PHP - PHP: Hypertext Preprocessor

Nicola Gessa

Introduzione

- PHP (acronimo ricorsivo per "PHP: Hypertext Preprocessor") è un linguaggio di scripting general-purpose Open Source molto utilizzato, specialmente indicato per lo sviluppo Web.
- PHP nasce nel 1994 per opera di Rasmus Lerdorf che lo utilizzava nell'implementazione delle proprie pagine web. Nel 1995 esce la prima versione.
- Nelle pagine PHP il codice PHP viene immerso nell'HTML. Il codice PHP è delimitato da speciali start e end tag che ne indicano l'inizio e la fine e che consentono di passare dal modo HTML al modo PHP.
- PHP può essere usato su tutti i principali sistemi operativi, inclusi Linux, molte varianti di Unix (compresi HP-UX, Solaris e OpenBSD), Microsoft Windows, MacOS X ed è supportato dalla maggior parte dei web server esistenti, quindi anche da Apache e IIS.

Installazione

PHP può essere utilizzato in tre ambiti differenti:

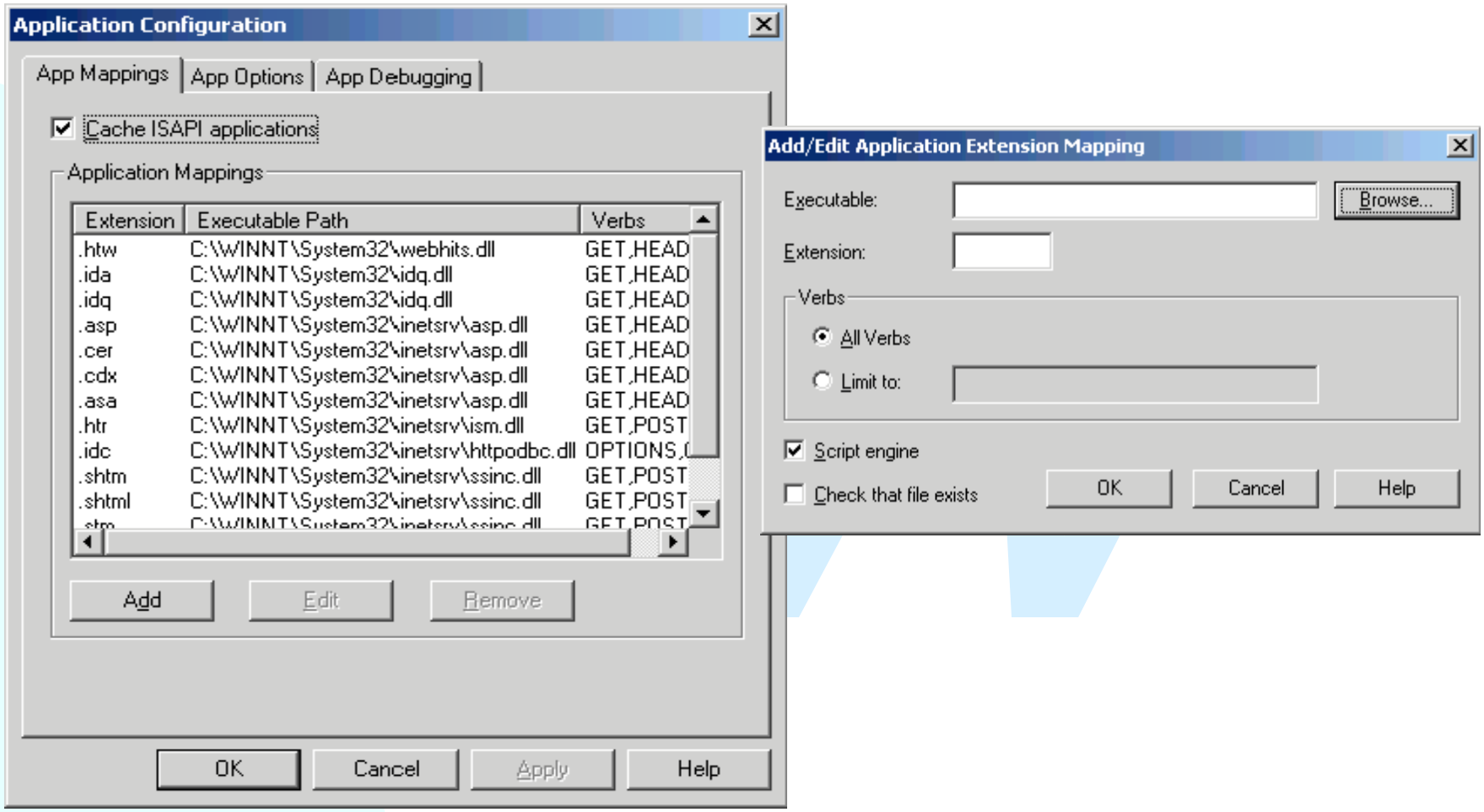
- **Server-side scripting**: in questo caso si può adottare un modulo per l'estensione del web server (es. file .dll) oppure se non si dispone del modulo si può utilizzare PHP come motore CGI (e quindi si deve disporre dell'eseguibile php.exe)
- **Command line scripting**: si installa e utilizza l'interprete **php.exe** per l'eseguzione da linea di comando.
- **Applicazioni GUI**. Consente di gestire finestre e oggetti grafici utilizzando l'estensione PHP-GTK che non è fornita con la distribuzione ufficiale di PHP.

Installazione di PHP per IIS

Per installare un modulo ISAPI per IIS:

- Nella console di configurazione di IIS, andare in 'Home Directory'.
- Premere il pulsante 'Configuration'.
- Aggiungere una nuova voce alle Application Mapping.
- Specificare il path del modulo 'php4isapi.dll'.
- Specificare l'estensione dei file da interpretare come .php.
- Marcare la checkbox "Script engine" per informare che si tratta di un engine per script.
- Fermare e riavviare IIS.

Installazione di PHP per IIS



La finestra di configurazione per le Applicazioni in IIS.

Nicola Gessa

File di configurazione

- La configurazione di PHP è registrata nel file **php.ini**.
- Il file php.ini contiene una lista di direttive.
- Nel file php.ini la sintassi da seguire è questa:
  - Righe bianche o che iniziano con **;** sono ignorate.
  - Gli header di sezione (es :[Foo]) sono ignorati.
  - Le direttive vanno specificate nella forma: **direttiva=valore**. I nomi delle direttive sono case sensitive. Il valore può essere un numero, una stringa o una costante
  - Le estensioni a PHP devono essere specificate prima del loro successivo uso nel file di configurazione.
- Se usato come linguaggio di script il file php.ini viene letto prima di ogni esecuzione di PHP. Se invece PHP è utilizzato da un web server come interprete del codice delle pagine web, il file è letto solo una volta all'avvio del server.

Un esempio con PHP

```
<html>
 <head>
  <title>Test PHP</title>
 </head>
 <body>
 <?php echo "Hello World!"; ?>
 </body>
</html>
```

- Il file non necessita di essere eseguibile. Questi file hanno l'estensione .php.
- Tutto ciò che fa è visualizzare 'Hello World!' usando la funzione echo di PHP inserita fra gli speciali tag **<?php** e **?>**
- All'interno di un file HTML si può entrare ed uscire dalla modalità PHP quante volte si desidera.

Uscire dalla modalità HTML

- Esistono 4 set di tag che possono essere utilizzati per delimitare blocchi di codice PHP:

1. `<?php echo("per inserire codice php si puo usare questo modo\n"); ?>`
2. `<?= espressione ?>`

Questa è un'abbreviazione per "<? echo espressione ?>" .

3.
```
<script language="php">
        echo ("altro modo per inserire codice
                    php");
    </script>
```
4. Tag nello stile ASP:

`<%= $variable; %>` Una abbreviazione per "<%echo .."%>

- Soltanto due di questi (<?php. . .?> e <script language="php">. . .</script>) sono sempre disponibili. Gli altri possono essere attivati o disattivati tramite il file di configurazione php.ini.

Uscire dalla modalità HTML

Il PHP permette l'uso delle strutture seguenti, dove l'output e condizionato dal valore di $expression:

```
<?php
if ($expression) {
    ?>
    <strong>Questa è vera.</strong>
    <?php
} else {
    ?>
    <strong>Questa è falsa.</strong>
    <?php
}
?>
```

Commenti

Il PHP supporta i commenti dei linguaggi 'C', 'C++' e della shell Unix. Per esempio:

```
<?php
    //Commento su una linea nella stile c++

    /* Commento su più linee
    ...ancora un'altra linea di commento */
?>
```

Lo stile di commento su "una linea", attualmente commenta solo fino alla fine della linea o del blocco corrente di codice PHP. Questo significa che il codice HTML posizionato dopo *// **?>*** sarà visualizzato

Tipi in PHP

PHP supporta 8 tipi primitivi.

4 tipi scalari :

- boolean
- integer
- float
- string

2 tipi composti:

- array
- object

2 tipi speciali:

- resource
- NULL

Tipi in PHP

- Il tipo di una variabile di solito non viene impostato dal programmatore ma deciso a runtime da PHP in base al contesto nel quale la variabile è utilizzata.
- Per controllare quale sia il tipo e i valori di una certa variabile si usa la funzione **var_dump().**
- Se si vuole una rappresentazione in stringa human-readable del tipo si usa la funzione **gettype().** Per farne delle verifiche si può utilizzate la funzione **is_type().**
- Il PHP consente il type casting per modificare il tipo di una variabile.

Esempio sull'uso delle variabili

```
<?php
$foo = "0";  // $foo è stringa (ASCII 48)
$foo += 2;   // $foo è un intero (2)
$foo = $foo + 1.3;  // $foo è un float (3.3)
$foo = 5 + "10 mele"; // $foo = 15
echo $foo."\n";                  // stampa 15

//Es. sul casting
$foo = 10;   // $foo è un intero
$bar = (boolean) $foo;   // $bar è un boolean
echo $bar."\n";   // stampa 1
?>
```

Esempio di tipi di PHP

```
<?php
$bool = TRUE;   // boolean
$str  = "foo";  // string
$int  = 12;     // integer

echo gettype($bool); // stampa "boolean"
echo gettype($str);  // stampa "string"

// se è un intero , lo incrementa di 4
if (is_int($int)) {
    $int += 4;
}

// Se $bool è una string, lo stampa
// (in questo caso non viene stampato niente)
if (is_string($bool)) {
    echo "String: $bool";
}
?>
```

Esempio var_dump()

```
<?php $a = array (1, 2, array ("a", "b", "c"));
var_dump ($a); ?>
/* output:
array(3) {
      [0]=> int(1)
      [1]=> int(2)
      [2]=> array(3) {
            [0]=> string(1) "a"
            [1]=> string(1) "b"
            [2]=> string(1) "c"
      }
} */
```

Stringhe in PHP

- Le stringhe in PHP non pongono limiti nella lunghezza.
- Si possono definire usando l'apice singolo. In questo caso la stringa non è mai interpretata.

```
echo 'questa è una semplice stringa';
```

- Si possono definire usando le doppie virgolette. In questo caso PHP interpreta le variabili e i caratteri di escape contenuti nella stringa.

```
echo "stampo la variabile \$var: $var";
```

- Si possono definire usando l'operatore <<<

Es:

```
$str = <<<EOD
Esempio di stringa definita usando la sintassi heredoc.
EOD;
```

Anche in questo caso la stringa è interpretata.

Stringhe in PHP

- Operatori di stringa:
  - l'operatore '**.**' Concatena due stringhe
  - l'operatore '**.=**'appende ad una variabile la stringa a destra dell'uguale.
  - Il confronto tra stringhe si esegue utilizzando gli operatori **==, !=, <, >, <=, >=**

Array in PHP

- PHP fornisce array associativi per memorizzare coppie chiave-valore. Non ci sono differenze fra array con indici e array associativi: PHP fornisce un solo tipo di array.
- Un array può essere creato usando il costrutto **array**:

```
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // bar
echo $arr[12];    // 1
```

- La chiave può essere sia un intero che una stringa.
- Un elemento di un array può essere di uno qualunque dei tipi PHP

```
$arr = array("somearray" => array(6 => 5, 13 => 9,
"a" => 42));

echo $arr["somearray"][6]; // 5

echo $arr["somearray"][13]; // 9

echo $arr["somearray"]["a"]; // 42
```

Array in PHP

- Se la chiave di un nuovo elemento è omessa, viene adottata come nuova chiave il massimo indice intero dell'array +1. Questo vale anche per gli indici negativi.
- Se non esiste un indice intero, la nuova chiave avrà valore 0.
- Se viene specificata una chiave già esistente il vecchio valore sarà sovrascritto.
- Non si possono usare array o oggetti come chiavi.
- Una volta creato l'array può essere modificato.

```
$arr = array(5 => 1, 12 => 2);
$arr[] = 56; // Come fare $arr[13] = 56;
$arr["x"] = 42; // Aggiunge un nuovo elemento
unset($arr[5]); // Rimuove l'elemento dall'array
unset($arr); // Cancella tutti gli elementi
```

Strutture di controllo - 1

- Costrutto **if.. else**

```
if ($a > $b) {
    print "a è maggiore di b";
} else {
    print "a NON è maggiore di b";
}
```

- **elseif** estende if aggiungendo la possibilità di eseguire un'altra istruzione nel caso in cui l'espressione contenuta nel ramo if sia **FALSE**

```
if ($a > $b) {
    print "a è maggiore di b";
} elseif ($a == $b) {
    print "a è uguale a b";
} else {
    print "a è minore di b";
}
```

Strutture di controllo - 2

- Costrutto **while**

```
$i = 1;

while ($i <= 10) {

    print $i++;

}
```

- Costrutto **do..while**

```
$i = 0;
do {
   print $i;
} while ($i>0);
```

- Costrutto **for**

```
for ($i = 1; $i <= 10; $i++) { print $i; }
```

Strutture di controllo - 3

- Costrutto **for..each**

```
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
    print "Valore corrente di \$a: $v.\n"; }
```

- Costrutto **switch**

```
switch ($i) {
    case 0: print "i è uguale a 0"; break;
    case 1: print "i è uguale a 1"; break;
    case 2: print "i è uguale a 2"; break; }
```

- **break** termina l'esecuzione di una struttura for, foreach, while, do..while o switch.
- **continue** si utilizza per interrompere l'esecuzione del ciclo corrente e continuare con l'esecuzione all'inizio del ciclo successivo

include() e require()

- include() e require() includono e valutano uno specifico file.
- include() e require() sono identiche in ogni senso eccetto per come esse trattano gli errori:include() produce un warning mentre require() restituisce un Fatal Error.
- Quando un file viene incluso, il codice che esso contiene eredita lo scope delle variabili della riga in cui si verifica l'inclusione.
- require() e include() devono essere inclusi all'interno di blocchi di istruzioni se si trovano in un blocco condizionale.
- È possibile eseguire un'istruzione return() in un file incluso per terminare l'esecuzione di quel file e restituirlo allo script che l'ha chiamato.

Funzioni in PHP

Una funzione può essere definita usando la sintassi

```
function myfunc($arg_1, $arg_2, ..., $arg_n)
{
    echo "Funzione di esempio.\n";
    return $retval;
}
```

- PHP non supporta l'overloading di funzioni
- PHP 4 supporta un numero variabile di argomenti e gli argomenti di default
- PHP supporta il passaggio di argomenti per valore e per riferimento (di default, gli argomenti della funzione sono passati per valore)
- Il passaggio per riferimento si ottiene anteponendo un ampersand **(&)** al nome dell'argomento nella definizione della funzione
- I valori vengono restituiti usando l'istruzione opzionale **return**.
- Può essere restituito qualsiasi tipo, incluse liste ed oggetti.

Classi e oggetti

- Le classi sono tipi del linguaggio.
- Una classe si definisce usando la seguente sintassi:

```
<?php
class Cart{
    var $items;  // Articoli nel carrello
                 // lo uso come array associativo

 // Aggiunge $num articoli di $artnr nel carrello
    function add_item ($artnr, $num)
    {
        $this->items[$artnr] += $num;
    }
}
?>
```

Classi e oggetti

- Per creare una variabile oggetto si usa l'operatore '**new'.**

```
$cart = new Cart;
```

- Si può accedere ai metodi della classe usando l'operatore '**->'**

```
$cart->add_item("Mele",10 );
```

Questa operazione aggiunge all'array associativo items la nuova coppia (Mele,10).

- Per stampare i valori dell'array si può usare ancora l'operare **->**

```
echo $cart->items["Mele"];     //stampa 10
```

si specifica un solo simbolo di **$** per accedere alla variabili

- Per poter accedere all'interno della classe alle funzioni e alle variabili interne della stessa classe si usa la pseudo-variabile '**$this**'

Classi e oggetti

- E’ possibile generare classi per estensione di altre classi.
- Una classe estesa o derivata ha tutte le variabili e le funzioni della classe di base più tutto ciò che viene aggiunto dall'estensione.
- Non è possibile che una sottoclasse ridefinisca variabili e funzioni di una classe madre.
- L'eredità multipla non è supportata
- Le classi si estendono usando la parola chiave **'extends'**.

```
class Named_Cart extends Cart{
    var $owner;
      function set_owner ($name)
    {
        $this->owner = $name;
    }
}
```

Classi e oggetti

- In PHP si possono definire i costruttori di classe che vengono invocati automaticamente quando viene istanziato un oggetto con l'operatore new.

```
class Auto_Cart extends Cart {
        function Auto_Cart() {
                $this->add_item ("10", 1);
        }
}
```

- L'operatore **::** è usato per riferirsi alle funzioni di classi senza istanziarle. Si possono usare funzioni della classe, ma non le variabili della classe.

Variabili speciali

PHP definisce un certo numero di array associativi speciali disponibili all'interno degli script server-side tra le quali:

- **$_SERVER**: contiene variabili impostate dal web server e relative all'ambiente di esecuzione dello script
- **$_GET**: contiene variabili ricevute dallo script via HTTP GET
- **$_POST**: contiene variabili ricevute dallo script via HTTP POST
- **$_ COOKIE**: contiene variabili ricevute dallo script tramite l'invio di cookie
- **$_ENV**: contiene variabili d'ambiente dello script.
- **$_SESSION**: contiene variabili che sono correntemente registrate nella sessione di esecuzione dello script.

Esempi con le variabili speciali

Vediamo come verificare che tipo di browser sta utilizzando la persona che visita le nostre pagine.

Per fare questo si controlla la stringa dell'user agent che il browser invia come parte della richiesta HTTP. Quest'informazione viene registrata in una variabile.

- La variabile alla quale ci riferiamo adesso è `$_SERVER["HTTP_USER_AGENT"]`

```
<?php echo $_SERVER["HTTP_USER_AGENT"]; ?>
```

- L'output (risultato) di questo script potrebbe essere:

```
Mozilla/4.0 (compatible; MSIE 5.01; Windows NT
5.0)
```

- $_SERVER è soltanto un array che automaticamente viene reso disponibile da PHP.

Gestire le form

Qualsiasi elemento inviato tramite una form è automaticamente disponibile negli script PHP.

Es:

```
<form action="action.php" method="POST">
 Il tuo Nome: <input type="text" name="name"
value="" />
 La tua et&agrave;: <input type="text" name="age"
value ="" />
 <input type="submit">
</form>
```

Quando l'utente riempie questa form e preme il pulsante submit, viene richiamata la pagina action.php.

Gestire le form - 2

- Nella pagina action.php lo script di gestione della form precedente potrebbe essere:

```
Ciao <?php echo $_POST["name"]; ?>.
La tua et&agrave; &egrave; di <?php echo
$_POST["age"]; ?> anni.
```

- Ecco un possibile output di questo script:

```
Ciao Joe.
La tua età è di 22 anni.
```

- Le variabili $_POST["name"] e $_POST["age"] vengono impostate automaticamente dal PHP.
- Se usassimo il metodo GET le informazioni ricavate dalla nostra form si troverebbero invece in **$_GET**.

Gestione delle sessioni

- PHP consente nel mantenere certi dati attraverso accessi successivi con l'uso delle sessioni.
- Il sistema di gestione delle sessioni supporta un numero di opzioni di configurazione che possono essere impostate nel file php.ini.
- Al visitatore del sito web viene assegnato un id unico, il cosidetto id di sessione. Questo viene registrato in un cookie sul lato utente o è propagato tramite l'URL.Quando un visitatore accede al sito, PHP controllerà automaticamente (se **session.auto_start** è settato a 1 in php.ini) se uno specifico id di sessione sia stato inviato con la richiesta.
- Le sessioni sono gestite un 2 modi
  - Tramite **session_register()** che registra variabili nella sessione corrente.
  - Con $_SESSION. Si può accedere alle variabili di sessione come a variabili normali in un array.

Gestione delle sessioni

- Registrare una variabile con $_SESSION.

```
//isset verifica se la variabile è definita
if (!isset($_SESSION['count'])) {
      $_SESSION['count'] = 0; }
else {
      $_SESSION['count']++;
}
```

- Resettare una variabile con $_SESSION.

```
unset($_SESSION['count']);
```

- Registrare una variabile con session_register().

```
$nome = ”Mario";
session_register(”nome");
```

Espressioni regolari

- Il PHP fornisce un insieme di funzioni che forniscono l'interfaccia per l'uso di espressioni regolari con una sintassi compatibile con Perl 5.
- Il supporto per le espressioni regolari è ottenuto mediante la libreria PCRE, che è un software open source, scritto da Philip Hazel, ed il cui copyright è detenuto dalla Università di Cambridge, Inghilterra. A partire dalla versione 4.2.0 di PHP queste funzioni sono abilitate per default.
- Le espressioni regolari devono essere racchiuse tra delimitatori, ad esempio /. Il delimitatore finale può essere seguito da vari modificatori che agiscono sul criterio di riconoscimento.
- Le funzioni per le espressioni regolari consentono anche le operazioni di sostituzione e split delle stringhe.

Espressioni regolari - funzioni

- La funzione **preg_grep()** restituisce un array composto dagli elementi dell'array preso in input che soddisfano i criteri impostati nell'espressione regolare specificata.

```
$array = array("1" => "stringa1", 2 => "stringa2");
$f_array = preg_grep ("/ga1/", $array);
echo $f_array[1];          //stampa stringa1
```

- La funzione **preg_match()** restituisce il numero di volte in cui è avvenuto il riconoscimento della espressione regolare nelle stringa di testo.

```
if (preg_match ("/php/i", "PHP supporta le e. r.")) {
   echo "Riconoscimento avvenuto."; }// stampa questo
else { echo "Testo non riconosciuto."; }
```

Esempio

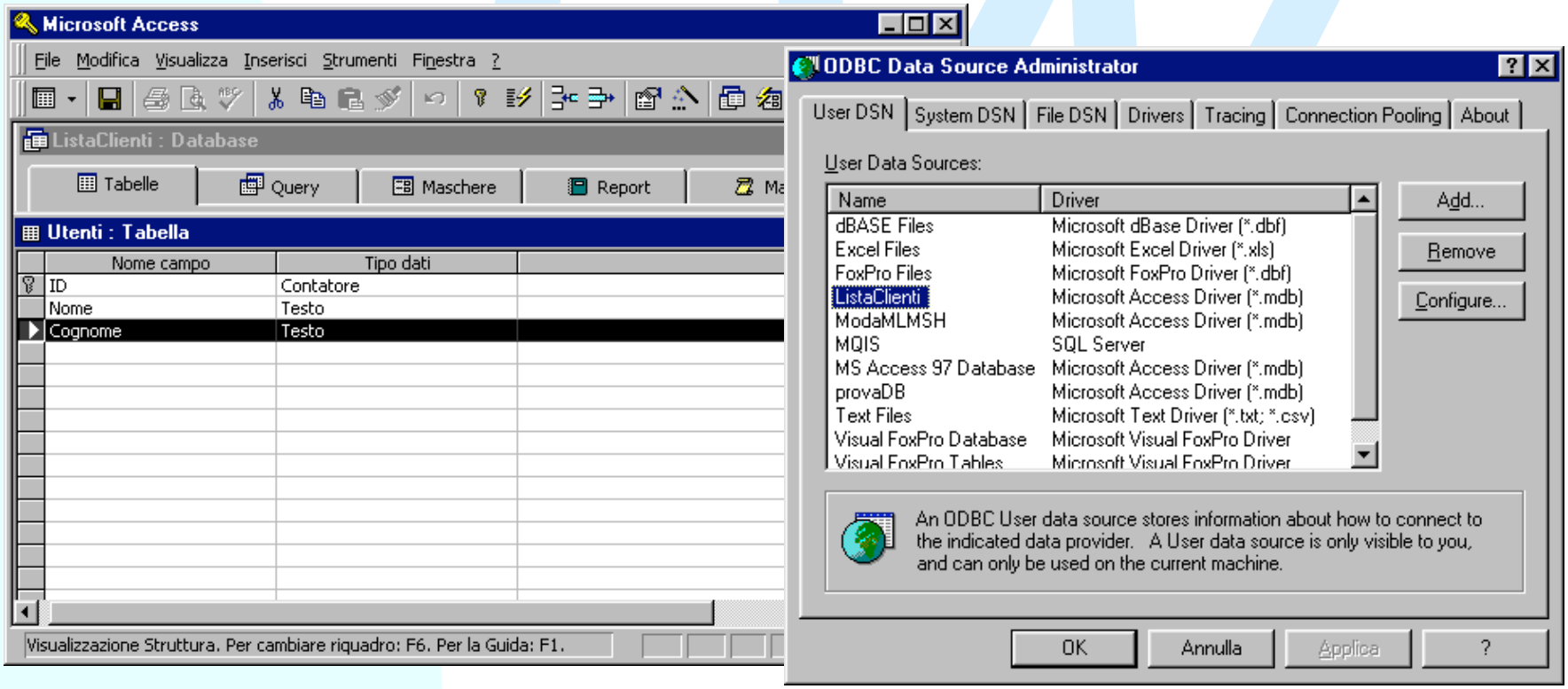
Si vuole creare una pagina PHP che consenta a degli utenti di inserire il proprio nome da aggiungere ad una lista.

Per fare questo si deve:

- Predisporre un database per la registrazione dei dati ricevuti, quindi creare le tabelle con i campi necessari.
- Creare una pagina HTML che consenta agli utenti di inserire i dati da registrare.
- Creare la pagina php che ricevuti i dati via web, utilizza le funzioni di accesso al database per la registrazione dei dati.

Predisporre il database

- Si crea il database ListaClienti con Access.
- Si crea la tabella Utente con i campi Nome, Cognome e ID.
- Si crea una sorgente dati ODBC per questo data base



Nicola Gessa

La pagina php - 1

Sia l'invio che la registrazione dei dati vengono fatti nella stessa pagina

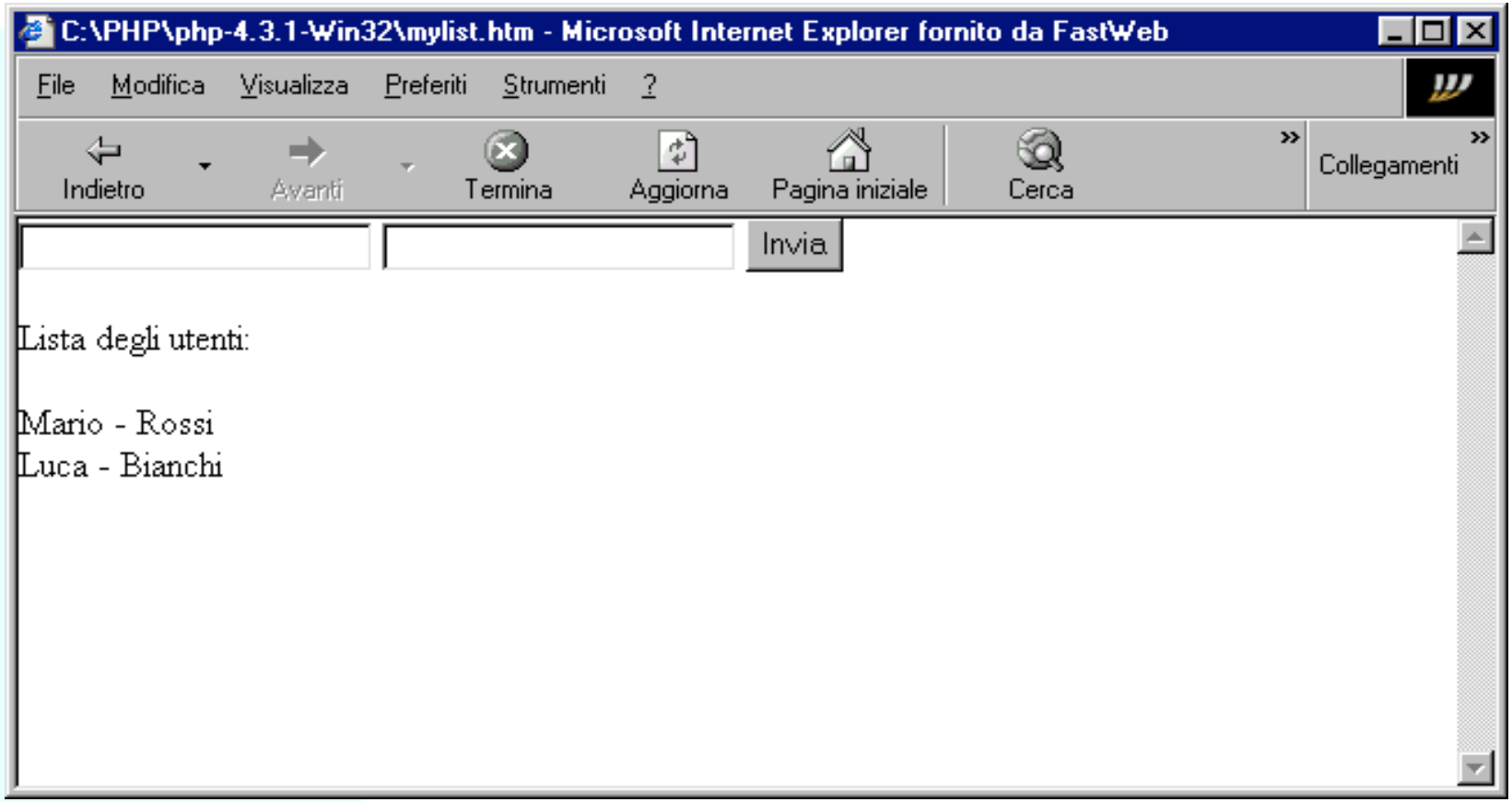
```
<?php
$myconn=odbc_connect("ListaClienti","","");
$query = <<<EOD
insert into UTENTI (Nome,Cognome) values
('$_POST[Nome]','$_POST[Cognome]')
EOD;

if($_POST["Nome"] <>""){
        $res=odbc_exec($myconn, $query);
}
?>
<HTML><HEAD></HEAD>
<BODY leftmargin="0" topmargin="0" marginwidth="0"
marginheight="0" >
```

La pagina php - 2

```
<FORM action=utenti.php method=POST>
        <INPUT TYPE="text" name="Nome">
        <INPUT TYPE="text" name="Cognome">
        <INPUT TYPE="submit" name="Invio" value="Invia"><BR>
</FORM>
<p>Lista degli utenti:<p>
<?php
$res=odbc_exec($myconn, "select * from UTENTI");
//odbc_result_all($res);   //inserisce il risultato in una
//tabella html
$i=1;
while(odbc_fetch_row($res,$i)){
        $nome = odbc_result($res, "Nome");
        $cognome = odbc_result($res, "Cognome");
        $i++;
        echo $nome." - ".$cognome."<br>";
}?>
</BODY></HTML>
```

Il risultato



Link Utili

- http://www.php.net/
- http://php.resourceindex.com/
- http://www.hotscripts.com/PHP/
- http://www.phpworld.com/