SPAW
“Progetto teledidattico”

Introduzione

Il progetto nasce con l'idea di creare un supporto per gli studenti iscritti al corso di laurea di Ingegneria Informatica e Ingegneria Meccanica con “didattica a distanza”, a seguito delle difficoltà e delle preoccupazioni espresse dagli studenti, motivate principalmente dal graduale abbandono di un supporto concreto da parte del Consorzio Nettuno.
L'invito a portare avanti questa idea è stata dello stesso prof. Calabrese che dava disponibilità a seguire un gruppetto di studenti che fosse interessato a sviluppare appunto un progetto utile a compendio del già presente CorsiOnLine (my.unipr.it). Il “gruppetto” avrebbe dato certamente un'impronta meno personale e senza dubbio più funzionale rispetto all'attuale risultato, ma per vari motivi mi sono ritrovato a portare avanti in solitudine un'idea che già comunque era partita, sia pure con un forte e importante sostegno da parte dei colleghi iscritti.
Risulta però interessante notare che la trama del progetto è sviluppata in modo tale da poter essere facilmente ampliata e implementata anche per estenderne in futuro i servizi, col supporto dei docenti e dei tutori, a tutti gli studenti, con particolare attenzione a quegli studenti lavoratori che intendessero iscriversi all'Università di Parma malgrado la sospensione della modalità “teledidattica”.

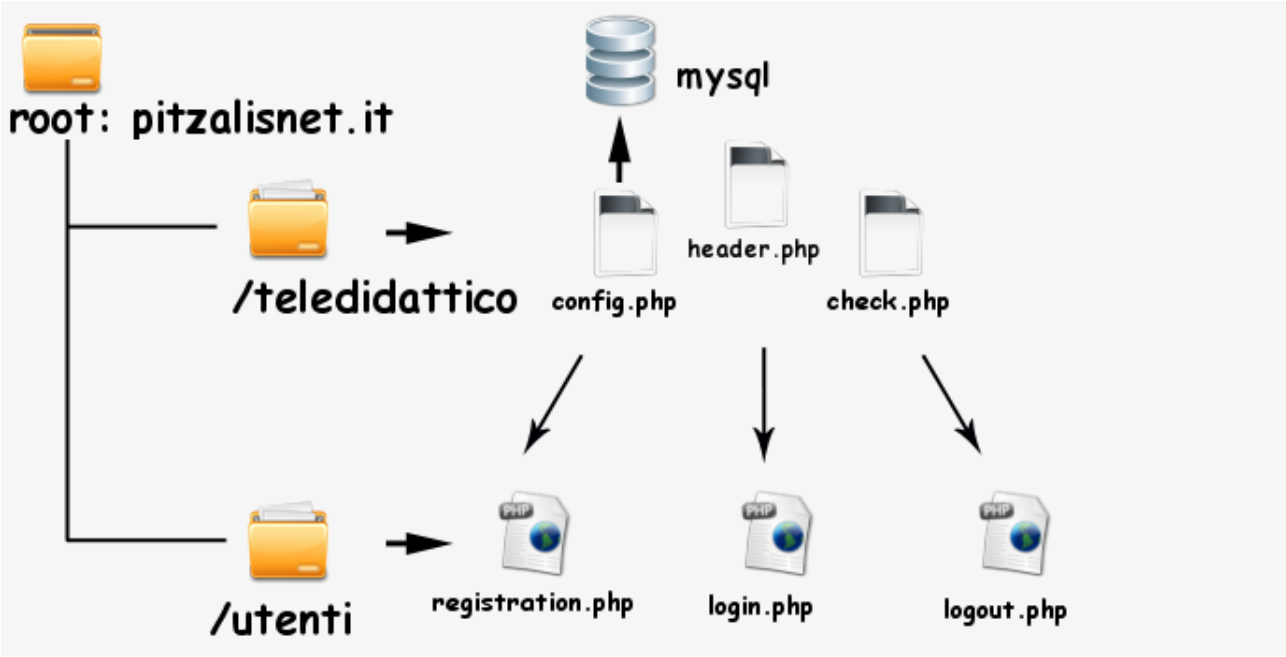
Struttura generale

Inizialmente, partendo da nozioni minime di linguaggio HTML e con l'ausilio di strumenti quali Dreamweaver e Namo Web Editor in versione trial, noti come programmi “tuttofare” per neofiti, ho allestito, nel dominio da me acquistato presso il provider Aruba, alcune pagine relative a corsi di cui avevo già conseguito l'esame, mettendo a disposizione suggerimenti, appunti, materiali e quant'altro in mio possesso, oltre a link utili. Malgrado l'elementarità del sito web, gli accessi allo stesso, non solo da parte dei cosiddetti teledidattici, cominciavano ad essere considerevoli.
Le pagine, com'è prevedibile, erano inizialmente ben lontane dal poter essere convalidate dal W3C.
Con l'inizio dello studio di strumenti per Applicazioni Web, ho abbandonato un po' alla volta i suddetti strumenti sforzandomi di elaborare le pagine col solo strumento di un text editor potente, come Notepad++. Mi sono quindi inizialmente dedicato a una sorta di pulizia (non ancora terminata) delle pagine web già in uso, cimentandomi a togliere, almeno dalle pagine più importanti, le parti ridondanti, come, a titolo di esempio, il richiamo continuo del tag *font* (Namo Web Editor, per ogni riga di testo, mi apriva e chiudeva il tag font sia pure con lo stesso carattere impostato!), riunendo in due pagine di stile (style.css e style2.css) posizionate nella root, la maggior parte degli aspetti fondamentali che identificano le pagine web da me costruite. In tal modo, nel rispetto delle direttive del W3C, a cui mi sono appoggiato per cercare eventuali errori, ho notevolmente alleggerito le pagine originarie costruite con gli strumenti prima menzionati, separando nettamente il contenuto dalla presentazione, come richiesto dalle specifiche CSS.
Un passo alla volta sono passato dall'HTML alla sua recente evoluzione XHTML, tenendo quindi conto del corretto annidamento dei tag, così come dell'uso delle minuscole per elementi e attributi, della chiusura degli elementi non vuoti, l'uso delle virgolette, l'uso di `id` anziché di `name`, eccetera.
Contemporaneamente allo studio e al perfezionamento delle pagine di stile, ho cominciato a leggere e successivamente a utilizzare piccoli codici javascript, che mi hanno permesso, ad esempio, di aprire finestre separate, successivamente di creare effetti tipo dock del Mac, modificando e adattando alle mie esigenze più script trovati in rete, fino a crearne personalmente di semplici, come il cal-

colo della media universitaria sia per informatici che per meccanici.
In seguito mi sono avvicinato al linguaggio di scripting PHP, che mi ha permesso di sviluppare pagine dinamiche dando più “vita” al progetto e permettendomi il collegamento con il database, sia per la registrazione al sito, sia per il funzionamento di un gradevole Guestbook con interfaccia grafica flash dove, per ogni corso, lo studente può lasciare una comunicazione agli altri utenti.
Ho rinunciato al forum per vari motivi: innanzitutto al momento ve n'è già uno molto ben funzionante a cui fa riferimento il sito tramite ripetuti link e ad esso è strettamente legato; esistono dei “pacchetti” già fatti come il phpBB, credo uno dei più in uso, che in fase di prova mi ha dato dei problemi di installazione dovuti al fatto che Aruba limita fortemente la modifica dei permessi di scrittura, lettura ed esecuzione dei file, specialmente per gli host windows, come è quello da me acquistato; tra l'altro, per come è strutturato il sito, non vi è necessità di un forum vero e proprio all'interno dello stesso, in quanto, come detto sopra, esiste già una semplice ed efficace interfaccia che permette all'utente di lasciare un “contributo”, come, nel caso specifico, preferisco definirlo.
Per agevolare l'utente nelle ricerche, la pagina principale contiene, un piccolo form di ricerca all'interno del sito, costituito da un servizio offerto da Google e da un file in formato xml, detto `sitemap`, che include le informazioni riguardanti il contenuto del sito e delle pagine che lo costituiscono. Questo file, una volta generato tramite un tool messo a disposizione dal provider, è stato inserito all’interno dello spazio web del mio dominio in modo tale che venga segnalata a Google la lista di tutte le pagine presenti all'interno dello spazio web stesso. Saltuariamente, specialmente in caso di modifiche e aggiunte al sito, devo provvedere all'aggiornamento di questo file .xml.
E, alla fine di questa introduzione, ciò che forse rende più interessante il progetto è la possibilità da parte dell'utente, che preferibilmente potrebbe essere un docente piuttosto che uno studente, di fare l'upload di materiale didattico specifico per ogni materia.

Accesso al sito

Premetto che fino a al 10 giugno scorso l'accesso alle pagine dedicate ai teledidattici era libero a chiunque, ma non vi era alcuna interattività, se non il download di materiale messo a disposizione dal sottoscritto o da altri studenti che me lo inviavano tramite e-mail, procurandomi un discreto impegno, pertanto vorrei soffermarmi sullo sviluppo del nuovo accesso alla sezione dedicata ai teledidattici.
Con il seguente schema intendo riassumere il lavoro svolto.



Come si vede, nella cartella 'utenti' ho inserito tre file, uno di registrazione, uno di login e uno di logout.
La pagina di login si presenta come una normale pagina HTML in cui vengono date alcune informazioni essenziali (tra cui la necessità di iscriversi alla lista del Nettuno), un form per il login e uno per registrarsi nel caso non lo si fosse già.
In fondo alla pagina un altro form, anch'esso scritto in php, che serve all'utente per inviarmi velocemente e direttamente al mio indirizzo richieste o informazioni senza necessariamente passare da un programma di posta elettronica come Outlook Express o altri.
Per quanto riguarda la registrazione, che clickando sul bottone apposito si presenta con una finestra simile a un popup, il file **registration.php** richiede la presenza, fondamentale, del file **config.php** dove sono inseriti i dati relativi al database: 'host, user, name e password; quindi, prima di procedere alla registrazione, verifica che l'utente non sia già loggato tramite il codice

```
if ($_SESSION['logged_in'] ==1) eccetera...
```

che forse potrebbe risultare inutile.
Inoltre ho incluso la presenza del file **header.php** di cui discuterò in seguito.
Se l'utente non è già loggato, all'invio del modulo compilato vengono fatti alcuni controlli, tra cui essenzialmente, se l'utente, la cui username di accesso deve coincidere con un indirizzo e-mail, è inserito o meno nella lista degli iscritti alla mailing list del Nettuno, oltre, naturalmente, ad inserimenti a mia discrezione.
Questa parte di controllo si può riassumere nelle seguenti righe:

```
<?php session_start();
switch ($_REQUEST["campoNome"]) {
case 'user1@studenti.unipr.it':
case 'user2@studenti.unipr.it':
....
....
break;
default:
die("Hai usato un indirizzo non valido nel campo username.<br />
Oppure non sei inserito nella lista!!
<meta http-equiv='refresh' content='$delay; url=$url' /><br />
<br />Stai per essere reindirizzato nella pagina di registrazione...");
}
```

Dove viene confrontato il nome inserito nel campo “campoNome” con la lunga lista che segue. In caso negativo si indica all'utente di inserire un username valido oppure lo si avvisa che non è iscritto nella lista.
Questo purtroppo mi obbliga a verificare continuamente eventuali nuove iscrizioni alla lista del Nettuno e aggiornare il file **registration.php** manualmente, ma più avanti esporrò una possibile soluzione.
Nel caso di errori nella compilazione del modulo, ho valutato ogni possibile caso spiegando eventuali incorrettezze da parte dell'utente, con ritorno automatico alla pagina di registrazione in caso di errore o chiusura automatica della finestra.
Ovviamente avevo precedentemente preparato nel database una tabella di nome “utenti” la cui struttura è importata tramite il seguente file di testo:

```
CREATE TABLE `utenti` (
`user_id` mediumint(8) NOT NULL auto_increment,
`user_email` varchar(255) NOT NULL default '',
`user_password` varchar(32) NOT NULL default '',
PRIMARY KEY (`user_id`)
);
```

Non mi soffermo sulla struttura della tabella, se non per notare che ho scelto di inserire solo la user_mail al posto di una user_name che, viceversa, ho escluso dalla registrazione.
Quindi, se l'user_mail non è già inserita nel database e, al contrario, è inserita nella suddetta lista del Nettuno, la registrazione ha buon fine: il nuovo utente viene aggiunto in una riga successiva della tabella del database e gli viene inviata automaticamente un'email di conferma. La password, come è di consuetudine, viene inserita attraverso l'algoritmo di criptazione **md5**: una volta criptata, la password non può essere più convertita in chiaro, per cui se l'utente dovesse smarrirla sarà necessario contattarmi per eliminare la relativa riga nella tabella del database e generarne una ex-novo.
Portata a buon fine la registrazione e chiusa automaticamente la finestra ci si trova nuovamente nella pagina **login.php**, dove l'utente può finalmente richiedere l'accesso ai servizi offerti. Ogni errore commesso nella compilazione del form verrà segnalato e riporterà l'utente nella pagina di login.
Per quanto riguarda il codice php del login, che anch'esso includerà il collegamento al database e all'**header.php**, lo script controllerà innanzitutto se siamo già loggati, in tal caso ci reindirizzerà direttamente verso la homepage del teledidattico. Da notare, che la stessa pagina html del login verrà inviata dal server solo dopo aver valutato alcune condizioni:

```
<?php
session_start();
…
if(.....)
{
.
.
} else {
//visualizza il form login
?>
<!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" xml:lang="it" lang="it">
<head>
...
<form...
...
</body>
</html>
<?
}
?>
```

Nel caso della spunta della casella 'ricordami', si provvederà a creare il cookie attraverso la riga di codice

```
setcookie($nome_cookie, $joined, time()+$scadenza, $percorso, $dominio);
```

dove le variabili `$nome_cookie, $scadenza, $percorso, $dominio` sono state precedentemente definite nel file **header.php** mentre `$joined` non è altro che il “valore” del cookie, definito da:

```
$joined =''.$_POST['email'].'[]'.md5($_POST['password']).'';
```

Dopo di che si passa al recupero delle informazioni con il relativo confronto nel database e in caso di esito positivo verranno impostate le variabili di sessione nel seguente modo:

```
$_SESSION['logged_in'] = 1;
$_SESSION['email'] = $_POST['email'];
$_SESSION['password'] = $_POST['password'];
session_write_close();
```

A questo punto saremo loggati e reindirizzati verso la homepage del teledidattico:

```
header('Location: http://www.pitzalisnet.it/teledidattico');
```

Infine, il file **header.php**, che come si è visto viene richiamato sia da **registration.php** che da login.php, si occupa, oltre che a definire i parametri precedentemente discussi, di controllare se nella nostra macchina è presente un cookie valido per il dominio, in caso di riscontro positivo preleva le informazioni necessarie (user_email e password) e imposta la sessione con il valore '1':

`$_SESSION['logged_in']=1`.

Tale controllo non viene effettuato se la sessione è già attiva, come possiamo notare dalla seguente condizione:

```
if($_SESSION['logged_in'] != 1 && isset($_COOKIE['login_teledidattico']))
```

ob_start, presente nel codice, dovrebbe servire per ottimizzare l'esecuzione dello script con un discreto miglioramento della velocità nel caricamento della pagina. Ob sta per Output Buffering, e il suo scopo è di inviare ogni dato dello script in una sorta di memoria temporanea, il buffer appunto.

Il piccolo file **check.php**, che verrà incluso in tutte le pagine web che richiedono protezione, verifica in sostanza, col seguente codice:

```
if ($_SESSION['logged_in'] ==0) {...
```

se l'utente è autorizzato o meno. In caso positivo la pagina verrà aperta, altrimenti viene creata una pagina html di avviso che reindirizza automaticamente nella pagina di login.
Le pagine che necessitano di protezione avranno quindi in testa semplicemente il richiamo a check.-php:

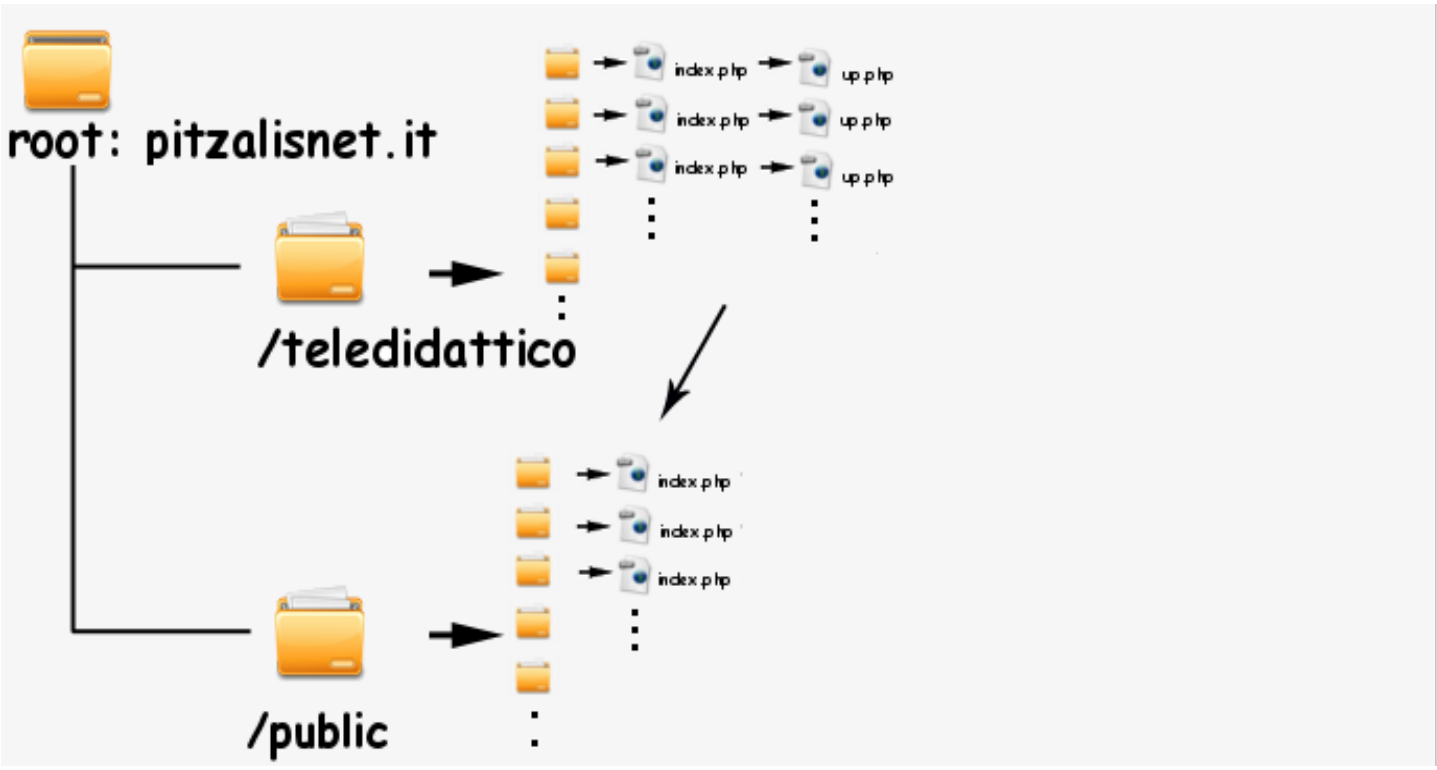
```
<? include("check.php");?>

<!DOCTYPE html PUBLIC '-//W3C//DTD XHTML 1.0 Transitional//EN'
...
```

Upload

Come accennato nell'introduzione, le pagine relative a ogni materia contengono un form contenente la possibilità di inviare in una cartella apposita (anch'essa una per ogni materia) posizionata a sua volta in una cartella `public` (pubblica appunto), del materiale utile al corso stesso.

Schematizzo in questo modo il lavoro:



In pratica nei file `index.php`, dopo il controllo dato dall'inclusione del file `check.php` come esposto nella sezione più sopra, vi è un form in cui viene richiamato il file **up.php.**
La riga chiave di codice del form è:

```
<form action="up.php" method="post" enctype="multipart/form-data">...
```

dove, oltre al richiamo del file **up.php**, specificando il valore "multipart/form-data" per enctype, il contenuto di ciascun file sarà 'impacchettato' per l'inoltro in una sezione separata di un documento multi-parte, cioè per inviare grandi quantità di dati (file, immagini ecc.).
Nel file **up.php** prima viene calcolato il 'document_root' sottraendo il numero di caratteri del 'path_translated', con l'uso della funzione booleana isset che restituisce **TRUE** se la variabile esiste, **FALSE** in caso contrario; successivamente definisco i parametri

```
$PercorsoDominio = $_SERVER['DOCUMENT_ROOT'];
$public = "/public/analisiAB/";
```

che nel caso specifico fanno riferimento alla cartella di upload relativa al corso di Analsi Matematica AB: i suddetti parametri servono per indirizzare i file proprio in questa cartella.

Dopo aver fatto alcune verifiche che comprendono l'esistenza della cartella di upload, un filtro per ammettere o meno 'Mime files' (`if (($_FILES["file"]["type"] == …)` e un controllo sulle dimensioni del file, attraverso il seguente codice

```
move_uploaded_file($_FILES["file"]["tmp_name"],
$PercorsoDominio. $public . $_FILES["file"]["name"]);
```

il file (precedentemente copiato in un file temporaneo) viene trasferito nella cartella richiesta.
Sempre all'interno della pagina **index.php** relativa al corso, vi è un link che, tramite una finestra separata, porta direttamente l'utente alla cartella di upload.
A titolo esemplificativo, nella pagina relativa a Analisi Matematica AB è possibile verificare la presenza o meno, nella cartella /public/analisiAB, di file inseriti dagli utenti. In tale cartella vi è semplicemente un file index.php la cui parte principale è

```
<?php
$dir='.';
         if ($handle = opendir($dir))
         {
           while (false !== ($file = readdir($handle)))
           {
             //qui ho messo le estensioni che non voglio vengano lette
             if ( ( substr($file,strpos($file,'.')+1)!='.' ) and
                (substr($file,strpos($file,'.')+1)!='' ) and
                ($file !='index.php' )   //evito di mostrare il file index.php
           {
               $cartella[$i]=$file;
               $i++;
           }
          }
           closedir($handle);
         }
if (is_array($cartella))
foreach ( $cartella as $file )
{
    echo "<a href=\"$dir/$file\">$file</a><br>";
}
?>
```

Con il ciclo delle ultime righe, viene letteralmente scritto un codice html che non solo mostra il contenuto della cartella, ma mi consente, per ogni file, di consentirne il download a causa del tag `"<a href=..."` posto davanti a ogni riga creata.

Guestbook con flash

Come detto nell'introduzione, ogni pagina relativa a un dato corso, è comprensiva di un'interfaccia grafica flash che consente all'utente di lasciare un commento o un suggerimento.
Tralascio la discussione relativa all'interfaccia grafica in quanto mi sono limitato a prelevare un pac-

chetto flash già fatto e modificarlo ad uso del progetto in questione tramite il programma Macromedia Flash 8.
Per rendere il più utile possibile questo servizio, ho pazientemente creato nel database una tabella per ogni corso:

```
CREATE TABLE corso (
  ID int(5) NOT NULL auto_increment,
  name text NOT NULL,
  email text NOT NULL,
  comments text NOT NULL,
  time datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (ID)
)
```

dove 'corso' verrà di volta sostituito col nome della tabella relativa alla materia.
Brevemente, attraverso l'interfaccia flash **contrib.swf**, collocata nella pagina principale del teledidattico e richiamata da tutte le pagine relative ai corsi, vengono inviati al file **guestbook.php** i dati necessari per completare la tabella del database (`nome`, `email` e `contributo`, mentre `id` e `time` vengono inseriti automaticamente).
Ogni pagina contiene il suo file **guestbook.php** che si differenzia solamente per la parte riguardante il nome della tabella:

```
// Database Connectivity Variables and other Variables
    $DBhost = "host";   // Database Server
    $DBuser = "Sql----";            // Database User
    $DBpass = "********";            // Database Pass
    $DBName = "Sql----_1";            // Database Name
    $table = "corso";             // Database Table
    $numComments = 10;       // Number of Comments per page
```

dove `corso` viene di volta in volta modificato.
Il file si divide sostanzialmente in due parti:

```
 case 'read' :
...
...
```

e

```
 case 'write' :
...
...
```

cioè una parte dove viene fatta la lettura da parte degli utenti e una dove può avvenire la scrittura.
La lettura dei dati avviene, salvo errori, e dopo aver impostato i seguenti parametri

```
$fewComments = mysql_query($sql, $DBConn);
$numfewComments = mysql_num_rows($fewComments);
```

tramite il ciclo

```
while ($array = mysql_fetch_array($fewComments)) {
```

```
                $name = mysql_result($fewComments, $i, 'name');
                $email = mysql_result($fewComments, $i, 'email');
                $comments = mysql_result($fewComments, $i, 'comments');
                $time = mysql_result($fewComments, $i, 'time');

                print '<b>Nome: </b>' . $name . '<br><b>Email: </b>' . $email .
'<br><b>Ha scritto: </b>' . $comments . '<br><i>In data: ' . $time .
'</i><br><br>';
                $i++;
            }
```

L'inserimento dei dati invece avviene con la seguente modalità:

```
$sql = 'INSERT INTO ' . $table .
                ' (`ID`,
                   `name`,
                   `email`,
                   `comments`,
                   `time`
                  )
                  VALUES
                  (\'\','
                   . '\'' . $name . '\','
                   . '\'' . $email . '\','
                   . '\'' . $comments . '\','
                   . '\'' . $submitted_on . '\'
                   )';
            $insert = mysql_query($sql, $DBConn)
```

Il resto del codice è sufficientemente intuitivo e ne tralascio la descrizione.
Il sorgente originario del codice, da me modificato e adattato al progetto, è stato prelevato da http://vademecum.aruba.it/start/flash/guestbook/index.asp

Un capitolo a parte meriterebbe l'essenziale form presente nella pagina di login, anch'esso scritto in linguaggio PHP, che consente anche all'utente non registrato di inviare un'email all'amministratore del sito.

Appendice

Possibili miglioramenti

Certamente un simile sistema con un 'eccesso' di interattività avrebbe bisogno di una modalità più efficace di controllo da parte dell'amministratore, senza dover continuamente accedere al database. Pertanto prevedo di aggiungere nel sito un modulo, con accesso esclusivo da parte del webmaster, che gli consenta di verificare velocemente sia la presenza di nuovi contributi da parte degli utenti nel guestbook, sia la presenza di nuovi file inseriti tramite apposita interfaccia di upload, consentendone eventualmente l'immediata cancellazione per scorrettezze o altro.

Sarebbe interessante in futuro, in accordo col CEDI di Parma, implementare la possibilità di inserire, da parte del docente, lezioni precedentemente videoregistrate, di modo che lo studente possa visionarle in streaming senza necessariamente dover scaricare il file. Naturalmente l'accesso a questa particolare area di upload sarebbe consentita solo ed esclusivamente al personale docente o comunque di servizio all'Università.

Possibile soluzione per l'accesso agli iscritti alla mailing list

Per ciò che concerne il problema d'accesso al sito ai nuovi iscritti nella mailing list del Polo Tecnologico, di cui si è accennato nella parte relativa alla registrazione, si potrebbe automatizzare il processo di lettura delle presenze nella lista in questo modo: il CEDI potrebbe mettere in una cartella privata, quindi non accessibile direttamente dall'utente, un file di testo con l'elenco aggiornato degli iscritti e nel file **registratio.php**, eliminando la lunga lista degli user_mail, si potrebbe andare a leggere il suddetto file di testo confrontandolo, riga per riga, col dato inserito dall'utente.
In pratica: il CEDI produce in automatico in una cartella privata, un file di testo chiamato, ad esempio, **iscritti.txt** dove vi è solo una lista di indirizzi e-mail degli iscritti. Dall'altra parte il file **registration.php**, può mantenere la seguenti righe di codice:

```
switch ($_REQUEST["campoNome"]) {
case 'user1@studenti.unipr.it':
case 'user2@studenti.unipr.it':
...
```

coi soli indirizzi a discrezione dell'amministratore, ma in coda dovrebbe inserire l'ulteriore codice che approssimativamente dovrebbe risultare come segue:

```
if (isset($_REQUEST["campoNome"])) {
  $puntatore = fopen("http://unipr../private/iscritti.txt", "r");
  $trovato = 0;
  while ((!feof($puntatore)) && (!$trovato)) {
    $linea = fgets($puntatore);
    $trovato = stristr($linea, $_POST["email"]);
    $puntatore++;
  }
  fclose($puntatore);
  if (($trovato)
...
ecc.
```

che espleta la funzione di leggere i dati dal file di testo e confrontarlo con quello immesso dall'utente.