Basi di Dati

PHP e MySQL

Prof. **Mauro Conti**

Dipartimento di Matematica - Università degli studi di Padova

conti@math.unipd.it - http://www.math.unipd.it/~conti

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Slides credits to:

Paolo Baldan

PHP e MySQL

Perché?

- Le applicazioni hanno bisogno di operare su dati persistenti
- ... e per gestire moli significative di dati (condivisi) la soluzione ovvia è un DB
- È opportuno separare logicamente (e fisicamente) la logica dell'applicazione, dalla gestione dei dati
- (cfr. Architetture software, es: Architettura three-tier, MVC (model-view-control, ...)

Come?

- L'accesso a un server MySQL da PHP tramite un API (mysql, mysqli, PDO)
- La sequenza dei passi da effettuare è:
  - Effettuare una **connessione** al server (MySQL)
  - **Selezionare il DB** o crearlo se non esiste
  - **Eseguire** la/le query (creare tabelle, inserire o selezionare dati…)
  - Nel caso di una select, elaborazione dei dati recuperati
    - con logica a cursore
    - memorizzandoli in un array

Connessione

- Attiva una connessione e restituisce un **identificatore** per questa o una segnalazione di **errore** (FALSE)
- l'identificatore della connessione sarà usato in tutti gli accessi successivi

```
/* parametri per la connessione */
$host="localhost";              /* server MySQL */
$user="baldan";                 /* utente       */
$pwd= "xxxxx";                  /* password     *//*
connessione al server */
$conn=mysql_connect($host, $user, $pwd)
   or die($_SERVER['PHP_SELF'] . "Connessione fallita!");
```

Messaggi di errore

- Può essere comodo definire una funzione che dia informazioni dettagliate e personalizzate sull'errore

```
function fail($msg) {
    die($_SERVER['PHP_SELF'] . ": $msg<br />");
    }
```

- da includere nei vari script

```
require('Errors.php');
...
...
/* connessione al server */
$conn=mysql_connect($host, $user, $pwd)
    or fail("Connessione fallita!");
```

Esecuzione di una query


```
/* seleziona il database da usare */

$dbname="Universita";
mysql_select_db($dbname);
```

```
/* prepara lo statement: media dei voti degli studenti di una
data provincia */

$prov="VE";
$query="SELECT s.Matricola, s.Nome, s.Cognome,
               ROUND(AVG(e.Voto)) AS Media
        FROM Studenti s JOIN ESAMI e ON (s.Matricola=e.Candidato)
        WHERE s.Provincia=\"$prov\"
        GROUP BY s.Matricola, s.Nome, s.Cognome";
```

Esecuzione di una query


```
/* Stampa la query a video ... utile per debug */
echo "<b>Query</b>: $query <br />";


/* ... e la esegue, ottenendo un handler per i risultati */

$studenti = mysql_query($query,$conn)
    or die("Query fallita" . mysql_error($conn));
```

Qualche risultato?

- Si può controllare la dimensione della tabella restituita dalla query eseguita con

```
mysql_num_rows($risultato)
```

- Continuando l'esempio ...

```
/* numero di righe nel risultato */
$num_righe=mysql_num_rows($studenti);

if (! $num_righe)
    echo "<p>Nessuno studente della provincia $prov
                ha fatto esami.</p>";
    else {
  /* gestione del risultato non vuoto */
}
```

Recuperare i risultati

- Il risultato di una query **SELECT** si elabora con una logica a cursore
- Per leggere la riga corrente in un array e posizionarsi sulla prossima
  - `$row = mysql_fetch_row($result)`
    ritorna la riga corrente come un array enumerativo ($row[0] = primo campo, $row[1] secondo campo, etc.)
  - `$row = mysql_fetch_assoc($result)`
    ritorna la riga corrente come un array associativo, indicizzato dai nomi dei campi
  - `$row = mysql_fetch_array($result)`
    ritorna un array enumerativo e associativo

Recuperare i risultati: row


```
if (! $num_righe)
...
else {
    echo "<p>Trovati $num_righe studenti di $prov
                che hanno fatto esami.<br />";
   echo "Ecco le loro medie:<br />";

       while ($row = mysql_fetch_row($studenti)) {

           $matricola=$row[0];     /* primo campo   */
           $nome=$row[1];          /* secondo campo */
           $cognome=$row[2];       /* ...           */
           $media=$row[3];         /* ...           */

       echo "$matricola - $nome $cognome - $media<br />";
  };

  echo "</p>";
```

Recuperare i risultati: assoc


```
if (! $num_righe)
...
else {
   echo "<p>Trovati $num_righe studenti di $prov
           che hanno fatto esami.<br />\n";

   echo "Ecco le loro medie:<br />";

   while ($row = mysql_fetch_assoc($studenti)) {

   $matricola=$row['Matricola'];
   $nome=$row['Nome'];
   $cognome=$row['Cognome'];
   $media=$row['Media'];

    echo "$matricola - $nome $cognome - $media<br />";
  };

  echo "</p>";
```

Recuperare i risultati: array


```
if (! $num_righe)
...
else {
   echo "<p>Trovati $num_righe studenti di $prov
           che hanno fatto esami.<br />\n";

   echo "Ecco le loro medie:<br />";

   while ($row = mysql_fetch_array($studenti)) {
       $matricola=$row['Matricola'];
       $nome=$row[1];
       $cognome=$row['Cognome'];
       $media=$row[3];

    echo "$matricola - $nome $cognome - $media<br />";
  };

  echo "</p>";
```

Visualizzare i risultati in una tabella HTML


```
/* funzione per stampare un array, come riga di tabella html */

function echo_row($row) {
    echo "<tr>";
    foreach ($row as $field)
        echo "<td>$field</td>";
    echo "</tr>";
};
...
```

```
/* Intestazione della tabella */
echo "
<table border=\"1\"><tr>
    <th>Matricola</th>
    <th>Nome</th>
    <th>Cognome</th>
    <th>Media</th>
</tr>";
```

Visualizzare i risultati in una tabella HTML


```
/* funzione per stampare un array, come riga di tabella html */
function echo_row($row) {
    echo "<tr>";
    foreach ($row as $field)
    echo "<td>$field</td>";
    echo "</tr>";
};
...
```

```
/* Intestazione della tabella con "here document" */
echo <<<END
<table border="1">
<tr>
   <th>Matricola</th>
   <th>Nome</th>
   <th>Cognome</th>
   <th>Media</th>
</tr>
END;
```

Visualizzare i risultati in una tabella HTML


```
/* stampa le righe della tabella */

while ($row = mysql_fetch_row($studenti))
       echo_row($row);

echo "</table>";
```

- **Nota**: L'uso di `mysql_fetch_row` è importante per il funzionamento di `echo_row`

Altre interrogazioni

- Con lo stesso meccanismo si possono eseguire altre statement SQL

```
/* inserimento ... */
$query="INSERT INTO Docenti VALUES
        (\"MM1\", \"Mino\",\"Monti\")";

$ins=mysql_query($query,$conn)
   or die("Inserimento fallito" . mysql_error($conn));
```

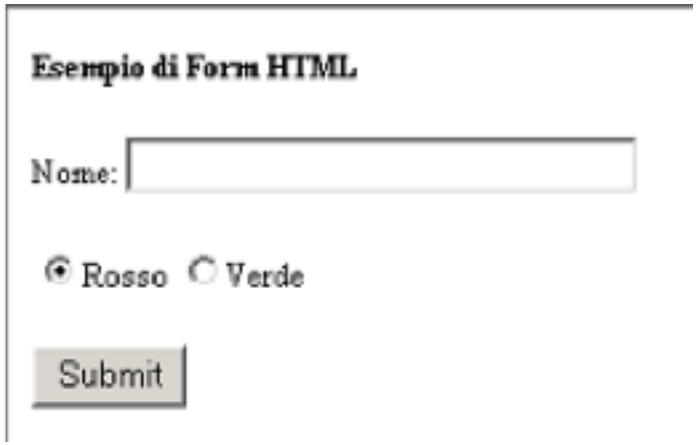
```
/* ... e la esegue, ottenendo un handler per i risultati */
$query=<<<END
DELETE    FROM Docenti
          WHERE CodDoc="MM1";
END;

$del=mysql_query($query,$conn)
   or die("Cancellazione fallita" . mysql_error($conn));
```

Interazione con l'utente

Form

- L'interazione con l'utente avviene principalmente mediante l'uso di form HTML
  - HTML fornisce vari tag per visualizzare e formattare opportunamente le FORM
  - Quando l'utente "conferma" i dati nella form, le informazioni vengono codificate e inviate al server tramite HTTP
  - Il server elabora i dati e li gestisce (nel nostro caso tramite PHP)




```
<form  action="pagina.php"
       method="get"|"post">
...
<input type="text" name="username" />
<input type="radio" name="color"
       value="Rosso" />
...
<input type="submit" />
```

- Una form, ha in generale la seguente struttura

```
<form action="manage_form.php"  method="get|post">
<fieldset>
   <legend>descrizione</legend>
   <label for="campo1">input_1</label>
   <input type="..." id="campo1" name="campo1">
  ...
   <label for="campon">input_n</label>
   <input type="..." id="campon" name="campon">
   <input type="submit" value="Procedi">
   <input type="reset"  value="Cancella">
</fieldset>

</form>
```

Form

- **<fieldset>** raggruppa input logicamente correlati
- **<label>** descrive il significato dell'input
- l'attributo **name** permette di riferire il valore dell'input nel gestore

- Per ragioni di spazio nelle slide spesso ometteremo i tag **fieldset** e **label**, ma si intende che la forma corretta è quella indicata nella slide precedente

```
<form action="manage_form.php"  method="get|post">
...
  input_1
   <input type="..." name="campo1">
  ...
  input_n
   <input type="..." name="campon">

   <input type="submit" value="Procedi">
   <input type="reset"  value="Cancella">
...
</form>
```

Input

- L'input può essere di vari tipi
  - Text
  - Password
  - Textarea
  - Radio
  - Checkbox
  - Selection/Option (menù)
  - ...

Text, password, textarea


```
<form method="get" action="manage.php">
<fieldset>
   Name
  <input type="text" name="username" value="anonymous">

   Password
  <input type="password" name="password" maxlength="8" size="8">

   Commenti
   <textarea name="comments" rows="7" cols="40">
   </textarea>

   <input type="submit" value="Invia">
   <input type="reset" value="Cancella">
</fieldset>
</form>
```

Text.html

Radio, Checkbox, Option


```
<form method="get"  action="manage.php">
    ...
    Iniziali del nome:
    <select name="nome">
    <option>---</option>
    <option>A-H</option>
    <option>I-Z</option>
    </select>

    Provincia:
    <input type="radio" name="prov" value="VE" /> VE
    <input type="radio" name="prov" value="PD" /> PD

    <input type="checkbox" name="tutor[]" value="HAS"/> Ha tutor
    <input type="checkbox" name="tutor[]" value="IS" /> E' tutor
    ...

</form>
```

Passaggi parametri: GET e POST

- **GET**
  - i parametri sono passati accodandoli alla URL del gestore della form

```
http://localhost/manage.php?dato1=pippo&dato2=pluto
```

  - la stringa dei parametri (**visibile** nel browser) viene detta **query string**
  - può essere costruita artificalmente o inserita in un bookmark
  - lunghezza massima querystring 256 caratteri
- **POST**
  - i parametri vengono passati nel body del messaggio HTTP
  - non sono visibili

Gestione dei parametri

- Uno script PHP può accedere ai parametri in tre modi:
  - `$_POST["nomepar"]` per il metodo POST
  - `$_GET["nomepar"]` per il metodo GET
- Oppure accedendo all'array globale delle richieste:
  - `$_REQUEST["nomepar"]`
  - vale per entrambi i metodi
  - lo script PHP è indipendente dal metodo usato dalla FORM
- Tutti i parametri di passaggio nelle form sono nell'ambiente predefinito di php e sono quindi visualizzabili con la funzione phpinfo()

Pagine per form

- La gestione delle form prevede due passi:
  - la visualizzazione della FORM in HTML
  - gestione dei parametri in PHP.
- Soluzione tipica: Due pagine distinte,
  - una in HTML (estensione.html)
  - l'altra come pagina PHP (estensione.php).

Esempio di Form

```
<form action="FormManager.php"  method="get">

<fieldset>
    Cognome:
    <input type="text" name="cognome"><br />
    Matricola: <input type="text" name="matricola" maxlength="6">

    Iniziale del nome:
    <select name="nome">
        <option>---</option>
        <option>A-H</option>
        <option>I-Z</option>
    </select>

    Provincia:
    <input type="radio" name="prov" value="VE" /> VE
    <input type="radio" name="prov" value="PD" /> PD
```

Esempio di Form


```
    <input type="checkbox" name="tutor[]" value="HAS" />
    Ha un tutor

    <input type="checkbox" name="tutor[]" value="IS" />
    E' tutor

    <textarea name="commento" rows="5" cols="20">
    Mah ...
    </textarea> <br /><br />

    <input type="submit" value="Invia" />
</form>
```

Gestore della Form: Recupero dei parametri

- Parametri della form recuperati dall'array **$_GET** o **$_REQUEST**

```php
<?php
$cognome = $_GET["cognome"];
$matricola = $_GET["matricola"];
$nome = $_GET["nome"];
$provincia = $_GET["prov"];
```

- Nel caso di scelte multiple (es. checkbox e select), il parametro è un array

```php
if ($_GET["tutor"]) {
    $hatutor = in_array('HAS', $_GET["tutor"]);
    $etutor =  in_array('IS',  $_GET["tutor"]);
};
$commento = $_GET["commento"];
```

Gestore della Form: Recupero dei parametri

- In fase di debug è una buona idea stampare i parametri della form
- Qui lo facciamo con HERE document

```
echo<<<END
Ecco i parametri:<br />
    <ul>
    <li>Cognome:  $cognome</li>
    <li>Iniziali del nome: $nome</li>
    <li>Matricola: $matricola</li>
    <li>Provincia: $prov</li>
    <li>Ha tutor?: $hatutor</li>
    <li>&Egrave; un tutor?: $etutor</li>
    <li>Commento: $commento</li>
</ul>
END;
```

Gestore della Form: Costruzione della query

- Costruzione della query secondo le esigenze dell'utente

```
$query="SELECT DISTINCT s.* FROM Studenti s";
```

```
/* da aggiungere in JOIN se si vuole che lo studente sia
   un Tutor */
if ($etutor)
   $join=" JOIN Studenti s1 ON (s.Matricola = s1.Tutor)";
```

```
/* clausola WHERE */
   $where=" WHERE TRUE";

/* verifica se c'e` un vincolo sul cognome ed eventualmente
   lo aggiunge al WHERE */
   if ($cognome)
      $where .= " AND  s.cognome =\"".  $cognome . "\"";
```

Gestore della Form: Costruzione della query


```
  /* verifica se c'e` un vincolo su matricola ed
     eventualmente lo aggiunge al WHERE */
if ($matricola)
    $where .= " AND  s.matricola =\"".  $matricola . "\"";
```

```
  /* verifica se c'e` un vincolo su provincia ed
     eventualmente lo aggiunge al WHERE */
if ($prov)
    $where .= "AND  s.Provincia =\"".  $prov . "\"";
```

Gestore della Form: Costruzione della query


```
  /* da aggiungere nel WHERE se vogliamo che lo studente
     abbia un nome con iniziali prefissate */
switch ($nome) {
case 'A-H':
    $where .= " AND (s.Nome REGEXP \"[A-H].*\")";
    break;

    case 'I-Z':
    $where .= " AND (s.Nome REGEXP \"[I-Z].*\")";
    break;

  /* se ($nome='---') non inserisce niente nel where! */
}
```

Gestore della Form: Costruzione della query

- Costruzione della query secondo le esigenze dell'utente

```
  /* da aggiungere nel WHERE se vogliamo che lo studente
     abbia un tutor */
if ($hatutor)
    $where .= " AND (s.Tutor IS NOT NULL)";
```

```
  /* completa la query */
$query = $query . $join . $where;

  /* come al solito conviene stamparla ... */
  echo "<b>Query</b>: $query";
```

```
  /* e la esegue */
  $studenti = mysql_query($query,$conn)
       or die("Query fallita" . mysql_error($conn));
```

Gestore della Form: Output dei risultati

- Funzioni di supporto per la gestione delle tabelle HTML

```
/* Inizia una tabella html. In input l'array degli
   header delle colonne */
function table_start($row) {
    echo "<table border=\"1\">";
    echo "<tr>";
    foreach ($row as $field)
        echo "<th>$field</th>";
    echo "</tr>";
};
```

Gestore della Form: Output dei risultati

- Funzioni di supporto per la gestione delle tabelle HTML

```
  /* Stampa un array, come riga di tabella html */function
table_row($row) {
    echo "<tr>";
    foreach ($row as $field)
        if ($field)                /* gestione valori nulli! */
            echo "<td>$field</td>";
        else
            echo "<td>---</td>";  echo "</tr>";};

/* funzione per terminare una tabella html */
function table_end() {
    echo "</table>";};
```

Gestore della Form: Output dei risultati

- Occorre includere il file delle funzioni con

```
require("table_fun.php");
```

- Quindi l'ultima parte del codice per l'ouput è:

```
/* fornisce in output i risultati in forma di tabella */
table_start(array("Nome", "Cognome",
                  "Matricola", "Nascita",
                  "Provincia", "Tutor"));
while ($row = mysql_fetch_row($studenti))
    table_row($row);

table_end();
```

Gestore della Form: Controllo dei parametri

- Spesso la prima cosa da fare è controllare che i parametri immessi nella form soddisfino i requisiti ...
  - valori nulli
  - campi numerici
  - lunghezza dei campi stringa
  - spazi in eccesso all'inizio o alla fine ... (trim)

Riassunto

- Form in un file html
- Gestore in un file php
  - recupero dei parametri da **`_GET`**, **`_POST`** o **`_REQUEST`**
  - verifica degli errori
  - elabora i dati
    - query
    - visualizza l'output sulla base del risultato

Gestore della Form: Controllo dei parametri


```
/* Verifica dei dati immessi */

/* Elimina spazi superflui */
$cognome   = trim($cognome);
$commento  = trim($commento);
$matricola = trim($matricola);

/* la variabile "errore" indica se tra i dati abbiamo
   trovato un errore */
$errore=FALSE;

/* verifica che almeno uno tra matricola e cognome siano
   non vuoti */
if (!$cognome && !$matricola) {
    echo "<b>Errore! Almeno uno tra nome e matricola
              devono essere specificati!</b><br />";
$errore=TRUE;};
```

Gestore della Form: Controllo dei parametri


```
/* verifica che la matricola sia numerica */
if ($matricola && ! ctype_digit($matricola)) {
    echo "<b>Errore! Matricola deve essere numerica!</b><br />";
$errore=TRUE;
};

    /* ... e il cognome alfabetico */
if (! preg_match('/^[a-zA-Z]*$/', $cognome)) {
    echo "<b>Errore! Cognome deve essere alfabetico!</b><br />";
$errore=TRUE;
};

if (!$errore) {
    /* Inizia la costruzione della query */
   ...
};
```

Form e gestore in un unico script

- Soluzione diversa: form e gestore in un unico script
- Necessità di distinguere il caso in cui si deve mostrare la form e quello in cui si deve elaborarla
  - **Name/Value** per **all'input submit**
  - Uso di **self**

```
if (isset($_REQUEST['submit']))
    # trattamento dei parametri
else {
    # visualizza la form
    echo<<<END
    <form method="post"
          action="$_SERVER['PHP_SELF']">
    ...
    ...
    <input type="submit" name="submit"
                          value = "submit">
    END;
};
```

Form e gestore in un unico script

- Es. mantenere il valore dei campi in caso di errore

```
/* verifica se si esegue lo script per la prima volta:
   in questo caso bisogna presentare la form  */
$first= ! isset($_REQUEST["submit"]);

if (! $first) {
      /* recupera i dati della form */  $nome=$_REQUEST
   ['nome'];
   $cognome=$_REQUEST['cognome'];

      /* e verifica se il nome soddisfa i criteri
        (alfabetico)*/
      $errore_nome = ! preg_match("/^[a-zA-Z]*$/",$nome);
};

...
```

Form e gestore in un unico script


```
if ($first || $errore_nome) {
/* se lo script si esegue per la prima volta oppure i dati
   sono errati mostra la form ed eventualmente segnala
   l'errore */

$self = $_SERVER['PHP_SELF']; /* nome dello script corrente */
echo<<<END
<form method="post" action="$self">
    Nome    <input type="text" name="nome" value="$nome" />
    Cognome <input type="text" name="cognome" value="$cognome"/>
   <input type="submit" name="submit" value="Invia" />
    <input type="reset" value="Cancella" />
</form>
END;

/* e se c'era un errore nei dati lo segnala */
...
```

Form e gestore in un unico script


```
/* e se c'era un errore nei dati lo segnala */
    if ($errore_nome)
        echo "<b>Errore nel nome!!! Deve essere alfabetico!</b>";
}

else { /* ! $first && ! $errore_nome */  /* se invece i dati ci
sono e sono corretti, li elabora */
    echo "elaboro i dati <br />";
    echo "Nome: $nome<br />";
    echo "Cognome: $cognome<br />";
};
```

Invio di mail

- Da uno script php possiamo inviare una email con la funzione

```
mail (to, subject, body, headers)
```

- oppure

```
mail (to, subject, body, headers)
```

```
$to      = 'conti@math.unipd.it';
$subject = 'Ieri';
$message = 'Ciao, come va?';
$headers = 'From: webmaster@bd.com' . "\r\n".
     'Reply-To: webmaster@bd.com';

mail($to, $subject, $message, $headers);
```

Upload di file: form

- Occorre utilizzare una form HTML adeguata ...

```
<form enctype="multipart/form-data"
      action="Upload.php" method="post">
   ...
   Nome del file:
   <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
   <input type="file" name="myfile" />
   <br />

   <input type="submit" name="submit" value="Invia" />
   ...
</form>
```

Upload.html

Upload di file

- I dati relativi ai file inviati sono accessibili mediante la variabile `$_FILES`
  - `$_FILES['myfile']['name']`
    Il nome del file sulla macchina di origine
  - `$_FILES['myfile]['type']`
    Il mime type del file (es. "image/gif"). Non sempre affidabile.
  - `$_FILES['myfile']['size']`
    La dimensione del file
  - `$_FILES['myfile']['tmp_name']`
    Nome temporaneo del file sul server.
  - `$_FILES['myfile']['error']`
    Codice di errore associato all'upload

Upload di file: recupero dei dati


```php
/* dimensione massima di upload */
define(DIM_MAX,30000);

/* directory locale per la memorizzazione */
$localdir="upload";

/* informazioni sul file */
$error= $_FILES["myfile"]["error"];    /* codice di errore */
$size = $_FILES["myfile"]["size"];     /* dimensione       */
$type = $_FILES["myfile"]["type"];     /* mime-type        */
$name = $_FILES["myfile"]["name"];     /* nome sul client  */
$tmp  = $_FILES["myfile"]["tmp_name"]; /* nome sul server  */
```

Upload di file: error check


```
/* verif. se il file uploaded ha caratteristiche appropriate */

/* si e` verificato un errore ? */
if ($error != UPLOAD_ERR_OK)  {
    echo "Errore di upload. Codice: $error<br />";
}

/* tipo corretto ? */
elseif (($type != "image/gif")    && ($type != "image/jpeg")) {
    echo "Errore: File di tipo $type. Errato! <br />";
}

/* dimensione */
elseif ($size > DIM_MAX) {
    echo "Errore: File troppo grande ($size bytes)! <br />";
}
else {
    /* Tutto ok! Mostra i dati */
```

Upload di file: recupera il file:


```
  /* recupera il file */

  /* se non è già presente ... */
if (file_exists($locadir . "/" . $name))
    {
    echo $name . " gia` presente.";
    }
   else
    {
       /* lo porta a destinazione ... */       move_uploaded_file
    ($tmp, $localdir . "/" . $name);
    echo "Memorizzato in: " . $localdir . "/" . $name;
    }
};
```

Upload di file

- Quando si gestiscono operazioni invasive come un upload, può essere opportuno tenere un logfile
- Supporto in `PHP` per operare sul log di sistema (**`/var/log/system.log`**)
- **Esempio**
  - `openlog("Upload.php", LOG_PID, LOG_LOCAL0)`
  - `...`
  - `syslog(LOG_WARNING,`
    `"Upload di $name da parte di " . $_SERVER["SERVER_ADDR"]);`
  - `closelog()`