Jak się to kiedyś robiło - Bazy danych - Podstawy PHP

Jak to się robiło kiedyś?

PHP Data Objects jest bardzo młodą biblioteką i mimo swoich zalet, wciąż tysiące skryptów napisanych wcześniej korzystają ze starych oraz niewygodnych funkcji komunikacji z bazami danych. Dlatego podręcznik ten zawiera także im poświęcony rozdział.

Pobieranie wyników

Ten zestaw funkcji w ogóle nie korzysta z dobrodziejstw programowania obiektowego - kiedy powstawał, w PHP po prostu jeszcze takowego nie było! Ponieważ znamy już się nieco na pracy z bazami danych, zaczniemy od razu od pobrania listy naszych produktów:

```
<?php

mysql_connect('localhost:3305', 'root', 'root'); // 1
mysql_select_db('produkty'); // 2

$r = mysql_query('SELECT `id`, `nazwa`, `ilosc` FROM `produkty` ORDER BY `ilosc`'); // 3

echo '<ul>';
while($row = mysql_fetch_assoc($r)) // 4
{
echo '<li>'.$row['id'].' - '.$row['nazwa'].' - '.$row['ilosc'].'</li>';
}
echo '</ul>';

mysql_close(); // 5

?>
```

Opis:

1. Funkcja *mysql_connect()*, która przyjmuje parametry: serwer, nazwa użytkownika, hasło, powoduje nawiązanie połączenia z bazą danych.
2. Funkcja *mysql_select_db()* wybiera bazę danych, na której będziemy pracować.
3. Funkcja *mysql_query()* wysyła zapytanie do bazy. W zależności od jego rodzaju generuje:
    - Zbiór wyników - dla zapytań *SELECT*.
    - **true** - dla zapytań typu *INSERT* jeśli wykonanie zapytania się powiodło
    - **false** - w przypadku jakiegokolwiek błędu w zapytaniu.
4. *mysql_fetch_assoc()* pobiera kolejny rekord ze zbioru wyników *$r* jako tablicę asocjacyjną. Istnieją jeszcze *mysql_fetch_num()* (numeryczne indeksy tablicy) oraz *mysql_fetch_array()* (połączenie obu tych sposobów).
5. *mysql_close()* zamyka połączenie z bazą.

Zauważ, że nie ma tutaj w ogóle czegoś takiego, jak zamykanie kursora. Jeśli korzystasz z tych funkcji, jest ono niepotrzebne, ponieważ to rozszerzenie tak naprawdę oszukuje. Wszystkie rekordy w są pobierane automatycznie przez *mysql_query()* i zapisywane do specjalnego bufora, skąd odczytuje je *mysql_fetch_assoc()*. Analogiczna metoda *PDOStatement::fetch()* pobierała dane bezpośrednio z serwera DB, umożliwiając ich natychmiastowe przetwarzanie. Obie techniki mają swoje plusy i minusy. PDO dzięki temu jest znacznie wydajniejsze, szczególnie przy większej liczbie rekordów, lecz nie można w nim sprawdzić, ile wyników ostatecznie dało nam zapytanie, dopóki ich wszystkich nie pobierzemy.

Obsługa błędów

Spróbujmy dodać do naszej listy produktów sortowanie:

```
<?php

mysql_connect('localhost:3305', 'root', 'root');
mysql_select_db('produkty');

$r = mysql_query('SELECT `id`, `nazwa`, `ilosc` FROM `produkty` ORDER BY `ilosc` DECS');

echo '<ul>';
while($row = mysql_fetch_assoc($r))
{
echo '<li>'.$row['id'].' - '.$row['nazwa'].' - '.$row['ilosc'].'</li>';
}
echo '</ul>';

mysql_close();

?>
```

Po uruchomieniu skryptu dostajemy dziwny komunikat:

Warning: mysql_fetch_assoc(): supplied argument is not a valid MySQL result resource in D:\Serwer\www\mysql\skrypt.php on line 9

Zobaczmy: mamy literówkę w zapytaniu; napisaliśmy *DECS* zamiast *DESC*, jednak *mysql_query()* w ogóle nie zgłosił żadnego komunikatu! Jedynym sygnałem, że coś jest nie tak, było zwrócenie wartości **false** zamiast zbioru wyników, co po wstawieniu do funkcji *mysql_fetch_assoc()* zaowocowało komunikatem o podaniu niewłaściwego parametru. Czy więc jest tu w ogóle jakaś obsługa błędów? Oczywiście, tyle że zakłada ona, że programista lubi monotonię i dopisywanie po każdym wywołaniu *mysql_query()* kodu

```
if(mysql_errno())
{
die('Blad MySQL: '.mysql_error().'<br/>');
}
```

W praktyce bardziej opłacało się tu napisać własny wariant *mysql_query()*, który automatyzuje tę czynność i samodzielnie zgłasza nam błędy, jak trzeba.

Wstawianie danych

Do wykonywania zapytań *INSERT* czy *UPDATE* także używana jest funkcja *mysql_query()*, lecz tym razem będzie ona za każdym razem zwracała wartość **true**. Aby sprawdzić, ile rekordów zostało zmodyfikowanych, musimy wywołać dodatkowo *mysql_affected_rows()*. Oto przepisany przykład z poprzedniego rozdziału dodający nowe produkty do bazy:

```
<?php

if($_SERVER['REQUEST_METHOD'] == 'POST')
{
mysql_connect('localhost:3305', 'root', 'root');
mysql_select_db('produkty');

mysql_query('INSERT INTO `produkty` (`nazwa`, `opis`, `ilosc`, `cena`, `jakosc`)        VALUES(
\''.mysql_real_escape_string($_POST['nazwa']).'\',
\''.mysql_real_escape_string($_POST['opis']).'\',
\''.mysql_real_escape_string($_POST['ilosc']).'\',
```

```
\''.mysql_real_escape_string($_POST['cena']).'\',
\''.mysql_real_escape_string($_POST['jakosc']).'\')');
$ilosc = mysql_affected_rows();

if($ilosc > 0)
{
echo 'Dodano: '.$ilosc.' rekordow';
}
else
{
echo 'Wystąpił błąd podczas dodawania rekordów!';
}

mysql_close();
}
else
{
?>
<form method="post" action="mysql_4.php">
<p>Nazwa: <input type="text" name="nazwa"/></p>
<p>Opis: <input type="text" name="opis"/></p>
<p>Ilosc: <input type="text" name="ilosc"/></p>
<p>Cena: <input type="text" name="cena"/></p>
<p>Jakosc: <select name="jakosc">
<option value="1">1</option>
<option value="2">2</option>
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
<option value="6">6</option>
</select></p>
<p><input type="submit" value="Dodaj"/></p>
</form>
<?php
}

?>
```

Zauważ, jak musimy tutaj umieszczać dane w zapytaniu. Nie tylko wymaga to zabawy operatorem łączenia ciągów, ale też konieczność wywoływania funkcji *mysql_real_escape_string()* do escape'owania danych i zapobiegania atakom SQL Injection. Oczywiście, gdy *magic quotes* było włączone, funkcji tej nie powinno się używać.

Informacje dodatkowe

Ze względu na charakter pobierania rekordów przez to rozszerzenie, umożliwia ono policzenie ilości zwróconych wyników jeszcze przed rozpoczęciem ich pobierania. Aby to wykonać, należy skorzystać z funkcji *mysql_num_rows()* z podanym jako parametr zbiorem wyników.

```
<?php

mysql_connect('localhost:3305', 'root', 'root');
mysql_select_db('produkty');

$r = mysql_query('SELECT `id`, `nazwa`, `ilosc` FROM `produkty` ORDER BY `ilosc`');

echo '<p>Pobrano '.mysql_num_rows($r).' wyników</p>';

echo '<ul>';
```

```
while($row = mysql_fetch_assoc($r))
{
echo '<li>'.$row['id'].' - '.$row['nazwa'].' - '.$row['ilosc'].'</li>';
}
echo '</ul>';

mysql_close();

?>
```

Jeśli dodaliśmy nowy rekord, możemy pobrać jego ID funkcją *mysql_insert_id()* wykonaną zaraz po funkcji *mysql_query()* z zapytaniem *INSERT*.

Zakończenie

Jak wspomnieliśmy, rozszerzenie to ma charakter historyczny. Twórcy PHP stopniowo ograniczają wsparcie dla niego; nie ma w nim np. żadnej implementacji mechanizmu podpinania, choć biblioteki klienckie MySQL jak najbardziej na to zezwalają. Jest ono także niewygodne w użyciu oraz na dłuższą metę mało efektywne. W codziennej praktyce chyba żaden szanujący się programista nie stosował żadnej z tych funkcji bezpośrednio, lecz korzystał z dodatkowej, napisanej w PHP nakładki automatyzującej wszystkie nużące czynności i zapewniającej wsparcie programowania obiektowego. Teraz, gdy do dyspozycji jest biblioteka PDO, sens korzystania z tych funkcji jest bardzo wątpliwy.