Подготвени заявки в MySQLi (Prepared Statements)
Изпълнение на транзакции с MySQLi.
Операции чрез ODBC без DSN върху Access БД

Подготвени заявки в MySQLi (Prepared Statements)

1. Подготвяне на заявка (Prepared Statements) и прикрепени параметри към нея (Bound Parameters)

Една „подготвена заявка" (prepared statement) е възможност за многократно използване на SQL заявка, с различни параметри, което повишава ефикасността на самата заявка.
Подготвените заявки (Prepared statements) обикновено работят по следния начин:

1. Prepare (Подготовка): Създава се SQL шаблон на заявка (на изречение) и се изпраща към БД. Някои стойности остават неспецифицирани, наричат се параметри (всеки, указан с ? – при повече от един ?,?,?...). Пример: INSERT INTO MyGuests VALUES(?, ?, ?)
2. Базата от данни анализира (parses), компилатора и оптимизира SQL шаблона, съхранявайки получения резултат (съхранен SQL шаблон), без да го изпълнява.
3. Execute (Изпълнение): в по-късен момент, приложението свързва (прикрепя - binds) стойностите към параметрите, а базата данни изпълнява получения, съгласно т.2 резултат (съхранения SQL шаблон). Заявката се изпълнява толкова пъти, колкото се поиска, с различни стойности, прикрепени към параметрите .

В сравнение с изпълнението на SQL заявки директно, подготвените заявки имат две основни предимства: въпреки, че заявката се изпълнява няколко пъти, времето за анализ се намалява, тъй като това се прави само веднъж. • чрез обвързване на параметри се минимизира трафика към сървъра, тъй като се изпращат само параметрите, а не на цялата заявка• подготвените заявки (Prepared statements) са много полезни и срещу SQL инжекции, оригиналният шаблон на заявката не идва отвън, следователно SQL инжекция не може да се случи.
Аргументите могат да са от следните типове:

- i - integer
- d - double
- s - string
- b - BLOB

SQL injection

- *SQL injection* – внедряване в SQL-кода (англ. *SQL injection*) — един от разпространените способи за взлом в сайтовете и програмите, работещи с бази от данни, основан на внедряване в заявката (запитването) на произволен SQL-код.
- *SQL injection-а*, в зависимост от типа на използваното СУБД и условията на внедряване, може да даде възможност на атакуващия да изпълни произволно запитване към БД (*например, да прочете съдържанието на произволна таблица, да изтрие, измени или добави данни*), да получи възможност за четене или запис в локалните файлове и да изпълни произволна команда на атакуемия сървър.
- Атака от тип *SQL injection* може да бъде възможна заради некоректна обработка на входните данни, използвани в SQL-заявките. Разработчика, работещ с БД, трябва да знае за тази уязвимост и да взема мерки за справяне с проблема.

2. Подготвени заявки в MySQLi – примери

Пример1.

Пример: MySQLi позволява да се създадат еднократно изрази, след което многократно да се използват, с различни параметри. Така се позволява да се разделят променливите от заявката.

```
<?php
$con = new mysqli('localhost', 'root', '', 'universitet');
//$con = mysqli_connect('localhost', 'root', '','universitet');
/* проверяваме конекцията */
if ($con->connect_errno) // if (mysqli_connect_errno())
{
   echo 'Could not connect: ' . $con->connect_error;
   exit();
}
$name = "KST"; /* създаваме израз */
if ($stmt = $con->prepare("SELECT id FROM universitet.specialnosti WHERE ime=?")) {
   /* свързваме с параметър */
   $stmt->bind_param("s", $name);
   /* изпълняваме заявката */
   $stmt->execute();
   /* прикрепяме резултата*/
   $stmt->bind_result($id);
   /* избираме значение */
   $stmt->fetch();
   echo "$name is with id $id";
   $stmt->close();
}
/* затваряме съединението */
$con->close();
//mysqli_close($con);
?>
```

Резултат: KST is with id 2

Пример2.

1. Създаване на БД myDB и таблица MyGuests в нея:

```
<html>
   <head>
      <meta charset="UTF-8">
      <title></title>
   </head>
   <body>
     <?php
$servername = "localhost";
$username = "root";
$password = "";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
```

```
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
   echo "Database created successfully";
} else {
   echo "Error creating database: " . $conn->error;
}
$sql = "CREATE TABLE myDB.MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
   echo "Table MyGuests created successfully";
} else {
   echo "Error creating table: " . $conn->error;
}
$conn->close();
?>
   </body>
</html>
```

2. **Следващият код използва подготвени заявки и прикрепя параметри към тях (MySQLi) - http://www.w3schools.com/php/php_mysql_create_table.asp:**

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
```

```
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
?>
```

Резултат:
New records created successfully

Използвана литература:
http://www.w3schools.com/php/php_mysql_prepared_statements.asp
http://www.w3schools.com/php/php_mysql_connect.asp
http://www.w3schools.com/php/func_mysqli_rollback.asp

Изпълнение на MySQL - базирани транзакции с MySQLi

Както знаете, transaction е просто блок от SQL команди, които трябва да бъдат изпълнени в режим "всичко или нищо", тъй като обикновено те са взаимно зависими една от друга. Те са станаха възможни за изпълнение с появата на MySQL Improved, от PHP 5.x нагоре. Една транзакция се счита за успешна само ако всичките команди са изпълнени успешно. Неуспех на която и да е команда предизвиква връщане назад на системата ("rolled back"), за да не се „счупи“ системата.
Добър пример за това е трансфер на пари между две банкови сметки. На ниво на базата данни, такъв трансфер включва две стъпки: първо, изваждане на трансферната сума от баланса на сметката - източник и след това да трансферната сума към баланса на целевата сметка. Ако възникне грешка във втората стъпка, следва първата стъпка да бъде върната за да се избегне несъответствие (и тълпи от ядосани клиенти). Една сигурна откъм транзакции система автоматично ще изпълнява този обрат към изходното състояние на система.
Повечето бази от данни изпълняват това посредством смесица от SQL команди:

- START TRANSACTION - маркира началото на нова транзакция. Следват няколко SQL команди.
- COMMIT команда маркира края на транзакцията и сигнализира, че всички направени промени ще се реализират.
- ROLLBACK команда маркира края на транзакцията и сигнализира, че всички направени промени ще се отменят.

Обобщен пример:

```
<?php
// съединение с базата
$dbh = mysqli_connect($host, $user, $pass, $db);

// изключване на автоматичното реализиране commit
mysqli_autocommit($dbh, FALSE);

// изпълнение на query 1
$result = mysqli_query($dbh, $query1);
if ($result !== TRUE) {
   mysqli_rollback($dbh);  // ако възникне грешка, roll back transaction
}

// run query 2
$result = mysqli_query($dbh, $query2); if ($result !== TRUE) {
   mysqli_rollback($dbh);  // ако възникне грешка, roll back transaction
}
// и т.н....
// ако няма грешки, commit transaction
mysqli_commit($dbh);
// затваряне на конекцията
mysqli_close($dbh);
?>
```

Работен пример

За да видите как работи на практика транзакцията, нека се върнем към примера за банков превод, обсъден по-рано. Да предположим, че вашата задача е да се изгради просто уеб приложение, което да позволи на потребителите да прехвърлят пари между банковите си сметки. Нека приемем, че сметките на отделните потребители се съхраняват в база данни MySQL, която изглежда като този:

```
mysql> SELECT * FROM accounts;
```

```
+----+------------+---------+
| id | label      | balance |
+----+------------+---------+
|  1 | Savings #1 |    1000 |
|  2 | Current #1 |    2000 |
|  3 | Current #2 |    3000 |
+----+------------+---------+
```
3 rows in set (0.34 sec)

Сега е необходимо да се изгради прост интерфейс, който позволява на потребителите да въведат сума пари в брой и да я прехвърлят от една сметка в друга. „Действителната сделка" (транзакция) ще се извършва с помощта на 2 UPDATE команди, едната сума теглеща от сметката на източника, и другата - кредитиране на целевата сметка. Като се има предвид, че всичко, което правим, е прехвърляне на пари между сметки, общия наличен баланс във всички сметки ($ 6000) трябва да остане постоянен по всяко време.

1. **Създаване на БД**

Процедурен интерфейс:

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = '';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(!$conn )
{  die('Could not connect: ' . mysql_error());}
echo 'Connected successfully';
$sql = 'CREATE Database test';
$retval = mysql_query( $sql, $conn );
if(! $retval )
{  die('Could not create database: ' . mysql_error());}
echo "Database test created successfully\n";
$sql ="CREATE TABLE `accounts` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `label` VARCHAR(20) NOT NULL,
  `balance` INT NOT NULL,
  PRIMARY KEY  (`id`))";
mysql_select_db('test');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
  die('Could not create table: ' . mysql_error());
}
echo "Table accounts created successfully\n";
mysql_close($conn);
?>
```

ОО интерфейс:

```
<?php
// connect  to the MySQL server
$conn = new mysqli('localhost', 'root', '');
// check connection
If ($conn->connect_errno)            // if (mysqli_connect_errno())
```

```
{   exit('Connect failed: '. $conn->connect_error);
      //mysqli_connect_error()
}     // sql query with CREATE DATABASE
$sql = "CREATE DATABASE test";
// Performs the $sql query on the server to create the database
if ($conn->query($sql) === TRUE)
{  echo 'Database "test" successfully created<BR>';}
else { echo 'Error: '. $conn->error;}
$sql = "CREATE TABLE test.accounts"
      . "(id int not null auto_increment, label VARCHAR(20) NOT NULL, "
      . "balance INT NOT NULL,  PRIMARY KEY  (id))";
if ($conn->query($sql) === TRUE)
{  echo 'Table " accounts " successfully created';}
else
{ echo 'Error: '. $conn->error;}
$conn->close();
?>
```

```
Database "test" successfully created
Table " accounts " successfully created
```

2. Запълване с данни

Процедурен интерфейс:

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = '';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
  die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
mysql_select_db('test');
$sql = 'INSERT INTO accounts '.
      '(id,label, balance) '.
      'VALUES ( "1", " Savings #1", 1000)';
/* $sql = 'INSERT INTO accounts '.
      '(id,label, balance) '.
      'VALUES ( "2", " Current #1", 2000)';
$sql = 'INSERT INTO accounts '.
      '(id,label, balance) '.
      'VALUES ( "3", " Current #2", 3000)' ;  */
if (!mysql_query($sql,$conn))
  {
  die('Error: ' . mysql_error());
  }
echo "1 record added";
mysql_close($conn);
?>
```

ОО интерфейс:

```
<?php
```

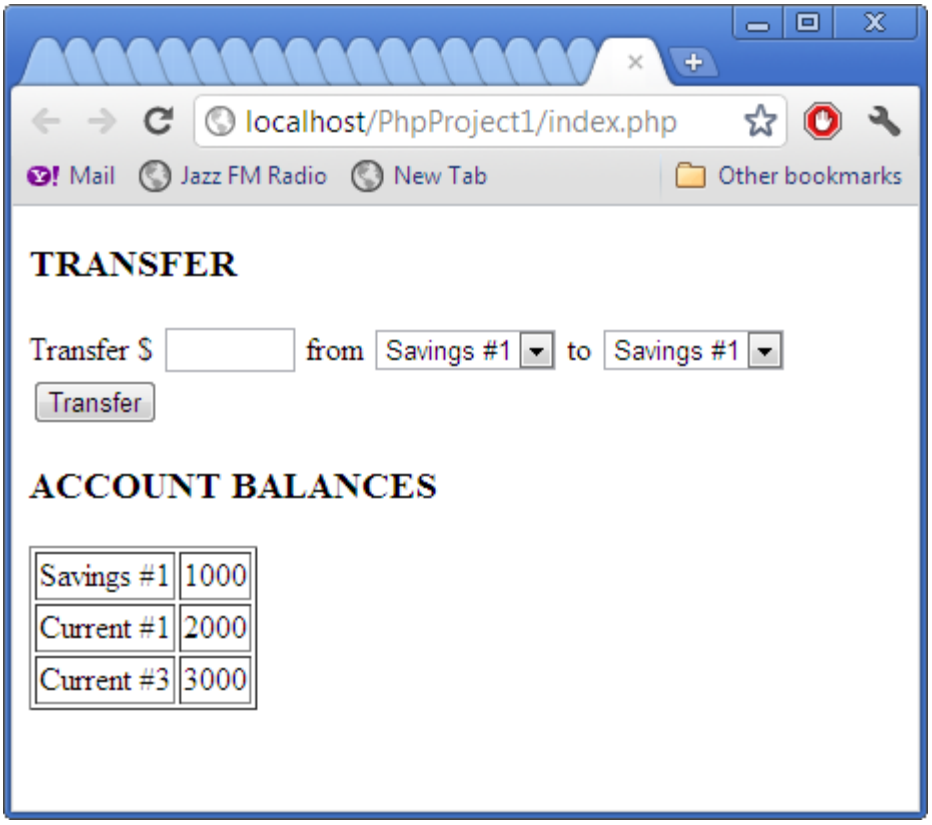
```
$dbhost = 'localhost'; $dbuser = 'root'; $dbpass = ''; $db="test";
//$conn = mysql_connect($dbhost, $dbuser, $dbpass);
$conn = new mysqli('localhost', 'root', '', $db);
If ($conn->connect_errno)            // if (mysqli_connect_errno())
{   exit('Connect failed: '. $conn->connect_error);
                                    //mysqli_connect_error()
}
echo 'Connected successfully';
/*$sql = 'INSERT INTO accounts '.
      '(id,label, balance) '.
      'VALUES ( "1", " Savings #1", 1000)'; */
$sql = 'INSERT INTO accounts '.
      '(id,label, balance) '.
      'VALUES ( "2", " Current #1", 2000)';
/*$sql = 'INSERT INTO accounts '.
      '(id,label, balance) '.
      'VALUES ( "3", " Current #2", 3000)' ;  */
if ($conn->query($sql) === TRUE)
{  echo '1 record added ';}
else
{ echo 'Error: '. $conn->error;}
$conn->close();
?>
```

3. **Изпълнение на транзакцията**

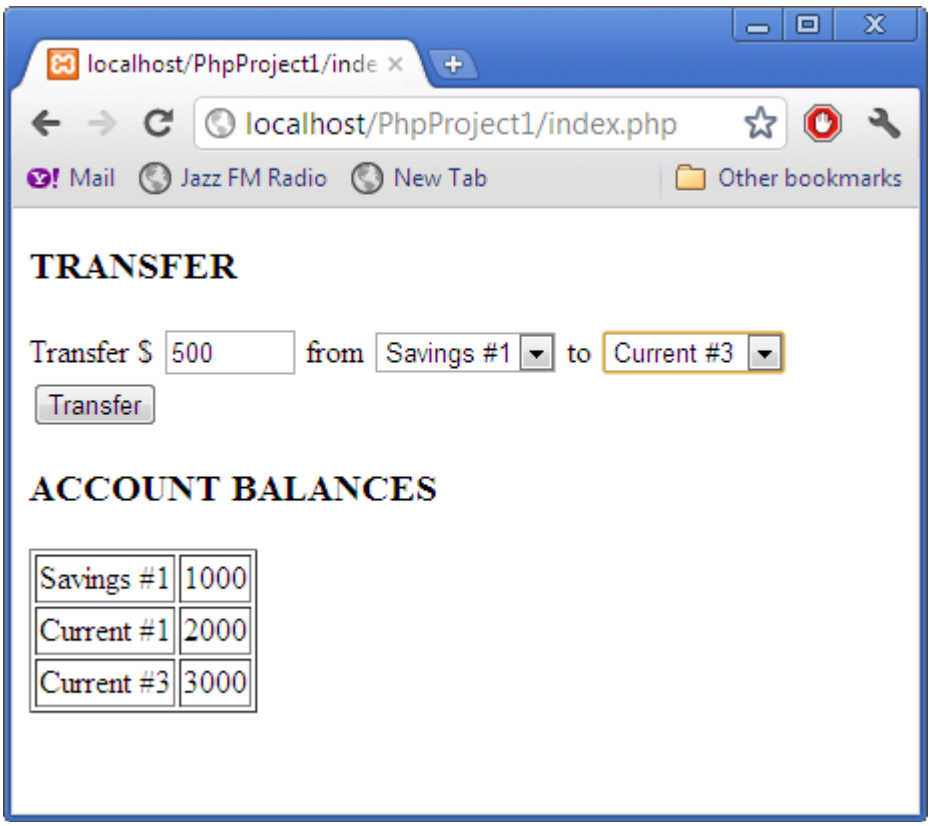
Процедурен интерфейс:

```
<?php
// connect to database
$dbh = mysqli_connect("localhost", "root", "", "test") or die("Cannot connect");

// turn off auto-commit
mysqli_autocommit($dbh, FALSE);
// look for a transfer
if(isset($_POST["submit"])&& is_numeric($_POST['amt']) ){
   // add $$ to target account
   $result = mysqli_query($dbh, "UPDATE accounts SET balance = balance + " . $_POST['amt'] . "
WHERE id = " . $_POST['to']);
   if ($result !== TRUE) {
 mysqli_rollback($dbh);  // if error, roll back transaction
   }
   // subtract $$ from source account
$result = mysqli_query($dbh, "UPDATE accounts SET balance = balance - " . $_POST['amt'] . " WHERE
id = " . $_POST['from']);
   if ($result !== TRUE) {
      mysqli_rollback($dbh);  // if error, roll back transaction
   }
 // assuming no errors, commit transaction
   mysqli_commit($dbh);
}
// get account balances
// save in array, use to generate form
$result = mysqli_query($dbh, "SELECT * FROM accounts");
```

```
while ($row = mysqli_fetch_assoc($result)) {
   $accounts[] = $row;}
// close connection
mysqli_close($dbh);
?>
<html>
<head></head>
<body>

<h3>TRANSFER</h3>
<form action="<?php $_PHP_SELF ; ?>" method="post">
Transfer $ <input type="text" name="amt" size="5"> from
<select name="from">
<?php
foreach ($accounts as $a) {
   echo "<option value=\"" . $a['id'] . "\">" . $a['label'] . "</option>";
}
?>
</select>
to
<select name="to">
<?php
foreach ($accounts as $a) {
   echo "<option value=\"" . $a['id'] . "\">" . $a['label'] . "</option>";
}
?>
</select>
<input type="submit" name="submit" value="Transfer">
</form>
<h3>ACCOUNT BALANCES</h3>
<table border=1>
<?php
foreach ($accounts as $a) {
   echo "<tr><td>" . $a['label'] . "</td><td>" . $a['balance'] . "</td></tr>";
}
?>
</table>
</body>
</html>
```



След Transfer:



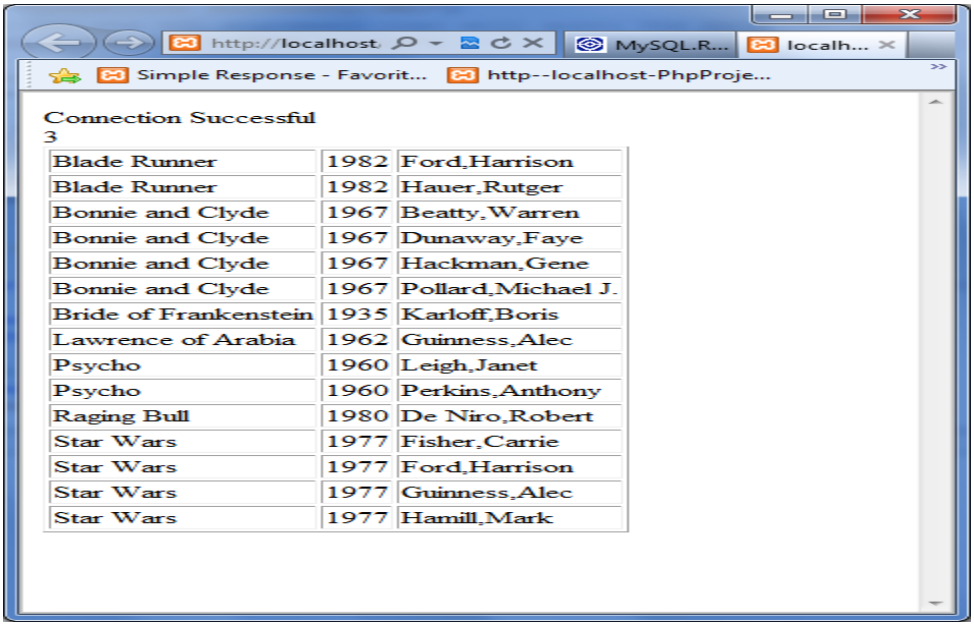
Както можете да видите, скриптът започва със свързване с БД и изключване на automatic commits. След това се изпълнява SELECT query за намиране на текущите баланси за всички акаунти, следва конструиране на форма с drop-down interface за селектиране на акаунт източник и акаунт – цел, за транзакцията.

ОО интерфейс: /за самостоятелна разработка/

Операции чрез ODBC (Open Database Connectivity – стандартизиран програмен интерфейс на Microsoft) без DSN (Data Source Name) с Access БД

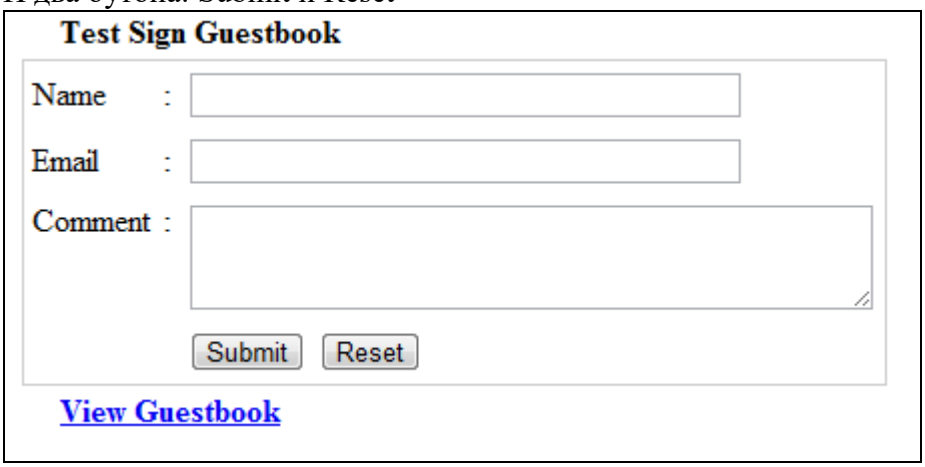
Пример:

```
<?php
//create an instance of the  ADO connection object
$conn = new COM ("ADODB.Connection")  or die("Cannot start ADO");

//define connection string, specify database driver
$connStr = "PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source= E:\\movies.mdb";
$conn->open($connStr); //Open the connection to the database
echo "Connection Successful<BR>";

//declare the SQL statement that will query the database
$query = "SELECT * FROM Actors";

//execute the SQL statement and return records
$rs = $conn->execute($query);

$num_columns = $rs->Fields->Count();
echo $num_columns . "<br>";

for ($i=0; $i < $num_columns; $i++) {
   $fld[$i] = $rs->Fields($i);
}

echo "<table border>";
while (!$rs->EOF)  //carry on looping through while there are records
{  echo "<tr>";
   for ($i=0; $i < $num_columns; $i++) {
      echo "<td>" . $fld[$i]->value . "</td>";
   }
   echo "</tr>";
   $rs->MoveNext(); //move on to the next record
}
echo "</table>";
//close the connection and recordset objects freeing up resources
$rs->Close();
$conn->Close();
$rs = null;
$conn = null;
?>
```

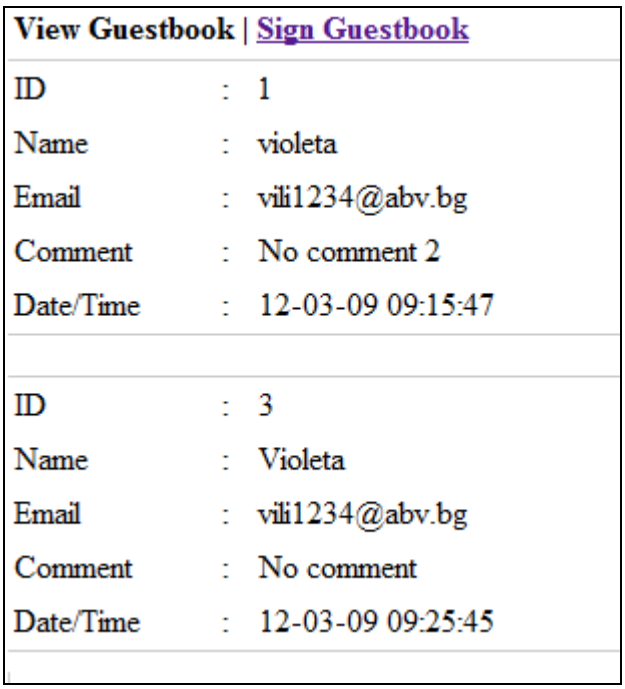


Задача (за самостоятелна работа): Да се създаде елементарна книга за гости.

Интерфейсът трябва да включва следната примерна стартова форма с 3 полета: Name, Email, Comment
И два бутона: Submit и Reset



При натискане на връзката View Gestbook се разпечатва списък на посетителите, в следния примерен формат:



При натискане на връзката Sign Gestbook се връщаме обратно към стартовата форма (Test Sign Guestbook).

```
CREATE TABLE `web_members` (
`id` int(4) NOT NULL auto_increment,
`name` varchar(65) NOT NULL default '',
`lastname` varchar(65) NOT NULL default '',
`email` varchar(65) NOT NULL default '',
PRIMARY KEY (`id`)
) TYPE=MyISAM AUTO_INCREMENT=1 ;
```

Проект: GestBook (Примерна реализация)

Създаване 3 php файла:

- Index.php (вместо guestbook.php)
- addguestbook.php
- viewguestbook.php

Index.php

```
<table width="400" border="0" align="center" cellpadding="3" cellspacing="0">
<tr>
<td><strong>Test Sign Guestbook </strong></td>
</tr>
</table>
<table width="400" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#CCCCCC">
<tr>
<form id="form1" name="form1" method="post" action="addguestbook.php">
<td>
<table width="400" border="0" cellpadding="3" cellspacing="1" bgcolor="#FFFFFF">
<tr>
<td width="117">Name</td>
<td width="14">:</td>
<td width="357"><input name="name" type="text" id="name" size="40" /></td>
```

```
</tr>
<tr>
<td>Email</td>
<td>:</td>
<td><input name="email" type="text" id="email" size="40" /></td>
</tr>
<tr>
<td valign="top">Comment</td>
<td valign="top">:</td>
<td><textarea name="comment" cols="40" rows="3" id="comment"></textarea></td>
</tr>
<tr>
<td> </td>
<td> </td>
<td><input type="submit" name="Submit" value="Submit" /> <input type="reset" name="Submit2"
value="Reset" /></td>
</tr>
</table>
</td>
</form>
</tr>
</table>
<table width="400" border="0" align="center" cellpadding="3" cellspacing="0">
<tr>
<td><strong><a href="viewguestbook.php">View Guestbook</a> </strong></td>
</tr>
</table>
```

addguestbook.php

```
<?php
$host="localhost"; // Host name
$username="peter"; // Mysql username
$password=""; // Mysql password
$db_name="my_db"; // Database name
$tbl_name="guestbook"; // Table name
// Connect to server and select database.
mysql_connect("$host", "$username", "$password")or die("cannot connect server ");
mysql_select_db("$db_name")or die("cannot select DB");
$datetime=date("y-m-d h:i:s"); //date time
$name=$_REQUEST["name"];
$email=$_REQUEST["email"];
$comment=$_REQUEST["comment"];
$sql="INSERT INTO $tbl_name(name, email, comment, datetime)VALUES('$name', '$email', '$comment',
'$datetime')";
$result=mysql_query($sql);
//check if query successful
if($result){
echo "Successful";
echo "<BR>";
echo "<a href='viewguestbook.php'>View guestbook</a>"; // link to view guestbook page
}
else {echo "ERROR";}
```

```
mysql_close();
?>
```

Viewguestbook.php

```
<table width="400" border="0" align="center" cellpadding="3" cellspacing="0">
<tr>
<td><strong>View Guestbook | <a href="index.php">Sign Guestbook</a> </strong></td>
</tr>
</table>
<br>
<?php
$host="localhost"; // Host name
$username="peter"; // Mysql username
$password=""; // Mysql password
$db_name="my_db"; // Database name
$tbl_name="guestbook"; // Table name
// Connect to server and select database.
mysql_connect("$host", "$username", "$password")or die("cannot connect server ");
mysql_select_db("$db_name")or die("cannot select DB");
$sql="SELECT * FROM $tbl_name";
$result=mysql_query($sql);
while($rows=mysql_fetch_array($result)){
?>
<table width="400" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#CCCCCC">
<tr>
<td><table width="400" border="0" cellpadding="3" cellspacing="1" bgcolor="#FFFFFF">
<tr>
<td>ID</td>
<td>:</td>
<td><? echo $rows['id']; ?></td>
</tr>
<tr>
<td width="117">Name</td>
<td width="14">:</td>
<td width="357"><? echo $rows['name']; ?></td>
</tr>
<tr>
<td>Email</td>
<td>:</td>
<td><? echo $rows['email']; ?></td>
</tr>
<tr>
<td valign="top">Comment</td>
<td valign="top">:</td>
<td><? echo $rows['comment']; ?></td>
</tr>
<tr>
<td valign="top">Date/Time </td>
<td valign="top">:</td>
<td><? echo $rows['datetime']; ?></td>
</tr>
</table></td>
```

```
</tr>
</table>
<BR>
<? }
mysql_close(); //close database?>
```