Complete CRUD in PHP

In deze les gaan we een complete CRUD maken op basis van PHP en PDO.

Wat ga je leren?

- **C**reate, aanmaken record/regels in een MySQL database via PHP/PDO.
- **R**ead, lezen van MySQL records en tonen in een HTML tabel.
- **U**pdate, het aanpassen van MySQL records via PHP/PDO.
- **D**elete, het verwijderen van records via PHP/PDO.
- GET en POST gegevens.
- Aanmaken van SQL queries met het prepare statement.

Stap 1 - set-up

We maken de volgende file structuur:

```
\-- phpcrud
    |-- index.php
    |-- create.php
    |-- read.php
    |-- update.php
    |-- delete.php
    |-- functions.php
    |-- style.css
```

- *index.php* — Home page.
- *create.php* — Create, aanmaken van nieuwe records.
- *read.php* — Tonen van records in een tabel en navigeren met pagina's.
- *update.php* — Update van records.
- *delete.php* — Verwijderen van records (met confirm).
- *functions.php* — Templates en MySQL connectie code om ervoor te zorgen dat je niet telkens dezelfde code goeft te maken.

- *style.css* — The stylesheet.

Database aanmaken

1. Ga naar phpmyadmin: http://localhost/phpmyadmin
2. Maak een nieuwe database met de naam *phpcrud* (selecteer utf8_general_ci) voor collation.
3. Selecteer de nieuwe database en voer de volgende SQL uit:

```
CREATE TABLE IF NOT EXISTS `contacts` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `phone` varchar(255) NOT NULL,
  `title` varchar(255) NOT NULL,
  `created` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=13 DEFAULT CHARSET=utf8;

INSERT INTO `contacts` (`id`, `name`, `email`, `phone`, `title`, `created`) VALUES
(1, 'John Doe', 'johndoe@example.com', '2026550143', 'Lawyer', '2019-05-08 17:32:00'),
(2, 'David Deacon', 'daviddeacon@example.com', '2025550121', 'Employee', '2019-05-08
17:28:44'),
(3, 'Sam White', 'samwhite@example.com', '2004550121', 'Employee', '2019-05-08 17:29:27'),
(4, 'Colin Chaplin', 'colinchaplin@example.com', '2022550178', 'Supervisor', '2019-05-08
17:29:27'),
(5, 'Ricky Waltz', 'rickywaltz@example.com', '7862342390', '', '2019-05-09 19:16:00'),
(6, 'Arnold Hall', 'arnoldhall@example.com', '5089573579', 'Manager', '2019-05-09 19:17:00'),
(7, 'Toni Adams', 'alvah1981@example.com', '2603668738', '', '2019-05-09 19:19:00'),
(8, 'Donald Perry', 'donald1983@example.com', '7019007916', 'Employee', '2019-05-09
19:20:00'),
(9, 'Joe McKinney', 'nadia.doole0@example.com', '6153353674', 'Employee', '2019-05-09
19:20:00'),
(10, 'Angela Horst', 'angela1977@example.com', '3094234980', 'Assistant', '2019-05-09
19:21:00'),
(11, 'James Jameson', 'james1965@example.com', '4002349823', 'Assistant', '2019-05-09
19:32:00'),
(12, 'Daniel Deacon', 'danieldeacon@example.com', '5003423549', 'Manager', '2019-05-09
```

```
19:33:00');
```

Deze SQL code maakt een table *contacts* die we in onze applicatie gebruiken.De tabel wordt gevuld met standaard testdata. Deze data kunnen we later met behulp van onze applicatie aanpassen.

Er zijn 6 kolommen in de contacts tabel (*id*, *name*, *email*, *phone*, *title*, en *created*), *Title* is de rol van de gebruiker. Deze contacts tabel kan later wordne uitgebreid met een *password* kolom en kan dan worden gebruikt om aan te kunnen loggen in een applicatie.

De database in phpMyAdmin ziet er als volgt uit:

image-1631822941894.png

Stylesheet maken

In de style sheet, *style.css* zet he de volgende code:

```
* {
  box-sizing: border-box;
  font-family: -apple-system, BlinkMacSystemFont, "segoe ui", roboto, oxygen, ubuntu,
cantarell, "fira sans", "droid sans", "helvetica neue", Arial, sans-serif;
  font-size: 16px;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
body {
  background-color: #FFFFFF;
  margin: 0;
}
.navtop {
  background-color: #3f69a8;
  height: 60px;
  width: 100%;
  border: 0;
}
.navtop div {
  display: flex;
  margin: 0 auto;
  width: 1000px;
  height: 100%;
}
.navtop div h1, .navtop div a {
```

```
  display: inline-flex;
  align-items: center;
}
.navtop div h1 {
  flex: 1;
  font-size: 24px;
  padding: 0;
  margin: 0;
  color: #ecf0f6;
  font-weight: normal;
}
.navtop div a {
  padding: 0 20px;
  text-decoration: none;
  color: #c5d2e5;
  font-weight: bold;
}
.navtop div a i {
  padding: 2px 8px 0 0;
}
.navtop div a:hover {
  color: #ecf0f6;
}
.content {
  width: 1000px;
  margin: 0 auto;
}
.content h2 {
  margin: 0;
  padding: 25px 0;
  font-size: 22px;
  border-bottom: 1px solid #ebebeb;
  color: #666666;
}
.read .create-contact {
  display: inline-block;
  text-decoration: none;
  background-color: #38b673;
  font-weight: bold;
  font-size: 14px;
```

```
  color: #FFFFFF;
  padding: 10px 15px;
  margin: 15px 0;
}
.read .create-contact:hover {
  background-color: #32a367;
}
.read .pagination {
  display: flex;
  justify-content: flex-end;
}
.read .pagination a {
  display: inline-block;
  text-decoration: none;
  background-color: #a5a7a9;
  font-weight: bold;
  color: #FFFFFF;
  padding: 5px 10px;
  margin: 15px 0 15px 5px;
}
.read .pagination a:hover {
  background-color: #999b9d;
}
.read table {
  width: 100%;
  padding-top: 30px;
  border-collapse: collapse;
}
.read table thead {
  background-color: #ebeef1;
  border-bottom: 1px solid #d3dae0;
}
.read table thead td {
  padding: 10px;
  font-weight: bold;
  color: #767779;
  font-size: 14px;
}
.read table tbody tr {
  border-bottom: 1px solid #d3dae0;
```

```
}
.read table tbody tr:nth-child(even) {
  background-color: #fbfcfc;
}
.read table tbody tr:hover {
  background-color: #376ab7;
}
.read table tbody tr:hover td {
  color: #FFFFFF;
}
.read table tbody tr:hover td:nth-child(1) {
  color: #FFFFFF;
}
.read table tbody tr td {
  padding: 10px;
}
.read table tbody tr td:nth-child(1) {
  color: #a5a7a9;
}
.read table tbody tr td.actions {
  padding: 8px;
  text-align: right;
}
.read table tbody tr td.actions .edit, .read table tbody tr td.actions .trash {
  display: inline-flex;
  text-align: right;
  text-decoration: none;
  color: #FFFFFF;
  padding: 10px 12px;
}
.read table tbody tr td.actions .trash {
  background-color: #b73737;
}
.read table tbody tr td.actions .trash:hover {
  background-color: #a33131;
}
.read table tbody tr td.actions .edit {
  background-color: #37afb7;
}
.read table tbody tr td.actions .edit:hover {
```

```
  background-color: #319ca3;
}
.update form {
  padding: 15px 0;
  display: flex;
  flex-flow: wrap;
}
.update form label {
  display: inline-flex;
  width: 400px;
  padding: 10px 0;
  margin-right: 25px;
}
.update form input {
  padding: 10px;
  width: 400px;
  margin-right: 25px;
  margin-bottom: 15px;
  border: 1px solid #cccccc;
}
.update form input[type="submit"] {
  display: block;
  background-color: #38b673;
  border: 0;
  font-weight: bold;
  font-size: 14px;
  color: #FFFFFF;
  cursor: pointer;
  width: 200px;
margin-top: 15px;
}
.update form input[type="submit"]:hover {
  background-color: #32a367;
}
.delete .yesno {
  display: flex;
}
.delete .yesno a {
  display: inline-block;
  text-decoration: none;
```

```
  background-color: #38b673;
  font-weight: bold;
  color: #FFFFFF;
  padding: 10px 15px;
  margin: 15px 10px 15px 0;
}
.delete .yesno a:hover {
  background-color: #32a367;
}
```

Natuurlijk kun je de style sheet zelf aanpassen.

Maak de CRUD app

functions.php

```
<?php
function pdo_connect_mysql() {
    $DATABASE_HOST = '';
    $DATABASE_USER = '';
    $DATABASE_PASS = '';
    $DATABASE_NAME = '';
    try {
    return new PDO('mysql:host=' . $DATABASE_HOST . ';dbname=' . $DATABASE_NAME .
';charset=utf8', $DATABASE_USER, $DATABASE_PASS);
    } catch (PDOException $exception) {
    // If there is an error with the connection, stop the script and display the error.
    exit('Failed to connect to database!');
    }
}
function template_header($title) {
echo <<<EOT
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>$title</title>
<link href="style.css" rel="stylesheet" type="text/css">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
</head>
```

```
<body>
    <nav class="navtop">
    <div>
    <h1>Website Title</h1>
            <a href="index.php"><i class="fas fa-home"></i>Home</a>
    <a href="read.php"><i class="fas fa-address-book"></i>Contacts</a>
    </div>
    </nav>
EOT;
}
function template_footer() {
echo <<<EOT
    </body>
</html>
EOT;
}
?>
```

Pas de PDO-code aan zodat je een verbinding met jouw database kan maken. Pas hiervoor regel 3, 4, 5 en 6 aan.

Laad de *functions.php* en controleer of je geen foutmelding krijgt. Als het werkt betekent dat je een verbinding naar je database kan maken!

Home page

index.php

```
<?php
// pas deze regel aan en zorg dat de code uit functions.php wordt geladen.
// Your PHP code here.

// Home Page template below.
?>

<?=template_header('Home')?>

<div class="content">
<h2>Home</h2>
<p>Welcome to the home page!</p>
</div>
```

```
<?=template_footer()?>
```

Dit is de standaard home page. Op regel 8 en 15 wordt een functie aangeroepen. Deze functie staat in functions.php. Verander regel 2 zodat functions.php wordt geladen.

Check of de homepage werkt: *http://localhost/phpcrud/*

image-1631823485549.png

Verander eventueel zelf de homepagina.

Read pagina

read.php

```
<?php
include 'functions.php';
// Connect to MySQL database
$pdo = pdo_connect_mysql();
// Get the page via GET request (URL param: page), if non exists default the page to 1
$page = isset($_GET['page']) && is_numeric($_GET['page']) ? (int)$_GET['page'] : 1;
// Number of records to show on each page
$records_per_page = 5;
```

- De standaard functies worden *included*.
- De verbinding met de database wordt gemaakt.
- Er wordt gekeken welke pagina moet worden getoond, standaard is dit pagina 1.
- Er worden 5 regels per pagina getoond.

Verander regel 6 in een vorm met een *if-then-else* structuur.

Voeg dit toe aan *read.php*

```
// Prepare the SQL statement and get records from our contacts table, LIMIT will determine the
page
$stmt = $pdo->prepare('SELECT * FROM contacts ORDER BY id LIMIT :current_page,
:record_per_page');
$stmt->bindValue(':current_page', ($page-1)*$records_per_page, PDO::PARAM_INT);
$stmt->bindValue(':record_per_page', $records_per_page, PDO::PARAM_INT);
```

```
$stmt->execute();
// Fetch the records so we can display them in our template.
$contacts = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

- De SQL query wordt gemaakt en je wil alleen pagina X zien. Stel je hebt 5 regels per pagina en je wilt pagina 3 zien.
- Als $page=3 dan wordt :current_page 2*5 dus 10 en worden dus 5 regels te beginnen met regel 10 getoond.
- De query wordt uitgevoerd en alle resultaten worden in $contacts gezet. *fetchAll()* haalt alle regels op.

Controleer of deze code werkt door de inhoud van $contacts af te drukken. Gebruik hiervoor de volgende code.

```
echo "<pre>";
var_dump($contacts);
echo "</pre>";
```

Als alles goed werkt haal dan de deubg-code waarmee je de inhoud van $contacts kan bekijken weg en voeg het volgende toe aan read.php

```
// Get the total number of contacts, this is so we can determine whether there should be a
next and previous button
$num_contacts = $pdo->query('SELECT COUNT(*) FROM contacts')->fetchColumn();
?>
```

> Vraag: Het aantal contacten wordt bepaald. Let op *fetchColumn()* dit haalt de *eerste* resultaatregel op. Waarom gebruiken we eigenlijk hier geen *fetchAll()* ?

Controleer of de inhoud van $num_contacts klopt. Als dat zo is voeg dan het volgende toe aan *read.php*

```
<?=template_header('Read')?>

<div class="content read">
	<h2>Read Contacts</h2>
	<a href="create.php" class="create-contact">Create Contact</a>
        <?php foreach ($contacts as $contact): ?>
            <?=$contact['id']?>
            <?=$contact['name']?>
```

```
            <?=$contact['email']?>
            <?=$contact['phone']?>
            <?=$contact['title']?>
            <?=$contact['created']?>

            <a href="update.php?id=<?=$contact['id']?>" class="edit"><i class="fas fa-pen fa-
xs"></i></a>
            <a href="delete.php?id=<?=$contact['id']?>" class="trash"><i class="fas fa-trash
fa-xs"></i></a>
        <?php endforeach; ?>

□<div class="pagination">

□□<?php if ($page > 1): ?>
□□□<a href="read.php?page=<?=$page-1?>"><i class="fas fa-angle-double-left fa-sm"></i></a>
□□<?php endif; ?>

□□<?php if ($page*$records_per_page < $num_contacts): ?>
□□□<a href="read.php?page=<?=$page+1?>"><i class="fas fa-angle-double-right fa-sm"></i></a>
□□<?php endif; ?>

□</div>
</div>

<?=template_footer()?>
```

De resultaten worden afgedrukt in de loop van regel 6 tot en met 12.

Maak nu een tabel waarin alle resultaten netjes onder elkaar worden afgedrukt zoals in het voorbeeld hieronder.

Maak hierbij gebruik van <theader> </theader> en <tbody> </tbody>.

Let op de *pagination* onderaan de pagina.

Als er de huidige pagina 1 is, dan wordt de link naar de vorige pagina getoond.

Als de huidige pagina * aantal regels per pagina > is dan het totale aantal pagina's dan wordt de link naar de volgende pagina niet getoond.

En wat zien we nu in *http://localhost/phpcrud/read.php*

image-1631823843446.png

De *edit* en *delete* knop (rechts naast elke regel) verwijzen naar php files die nog niet bestaan. Die gaan we nog maken.

Create

create.php

```
<?php
include 'functions.php';
$pdo = pdo_connect_mysql();
$msg = '';
// Check if POST data is not empty
if (!empty($_POST)) {
    // Post data not empty insert a new record
    // Set-up the variables that are going to be inserted, we must check if the POST variables exist if not we can default them to blank
    $id = isset($_POST['id']) && !empty($_POST['id']) && $_POST['id'] != 'auto' ? $_POST['id'] : NULL;
    // Check if POST variable "name" exists, if not default the value to blank, basically the same for all variables
    $name = isset($_POST['name']) ? $_POST['name'] : '';
    $email = isset($_POST['email']) ? $_POST['email'] : '';
    $phone = isset($_POST['phone']) ? $_POST['phone'] : '';
    $title = isset($_POST['title']) ? $_POST['title'] : '';
    $created = isset($_POST['created']) ? $_POST['created'] : date('Y-m-d H:i:s');
    // Insert new record into the contacts table
    $stmt = $pdo->prepare('INSERT INTO contacts VALUES (?, ?, ?, ?, ?, ?)');
    $stmt->execute([$id, $name, $email, $phone, $title, $created]);
    // Output message
    $msg = 'Created Successfully!';
}
?>

<?=template_header('Create')?>

<div class="content update">
	<h2>Create Contact</h2>
    <form action="create.php" method="post">
        <label for="id">ID</label>
        <label for="name">Name</label>
```

```
        <input type="text" name="id" placeholder="26" value="auto" id="id">
        <input type="text" name="name" placeholder="John Doe" id="name">
        <label for="email">Email</label>
        <label for="phone">Phone</label>
        <input type="text" name="email" placeholder="johndoe@example.com" id="email">
        <input type="text" name="phone" placeholder="2025550143" id="phone">
        <label for="title">Title</label>
        <label for="created">Created</label>
        <input type="text" name="title" placeholder="Employee" id="title">
        <input type="datetime-local" name="created" value="<?=date('Y-m-d\TH:i')?>"
id="created">
        <input type="submit" value="Create">
    </form>
    <?php if ($msg): ?>
    <p><?=$msg?></p>
    <?php endif; ?>
</div>

<?=template_footer()?>
```

Bestudeer de code.

Op regel 17 en 18 worden de gegevens in de database gezet. Deze gebruikte methode is foutgevoelig omdat de volgorde van parameters overeen moet komen met de volgorde van de kolommen in de database.

We gaan dit verbeteren. Bekijk de volgende voorbeeld-code:

```
<?php
$dbhost = 'localhost';
$dbname = 'pdo';
$dbusername = 'root';
$dbpassword = '845625';

$link = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbusername, $dbpassword);

$statement = $link->prepare('INSERT INTO testtable (name, lastname, age)
    VALUES (:fname, :sname, :age)');
```

```
$statement->execute([
    'fname' => 'Bob',
    'sname' => 'Desaunois',
    'age' => '18',
]);
```

Opdracht

Verander de code in *create.php* zodat je op de manier zoals in het voorbeeld is voor gedaan de variabelen aan de query bindt. Op deze manier is de volgorde van de tabel definitie niet van belang.

Controleer op: *http://localhost/phpcrud/create.php*

image-1631824036477.png

Update

update.php

```
<?php
include 'functions.php';
$pdo = pdo_connect_mysql();
$msg = '';
// Check if the contact id exists, for example update.php?id=1 will get the contact with the
id of 1
if (isset($_GET['id'])) {
    if (!empty($_POST)) {
        // This part is similar to the create.php, but instead we update a record and not
insert
        $id = isset($_POST['id']) ? $_POST['id'] : NULL;
        $name = isset($_POST['name']) ? $_POST['name'] : '';
        $email = isset($_POST['email']) ? $_POST['email'] : '';
        $phone = isset($_POST['phone']) ? $_POST['phone'] : '';
        $title = isset($_POST['title']) ? $_POST['title'] : '';
        $created = isset($_POST['created']) ? $_POST['created'] : date('Y-m-d H:i:s');
        // Update the record
        $stmt = $pdo->prepare('UPDATE contacts SET id = ?, name = ?, email = ?, phone = ?,
title = ?, created = ? WHERE id = ?');
        $stmt->execute([$id, $name, $email, $phone, $title, $created, $_GET['id']]);
        $msg = 'Updated Successfully!';
```

```
    }
    // Get the contact from the contacts table
    $stmt = $pdo->prepare('SELECT * FROM contacts WHERE id = ?');
    $stmt->execute([$_GET['id']]);
    $contact = $stmt->fetch(PDO::FETCH_ASSOC);
    if (!$contact) {
        exit('Contact doesn\'t exist with that ID!');
    }
} else {
    exit('No ID specified!');
}
?>
```

Verander hier regel 16 en 17 op dezelfde manier aan de hand van het voorbeeld waarbij geen vraagtekens meer worden gebruikt.

voeg de volgende code van het form toe aan update.php

```
<?=template_header('Read')?>

<div class="content update">
<h2>Update Contact #<?=$contact['id']?></h2>
    <form action="update.php?id=<?=$contact['id']?>" method="post">
        <label for="id">ID</label>
        <label for="name">Name</label>
        <input type="text" name="id" placeholder="1" value="<?=$contact['id']?>" id="id">
        <input type="text" name="name" placeholder="John Doe" value="<?=$contact['name']?>"
id="name">
        <label for="email">Email</label>
        <label for="phone">Phone</label>
        <input type="text" name="email" placeholder="johndoe@example.com"
value="<?=$contact['email']?>" id="email">
        <input type="text" name="phone" placeholder="2025550143"
value="<?=$contact['phone']?>" id="phone">
        <label for="title">Title</label>
        <label for="created">Created</label>
        <input type="text" name="title" placeholder="Employee" value="<?=$contact['title']?>"
id="title">
        <input type="datetime-local" name="created" value="<?=date('Y-m-d\TH:i',
strtotime($contact['created']))?>" id="created">
        <input type="submit" value="Update">
```

```
    </form>
    <?php if ($msg): ?>
    <p><?=$msg?></p>
    <?php endif; ?>
</div>

<?=template_footer()?>
```

Check of de update werkt.

image-1631824374219.png

Delete

delete.php

```
<?php
include 'functions.php';
$pdo = pdo_connect_mysql();
$msg = '';
// Check that the contact ID exists
if (isset($_GET['id'])) {
    // Select the record that is going to be deleted
    $stmt = $pdo->prepare('SELECT * FROM contacts WHERE id = ?');
    $stmt->execute([$_GET['id']]);
    $contact = $stmt->fetch(PDO::FETCH_ASSOC);
    if (!$contact) {
        exit('Contact doesn\'t exist with that ID!');
    }
    // Make sure the user confirms beore deletion
    if (isset($_GET['confirm'])) {
        if ($_GET['confirm'] == 'yes') {
            // User clicked the "Yes" button, delete record
            $stmt = $pdo->prepare('DELETE FROM contacts WHERE id = ?');
            $stmt->execute([$_GET['id']]);
            $msg = 'You have deleted the contact!';
        } else {
            // User clicked the "No" button, redirect them back to the read page
            header('Location: read.php');
            exit;
```

```
        }
    }
} else {
    exit('No ID specified!');
}
?>
```

Bestudeer de code en voeg het volgende toe aan *delete.php*

```
<?=template_header('Delete')?>

<div class="content delete">
	<h2>Delete Contact #<?=$contact['id']?></h2>
    <?php if ($msg): ?>
    <p><?=$msg?></p>
    <?php else: ?>
	<p>Are you sure you want to delete contact #<?=$contact['id']?>?</p>
    <div class="yesno">
        <a href="delete.php?id=<?=$contact['id']?>&confirm=yes">Yes</a>
        <a href="delete.php?id=<?=$contact['id']?>&confirm=no">No</a>
    </div>
    <?php endif; ?>
</div>

<?=template_footer()?>
```

Klik op de delete knop op de read page en test de delete.

image-1631824543510.png

Afsluiting

We hebben een complete CRUD gemaakt.

We hebben geleerd hoe PDO de resultaten in een PHP variabele zet en hoe we deze inhoud afdrukken in een nette tabel.

We hebben een verbinding met de database gemaakt en we hebben geleerd hoe we op een juiste manier variabele aan onze queries moeten binden (variabel binding).

--

Revision #5
Created 2021-09-17 10:09:46 UTC by Max
Updated 2022-07-17 10:18:52 UTC by Max