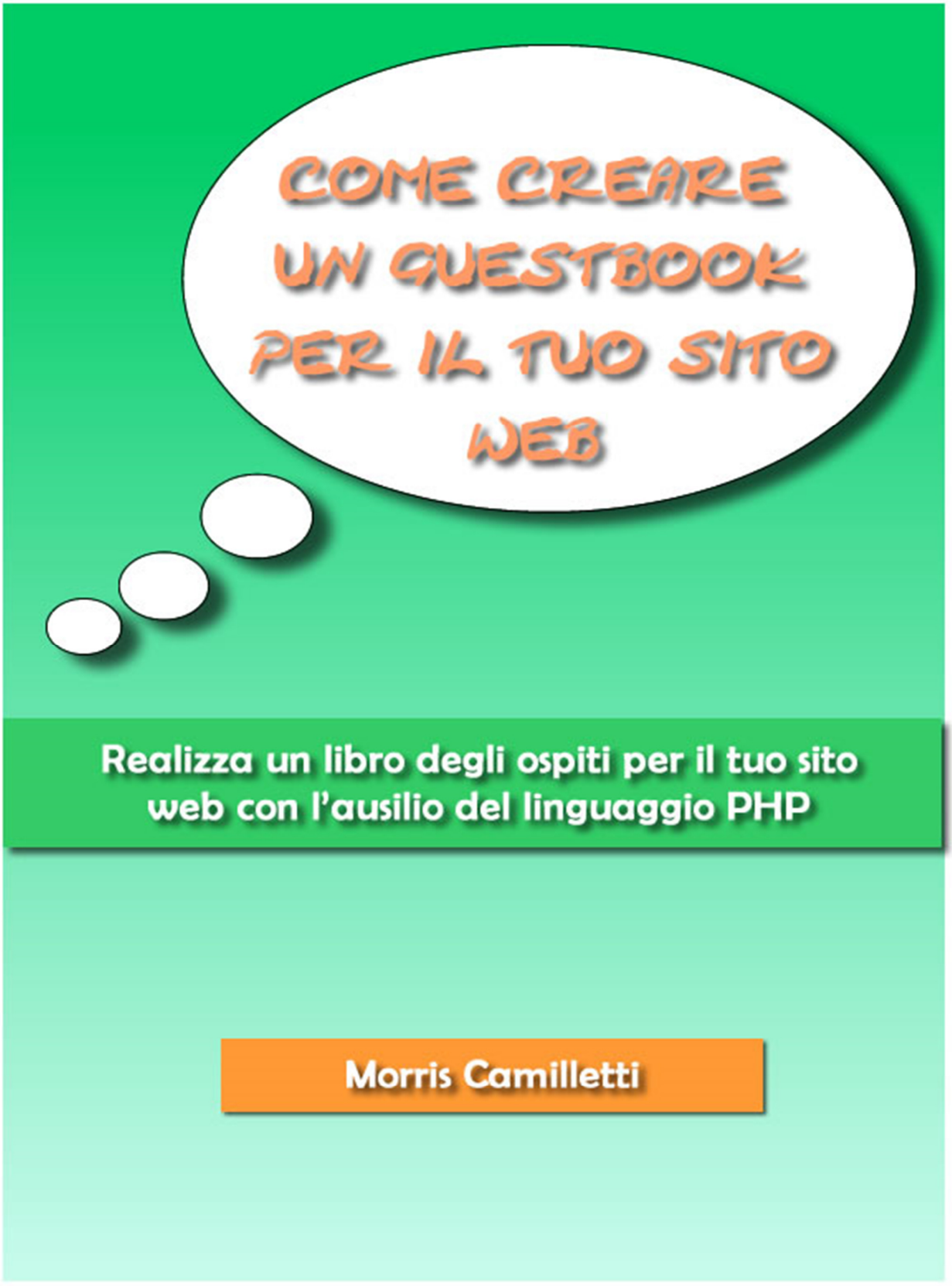

COME CREARE UN GUESTBOOK PER IL TUO SITO WEB
Realizza un libro degli ospiti per il tuo sito web con l'ausilio del linguaggio PHP
Morris Camilletti

EBOOK SCRITTO

DA MORRIS CAMILLETTI

Luglio 2013

Sommario

CHI SONO E COSA TRATTA QUESTO EBOOK..... 4

COSA TI SERVIRA' PER REALIZZARE LO SCRIPT 5

CARATTERISTICHE DEL GUESTBOOK "PRETTY GUESTBOOK" 6

FILES PER REALIZZARE IL "PRETTY GUESTBOOK" 7

COSTRUIAMO IL FILE CONFIG.PHP 8

COSTRUIAMO IL FILE LANGUAGE.DAT 10

CONTENUTO DEL FILE LANGUAGE.DAT 11

COSTRUIAMO IL FILE FORM.PHP 13

COSTRUIAMO IL FILE INSERT.PHP 20

COSTRUIAMO IL FILE VIEW.PHP 27

COSTRUIAMO IL FILE DELETE.PHP 38

COSTRUIAMO IL FILE LOGIN.PHP 49

CONCLUSIONI 52

CHI SONO E COSA TRATTA QUESTO EBOOK

Ciao, mi chiamo Morris e lavoro come programmatore. Giunto quasi al termine degli anni universitari ho cominciato ad affacciarmi alla programmazione web e passo dopo passo, o meglio tag dopo tag, mi sono addentrato sempre più nel mondo dell'HTML.

Ho trovato molto piacevole realizzare siti web e proporre sempre nuovi contenuti ai visitatori, e questo ha reso l'apprendimento del linguaggio HTML meno difficoltoso e pesante.

Con il tempo, tuttavia, mi sono reso conto che un elemento di fondamentale importanza per un sito web è l'interattività che offre; un visitatore deve essere in grado di interagire con i contenuti, esprimendo dei pareri, lasciando dei commenti e comunque in qualche modo deve prenderne parte, del resto un sito web è rivolto al visitatore e senza di lui non avrebbe senso di esistere.

Così fu che mi avvicinai al mondo del PHP, un linguaggio di programmazione orientato ad internet, che senza troppa difficoltà permette di realizzare progetti di una certa complessità.

Cominciai dunque a realizzare sondaggi, guestbook, chat e così via altri programmini (detti script) che rendevano il mio sito sempre più appetibile agli occhi di un visitatore.

Devo dire che ho trascorso molte ore sul computer per cercare di apprendere come realizzare al meglio gli script, in particolare un guestbook, che ho chiamato Pretty Guestbook, per via del suo aspetto piacevole (per lo meno ai miei occhi ☺) . Un guestbook, se già non lo sai, è un libro degli ospiti, dove un visitatore può lasciare un messaggio. Un guestbook vivacizza un sito web e coinvolge maggiormente un visitatore, che si sente libero di esprimere un parere su ciò che ha appena visitato.

Il ricordo del tempo che ho passato davanti al monitor alla ricerca di documentazioni valide mi ha spinto a scrivere questa guida, che passo dopo passo, ti condurrà a realizzare uno script con caratteristiche interessanti e ad apprendere concetti che potrai riutilizzare in altri lavori.

Ho cercato di spiegare l codice nel modo più semplice possibile, in modo tale che anche se stai muovendo i primi passi col PHP, non ti risulti di difficile comprensione.

Bene, credo di aver detto tutto, concludo con l'augurarti una buona lettura e che possa risultarti divertente realizzare script in PHP!

COSA TI SERVIRA' PER REALIZZARE LO SCRIPT

Per realizzare il guestbook ti serviranno i seguenti strumenti:

- Un server in locale che elabori le pagine php
- I files del php
- Un editor per scrivere il codice del programma

Si possono scaricare in rete dei pacchetti che contengono tutto ciò che ho citato sopra, e anche di più.
Ne cito giusto due, di cui ne sono a conoscenza:

- XAMPP
- Easy Php

Una volta che hai provato lo script sul tuo computer, per renderlo visibile ai visitatori, devi trasferire tutti i files che hai utilizzato, nel server che ospita il tuo sito web, che deve supportare il linguaggio PHP.
Per fare questo puoi utilizzare, ad esempio, dei programmi che gestiscono il trasferimento via FTP di files.
Anche in questo caso ne cito uno, di cui ne sono a conoscenza:

- Filezilla

CARATTERISTICHE DEL GUESTBOOK "PRETTY GUESTBOOK"

Cominciamo subito con l'andare a vedere le caratteristiche del guestbook che andremo a costruire:

- Paginazione dei messaggi
- Controllo validità indirizzo email inserito
- Blocco messaggi con parole troppo lunghe
- Blocco messaggi con tag html
- Controllo CAPTCHA
- Cancellazione messaggi
- Guestbook multilingua

Se alcune caratteristiche ti appaiono un po' oscure, non ti preoccupare, poi le approfondirò.

FILES PER REALIZZARE IL "PRETTY GUESTBOOK"

I files che compongono questo progetto sono i seguenti:

- CONFIG.PHP (configurazione guestbook)
- DELETE.PHP (gestisce la cancellazione dei messaggi)
- FORM.PHP (form per scrivere i messaggi)
- INSERT.PHP (gestisce l'inserimento dei messaggi)
- LANGUAGE.DAT (file per gestire la lingua)
- LOGIN.PHP (gestisce il login al pannello di amministrazione)
- VIEW.PHP (gestisce la visualizzazione dei messaggi)
- 36 immagini jpg per comporre il codice CAPTCHA

Sarai immagino curioso di sapere anche come apparirà il lavoro compiuto. Puoi vedere in azione il guestbook nel mio blog, cliccando su questo link:

http://www.tuttophp.altervista.org/prettyguest2/form.php

Il pacchetto zip, contenente tutti i files che ho menzionato sopra, puoi scaricarlo sempre nel mio blog, al seguente link:

http://www.tuttophp.altervista.org/prettyguest2/prettyguest2.zip

COSTRUIAMO IL FILE CONFIG.PHP

Vorrei partire con lo spiegare due files che saranno richiamati da tutte le pagine del progetto:

- CONFIG.PHP
- LANGUAGE.DAT

Ecco qui sotto il codice del file CONFIG.PHP.

```
<html>
<head>
<title>Pretty Guestbook 2.0</title>
</head>
<body>
<?

// ****** Pannello di configurazione guestbook ******

// Homepage del vostro sito
$home = "http://www.tuttophp.altervista.org";

// Numero dei messaggi visualizzabili per pagina
$risultati_per_pagina = 10;

// Lunghezza massima di ciascuna parola nel messaggio
$word_lenght="35";

// Autenticazione al pannello cancellazione messaggi
$user="test";
$password="test";

?>
</body>
</html>
```

Il file CONFIG.PHP contiene dei parametri personalizzabili a piacimento dall'utente. Quali sono questi parametri? Andiamo a vederli:

- $home : homepage del sito.
- $risultati_per_pagina : definisce quanti messaggi si vuole visualizzare in ogni pagina del guestbook
- $word_lenght = lunghezza massima di una singola parola che si inserisce nel messaggio
- $user : username per il panello di amministrazione dove cancellare messaggi inseriti
- $password : password per il pannello di amministrazione dove cancellare messaggi inseriti

Con un qualsiasi editor, anche Notepad, il webmaster di un sito può assegnare a suo piacimento i parametri elencati sopra. Fin qui tutto facile, no? ☺ Procediamo oltre…

COSTRUIAMO IL FILE LANGUAGE.DAT

Se realizzi un guestbook e vuoi farlo scaricare ad altre persone, devi tenere presente che queste persone possono non parlare la tua stessa lingua e magari, nonostante apprezzino il tuo lavoro, non lo scaricheranno perché non ne capiscono i contenuti.
A tal proposito è necessario realizzare una struttura multilingua che consenta a persone appartenenti a nazioni differenti di apprezzare a pieno il tuo lavoro e di poterne usufruire!
La realizzazione di questa cosa è più semplice di quanto tu stia pensando, è infatti sufficiente creare un file di testo (in questo caso ho scelto l'estensione .dat che puoi aprire anche con Notepad) nel quale andrai ad inserire le traduzioni delle parole che intendi utilizzare.

Pertanto

- Ogni parola che utilizzeremo nel guestbook verrà inserita in un array di nome **$LAN.**
- Il file LANGUAGE.DAT non è altro che un file contenitore dell'array $LAN.
- Occorre creare un file LANGUAGE.DAT per ogni lingua che si intende utilizzare

I vantaggi di questa tecnica:

- Per tradurre il guestbook non devi mettere mano nel codice del guestbook
- Non devi realizzare tanti pacchetti di DOWNLOAD con i files del guestbook quante sono le lingue che vuoi tradurre

E' sufficiente solamente che tu crei tanti files LANGUAGE.DAT quante sono le lingue che vuoi tradurre. Devi quindi gestire solamente questo file, traducendone le parole al suo interno

CONTENUTO DEL FILE LANGUAGE.DAT

```
<?php

$lan[0]="Torna indietro";

$lan[1]="Cancellazione del messaggio";

$lan[2]="PRETTY GUESTBOOK V.2.0";

$lan[3]="Homepage";

$lan[4]="Guarda i messaggi";

$lan[5]="Nome";

$lan[6]="Email";

$lan[7]="Url";

$lan[8]="Messaggio";

$lan[9]="Digita il codice che vedi";

$lan[10]="Invia";

$lan[11]="Codice digitato non valido";

$lan[12]="Rilevati tag html nel nome";

$lan[13]="Rilevati tag html nell'email";

$lan[14]="Rilevati tag html nell'URL";

$lan[15]="Rilevati tag html nel messaggio";

$lan[16]="Hai inserito una parola troppo lunga nel messaggio";

$lan[17]="Indirizzo email non valido";

$lan[18]="URL non valido";

$lan[19]="Non hai inserito il nome";

$lan[20]="Non hai inserito il messaggio";

$lan[21]="<a href='http://www.tuttophp.altervista.org' target='_blank'><font color='black' face='Verdana' size='1'
title='Guestbook realizzato da TuttoPhp'>Realizzato da TuttoPhp - Php Script</font></a>";

$lan[22]="Cancellazione messaggi";

$lan[23]="Messaggio con id";

$lan[24]="cancellato con successo";

$lan[25]="Vai alla homepage";
```

```
$lan[26]="Scrivi un messaggio";

$lan[27]="Cancella un messaggio";

$lan[28]="Pagina precedente";

$lan[29]="Pagina successiva";

$lan[30]="Non esiste il messaggio con id impostato";

$lan[31]="Numero id da cancellare:";

$lan[32]="Non ci sono ancora messaggi";

$lan[33]="Pagina";

$lan[34]="di";

$lan[35]="pagine totali";

$lan[36]="Totale messaggi:";

$lan[37]="data";

$lan[38]="ora";

$lan[39]="Indirizzo email di";

$lan[40]="Sito di";

$lan[41]="Login";

$lan[42]="Username";

$lan[43]="Password";

$lan[44]="Username o password non corrette";

?>
```

Come annunciato prima, il file LANGUAGE.DAT contiene l'array $lan con tutte le diciture che andremo ad utilizzare nel guestbook. Per rendere lo script fruibile in altre lingue è sufficiente solamente tradurre le parole contenute in questo array nella lingua desiderata!
Nel codice che andremo a scrivere quindi, piuttosto che inserire la singola parola, menzioneremo l'array. Per fare un esempio, se nel codice devo inserire la frase "Torna indietro", userò la dicitura $lan[0].

COSTRUIAMO IL FILE FORM.PHP

Cos'è un form? Il form è un modulo che consente al visitatore di un sito di inserire dei dati, che poi verranno processati dal programma. In questo caso, il visitatore inserirà a video Nome, Email, Url del suo sito web, codice CAPTCHA e ovviamente il messaggio. I campi obbligatori sono **Nome** e **Messaggio**



Forse ti starai chiedendo cos'è il codice CAPTCHA, ovvero quel codice che (come puoi vedere in figura) è formato da 5 letterine su sfondo giallo. Ebbene, probabilmente avrai già sentito parlare di SPAM, cioè di spazzatura che transita attraverso la rete. Esistono dei sistemi automatici che in rete rilevano dei form (come quello del guestbook) e inseriscono a ripetizione dei messaggi pieni zeppi di link verso altri siti.
Il codice CAPTCHA obbliga il visitatore a digitare il codice mostrato a video per poter inserire il messaggio e rende quindi il guestbook invulnerabile a questi attacchi automatici.

Il codice per costruire il form è il seguente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Pretty Guestbook 2.0</title>
<link rel="stylesheet" type="text/css" href="../style_pagine.css" />

<style type="text/css">
body {
      background-color: #C2E1F3;
}
</style>
</head>
<body>
<?php
include("config.php");
include("language.dat");

// Nome del file dove vengono memorizzati i messaggi
$file = "guest.txt";

// Nome del file dove vengono memorizzati i numeri id
$idfile = "id.txt";

// Creazione del file "messaggi" se non esiste
if (!(file_exists($file)))
{fopen($file,"w");}

// Creazione del file per la memorizzazione del numero id se non esiste
if (!(file_exists($idfile)))
{
$new_id=fopen($idfile,"w");
fwrite($new_id,0);
fclose($new_id);
}

// Conteggio numero messaggi inseriti
$arr_mess=file($file);
$count=count($arr_mess);

// Conteggio della pagina alla quale si è arrivati
```

```
$pagine_totali= ceil($count/$risultati_per_pagina);

 // Generazione carattere Captcha
 function generacod()
 {
$caratteri="1234567890abcdefghijklmnopqrstuvwxyz";
$cargen=$caratteri{rand(0,35)};
$arrgen[0]=$cargen;
$codice=$cargen;
for ($i=1;$i<5;$i++)
{
$cargen=$caratteri{rand(0,35)};
$arrgen[$i]=$cargen;
$codice.=$cargen;
}
$arrgen[5]=$codice;
return $arrgen;
}

// Generazione del codice Captcha
$codicegen=generacod();
 ?>

<!-- Form per l'inserimento dei messaggi -->
<form action="insert.php" method="post">
  <table width="350" align="center" cellpadding="5" bordercolor="#4166BE"
bgcolor="#FFFFFF" style="border:1px solid;">
<tr bgcolor="#21ADE2">
<td colspan="2"><div align="center"><font color="#FFFFFF" size="1"
face="Verdana, Arial, Helvetica, sans-serif"><strong><?php echo $lan[2];
?></strong> </font></div></td>
</tr>
<tr bgcolor="#A8DFF4">
 <td colspan="2"><div align="center"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif"><?php echo "<a href=\"$home\"><font
color=\"#000000\">$lan[3]</font></a> - "."<a
href='view.php?pagina=$pagine_totali'><font
color=\"#000000\">$lan[4]</font></a>"; ?></font></div></td>
</tr>
<tr>
<td bgcolor="#FFFFFF"><div align="right"><font size="1" face="Verdana,
Arial, Helvetica, sans-serif"><?php echo $lan[5]; ?></font></div></td>
<td bgcolor="#FFFFFF"><input name="nome" type="text" id="nome" style="font-
family:Verdana;font-size:10px;" size="40" maxlength="14"></td>
</tr>
```

```
<tr>
<td bgcolor="#FFFFFF"><div align="right"><font size="1" face="Verdana,
Arial, Helvetica, sans-serif"><?php echo $lan[6]; ?></font></div></td>
<td bgcolor="#FFFFFF"><input name="email" type="text" id="email"
style="font-family:Verdana;font-size:10px;" size="40" maxlength="50"></td>
</tr>
<tr>
<td bgcolor="#FFFFFF"><div align="right"><font size="1" face="Verdana,
Arial, Helvetica, sans-serif"><?php echo $lan[7]; ?></font></div></td>
<td bgcolor="#FFFFFF"><input name="url" type="text" id="url" style="font-
family:Verdana;font-size:10px;" value="http://" size="40"
maxlength="60"></td>
</tr>
<tr>
<td valign="top" bgcolor="#FFFFFF"> <div align="right"><font size="1"
face="Verdana, Arial, Helvetica, sans-serif"><?php echo $lan[8];
?></font></div></td>
<td bgcolor="#FFFFFF"><textarea name="messaggio" cols="40" rows="7"
id="messaggio" style="font-family:Verdana;font-size:10px;"></textarea></td>
</tr>
<tr>
<td colspan="2" valign="top" bgcolor="#FFFFFF"> <table width="150"
border="0" align="left" cellpadding="0" cellspacing="0">
<tr>
<!-- Caricamento delle immagini Captcha -->
<td><?php echo "<img src='captcha/$codicegen[0].jpg'>"; ?></td>
<td><?php echo "<img src='captcha/$codicegen[1].jpg'>"; ?></td>
<td><?php echo "<img src='captcha/$codicegen[2].jpg'>"; ?></td>
<td><?php echo "<img src='captcha/$codicegen[3].jpg'>"; ?></td>
<td><?php echo "<img src='captcha/$codicegen[4].jpg'>"; ?></td>
</tr>
</table>
<table width="200" border="0">
<tr>
<td><font size="1" face="Verdana, Arial, Helvetica, sans-serif"><?php echo
$lan[9]; ?><br>
<input name="txtcaptcha" type="text" id="txtcaptcha" style="font-
family:Verdana;font-size:10px;" size="5">
</font>
</td>
</tr>
</table>
</td>
</tr>
<tr bgcolor="#A8DFF4">
```

```
<td colspan="2">
<div align="center">
<?php
// Si preleva il codice Captcha dall'array generato dalla funzione
generacod()
$codicecap=$codicegen[5];

// Si invia il codice generato alla pagina insert.php tramite campo nascosto
echo "<input type='hidden' name='captcha' value='$codicecap'>";
?>
<input type="submit" name="Submit" value="<?php echo $lan[10]; ?>"
style="background:#FBF7BF;border:1px solid;">
</div>
</td>
</tr>
</table>
</form>


<p>

</body>
</html>
```

Soffermiamoci su alcuni punti:

- In ogni pagina php che compone il guestbook troviamo le istruzioni:

  include("config.php");

  include("language.dat");

  L'istruzione *include* rende disponibile, nella pagina dove viene definita, il codice contenuto nel file a cui fa riferimento l'include stessa. In questo caso, in ogni pagina del progetto avremo sempre a disposizione le personalizzazioni definite dall'utente nel file CONFIG.PHP e le traduzioni dei testi nel file LANGUAGE.DAT.
- I dati che il visitatore deve inserire sono contenuti all'interno di una tabella (tag *<table>)*
- I dati inseriti dall'utente devono essere passati alla pagina INSERT.PHP che effettua dei controlli e li memorizza nel file "guest.txt". Dal momento che è necessario un trasferimento dati, usiamo un form (identificato dai tag *<form>)* per racchiudere la tabella. La notazione usata è la seguente:

  ```
  <form action="insert.php" method="post">
  ```

  Nel parametro "action" va inserito il nome della pagina che processa i dati, nel parametro "method" inseriamo il modo con il quale tali dati vengono passati alla pagina "insert". I metodi che si possono utilizzare sono GET e POST, in questo caso ho scelto il metodo POST.
- Ho definito la funzione GENERACOD() per generare il codice CAPTCHA.. La riprendo nella pagina seguente.

```
// Generazione carattere Captcha
function generacod()
 {
$caratteri="1234567890abcdefghijklmnopqrstuvwxyz";
$cargen=$caratteri{rand(0,35)};
$arrgen[0]=$cargen;
$codice=$cargen;
for ($i=1;$i<5;$i++)
{
$cargen=$caratteri{rand(0,35)};
$arrgen[$i]=$cargen;
$codice.=$cargen;
}
$arrgen[5]=$codice;
return $arrgen;
}

// Generazione del codice Captcha
$codicegen=generacod();
```

Nella funzione GENERACOD ho definito una stringa, *$CARATTERI*, contenente una lista di 36 possibili caratteri che possono essere inseriti nel codice CAPTCHA. Tramite la funzione *RAND* ottengo in modo casuale un numero compreso tra 0 e 35 che fungerà da indice della lista *$CARATTERI*.
Il primo carattere da inserire nel codice CAPTCHA lo pongo nell'array *$ARRGEN* nella posizione 0, poi tramite un ciclo *FOR*, estraggo gli altri 4 caratteri sempre in modo casuale, così come ho fatto per il primo.

La funzione GENERACOD restituisce dunque un array, nella fattispecie l'array *$ARRGEN*. Chiamo questa funzione *GENERACOD* e assegno i caratteri del codice CAPTCHA all'array *$CODICEGEN*.

Supponiamo di avere una cartella, di nome CAPTCHA, nel quale sono contenuti tutti e 36 i possibili caratteri del codice CAPTCHA chiamati col nome del carattere stesso. (es: 1.jpg, 2.jpg, 3.jpg, ecc..).
Sarà sufficiente scrivere le seguenti istruzioni per visualizzare a video tutti e cinque i caratteri che compongono il codice CAPTCHA.

```
<?php echo "<img src='captcha/$codicegen[0].jpg'>"; ?>
<?php echo "<img src='captcha/$codicegen[1].jpg'>"; ?>
<?php echo "<img src='captcha/$codicegen[2].jpg'>"; ?>
<?php echo "<img src='captcha/$codicegen[3].jpg'>"; ?>
<?php echo "<img src='captcha/$codicegen[4].jpg'>"; ?>
```

COSTRUIAMO IL FILE INSERT.PHP

Questo file è probabilmente quello più laborioso, perché si occupa di memorizzare nel file "guest.txt" i dati inseriti dal visitatore.

Così come fatto per i precedenti file, ti mostrerò inizialmente il codice completo, poi passerò a spiegare nel dettaglio le parti che lo compongono.

```
<html>
<head>
<title>Pretty Guestbook 2.0</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" type="text/css" href="../style_pagine.css" />
<style type="text/css">
body {
      background-color: #C2E1F3;
}
</style>
</head>

<body>
<?php

// Inclusione del file di configurazione e del linguaggio
include("config.php");
include("language.dat");

// Nome del file dove vengono memorizzati i messaggi
$file = "guest.txt";

// File del numero id
$idfile="id.txt";

// Ricezione dati
$nome=$_POST['nome'];
$email=$_POST['email'];
$url=$_POST['url'];
$messaggio=$_POST['messaggio'];

// Conteggio del numero dei messaggi già inseriti
$arr_mess=file($file);
$count=count($arr_mess);

// Conteggio della pagina alla quale si è arrivati
```

```
$pagine_totali= ceil($count/$risultati_per_pagina);

// Se l'url non è stato inserito azzero il campo
$url=trim($url);
if ($url=="http://")
{$url="";}

// codicecap = codice giusto, codicecapins = codice inserito dall'utente
$codicecap=$_POST['captcha'];
$codicecapins=$_POST['txtcaptcha'];

// Non importa se l'utente inserisce il codice in maiuscolo o minuscolo
$codicecapins=strtolower($codicecapins);

// Layout per i messaggi di errore a video
function taberrori($errmessage)
{
include("config.php");
include("language.dat");
$tab_err= "<table width=\"400\" align=\"center\" cellpadding=\"5\"
cellspacing=\"0\" bordercolor=\"#333333\" bgcolor=\"#FFFFFF\"
style=\"border:1px solid;\">";
$tab_err .= "<tr><td><center><font family='Verdana' size='2'>$errmessage<br><a
href='form.php'><font color='black'>$lan[0]</font></a></font></center>";
$tab_err .= "</td></tr></table>";
echo $tab_err;
}

// Controllo del codice Captcha
if ($codicecapins <> $codicecap)
{
taberrori($lan[11]);
exit;
}

// Rilevazione tags html in tutti i campi del form
$cercanome = (strstr($nome,"<") || strstr($nome,">"));
$cercaem = (strstr($email,"<") || strstr($email,">")) ;
$cercaurl = (strstr($url,"<")  ||  strstr($url,">"));
$cercames = (strstr($messaggio,"<") || strstr($messaggio,">"));

// Controllo lunghezza parole nel messaggio
$parole=explode(" ",$messaggio);
$dim_parole=count($parole);
$check=1;
```

```
for ($i=0;$i<$dim_parole;$i++)
{
if ( strlen($parole[$i]) > $word_lenght )
{$check=0;}
}

// Controllo tags sul nome
if ( $cercanome ){taberrori($lan[12]);exit;}

// Controllo tags sull'email
if ( $cercaem ){taberrori($lan[13]);exit;}

// Controllo tags su URL
if ( $cercaurl ){taberrori($lan[14]);exit;}

// Controllo tags su messaggio
if ($cercames){taberrori($lan[15]);exit;}

// Controllo parole troppo lunghe nel messaggio
if ($check==0) {taberrori($lan[16]);exit;}

// Controllo della validità dell'email
if (($email!="") & !(ereg("([[:alnum:]\.\-]+)(\@[[:alnum:]\.\-]+\.+)",
$email))) {taberrori($lan[17]);exit;}

// Controllo validità dell'url
if ( ($url!="") & !(eregi("http://",$url)) ) { taberrori($lan[18]);exit;}

// Controllo se il campo nome è vuoto
if ($nome == "") { taberrori($lan[19]); exit;}

// Controllo se il campo messaggio è vuoto
if ($messaggio == "") { taberrori($lan[20]); exit; }

// ****** Inserimento del messaggio ******

// Lettura del numero id
$arrid=file($idfile);
$numid=trim($arrid[0]);

// Scrittura del numero id su file
$numid++;
$nuovoid=fopen($idfile,"w");
fwrite($nuovoid,$numid);
fclose($nuovoid);
```

```
// Sostituzione dei return con punti
$messaggio=ereg_replace("\n",".",$messaggio);

// Si tolgono le barre oblique dal nome e dal messaggio
$messaggio=stripslashes($messaggio);
$nome=stripslashes($nome);

// Orario e data
$data=date("d/m/Y");
$orario=date("H.i");

// Modalità di inserimento dati su files
$commento=$nome."|".$email."|".$url."|".$messaggio."|".$data."|".$orario."|".$n
umid."\n";

// Creazione del file se non esiste
if (!(file_exists($file)))
{fopen($file,"w");}

// Scrittura del nuovo messaggio su file
$new_mess=fopen($file,"a+");
fwrite($new_mess,$commento);
fclose($new_mess);

// Conteggio dei messaggi inseriti
$arr_mess=file($file);
$count=count($arr_mess);
$pagine_totali= ceil($count/$risultati_per_pagina);

// Redirect automatico
header("Location: view.php?pagina=$pagine_totali");

?>
</body>
</html>
```

Esaminiamo in dettaglio le parti che compongono questo file:

- Come per le altre pagine includiamo i file CONFIG.PHP e LANGUAGE.DAT
- Utilizziamo due file di testo, uno “GUEST.TXT” menzionato in precedenza, per memorizzare i messaggi inseriti dai visitatori, l’ altro “ID.TXT” che tiene in memoria l’identificativo dell’ultimo

messaggio inserito. L'identificativo è importante se si vuole cancellare un messaggio. Per mezzo dell'identificativo riusciamo a caratterizza univocamente un messaggio e quindi a cancellarlo.

- La pagina INSERT.PHP riceve i dati dalla pagina FORM.PHP attraverso le seguenti istruzioni:

```
$nome=$_POST['nome'];
$email=$_POST['email'];
$url=$_POST['url'];
$messaggio=$_POST['messaggio'];
```

- Occorre effettuare dei controlli sui dati inseriti dal visitatore, per tale motivo ho definito la funzione *TABERRORI* che prende in ingresso il messaggio da visualizzare su schermo nel caso sia stata riscontrata un anomalia. La funzione *TABERRORI* crea una tabellina contenente il messaggio d'errore e inoltre anche il link "TORNA INDIETRO" in modo da permettere al visitatore di tornare a reinserire i dati in modo corretto.
  Come puoi vedere nel codice, la funzione *TABERRORI* viene chiamata per tutti i controlli effettuati dal guestbook, cioè nel caso il codice CAPTCHA non coincida con quello inserito dalvisitatore, nel caso in cui vengano rilevati tag html nel messaggio (<>), nel caso venga lasciato vuoto il campo NOME, nel caso venga lasciato vuoto il campo messaggio, ecc… ecc..

Mi soffermo su un particolare controllo, quello legato all'inserimento di parole troppo lunghe. Ritengo questo controllo molto utile in quanto una parola troppo lunga potrebbe far saltare il layout del guestbook (es: la tabellina che contiene il messaggio si allargherebbe a tal punto da rovinare l'estetica del guestbook). Per tenere sotto controllo le parole troppo lunghe ho utilizzato l'istruzione:

```
$parole=explode(" ",$messaggio);
```

La funzione *explode* in questo caso suddivide il messaggio nelle parole che lo compongono. Come puoi vedere ho utilizzato il carattere spazio " " come separatore.
Tutte le parole del messaggio vengono inserite nell'array *$parole*, e quindi ora le scansiono una ad una tramite la seguenti istruzioni:

```
$dim_parole=count($parole);
$check=1;
for ($i=0;$i<$dim_parole;$i++)
{
if ( strlen($parole[$i]) > $word_lenght )
  {$check=0;}
}
```

Inizialmente conto le parole contenute nel messaggio *($dim_parole*), quindi con un ciclo for sull'array le

analizzo controllando la lunghezza attraverso la funzione *strlen* e confrontandola con quella definita nel file CONFIG.PHP *($word_lenght*)

- Una volta che il messaggio ha superato tutti i controlli, occorre ottimizzarlo, ad esempio sostituisco le eventuali pressioni del tasto INVIO con dei punti (infatti qualche visitatore, volutamente, potrebbe premere così tante volte INVIO da rendere l'altezza del messaggio enorme!), poi elimino con la funzione *stripslashes* eventuali barre oblique inserite in automatico da PHP.

Il messaggio, infine, deve essere in qualche modo memorizzato e io come base dati ho scelto un file di testo (chiamato *guest.txt*). Tramite le seguenti istruzioni aggiungo al file l'ultimo messaggio inserito:

```
$new_mess=fopen($file,"a+");
fwrite($new_mess,$commento);
fclose($new_mess);
```

L'istruzione *fopen* apre un puntatore (*$new_mess*) al file in aggiunta (a+), l'istruzione *fwrite* scrive il messaggio mentre l'istruzione *fclose* chiude la scrittura

Il Pretty Guestbook gestisce la paginazione cioè suddivide l'intero numero dei messaggi in tante pagine, ciascuna contenente un numero dei messaggi pari a quello definito nel file CONFIG.PHP nella variabile *$risultati_per_pagina.* La paginazione verrà gestita principalmente nel file VIEW.PHP preposto alla visualizzazione dei messaggi, ora ci basta sapere solo il numero totale dei messaggi inseriti. Questo è necessario perché il visitatore, dopo che ha inserito il messaggio, deve visualizzare l'ultima pagina del guestbook, contenente i messaggi più recenti.
Il conteggio del numero totale dei messaggi è affidato alle seguenti istruzioni:

```
$arr_mess=file($file);
$count=count($arr_mess);
$pagine_totali= ceil($count/$risultati_per_pagina);
```

La funzione *CEIL* arrotonda in eccesso il risultato della divisione tra numero totale di messaggi (*$count*) e i messaggi per pagina (*$risultati_per_pagina*). Questa divisione restituisce proprio il numero totale di pagine (*$pagine_totali*)

La funzione HEADER esegue un redirect automatico (reindirizzamento automatico) verso l'ultima pagina del guestbook, quella contenente i messaggi più recenti. L'istruzione utilizzata è la seguente:

```
header("Location: view.php?pagina=$pagine_totali");
```

Come si può notare viene chiamata la pagina VIEW.PHP , preposta alla visualizzazione dei messaggi, seguita

da ? che precede la variabile pagina che assume come valore *$pagine_totali* (numero di pagine del guestbook , cioè l'ultima pagina)

- Ricordiamo che ogni messaggio inserito deve incrementare il contatore dei numeri ID che identificano ogni singolo messaggio. Questo numero poi lo vedremo meglio quando parlerò della cancellazione dei messaggi. Le istruzioni per l'aggiornamento dell'ID sono le seguenti:

  ```
  $numid++;
  $nuovoid=fopen($idfile,"w");
  fwrite($nuovoid,$numid);
  fclose($nuovoid);
  ```

  Prima incrementiamo il numero ID tramite l'istruzione *$numid++* e poi tramite le funzioni *fopen* e *fwrite* lo scriviamo sul file ID.TXT

- Un ultima nota importante riguarda la modalità di scrittura dei messaggi nel file di testo GUEST.TXT. I dati inseriti dal visitatore vengono concatenati in un'unica stringa, separati dal carattere |. La forma è cioè la seguente:

  ```
  $commento=$nome."|".$email."|".$url."|".$messaggio."|".$data."|".$orario."
  |".$numid."\n";
  ```

  Il carattere \n indica di andare a capo nel file. Capiremo l'utilità di questa notazione quando esamineremo il file VIEW.PHP per la visualizzazione dei messaggi.

COSTRUIAMO IL FILE VIEW.PHP

Dopo aver visto come si inseriscono i messaggi nel guestboiok, ora ci occupiamo della loro visualizzazione.

Qui di seguito andiamo ad analizzare il codice della pagina VIEW.PHP.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Pretty Guestbook 2.0</title>

<style type="text/css">
.Stile13 {  font-family: Verdana;
      font-size: 13px;
}
body {
      background-color: #C2E1F3;
}
</style>

</head>

<body>
<?php

include("config.php");
include("language.dat");

// Nome del file dove vengono memorizzati i messaggi
$file = "guest.txt";

// File del numero id
$idfile="id.txt";

// Creazione del file di testo se non esiste
if (!(file_exists($file)))
{ fopen($file,"w");     }

// Pagina alla quale si è arrivati
$pagina=$_GET['pagina'];
$pagina=strip_tags($pagina);

// Conteggio del numero messaggi
```

```
$rows_file=file($file);
$count=count($rows_file);
$numero_messaggi=$count;

// Messaggi inseriti=0
if ($numero_messaggi == 0)
{echo "<center><font family=\"Verdana\"
size=\"2\">$lan[32]</font></center>";}

// Paginazione dei messaggi
$inizio=0;
$pagine_totali= ceil($count/$risultati_per_pagina);

// Pagina che si sta visualizzando
if (isset($pagina) AND $pagina > 0)
{
if ($pagina > $pagine_totali)
{
$pagina = $pagine_totali;
}
else
{
$inizio=$pagina*$risultati_per_pagina - $risultati_per_pagina;
}
}
else
{
$pagina = 1;
$inizio = 0;
}

$pagina_precedente = $pagina - 1;
$pagina_seguente   = $pagina + 1;

if ($pagina_precedente < 1)
{
$pagina_precedente = 1;
}

if ($pagina_seguente > $pagine_totali)
{
$pagina_seguente = $pagine_totali;
}

?>
```

```
 <!-- Tabella contenente i messaggi degli utenti  -->
<table width="550" align="center" cellpadding="3" bordercolor="#4166BE"
bgcolor="#DAF4FE" style="border:1px solid;">
<tr>
<td bgcolor="#8DD6F1"><div align="center"><img src="images/titolo.gif"
width="500" height="60"></div></td>
</tr>
<tr>
<td bgcolor="#B4D2F5"><div align="center">
<table width="450" border="0" align="center">
<tr>
<td><div align="center"><? echo "<font face=\"Verdana\" size=\"1\">$lan[33]
$pagina $lan[34] $pagine_totali $lan[35]</font>"; ?></div></td>
<td><div align="center"><? echo "<font face=\"Verdana\" size=\"1\">$lan[36]
$numero_messaggi</font>"; ?></div></td>
<td><div align="center"><? echo "<a href=\"$home\"><img
src=\"images/home.gif\" border=\"0\" title='$lan[25]' width=\"16\"
height=\"16\">";?></div></td>
<td><div align="center"><? echo "<a href=\"form.php\"><img
src=\"images/write.gif\" border=\"0\" title='$lan[26]' width=\"20\"
height=\"20\">";?></div></td>
<td><div align="center"><? echo "<a href=\"delete.php\"><img
src=\"images/desktop.gif\" border=\"0\" title='$lan[27]' width=\"20\"
height=\"20\">";?></div></td>
</tr>
</table>
</div>
</td>
</tr>
<tr>
<td bgcolor="#B4D2F5"><div align="center">
<table width="300" border="0">
<tr>
<td height="21"><div align="center">
<?
if (!($pagina == 1))
{
if (!($pagine_totali == 0))
{
echo "<a href=\"view.php?pagina=$pagina_precedente\"><img
src=\"images/left.gif\" width=\"77\" height=\"17\" border=\"0\"
title='$lan[28]'></a>";
}
}
```

```
?>
</div>
</td>
<td>
<div align="center">
<?
if (!($pagina == $pagine_totali)){
if (!($pagine_totali == 0)){  echo "<a
href=\"view.php?pagina=$pagina_seguente\"><img src=\"images/right.gif\"
width=\"77\" height=\"17\" border=\"0\" title='$lan[29]'></a>";
}}?></div></td>
</tr>
</table>
</div>
</td>
<?
// =================
// Estrazione dati e paginazione
// =================

// Estrazione dei messaggi e stampa su schermo
$mess_pag=array_slice($rows_file,$inizio,$risultati_per_pagina);
$mess_pag=array_reverse($mess_pag);
$cont1=count($mess_pag);
$cont2=0;

while ($cont2<$cont1) {
$mess=$mess_pag[$cont2];
$info=explode("|",$mess);
$nome=$info[0];
$email=$info[1];
$url=$info[2];
$messaggio=$info[3];
$data=$info[4];
$orario=$info[5];
$id=$info[6];

// ==================================
// Creazione della tabella contenente tutti i dati del messaggio
// ==================================

// Email e URL non inseriti
if  (($email=="") & ( $url==""))
{
$riga="<tr><td bgcolor=\"#A8DFF4\"><font size=\"1\"
```

```
face=\"Verdana\"><b>$nome</b> | $lan[37]: $data | $lan[38]: $orario | id:
$id</font></td></tr>";
$riga.="<tr><td bgcolor=\"#FFFFFF\"><font size=\"1\"
face=\"Verdana\">$messaggio</td></tr>";
}
// Email inserita, URL non inserito
if  ( (!($email=="")) & ( $url=="") )
{
$riga="<tr><td bgcolor=\"#A8DFF4\"><font size=\"1\"
face=\"Verdana\"><b>$nome</b> | $lan[37]: $data | $lan[38]: $orario | id:
$id</font><a href=\"mailto:$email\" title=\"$lan[39] $nome\"><img
src=\"images/email.gif\"  border=\"0\"  align=\"absmiddle\"></a></td></tr>";
$riga.="<tr><td bgcolor=\"#FFFFFF\"><font size=\"1\"
face=\"Verdana\">$messaggio</td></tr>";
}
// Email non inserita, URL inserito
if  ( ($email=="") & (!( $url=="")) )
{
$riga="<tr><td bgcolor=\"#A8DFF4\"><font size=\"1\"
face=\"Verdana\"><b>$nome</b> | $lan[37]: $data | $lan[38]: $orario | id:
$id</font><a href=\"$url\" target=\"_blank\" title=\"$lan[40] $nome\"><img
src=\"images/home.gif\" border=\"0\"  align=\"absmiddle\"></a></td></tr>";
$riga.="<tr><td bgcolor=\"#FFFFFF\"><font size=\"1\"
face=\"Verdana\">$messaggio</td></tr>";
}
// Email e URL non inseriti
if  ( (!($email=="")) & (!( $url=="")) )
{
$riga="<tr><td bgcolor=\"#A8DFF4\"><font size=\"1\"
face=\"Verdana\"><b>$nome</b> | $lan[37]: $data | $lan[38]: $orario | id:
$id</font><a href=\"mailto:$email\" title=\"$lan[39] $nome\"><img
src=\"images/email.gif\"  border=\"0\"  align=\"absmiddle\"></a> <a
href=\"$url\" target=\"_blank\" title=\"$lan[40] $nome\"><img
src=\"images/home.gif\"  border=\"0\" align=\"absmiddle\"></a></td></tr>";
$riga.="<tr><td bgcolor=\"#FFFFFF\"><font size=\"1\"
face=\"Verdana\">$messaggio</td></tr>";
}
echo $riga;
$cont2++;
}
?>
<tr><td bgcolor="#B4D2F5"><div align="right"><font size='1'
face='Verdana'><? echo $lan[21]; ?></font> </div></td></tr>
</table>
</body>
```

```
</html>
```

Esaminiamo nel dettagli ole righe di codice:

- Notiamo subito come viene recuperata la pagina da visualizzare

```
$pagina=$_GET['pagina'];
$pagina=strip_tags($pagina);
```

Utilizziamo l'array GET recuperiamo la variabile *PAGINA* passata attraverso l'URL. Avevamo infatti visto nella pagina INSERT.PHP il link:

```
"<a href=\"view.php?pagina=$pagina_precedente\"> ….. ….
```

- Si passa successivamente al conteggio dei messaggi tramite le istruzioni:

```
$rows_file=file($file);
$count=count($rows_file);
$numero_messaggi=$count;
```

L'istruzione *file* pone tutte le righe del file GUEST.TXT in un array (*$ROWS_FILE*) e quindi tramite l'istruzione *count* viene letta la dimensione dell'array.

- Soffermiamoci ora su un aspetto interessante, la paginazione del guestbook. Esaminiamo la seguente porzione di codice:

```
// Paginazione dei messaggi
$inizio=0;
$pagine_totali= ceil($count/$risultati_per_pagina);

// Pagina che si sta visualizzando
if (isset($pagina) AND $pagina > 0)
{
if ($pagina > $pagine_totali)
{
$pagina = $pagine_totali;
}
else
{
$inizio=$pagina*$risultati_per_pagina - $risultati_per_pagina;
}
}
else
{
$pagina = 1;
$inizio = 0;
}

$pagina_precedente = $pagina - 1;
$pagina_seguente   = $pagina + 1;

if ($pagina_precedente < 1)
{
$pagina_precedente = 1;
}

if ($pagina_seguente > $pagine_totali)
{
$pagina_seguente = $pagine_totali;
}
```

- Si inizializza la variabile *$inizio* a 0 e la variabile *$pagine_totali* al numero di pagine che risultano dalla divisione tra il numero totale dei messaggi e i risultati per pagina.
- Poi si utilizza la funzione *$isset* per controllare se la variabile *$pagina* esiste, se è stata cioè passata e in tal caso si fanno delle considerazioni:

  - Se il numero pagina passato è maggiore del numero di pagine totali, la pagina da visualizzare è

l'ultima

- Se la pagina passata non è l'ultima si valorizza la variabile $inizio così:

```
$inizio=$pagina*$risultati_per_pagina - $risultati_per_pagina;
```

Se vogliamo visualizzare la pagina 4 e supponiamo che ogni pagina contiene 10 messaggi, avremo che: *$inizio* = (4 x 10) – 10 = 30

- Se per qualche motivo la variabile *$pagina* non viene passata allora si inizializza la variabile *$pagina* a 1 e quindi si visualizza la prima pagina del guestbook
- Fatte queste considerazioni, si determina la pagina precedente e la pagina seguente rispetto quella chiamata ovviamente togliendo e aggiungendo 1 alla pagina chiamata
- Si eseguono poi dei controlli sulle variabili contenenti la pagina precedente (*$pagina_precedente*) e la pagina seguente (*$pagina_seguente*) per verificare che non escano fuori dal range delle pagine previste:

```
if ($pagina_precedente < 1)
{$pagina_precedente = 1;}

if ($pagina_seguente > $pagine_totali)
{$pagina_seguente = $pagine_totali;}
```

- La navigazione tra le pagine del guestbook è affidata a due immagini, due frecciettine. SI fanno delle considerazioni per capire se visualizzarle entrambe oppure solamente una delle due.
  Se ad esempio viene chiamata l'ultima pagina, deve essere visualizzata solamente la frecciettina che punta alla pagina precedente, viceversa se viene chiamata la prima pagina, deve essere visualizzata solo la frecciettina che punta alla pagina seguente.



- L'estrazione dei dati inseriti dai visitatori è affidata alla seguente routine:

```
// Estrazione dei messaggi e stampa su schermo
$mess_pag=array_slice($rows_file,$inizio,$risultati_per_pagina);
$mess_pag=array_reverse($mess_pag);
$cont1=count($mess_pag);
$cont2=0;

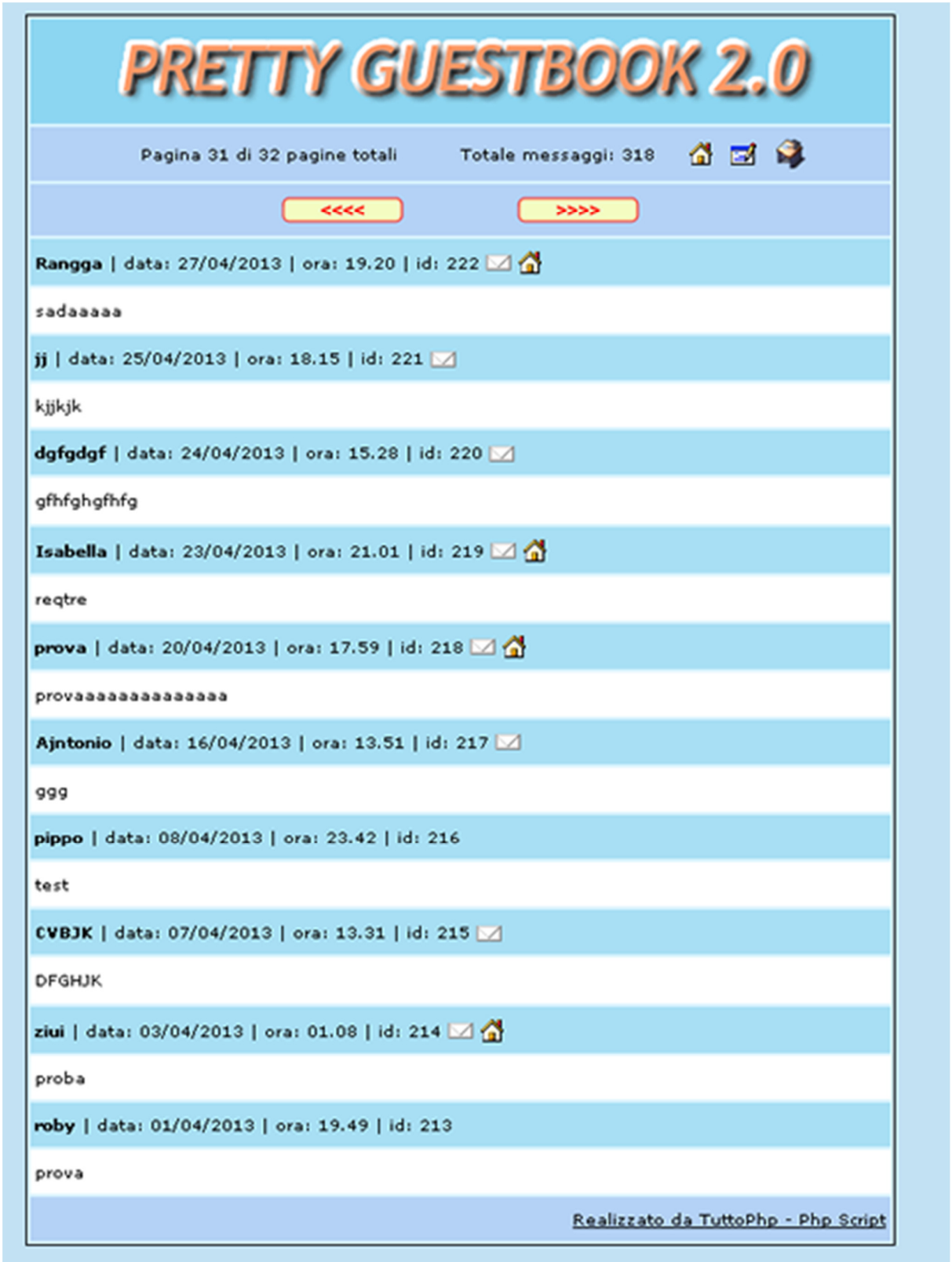
while ($cont2<$cont1) {
$mess=$mess_pag[$cont2];
$info=explode("|",$mess);
$nome=$info[0];
$email=$info[1];
$url=$info[2];
$messaggio=$info[3];
$data=$info[4];
$orario=$info[5];
$id=$info[6];
```

- La funzione *$array_slice* prende in input una porzione dell'array *$rows_file* contenente i messaggi dei visitatori, in particolare i messaggi che vanno dall'indice *$inizio* per un numero pari a *$risultati per pagina*. Per riprendere l'esempio pratico di prima, viene preso dal 30° messaggio (*$inizio* = 30) fino al 40* messaggio (*$risultati_pagina*=10) . Questa porzione di array viene chiamata *$mess_pag*.
- Successivamente l'array contenente la porzione dei messaggi (*$mess_pag)* viene rovesciato tramite la funzione reverse per avere come primo messaggio quello più recente.
- A questo punto nella variabile *$cont1* mettiamo la dimensione dell'array che dovrà essere 10
- Attraverso il ciclo *while* estraiamo le informazioni inserite dal visitatore. E' da notare come l'array *$mess_pag* contenga delle stringhe del tipo:

  nome visitatore | email visitatore | url visitatore | messaggio | data messaggio | ora messaggio | id messaggio

  Abbiamo già visto come le informazioni inserite dal visitatore siano state tutte concatenate con il carattere separatore |.
- Occorre separare ogni singola informazione tramite la funzione *explode* indicandogli come elemento separatore il carattere | . La funzione *explode* restituisce un array *($info*) contenente proprio ogni singola informazione inserita dal visitatore

- A questo punto ogni gruppo di informazioni inserite dai visitatori costituisce una riga della tabella (variabile *$riga*). Il ciclo *while* gestisce quindi l'estrazione delle informazioni dall'array e l'inserimento nelle righe della tabella che costituisce il guestbook.


PRETTY GUESTBOOK 2.0
Pagina 31 di 32 pagine totali
Totale messaggi: 318
<<<<
>>>>
Rangga | data: 27/04/2013 | ora: 19.20 | id: 222
sadaaaaa
jj | data: 25/04/2013 | ora: 18.15 | id: 221
kjjkjk
dgfgdgf | data: 24/04/2013 | ora: 15.28 | id: 220
gfhfghgfhfg
Isabella | data: 23/04/2013 | ora: 21.01 | id: 219
reqtre
prova | data: 20/04/2013 | ora: 17.59 | id: 218
provaaaaaaaaaaaaa
Ajntonio | data: 16/04/2013 | ora: 13.51 | id: 217
ggg
pippo | data: 08/04/2013 | ora: 23.42 | id: 216
test
CVBJK | data: 07/04/2013 | ora: 13.31 | id: 215
DFGHJK
ziui | data: 03/04/2013 | ora: 01.08 | id: 214
proba
roby | data: 01/04/2013 | ora: 19.49 | id: 213
prova
Realizzato da TuttoPhp - Php Script

COSTRUIAMO IL FILE DELETE.PHP

Ora che abbiamo visto come inserire e visualizzare i messaggi inseriti dai visitatori andiamo a costruire un pannello di amministrazione che consente all'amministratore del guestbook di cancellare messaggi indesiderati.



Una piccola premessa. La prima icona del guestbook, la casetta, serve per andare all'homepage, impostabile nel file CONFIG.PHP, l'icona centrale rettangolare consente di andare al FORM per l'inserimento dei dati, mentre l'ultima icona, il tavolino, consente di accedere al pannello di amministrazione dove cancellare i messaggi indesiderati.
Andiamo quindi a vedere il codice del pannello di login per poter cancellare i messaggi indesiderati.

```
<?php

// ****** Controllo del cookie per il login ******

if (isset($_COOKIE["pretty2"]))
{
// ****** Il cookie c'è ******
$contenuto=$_COOKIE["pretty2"];
$arr_cont=explode("-",$contenuto);
$user_cookie=trim($arr_cont[0]);
$pwd_cookie=trim($arr_cont[1]);
$result=1;
}
else
{
// ****** Non c'è il cookie ******
$result=0;
}
?>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Pretty Guestbook 2.0</title><style type="text/css">
<!--
body {
      background-image: url();
      background-color: #D6E6F5;
}
-->
</style>


</head>

<body >

<br>
<br>
<?

// Inclusione del file di configurazione e del linguaggio
include("config.php");
include("language.dat");
```

```
// Nome del file dove vengono memorizzati i messaggi
$file = "guest.txt";

// Creazione del file "messaggi" se non esiste
if (!(file_exists($file)))
{fopen($file,"w");}

// Conteggio del numero dei messaggi già inseriti
$arr_mess=file($file);
$count=count($arr_mess);

// Conteggio della pagina alla quale si è arrivati
$pagine_totali= ceil($count/$risultati_per_pagina);

// Definizione tabella login
$tab_login="<form method='post' action='login.php'><table width='400'
style='border:1px solid;' align='center' cellpadding='5'
bordercolor='#4166BE' bgcolor='#FFFFFF'>";
$tab_login.="<tr><td colspan='3' bgcolor='#21ADE2'><center><font
face='Verdana' color='white'
size='2'><b>$lan[41]</b></font></center></td></tr>";
$tab_login.="<tr><td width='150'><div align='center'><font size='1'
face='Verdana'>$lan[42] <input name='user_form' type='text' style='font-
family:Verdana; font-size:10px; background-color:#E4F4FC'
size='10'></font></div></td><td width='150'><div align='center'><font
size='1' face='Verdana'>$lan[43] <input name='pwd_form' type='password'
style='font-family:Verdana; font-size:10px; background-color:#E4F4FC'
size='10'></font></div></td><td><input type='submit' name='button'
value=$lan[10]></td></tr>";
$tab_login.="<tr><td colspan='3'><div align='center'><a
href='javascript:history.back();'><font face='Verdana' color='black'
size='2'>$lan[0]</font></a></div></td></tr>";
$tab_login.="</table></form>";

// Non esiste il cookie - Tabella di login
if ($result==0)
{
echo $tab_login;
exit;
}
else
{
     if (($user_cookie==$user) & ($pwd_cookie==$password))
     {
```

```
     // ****** Login corretto ******
     }
     else
     {
     // ****** Login non corretto ******
     echo "<p><table align='center' bgcolor='white'
cellpadding='4'><tr><td><font face='Verdana'
size='2'>$lan[44]</font></td></tr></table></p>";
     echo "<p>".$tab_login."</p>";
       exit;
       }
}

// Se la variabile flg non esiste allora si è in modalità interattiva
if (isset($_GET['flg']))
{

$id=$_GET['id'];

// ****** Modalità cancellazione del messaggio corrispondente all'id
inserito ******

// Nome del file dove vengono memorizzati i messaggi
$file = "guest.txt";

// Inizializzazione del flag cancellazione
$flgcanc="n";

// Si riversano e si contano i messaggi in un array
$arrmessaggi=file($file);
$dimarray=count($arrmessaggi);

// Azzeramento del nuovo file messaggi
$nuovofile=fopen($file,"w");
fclose($nuovofile);

// Ricerca messaggi da cancellare
$nuovofile=fopen($file,"a+");
for ($i=0;$i<$dimarray;$i++)
{
$rigamess=trim($arrmessaggi[$i])."\n";
$info=explode("|",$rigamess);
$numid=trim($info[6]);
if ($id != $numid)
{fwrite($nuovofile,$rigamess);}
```

```
else
{$flgcanc="s";}
}
fclose($nuovofile);

if ($flgcanc=="s")
{
// Avviso di cancellazione del messaggio
$tab_msg= "<table width=\"400\" align=\"center\" cellpadding=\"5\"
cellspacing=\"0\" bordercolor=\"#333333\" bgcolor=\"#FFFFFF\"
style=\"border:1px solid;\">";
$tab_msg .= "<tr><td><center><font family=\"Verdana\"
size=\"2\">".$lan[23]."=".$id." ".$lan[24]."<br><a href=\"delete.php\"><font
color='black'>$lan[0]</font></a></font></center>";
$tab_msg .= "</td></tr></table>";
}
else
{
// Messaggio che si vuole cancellare non trovato
$tab_msg= "<table width=\"400\" align=\"center\" cellpadding=\"5\"
cellspacing=\"0\" bordercolor=\"#333333\" bgcolor=\"#FFFFFF\"
style=\"border:1px solid;\">";
$tab_msg .= "<tr><td><center><font family=\"Verdana\"
size=\"2\">".$lan[30]."<br><a href=\"javascript:history.back();\"><font
color='black'>$lan[0]</font></a></font></center>";
$tab_msg .= "</td></tr></table>";
}

echo $tab_msg;
}

else
{

// ****** Form per la cancellazione dei messaggi ******

$tab_form="<form name='form_del' action='delete.php' method='get'>";
$tab_form.="<input type='hidden' name='flg' value='del'>";
$tab_form.="<table width='400' align='center' cellpadding='5'
bordercolor='#4166BE' bgcolor='#FFFFFF' style='border:1px solid;'>";
$tab_form .="<tr bgcolor='#21ADE2'><td colspan='3' bgcolor='#21ADE2'><div
align='center'><font color='#FFFFFF' size='2'
face='Verdana'><strong>$lan[1]</strong></font></div></td></tr>";
$tab_form.="<tr><td width='160' bgcolor='#FFFFFF'><div align='right'><font
size='1' face='Verdana'>$lan[31]</font></div></td><td
```

```
bgcolor='#FFFFFF'><input name='id' type='text' style='font-
family:Verdana;font-size:10px;background-color:#E4F4FC;' size='6'
maxlength='6'></td><td bgcolor='#FFFFFF'><input type='submit' name='Submit'
value=$lan[10]></td></tr>";
$tab_form.="<tr bgcolor='#A8DFF4'><td colspan='3' bgcolor='#FFFFFF'><div
align='center'></div></td></tr>";
$tab_form.="<tr bgcolor='#A8DFF4'><td colspan='3' bgcolor='#21ADE2'><div
align='center'><font size='2' face='Verdana'><strong><a
href='view.php?pagina=$pagine_totali'><font
color='#FFFFFF'>$lan[4]</font></a></strong></font></div></td></tr>";
$tab_form.="</table></form>";

echo $tab_form;
}
?>
</body>
</html>
```

Analizziamo il codice:

- Appena entriamo, il programma controlla se è stato scaricato in passato sul nostro computer un cookie, ovvero un piccolo file identificato dal nome *PRETTY2. Qu*esto cookie serve a far capire al programma se il login è già stato effettuato in precedenza oppure se è la prima volta che viene effettuato.

```
if (isset($_COOKIE["pretty2"]))
{
// ****** Il cookie c'è ******
$contenuto=$_COOKIE["pretty2"];
$arr_cont=explode("-",$contenuto);
$user_cookie=trim($arr_cont[0]);
$pwd_cookie=trim($arr_cont[1]);
$result=1;
}
else
{
// ****** Non c'è il cookie ******
$result=0;
}
```

Nel caso in cui il cookie esista, viene recuperato lo username e la password in esso memorizzati.
La variabile *$result* vale 1 se il cookie esiste, vale 0 se il cookie non esiste. Da notare come questa routine è stata inserita prima del codice delimitato dai tag <html>.

- Nella variabile *$tab_login* viene inserito il codice della tabella che contiene i campi USERNAME e PASSWORD. Ovviamente questa tabella sta dentro un form in quanto i dati inseriti devono essere poi passati alla pagina LOGIN.PHP per essere processati. Nel caso in cui la variabile *$result* vale 1, il form per il login non viene mostrato in quanto i login è già stato effettuato con successo in precedenza. Il form pertanto viene mostrato solo se la variabile *$result*=0



- Nel caso in cui il login sia valido, quindi username e password (impostati nel file CONFIG.PHP) sono stati inseriti correttamente, viene creato un cookie contenente username e password (questa operazione la compie il file LOGIN.PHP che vedremo successivamente) e poi si passa alla visualizzazione del form che ci consente di inserire l'identificativo (ID) del messaggio da cancellare.



- Ora c'è da dire che la funzione DELETE.PHP si occupa di due cose:
  - Visualizzare il form per la cancellazione del messaggio (screenshot sopra)
  - Cancellazione del messaggio

Per capire quale azione eseguire, la funzione si serve del flag *flg*. Ma vediamo meglio nel prossimo punto questa cosa.

- Una volta effettuato il login correttamente, la funzione prevede la seguente istruzione:

```
if (isset($_GET['flg']))
```

Questa istruzione significa SE ESISTE LA VARIABILE FLG…ecc..ecc..
La variabile *flg* viene valorizzata a DEL quando si sta eseguendo la cancellazione del messaggio (lo vedremo successivamente), mentre non viene valorizzata quando viene mostrato a video il form per l'inserimento dell'ID del messaggio da cancellare.

- Nel caso non esista la variabile *flg*, viene mostrato a video il form per l'inserimento dell'ID del messaggio da cancellare. Questo form è contenuto nella variabile *tab_form*.
Questo form deve richiamare sempre la pagina DELETE.PHP passandogli però il flag *flg=del* per fargli capire questa volta che deve cancellare un messaggio e non deve visualizzare il form.
La valorizzazione del flag *flg* è affidata alla seguente istruzione:

```
$tab_form.="<input type='hidden' name='flg' value='del'>";
```

Ho inserito nel form un campo nascosto, di tipo hidden, l'ho chiamato flg, e gli ho assegnato il valore *del* (simbolicamente sta per la parola delete). Questo form chiamerà la pagina DELETE.PHP passandogli il flag flg=del ed eseguirà la cancellazione del messaggio.

- Una volta capito come la funzione DELETE.PHP distingue tra visualizzazione del form e cancellazione del messaggio, si può analizzare la parte di codice che presiede alla cancellazione del messaggio. Questa è la parte di codice eseguita quando esiste la condizione mostrata poco sopra, cioè:

```
if (isset($_GET['flg']))
```

```
$id=$_GET['id'];

// ****** Modalità cancellazione del messaggio corrispondente all'id
inserito ******

// Nome del file dove vengono memorizzati i messaggi
$file = "guest.txt";

// Inizializzazione del flag cancellazione
$flgcanc="n";

// Si riversano e si contano i messaggi in un array
$arrmessaggi=file($file);
$dimarray=count($arrmessaggi);

// Azzeramento del nuovo file messaggi
$nuovofile=fopen($file,"w");
fclose($nuovofile);

// Ricerca messaggi da cancellare
$nuovofile=fopen($file,"a+");
for ($i=0;$i<$dimarray;$i++)
{
$rigamess=trim($arrmessaggi[$i])."\n";
$info=explode("|",$rigamess);
$numid=trim($info[6]);
if ($id != $numid)
{fwrite($nuovofile,$rigamess);}
else
{$flgcanc="s";}
}
fclose($nuovofile);

if ($flgcanc=="s")
{
// Avviso di cancellazione del messaggio
$tab_msg= "<table width=\"400\" align=\"center\" cellpadding=\"5\"
cellspacing=\"0\" bordercolor=\"#333333\" bgcolor=\"#FFFFFF\"
style=\"border:1px solid;\">";
$tab_msg .= "<tr><td><center><font family=\"Verdana\"
size=\"2\">".$lan[23]."=".$id." ".$lan[24]."<br><a href=\"delete.php\"><font
color='black'>$lan[0]</font></a></font></center>";
$tab_msg .= "</td></tr></table>";
}
```

```
else
{
// Messaggio che si vuole cancellare non trovato
$tab_msg= "<table width=\"400\" align=\"center\" cellpadding=\"5\"
cellspacing=\"0\" bordercolor=\"#333333\" bgcolor=\"#FFFFFF\"
style=\"border:1px solid;\">";
$tab_msg .= "<tr><td><center><font family=\"Verdana\"
size=\"2\">".$lan[30]."<br><a href=\"javascript:history.back();\"><font
color='black'>$lan[0]</font></a></font></center>";
$tab_msg .= "</td></tr></table>";
}

echo $tab_msg;
```

- Inizialmente viene recuperato l'ID del messaggio da cancellare attraverso l'array *$_GET*
- Tramite la funzione *file* si riversano i messaggi dal file all'array *arrmessaggi* e il numero di messaggi inseriti viene inserito nella variabile *dimarray.*
- A questo punto il file dei messaggi GUEST.TXT, viene prima ricreato vuoto e successivamente riempito con i messaggi dei visitatori. Le istruzioni per ricrearlo vuoto sono le seguenti:

```
$nuovofile=fopen($file,"w");
fclose($nuovofile);
```

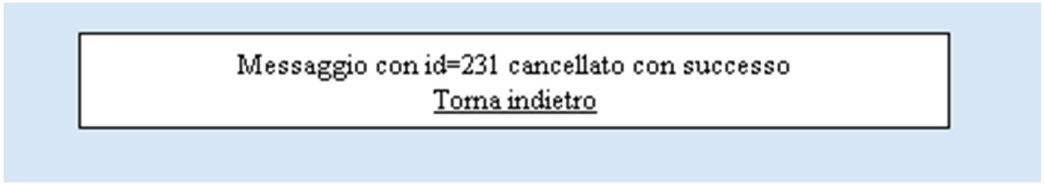
- Un ciclo for scandisce l'array *arrmessaggi* e vengono compiute le seguenti azioni:
  - Se viene individuato il messaggio con l'ID da cancellare, il messaggio non viene scritto nel file GUEST.TXT
  - Se il messaggio non corrisponde a quello da cancellare, viene riscritto nel file GUEST.TXT
- Puoi quindi capire come nel file GUEST.TXT, ricreato nuovo, sono ora contenuti tutti i messaggi tranne quello che si è voluto cancellare.
- Per individuare l'ID del messaggio, dobbiamo tenere presente una informazione già detta in precedenza quando si parlava della funzione INSERT.PHP e cioè che i messaggi vengono inseriti nel guestbook nella seguente forma:

nome visitatore | email visitatore | url visitatore | messaggio | data messaggio | ora messaggio | id messaggio

Attraverso la funzione *explode* , che ti ricordo suddivide una stringa in base al separatore | (in questo caso), otteniamo un array con le singole parti del messaggio (array *info*). A noi interessa recuperare l'ID , cioè l'ultima informazione dell'array. L'indicizzazione di un array parte dallo zero, per cui essendo il numero ID la settima informazione, corrisponderà alla posizione 6. L'istruzione per leggere il numero ID sarà pertanto la seguente:

```
$numid=trim($info[6]);
```

- Nel caso il messaggio con l'ID da cancellare viene rilevato nel file, il flag *flgcanc* viene posto uguale a S. Alla fine quindi viene emessa una tabellina (*$tab_msg*) contenente il messaggio da mostrare a video, di avvenuta o di NON avvenuta cancellazione del messaggio in base al valore che assume il flag *flgcanc.*



COSTRUIAMO IL FILE LOGIN.PHP

L'ultimo file da analizzare è il file LOGIN.PHP, chiamato dalla funzione DELETE.PHP che si occupa della cancellazione dei messaggi. Questo file in realtà si occupa solamente di memorizzare in un cookie lo username e la password dell'utente , inseriti in fase di login per la cancellare un messaggio.

Innanzitutto, se già non lo sai, un cookie non è altro che un file creato dal programma nel tuo computer. Il cookie contiene delle informazioni che il programma che l'ha generato è in grado di recuperare ogni qualvolta vi si accede.

In questo caso il cookie memorizza le credenziali di autenticazione al pannello di cancellazione messaggi, in modo tale che l'amministratore del guestbook non le debba sempre inserirle ogni qualvolta vuole cancellare un messaggio. Andiamo a vedere il codice di questo file.

```
<?

// Ricevo i dati dell'autenticazione
$user_form=trim($_POST['user_form']);
$pwd_form=trim($_POST['pwd_form']);

// Contenuto del cookie
$contenuto=$user_form."-".$pwd_form;

// Cookie di durata mezz'ora
setcookie("pretty2", $contenuto, time()+1800);

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Pretty Guestbook 2.0</title>
<style type="text/css">
<!--
body {
      background-color: #D6E6F5;
}
-->
</style></head>

<?

// Redirect automatico
header("Location: delete.php");

?>
<body>
</body>
</html>
```

Notiamo le seguenti cose:

- Il file LOGIN.PHP riceve tramite l'array *$_POST* lo username e la password
- Username e password vengono concatenati con un trattino –
- La funzione *setcookie* scrive un cookie , di nome *pretty2*, di durata mezzora contenente la stringa concatenata username-password. Quindi, per un intervallo temporale di mezzora, l'amministratore del guestbook può uscire ed entrare nel pannello di amministrazione per la cancellazione dei messaggi senza dover reinserire ogni volta username e password.
- La scrittura del cookie avviene prima dell'inserimento del codice html
- Una volta scritto il cookie, si viene reindirizzati alla pagina DELETE.PHP tramite la funzione *header*

CONCLUSIONI

Siamo giunti al termine di questo ebook; abbiamo realizzato un guestbook multilingua con importanti caratteristiche come la paginazione, il controllo CAPTCHA e la cancellazione dei messaggi.
Mi auguro che questo ebook si sia rivelato utile allo scopo che si era prefisso e che il linguaggio usato nelle spiegazioni abbia facilitato il tuo apprendimento.
Eventuali feedback sono ben accetti per cui, se vuoi, puoi contattarmi tramite l'area contatti del mio blog all'indirizzo:

http://tuttophp.altervista.org/

Non mi resta che salutarti.

Morris Camilletti