PDO

1 Verbinden met een database

datasource

? Leerdoelen

- Je weet wat PDO is en waarom het gebruikt wordt.
- Je kunt verbinding maken met een database via PDO.
- Je begrijpt waarom het handig is om een apart bestand voor de connectie te maken.

? Uitleg

PDO (PHP Data Objects) is een moderne manier om met databases te werken in PHP. Het ondersteunt meerdere soorten databases zoals MySQL, SQLite en PostgreSQL, maar in deze lessen gebruiken we alleen MySQL.



Je maakt verbinding met een database via een zogeheten DSN (Data Source Name) en slaat de connectie op in een variabele. Om herhaling te voorkomen, zet je dit in een apart bestand zoals `connection.php`. Dit maakt het ook makkelijk om de instellingen aan te passen wanneer je de website van je localhost naar een liveserver verplaatst.

`connection.php`

Het bestand `connection.php` bevat de code om verbinding te maken met de database. In plaats van in elk PHP-bestand opnieuw een connectie te moeten schrijven, zet je die één keer netjes in dit aparte bestand. Zo houd je je code overzichtelijk en voorkom je fouten.

Door `connection.php` te gebruiken, hoef je later bij het online zetten van je site alleen in **dat ene bestand** de instellingen aan te passen (zoals wachtwoord of host), in plaats van in alle bestanden waar je met de database werkt.

```php
<?php
$dsn = 'mysql:host=localhost;dbname=database_name;charset=utf8mb4';
$user = 'root';
$pass = '';

try {
  $pdo = new PDO($dsn, $user, $pass);
} catch (PDOException $e) {
  echo "Verbinding mislukt: " . $e->getMessage();
}
```

In het connection.php bestand wordt de database naam, en het user id en password ingesteld. Op een XAMPP ontwikkel server is standaard de user `root` en heeft geen password. Op een productieserver is dat natuurlijk ander!

? Opdracht

1. Maak een database `voorbeeld` aan met één tabel `dieren` met de kolommen `id` (INT, AUTO_INCREMENT, PRIMARY KEY), `naam` (VARCHAR), en `soort` (VARCHAR).
2. Maak een bestand `connection.php` dat de connectie maakt zoals hierboven.
3. Maak een tweede bestand `testverbinding.php` waarin je `require 'connection.php';` gebruikt om verbinding te maken.
4. Laat met `echo` zien of de verbinding is gelukt (bijv. "Verbinding gelukt!").

? Reflectie

- Welke manieren zijn er in PHP om met een database te werken, en waarom gebruiken wij PDO?
- Wat zijn voordelen van een apart `connection.php` bestand?
- connection.php heeft op een development omgeving een andere inhoud dan op een productieserver. Wat is het verschil en waarom?

? Inleveren

- Beantwoord in eigen woorden de reflectievragen (txt of pdf).

2 Gegevens uitlezen met - SELECT

? Leerdoelen

- Je weet hoe je gegevens uit een MySQL-database ophaalt met PDO.
- Je kunt een SELECT-query uitvoeren via PDO.
- Je kunt resultaten weergeven in HTML via PHP.

? Uitleg

In deze opdracht gebruik je een bestaande database met studentgegevens. Je voert met behulp van PDO een `SELECT`-query uit en toont de resultaten in een HTML-tabel.

Database

Je gebruikt een SQL-bestand student.sql om snel een database en tabel aan te maken met voorbeeldgegevens:

1. Open **phpMyAdmin** (via XAMPP of MAMP).
2. Klik op "Importeren".
3. Selecteer het bestand `student.sql` dat je van je docent krijgt of downloadt.
4. Klik op "Start" om het script uit te voeren. Je krijgt nu een database met de tabel `studenten`.

read.php

Maak `read.php` en zet daar deze code in:

```
<?php
require 'connection.php';

$sql = "SELECT id, voornaam, achternaam, woonplaats FROM studenten";
$stmt = $pdo->query($sql);
$studenten = $stmt->fetchAll();
?>
<table border="1">
  <tr>
```

```
    <th>ID</th>
    <th>Naam</th>
    <th>Woonplaats</th>
  </tr>
  <?php foreach ($studenten as $student): ?>
  <tr>
    <td><?= $student['id'] ?></td>
    <td><?= $student['voornaam'] . ' ' . $student['achternaam'] ?></td>
    <td><?= $student['woonplaats'] ?></td>
  </tr>
  <?php endforeach; ?>
</table>
```

?? Opdracht – studentenlijst weergeven

1. Importeer `student.sql` in phpMyAdmin om de database en tabel aan te maken.
2. Maak een bestand `read.php`.
3. Maak een `connection.php` bestand zoals dat in de vorige les is uitgelegd.
4. Gebruik de gegeven code en test of de lijst met studenten goed wordt weergegeven.
5. **Extra (otionele) opdracht:** Voeg de kolom `email` toe aan je SELECT-query en aan de HTML-tabel. Zorg ervoor dat het e-mailadres klikbaar is via een `mailto:` link.

? Reflectie

(zoals altijd: leg uit in eigen woorden!)

- Waar en hoe wordt het `connection.php` bestand ingelezen in `read.php`?
- Wat doet `$pdo->query()` precies?
- Wat is het verschil tussen `fetch()` en `fetchAll()`?
- Hoe toon je data uit een array netjes in een HTML-tabel?
- Wat doet een `mailto:`-link precies?

? Inleveren

- Lever het bestand `read.php` in via Teams of Canvas.

- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

3 Studentgegevens toevoegen - INSERT

? Leerdoelen

- Je weet hoe je data toevoegt aan een database met PDO.
- Je kunt een formulier maken en de ingevoerde gegevens veilig verwerken.
- Je begrijpt het gebruik van `prepare()` en `execute()` in PDO.

? Uitleg

Als je gegevens naar de database wilt sturen (bijvoorbeeld via een formulier), gebruik je een **INSERT-query**. Bij PDO doe je dit veilig met `prepare()` en `execute()`. Zo voorkom je problemen zoals **SQL-injectie**. SQL-injectie kan worden gebruikt om te 'hacken' en wordt later in een ander module uitgelegd.

Je gebruikt `prepare()` om de query voor te bereiden met placeholders (`:naam`), en daarna geef je met `execute()` de daadwerkelijke waarden door.

Voorbeeld – voorbereiding op INSERT

Maak een bestand `create.php` met deze HTML en PHP-code:

```
<?php
require 'connection.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $sql = "INSERT INTO studenten (voornaam, achternaam, woonplaats, email)
          VALUES (:voornaam, :achternaam, :woonplaats, :email)";
  $stmt = $pdo->prepare($sql);
  // Vul hier de juiste execute() aan
}
?>
```

```
<form method="post">
  <label>Voornaam: <input type="text" name="voornaam"></label><br>
  <label>Achternaam: <input type="text" name="achternaam"></label><br>
  <label>Woonplaats: <input type="text" name="woonplaats"></label><br>
  <label>E-mail: <input type="email" name="email"></label><br>
  <button type="submit">Toevoegen</button>
</form>
```

?? Opdracht – student toevoegen via formulier

1. Maak het bestand `create.php` aan en zet de code hierboven erin.
2. Test of het formulier zichtbaar is in je browser.
3. **Vul de ontbrekende code aan:** zorg dat de `execute()` functie de juiste data gebruikt uit het formulier.
4. Voeg een `echo` toe na het invoegen (bijv. “Student toegevoegd!”) zodat je weet dat het gelukt is.
5. Controleer in **phpMyAdmin** of de student correct is toegevoegd aan de database.

? Reflectie

(zoals altijd: leg uit in eigen woorden!)

- Waarom gebruik je `prepare()` in plaats van direct een query uitvoeren?
- Hoe weet PHP welke waarden in de query moeten komen?
- Wat is de functie van regel 4 (de regel die met if begint)?

? Inleveren

- Lever het bestand `create.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

4 Studentgegevens bewerken - UPDATE

? Leerdoelen

- Je weet hoe je bestaande data wijzigt in een database met PDO.
- Je kunt een formulier maken dat bestaande gegevens toont en laat aanpassen.
- Je begrijpt hoe je een `UPDATE`-query uitvoert met `prepare()` en `execute()`.

? Uitleg

Met een `UPDATE`-query kun je bestaande gegevens in de database aanpassen. Je gebruikt ook hier `prepare()` en `execute()` zodat de invoer veilig verwerkt wordt.

Vaak haal je eerst de huidige gegevens op, zodat de gebruiker weet wat hij gaat bewerken. Daarna verwerk je de aangepaste gegevens.

Voorbeeld – basisopzet update.php

Maak een bestand `update.php` en vul deze code in:

```
<?php
require 'connection.php';

$id = $_GET['id'] ?? null;

if (!$id) {
  echo "Geen ID opgegeven.";
  exit;
}

$stmt = $pdo->prepare("SELECT * FROM studenten WHERE id = ?");
$stmt->execute([$id]);
$student = $stmt->fetch();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $sql = "UPDATE studenten SET voornaam = :voornaam, achternaam = :achternaam, woonplaats =
:woonplaats, email = :email WHERE id = :id";
  $stmt = $pdo->prepare($sql);
  // Vul hier de juiste execute() functie aan
}
?>
```

```
<form method="post">
  <label>Voornaam: <input type="text" name="voornaam" value="<?= $student['voornaam']
?>"></label><br>
  <label>Achternaam: <input type="text" name="achternaam" value="<?= $student['achternaam']
?>"></label><br>
  <label>Woonplaats: <input type="text" name="woonplaats" value="<?= $student['woonplaats']
?>"></label><br>
  <label>E-mail: <input type="email" name="email" value="<?= $student['email']
?>"></label><br>
  <button type="submit">Opslaan</button>
</form>
```

?? Opdracht – studentgegevens aanpassen

1. Maak een bestand `update.php`.
2. Gebruik de code hierboven. Let op: gebruik een bestaand ID uit je database in de URL (bijv. `update.php?id=3`).
3. Test of de juiste gegevens zichtbaar zijn in het formulier.
4. **Vul zelf de `execute()` regel aan** zodat de gegevens uit het formulier worden opgeslagen in de database.
5. Voeg na het opslaan een `echo` toe met "Wijziging opgeslagen".
6. Controleer of de database wordt aangepast.

? Reflectie

- Hoe weet PHP welk studentrecord moet worden aangepast, leg uit heo dat werkt?
- Waarom gebruik je een `WHERE` clausule in een `UPDATE` query?
- Hoe weet je vanuit de code **zeker** dat de juiste gegevens zijn opgeslagen?

? Inleveren

- Lever het bestand `update.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

5 Student verwijderen

? Leerdoelen

- Je weet hoe je een record uit een database verwijdert met PDO.
- Je begrijpt waarom je altijd moet controleren welk record je verwijdert.
- Je kunt veilig en gecontroleerd een `DELETE`-query uitvoeren.

? Uitleg

Met een `DELETE`-query verwijder je een record uit de database. Omdat deze actie niet teruggedraaid kan worden, wil je dit altijd eerst bevestigen. Ook gebruik je een `WHERE` clausule om precies aan te geven welk record je bedoelt.

In deze opdracht toon je eerst de naam van de student die je gaat verwijderen, zodat je als gebruiker weet wat je doet. Daarna kun je pas bevestigen en verwijderen.

Voorbeeld – delete.php

```
<?php
require 'connection.php';

$id = $_GET['id'] ?? null;

if (!$id) {
  echo "Geen ID opgegeven.";
  exit;
}

// TO DO: Haal hier de student op via SELECT zodat je zijn/haar naam kunt tonen
// $student = ...

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $stmt = $pdo->prepare("DELETE FROM studenten WHERE id = :id");
  $stmt->execute(['id' => $id]);
  echo "Student verwijderd.";
  exit;
}
```

```
?>

<p>Weet je zeker dat je student <strong><?= '... hier de naam' ?></strong> wilt
verwijderen?</p>

<form method="post">
  <button type="submit">Ja, verwijder</button>
</form>

<p><a href="read.php">Annuleer</a></p>
```

?? Opdracht – student verwijderen met naam

1. Maak een bestand `delete.php`.
2. Gebruik de bovenstaande code.
3. **Vul zelf de `SELECT`-query aan** zodat je de gegevens van de student ophaalt op basis van het ID.
4. Laat de voornaam en achternaam van de student zien in de bevestigingsvraag.
5. Test de pagina via de URL: `delete.php?id=3` (of een ander bestaand ID).

? Reflectie

- Waarom is het belangrijk om eerst de naam van de student te tonen voordat je verwijdert?
- Hoe weet PHP welke gegevens bij het opgegeven ID horen?
- Wat gebeurt er als je probeert een niet-bestaand ID te verwijderen?
- Waarom gebruiken we `POST` voor het verwijderen?

? Inleveren

- Lever het bestand `delete.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

6 Studenten zoeken

? Leerdoelen

- Je weet hoe je een `WHERE` clausule gebruikt in een SELECT-query met PDO.
- Je begrijpt hoe `LIKE` gebruikt wordt voor zoeken op (deel van) tekst.
- Je kunt input van een formulier gebruiken om gegevens te filteren.

? Uitleg

Met een `WHERE`-clausule kun je filteren welke rijen je uit de database haalt. Als je wilt zoeken op een deel van een naam, gebruik je `LIKE` met het procentteken `%`.

Bijvoorbeeld: `SELECT * FROM studenten WHERE voornaam LIKE :zoek`, en dan geef je `%zoekwoord%` mee aan de placeholder `:zoek`.

Voorbeeldopzet – zoek.php

Maak een bestand `zoek.php` met deze opzet:

```
<form method="get">
  <input type="text" name="zoek" placeholder="Zoek op voornaam">
  <button type="submit">Zoeken</button>
</form>

<?php
require 'connection.php';

$zoekwoord = $_GET['zoek'] ?? '';

if ($zoekwoord) {
  // TO DO: vul de juiste SELECT-query en fetchAll() hier aan
}
?>
```

?? Opdracht – zoekfunctie bouwen

1. Maak het bestand `zoek.php`.
2. Gebruik de bovenstaande code als startpunt.
3. **Vul zelf de SELECT-query aan** met een `WHERE voornaam LIKE :zoek` clausule.

4. Voer de query uit met `prepare()` en `execute()` – gebruik `%` wildcards om ook op deelwoorden te zoeken.
5. Toon de resultaten in een tabel met voornaam, achternaam, woonplaats en e-mail.

? Reflectie

- Wat is het verschil tussen een gewone vergelijking (`=`) en `LIKE`?
- Hoe werkt `%` in een LIKE-query precies?
- Wat gebeurt er als je een lege zoekterm invoert?
- Waarom gebruik je `prepare()` ook bij zo'n zoekopdracht?

? Inleveren

- Lever het bestand `zoek.php` in via Teams of Canvas.
- Beantwoord de reflectievragen in een .txt of .pdf bestand en lever die ook in.

7 Mini-project: CRUD applicatie

? Leerdoelen

- Je kunt een complete CRUD-toepassing bouwen met PDO.
- Je past SELECT, INSERT, UPDATE, DELETE en LIKE correct toe.
- Je gebruikt `connection.php` om je project onderhoudbaar te houden.

Wat is CRUD?

CRUD staat voor **Create, Read, Update, Delete** – dit zijn de vier basisbewerkingen die je met gegevens in een database kunt uitvoeren:

- **Create**: nieuwe gegevens toevoegen (bijv. een nieuwe student inschrijven)
- **Read**: gegevens opvragen en tonen (bijv. een lijst van alle studenten)
- **Update**: bestaande gegevens aanpassen (bijv. een fout corrigeren in een naam)
- **Delete**: gegevens verwijderen (bijv. een student uitschrijven)

Een CRUD-applicatie is een programma dat deze vier functies ondersteunt. In webontwikkeling gebruik je vaak formulieren en SQL in combinatie met PHP of andere programmeertalen om deze acties uit te voeren.

? Uitleg

Je hebt nu alle onderdelen geleerd om een volledige webapplicatie te maken die met een database werkt. In dit project bouw je een kleine CRUD-app voor studenten waarin je:

- een lijst van studenten toont (**read**)
- nieuwe studenten kunt toevoegen (**create**)
- bestaande studenten kunt bewerken (**update**)
- studenten kunt verwijderen (met bevestiging) (**delete**)
- kunt zoeken op voornaam

?? Opdracht – CRUD-app bouwen

1. Maak een eigen map of project met minimaal de volgende bestanden:
   - `read.php` – toont de lijst van studenten
   - `create.php` – formulier om nieuwe studenten toe te voegen
   - `update.php` – formulier om bestaande studenten te bewerken
   - `delete.php` – bevestiging en verwijdering van een student
   - `zoek.php` – zoekfunctie op voornaam
   - `connection.php` – je databaseverbinding
2. Zorg dat je hiervan één web applicatie van en zorg dat je via een menu bar kan navigeren.
3. Voeg bovenaan `read.php` een navigatie toe zodat je snel naar de andere pagina’s kunt.
4. Controleer of je code werkt voor verschillende studenten en test het met minstens 3 zelf ingevoerde records.
5. Je mag de opmaak aanpassen met CSS als je wilt, maar dat is optioneel.

? Reflectie

- Welke onderdelen van PDO vind je makkelijk, en welke lastig?

- Waar moet je vooral op letten bij UPDATE en DELETE?
- Wat zou je verbeteren als je meer tijd had voor dit project?
- Wat betekent het als we zeggen dat PDO "veilig" is? Wanneer is dat waar?

? Inleveren

- Lever je hele projectmap in (alle .php-bestanden).
- Lever de reflectie in (txt of pdf). En gebruik je eigen woorden!

8 Toetsvragen: PDO en CRUD

? Leerdoelen

- Je herkent de belangrijkste onderdelen van PDO-gebruik in PHP.
- Je begrijpt hoe CRUD-operaties technisch en logisch werken.
- Je kunt fouten of onvolledigheden in PDO-code herkennen en verklaren.

?? Toetsvragen

Wat doet PDO, waarvoor hebben we het nodig?

PDO staat voor PHP Data Objects. Het is een library met functies waarmee je PHP kan 'verbinden' met een database. Je kan dan in PHP SQL queries gebruiken om de data in de database op te vragen of aan te passen.

Hoe maak je een verbinding met de database via PDO?

Je maakt een DSN-string aan (bijv. `mysql:host=localhost;dbname=voorbeeld;charset=utf8mb4`), geeft gebruikersnaam en wachtwoord door, en instancieert een nieuwe `PDO`-object.

Het meest eenvoudige voorbeeld is:

```
$pdo = new PDO( 'mysql:host=localhost;dbname=database_name;charset=utf8mb4','user', 'password');
```

Waarom is het handig om connection.php te gebruiken?

Door de connectiecode in één bestand te zetten, onderhoud je die centrale plek makkelijker.

Als je wachtwoord wijzigt of je plaatst je code vanaf je laptop op een productie server dan hoef je op maar één paats je wachtwoord aan te passen.

Waarom zou je trouwens je wachtwoord willen aanpassen als je je code op een prodctie server zet?

Op jouw laptop hgebruik je voor je database meestal geen password, lekker makkelijk. Een productieserver kan de hele wereld 'zien' en als je geen password gebruikt dan is je database binnen de kortste keren gehacked.

Tip: verander niet alleen je wachtwoord, maar pas ook je **user name** aan dan maak je het hackers nog iets lastiger.

Hoe gebruik je fetchAll() en wanneer gebruik je fetch()?

`fetchAll()` haalt alle rijen in één keer op als array van arrays (of objects). `fetch()` haalt per aanroep één rij op, handig als je maar één record verwacht of in een loop wilt verwerken.

Waarom gebruik je bij INSERT en UPDATE altijd POST in plaats van GET?

Omdat `POST` bedoeld is voor acties die data wijzigen en niet in de URL verschijnt, zodat gevoelige data niet zichtbaar wordt en de actie niet per ongeluk herhaald wordt bij refresh van de pagina.

Waarvoor gebruik je WHERE bij een UPDATE of DELETE?

De `WHERE`-clausule specifieert precies welke rij(en) aangepast of verwijderd moeten worden. Zonder `WHERE` wordt de operatie op álle rijen uitgevoerd.

Wat betekent CRUD en hoe komen deze onderdelen terug in de module?

CRUD staat voor Create, Read, Update, Delete. In de module leer je:

- **Create:** via `INSERT` en `prepare()/execute()` (create.php).
- **Read:** via `SELECT` en `fetchAll()` (read.php).
- **Update:** via `UPDATE` met prepared statements (update.php).
- **Delete:** via `DELETE` met bevestiging en `WHERE` (delete.php).

Wat doet require 'connection.php' ?

Dit voegt het bestand `connection.php` in en maakt de databaseconnectie beschikbaar. Hierdoor hoef je niet in elk stukje code waarin je PDO gebruikt opnieuw de connectie te schrijven.

? Inleveren

Gebruik deze vragen om jezelf te testen of als voorbereiding op de kennis-check.

Maak één zelf bedachte vraag over PDO gaat en die hier boven niet staat.

- Stel de vraag, geef het antwoord en een korte uitleg. Gebruik het formaat dat hier ook wordt gebruikt, Lever in in PDF of TXT.

--

Revision #12
Created 2025-06-13 19:04:30 UTC by Max
Updated 2026-07-10 09:05:37 UTC by Max