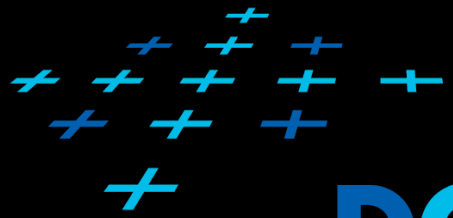


DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Bezpečnost, něco navíc

Martin Klíma

Bezpečnost

- **Bezpečnost je veličina, ne vlastnost**
  - aplikace může být víc nebo méně bezpečná
  - aplikace nemůže být absolutně bezpečná
- **Je potřebné poznat bezpečnostní hrozby a předcházet jim**
  - při návrhu aplikace
  - při implementaci

- **Nezneužívejte prosím informace z této přednášky :)**

DCGI

Ošetření vstupu

- **Uživatelský vstup není důvěryhodný:**
  - Může být částečně ztracen nebo změněn cestou na server.
  - Může být chybně zadán uživatelem.
  - Může být záměrně zadán za účelem získat kontrolu nad aplikací.
- **Každý uživatelský vstup musí být před použitím validován!**

Vstupní data

- PHP množinu super globalních proměnných pro přístup ke vstupním hodnotám:
  - $_GET – data z požadavků GET.
  - $_POST – data z požadavků POST.
  - $_COOKIE – cookie data.
  - $_FILES – uploadované soubory.
  - $_SERVER – serverová data
  - $_ENV – proměnné prostředí
  - $_REQUEST – kombinace GET/POST/COOKIE

- A k tomu jsou možná různá nastavení v
  - php.ini

$_REQUEST

- Pole $_REQUEST kombinuje vstupy z POST, GET a COOKIE

- **Může dojít ke kolizi hodnot... nepoužívat!**

Cross Site Scripting (XSS)

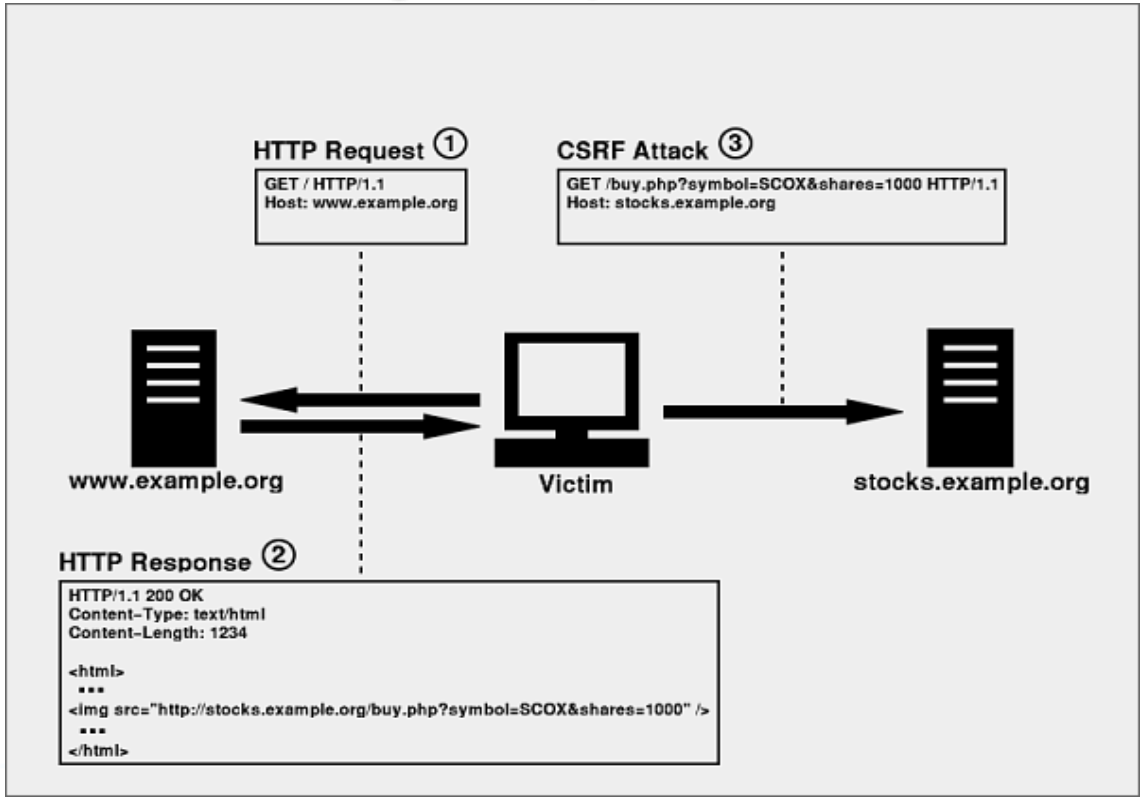
- Cross Site Scripting (XSS) je situace, kdy útočník vloží do stránky škodlivý HTML kód, který se pak zobrazí na stránce (např. ve fóru)
  - Znehodnocení stránky.
  - Převzetí session.
  - Krádeže hesla.
  - Sledování činnosti uživatelů.

Prevence XSS

- Prevence spočívá ve filtrování zobrazovaného uživatelského vstupu následovnými funkcemi:
  - `htmlspecialchars()`
    - Převede zvláštní znaky na HTML entity (‘, “, <, >, &)
  - `htmlentities()`
    - Převést všechny použitelné znaky na HTML entity.
  - `strip_tags()`
    - Odstraní z řetězce HTML a PHP tagy.
  - `mysqli_real_escape_string()`
    - Ošetření speciálně pro použití v MySQL.

Cross-Site Request Forgeries (XSRF)

- Uživatel (oběť) navštíví stránku útočníka.
- Stránka bez vedomí uživatele vykoná akci na (cílovém) serveru, na který je uživatel legálně přihlášen.



obrázek z
http://shiflett.org/articles/foiling-cross-site-attacks

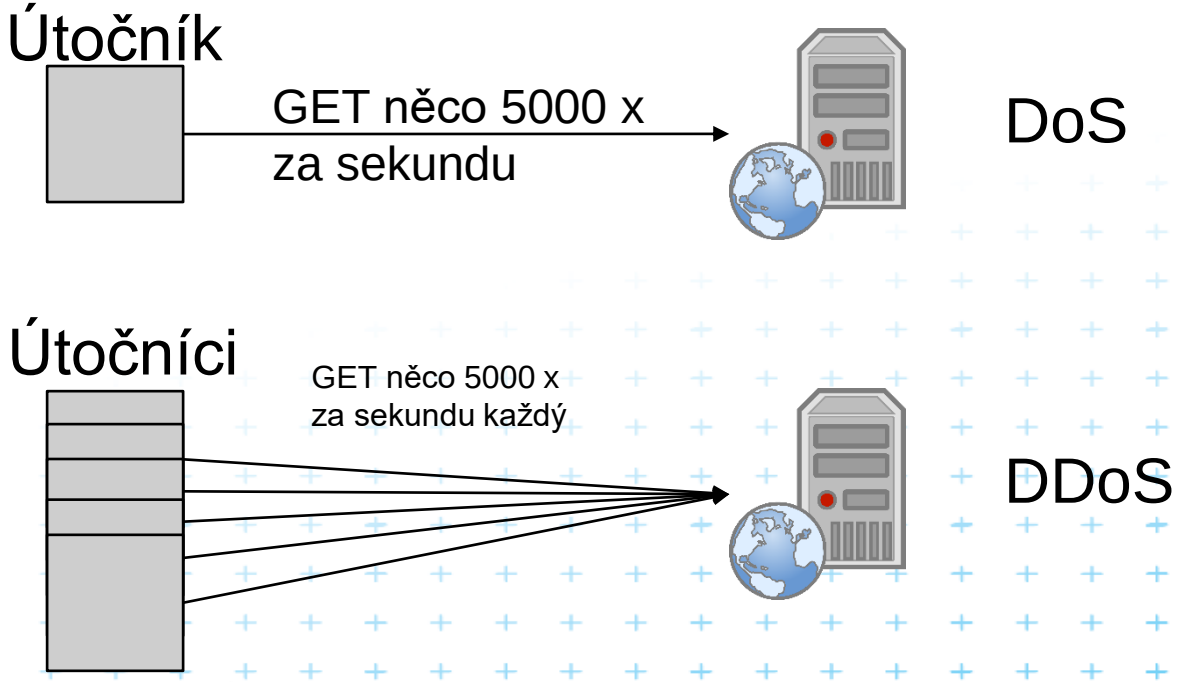
Prevence XSRF

- Používat POST pro všechny akce.
- Vyžadujte potvrzení akce
  - a. potvrzení dodatečným kliknutím
  - b. potvrzení heslem (u důležitých akcí)
- *Anti-CSRF Token*
  - *kód vygenerován u každého zobrazení formuláře*
  - *kód se zobrazí u formuláře a zároveň uloží do SESSION*
  - *při přijetí požadavku s daty formuláře porovnáme $_POST['token'] a $_SESSION['token']*
  - *timeout může být nastaven pro platnost tokenu*

DoS a DDoS útoky

- DoS = Denial of Service
- DDoS = Distributed DoS

- Cílem je zahltit server tak, aby neodpovídal na skutečně užitečný provoz.



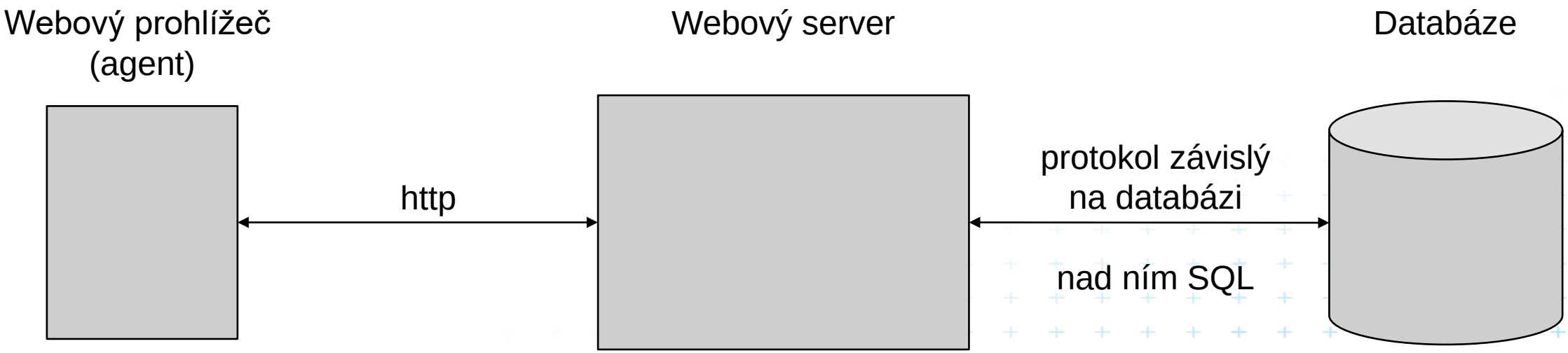
DCGI

SQL injection – úvod

Součástí většiny webových aplikací je databáze.

Většinou se jedná o Relační databázi

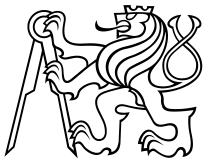
Takto vypadá typická architektura



DCGI

Protokoly RDB (Relational DataBases)

- Každý výrobce má svoji implenetaci
  - bohužel nedošlo ke sjednocení
- Došlo ale ke sjednocení (+-) dotazovacího jazyka
- Jazyk **SQL** zahrnuje několik podjazyků
  - Structured query language
    - DQL, DML = SELECT, INSERT, UPDATE, DELETE | Dotazování a práce s daty
    - DDL = CREATE, ALTER, DROP | Definice datových struktur
    - DCL = GRANT, REVOKE | Definice práv pro přístup a manipulaci s daty



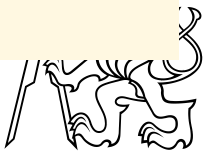
Nutné kroky při práci s DB

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

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

Implementace pokr.

```
// 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);
```

SQL Injection

- *SQL injection* se podobá XSS.
- Nekontrolovaný uživatelský vstup je použit při vytváření SQL dotazu.
- Pomocí vloženého dodatečného SQL dotazu do vstupu může útočník:
  - smazat, vložit nebo měnit data
  - zapříčinit DoS

DCGI

SQL Injection

zkuste url:

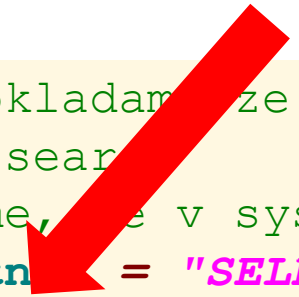
sql_injection.php?max=aaa

```php
// 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'%

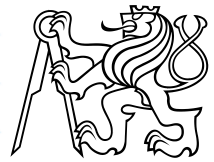


```php
// predpokladam, ze uzivatel zadal vyhledavaci retezec nazvu
// param sear
// rekneme, ze v systemu jsou dve role: 1: admin, 2: obyc uzivatel
$sql_spatn = "SELECT * FROM zbozi WHERE Priv = 2 AND Nazev like
'%".$_GET['search']."'";

$search = mysql_real_escape_string($_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

Prevence SQL injection: SQL escaping

- PHP API pro databáze má často speciální funkce pro úpravu vstupu:
  - MySQL
    - `mysql_escape_string()`
    - `mysqli_real_escape_string()`
  - PostgreSQL
    - `pg_escape_string()`
    - `pg_escape_bytea()`
  - SQLite
    - `sqlite_escape_string()`



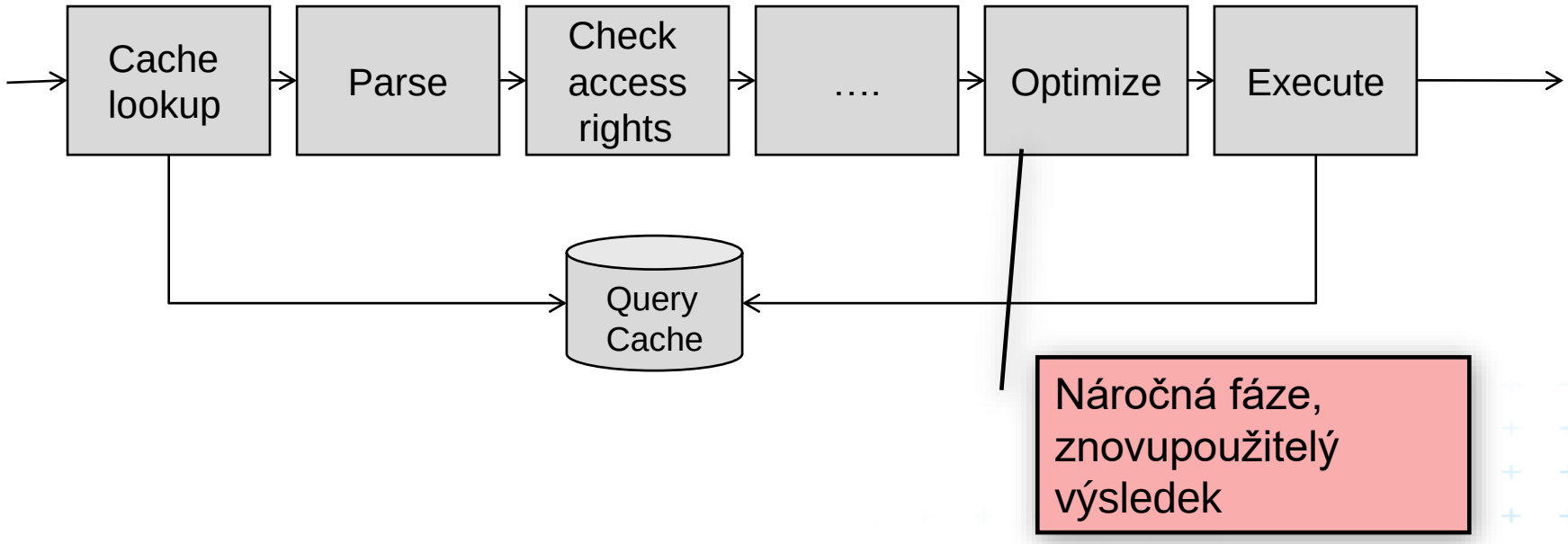
Prevence SQL injection: Prepared Statements

- *Prepared statements* slouží k zabezpečení a optimalizaci vykonávaných dotazů.

- SQL “zkompiluje” dotaz a pak při každém vykonání jenom nahrazuje hodnoty přeměnných.
  - Vyšší výkon – jedno kompilování na dotaz.
  - Vyšší bezpečnost, vložená data nejsou považována za další dotaz.

Zpracování dotazu databází

Typické fáze zpracování



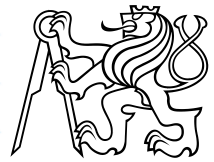
PreparedStatement provádí optimalizaci jednou, dále posílá data do fáze execute

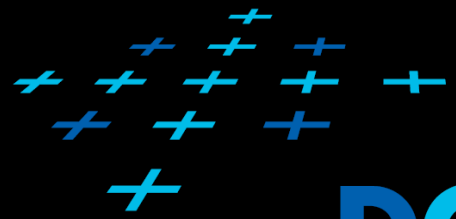
DCGI

```php
$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();
```

Prevence SQL injection: SQL escaping

- PHP API pro databáze má často speciální funkce pro úpravu vstupu:
  - MySQL
    - `mysql_escape_string()`
    - `mysqli_real_escape_string()`
  - PostgreSQL
    - `pg_escape_string()`
    - `pg_escape_bytea()`
  - SQLite
    - `sqlite_escape_string()`





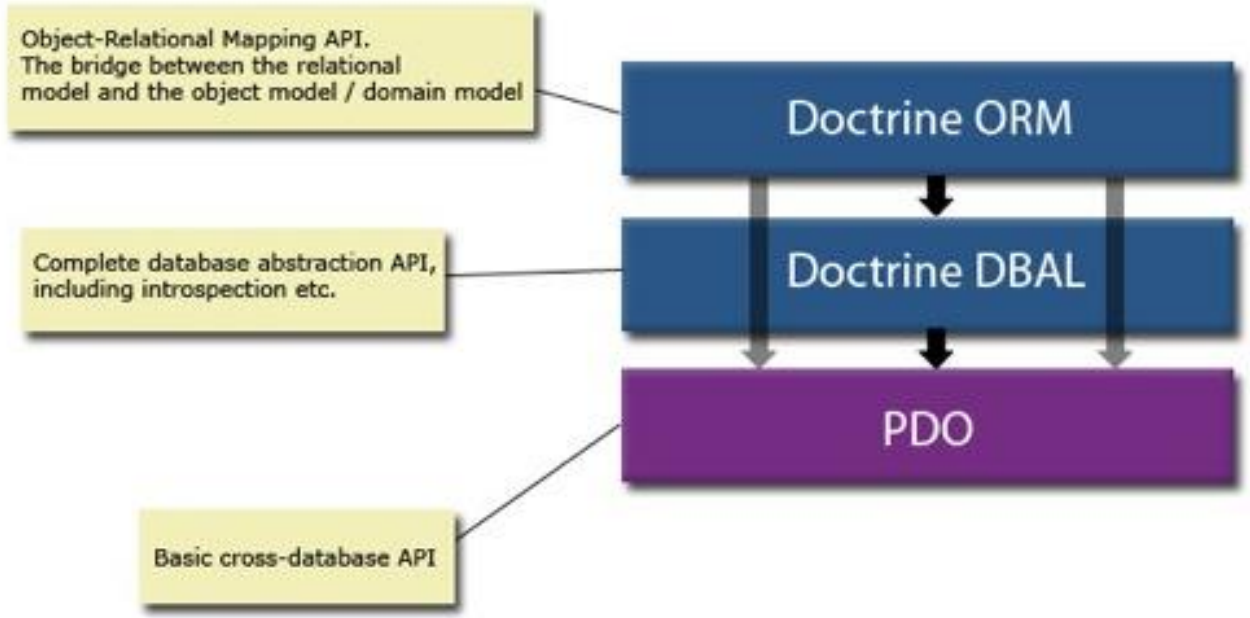
DCGI

KATEDRA POČÍTAČOVÉ GRAFIKY A INTERAKCE

Doctrine ORM

aneb jak se to dělá opravdu

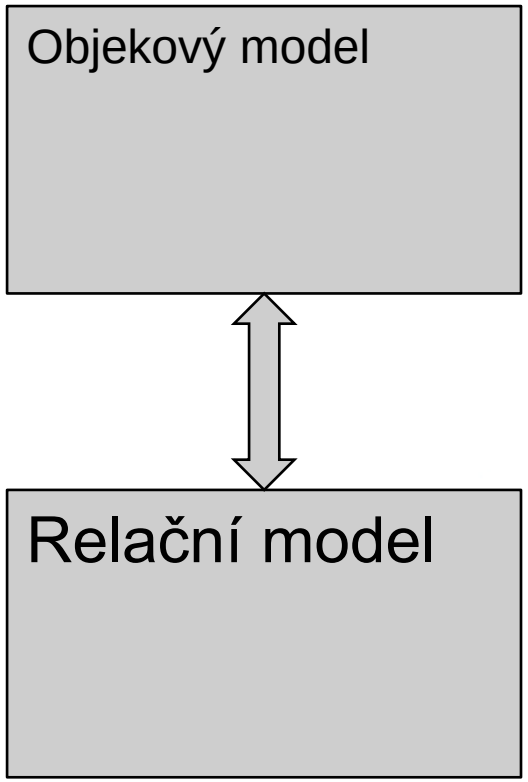
Doctrine – co to je ORM



DCGI

Cíl – mapování objektového modelu a ER

- Object – Relational Mapping = ORM



Ukázka – navázání spojení

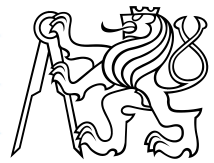
bootstrap

```
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);
```

Lazy verze

```
 // 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');
```

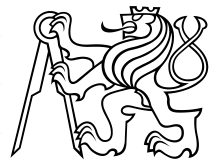


Ukázka použití

```
$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);
```

Výsledek

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

)
```



Definice dat, toto je to samotné mapování O na R

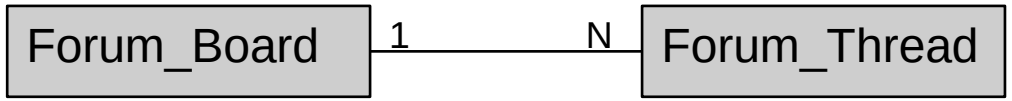
Model

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

Použití

```php
// test.php

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

Relační závislosti



Model Forum_Board

```php
// 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'
            )
        );
    }
}
```

Model Forum_Thread

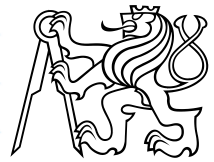
```php
// models/Forum_Thread.php

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

```php
$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();
```

DCGI



Práce s modelem

```php
// test.php
$user = new User();
$user['username'] = 'jwage';
$user['password'] = 'changeme';

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


$user->save();
```

Ekvivalentní

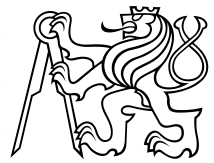
Ekvivalentní

DCGI

Dotazování a další operace

```php
// 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();
```

DCGI



Dotazy?

DĚKUJI ZA POZORNOST

DCGI