



Web Adv

PHP: PDO

DE HOGESCHOOL MET HET NETWERK

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

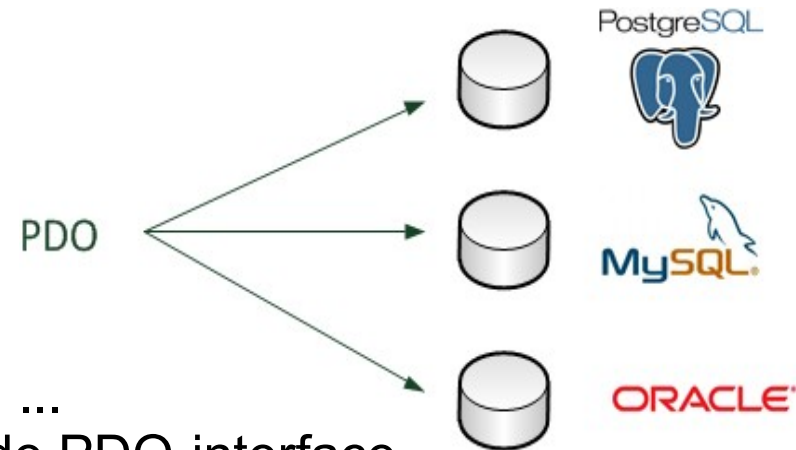


PDO (Inleiding)



- PHP Data Objects
- PHP extension sinds PHP5 (2004)
(geschreven in C++)
- Data access abstraction layer:

dezelfde code voor de interactie
met PostgreSQL, MySQL, Oracle, ...
(op voorwaarde dat de driver die de PDO-interface
implementeert geïnstalleerd is)



alle configuratie in de [connection-string](#)

```
$pdo=new PDO( "mysql:host=localhost;dbname=test",  
              $user,$pass );
```

alternatief = aparte methodes voor elke DB

- | | | | |
|-------------------|-----------------|---------------|-----|
| - pg_connect, | pg_execute, | pg_fetch_all, | ... |
| - mysqli_connect, | mysqli_execute, | mysqli_fetch | ... |
| - oci_connect, | oci_execute, | oci_fetch | ... |

PDO (Overzicht)



(1) - connectie maken

- interactie met databank

- methode **exec** om gegevens te wijzigen
(INSERT, UPDATE, DELETE)

- methode **query** om gegevens op te halen
(SELECT)

- connectie sluiten

(2) Preparedstatements

(3) Transactions

PDO (1. Connectie maken en sluiten)



```
$user='root';
$password='...';

$pdo=null;
try {
    // connectie maken met databank test

    $pdo = new PDO('mysql:host=localhost;dbname=test',
        $user, $$password);

    // gegevens ophalen of wijzigen

} catch (PDOException $e) {
    print 'Exception!: ' . $e->getMessage();
}
// connectie sluiten
$pdo = null;
```

exec / query
zie volgende slides

PDO (1. Connectie maken en sluiten)



PDOException wordt altijd opgeworpen bij misse waarde voor user, password of databasname.

Bijkomende exceptions, warnings afhankelijk van de **errormode**:

ERRMODE_SILENT (default)

Geen verdere exceptions/warnings

ERRMODE_WARNING

Warning bij verdere fouten (bijvoorbeeld fout in SQL-query)

ERRMODE_EXCEPTION

Exceptions bij verdere fouten

Errormode wijzigen via **setAttribute**:

```
$pdo = new PDO('mysql:host=localhost;dbname=test',  
               $user, $pass);  
$pdo->setAttribute( PDO::ATTR_ERRMODE,  
                   PDO::ERRMODE_EXCEPTION );
```

PDO (1. Exec)



exec

- gegevens wijzigen
- return-value = aantal aangepaste rijen

```
...
$pdo=null;
try {
    $pdo = new PDO('mysql:host=localhost;dbname=cdcol',
                    $user, $pass);
    $pdo->setAttribute( PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION );

    $nr = $pdo->exec("DELETE FROM cds WHERE ".
                    "titel LIKE 'T%'");
    print("$nr rows modified");
} catch (PDOException $e) {
    print 'Exception!: ' . $e->getMessage();
}
$pdo = null;
```

PDO (1. Query)



query

- gegevens opvragen
- returnvalue = 'resultset' (PDOStatement)

```
$pdo=null;
try {
    $pdo = new PDO('mysql:host=localhost;dbname=cdcol',
                  $user, $password);
    $pdo->setAttribute( PDO::ATTR_ERRMODE,
                      PDO::ERRMODE_EXCEPTION );
    $stmt = $pdo->query('SELECT * from cds');
    $i=0;
    while($row = $stmt->fetch()) {
        print("$i\n");
        print_r($row);
        $i++;
    }
} catch (PDOException $e) {
    print 'Exception!: ' . $e->getMessage();
}
$pdo=null;
```

PDO (1. Query)



```
$stmt = $pdo->query('SELECT * from cds');
```

resultset, **cursor** duidt 1 rij aan



titel	Interpret	Jahr	Id
Beauty	Ryuichi Sakamoto	1990	1
Goodbye Country (Hello Nightclub)	Groove Armada	2001	4
Glee	Bran Van 3000	1997	5

```
while($row = $stmt->fetch()) {  
    print_r($row);  
}
```

fetch schuift de cursor 1 positie op.

- de geselecteerde rij wordt teruggegeven
- voorbij de laatste rij return-value = false

fetchAll: alle gegevens uit het resultset worden teruggegeven als 2D array

PDO (1. Query)



Default fetch-mode = **PDO::FETCH_BOTH**

De fetched array heeft zowel de naam als de index van de kolom als key

naam

index

0

Array
(

▶ [titel] => Beauty

▶ [0] => Beauty

[interpret] => Ryuichi Sakamoto

[1] => Ryuichi Sakamoto

[jahr] => 1990

[2] => 1990

[id] => 1

[3] => 1

)

1

Array
(

[titel] => Goodbye Country (Hello Nightclub)

[0] => Goodbye Country (Hello Nightclub)

[interpret] => Groove Armada

[1] => Groove Armada

[jahr] => 2001

PDO (1. Query)



Fetch-mode wijzigen via setFetchMode

```
$stmt = $pdo->query('SELECT * from cds');  
$stmt->setFetchMode(PDO::FETCH_ASSOC);
```

```
0  
Array  
(  
    [titel] => Beauty  
    [interpret] => Ryuichi Sakamoto  
    [jahr] => 1990  
    [id] => 1  
)
```

```
1  
Array  
(  
    [titel] => Goodbye Country (Hello Nightclub)  
    [interpret] => Groove Armada  
    [jahr] => 2001  
    [id] => 4  
)
```

```
2  
Array  
(  
    [titel] => Glee
```

De fetched array heeft enkel
de naam van de kolom als
key
- FETCH_NUM
...

PDO (1. Query)



rowCount	aantal rijen in resultset
columnCount	aantal kolommen in resultset
getColumnMeta	metadata over kolom

```
if ($stmt->rowCount() > 0) {
    $columnNames=array();
    for ($i = 0; $i < $stmt->columnCount(); $i++) {
        $col = $stmt->getColumnMeta($i);
        $columnNames[] = $col['name'];
    }

    print("<table>");
    print('<tr><th>'.implode('</th><th>',$columnNames).
        '</th></tr>');

    while($row = $stmt->fetch()) {
        print('<tr><td>'.implode('</td><td>',$row).
            '</td></tr>');
    }
    print("</table>");
}
```

PDO (2. Prepared statements)



Prepared statement

- via de methode **prepare** wordt de prepared statement doorgestuurd naar de databank
- via de methode **execute** wordt de prepared statement uitgevoerd
bij de executie kunnen parameters meegegeven worden
- named en unnamed parameters
- voordelen van prepared statements:
 - efficiëntie wanneer query meerdere keren uitgevoerd moet worden
 - parameters worden geëscaped (**sql-injection**)

PDO (2. Prepared statements)



Unnamed parameters

- **prepare** maakt de prepared statement klaar
- binnen de query worden parameters aangeduid als **?**
- via **bindParam** wordt het eerste **?** verbonden met \$titel, ...
- **execute** voert de preparedstatement uit

```
$stmt = $pdo->prepare("INSERT INTO cds ".  
    "(titel, interpret, jahr) VALUES (?, ?, ?);");  
$stmt->bindParam(1, $titel, PDO::PARAM_STR);  
$stmt->bindParam(2, $interpret, PDO::PARAM_STR);  
$stmt->bindParam(3, $year, PDO::PARAM_INT);  
$nr=$stmt->execute();  
print("$nr rows modified");
```

PDO (2. Prepared statements)



Named parameters

- **prepare** maakt de prepared statement klaar
- binnen de query worden parameters aangeduid als **:naam**
- via **bindParam** wordt de parameter :id verbonden met een variabele
- **execute** voert de preparedstatement uit


```
$id = 1;  
$stmt=$pdo->prepare('SELECT * FROM cds WHERE id = :id');  
$stmt->bindParam(':id', $id, PDO::PARAM_INT);  
$stmt->setFetchMode(PDO::FETCH_ASSOC);  
$stmt->execute();  
var_dump($stmt->fetch());
```

PDO (2. SQL-injection)



Aanval door de ingave van kwaadwillige code in een SQL commando.

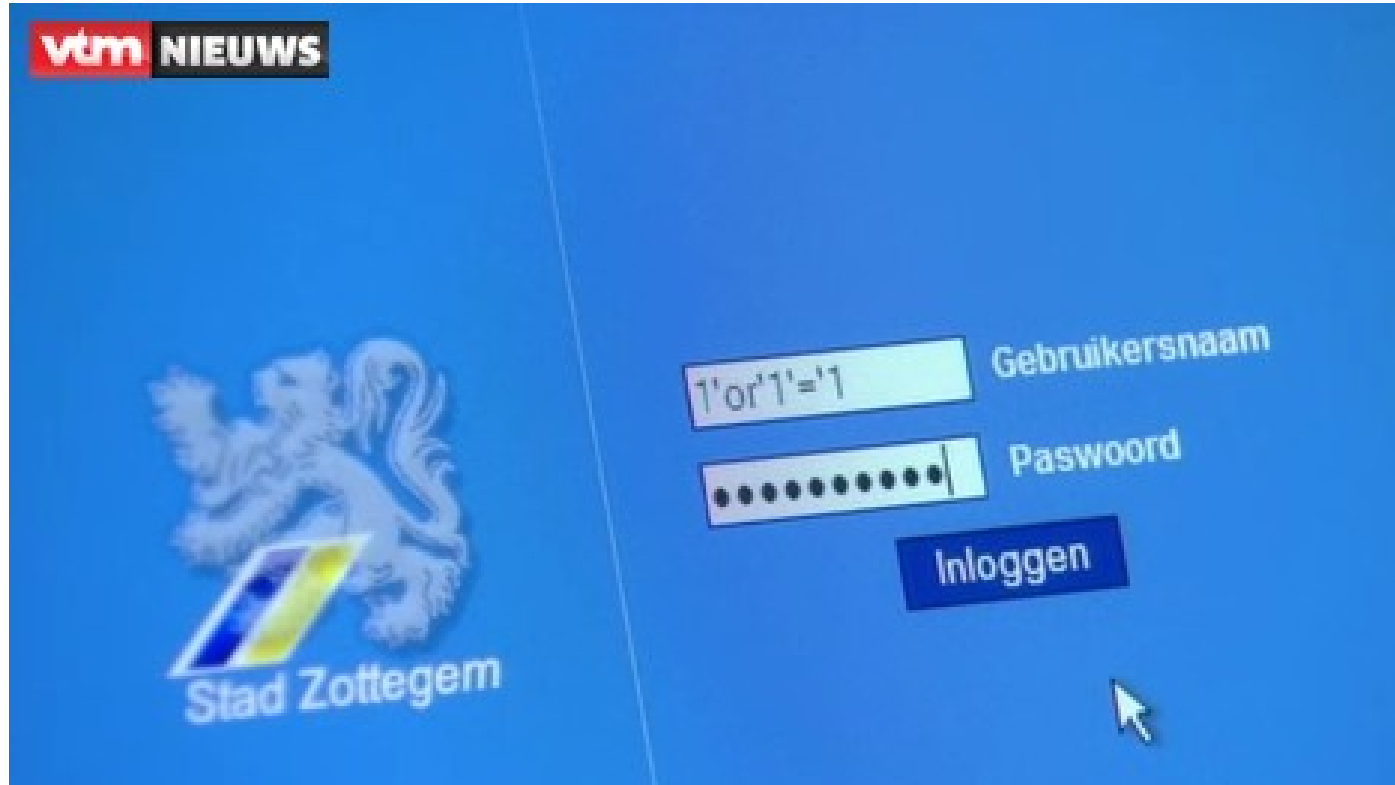


OWASP Top 10 – 2013 (New)	
	A1 – Injection
	A2 – Broken Authentication and Session Management
	A3 – Cross-Site Scripting (XSS)

PDO (2. SQL-injection)



September 2013



PDO (2. SQL-injection)



Databank met tabel users

id	username	password
1	jan	270f5ebbf3a30ec831ae1fe4ae69519fa

md5('jan_p')

Gewoon gebruik

invoer.html

id:

Password:



id en password worden
rechtstreeks in
commando geplaatst

verwerk.php

```
SELECT * FROM users WHERE id = 1 AND password = MD5('jan_p')
```

Ingelogd als 1 jan

PDO (2. SQL-injection)



```
$id = $_POST['user'];
$pw = $_POST['password'];

$query = 'SELECT * FROM users WHERE id = ' . $id . '
        AND password = MD5(\'' . $pw . '\') ';

$stmt = $pdo->query($query);
echo '<pre>' . $stmt->queryString . '</pre>';

$stmt->setFetchMode(PDO::FETCH_ASSOC);

$row=$stmt->fetch();
if($row !== false){
    echo "<p>Ingelogd als " . $row['id'] . ' ' .
        $row['username'] . " </p>";
}
else{
    echo "<p>Login gefaald</p>";
}
```

PDO (2. SQL-injection)



Aanval 1

invoer.html

user:

Password:



id en password worden
rechtstreeks in
commando geplaatst

verwerk.php

```
SELECT * FROM users WHERE id = 1 # AND password = MD5('')
```

Ingelogd als 1 jan



Ingelogd als 1 jan

PDO (2. SQL-injection)



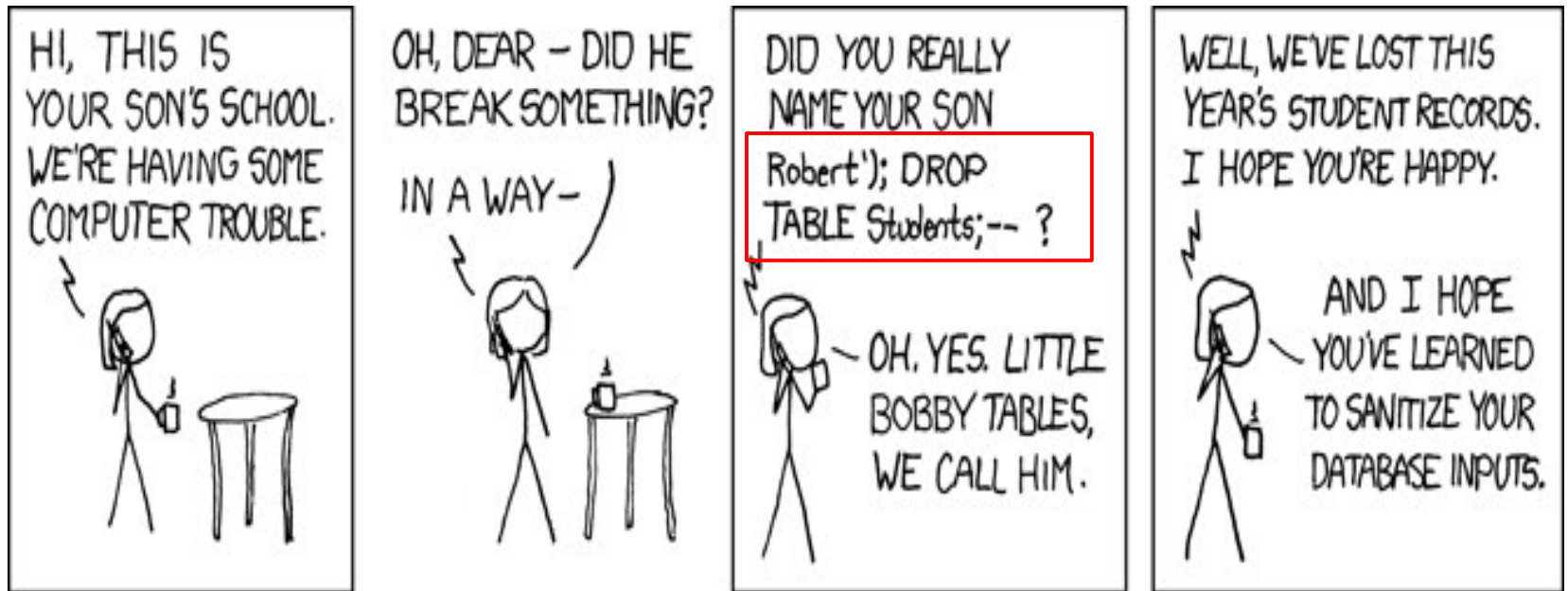
```
$id = $_POST['user'];  
$pw = $_POST['password'];  
  
$query = 'SELECT * FROM users WHERE id = ? '.  
         'AND password = MD5(?) ';  
  
$stmt=$pdo->prepare($query);  
  
$stmt->bindParam(1, $id , PDO::PARAM_INT);  
$stmt->bindParam(2, $pw , PDO::PARAM_STR);  
  
$stmt->execute();  
...
```

PDO::PARAM_INT

(int) "1 #" → 1

(int) "1 OR 1 = 1" → 1

PDO (2. SQL-injection)



PDO (3. Transactions)



Transactie

- meerdere sql-statements als 1 geheel uitvoeren
- fout-afhandeling als er iets misgaat

- bijvoorbeeld bankoverschrijving

x geld rekening 1 → rekening 2

mocht er iets misgaan bij het afnemen van rekening 1
plaats geen geld op rekening 2

mocht er iets mislopen bij het plaatsen op rekening 2
zet x geld terug op rekening 1

PDO (3. Transactions)



beginTransaction start de transactie (autocommit wordt afgezet)

commit voer de transactie uit

rollback de transactie wordt ongedaan gemaakt

```
try{  
  
    $pdo->beginTransaction();  
  
    // interactie met de databank  
  
    $pdo->commit();  
  
} catch (Exception $e) {  
    $pdo->rollBack();  
}
```


PDO (3. Transactions)



```
$from=1001;
$to=1002;
$amount=23.4;
try{
    $pdo->beginTransaction();

    $sql = 'SELECT amount FROM accounts WHERE id=:from';
    $stmt = $pdo->prepare($sql);
    $stmt->bindParam(':from', $from, PDO::PARAM_INT );
    $stmt->execute();

    $originalAmount = (double)$stmt->fetchColumn();
    if($originalAmount < $amount){
        throw new Exception();
    }
}
```

PDO (3. Transactions)



```
$sql = 'UPDATE accounts '.  
      'SET amount = amount - :amount '.  
      'WHERE id = :from';  
$stmt = $pdo->prepare($sql);  
$stmt->bindParam(':from', $from, PDO::PARAM_INT );  
$stmt->bindParam(':amount', $amount, PDO::PARAM_INT );  
$stmt->execute();
```

```
$sql = 'UPDATE accounts '.  
      'SET amount = amount + :amount '.  
      'WHERE id = :to';  
$stmt = $pdo->prepare($sql);  
$stmt->bindParam(':to', $to, PDO::PARAM_INT );  
$stmt->bindParam(':amount', $amount, PDO::PARAM_INT );  
$stmt->execute();
```

PDO (3. Transactions)



```
$pdo->commit();  
echo 'succes';  
  
} catch (Exception $e) {  
    echo 'failure';  
    $pdo->rollBack();  
}
```

PDO (Besluit)



PDO = data access abstraction layer

dezelfde code voor verschillende databanken
alle configuratie via de connection-string

connectie maken via de constructor van PDO

query (SELECT)
resultset, fetch

exec (INSERT, UPDATE, DELETE)

prepared statements
named & unnamed parameters
bindParam
execute

transactions (beginTransaction, commit, rollBack)

Oefening 1



Maak de toepassing die getoond wordt in onderstaande figuur. De toepassing bestaat uit `ingave.php` en `verwerking.php`. In `ingave.php` worden alle databanken opgehaald uit de MySQL-server. De databanken worden in een drop-down menu geplaatst. Ook kan via een textarea een SQL-query uitgevoerd worden. Via `verwerking.php` wordt de query uitgevoerd op voorwaarde dat er geen drop, delete of truncate in staat. Het resultaat wordt ook getoond.

Databank

SQL-QUERY (drop, delete en truncate niet toegelaten)

```
select * from werknemers
```

```
query = select * from werknemers
databank = voorbeeld_a
1 Martin   George 3
2 Feynman  Richard 2
3 Graves   Robert  2
```

Hint: De SQL query `SHOW DATABASES` kan gebruikt worden om de namen van alle databanken te vinden.

Hint: De controle of drop, delete, truncate in de SQL-query staat kan via de functie `stristr`:

<http://php.net/manual/en/function.stristr.php>

Oefening 2



Maak een databank met daarin de tabel gebruikers. Elke gebruiker heeft een id een naam.

Maak een PHP-toepassing waarmee je een overzicht van de gebruikers krijgt (overzicht.php). Ook worden de volgende acties ondersteund: gebruikers kunnen verwijderd worden (verwijder.php), gebruikers kunnen aangepast worden (wijzig.php) en gebruikers kunnen aangemaakt worden (toevoegen.php).

Extra Oefening

Zorg ervoor dat bij het verwijderen bevestiging gevraagd wordt (bevestig.php). Je kan gebruik maken van hidden input om gegevens te bewaren.

PDO (Bronnen)



<http://www.pluralsight.com/courses/build-dynamic-web-sites-mysql-php>

▶ Accessing a Database from PHP			
Introducing PDO	✓	🔖	1:55
Connecting to the Database	✓	🔖	6:38
Book Search Demonstration	✓	🔖	10:08

▶ Doing More with the Database			
Using Prepared Statements	✓	🔖	12:04
Non-queries and Summary Functions	✓	🔖	8:14



<http://www.pluralsight.com/courses/exercise-files/build-dynamic-web-sites-mysql-php>

PDO (Bronnen)



<http://www.phpro.org/tutorials/Introduction-to-PHP-PDO.html>



<http://php.net/manual/en/pdo.connections.php>

<http://php.net/manual/en/pdo.exec.php>

<http://php.net/manual/en/pdo.query.php>

<http://php.net/manual/en/pdostatement.fetch.php>

<http://php.net/manual/en/pdo.prepared-statements.php>

<http://php.net/manual/en/pdo.transactions.php>