva riga (2ª) della “tabella astratta”

**echo $dato[1];** //stampa su html del valore “Bianchi”

**echo $dato[0];** //stampa su html del valore “Giusto”

Con il vettore associativo l’esempio sarebbe, invece:

**$dato= mysql_fetch_array($ricerca);** // lettura della prima riga della “tabella astratta”

**echo $dato[“nome”];** //stampa su html del valore “Claudio”

**echo $dato[“cognome”];** //stampa su html del valore “Rossi”

etc...

Come visibile dall’esempio per ogni chiamata della funzione mysql_fetch_row (fetch_array) si legge una riga e finchè non viene richiamata la funzione non se ne leggono altre! I valori sono dati in formato vettoriale (a ciò è dovuta la parentesi quadra dopo il nome della variabile prescelta per ospitare i dati estratti) , in maniera che il primo valore in uscita dalla query ha la posizione 0 mentre i successivi hanno posizioni 1,2,3,4,5,etc. dipendentemente dal numero di colonne selezionate dalla query (nel nostro caso soltanto due, quindi le posizioni 0 e 1).

Per ovviare ad infinite ed indefinite chiamate di mysql_fetch_row si dovrà utilizzare un ciclo, preferibilmente un ciclo while con un costrutto in grado di effettuare estrazioni finché l’oggetto

trasmesso dal mysql_query contiene dati. Si da di seguito l'esempio di estrazione dati in ciclo, senza codice di stampa che si lascia come esercizio.

**while ($dato=mysql_fetch_row($ricerca)) {…}**

Si ponga l'attenzione sul fatto che l'uguale all'interno della parentesi non è un paragone ma un'assegnazione, quindi grazie alle potenzialità del linguaggio PHP il ciclo avverrà FINCHE' E' VERA L'ASSEGNAZIONE su dato dei valori provenienti da mysql_fetch_row e quindi finché ci sono dati da estrarre!

Nota importante: query di inserimento, aggiornamento e cancellazione non hanno bisogno di cicli del *mysql_fetch_row* perché agiscono su un solo valore e quindi direttamente dopo la chiamata della funzione *mysql_query*! Addirittura in questo caso si potrebbe fare un *$dato=mysql_fetch_row(mysql_query("SELECT …"))*.

Tutti i comandi che sono stati descritti riguardano il php in versione 5, linguaggio ancor più che valido nonostante siano presente versioni successive (7 ed 8 su tutte). Per evitare di incappare in problemi di compatibilità fra le varie versioni, spesso contenenti novità proprio per l'interfacciamento con mysql, è possibile usare mysqli, un driver che utilizza l'RDBMS mysql più o meno come un framework e che è presente in maniera identica nella versione 5 e nella versione 7 di php.

Le funzionalità di mysqli sono molto simili a quelle di mysql, varia soltanto qualche parametro, come si vedrà nelle seguenti righe.

Per connettersi mysqli ingloba l'interfacciamento al server e la selezione del database in *mysqli_connect()* una funzione che ha bisogno degli stessi parametri della mysql_connect con l'aggiunta in coda del parametro del database da selezionare, in modo da restituire il puntatore alla connessione completo del database da utilizzare e poter così "switchare" velocemente da un database ad un altro senza bisogno di rifare la connessione, vero e proprio limite della mysql_connect.

**$contatto2=mysqli_connect ("localhost","ciccio","luca","scuola");** //ripetizione della precedente

Varia lievemente anche il passaggio della query al server mysql, perché si deve dichiarare non solo il codice mysql ma anche il puntatore a connessione per il database sul quale si lancia la query.

**$ricerca2=mysqli_query($contatto2, "SELECT nome,cognome FROM alunni WHERE eta>17");**//esempio già visto prima ma in modalità mysqli

Molto più versatile è l'estrazione dei dati dall'oggetto restituito dalla query, come visibile nelle successive righe, infatti specificando in mysqli_fetch_array la modalità si può avere un vettore con indicizzazione classica o associativa senza variare la funzione di estrazione.

**$dato2= mysqli_fetch_array($ricerca2,MYSQLI_NUM);** // lettura della prima riga

**echo $dato2[0];** //stampa su html del valore "Claudio"

**echo $dato2[1];** //stampa su html del valore "Rossi"

**$dato2= mysqli_fetch_array($ricerca2,MYSQLI_ASSOC);** // lettura della successiva riga

**echo $dato2[nome];** //stampa su html del valore “Giusto”

**echo $dato2[cognome];** //stampa su html del valore “Bianchi”

Ancor più performante è l’utilizzo (consigliatissimo) di mysqli ad oggetti come visibile dallo stralcio di codice esposto di seguito:

```
//creazione dell’oggetto connessione

$p = new mysqli('localhost', 'username', 'password', 'nome_database');

if ($p->connect_error)
  {//mostra l’errore eventuale di connessione
   die('Errore di connessione ('.$p->connect_errno.')'.$p->connect_error);
  }
else echo 'Connesso. ' . $p->host_info . "\n";//parte opzionale
$q = $p->query("SELECT nome,cognome FROM utenti WHERE eta>17");
if($q->num_rows==0)
 echo "nessun risultato";//query “vuota”
else
 {//mostra i risultati
  echo "Elenco clienti maggiorenni<br/>";
  while($dato = $q->fetch_array(MYSQLI_NUM))
   {//ciclo di stampa  della query
    echo "<br/>nome: ".$dato[0]."<br/>";
    echo "cognome: ".$dato[1]."<br/>";
   }
 }
$p->close(); //opzionale ma utile
```

Se invece di un vettore ad indicizzazione numerica si vuol usufruire di un vettore associativo nella restituzione dei dati si dovranno sostituire le righe del ciclo di stampa con le le seguenti:

```
  while($dato = $q->fetch_array(MYSQLI_ASSOC)
   {//ciclo di stampa  della query
    echo "<br/>nome: ".$dato[nome]."<br/>";
    echo "cognome: ".$dato[cognome]."<br/>";

    }
```

Si lascia come esercizio la sostituzione in tutti gli esempi seguenti e precedenti della sintassi php 5 con la sintassi mysqli ad oggetti.

# Esempi classici di pagine PHP

Vengono di seguito elencati quattro esempi classici e frequentissimi di programmazione php con interazione php-mysql; questi vengo sempre richiesti negli ultimi punti dei compiti in classe nell'ultima parte dell'anno scolastico e nei compiti per l'esame di Stato.

Si vedrà come sia possibile autorizzare all'utilizzo di una pagina proteggendola da navigazioni non autorizzate, come autenticare e registrare un utente tramite user/password, come far stampare a schermo in una pagina web un elenco di valori tratti da una query ed infine come mandare una mail tramite script php direttamente da una pagina web e non tramite un client di posta elettronica.

***Inserimento dati, riconoscimento e registrazione utente:***

Viene spesso utilizzata la possibilità di autenticarsi tramite dati presenti in database; ecco un esempio di una pagina con i campi per farsi riconoscere tramite un ipotetico database residente su server mysql sulla stessa macchina dove è presente il server apache; nel server mysql su da per presente il database "scuola" con le tabelle "utenti" e "logregister", fra le altre, l'una nella quale sono presenti tutti gli utenti autorizzati ad accedere al sito, l'altra per registrare chi entra ed in che orario. Troveremo in quest'esempio le variabili poste all'interno di una query (sono sempre dei valori che possono essere utilizzati in qualsiasi maniera… anche in una stringa che è una query!), una query di selezione semplice ed una query di inserimento, oltre ad una funzione di conversione di timestamp (data e orario correnti) in stringa letterale "guidata" per l'orario; infine vedremo come all'interno di un controllo di tipo "if" abbiamo un'assegnazione, infatti se l'assegnazione va a buon fine e quindi esistono i valori da assegnare allora avremo l'ok al controllo, altrimenti se non esistono valori da assegnare il controllo sarà negativo e non verrà eseguito ciò che è sotto "if" ma ciò che è sotto "else":

```
<?php

$contatto=mysql_connect ("localhost","ciccio","luca"); //interfacciamento con il database

mysql_select_db("scuola",$contatto); //selezione del database "scuola" sul server mysql

//Login al database con registrazione dell'ingresso

if (isset($_POST['username'])) //esegue le query solo se viene inserito almeno lo username

{

 $username=($_POST['username']);

 $password=($_POST['password']);

// segue query con controllo dei campi "userID" e "password" della tabella "utenti"

$confutazione=mysql_query("SELECT nome, livelloAccesso FROM utenti WHERE UserID='$username' AND Password='$password'");

 if($dato = mysql_fetch_row($confutazione)) //se è stato trovato almeno un dato agisce e lo seleziona

  {

   $_SESSION['nomeIndex'] = $riga[0]; //conserva il nome dell'utente
```

```
  $_SESSION['livelloAccesso'] = $riga[1]; //conserva il livello d’accesso
  $_SESSION['autorizzato'] = true; //pone al valore vero la variabile di sessione “autorizzato”
  $Ingresso=date('y-m-d H:i:s'); //particolare funzione che passa a stringa l’orario corrente sul server
 // segue query d’inserimento dato nel registro degli ingressi, tabella “logregister”
  $registrazione=mysql_query("INSERT INTO logregister (Entrata,UserID) VALUES ('$Ingresso','$riga[0]')");
 }

  switch ($riga[1]) //smistamento nel corretto ingresso a seconda del livello
   {
    case 'allievo': header("Location: indexUtente.php"); break;
    case 'professore': header("Location: indexOspite.php"); break;
    case 'amministratore': header("Location: indexAdmin.php"); break;
   }
 }
 else //smistamento a pagina di avvertimento errati dati di autenticazione
  header("Location: login_fallito.php");
}
<center>
<form action=”” method="POST" name="formLogin" id="formLogin">
  <table border="1" align="center">
    <tr>
      <td><div align="right">Username:</div></td>
      <td><div align="center">
          <input name="username" type="text" id="username">
        </div></td>
    </tr>
    <tr>
      <td><div align="right">Password:</div></td>
      <td><div align="center">
```

```
        <input name="password" type="password" id="password">
      </div></td>
    </tr>
    <tr>
     <td colspan="2" align="center"><div align="center">
        <input type="submit" name="Submit" value="Accedi">
      </div>
     </td>
    </tr>
  </table>
</form>
</center>
</body>
</html>
```

Come si può intuire le pagine possono, ora, essere protette per la visita sia in base al “pass” della variabile “autorizzato” sia in base al livello di accesso; per esempio alcune pagine possono essere viste da alcuni utenti (quali gli amministratori) e non da altri, anche se entrambi sono autorizzati a navigare il sito o alcune parti di una pagina possono apparire solo a certi utenti, o ancora una pagina stessa può totalmente cambiare il suo aspetto in base al livello accertato dell’utente che naviga.

***Stampa valori da una query:***

E’ cosa molto frequente dover stampare a schermo i valori ricavati da una query di selezione, ecco come si può fare “incasellandoli” in una tabella stampata in html e riempita con un ciclo while che richiama sino ad esaurimento dei dati presenti nella tabella “astratta” la funzione *mysql_fetch_row*, in questo esempio vediamo come si stampano in una pagina html i nomi ed i cognomi (rifacendoci all’esempio dato nelle precedenti pagine) degli alunni che hanno la maggiore età:

```
<?php
$contatto=mysql_connect (“localhost”,“ciccio”,“luca”); //interfacciamento con il database
mysql_select_db(“scuola”,$contatto); //selezione del database “scuola” sul server mysql?>
<html>
<head>
<title>Alunni maggiorenni </title>
```

```
</head>
<body>
<p align="center" style=”color:#333333"><strong>Alunni che hanno più di 18 anni</strong></p>
<table width="90%" border="1"background=#333333”  align="center">
  <tr style=”background:#333333">
    <td ><div align="center"><em>Nome</em></div></td>
    <td ><div align="center"><em>Cognome</em></div></td>
  </tr>
<?php
    $ricerca=mysql_query(“SELECT nome,cognome FROM alunni WHERE eta>17”); //query che cerca i maggiorenni
    while($dati=mysql_fetch_row($ricerca)) //passaggio del dato riga per riga nel vettore “dati”
      {?>
    <tr>
    <td> <div align="center"><?php print($dati[0]); //stampa del nome ?></div></td>
    <td> <div align="center"><?php print($dati[1]); //stampa del cognome ?></div></td>
     </tr>
       }?>
</table>
</center>
</body>
</html>
```

Si otterrà quindi  una pagina dal titolo “Alunni maggiorenni” così fatta:

Alunni che hanno più di 18 anni

| *Nome* | *Cognome* |
|---|---|
| Claudio | Rossi |
| Giusto | Bianchi |

Come è possibile vedere il ciclo while è comandato dall’assegnazione stessa del dato dalla “tabella astratta” e quando questa non sarà più possibile per l’esaurimento dei dati il ciclo while non avrà più soddisfatta la condizione di ingresso e si esaurirà. Si noti inoltre come ogni iterazione del ciclo altro non fa che creare una riga con le apposite colonne.

***Invio di una mail con phpmailer:***

Noi tutti utilizziamo le e-mail per scambiarci messaggi di ogni tipo in tempi reali ed a qualsiasi orario e distanza; per far ciò utilizziamo dei programmi appositi (detti client) oppure possiamo utilizzare le utilities messeci a disposizione dal nostro provider di posta sul proprio sito internet.

Esistono, però, altre maniere di generare e spedire una mail, fra queste vediamo quella che tramite script php si appoggia alla classe "phpmailer" per spedire un messaggio tramite un form in una pagina web.

La classe "phpmailer" è l'insieme di una serie di istruzioni ben funzionanti "preconfezionate" che vengono richiamate (tramite il comando "include" di php) a far parte del nostro script ed a permetterci quindi l'utilizzo di particolari funzioni che servono alla composizione e spedizione di una mail, prendendo i valori dal form. Si presuppone che i file della classe siano nella stessa directory del file che stiamo programmando.

L'esempio spedisce dal sito del fabiotropia.it una mail all'indirizzo info@fabiotropia.it che si suppone esista e raccolga tutte le mail che richiedono chiarimenti ed informazioni. In quest'esempio non è necessario l'interfacciamento con il server mysql poiché non si accede a dati del database!

```
<?php

   $conferma_sped=false; //pone a falso la variabile di controllo della spedizione della mail

if ((isset($_POST['mittente'])) and (isset($_POST['oggetto'])) and (isset($_POST['testo']))) // controllo riempimento di tutti i campi

  {

            include "./class.phpmailer.php"; //richiamo delle funzionalità della classe phpmailer

            $mail = new PHPMailer(); //creazione della mail

            $mail->FromName = $_POST['mittente']; // nome del mittente

            $mail->Host = "smtp.gmail.com"; // si presuppone che sia presente una  casella apposita su gmail

            $mail->Mailer = "smtp";  // modalità di spedizione

            $mail->SMTPAuth = true;

            $mail->SMTPSecure = 'ssl';

            $mail->Port = '465';

            $mail->Username = 'prova@gmail.com';

            $mail->Password = '1234567';

            $mail->AddAddress("info@fabiotropia.it"); //indirizzo cui spedire

            $mail->Subject = $_POST['oggetto'];

            $mail->Body = $_POST['testo'];
```

```
            if($mail->Send()) // controlla che la mail sia stata spedita
              $conferma_sped=true; //pone a vero la variabile di controllo della spedizione della mail
    }?>
<html>
<head>
<title>Mail on-line </title>
</head>
<div align="center">
<table width="75%" border="0">
    <tr>
      <?php if ($conferma==false)  //caso della mail non ancora spedita
   {?>
    <form name="form1" method="post" action="">
          <td><div align="right"><strong>Mittente:</strong></div></td>
          <td><input name="mittente" type="text" id="mittente"> </strong></td>
        </tr>
        <tr>
          <td><div align="right"><strong>Oggetto del messaggio:</strong></div></td>
          <td><input name="oggetto" type="text" id="oggetto"></strong></td>
        </tr>
        <tr>
          <td><strong><em>Testo del messaggio:</em></strong></td>
          <td> </td>
        </tr>
        <tr>
          <td colspan="2"><textarea name="testo"  id="testo"></td>
        </tr>
      <p align="center">
        <input type="submit" name="Submit" value="Invia il messaggio">
```

```
            </p>
          </form>
          <?php
      }
      else //caso mail spedita con successo
      {?>
     <td>
      <br><br><br><p align="center"><font size="5" face="Times New Roman, Times, serif"><strong>
          Il messaggio è stato inviato
          </p>
      <td>
      </tr>
      <?php
      }?>
    </table>
  </body>
</html>
```

Grazie al codice scritto troveremo alla prima esecuzione della pagina l'immagine sottostante, mentre se la mail sarà inviata troveremo scritto in grassetto soltanto "Il messaggio è stato inviato".