PDO in PHP

Key concepts

PDO, query, exec, statement, resultset, prepared statement, SQL-injectie, transaction

Alternatieve bronnen

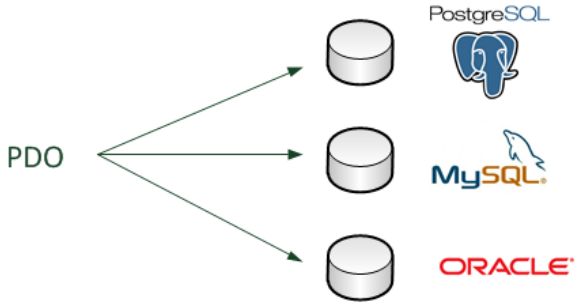
https://phpro.org/tutorials/Introduction-to-PHP-PDO.html

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

- Accessing a Database from PHP
- Doing More with the Database

1. PDO inleiding

PDO (PHP Data objects) is een PHP-extensie geschreven in C++ in 2004. PDO is een data abstraction layer die het mogelijk maakt om met dezelfde PHP-code te interageren met verschillende soorten databanken. Voor elk van deze databanken is er een driver voorzien die de PDO-interface implementeert.



Typisch wordt de onderstaande code gebruikt om de connectie met een databank te maken en te sluiten. Via de aanroep van de functie setAttribute wordt ervoor gezorgd dat exceptions gebruikt worden (http://php.net/manual/en/pdo.setattribute.php). Op de positie van de 3 puntjes gebeurt de interactie met de databank. De interactie kan bestaan uit het ophalen van gegevens uit de databank (query) of het wijzigen van de databank (exec).

app.php

```
<?php
$user='root';
$password='root';
$database='cdcol'
$pdo=null;
try {
    $pdo = new PDO( "mysql:host=localhost;dbname=$database",
                    $user, $password );
    $pdo->setAttribute( PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION );
    ...
} catch ( PDOException $e ) {
    print 'Exception!: ' . $e->getMessage();
}
$pdo = null;
```

2. Exec

Via de functie exec wordt een wijziging doorgevoerd. De terugkeerwaarde is het aantal aangepaste rijen.

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

app.php

```
<?php
$user='root';
$password='root';
$database='person';
$pdo=null;
try {
    $pdo = new PDO( "mysql:host=localhost;dbname=$database",
                    $user, $password );
    $pdo->setAttribute( PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION );
    $numberRows = $pdo->exec("DELETE FROM person WHERE ".
                                    "name LIKE 's%'");
    print("$numberRows rows modified");

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

2. Query

Via de functie query worden gegevens opgevraagd. De terugkeerwaarde van query is een PDOStatement die een resultset bevat.

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

Op het resultset kan de methode fetch aangeroepen worden om een volgende rij te selecteren wanneer voorbij de laatste rij gegaan wordt is de terugkeerwaarde False.

http://php.net/manual/en/pdostatement.fetch.php
http://php.net/manual/en/pdostatement.setfetchmode.php

app.php

```
<?php
$user='root';
$password='root';
$database='cdcol';
$pdo=null;
try {
    $pdo = new PDO( "mysql:host=localhost;dbname=$database",
                    $user, $password );
    $pdo->setAttribute( PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION );
    $statement = $pdo->query('SELECT * from cds');
    $statement->setFetchMode(PDO::FETCH_ASSOC);
    while($row = $statement->fetch()) {
        print_r($row);
    }
} catch ( PDOException $e ) {
    print 'Exception!: ' . $e->getMessage();
}
$pdo = null;
```

Via de methode fetchAll worden alle gegevens in één keer opgehaald en in een 2D array geplaatst.

http://php.net/manual/en/pdostatement.fetchall.php

Verder kunnen er nog gegevens over het resultset opgevraagd worden via

rowCount aantal rijen in resultset
http://php.net/manual/en/pdostatement.rowcount.php

columnCount aantal kolommen in resultset
http://php.net/manual/en/pdostatement.columncount.php

getColumnMeta metadata over kolom
http://php.net/manual/en/pdostatement.getcolumnmeta.php

app.php

```
<?php
$user='root';
$password='root';
$database='persondb';
$pdo=null;
try {
    $pdo = new PDO( "mysql:host=localhost;dbname=$database",
        $user, $password );
    $pdo->setAttribute( PDO::ATTR_ERRMODE,
        PDO::ERRMODE_EXCEPTION );
    $statement = $pdo->query('SELECT * from person');
    $statement->setFetchMode(PDO::FETCH_ASSOC);

    if($statement->rowCount() > 0){
        $columnNames=[];
        for ($i = 0; $i < $statement->columnCount(); $i++) {
            $columnData = $statement->getColumnMeta($i);
            $columnName = $columnData['name'];
            $columnNames[] = $columnName;
        }

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

        while($row = $statement->fetch()) {
            print('<tr><td>'.implode('</td><td>',$row).
                '</td></tr>');
        }
        print("</table>");
    }
} catch ( PDOException $e ) {
    print 'Exception!: ' . $e->getMessage();
}
$pdo = null;
```

3. Prepared statements

Buiten gewone statements kan er ook gewerkt worden met prepared statements. Via de methode prepare wordt een preparedstatement doorgestuurd naar de databank. In de prepared statement staan een aantal (named of unnamed) parameters die via de functie bindParam een waarde krijgen. Ten slotte zorgt de functie execute ervoor dat de preparedstatement uitgevoerd wordt.

http://php.net/manual/en/pdo.prepared-statements.php
http://php.net/manual/en/pdo.prepare.php
http://php.net/manual/en/pdostatement.bindparam.php
http://php.net/manual/en/pdostatement.execute.php

De voornaamste voordelen van het gebruik van prepared statements zijn betere efficiëntie wanneer statements meerdere keren uitgevoerd moeten worden en het voorkomen van SQL-injectie.

In onderstaand voorbeeld wordt gewerkt met unnamed parameters. Deze worden voorgesteld met een vraagteken. De methode bindParam specifieert over het hoeveelste vraagteken het gaat in het SQL-commando, wat de waarde is van de parameter en hoe de waarde geïnterpreteerd moet worden.

app.php

```
<?php
$user='root';
$password='root';
$database='cdcol';
$pdo=null;
try {
    $pdo = new PDO( "mysql:host=localhost;dbname=$database",
                    $user, $password );
    $pdo->setAttribute( PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION );
    $statement = $pdo->prepare('INSERT INTO cds '.
                 '(title, interpreter, year) VALUES (?, ?, ?);');
    $title='...';
    $interpreter='...';
    $year=2012
    $statement->bindParam(1, $titel, PDO::PARAM_STR);
    $statement->bindParam(2, $interpret, PDO::PARAM_STR);
    $statement->bindParam(3, $year, PDO::PARAM_INT);
    $numberRows=$statement->execute();
    print("$numberRows rows modified");
} catch ( PDOException $e ) {
    print 'Exception!: ' . $e->getMessage();
}
$pdo = null;
```

In het tweede voorbeeld wordt gewerkt met named parameters. De parameters krijgen nu een naam die begint met een dubbelpunt. De toekenning via bindParam gebeurt nu aan de hand van deze naam.

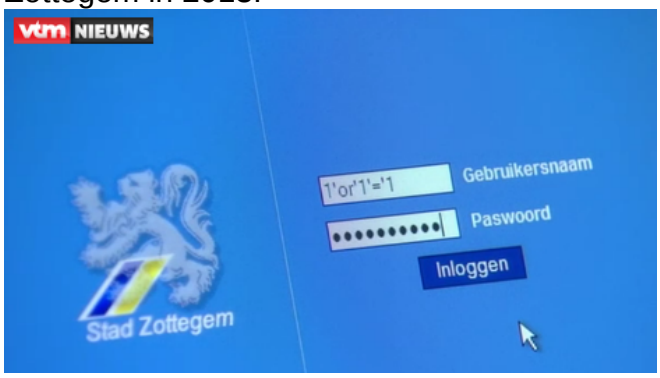
app.php

```
<?php
$user='root';
$password='root';
$database='cdcol'
$pdo=null;
try {
    $pdo = new PDO( "mysql:host=localhost;dbname=$database",
                    $user, $password );
    $pdo->setAttribute( PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION );
    $id = 1;
    $statement=$pdo->prepare('SELECT * FROM cds WHERE id = :id');
    $statement->bindParam(':id', $id, PDO::PARAM_INT);
    $statement->setFetchMode(PDO::FETCH_ASSOC);
    $statement->execute();
    var_dump($statement->fetch());
} catch ( PDOException $e ) {
    print 'Exception!: ' . $e->getMessage();
}
$pdo = null;
```

4. SQL-injection

SQL-injection is een van de meest courante manieren om een website aan te vallen. De aanval is mogelijk omdat invoer zonder controle rechtstreeks in een een SQL-commando geplaatst wordt.

Onderstaand voorbeeld toont een SQL-injectie aanval op de website van de gemeente Zottegem in 2013.



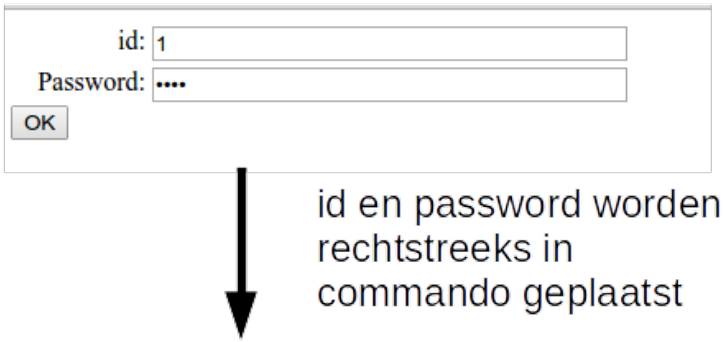
Eenvoudig voorgesteld ziet een SQL-injectie aanval er als volgt uit. In invoer.html wordt id en pasdword gevraagd in een formulier. Deze gegevens worden in verwerk.php rechtstreeks in een SQL-commando geplaatst.

verwerk.php

```
...
$id = $_POST['user'];
$password = $_POST['password'];
$query = 'SELECT * FROM users WHERE id = '. $id .
      ' AND password = MD5(\''. $password  .'\') ';
$statement = $pdo->query($query);
echo '<pre>' . $statement>queryString . '</pre>';
$statement>setFetchMode(PDO::FETCH_ASSOC);
$row=$statement>fetch();
if($row !== false){
    echo "<p>Ingelogd als ". $row['id']. ' ' .
                $row['username'].  "</p>";
} else {
    echo "<p>Login gefaald</p>";
}
```

In onderstaande figuur wordt het verwachte gebruik geïllustreerd. De gebruiker geeft voor id een getal en zijn paswoord in.

invoer.html



verwerk.php

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

Ingelogd als 1 jan

Een eerste aanval kan door 1 # in te geven voor id. Alles na # in het gegenereerde SQL-commando wordt genegeerd en er wordt ingelogd als de gebruiker met id 1.

invoer.html

id en password worden rechtstreeks in commando geplaatst

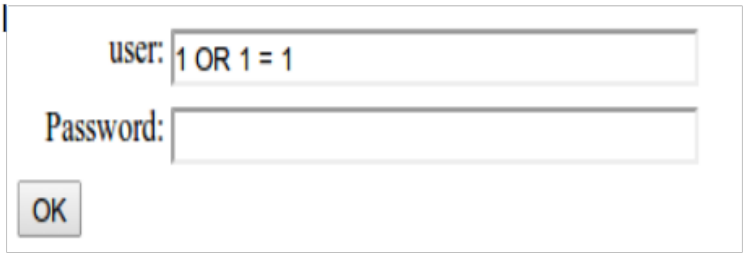
verwerk.php

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

Ingelogd als 1 jan

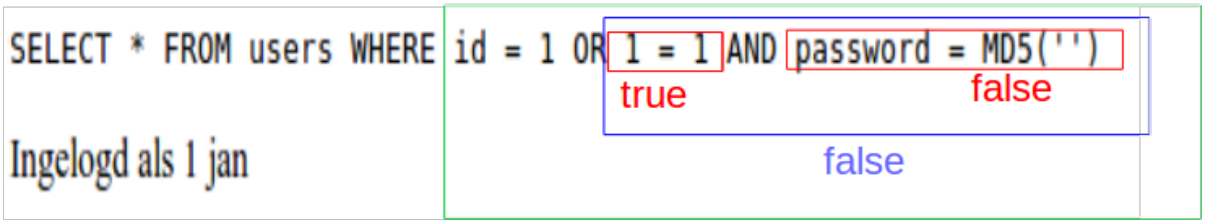
Een tweede aanval bestaat in de ingave van 1 OR 1 = 1 voor het id.

invoer.html



id en password worden rechtstreeks in commando geplaatst

verwerk.php



voor id=1: true OR false=true

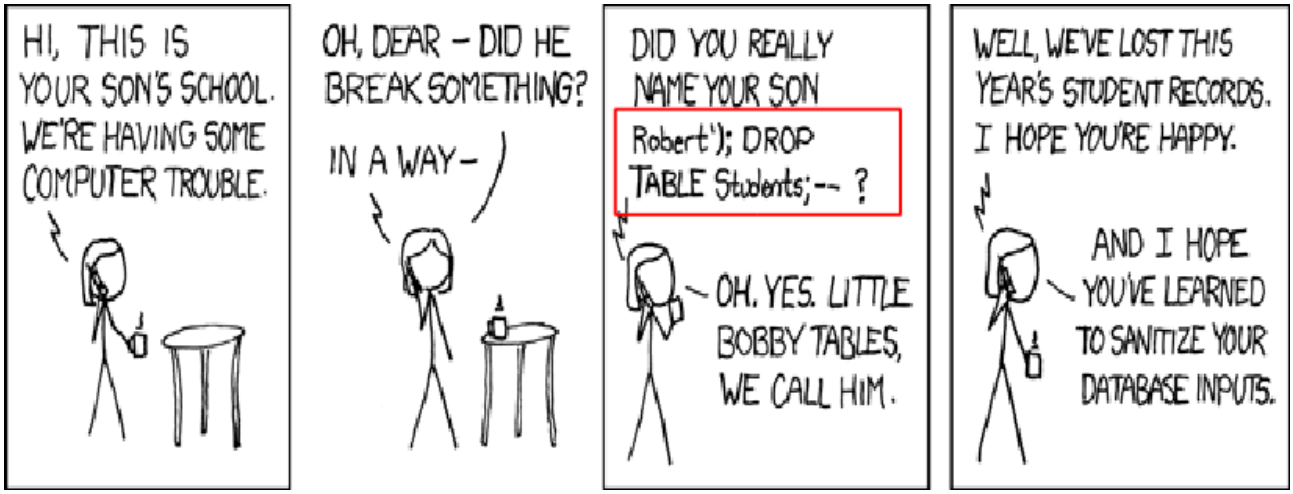
De aanval kan heel eenvoudig vermeden worden via prepared statements. Er wordt bij de eerste parameter aangegeven dat er met een int-waarde gewerkt wordt (bindParam). Bijgevolg wordt alle invoer die niet omgezet kan worden naar data-type int uit de invoer geweerd.

verwerk.php

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

Verder is het ook de taak van de programmeur om voldoende validatie op input in te bouwen: er moet altijd gecontroleerd worden of de invoer correct is.

https://www.explainxkcd.com/wiki/index.php/327:_Exploits_of_a_Mom
https://www.explainxkcd.com/wiki/index.php/Little_Bobby_Tables



5. Transactions (extra)

Bij een transactie worden meerdere SQL-statements als één geheel uitgevoerd. Mocht er bij het uitvoeren van de transactie een fout optreden dan wordt de transactie in zijn geheel ongedaan gemaakt (rollback).

beginTransaction start de transactie (autocommit wordt afgezet)
http://php.net/manual/en/pdo.begintransaction.php

commit voer de transactie uit
http://php.net/manual/en/pdo.commit.php

rollback de transactie wordt ongedaan gemaakt
http://php.net/manual/en/pdo.rollback.php

In onderstaand voorbeeld wordt geprobeerd om geld over te schrijven van de ene rekening naar de andere. De transactie bestaat uit 3 delen: het opvragen van het originele bedrag op rekening1, het updaten van rekening1 en het updaten van rekening2. Mocht er iets misgaan bij deze transactie dan wordt de begintoestand hersteld.

app.php

```
...
$accountFrom=1001;
$accountTo=1002;
$amount=23.4;
try{
   $pdo->beginTransaction();
   $sql = 'SELECT amount FROM accounts WHERE id=:from';
   $statement = $pdo->prepare($sql);
   $statement->bindParam(':from', $accountFrom, PDO::PARAM_INT );
   $statement->execute();
   $originalAmount = (double)$statement->fetchColumn();
   if($originalAmount < $amount){
       throw new Exception();
   }
   $sql  = 'UPDATE accounts '.
               'SET amount = amount - :amount '.
               'WHERE id = :from';
   $statement = $pdo->prepare($sql);
   $statement->bindParam(':from', $accountFrom, PDO::PARAM_INT );
   $statement->bindParam(':amount', $amount, PDO::PARAM_INT );
   $statement->execute();
   $sql  = 'UPDATE accounts '.
                'SET amount = amount + :amount '.
                'WHERE id = :to';
   $statement = $pdo->prepare($sql);
   $statement->bindParam(':to', $accountTo, PDO::PARAM_INT );
   $statement->bindParam(':amount', $amount, PDO::PARAM_INT );
   $statement->execute();
   $pdo->commit();
   echo 'succes';
} catch (Exception $e) {
   echo 'failure';
   $pdo->rollBack();
}
```