



Съвременно уеб програмиране с PHP 2011

РАБОТА С PHP И MYSQL

Какво ще правим днес?

- Ще се учим как да ползваме бази данни в истински PHP приложения
- Като направим един простичък уеб-магазин
 - Потребител да се логва
 - Да разглежда продукти и да праща имейл с поръчка
 - Да се пазят поръчките му
 - Админ панел за въвеждане на продукти

Не забравяйте полезните линкове



Стъпка 1: Дизайн на базата



Таблици

- ◎ InnoDB vs MyISAM
- ◎ Таблица за потребители
- ◎ Таблица за продукти
- ◎ Таблица за поръчки
- ◎ Foreign Keys...?
 - phpMyAdmin иска индекс
 - "No index defined" => може да е индекс, може да е грешен тип (размер, знак, null)
- ◎ Таблица за тагове

Въпросителни

- ⊙ Как да съхраняваме картинки?
- ⊙ Как да пазим пароли?
 - Какво е md5?
 - Salting, rainbow attacks
- ⊙ Как да работим с кирилица?
 - Разликата между charset и collation
 - SET NAMES UTF8

Интерлюдия: MySQL в PHP



Въведение

- ◎ Разширенията MySQL и mysqli
 - Малко ООП
 - Предимства на mysqli
- ◎ Инсталационни подробности
 - libmysql (< PHP 5.3.0)
 - Native driver (> PHP 5.3.0)
 - php_mysqli разширението в php.ini
- ◎ Алтернативи на MySQL
 - SQLite/PostGre/etc
 - PDO

Основните понятия в mysql

- ◎ Свързване с базата
 - обект-връзка
- ◎ Заявки – четене и писане
- ◎ Получаване на обект-резултат
 - Това е специален ресурсен тип в PHP
 - Само валидна заявка води до валиден резултат
- ◎ Извличане на данни от резултата
- ◎ Затваряне на връзката

Връзка с базата

- Създаваме mysqli обект

```
$mysqli = new mysqli('host','user','pass','database');  
// $mysqli = mysqli_connect('host','user','pass','database');
```

- **Забележка:** при MySQL
разширението избираме база!

```
$mysql = mysql_connect('host','user','pass');  
mysql_select_db('database',$mysql);
```

- Проверка за грешки

- Къде да пазим чувствителната информация?

Заявки

◎ Общ вид

```
$result = $mysqli->query("SELECT * FROM orders");  
// $result = mysqli_query($mysqli, "SELECT * FROM orders");
```

◎ Грешки

```
if (!$result)  
    echo "Error: ".$mysqli->error;
```

◎ За какво да ползваме insert_id

```
$result = $mysqli->query("INSERT INTO orders SET .... ");  
$new_order_id = $result->insert_id;
```

◎ За какво да ползваме affected_rows

Работа с резултата

● Проверка за валидност

- Заявките с резултат връщат ресурс
- Заявките без резултат – true
- Резултат false означава грешка

● Извличане на редове

```
$result = $mysqli->query("SELECT id, date FROM orders");  
$row = $result->fetch_row();  
$id = $row[0]; $id_again = $row['id'];  
$row = $result->fetch_assoc();  
$id = $row['id'];  
$row = $result->fetch_array();  
$id = $row[0];  
$row = $result->fetch_object();  
$id = $row->id;
```

Затваряне на връзката

◎ Общ вид

```
$mysqli->close();  
// $mysqli_close($mysqli);
```

◎ Persistent connections

- Идеята
- "p" пред името на хоста

```
$host="p:localhost";
```

- само с MySQL Native Driver

Стъпка 2: Логин



ОСНОВНИ МОМЕНТИ

- ⊙ Регистрационен модул
- ⊙ Забравена парола?
- ⊙ Ползване на сесията
 - `logout = session_destroy()`
- ⊙ SSL?

ПРИМЕРЧЕТО

just
another
example

Стъпка 3: Потребителската част



Дизайн

- ◎ Лого
- ◎ Заглавна лента с меню
- ◎ Управление на оторизацията
- ◎ Списък с продукти

Продукти и поръчки

- ⊙ Извличане от базата
 - **htmlspecialchars()**
- ⊙ Визуализация
- ⊙ Модул за поръчка
 - Отново използваме един и същи скрипт за обработка на поръчка и за показване на поръчките

```
//проверяваме дали имаме заявена поръчка
if (isset($_POST['order'])) {
    ... //обработваме
    $query = "INSERT INTO orders ....;
    db_query($query);
}
//после продължаваме с показването на поръчките...
```

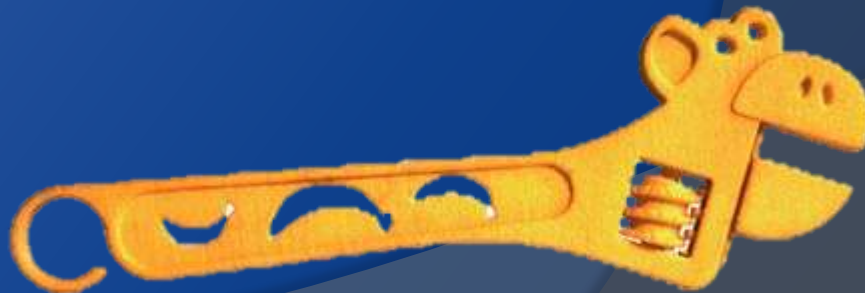
ВАЖНО: Потребителски вход

- ⦿ Какво е SQL Injection?
- ⦿ **Escaping** на всеки потребителски вход

```
$mysqli->real_escape_string($username);
```

- ⦿ Параметризирани заявки (вж. екстрите)

Стъпка 4: Админ панел



ОСНОВНИ ПОНЯТИЯ

- ◎ Колко сигурност трябва?
 - Оторизация чрез .htaccess (вж свитъчето)
 - Оторизация с логин модул (както в примера)
- ◎ Какво е CRUD?
 - Стандартните функции на админа: **Create, Read, Update, Delete**
 - Готови решения
 - Подавате схема на базата и се генерира админ
 - Symfony Framework
 - Всякакви CMS системи
 - ... и много други



Организация на формите

- ◎ Вариант 1 – всеки елемент е или си има отделен `<form>` таг и отделен `<input type=submit>` бутон
 - По-лесно за писане и за обработка на резултата
 - Позволява да се редактира само един елемент
 - В нашия пример е така
- ◎ Вариант 1 работи добре с AJAX, когато няма нужда да се презарежда страницата за всяка редакция

Организация на формите

- ◎ Вариант 2 – елементите са в обща форма и се събмитват заедно
 - Името на всяко поле от формата трябва да носи информация за кой елемент се отнася
 - Може да са от типа **name=description_18**, където 18 е пример за id на продукта, който редактираме
 - Или да ползваме масиви: **name=description[]**, след което да обхождаме масивите при обработка на резултата
 - Трябва да правим проверка за всяко поле от `$_POST` дали наистина е имало промяна
 - Позволява редактиране на много елементи наведнъж

Епилог: Няколко екстра неща



Транзакции с mysql

- ⦿ Кога да ползваме транзакции
 - (1) Майката на Иван иска да му преведе 200 лв.
 - (2) Нашият софтуер изважда 200 лв. от сметката
 - (3) Опитва да добави 200 лв. в сметката на Иван, но връзката с базата се чупи и скриптът спира.
 - (4) Иван и майка му са 200лв. назад
 - => когато имаме операции, зависими една от друга
- ⦿ За транзакции базата ни трябва да е InnoDB!
- ⦿ Първа стъпка: AUTOCOMMIT = FALSE

```
$mysql->autocommit(FALSE); //TRUE значи без транзакции  
// mysql_autocommit($mysql,FALSE);
```

Транзакции с mysqli - 2

- ◎ Rollback & Commit
- ◎ Особеност: AUTO INCREMENT ID

```
$mysqli->autocommit(FALSE);
$mysqli->query("INSERT INTO users (username, pass, level, email) VALUES
('test',MD5('test'),0,'test@test.bg')");
$result=$mysqli->query("SELECT COUNT(*) FROM users");
$row=$result->fetch_row();
echo 'Number of rows in USERS table '.$row[0]; //извежда 5
echo 'Last inserted id: '.$mysqli->insert_id; //извежда 12
$mysqli->rollback(); //връщаме транзакцията
$result=$mysqli->query("SELECT COUNT(*) FROM users")
$row=$result->fetch_row();
echo 'Number of rows in USERS table '.$row[0]; //извежда 4
$mysqli->query("INSERT INTO users (username, pass, level, email) VALUES
('test',MD5('test'),0,'test@test.bg')"); //няма грешка с уникално username
echo 'Last inserted id: '.$mysqli->insert_id; //извежда 13 въпреки rollback
$mysqli->commit(); //пускаме транзакцията
```

Параметри и готови заявки

- Заявките се оптимизират и преизползват
=> ефективност
- Параметрите подлежат на проверка
=> сигурност

```
$prepared = $mysqli->prepare('SELECT id FROM users WHERE username=? AND password=MD5(?) '); //въпросителната означава параметър
```

```
$prepared->bind_param('ss',$username,$pass); //s означава string
```

```
$prepared->bind_result($id); //$id ще пази резултата
```

```
$prepared->execute(); //изпълнява заявката, може многократно
```

```
if ($prepared->fetch()) //взима ред от резултата
```

```
    echo $id; //извежда id на търсения потребител
```

```
else
```

```
    echo "Wrong username/password";
```

```
// по-подробен пример вижте във файла login_param.php в примерите ;)
```

Капсулиране на работата с БД

◎ Идеята

- Поддържахме различни бази данни, без да пренаписваме код
- Управляваме поведението при грешки от едно единствено място

◎ Реализация

- Правим си свой db.php файл или клас (както е в нашия пример)
- PDO – PHP Data Objects
- PEAR::MDB2



Работа с PDO

- Трябва ни драйвер за нашата база (обикновено си има)
- Използва се подобно на готовите заявки
- Проверката за грешки е с изключения

```
$pdo = new PDO('mysql:dbname=shopsite;host=localhost',$USER,$PASS);  
// Първи начин: обхождаме резултата  
$result = $pdo->query('SELECT * FROM orders');  
foreach ($result as $row)  
    echo "Order No: ".$row['id'];  
//ако не сме обходили до край, трябва да викнем $result->closeCursor();  
//Втори начин: като готова заявка  
$statement = $pdo->prepare('SELECT * FROM orders WHERE user_id = ?');  
$statement->execute(array($user_id)); //подаваме масив с параметрите  
//параметри може да се подават и с bind_param, подобно на готовите заявки  
$orders = $statement->fetchAll(); //масив от всички редове на резултата  
// по-подробен пример вижте във файла db_pdo.php в примерите ;)
```

