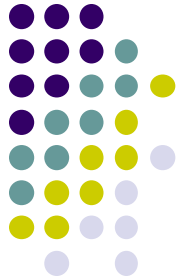
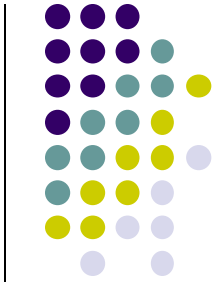


PHP и MySQLi

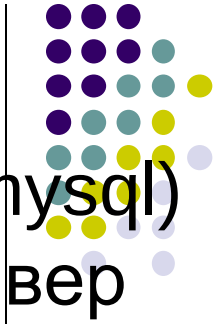
Виолета Божикова

MySQL – Основни характеристики



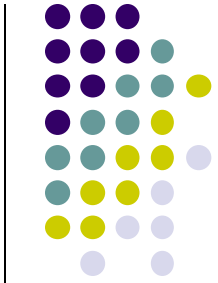
- Както вече стана въпрос, достъпът до MySQL Бази от данни се осъществява чрез драйверите:
  - **...\ext\php_mysql.dll,**
  - **...\ext\php_ mysqli.dll и**
  - **...\ext\php_pdo_mysql.dll**
- Разширението **mysqli** (т.е **mysql improved**) и **pdo (PHP Data Objects)** са изцяло разработени за 5-та версия на PHP - за да се предостави на програмистите достъп до новите възможности след MySQL 4.1.

Да припомни какво са MySQLi и PDO



- **php_mysqli.dll** (mysql improved/подобрен mysql) се явява подобрена версия на стария драйвер php_mysql.dll.
- **Защо е бил създаден?** Защото **php_mysql**
  - **не поддържа транзакции**,
  - **няма обектен интерфейс**,
  - **налице е уязвимост към подмяна за стойности в заявката**.
- **php_mysqli.dll** предоставя както процедурен така и ОО интерфейс.
- **php_pdo_mysql.dll** предоставя само ОО интерфейс.

Кой драйвер да използваме, ако ни трябва ОО интерфейс?

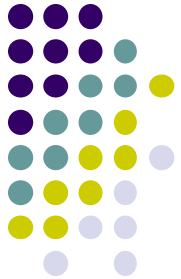


- И двата (**php_mysqli и php_pdo_mysql**) поддържат **Prepared Statements**, които предпазват от **SQL injection**, което е много важно за **сигурността** на web приложенията.
- Процедурният интерфейс, осигуряван чрез php_mysqli.dll е аналогичен на този на php_mysql.dll, тоест много лесно може **да се реновира приложение**, използващо php_mysql.dll.

Но:

- PDO разработи с 12 различни БД, докато MySQLi – само с MySQL БД.
- Тоест, ако вашият проект трябва да използва различни видове БД, то изберете PDO. Така ще трябва да промените само connection string-а си, а с MySQLi – трябва да пренапишете кода си. Ако ви трябва съвместимост с предни версии и работа единствено с MySQL БД – то MySQLi.

За “prepared statements”



- Казахме, че mysqli и PDO позволяват да се създадат еднократно заявки към БД – “**prepared statements”**, които могат многократно да се използват, с различни параметри. Така се позволява да се разделят променливите от заявката.
- Има 3 предимства за използване на така подготвените заявки (изрази):
  - Това е по-безопасно, отколкото да се оперира с данни вътре в израза;
  - Скоростта на изпълнение е по-висока;
  - По бързо се пише код.

MySQLi (Процедурен интерфейс)
функция за създаване на връзка с MySQL сървър

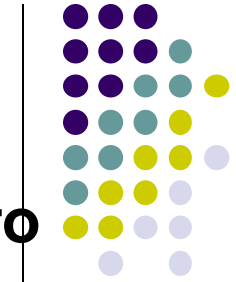
За създаване на връзка с MySQL сървър се функция mysqli_connect().

- Общ вид:

mysqli_connect (string host, string user, string password [,string dbname [, int port [, int socket]]])

- host - име или ip на компютъра на който се изпълнява MySQL сървър. Стойност NULL или localhost означава, че MySQL сървър се изпълнява на същия компютър на който работи приложението.
- user, password – служат за идентификация на потребител.
- dbname – задава подрабиращата се БД при изпълнение на заявки.
- port – номер на порт. Подразбира се 3306.
- socket – задава се при обмен на данни по именован канал за комуникация.

MySQLi (Процедурен интерфейс) - функции mysql_connect_error() и mysqli_connect_errno()

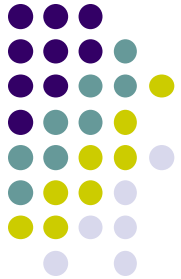


- Функцията връща обект от клас mysqli, който използваме като параметър за заявките. За получаване на информация за грешка се използва функция:

mysql_connect_error(), която връща текст с описание на последната възникнала грешка при установяване на връзка с MySQL; NULL – ако няма грешка!

mysqli_connect_errno() – връща номер на грешката. 0 – ако няма грешка!

- Функцията pconnect() се различава само по това, че след изпълнение на модула по подразбиране връзката със сървъра не се прекратява и при повторно обръщение към сървъра със същите параметри се използва създадената вече връзка.



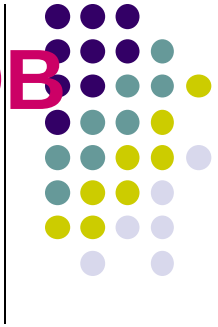
- **Пример 1. Връзка с MySQL/MariaDB сървъра, процедурен интерфейс - mysqli_connect() :**

```
$con = mysqli_connect('localhost', 'root',
   '','universitet');
if (!$con)
  {
  die('Could not connect: ' .
   mysqli_connect_error());
  }
```

- **Пример 2 - за връзка с MySQL/MariaDB сървъра, чрез mysqli драйвър - ОО интерфейс:**

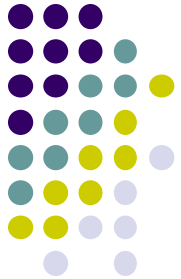
```
$con = new mysqli("localhost", "root", "", "universitet");
If ($con ->connect_errno)
//вместо: if (mysqli_connect_errno())
{
 echo 'Could not connect: ' . $conn->connect_error;
// echo 'Could not connect: ' . mysqli_connect_error();
exit();
}
```

Пример 3 - за връзка с MySQL/MariaDB сървъра, чрез pdo драйвър - ОО интерфейс:



```
$servername = "localhost";$username = "root";$password = "";
try
{
$con = new PDO("mysql:host=$servername;dbname= universitet", $username, $password);
// set the PDO error mode to exception
$con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
 }
catch(PDOException $e)
{  echo "Connection failed: " . $e->getMessage();    }
```

MySQLi (Процедурен интерфейс) - функции за изпълнение на заявки и получаване на записи



- **Функция mysqli_query() – общ вид:**

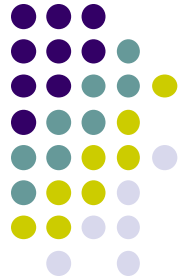
mixed mysqli_query(mysqli link, string query [, int result_mode])

- **link – обект от клас mysqli, получен при създаване на връзка с MySQL server, тоест с функция mysqli_connect().**
- **query – sql команда, тоест низ който се подава на сървъра за изпълнение.**
- **result_mode - определя типа на резултата.**
- **Възможни са 2 стойности:**
  - **MYSQLI_STORE_RESULT (по подразбиране) – за получаване на буферирана извадка от записи**
  - **или MYSQLI_USE_RESULT – за получаване на небуферирана извадка (за голям обем данни)**

Пример: if ($result = mysqli_query($con,"SELECT * FROM universitet.specialnosti", MYSQLI_USE_RESULT)) ...

MySQLi (ОО интерфейс): query(...)

```
if ($result =$con->query("SELECT * FROM universitet.specialnosti",
    MYSQLI_USE_RESULT) ...
```



- Буферираните извадки дават възможност за търсене и позициониране на запис и двупосочно движение на курсора в извадката-резултат.
- Ангажират ОП за съхранение на извадка.

Желателно е накрая на скрипта да се направи освобождаване на паметта, използвана за съхраняване на резултата от заявка:

mysqli_free_result($result);

- Небуферираните извадки не ангажират памет, но дават възможност само за последователно четене на записите.

MySQLi (Процедурен интерфейс) - функции за изпълнение на SQL заявки, без връщане на записи

bool mysqli_real_query(mysqli link, string query)

Аналогична на mysqli_query(), но не връща извадка, а TRUE при успешно изпълнение и FALSE при неуспех.

- link – обект от клас mysqli, получен при създаване на връзка с MySQL server, тоест с функция mysqli_connect().
- query – sql команда, тоест низ който се подава на сървъра за изпълнение.

Функции за получаване на запис от извадката, в масив

Различават се по индексирането на елементите в масива:

mysqli_fetch_row(*result*), където

- result – извадката (обект от клас **mysql_result**), връщана от mysqli_query(): буферирана (тип mysqli_store_result()) или небуферирана (тип mysqli_use_result()).
- връща запис от result, като масив с целочислени индекси, или NULL ако няма повече записи

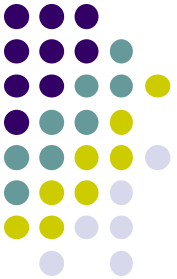
mysqli_fetch_assoc(*result*)

- връща запис от ***result*** като масив с индекси имената на полетата в извадката, или NULL ако няма повече записи за извличане.

Желателно е накрая на скрипта да се направи освобождаване на паметта, използвана за ***result:***

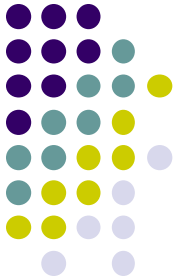
mysqli_free_result(*result*)

Пример1: MySQLi (ОО интерфейс). Създаване на БД “universitet”…



```
<?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 universitet";
// Performs the $sql query on the server to create the database
if ($conn->query($sql) === TRUE)
{  echo 'Database "universitet" successfully created';}
else { echo 'Error: '. $conn->error;}
$conn->close();
?> /*$conn->query($sql) – изпълняване на SQL заявка, връщайки резултата
     като PDO (PHP Data Object). */
```

Пример2: MySQLi (ОО интерфейс). Създаване на таблица “specialnosti”



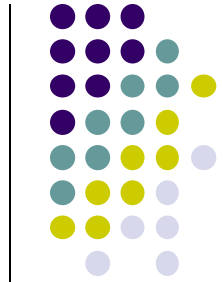
```
<?php
// връзка към MySQL server
$conn = new mysqli('localhost', 'root', '', 'universitet');
// проверка на връзката
if ($conn->connect_errno)      // if (mysqli_connect_errno())
   exit('Connect failed: '. $conn->connect_error);
                                //mysqli_connect_error()
// sql заявка за създаване на TABLE
$sql = "CREATE TABLE specialnosti (id int unsigned not null
    auto_increment, ime varchar(100) not null, primary
    key(id))CHARACTER SET cp1251";
// Изпълнение на $sql query на сървъра за създаване на таблицата
if ($conn->query($sql) === TRUE)
{  echo 'Table "specialnosti" successfully created';}
else
{ echo 'Error: '. $conn->error;}
$conn->close();
?>
```

Резултат: Table "specialnosti" successfully created

Пример3: MySQLi (ОО интерфейс). Добавяне на данни

```
<?php
                // връзка към MySQL server
$con = new mysqli('localhost', 'root', '', 'universitet');
              //$con = mysqli_connect('localhost', 'root', '','universitet');
$con->query("SET NAMES cp1251");
               //mysqli_query($con,"SET NAMES cp1251");
if (!$con)
  {
die('Could not connect: ' . $conn->connect_error);
    //mysqli_connect_error();
  }
$con->query("INSERT INTO specialnosti(ime) VALUES ('K')");
//mysqli_query($con,"INSERT INTO specialnosti(ime) VALUES ('K')");
$con->query("INSERT INTO specialnosti(ime) VALUES ('TTT')");
//mysqli_query($con,"INSERT INTO specialnosti(ime) VALUES ('TTT')");
$con->query("INSERT INTO specialnosti(ime) VALUES ('ETET')");
//mysqli_query($con,"INSERT INTO specialnosti(ime) VALUES ('ETET')");
$con->close();
     //mysqli_close($con);
?>
```

Кодирането на web приложение на ниво на връзката php – MySql:



- Осъществява се чрез SQL оператора **SET NAMES**.
- За целта, веднага след установяване на връзката **php – MySql** се изпълнява заявката:

```
$con->query("SET NAMES cp1251");
```

Или:

```
mysqli_query($con,"SET NAMES cp1251");
```

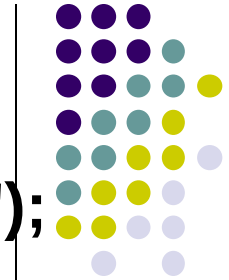
Ако данните ще са на кирилица, използва се кодиране cp1251 (Windows-1251) или UTF-8

Пример 4: MySQLi (ОО интерфейс). Извеждане на данните във вид на таблица.

```
<?php
$con = new mysqli('localhost', 'root', '', 'universitet');
//$con = mysqli_connect('localhost', 'root', '','universitet');
if (!$con)
  {  die('Could not connect: ' . $conn->connect_error);    }
if ($result = $con->query("SELECT * FROM
     universitet.specialnosti", MYSQLI_USE_RESULT))
{ echo "<table border='1'>
<tr> <th>ID</th><th>Ime</th> </tr>";
while($row = $result->fetch_assoc())
//while($row = mysqli_fetch_assoc($result))
{ echo "<tr>";
  echo "<td>" . $row['id'] . "</td>";
  echo "<td>" . $row['ime'] . "</td>";
  echo "</tr>";
  }
echo "</table>";  $con->close();    //mysqli_close($con);
}
?>
```

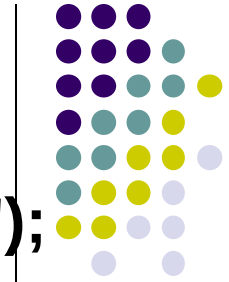
ID	Ime
1	K
2	KST
3	ETET

Пример: Модифициране на данни - mysqli:



```
<?php
$con = new mysqli('localhost', 'root', '', 'universitet');
//$con = mysqli_connect('localhost', 'root',
    '','universitet');
if (!$con)
  {
  die('Could not connect: ' . $conn->connect_error);
  }
echo 'Success... ' . "<br/>";
$con->query("UPDATE specialnosti SET ime = 'KST'
    WHERE id=2");
//mysqli_query($con,"UPDATE specialnosti SET ime =
    'KST' WHERE id=2");
$con->close();
//mysqli_close($con);
?>
```

Пример: Изтриване на записи от таблицата:



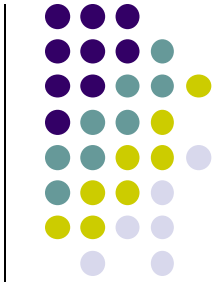
```
<?php
$con = new mysqli('localhost', 'root', '', 'universitet');
//$con = mysqli_connect('localhost', 'root',
    '','universitet');
if (!$con)
  {
  die('Could not connect: ' . $conn->connect_error);
  }
echo 'Success... ' . "<br/>";
//mysqli_query($con,"DELETE FROM specialnosti
    where id>2");
$con->query("DELETE FROM specialnosti where
    id>2");
$con->close();
//mysqli_close($con);
?>
```

Пример: MySQLi позволява да създаваме израз, който може да се използва многократно с различни параметри:

```
<?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 $stmt */
if ($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

Изпълнение на MySQL-базирани транзакции



- Както знаете, **транзакцията 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 станаха възможни за изпълнение с появата на MySQL Improved, тоест от PHP 5.x нагоре. Една транзакция се счита за успешна само ако всичките и команди са изпълнени успешно; При неуспех на която и да е команда, следва връщане в изходното състояние на системата ("rolled back"), с цел - да не се “счупи” системата.
- **Добър пример** за транзакция 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 се прибави трансферната сума към баланса на целевата сметка. Ако възникне грешка във втората стъпка, следва първата стъпка да бъде върната изходната ситуация за да се избегне несъответствие. Една сигурна откъм транзакции система автоматично ще изпълнява този обрат към изходното състояние на система.

Изпълнение на MySQL-базирани транзакции

Повечето бази от данни изпълняват транзакции, използвайки следните 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.

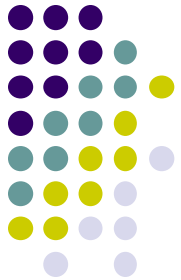
Вместо обичайната последователност команди:

START TRANSACTION, query, …, **COMMIT**

имаме:

$mysqli_autocommit(FALSE), query, …**, mysqli_commit();**

Изпълнение на MySQL- базирани транзакции – обобщен пример



```
<?php
    // съединение с базата
    $dbh = mysqli_connect($host, $user, $pass, $db);
    // FALSE - изключване на автоматичното
    реализиране commit
    mysqli_autocommit($dbh, FALSE);
    ...
    // изпълнение на query 1
    $result = mysqli_query($dbh, $query1);
    if ($result !== TRUE) {    mysqli_rollback($dbh); }
    //ако възникне грешка, roll back transaction
    // изпълнение на query 2
    $result = mysqli_query($dbh, $query2);
    if ($result !== TRUE) { mysqli_rollback($dbh); }
    // ако няма грешки, commit transaction
    mysqli_commit($dbh);
    // затваряне на конекцията
     mysqli_close($dbh);
?>
```

Изпълнение на MySQL- базирани транзакции – конкретен пример

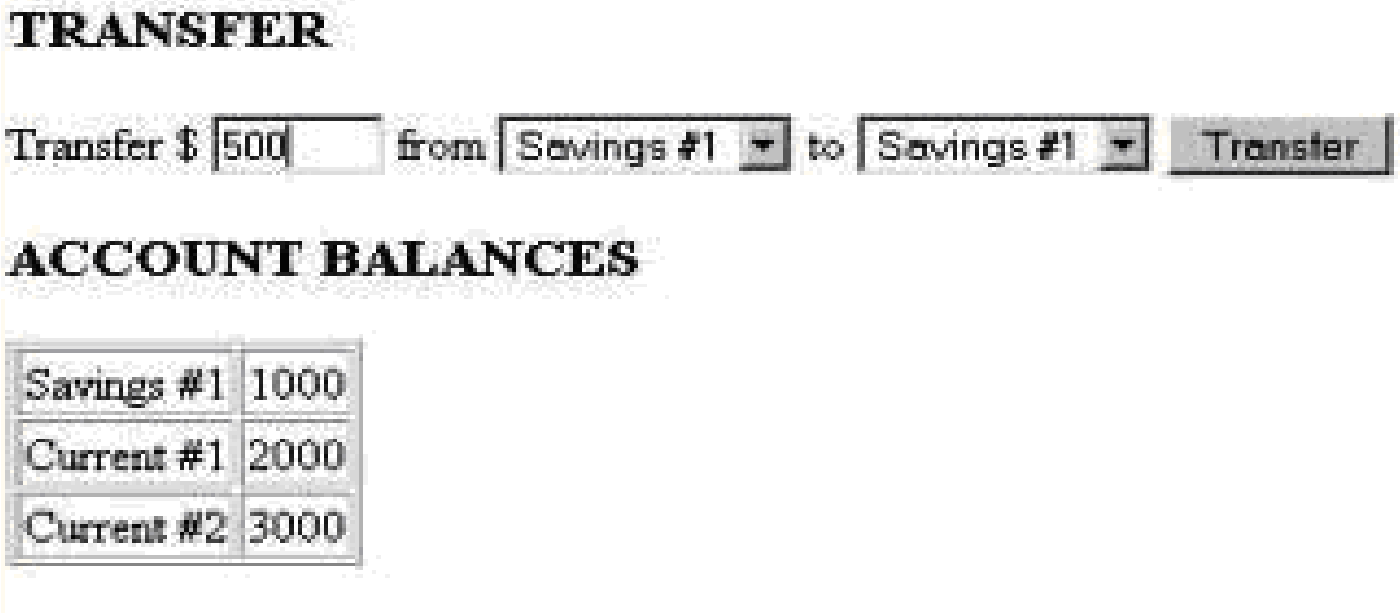
За да видите как работи на практика транзакцията, нека се върнем към примера за банков превод, обсъден по-рано.

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 таблица accounts на MySQL база данни "test", която изглежда като тази:

```
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)
```

Изпълнение на MySQL- базирани транзакции – конкретен пример

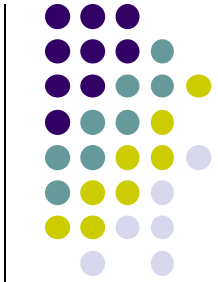
Сега е необходимо да се изгради елементарен интерфейс, който позволява на потребителите да въведат сума пари в брой и да я прехвърлят от една сметка в друга.



Действителната "сделка" (транзакция) ще се извършва с помощта на 2 UPDATE команди, едната за сума изтеглена от сметката на източника, и другата - кредитиране на целевата сметка.

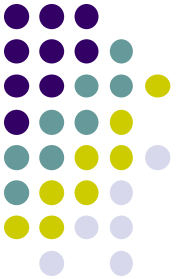
Като се има предвид, че всичко, което правим, е прехвърляне на пари между сметки, общия наличен баланс във всички сметки ($ 6000) трябва да остане постоянен по всяко време.

1. Създаване на БД test и таблица accounts – 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);
?>
```

1. Създаване на БД test и таблица accounts – ОО интерфейс

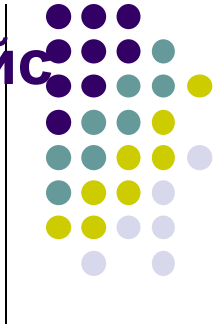


```
 <?php
// connect  to the MySQL server
$conn = new mysqli('localhost', 'root', '');
// check connection
If ($conn->connect_errno)           // if (mysqli_connect_errno())
{   exit('Connect failed: '. $conn->connect_error);
      //mysqli_connect_error()
}     // sql query with CREATE DATABASE
$sql = "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();
?>
```

2. Запълване с данни на таблицата – 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);
?>
```

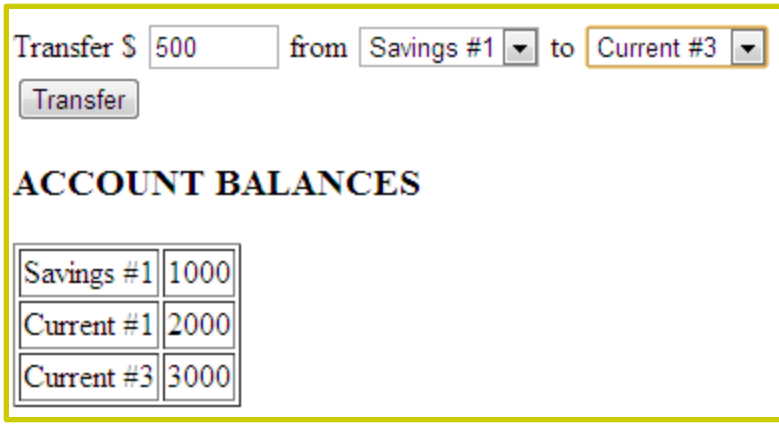
2. Запълване с данни на таблицата – ОО интерфейс



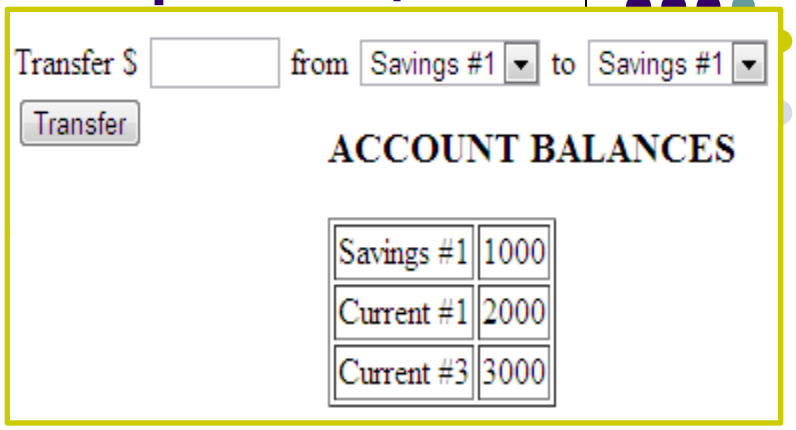
```
<?php
$dbhost = 'localhost'; $dbuser = 'root'; $dbpass = ''; $db="test";
//$conn = mysql_connect($dbhost, $dbuser, $dbpass);
$conn = new mysqli('localhost', 'root', '', $db);
If ($conn->connect_errno)            // if (mysqli_connect_errno())
{   exit('Connect failed: '. $conn->connect_error);
                                        //mysqli_connect_error()
}
echo 'Connected successfully';
/*$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 MySQL- базирана транзакция

```
<?php
 $dbh = mysqli_connect("localhost", "root", "", "test") or die("Cannot connect");
mysqli_autocommit($dbh, FALSE); // turn off auto-commit
// 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);
?>
```



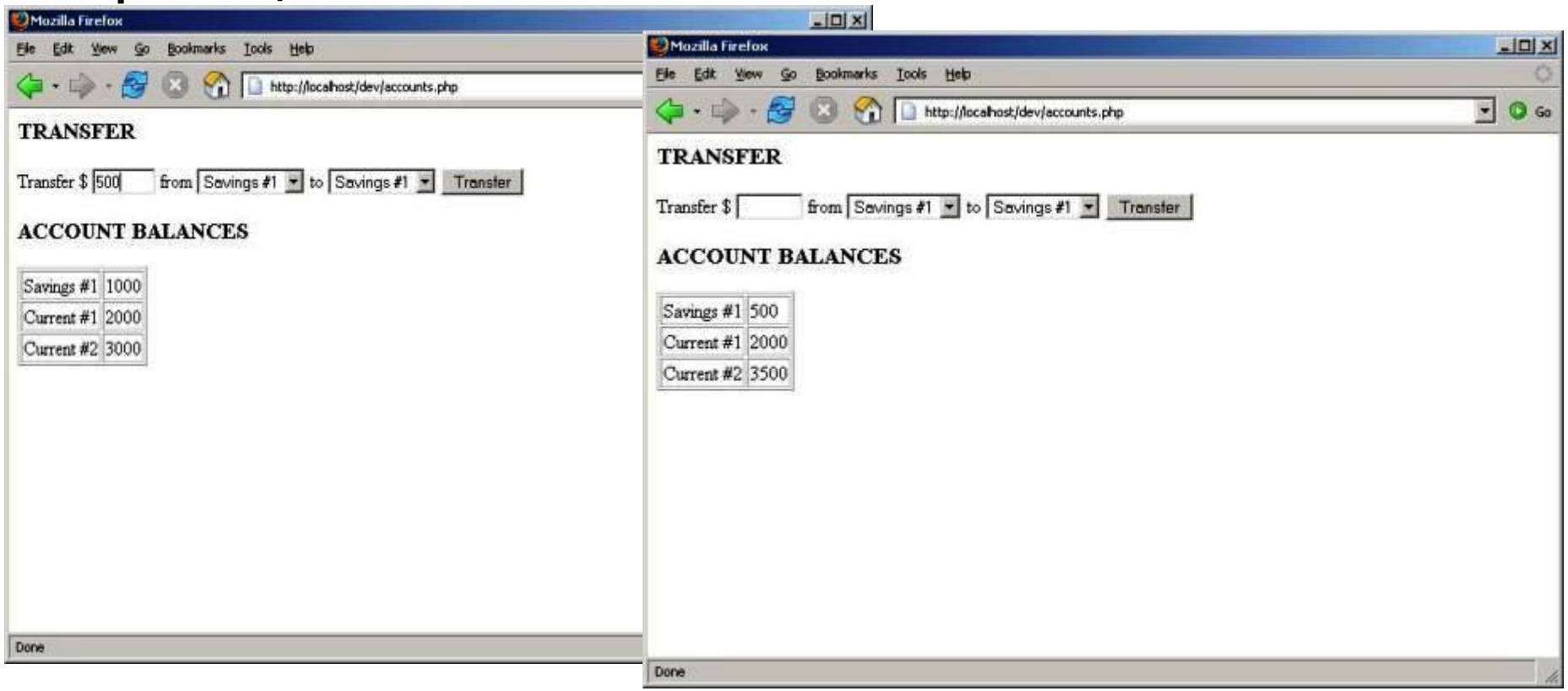
3.Изпълнение на MySQL- базирани транзакция (формата)



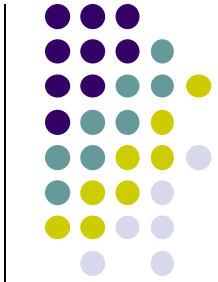

```
<html>
<head></head>
<body>
 <h3>TRANSFER</h3>
<form action="<?php $_PHP_SELF ; ?>" method="post">
Transfer $ <input type="text" name="amt" size="5"> from
 <select name="from">
<?php
foreach ($accounts as $a) {echo "<option value=\"" . $a['id'] . "\">" . $a['label'] . "</option>“;}
?>
</select>
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

Изпълнение на MySQL- базирани транзакции – конкретен пример

- 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 аккаунти, следва конструиране на форма с drop-down interface за селектиране на аккаунт източник и аккаунт цел, за транзакцията.
- Фигурите показват как изглежда формата, преди и след транзакцията.



Транзакции – още един пример



Имаме форма, за въвеждане на id, ime и email:

Index.php

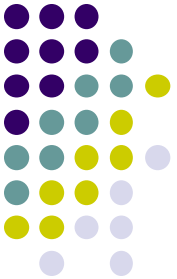
```
<form action="add.php” method="post">
ID: <input type="text" name="id" />
Ime: <input type="text" name="ime”/>
Email: <input type="text" name="email" />
<INPUT TYPE = "Submit" Name = "submit" VALUE =
   "save">
</select>
<form>
```

Add.php

```
<?php
//Добавяне на запис в транзакция
//$link = mysqli_connect("localhost", "root",
      "admin")
 $link = mysqli_connect("localhost", "root", "")
          or die('Could not connect: ' .
      mysqli_connect_error());
$sql_table =
 "CREATE TABLE IF NOT EXISTS test.tb1 (
id INTEGER PRIMARY KEY,ime VARCHAR(20),
 email VARCHAR(20)
) ENGINE=InnoDB ";
mysqli_query($link,$sql_table) or die("Could not
      create table". mysqli_error($link));
// Стартиране на транзакция
mysqli_query($link,"START TRANSACTION");
// Добавяне на запис
$sql_insert ="INSERT INTO test.tb1 (id, ime,
      email)    VALUES('$_POST[id]',
      '$_POST[ime]', '$_POST[email]')";
mysqli_query($link, $sql_insert) or die("Could
      not insert". mysqli_error($link));
```

```
// Получаване на параметъра save
if(isset($_POST["submit"]))
$save=$_POST["submit"];
else
$save="no";
// Проверка дали да се съхранява записа
if($save=="save"){
   mysqli_query($link, "COMMIT");
//Съхраняване на измененията
}
else
{
mysqli_query($link, "ROLLBACK");
//Анулиране на измененията
}
$result = mysql_query("SELECT * FROM test.tb1");
while($row = mysql_fetch_array($result))
  {
echo $row['id']; echo " " . $row['ime']; echo " " .
$row['email']; echo "<br />";
  }
mysqli_close($link); // Затваряне на връзката с
MySQL
?>
```

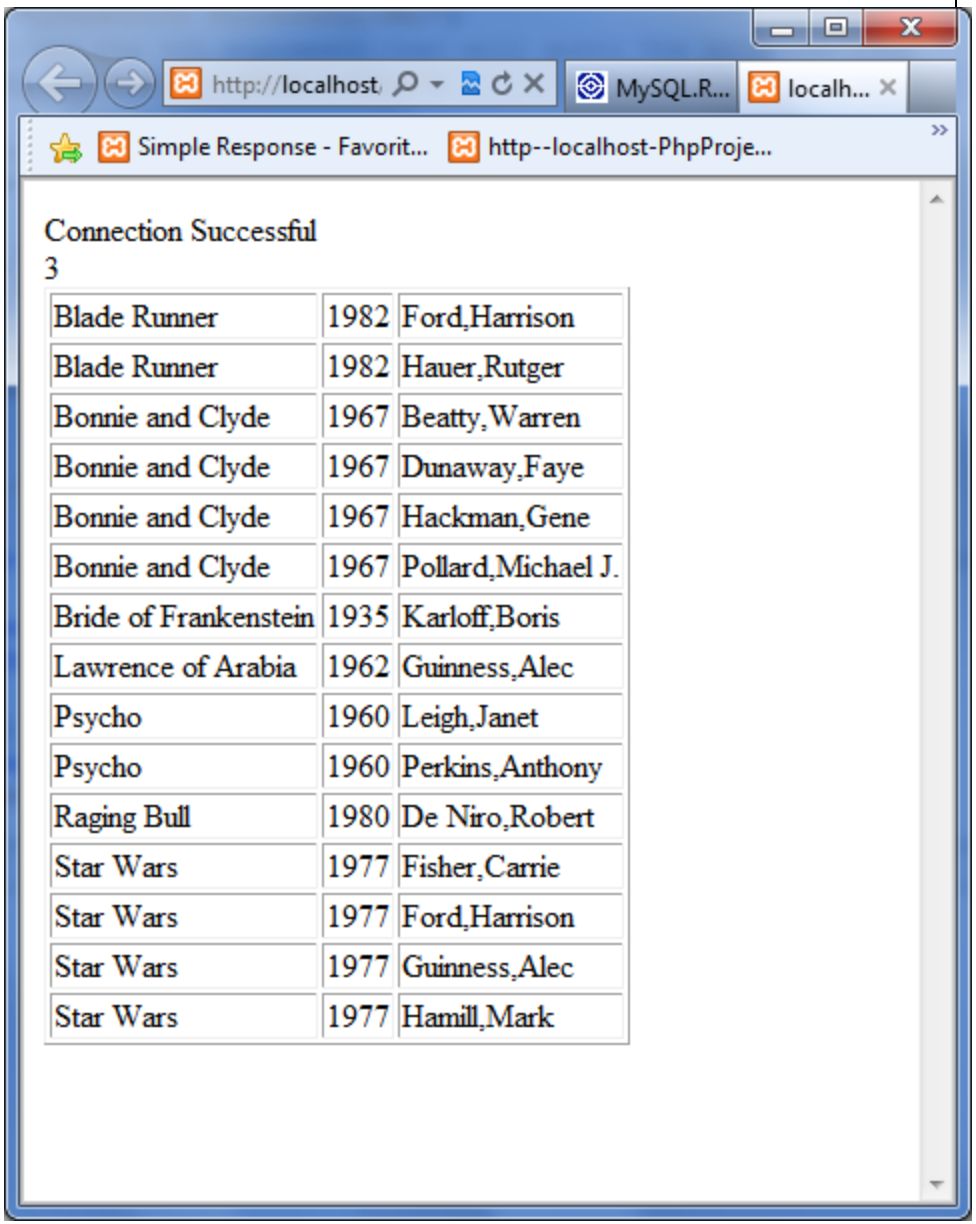
Операции чрез ODBC (Open Database Connectivity – стандартизиран програмен интерфейс на Microsoft) с Access БД db1 (без DSN, тоест без “Data Source Name”)



```
<?php
//създава се инстанция на ADO (ActiveX Data Object) 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;
?>
```

Резултат:

ADOdb — програмна библиотека, обезпечаваща приложен интерфейс за достъп до база от данни за езици за програмиране като PHP и Python, основана на някои концепция на Microsoft ActiveX Data Objects.



Blade Runner	1982	Ford,Harrison
Blade Runner	1982	Hauer,Rutger
Bonnie and Clyde	1967	Beatty,Warren
Bonnie and Clyde	1967	Dunaway,Faye
Bonnie and Clyde	1967	Hackman,Gene
Bonnie and Clyde	1967	Pollard,Michael J.
Bride of Frankenstein	1935	Karloff,Boris
Lawrence of Arabia	1962	Guinness,Alec
Psycho	1960	Leigh,Janet
Psycho	1960	Perkins,Anthony
Raging Bull	1980	De Niro,Robert
Star Wars	1977	Fisher,Carrie
Star Wars	1977	Ford,Harrison
Star Wars	1977	Guinness,Alec
Star Wars	1977	Hamill,Mark