

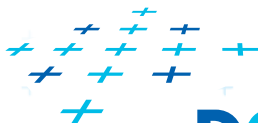
DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Přístup k databázi z webové aplikace v PHP

Martin Klíma

DATABÁZE ÚVOD



DCGI



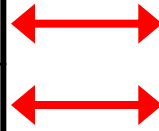
Relační databáze - pojmy

- Databázový systém – program pro práci a správu DB
- Databáze – souhrn datových struktur obsahující data
- Tabulky – databáze se skládá z tabulek, tabulka má sloupce a řádky
- Sloupec – popis vlastnosti objektu
- Řádek, záznam – konkrétní data uložená v tabulce
- Primární klíč – množina sloupců, které jednoznačně identifikují záznam
- Relace – vztah mezi objekty v databázi



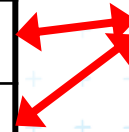
Databáze - relace

| ID_Objednavky | Mnozstvi | Produkt |
|---------------|----------|---------|
| 1 | 10 | Okurky |
| 2 | 15 | Papriky |



| ID_Transakce | ID_Objednavky | Datum |
|--------------|---------------|----------|
| 33 | 1 | 1.1.2005 |
| 34 | 2 | 5.1.2005 |

| ID_Objednavky | Mnozstvi | Produkt | ID_Zakaznika |
|---------------|----------|---------|--------------|
| 1 | 10 | Okurky | 102 |
| 2 | 15 | Papriky | 102 |

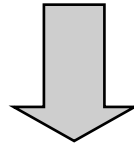
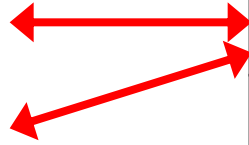


| ID_Zakaznika | Jmeno |
|--------------|--------|
| 102 | Novák |
| 103 | Omáčka |

Databáze - relace

| ID_reky | Jmeno |
|---------|--------|
| 1 | Labe |
| 2 | Vltava |

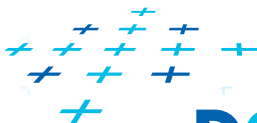
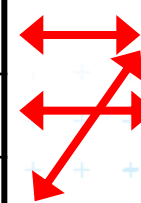
| ID_Zeme | Nazev |
|---------|---------|
| 102 | CR |
| 103 | Nemecko |

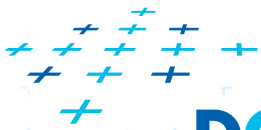


| ID_reky | Jmeno |
|---------|--------|
| 1 | Labe |
| 2 | Vltava |

| ID_reky | ID_Zeme |
|---------|---------|
| 1 | 102 |
| 1 | 103 |
| 2 | 102 |

| ID_Zeme | Nazev |
|---------|---------|
| 102 | CR |
| 103 | Nemecko |





DCGI



Manipulace s databází

- Jazyk SQL (Structured Query Language)
 - DDL = Data Definition Language
 - DML = Data Manipulation Language

■ DDL

- Create
- Alter
- Drop

■ DML

- Select
- INSERT
- Update
- Delete



Příklad použití databáze

- Databáze zboží
- Kategorie zboží
 - Kategorie může mít N podkategorií
 - Každá podkategorie patří do jedné kategorie (strom)
- Každé zboží může patřit do více kategorií



DB - příklad

| ID | Jmeno | Cena |
|----|-----------|------|
| 1 | Jablka č. | 35 |
| 2 | Jablka z. | 38 |

| ID | Nazev | Nadkategorie |
|----|-----------|--------------|
| 1 | Ovoce | null |
| 2 | Zelenina | null |
| 3 | Okurky | 103 |
| 4 | Potraviny | null |

| ID | IDZbozi | IDKategorie |
|----|---------|-------------|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 1 | 4 |
| 4 | 2 | 4 |



MYSQL



MySQL

- Velice úspěšný RDBMS systém
- Open source
- Poskytovaný skoro všude
- Vlastnosti
 - několik různých engine
 - rychlé i pomalé
 - transakční i netransakční
 - podpora různých kódování
 - uložené procedury
 - knihovny, PHP podpora od vzniku PHP



Databázové enginy

- MyISAM
- InnoDB
- MERGE
- MEMORY
- EXAMPLE
- FEDERATED
- ARCHIVE
- NDB

Zjistit aktuálně instalované enginy můžeme takto:
SHOW ENGINES\G

MyISAM Engine

- Velmi rychlý
- Není transakční (proto je také rychlý)
- Nepodporuje ref. integritu
 - Syntakticky ano, fakticky ne
- Každá tabulka je v samostatném souboru
 - soubor se jmenuje podle jména tabulky

Innodb engine - přehled vlastností

- Plně transakční zpracování
 - ACID kompatibilní = commit, rollback, zotavení
- Zamykání záznamů (po řádcích)
- Podporuje cizí klíče (FOREIGN KEY)
- Je součástí základní distribuce
- Rychlý engine, ale ne tak rychlý jako MyISAM
- GNU GPL License Version 2



InnoDB vlastnosti

- Tabulky jsou uloženy ve společném souboru (souborech) a to včetně indexů
- Tabulky mohou být uloženy ve více souborech i na více různých discích
- Tabulky mohou být uloženy i separátně, každá v jednom souboru
 - pozor, chová se to jinak než MyISAM
 - nelze jen tak kopírovat

Připojení k MySQL

- Pomocí klienta PHP
- Pomocí řádkového klienta v adresáři bin/mysql
- Pomocí klienta třetí strany

mysql -h localhost -u xklima -p xklima

Host, default je localhost

jméno uživatele

budu zadávat heslo interaktivně

jméno databáze, kterou otevírám

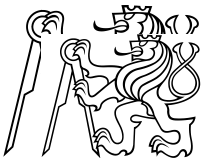


PHP A DATABÁZE



PHP a DB

- PHP má nativní podporu pro většinu existujících DB
- MySQL v první řadě
- Čtyři způsoby, jak může DB podporovat
 1. Nativní podpora pomocí dynamické knihovny
 - např. mysql.dll, mysqli.dll
 2. Nativní podpora v jádře PHP
 - je nutné zkompilovat
 3. Pomocí ODBC
 - je nutné mít podporu ODBC
 4. Pomocí abstraktní vrstvy, např. PDO



Naše DB

```
CREATE TABLE `zbozi` (  
  `ID` int(11) NOT NULL auto_increment,  
  `Nazev` varchar(100) NOT NULL,  
  `Popis` text,  
  `ObrazekURL` varchar(50) default NULL,  
  `Cena` double NOT NULL default '0',  
  PRIMARY KEY (`ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_czech_ci;  
  
GRANT ALL PRIVILEGES ON x36www.* TO x36www_user@localhost IDENTIFIED  
BY "x36heslo";  
  
FLUSH PRIVILEGES;  
  
set names utf8;  
insert into `zbozi`  
(`ID`,`Nazev`,`Popis`,`ObrazekURL`,`Cena`)  
values (1,'Myš','Počítačová myš','mys.jpg',100),  
(2,'HDD 500','Harddisk s kapacitou 500GB','hdd1.jpg',1200),  
(3,'HDD 750','Harddisk s kapacitou 750 GB','hdd2.jpg',1400),  
(4,'Monitor 1','Monitor s rozlišením 1024x768','monitor1.jpg',1000);
```



Nutné kroky při práci s DB

1. Připojení k databázovému stroji
 - URL databáze
 - ověření uživatele
2. Výběr databáze
 - body 1 a 2 lze sloučit
3. Sestavení a poslání dotazu
4. Čtení resultsetu (pokud ho daný dotaz vrací)
5. Uvolnění resultsetu
6. Uzavření spojení

Pozor!!! Každá operace může skončit chybou, musím na to správně reagovat



Implementace

```
define ("DB_HOST", "localhost");
define ("DB_NAME", "x36www");
define ("DB_USER", "x36www_user");
define ("DB_PASSWD", "x36heslo");

// pokusim se pripojit k DB stroji
$link = mysqli_connect(DB_HOST, DB_USER, DB_PASSWD);
if (!$link) {
    echo "Nepodařilo se spojit s DB.<br>";
    echo mysqli_connect_error();
    exit();
}

// pokusim se vybrat si správnou databazi
$success = mysqli_select_db($link, DB_NAME);
if (!$success) {
    echo "Nepodařilo se přepnout na správnou databázi";
    exit();
}
```



Implementace

```
// sestavim si dotaz
$sql = "SELECT * FROM zbozi WHERE zbozi.Cena <=100 ORDER BY
zbozi.Cena, zbozi.Nazev";

// provedu dotaz
$result = mysqli_query($link, $sql);
if ($result) {
    // iteruj vysledek a vypis ho na obrazovku
    while ($row = mysqli_fetch_assoc($result)) {
        echo "\n<div>";
        echo htmlspecialchars($row['Nazev']);
        echo ": ";
        echo $row['Cena'];
        echo "</div>";
    }
    // uvoni resultset
    mysqli_free_result($result);
}
// uzavri spojeni s db
mysqli_close($link);
```

Co vrací `mysqli_query`

Pro dotazy `SELECT`, `SHOW`, `DESCRIBE` a `EXPLAIN` vrací

`mysqli_query()` výsledek (objekt s tabulkou dat), který musí

být zpracován dalšími funkcemi.

Pro `UPDATE`, `INSERT`, `DELETE`, `DROP` a další dotazy pro

manipulaci s DB vrací *`mysqli_query()` hodnotu typu `boolean`.*

Zpracování výsledku z `mysqli_query`

Mnoho funkcí na zpracování **`$result`**

`mysqli_fetch_array` — Fetch a result row as an associative, a numeric array, or both

`mysqli_fetch_assoc` — Fetch a result row as an associative array

`mysqli_fetch_field_direct` — Fetch meta-data for a single field

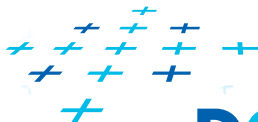
`mysqli_fetch_field` — Returns the next field in the result set

`mysqli_fetch_fields` — Returns an array of objects representing the fields in a result set

`mysqli_fetch_lengths` — Returns the lengths of the columns of the current row in the result set

`mysqli_fetch_object` — Returns the current row of a result set as an object

`mysqli_fetch_row` — Get a result row as an enumerated array



SQL injection

Často sestavujeme dotaz z parametrů zadaných uživatelem.

Musíme počítat s tím, že uživatel nemusí zadat "pěkná" data.

Uživatel může zničit sql dotaz.

Uživatel může pozměnit dotaz tak, že vrátí jiný výsledek.



SQL Injection

zkuste url:

sql_injection.php?max=aaa

```
// sestavim sql dotaz z $_GET parametru
// predpokladam, ze uzivatel ve formulari zadal max cenu zbozi
// parametr max
// spatne sestaveny sql dotaz
$sql_spatne = "SELECT * FROM zbozi WHERE Cena <=
" . $_GET['max'] . " ORDER BY Nazev";

$cena_max = intval($_GET['max']);
$sql_spravne = "SELECT * FROM zbozi WHERE Cena <= " . $cena_max . "
ORDER BY Nazev";

echo htmlspecialchars("Spatne: " . $sql_spatne);
echo "<br/>";
echo htmlspecialchars("Spravne: " . $sql_spravne);
echo "<br/>";
```

SQL Injection

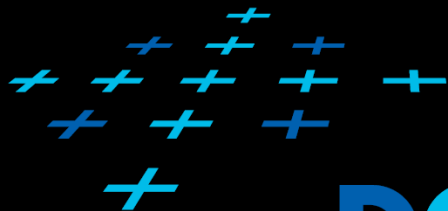
zkuste URL:

sql_injection.php?search=dd'%20OR%20true%20OR%20Nazev%20like%20'

```
// predpokladam, ze uzivatel zadal vyhledavaci retezec nazvu
// param search
// rekne, ze v systemu jsou dve role: 1: admin, 2: obyc
uzivatel
$sql_spatne2 = "SELECT * FROM zbozi WHERE Priv = 2 AND Nazev
like '%" . $_GET['search'] . "'";

$search = mysql_real_escape_string($_GET['search']);
$sql_spravne2 = "SELECT * FROM zbozi WHERE Priv = 2 AND Nazev
like '%" . $search . "'";

echo htmlspecialchars("Spatne 2: " . $sql_spatne2);
echo "<br/>";
echo htmlspecialchars("Spravne 2: " . $sql_spravne2);
```

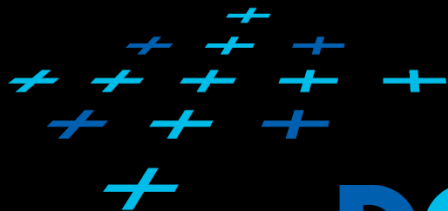


DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Veškeré skripty musí být ošetřeny na SQL Injection

..je to zápočtová podmínka



DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

PDO

Co to je PDO

PDO je rozšíření PHP modulu o databázovou vrstvu

Abstraktní přístup k DB

Implementace pro nejběžnější DB (mysql, oracle, pg, ...)

Podpora transakcí

Objektový přístup

```
<?php
$user = "x36www_user";
$pass = "x36heslo";

try {
    $dbh = new PDO('mysql:host=localhost;dbname=x36www',
    $user, $pass, array(
        PDO::ATTR_PERSISTENT => true
    ));

    $dbh->beginTransaction();
    $dbh->exec("insert into zbozi (Nazev, Popis, ObrazekURL,
Cena)
    values ('Podložka', 'Podložka pod myš', 'podlozka.jpg',
'46')");
    $dbh->commit();
} catch (Exception $e) {
    $dbh->rollback();
    echo "Failed: " . $e->getMessage();

}
?>
```

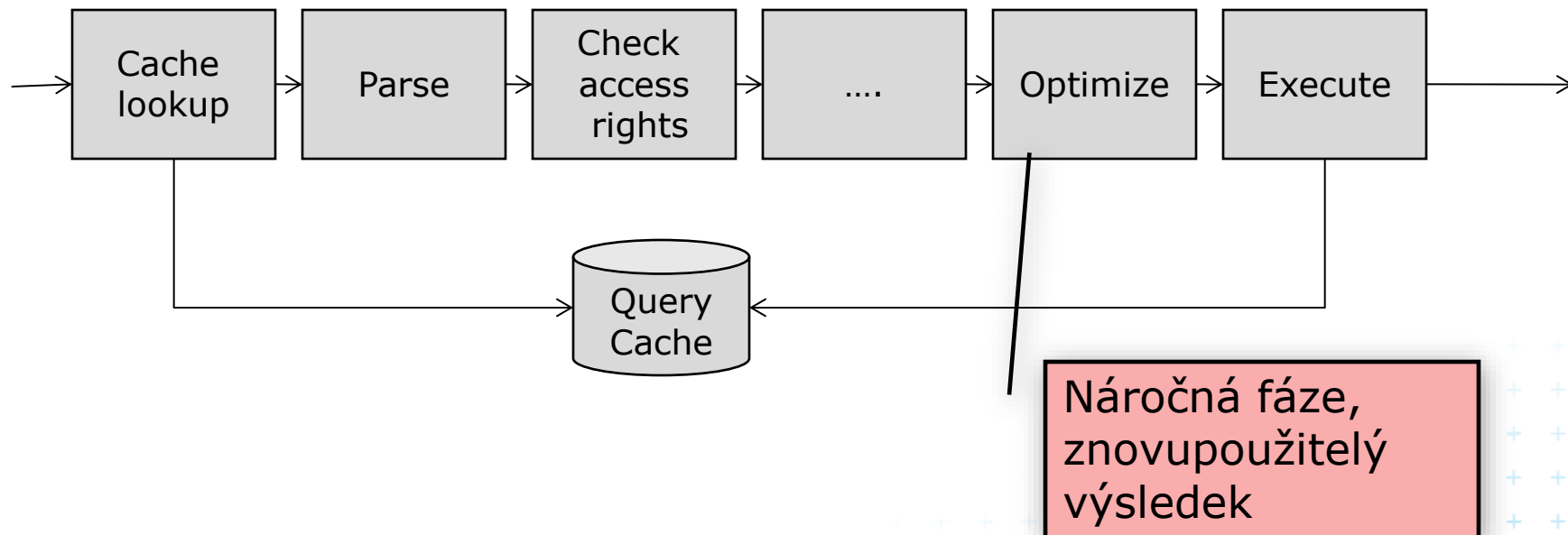


PREPARED STATEMENTS



Zpracování dotazu databází

Typické fáze zpracování



PreparedStatement provádí optimalizaci jednou, dále posílá data do fáze execute

```
$dbh->beginTransaction();
```

```
$stmt = $dbh->prepare("INSERT INTO zbozi (Nazev, Popis,  
ObrazekURL, Cena)
```

```
VALUES (:nazev, :popis, :url, :cena)");
```

```
$stmt->bindParam(':nazev', $nazev);
```

```
$stmt->bindParam(':popis', $popis);
```

```
$stmt->bindParam(':url', $url);
```

```
$stmt->bindParam(':cena', $cena);
```

```
// vloz jeden zaznam
```

```
$nazev = 'DVD mechanika';
```

```
$popis = 'DVD vypalovačka';
```

```
$url = 'dvdobrazek.jpg';
```

```
$cena = '350';
```

```
$stmt->execute();
```

```
// vloz dalsi zaznam
```

```
$nazev = 'DVD mechanika 2';
```

```
$popis = 'DVD vypalovačka 2';
```

```
$url = 'dvdobrazek2.jpg';
```

```
$cena = 190;
```

```
$stmt->execute();
```

```
$dbh->commit();
```

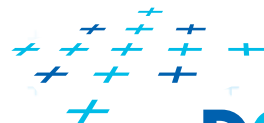


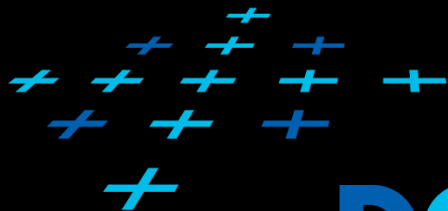
```
$dbh->beginTransaction();

$stmt = $dbh->prepare("INSERT INTO zbozi (Nazev, Popis,
ObrazekURL, Cena)
VALUES (?, ?, ?, ?)");
$stmt->bindParam(1, $nazev);
$stmt->bindParam(2, $popis);
$stmt->bindParam(3, $url);
$stmt->bindParam(4, $cena);

// vloz jeden zaznam
$nazev = 'DVD mechanika';
$popis = 'DVD vypalovačka';
$url = 'dvdobrazek.jpg';
$cena = '350';
$stmt->execute();

$dbh->commit();
```



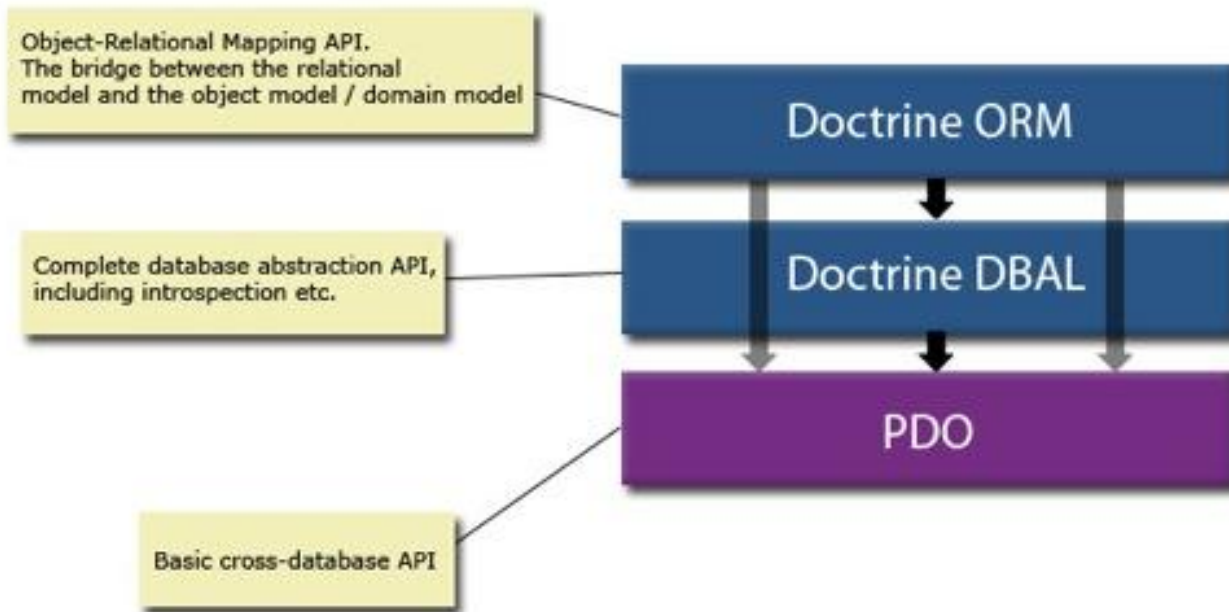


DCGI

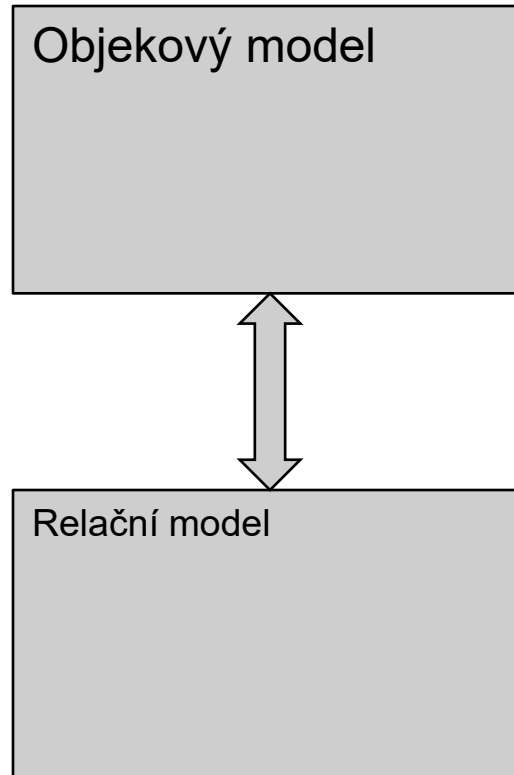
KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Doctrine ORM





Cíl – mapování objektového modelu a ER



Ukázka – navázání spojení

```
require_once('../doctrine/branches/1.2/lib/Doctrine.php');  
spl_autoload_register(array('Doctrine', 'autoload'));
```

```
$dsn = 'mysql:dbname=testdb;host=127.0.0.1';  
$user = 'dbuser';  
$password = 'dbpass';  
  
$dbh = new PDO($dsn, $user, $password);  
$conn = Doctrine_Manager::connection($dbh);
```

```
// At this point no actual connection to the database is created  
$conn = Doctrine_Manager::connection('mysql://username:password@localhost/test');  
  
// The first time the connection is needed, it is instantiated  
// This query triggers the connection to be created  
$conn->execute('SHOW TABLES');
```

Ukázka použití

```
$conn->export->createTable('test', array('name' => array('type' =>
'string')));
$conn->execute('INSERT INTO test (name) VALUES (?)', array('Martin'));

$stmt = $conn->prepare('SELECT * FROM test');
$stmt->execute();
$results = $stmt->fetchAll();
print_r($results);
```

```
Array
(
    [0] => Array
        (
            [name] => Martin
            [0] => Martin
        )
)
```



Definice dat

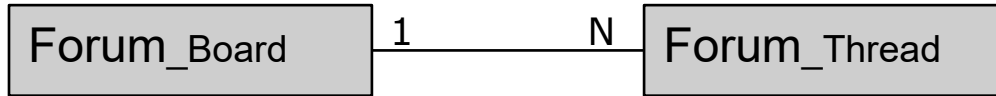
```
// models/Book.php
class Book extends Doctrine_Record
{
    public function setTableDefinition()
    {
        $this->hasColumn('bookTitle as title', 'string');
    }
}
```

```
// test.php

// ...
$book = new Book();
$book->title = 'Some book';
$book->save();
```



Relační závislosti



```
// models/Forum_Boadr.php

class Forum_Board extends Doctrine_Record
{
    public function setTableDefinition()
    {
        $this->hasColumn('name', 'string',
100);
        $this->hasColumn('description',
'string', 5000);
    }

    public function setUp()
    {
        $this->hasMany('Forum_Thread as
Threads', array(
            'local' => 'id',
            'foreign' => 'board_id'
        )
    );
    }
}
```

```
// models/Forum_Thread.php

class Forum_Thread extends Doctrine_Record
{
    public function setTableDefinition()
    {
        $this->hasColumn('user_id', 'integer');
        $this->hasColumn('board_id', 'integer');
        $this->hasColumn('title', 'string', 200);
        $this->hasColumn('updated', 'integer', 10);
        $this->hasColumn('closed', 'integer', 1);
    }

    public function setUp()
    {
        $this->hasOne('Forum_Board as Board', array(
            'local' => 'board_id',
            'foreign' => 'id'
        )
    );

        $this->hasOne('User', array(
            'local' => 'user_id',
            'foreign' => 'id'
        )
    );
    }
}
```

User definován
jinde

Ukázka použití

```
$board = new Forum_Board();  
$board->name = 'Some board';  
  
$board->Threads[0]->title = 'new thread 1';  
$board->Threads[1]->title = 'new thread 2';  
  
$user = new User();  
$user->username = 'jwage';  
$board->Threads[0]->User = $user;  
$board->Threads[1]->User = $user;  
  
$board->save();
```



Práce s modelem

```
// test.php
```

```
$user = new User();
```

```
$user['username'] = 'jwage';
```

```
$user['password'] = 'changeme';
```

Ekvivalentní

```
$email = $user->Email;
```

```
$email = $user->get('Email');
```

```
$email = $user['Email'];
```

Ekvivalentní

```
$user->save();
```



Dotazování a další operace

```
// select pres primarni klic
$user = Doctrine_Core::getTable('User')->find(1);
echo $user->Email['address'];
echo $user->Phonenumbers[0]->phonenumber;

// query
$q = Doctrine_Query::create()
    ->from('User u')
    ->leftJoin('u.Email e')
    ->leftJoin('u.Phonenumbers p')
    ->where('u.id = ?', 1);

$user = $q->fetchOne();
echo $user->Email['address'];
echo $user->Phonenumbers[0]['phonenumber'];

// test.php

// update
$user->Email['address'] = 'koskenkorva@drinkmore.info';
$user->Phonenumbers[0]['phonenumber'] = '123123';
$user->save();

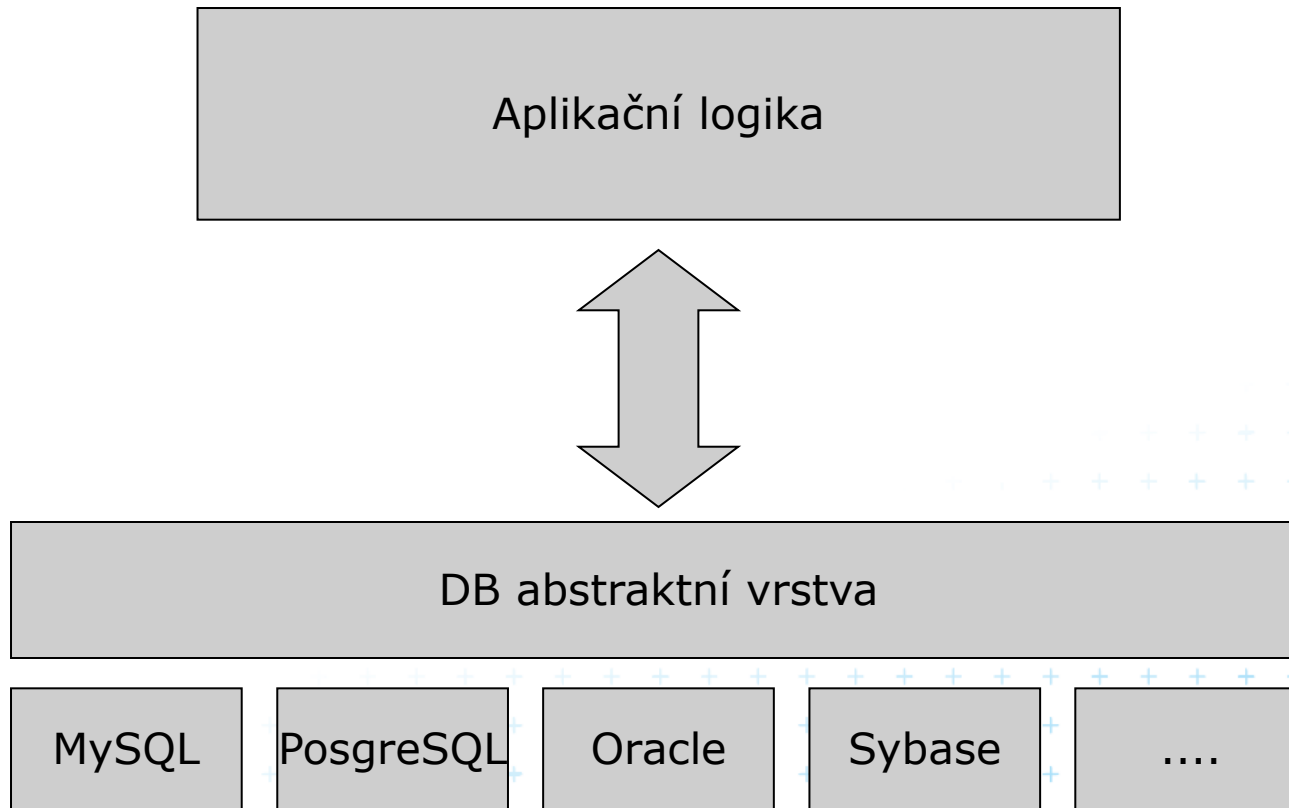
// smazani
$user->Email->delete();
```



DB A OBJEKTIVÉ PROG. VZORY



Abstrakce



Vzor Factory

Abstraktní vrstva

DBLayer

- connect(\$param)
- query(\$sql)
- ...

Implementace X

PGDBLayer

- connect(\$param)
- query(\$sql)
- ...

Implementace Y

MYSQldbLayer

- connect(\$param)
- query(\$sql)
- ...



Vzor Factory

- Používá se tehdy, když chceme získat instanci nějakého objektu, ale nechceme se starat o to, jak tento objekt vytvořit
- Příklad:
 - chceme přistupovat k databázi
 - databází je ale mnoho různých druhů (mysql, oracle, ...)
 - všechny db implementují stejné rozhraní

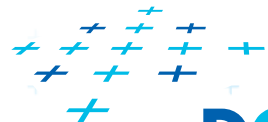


Vzor Factory Impl 1/2

```
interface DBLayer {
    public function connect($param);
    public function query($sql);
}

class MySQLDBLayer implements DBLayer {
    public function connect($params) {
        // mysql_connect(...)
    }
    public function query($sql) {
        // mysql_query(...)
    }
}

class PGDBLayer implements DBLayer {
    public function connect($params) {
        //pg_connect(...)
    }
    public function query($sql) {
        // pg_query(...)
    }
}
```



Vzor Factory Impl 2/2

```
// vzor factory = tovarena na objekty
class DBFactory {
    // zde to prijde!
    // vim, ze vratim rozhrani typu DBLayer
    public static function getDBLayer($type) {
        switch ($type) {
            case "MySQL": return new MySQLDBLayer(); break;
            case "PG": return new PGDBLayer(); break;
            default: return new MySQLDBLayer();
        }
    }
}
```

Poznámka:

často se setkáme s implementací factory bez parametru. Ten je potom zjištěn v těle metody například z nějakého konfiguračního souboru. Rozhodnutí o použité DB udělá uživatel při instalaci systému.

Jen jedno připojení k databázi

- Vzor Singleton pro připojení k DB
- Připojení je obecně drahá záležitost
- Nepřipojuji se tehdy, když už spojením mám

```
// jedina trida, ktera umi vsechno s DB
```

```
class DB {  
    private static $instance = null;  
    private $db_link = null;  
    private $result = null;  
  
    private function __construct() {  
    }  
    public static function getInstance() {  
        if (self::$instance == null) {  
            self::$instance = new DB();  
            self::$instance->connect();  
        }  
        return self::$instance;  
    }  
  
    public function connect() {  
        if ($this->db_link == null) {  
            $link = mysqli_connect(DB_HOST, DB_USER, DB_PASSWD,  
DB_NAME);  
            if (!$link) {  
                throw new DBException(mysqli_errno($link));  
            }  
            mysqli_select_db($link, DB_NAME);  
            $this->db_link = $link;  
        }  
        return $this->db_link;  
    }  
}
```



```
// pokračování
```

```
public function query($sql) {  
    $this->connect();  
    $this->result = mysqli_query($this->db_link, $sql);  
    if (mysqli)  
        if (!$this->result) {  
            throw new DBException(mysqli_error($this->db_link), $sql);  
        }  
    return $this->result;  
}
```



Martin Klíma

DĚKUJI ZA POZORNOST

