USO DI PHP E DATABASE

Inclusione di files

- Conviene che le librerie con le funzioni vengano scritte su files separati e incluse nella pagina web.
  - Due modalità; include e require: include se non trova il file segnala e prosegue, require si interrompe
  - require "nomefile.php"
  - Esiste una versione per assicurarsi che l'inclusione non venga ripetuta inutilmente:

    include_once, require_once
  - Il file incluso è un normale file php, in cui si possono mescolare tag html e php:

    <?php

    .....

    ?> testo <? php echo '.' ?>

Gestione degli errori

- Errori di sintassi provocano un messaggio sulla pagina web (ma attenzione al file di configurazione php.ini!)

  Parse error: syntax error, unexpected '<' in **/Library/PostgreSQL/EnterpriseDB-ApachePhp/apache/www/a/index.php** on line **10**

- Se l'errore avviene durante la compilazione non viene prodotto nessun risultato
- Si può definire il livello di errore mostrato nel file php.ini oppure attraverso i comandi da mettere nel file php:

  error_reporting(E_ALL & ~E_NOTICE);

  (E_ERROR, E_WARNING, E_PARSE, E_NOTICE, ...)

- Si può forzare la terminazione con un errore:

  die(messaggio);

  es: die('Impossibile connettersi al database');

- Si usa spesso la notazione:

  comando or die('non posso eseguire il comando');

Accesso a database

- Esistono librerie diverse per accedere ai vari database da PHP
  - Librerie DB-specifiche: pg_connect, ..., mysql_connect, ..., mysqli_connect, ...
  - Libreria indipendente dal DB: PDO (PHP Data Objects)
    Analoga a JDBC e ODBC
- **Usare solo la seconda!**
  - oltre a rendere più facile il porting dell'applicazione con DBMS diversi...
  - è più sicura da attacchi di tipo SQL-injection!

Comandi per accedere al database (PDO)

- $dbconn = new PDO('pgsql:host=dblab.dsi.unive.it;port=5432;dbname=utente100','utente100','xxx');
- Si indicano il tipo del dbms, il server, il nome del database, la login e la password
- Viene restituito un oggetto php di tipo PDO, che rappresenta una connessione aperta con un database, con una serie di metodi
- Gestione degli errori attraverso le eccezioni:
  $dbconn -> setAttribute (PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  dopo è possibile gestire l'eccezione, ad esempio per stampare un messaggio di errore:

```
try { $dbconn = new PDO('pgsql:host=localhost;...');
      $dbconn -> setAttribute (PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      ....
} catch (PDOException $e) {
    echo $e->getMessage();
}
```


Comandi SQL

- I comandi possono essere:
  - Stringhe costanti
  - "Prepared statement" (da utilizzare sempre quando il comando deve essere formato con dati inseriti dall'utente, calcolati, ecc.), cioè quando non è una stringa costante.
- Esempio con stringa:
  $dbconn -> exec('delete from questioni');
  $dbconn -> query('select count(*) from questioni');
- Esempio con prepared statement:

```
$dato = ... (espressione che calcola il dato, ad es. $_POST["campo_di_form"];)
$statement = $dbconn->prepare('insert into tabella values(?)');
$statement->execute(array($dato));
```

Operatori per SQL

- PDO->exec(stringa) esegue un comando di modifica e restituisce il numero delle righe modificate (anche 0)
- PDO->query(stringa) esegue una select e restituisce un "Result Set" (cursore) sotto forma di un oggetto di tipo PDOStatement
- PDO->prepare(string) "prepara", compila un comando di query e/o di modifica e restituisce un PDOStatement che verrà eseguito successivamente attraverso il metodo execute
- PDOStatement->execute(array di parametri) esegue un comando di query e/o modifica e restituisce un valore boolean per indicare il risultato
- Successivamente occorrerà fare dei "fetch" per prendere il risultato dal PDOStatement
- Quindi bisogna fare attenzione: un oggetto PDOStatement può "essere" sia una query che deve essere ancora eseguita, sia un result set di una query già eseguita

Prendere valori da un result set - foreach

```
<table border=1>
    <?php foreach($dbconn->query('select * from questioni') as $record) { ?>
    <tr>
      <td> <?php echo $record['domanda'] ?></td>
   </tr>
  <?php } ?>
</table>

alternativamente:

<table border=1>
    <?php $statement = $dbconn->prepare('select * from questioni');
          $statement->execute();
          foreach($statement as $record) { ?>
    <tr>
     <td> <?php echo $record['domanda'] ?> </td>
    </tr>
  <?php } ?>
</table>
```

Vari modi di prendere valori da un result set

- Result set = statement già eseguito
- Modalità di accesso: parametro che può essere dato agli operatori "Fetch". Può essere specificato come viene restituito ogni record:

  PDO::FETCH_NUM array numerico (con indice 0)

  PDO::FETCH_ASSOC array associativo nomecampo => valore

  PDO::FETCH_BOTH (default) sia numerico che associativo

  + modalità per ritornare i record come oggetti
- La modalità può essere il secondo parametro di PDO->query, oppure
- il primo parametro di fetch o fetchAll, oppure
- può essere associato ad uno statement esplicitamente con setFetchMode()

Modalità diretta

- Non si specifica nulla, si può usare il result set direttamente come iteratore:

```
- $statement = $dbconn -> prepare("select * from questioni");
$statement->execute();
echo "<table border=1>";
foreach ($statement as $record) {
  echo "<tr>";
  foreach ($record as $val) {
    echo "<td> $val </td>";
  }
  echo "</tr>";
}
echo "<table>";
```

fetch

- Accesso della riga successiva (come i cursori in altri linguaggi)
- Si usa con un ciclo while

```
$statement = $dbconn -> prepare("select * from questioni");

$statement->execute();
echo "<table border=1>";
while ($record = $statement->fetch(PDO::FETCH_NUM)) {
  echo "<tr>";
  foreach ($record as $val) {
    echo "<td> $val </td>";
  }
  echo "</tr>";
}
echo "<table>";
```

- Non c'è bisogno di "chiudere" esplicitamente il result set a meno che non si voglia interrompere la scansione prima della fine (in quel caso è obbligatorio usare PDOStatement->closeCursor()

fetchAll

- fetchAll(modalità)
  - restituisce un array (numerico) contenente tutti i record in memoria (molto costoso in termini di spazio se i record sono molti)

```
$s = $dbconn -> prepare("select * from questioni");

$s->execute();
$resultSet = $s->fetchAll(PDO::FETCH_NUM);
echo "<table border=1>";
foreach ($resultSet as $record) {
  echo "<tr>";
  foreach ($record as $val) {
    echo "<td> $val </td>";
  }
  echo "</tr>";
}
echo "<table>";
```