

GRAFISCH LYCEUM

ROTTERDAM



MPHP4

>Opfrisreader PHP applicatie

MEDIA- EN GAMEDEVELOPER | CREBO: 95313 | LEERJAAR 2
GRAFISCH LYCEUM ROTTERDAM
SAMENSTELLER: M. KATHMANN

INHOUDSOPGAVE

INLEIDING	2
WERKWIJZE EN BEOORDELING	3
INDELING.....	3
BEOORDELING	3
PLANNING	4
1. EEN CRUD APPLICATIE	5
WAT IS CRUD?	5
ONZE APPLICATIE: LEDENBEHEER	5
2. DE STRUCTUUR VAN DE APP BOUWEN	8
DE TABEL	8
DE CONFIG EN HOMEPAGE MAKEN	9
DE LEDEN LEZEN EN TONEN	11
3. NIEUWE LEDEN TOEVOEGEN.....	13
HET FORMULIER.....	13
HET FORMULIER VERWERKEN	14
4. LEDEN BEWERKEN	17
DE BEWERKLINK.....	17
HET FORMULIER.....	18
HET FORMULIER VERWERKEN	22
5. LEDEN VERWIJDEREN	23
DE VERWIJDERLINK.....	23
DE BEVESTIGINGSPAGINA	24
HET VERWIJDEREN UITVOEREN	25
6. DE APPLICATIE BEVEILIGEN.....	26
HET LOGIN-FORMULIER	26
DE LOGIN VERWERKEN	26
PAGINA'S BEVEILIGEN	29
UITLOGGEN.....	30
7. BESTANDEN UPLOADEN	31
HOE WERKT EEN UPLOAD?.....	31
DE UPLOAD-MAP	31
HET UPLOAD-FORMULIER	32
DE UPLOAD VERWERKEN	33
DE FOTO TONEN	36

INLEIDING

Voor je ligt een “opfrisreader” voor PHP. In deze reader worden eerder behandelde technieken uit het eerste leerjaar nog eens onder de loep genomen en behandeld om te testen hoe ver je bent met PHP en om de zaken die je nu zou moeten weten en kunnen op te frissen.

In deze reader ga je een volledige CRUD-applicatie maken waarbij alle technieken voor basisapplicaties in PHP gebruikt zullen worden. Aan het einde leer je nog wat nieuws wat veel in applicaties gebruikt wordt: het uploaden van bestanden met PHP.

WERKWIJZE EN BEOORDELING

INDELING

Deze reader is opgedeeld in een aantal hoofdstukken. Je krijgt per hoofdstuk van je docent een introductie over het onderwerp van dat hoofdstuk, waarna je het hoofdstuk verder zelfstandig gaat doorwerken.

BEOORDELING

Deze reader bevat GEEN oefeningen of opdrachten. In plaats daarvan word je beoordeeld op je totale applicatie, volgens onderstaand schema (het aantal punten is het maximaal te behalen punten per onderdeel):

Bestand	Omschrijving	Punten
config.inc.php	Configuratie en database login	5
home.php	Ledenlijst	5
	Link naar inschrijfformulier en uitlogpagina.	5
	Links per lid naar bewerk- en verwijderpagina	5
lid_nieuw.php	Inschrijfformulier	5
lid_nieuw_verwerk.php	Inschrijfformulier verwerken	5
lid_bewerk.php	Bewerkformulier	5
lid_bewerk_verwerk.php	Bewerkformulier verwerken	5
lid_verwijder.php	Verwijdering bevestigen	5
lid_verwijder_verwerk.php	Verwijdering uitvoeren	5
index.php	Inlogformulier	5
login.php	Login verwerken	5
logout.php	Uitloggen verwerken	5
session.inc.php	Controle op login	5
Basisdeel		70

Dit betekent dat je voor het geheel uitvoeren van alleen het basisdeel van deze reader maximaal een **7,0** kunt behalen.

Je kunt nog 20 extra punten (voor maximaal een **9,0**) krijgen voor het extra hoofdstuk over file uploads:

Bestand	Omschrijving	Punten
home.php	Link per lid naar de fotopagina	5
foto.php	Upload-formulier	5
	Foto tonen	5
foto_verwerk.php	Upload verwerken	5
Uitbreiding		20

Je docent kan nog maximaal 10 extra punten (voor een **10,0**) toewijzen voor extra werk dat je zelf toevoegt.

PLANNING

- Week 1** Introductie, de applicatie plannen en de hoofdpagina maken
- Week 2** Nieuwe leden toevoegen
- Week 3** Leden bewerken
- Week 4** Leden verwijderen en de applicatie beveiligen
- Week 5** Bestanden uploaden
- Week 6** Bestanden uploaden
- Week 7** Uitloop

1. EEN CRUD APPLICATIE

In deze reader gaan we een CRUD-applicatie maken. Hiervoor moet je wel eerst weten wat CRUD is en hoe je een applicatie ontwerpt.

WAT IS CRUD?

CRUD is een acroniem (afkortingswoord) voor:

- **Create**
- **Read**
- **Update**
- **Delete**

Dit beschrijft de vier basishandelingen die een applicatie met “zijn” data kan doen: Maken, lezen, wijzigen en verwijderen. Je ziet, als je de SQL opdrachten om met data te werken er naast zet, dat deze ook op deze functies zijn toegespitst:

Handeling	CRUD	SQL
Maken	Create	INSERT
Lezen	Read	SELECT
Wijzigen	Update	UPDATE
Verwijderen	Delete	DELETE

Bijna alle applicaties die je maakt gebruiken de vier stappen van CRUD wel ergens in de code. Het is dan ook een makkelijk ezelsbruggetje om te onthouden om zo voor jezelf te controleren of je applicatie wel compleet is (oftewel: alle stappen van CRUD kunnen worden uitgevoerd).

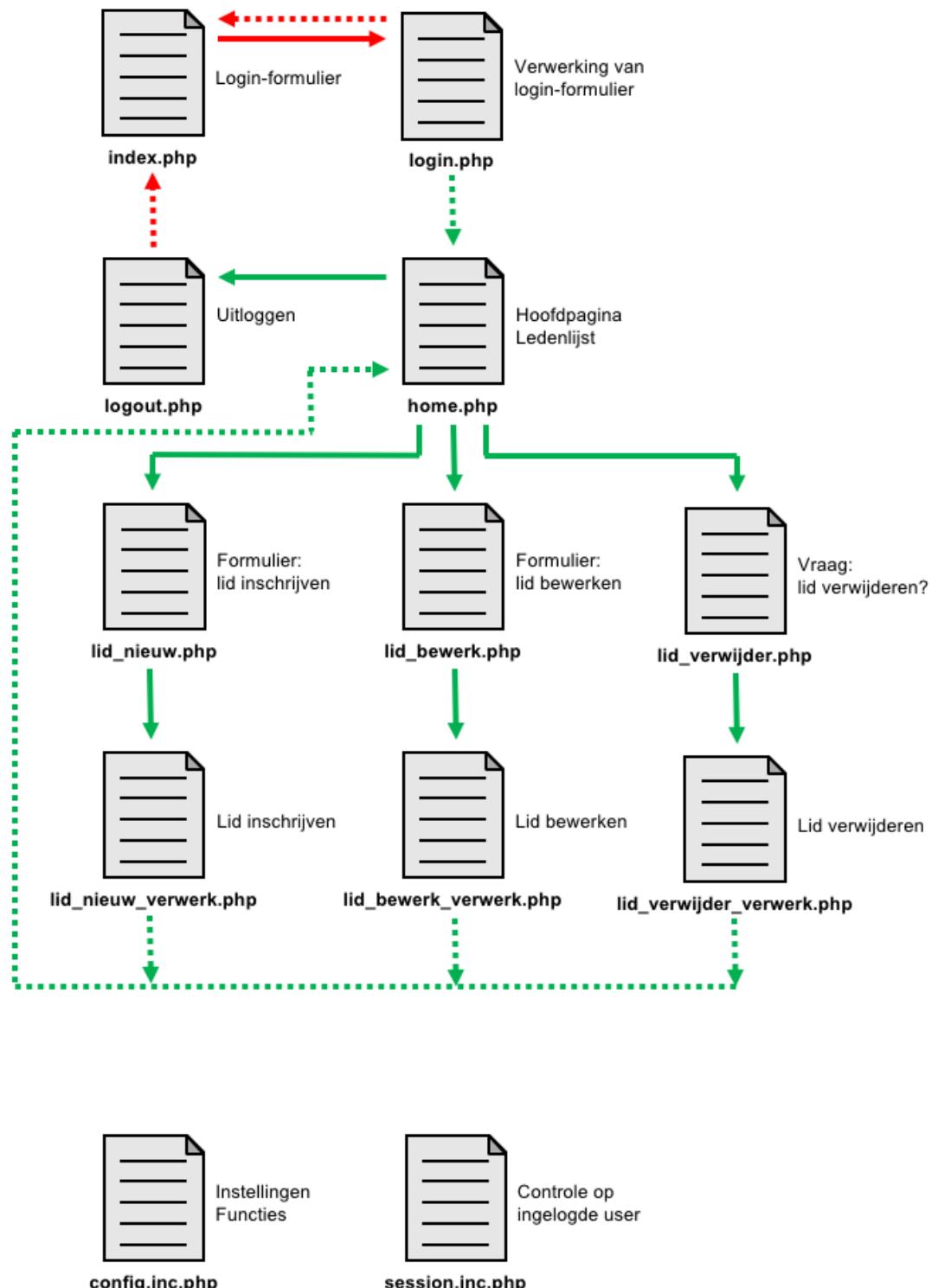
ONZE APPLICATIE: LEDENBEHEER

In deze reader gaan we een complete CRUD applicatie maken voor ledenbeheer. De applicatie is bedoeld voor de mensen die de ledenadministratie doen van een tennisclub. Deze mensen moeten kunnen:

Handeling	Onderdeel
Leden inschrijven	Create
Leden en ledenlijst bekijken	Read
Leden aanpassen	Update
Leden uitschrijven	Delete

Daarnaast moeten de gebruikers verplicht inloggen om zo de ledenadministratie te beschermen.

Voordat je begint aan het bouwen van een applicatie moet je eerst natuurlijk een ontwerp maken. Dit ontwerp gebruik je om vooraf zeker te weten dat je alle eisen hebt meegenomen en tijdens het project gebruik je de planning als “checklist” van wat je moet doen. Je start altijd bij de **index.php** en zet bij iedere pagina wat hij doet en hoe hij heet. Voor onze applicatie hebben we al een ontwerp voor je gemaakt:



In dit schema zie je dat:

- Een gebruiker begint op **index.php**, waar een login-formulier wordt getoond.
- De login wordt verwerkt door **login.php**.
- Uitloggen wordt verwerkt door **logout.php**.
- De hoofdpagina, **home.php**, toont de volledige ledenlijst.
- Er is een formulier voor het inschrijven van een nieuw lid, **lid_nieuw.php**, dat wordt verwerkt door een apart script, **lid_nieuw_verwerk.php**.

- Er is een formulier voor het bewerken van een lid, **lid_bewerk.php**, dat wordt verwerkt door een apart script, **lid_bewerk_verwerk.php**.
- Er is een pagina waar voor het verwijderen van het lid een bevestiging wordt gevraagd, **lid_verwijder.php**, waar als de gebruiker op “ja” klikt wordt doorverwezen naar **lid_verwijder_verwerk.php** waar het lid werkelijk verwijderd wordt.
- Er zijn twee scripts die worden gekoppeld die een gebruiker nooit ziet, maar die wel door de applicatie worden gebruikt. Dit noemen we soms ook wel “include files”, en zijn in deze applicatie herkenbaar aan de extensie **.inc.php** om ze te onderscheiden van de overige pagina’s.
In deze applicatie hebben we er twee:
 - **config.inc.php** bevat alle instellingen (database login, etc.)
 - **session.inc.php** wordt gebruikt voor de beveiliging

In dit schema geven de pijlen ook handelingen en beveiliging aan:

- Een **groene** pijl betekent dat een gebruiker ingelogd is, een **rode** pijl dat hij (nog) niet ingelogd is.
- Een solide pijl is een gebruikershandeling (klikken op een link of een formulier met de “submit” knop insturen), een gestippelde pijl is een automatische doorverwijzing (“redirect”) door de applicatie.

Zo zie je dat een schema als dit je de werking van de applicatie toont, je deze als checklist kunt gebruiken of je alle bestanden en functies hebt, en dat je meteen ziet welke files je wel of niet moet beveiligen. Aan het einde van deze reader hebben we al deze bestanden en functies gemaakt.

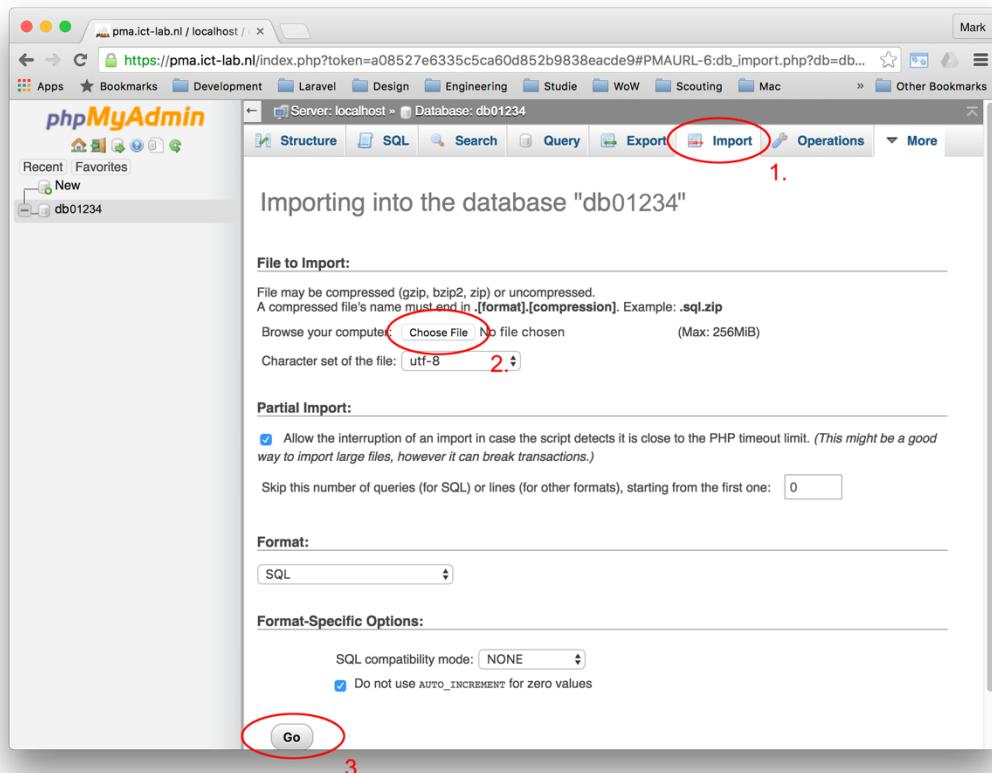
2. DE STRUCTUUR VAN DE APP BOUWEN

Maak op je eigen website een map “**MMPH4**”, dit wordt de hoofdmap van de applicatie waarin we alle bestanden gaan maken en opslaan.

DE TABEL

Voordat we aan de applicatie gaan werken moeten we eerst zorgen dat we wat data beschikbaar hebben. Bij deze reader hoort het bestand **mphp4_leden.sql.txt**, hierin staan de SQL opdrachten om de tabel **mphp4_leden** te maken.

Log in met je eigen username en (database!) password op PHPmyAdmin op het adres <https://pma.ict-lab.nl>. Selecteer je database, klik het tabblad “Import”, kies het bestand **mphp4_leden.sql.txt** en klik “Go”.



Als alles goed gegaan is heb je nu de tabel **mphp4_leden** tot je beschikking. Als je de structuur van deze tabel bekijkt zie je het volgende:

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id	int(11)		No	None	AUTO_INCREMENT	
2	birth_date	date		No	None		
3	first_name	varchar(14)	utf8_unicode_ci	No	None		
4	last_name	varchar(16)	utf8_unicode_ci	No	None		
5	gender	enum('M', 'F')	utf8_unicode_ci	No	None		
6	member_since	date		No	None		

Als eerste zie je dat het veld **id** de primary key is (dit zie je aan het feit dat de veldnaam onderstreept is), dat dit veld van het type **INT** is en dus alleen hele getallen bevat, en dat dit veld een “**AUTO_INCREMENT**” toevoeging heeft, wat inhoudt dat dit veld automatisch met een volgnummer wordt gevuld door de database-server.

INFO

De regel bij het ontwerpen van een tabel is dat een tabel met data **altijd** een primary key heeft, en dat die primary key los staat van de inhoud van het record (dus altijd “extra” data is). Hierdoor weet je twee dingen zeker:

- Ieder record heeft een uniek gegeven, waardoor je altijd het juiste record kunt vinden en selecteren
- Ook al verandert alle data in het record, de primary key blijft altijd gelijk

Deze punten zijn niet alleen belangrijk voor het vinden van de juiste data in de tabel, maar worden later nog belangrijker voor het koppelen van data in relaties.

De velden **birth_date** en **member_since** zijn van het type **DATE**. Let op dat een database-server een datum altijd opslaat als “**YYYY-MM-DD**”, dus als een volledig omgekeerde datum. Dit wordt zo gedaan zodat er makkelijk op datum gesorteerd kan worden.

De velden **first_name** en **last_name** zijn van het type **VARCHAR**, dit is het standaard veldtype voor tekst tot maximaal 255 tekens lang.

Het type van het **gender** veld is misschien nieuw voor je: een **ENUM** (kort voor “enumeration”) betekent een “lijst”. Dit veld is dus een tekstveld, maar er mag alleen iets uit de in het ontwerp vastgestelde waardes worden ingevuld en niets anders. In deze tabel mag er dus alleen een “M” of een “F” in dit veld worden ingevuld.

DE CONFIG EN HOMEPAGE MAKEN

Maak in de map “**MPHP4**” map twee nieuwe bestanden, **config.inc.php** en **home.php**. We gaan nu eerst de applicatie maken en later pas beveiligen, vandaar dat we beginnen bij de homepage.

Open eerst **config.inc.php**, en zet daarin de vier gegevens die je nodig hebt om in te loggen in je MySQL database in variabelen (gebruik natuurlijk jouw eigen gegevens):

```
<?php
// database logingegevens
$db_hostname = 'localhost';
$db_username = '01234';
$db_password = 'mijnpass';
$db_database = 'db01234';
```

Deze belangrijke gegevens zet je in een centraal config-bestand zodat je ze makkelijk kunt beheren. Als je login-gegevens ooit wijzigen hoeft je ze alleen in dit ene bestand aan te passen en zal je hele applicatie automatisch de nieuwe gegevens gebruiken.

INFO

Een variabele begint altijd met een dollarteken (\$). De naam van een variabele mag bestaan uit letters, cijfers en underscores (_) en is hooflettergevoelig. De naam van een variabele mag niet beginnen met een cijfer.

We willen eigenlijk ook graag dat als het config-bestand gelezen wordt er meteen automatisch verbinding met de database wordt gemaakt. Voeg daarom onderaan je **config.inc.php** de volgende code toe:

```
// maak de database-verbinding
$mysqli = mysqli_connect($db_hostname, $db_username, $db_password, $db_database);
```

Deze opdracht, **mysqli_connect()**, maakt de verbinding met de database-server, in dit geval met de login-gegevens die we eerder hebben benoemd.

Als dit bestand **config.inc.php** nu wordt uitgevoerd (en er zijn geen fouten), is de verbinding naar de database altijd beschikbaar als de variabele **\$mysqli**.

Open het bestand **home.php** en maak dit een HTML5 pagina met een titel en een kop. Voeg helemaal bovenin de pagina ook een PHP blok toe waarin je de net gemaakte config-file inleest:

```
<?php
// lees het config-bestand
require_once 'config.inc.php';

?><!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Ledenbeheer</title>
</head>

<body>

    <h1>Ledenbeheer</h1>

</body>
</html>
```

Test deze pagina in je browser om er zeker van te zijn dat er geen foutmeldingen verschijnen.

INFO

Om een PHP bestand in een ander PHP bestand in te lezen en uit te voeren zijn er vier mogelijke opdrachten die je kunt gebruiken:

- **include**
- **include_once**
- **require**
- **require_once**

Het verschil tussen een **include** en een **require** is hoe PHP omgaat met fouten. Als een bestand niet kan worden gevonden of geopend zal bij een **include** een foutmelding worden getoond, maar PHP gaat wel verder met de rest van het script. Bij gebruik van een **require** wordt ook een foutmelding getoond, maar PHP houdt ook meteen op met de uitvoer van het script.

Beide opdrachten hebben ook een variant met “**_once**” er achter. Deze controleert eerst of het externe bestand in de huidige pagina al eens is ingeladen en zal het externe bestand alleen inladen als dat nog niet eerder is gedaan.

We gebruiken in onze pagina nu **require_once** omdat we zeker willen weten dat het config-bestand wordt geladen voordat we verder gaan, maar het hoeft maar één keer geladen te worden.

DE LEDEN LEZEN EN TONEN

We willen op de homepage een lijst van alle leden in een tabel tonen, gesorteerd op achternaam. Hiervoor moeten we eerst alle rijen uit de tabel lezen met een SQL query:

```
// lees alle leden uit de tabel
$result = mysqli_query($mysqli, "SELECT * FROM mphp4_leden ORDER BY last_name ASC");
```

De functie **mysqli_query()** voert, zoals de naam al aangeeft, een SQL query uit. Deze functie verwacht twee parameters: de eerste is de verbinding die moet worden gebruikt voor de query, de tweede is de tekst van de query die moet worden uitgevoerd. De verbinding is in ons geval in de **config.inc.php** al gemaakt en opgeslagen in de variabele **\$mysqli**.

De variabele **\$result** bevat nu het volledige resultaat van de query (als het goed is 39 leden), maar niet in een leesbaar formaat. We noemen dit een “*result set*”, en deze moeten we eerst uitlezen (“*fetchen*”) naar een formaat waar we wel wat mee kunnen.

Wat we in de pagina willen weergeven is een tabel met de informatie van alle leden er in. Wat we in de code voor elkaar moeten krijgen zijn de volgende stappen:

- Start de tabel
- Start een tabelrij voor de kopjes
- Maak een tabelcel voor ieder kopje
- Sluit de tabelrij voor de kopjes
- Voor ieder lid doe:
 - Start een tabelrij
 - Maak een tabelcel per gegeven
 - Sluit de tabelrij
- Als alle leden zijn toegevoegd: Sluit de tabel

In PHP code (die we in de <body> van de pagina na de kop zetten) ziet dat er zo uit:

```

<?php
// start de tabel
echo "<table>";

// start een tabelrij voor de kopjes
echo "<tr>";

// maak de cellen voor de kopjes
echo "<th>ID</th>";
echo "<th>Geslacht</th>";
echo "<th>Voornaam</th>";
echo "<th>Achternaam</th>";
echo "<th>Geboortedatum</th>";
echo "<th>Lid sinds</th>";

// sluit de tabelrij voor de kopjes
echo "</tr>";

// loop door alle rijen data heen
while ($row = mysqli_fetch_array($result)) {
    // start een tabelrij
    echo "<tr>";

    // maak de cellen voor de gegevens
    echo "<td>" . $row['id'] . "</td>";
    echo "<td>" . $row['gender'] . "</td>";
    echo "<td>" . $row['first_name'] . "</td>";
    echo "<td>" . $row['last_name'] . "</td>";
    echo "<td>" . $row['birth_date'] . "</td>";
    echo "<td>" . $row['member_since'] . "</td>";

    // sluit de tabelrij
    echo "</tr>";
}

// sluit de tabel
echo "</table>";

?>

```

De belangrijkste code in dit blok is de werkelijke verwerking van de result set:

```
while ($row = mysqli_fetch_array($result)) {
```

In deze regel gebeuren drie dingen tegelijk:

- De functie **mysqli_fetch_array()** leest 1 rij uit de result set **\$result**
- De rij wordt opgeslagen als de **array \$row**
- De **while()** herhaalt deze stappen zo lang er nog data in **\$result** zit

Het resultaat is dus dat één voor één alle rijen uit de result set worden uitgelezen en per stuk telkens tijdelijk opgeslagen in de array **\$row**. Binnen in de **while()** lus kun je dus over de velden van de rij beschikken door de velden in deze array uit te lezen. Het veld “**first_name**” krijg je dus door het element **\$row['first_name']** te gebruiken.

Als je de pagina nu test zou deze, als alles goed is gegaan, de hele ledenlijst moeten tonen, netjes geordend in een tabel.

3. NIEUWE LEDEN TOEVOEGEN

Nu we de homepage van onze applicatie klaar hebben, kunnen we de volgende stap maken: het toevoegen van leden.

Maak twee nieuwe bestanden aan: een HTML5 pagina met de naam **lid_nieuw.php** en een leeg PHP bestand met de naam **lid_nieuw_verwerk.php**. De eerste gaat het formulier bevatten waarin de gegevens van nieuwe leden zullen worden ingevoerd, de tweede verwerkt dit formulier en voegt het nieuwe lid toe aan de database.

HET FORMULIER

Open **lid_nieuw.php** en voeg aan de body een formulier toe waarin het nieuwe lid kan worden ingevoerd:

```
<h1>Nieuw lid inschrijven</h1>

<form action="lid_nieuw_verwerk.php" method="post">

    <p>
        <label>
            <input type="radio" name="gender" id="gender_m" value="M" checked="checked">
            Man</label>
        <br>
        <label>
            <input type="radio" name="gender" id="gender_f" value="F">
            Vrouw</label>
    </p>

    <p>
        <label for="first_name">Voornaam:</label>
        <input type="text" name="first_name" id="first_name" required="required">
    </p>

    <p>
        <label for="last_name">Achternaam:</label>
        <input type="text" name="last_name" id="last_name" required="required">
    </p>

    <p>
        <label for="birth_date">Geboortedatum:</label>
        <input type="date" name="birth_date" id="birth_date" required="required">
    </p>

    <p>
        <label for="member_since">Lid sinds:</label>
        <input type="date" name="member_since" id="member_since" required="required">
    </p>

    <p>
        <input type="submit" name="submit" id="submit" value="Opslaan">
        <button onclick="history.back();return false;">Annuleren</button>
    </p>
</form>
```

Je ziet dat ieder veld dezelfde naam heeft als de overeenkomstige velden in de database. Dit hoeft niet, maar maakt het jou als developer wel makkelijker om data te benoemen en herkennen.

TIP

Geef al je variabelen, veldnamen, bestanden e.d. altijd namen die iets betekenen en die iets zeggen over het doel en/of de inhoud van de variabele. Een variabele met de naam **\$voornaam** is een stuk duidelijker dan een variabele met de naam **\$fv1**, net als dat een pagina met de naam **lid_nieuw.php** meer zegt over de functie van de pagina dan bijvoorbeeld **untitled_2.php**.

Het formulier is nu klaar voor gebruik maar de gebruiker kan er nog niet makkelijk komen. Open daarom je **home.php** en voeg onderaan een link naar deze pagina toe:

```
<p><a href="lid_nieuw.php">Klik hier</a> om een nieuw lid in te schrijven.</p>
```

HET FORMULIER VERWERKEN

In het formulier in de pagina **lid_nieuw.php** heb je de volgende **<form>** tag gemaakt:

```
<form action="lid_nieuw_verwerk.php" method="post">
```

De twee parameters van deze **<form>** tag geven aan dat de ingevulde data uit het formulier verstuurd moet worden d.m.v. de opdracht **post** aan het bestand **lid_nieuw_verwerk.php**. Deze gaan we nu verder maken.

Open het bestand **lid_nieuw_verwerk.php**, maak het helemaal leeg en voeg bovenaan de volgende code toe:

```
<?php
// lees het config-bestand
require_once 'config.inc.php';
```

Hiermee hebben we alvast de verbinding met de database geopend en klaar staan voor gebruik. Nu kunnen we de formulervelden gaan lezen. Omdat het formulier de "method" **post** gebruikt kunnen we de data uitlezen m.b.v. **\$_POST**:

```
// lees alle formulervelden
$gender      = $_POST['gender'];
$first_name  = $_POST['first_name'];
$last_name   = $_POST['last_name'];
$birth_date  = $_POST['birth_date'];
$member_since = $_POST['member_since'];
```

Hoewel we in het formulier alle velden verplicht hebben gemaakt moeten we toch nog eens controleren of alle velden wel ingevuld waren om later fouten met de communicatie met de database te voorkomen. We kunnen in een **if** natuurlijk als eerste checken of alle variabelen wel een lengte (en dus inhoud) hebben. Hiervoor gebruiken we de functie **strlen()** die de lengte van een string meet:

```
// controleer of alle formulervelden waren ingevuld
if (strlen($first_name)    > 0 &&
    strlen($last_name)     > 0 &&
    strlen($birth_date)    > 0 &&
    strlen($member_since)  > 0 &&
    strlen($gender)        > 0) {
```

Als de code goed door deze controle heenkomt moeten we kijken of de twee kalenderdata (**birth_date** en **member_since**) wel bestaande kalenderdata zijn (en

niet bijvoorbeeld een ongeldige datum zoals 42 Augustus). Voor deze controle gebruiken we de functies **strtotime()** en **date()**.

De functie **strtotime()** maakt een *timestamp* van een datum-string, dus bijvoorbeeld "2015-04-25" wordt "1429912800" en "2015-05-14 06:10:00" wordt "1431576600". Deze timestamps gebruik je als invoer voor de **date()** functie om de datum van dat specifieke moment te krijgen. Door de uitvoer daarvan te vergelijken met de origineel ingevoerde datum kunnen we zien of de kalenderdata "kloppen":

```
// controleer of de data wel correct zijn
$check1 = strtotime($birth_date);
$check2 = strtotime($member_since);
if (date('Y-m-d', $check1) == $birth_date && date('Y-m-d', $check2) == $member_since) {
```

Na deze controle weten we dat alle variabelen goede gegevens bevatten, zodat we nu de query kunnen maken en uitvoeren om het nieuwe lid werkelijk toe te voegen:

```
// alle data zijn OK, maak de query
$query = "INSERT INTO mphp4_leden
          (gender, first_name, last_name, birth_date, member_since)
          VALUES (
            '$gender',
            '$first_name',
            '$last_name',
            '$birth_date',
            '$member_since')";

// voer de query uit
$result = mysqli_query($mysqli, $query);
```

Nu zou het nieuwe lid moeten zijn toegevoegd aan de database.

INFO

Je kunt bij een string (letterlijke tekst) gebruik maken van dubbele quotes ("") of enkele quotes (''). Er is in PHP wel een verschil tussen deze twee: Als je een variabele tussen dubbele quotes zet wordt de inhoud van de variabele in de string gezet, tussen enkele quotes wordt de naam van de variabele getoond.

Voorbeelden:

echo \$a; ← dit toont de inhoud van \$a
echo “\$a” ← dit toont ook de inhoud van \$a
echo ‘\$a’ ← dit toont de tekst “\$a”

In de code hierboven zie je dat de variabele **\$query** dubbele quotes gebruikt. De enkele quotes daarbinnen worden letterlijk doorgegeven in de MySQL query en de variabele-namen worden vervangen door de inhoud van de variabelen.

Na het uitvoeren van de query moeten we wel controleren of het uitvoeren van de query goed is gegaan:

```
// controleer het resultaat
if ($result) {
    // alles OK, stuur terug naar de homepage
    header("Location:home.php");
    exit;
} else {
    echo 'Er ging wat mis met het toevoegen!';
}
```

Hier gebruiken we, als de query goed is uitgevoerd, de functie **header()** om de gebruiker automatisch door te sturen naar de homepage. Als alles goed gaat ziet de gebruiker dus (bijna) niet dat hij vanuit het formulier naar de pagina **lid_nieuw_verwerk.php** wordt doorgestuurd want hij komt bijna meteen terug op de homepage waar meteen het nieuw toegevoegde lid wordt getoond.

INFO

Opdrachten die werken met de HTTP header, zoals bijvoorbeeld **session_start()** en **header()**, moeten worden uitgevoerd **voordat** er output naar de browser wordt gestuurd!

De volledige code van deze pagina is:

```
<?php
// lees het config-bestand
require_once 'config.inc.php';

// lees alle formuliervelden
$gender      = $_POST['gender'];
$first_name  = $_POST['first_name'];
$last_name   = $_POST['last_name'];
$birth_date  = $_POST['birth_date'];
$member_since = $_POST['member_since'];

// controleer of alle formuliervelden waren ingevuld
if (strlen($first_name)  > 0 &&
    strlen($last_name)   > 0 &&
    strlen($birth_date)  > 0 &&
    strlen($member_since) > 0 &&
    strlen($gender)       > 0) {

    // controleer of de data wel correct zijn
    $check1 = strtotime($birth_date);
    $check2 = strtotime($member_since);
    if (date('Y-m-d', $check1) == $birth_date &&
        date('Y-m-d', $check2) == $member_since) {

        // alle data zijn OK, maak de query
        $query = "INSERT INTO mphp4_leden
                  (gender, first_name, last_name, birth_date, member_since)
                  VALUES (
                    '$gender',
                    '$first_name',
                    '$last_name',
                    '$birth_date',
                    '$member_since')";

        // voer de query uit
        $result = mysqli_query($mysqli, $query);

        // controleer het resultaat
        if ($result) {
            // alles OK, stuur terug naar de homepage
            header("Location:home.php");
            exit;
        } else {
            echo 'Er ging wat mis met het toevoegen!';
        }
    } else {
        // er is iets mis met een datum
        echo 'Een van de ingevulde data was ongeldig!';
    }
} else {
    echo 'Niet alle velden waren ingevuld!';
}
```

4. LEDEN BEWERKEN

In onze applicatie moeten we natuurlijk bestaande leden ook kunnen wijzigen. Hiervoor hebben we drie zaken nodig:

- De link van het lid naar het bewerkformulier
- Het bewerkformulier
- Het script om het bewerkformulier te verwerken

DE BEWERKLINK

Om van een specifiek lid te kunnen linken naar het formulier waarin de gegevens van het juiste lid staan moeten we het bewerkformulier kunnen openen met het ID van het lid. Het ID van het juiste record moet dus “meereizen” met wat de gebruiker doet:

- De “bewerk”-link die de gebruiker klikt geeft het ID mee in de URL
- Het bewerkformulier bevat het ID in een verborgen veld, waardoor het bij het opsturen van het formulier ook mee wordt gegeven

Op de homepage willen we dus achter ieder lid een “bewerk” link tonen die het juiste ID doorgeeft. Open het bestand **home.php** en pas de code die de tabel maakt aan door in de header een extra lege cel toe te voegen en in de dataregels ook een extra cel toe te voegen met daarin een link naar het bewerkformulier (zie bij de pijlen):

```
<?php
// start de tabel
echo "<table>";

// start een tabelrij voor de kopjes
echo "<tr>";

// maak de cellen voor de kopjes
echo "<th>ID</th>";
echo "<th>Geslacht</th>";
echo "<th>Voornaam</th>";
echo "<th>Achternaam</th>";
echo "<th>Geboortedatum</th>";
echo "<th>Lid sinds</th>";
echo "<th></th>"; ←

// sluit de tabelrij voor de kopjes
echo "</tr>";

// loop door alle rijen data heen
while ($row = mysqli_fetch_array($result)) {
    // start een tabelrij
    echo "<tr>";

    // maak de cellen voor de gegevens
    echo "<td>" . $row['id'] . "</td>";
    echo "<td>" . $row['gender'] . "</td>";
    echo "<td>" . $row['first_name'] . "</td>";
    echo "<td>" . $row['last_name'] . "</td>";
    echo "<td>" . $row['birth_date'] . "</td>";
    echo "<td>" . $row['member_since'] . "</td>";
    echo "<td><a href='lid_bewerk.php'>bewerk</a></td>"; ←

    // sluit de tabelrij
    echo "</tr>";
}

// sluit de tabel
echo "</table>";

?>
```

Nu staat er in het overzicht wel een link, maar deze link geeft nog niet het ID van het lid door. Hiervoor passen we de link als volgt aan:

```
echo "<td><a href='lid_bewerk.php?id=" . $row['id'] . "'>bewerk</a></td>";
```

INFO

Een variabele doorgeven in de URL (en dus de link) van een pagina doe je door achter de bestandsnaam een vraagteken te zetten, gevolgd door de naam van de variabele, een =-teken en de inhoud van de variabele. Bijvoorbeeld:

pagina1.php?naam=Piet

Als je meerdere variabelen door wilt geven kun je alle opvolgende variabelen koppelen met een ampersand (&), zoals in dit voorbeeld:

pagina1.php?naam=Piet&nummer=34&geslacht=M

Hier zie je dat er drie variabelen worden doorgegeven: naam, nummer en geslacht.

Nu de link klopt kunnen we het bewerkformulier gaan maken.

HET FORMULIER

We hoeven het formulier om een lid te bewerken natuurlijk niet helemaal opnieuw te maken, we hebben eerder al de basis voor dit formulier gemaakt. Open het bestand **lid_nieuw.php** en sla deze op als **lid_bewerk.php**.

Als eerste moeten we nu aan dit bestand de PHP code toevoegen om het juiste lid uit de database te lezen, waarbij we het via de URL doorgegeven ID gebruiken om te weten welk lid we moeten uitlezen. Voeg de volgende PHP code toe helemaal bovenaan de pagina (dus nog voor de **doctype**):

```
<?php
// lees het config-bestand
require_once 'config.inc.php';

// lees het ID uit de URL
$id = $_GET['id'];
```

Hier zie je dat het config-bestand wordt gelezen (want we willen zo met de database communiceren), en daarna meteen het doorgegeven ID uit de URL wordt gelezen.

We willen nu wel even checken of het ID wel een nummer is, om fouten te voorkomen. Hiervoor gebruiken we de functie **is_numeric()**:

```
// is het ID een nummer?
if (is_numeric($id)) {
```

Als dit goed gaat kunnen we het lid gaan proberen uit de database te lezen:

```
// lees het lid uit de database
$result = mysqli_query($mysql, "SELECT * FROM mp4_leden WHERE id = $id");

// is er een lid gevonden met dit ID?
if (mysqli_num_rows($result) == 1) {
    // ja, lees het lid uit de dataset
    $row = mysqli_fetch_array($result);
} else {
    echo "Geen lid gevonden.";
    exit;
}
```

We gebruiken hier de functie **mysqli_num_rows()** om eerst te kijken hoeveel rijen data er zijn gevonden door de query. Aangezien we een specifiek ID hebben geselecteerd kan er maximaal 1 rij worden gevonden, dus met een check op dat aantal zie je snel of er iemand is gevonden.

Als alles goed gaat is aan het einde van dit blokje code de array **\$row** gevuld met de gegevens van het juiste lid. Deze gegevens moeten we nu in het formulier tonen zodat ze gewijzigd kunnen worden.

Als eerste moeten we de *action* van het formulier aanpassen naar **lid_bewerk_verwerk.php**:

```
<form action="lid_bewerk_verwerk.php" method="post">
```

Nu willen we in dit formulier het ID van de gebruiker ook meegeven naar het verwerkingsscript, omdat we daar anders niet weten welk lid we moeten aanpassen. Omdat een formulier met een POST wordt verstuurd kan het echter niet via de URL, dus de beste manier is het ID als een verborgen formulieveld mee te geven:

```
<input type="hidden" name="id" value="php echo $id; ?&gt;"&gt;</pre

```

INFO

In het algemeen geldt: Een link of (getypte) URL wordt geopend met het HTTP commando GET, een formulier wordt ingestuurd met een POST.

In PHP gebruik je voor deze gevallen de specifieke arrays **\$_GET** en **\$_POST** om de variabelen uit te lezen. Het is onveilig om **\$_REQUEST** te gebruiken!

De volgende stap is het aanpassen van de *radio group* voor het geslacht, zodat deze de juiste selectie toont. Een radio button of checkbox in een HTML formulier kun je aangevinkt tonen door de attribuut **checked** de inhoud "checked" te geven:

```
<p>
    <label>
        <input type="radio" name="gender" id="gender_m" value="M"
            <?php if ($row['gender'] == 'M') echo 'checked="checked"'; ?>
        Man</label>
    <br>
    <label>
        <input type="radio" name="gender" id="gender_f" value="F"
            <?php if ($row['gender'] == 'F') echo 'checked="checked"'; ?>
        Vrouw</label>
</p>
```

En hierna kunnen alle invulvelden de juiste inhoud krijgen door in hun **value** de juiste data met **echo** te tonen:

```
<p>
    <label for="first_name">Voornaam:</label>
    <input type="text" name="first_name" id="first_name" required="required"
    value=<?php echo $row['first_name']; ?>>
</p>

<p>
    <label for="last_name">Achternaam:</label>
    <input type="text" name="last_name" id="last_name" required="required"
    value=<?php echo $row['last_name']; ?>>
</p>

<p>
    <label for="birth_date">Geboortedatum:</label>
    <input type="date" name="birth_date" id="birth_date" required="required"
    value=<?php echo $row['birth_date']; ?>>
</p>

<p>
    <label for="member_since">Lid sinds:</label>
    <input type="date" name="member_since" id="member_since" required="required"
    value=<?php echo $row['member_since']; ?>>
</p>
```

Hiermee is de pagina goed aangepast voor het bewerken van leden, de rest van de pagina blijft ongewijzigd.

Voor de volledigheid nog de hele inhoud van de pagina op rij:

```
<?php
// lees het config-bestand
require_once 'config.inc.php';

// lees het ID uit de URL
$id = $_GET['id'];

// is het ID een nummer?
if (is_numeric($id)) {
    // lees het lid uit de database
    $result = mysqli_query($mysqli, "SELECT * FROM mphp4_leden WHERE id = $id");

    // is er een lid gevonden met dit ID?
    if (mysqli_num_rows($result) == 1) {
        // ja, lees het lid uit de dataset
        $row = mysqli_fetch_array($result);
    } else {
        echo "Geen lid gevonden.";
        exit;
    }
} else {
    // het ID was geen nummer
    echo "Onjuist ID.";
    exit;
}

?><!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Ledenbeheer</title>
</head>
```

```

<body>

    <h1>Lid bewerken</h1>

    <form action="lid_bewerk_verwerk.php" method="post">

        <input type="hidden" name="id" value=<?php echo $id; ?>>

        <p>
            <label>
                <input type="radio" name="gender" id="gender_m" value="M"
                    <?php if ($row['gender'] == 'M') echo 'checked="checked"'; ?>
                    Man</label>
            <br>
            <label>
                <input type="radio" name="gender" id="gender_f" value="F"
                    <?php if ($row['gender'] == 'F') echo 'checked="checked"'; ?>
                    Vrouw</label>
        </p>

        <p>
            <label for="first_name">Voornaam:</label>
            <input type="text" name="first_name" id="first_name" required="required"
                value=<?php echo $row['first_name']; ?>>
        </p>

        <p>
            <label for="last_name">Achternaam:</label>
            <input type="text" name="last_name" id="last_name" required="required"
                value=<?php echo $row['last_name']; ?>>
        </p>

        <p>
            <label for="birth_date">Geboortedatum:</label>
            <input type="date" name="birth_date" id="birth_date" required="required"
                value=<?php echo $row['birth_date']; ?>>
        </p>

        <p>
            <label for="member_since">Lid sinds:</label>
            <input type="date" name="member_since" id="member_since" required="required"
                value=<?php echo $row['member_since']; ?>>
        </p>

        <p>
            <input type="submit" name="submit" id="submit" value="Opslaan">
            <button onclick="history.back();return false;">Annuleren</button>
        </p>

    </form>

</body>
</html>

```

HET FORMULIER VERWERKEN

Open het bestand **lid_nieuw_verwerk.php** en sla het op als **lid_bewerk_verwerk.php**. In dit bestand moeten we nu wat zaken aanpassen, als eerste moet het ID worden toegevoegd aan de formuliervelden die worden ingelezen:

```
// lees alle formuliervelden
$id      = $_POST['id'];
$gender   = $_POST['gender'];
$first_name = $_POST['first_name'];
$last_name = $_POST['last_name'];
$birth_date = $_POST['birth_date'];
$member_since = $_POST['member_since'];
```

Hierna moeten we dit veld ook toevoegen aan de controles die we doen op de inhoud van de velden. Voor het ID voegen we twee checks toe, een voor de lengte en een om te kijken op het ID wel een nummer is (net als bij de vorige pagina):

```
// controleer of alle formuliervelden waren ingevuld
if (is_numeric($id) &&
    strlen($id)          > 0 &&
    strlen($first_name)  > 0 &&
    strlen($last_name)   > 0 &&
    strlen($birth_date)  > 0 &&
    strlen($member_since) > 0 &&
    strlen($gender)       > 0) {
```

Het laatste onderdeel dat moet worden aangepast is de SQL query, zodat deze het bestaande lid aanpast o.b.v. het ID i.p.v. het toevoegen van een nieuw lid:

```
// alle data zijn OK, maak de query
$query = "UPDATE mphp4_leden
          SET
            gender = '$gender',
            first_name = '$first_name',
            last_name = '$last_name',
            birth_date = '$birth_date',
            member_since = '$member_since'
          WHERE id = $id";
```

De rest van de pagina blijft gelijk (behalve mogelijk nog de foutmeldingen), dus hiermee is dit bestand klaar voor gebruik.

5. LEDEN VERWIJDEREN

Om leden te kunnen verwijderen hebben we weer drie elementen nodig:

- Een link van het lid naar de bevestigingspagina
- De bevestigingspagina
- Het script om het verwijderen uit te voeren

DE VERWIJDERLINK

Net als bij de bewerklink uit het vorig hoofdstuk maken we ook nu weer een link die het ID van een lid via de URL meestuurt.

Open het bestand **home.php** en voeg weer een headercel en een datacel toe met de juiste link naar **lid_verwijder.php** (zie bij de pijlen):

```
<?php
// start de tabel
echo "<table>";

// start een tabelrij voor de kopjes
echo "<tr>";

// maak de cellen voor de kopjes
echo "<th>ID</th>";
echo "<th>Geslacht</th>";
echo "<th>Voornaam</th>";
echo "<th>Achternaam</th>";
echo "<th>Geboortedatum</th>";
echo "<th>Lid sinds</th>";
echo "<th></th>";
echo "<th></th>"; ←

// sluit de tabelrij voor de kopjes
echo "</tr>";

// loop door alle rijen data heen
while ($row = mysqli_fetch_array($result)) {
    // start een tabelrij
    echo "<tr>";

    // maak de cellen voor de gegevens
    echo "<td>" . $row['id'] . "</td>";
    echo "<td>" . $row['gender'] . "</td>";
    echo "<td>" . $row['first_name'] . "</td>";
    echo "<td>" . $row['last_name'] . "</td>";
    echo "<td>" . $row['birth_date'] . "</td>";
    echo "<td>" . $row['member_since'] . "</td>";
    echo "<td><a href='lid_bewerk.php?id=" . $row['id'] . "'>bewerk</a></td>";
    echo "<td><a href='lid_verwijder.php?id=" . $row['id'] . "'>verwijder</a></td>"; ←

    // sluit de tabelrij
    echo "</tr>";
}

// sluit de tabel
echo "</table>";

?>
```

DE BEVESTIGINGSPAGINA

Maak een nieuwe HTML5 pagina met de naam **lid_verwijder.php**. Voeg helemaal bovenaan de pagina, nog voor de *doctype*, een PHP blok en voeg de volgende code toe:

```
// lees het config-bestand
require_once 'config.inc.php';

// lees het ID uit de URL
$id = $_GET['id'];

// is het ID een nummer?
if (is_numeric($id)) {
    // lees het lid uit de database
    $result = mysqli_query($mysqli, "SELECT * FROM mphp4_leden WHERE id = $id");

    // is er een lid gevonden met dit ID?
    if (mysqli_num_rows($result) == 1) {
        // ja, lees het lid uit de dataset
        $row = mysqli_fetch_array($result);
    } else {
        echo "Geen lid gevonden.";
        exit;
    }
} else {
    // het ID was geen nummer
    echo "Onjuist ID.";
    exit;
}
```

Het valt je misschien op dat deze code identiek is aan de code bovenin **lid_bewerk.php**, en dat klopt: we doen namelijk precies hetzelfde: op basis van het ID in de URL het juiste lid uitlezen.

Deze gegevens kunnen we nu gebruiken om in de **<body>** de juiste bevestigingstekst te tonen:

```
<h1>Lid verwijderen</h1>

<p>
    Weet je zeker dat je het lid
    <strong><?php echo $row['first_name'] . " " . $row['last_name']; ?></strong>
    wilt verwijderen?
</p>

<p>
    <a href="lid_verwijder_verwerk.php?id=<?php echo $id; ?>">Ja, verwijderen</a>
    /
    <a href="home.php">Nee, terug</a>
</p>
```

Ook hier zie je dat het ID weer via de URL aan de volgende pagina wordt doorgegeven in de “ja”-link.

HET VERWIJDEREN UITVOEREN

Maak een nieuw leeg bestand aan met de naam **lid_verwijder_verwerk.php**. Dit script gaat de verwijdering van het lid werkelijk uitvoeren. Geef dit bestand de volgende inhoud:

```
<?php
// lees het config-bestand
require_once 'config.inc.php';

// lees het ID uit de URL
$id = $_GET['id'];

// is het ID een nummer?
if (is_numeric($id)) {
    // verwijder het lid uit de database
    $result = mysqli_query($mysqli, "DELETE FROM mphp4_leden WHERE id = $id");

    // controleer het resultaat
    if ($result) {
        // alles OK, stuur terug naar de homepage
        header("Location:home.php");
        exit;
    } else {
        echo 'Er ging wat mis met het verwijderen!';
    }
} else {
    // het ID was geen nummer
    echo "Onjuist ID.";
    exit;
}
```

Je ziet dat alle elementen, behalve de SQL query, al eerder in onze applicatie zijn gebruikt.

6. DE APPLICATIE BEVEILIGEN

Bijna alle applicaties waarin met gegevens wordt omgegaan vereisen een vorm van beveiliging. Je wilt beheer van je gegevens in de applicatie natuurlijk niet zo maar open en bloot door iedereen laten uitvoeren. Hierom gaan we nu een login en sessiebeheer aan de applicatie toevoegen.

Maak een nieuwe HTML5 pagina aan met de naam **index.php**, en drie lege bestanden met de namen **login.php**, **logout.php** en **session.inc.php**.

Maak in je database ook een nieuwe tabel aan door de volgende SQL code uit te voeren:

```
CREATE TABLE IF NOT EXISTS `mphp4_users` (
  `id` int(11) NOT NULL,
  `username` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(32) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

ALTER TABLE `mphp4_users` ADD PRIMARY KEY (`id`);
ALTER TABLE `mphp4_users` MODIFY `id` int(11) NOT NULL
AUTO_INCREMENT;
ALTER TABLE `mphp4_users` ADD UNIQUE(`username`);
INSERT INTO `mphp4_users` (`id`, `username`, `password`)
VALUES (NULL, 'admin', MD5('geheim'));
```

HET LOGIN-FORMULIER

Open de pagina **index.php**. Hieraan voegen we een redelijk standaard login-formulier toe:

```
<h1>Inloggen</h1>

<form action="login.php" method="post">

  <p>
    <label for="username">Gebruiker:</label>
    <input type="text" name="username" id="username" required="required">
  </p>

  <p>
    <label for="password">Wachtwoord:</label>
    <input type="password" name="password" id="password" required="required">
  </p>

  <p>
    <input type="submit" name="submit" id="submit" value="Inloggen">
  </p>

</form>
```

That's it, meer is er niet nodig voor het login-formulier.

DE LOGIN VERWERKEN

Open het bestand **login.php**. Dit bestand gaat de ingevoerde username en password controleren met de gebruikerstabel die we eerder hebben gemaakt.

Als eerste moet dit script natuurlijk verbinding maken met de database en dan de ingevoerde formuliervelden uitlezen en controleren op invoer:

```
<?php
// lees het config-bestand
require_once 'config.inc.php';

// lees alle formuliervelden
$username = $_POST['username'];
$password = $_POST['password'];

// controleer of alle formuliervelden waren ingevuld
if (strlen($username) > 0 && strlen($password) > 0) {
```

Nu moeten we het wachtwoord eerst “versleutelen” (ook wel “encrypten”), want een wachtwoord als “cleartext” leesbaar opslaan in de database is natuurlijk vreselijk onveilig. Je moet altijd minimaal een eenrichtings-encryptie op het wachtwoord loslaten, zoals MD5 of SHA1

INFO

MD5 en SHA1 zijn eenrichtings-encryptieprotocollen, wat inhoudt dat ze een tekst (zoals een wachtwoord) versleutelen op zo’n manier dat het resultaat onleesbaar is geworden, en ook dat het niet meer “terug kan worden gerekend”. Op deze manier zorgen ze er voor dat zelf als iemand in je database kan hij niet makkelijk de wachtwoorden van je gebruikers achterhaalt.

MD5 levert een string op van 32 tekens lang, SHA1 een string van 40 tekens lang.

Om een tekst met MD5 te versleutelen gebruik je in PHP eenvoudigweg de functie **md5()**:

```
// versleutel het wachtwoord
$password = md5($password);
```

Nu kunnen we het wachtwoord vergelijken met het versleutelde wachtwoord in de database. Als je de SQL queries op de vorige pagina goed leest zie je dat we al een gebruiker aan de gebruikerstabel hebben toegevoegd met de username **admin** en het wachtwoord **geheim** (dat met MD5 is versleuteld).

```
// maak de controle-query
$query = "SELECT * FROM mphp4_users
          WHERE username = '$username'
          AND password = '$password';

// voer de query uit
$result = mysqli_query($mysqli, $query);

// controleer of de login correct was
if (mysqli_num_rows($result) == 1) {
```

Als alles goed is gegaan is er door de query 1 user gevonden die voldoet aan de username en het versleutelde password. De **if()** controleert of er inderdaad precies 1 user is gevonden door de query met de functie **mysql_num_rows()**.

Als de user de juiste inloggegevens heeft ingevoerd kunnen we nu de sessie starten en de username opslaan in de sessie:

```
// login correct, start de sessie
session_start();

// sla de username op in de sessie
$_SESSION['username'] = $username;
```

INFO

Normaal “weet” een PHP niets van andere scripts die de gebruiker al heeft uitgevoerd, iedere keer dat een pagina wordt geladen begint PHP geheel schoon en blanco, en als hij klaar is worden alle bestanden en variabelen weer uit het geheugen gewist. Om toch gegevens te kunnen bewaren tussen verschillende scripts kun je gebruik maken van sessies. Een sessie is een soort mini-opslag op de webserver waar je variabelen in kunt schrijven en weer uitlezen.

De webserver bewaakt dat iedere bezoeker van je site zijn eigen beveiligde sessie heeft. Als een gebruiker enige tijd (meestal 30 minuten) niets meer doet op je website wordt de sessie automatisch verwijderd.

Om in je PHP scripts gebruik te kunnen maken van de sessie, moet je altijd eerst de functie **session_start()** uitvoeren.

De in de sessie opgeslagen username gaan we op de overige pagina's gebruiken om te controleren of er wel een geldige gebruiker is ingelogd, maar dat komt iets later.

Nu we de sessie hebben gevuld met de username kunnen we de gebruiker doorsturen naar de homepage:

```
// stuur door naar de homepage
header("Location:home.php");
```

Het hele script ziet er als volgt uit:

```

<?php
// lees het config-bestand
require_once 'config.inc.php';

// lees alle formuliervelden
$username = $_POST['username'];
$password = $_POST['password'];

// controleer of alle formuliervelden waren ingevuld
if (strlen($username) > 0 && strlen($password) > 0) {
    // versleutel het wachtwoord
    $password = md5($password);

    // maak de controle-query
    $query = "SELECT * FROM mphp4_users
              WHERE username = '$username'
              AND password = '$password'";

    // voer de query uit
    $result = mysqli_query($mysqli, $query);

    // controleer of de login correct was
    if (mysqli_num_rows($result) == 1) {
        // login correct, start de sessie
        session_start();

        // sla de username op in de sessie
        $_SESSION['username'] = $username;

        // stuur door naar de homepage
        header("Location:home.php");
    } else {
        // login incorrect, terug naar het login-formulier
        header("Location:index.php");
        exit;
    }
} else {
    echo "Niet alle velden zijn ingevuld!";
    exit;
}

```

PAGINA'S BEVEILIGEN

Nu een gebruiker kan inloggen moeten we nog de pagina's van de applicatie beveiligen. Nu kun je de code die de sessie controleert natuurlijk kopiëren en plakken naar alle PHP bestanden, maar het is makkelijker om, net als bij de config, een include-bestand te maken zodat je met 1 regel per pagina klaar bent.

Open het bestand **session.inc.php**, hierin gaan we de login controleren. Het doel is om:

- als er geen geldige login is de bezoeker terug te sturen naar het inlogformulier
- dit helemaal aan het begin van alle beveiligde pagina's te controleren

Om te beginnen moeten we eerst de sessie starten:

```
// start de sessie
session_start();
```

Hiermee is de sessie klaar voor gebruik, en kunnen we gaan controleren of de username correct is opgeslagen in de sessie:

```
// controleer of er een username is opgeslagen
if (!isset($_SESSION['username']) || strlen($_SESSION['username']) == 0) {
    // geen geldige login, stuur naar het inlogformulier
    header("Location:index.php");
    exit;
}
```

Je ziet dat de **if** twee dingen controleert: of de variabele **\$_SESSION['username']** niet bestaat, of dat de variabele geen inhoud heeft (lengte 0). Als een van deze twee voorwaarden waar is, is er wat mis en wordt de bezoeker meteen naar de inlogpagina gestuurd.

Het script **session.inc.php** is hiermee al helemaal klaar, het moet alleen nog goed worden ingezet.

Voeg aan alle bestanden die moeten worden beveiligd (dus alle pagina's behalve **index.php**, **login.php**, **logout.php**, **config.inc.php** en **session.inc.php**) helemaal bovenin, voor alle andere code en inhoud, de volgende regel toe:

```
require_once 'session.inc.php';
```

Door het inlezen van deze file is een pagina automatisch beveiligd.

UITLOGGEN

Als je een gebruiker laat inloggen, moet je ook de mogelijkheid bieden om uit te loggen. Open het bestand **logout.php** en geef het deze inhoud:

```
<?php
// start de sessie
session_start();

// vernietig de sessie
session_destroy();

// ga naar de inlogpagina
header("Location:index.php");
```

Dat is alles wat nodig is om uit te loggen: de sessie starten, dan de sessie vernietigen met **session_destroy()** en dan, om het netjes af te handelen, de gebruiker naar het inlogformulier terugsturen.

Als laatste heb je alleen nog een link nodig naar **logout.php**, anders kan de gebruiker er natuurlijk niet makkelijk komen. Open **home.php** en voeg helemaal onderaan de pagina de link toe:

```
<p>Je bent ingelogd als <?php echo $_SESSION['username']; ?><br>
<a href="logout.php">Klik hier</a> om uit te loggen.</p>
```

7. BESTANDEN UPLOADEN

Nu we het basisideel van het ledenbeheer af hebben willen we graag nog wat extra's toevoegen: we willen per gebruiker een foto kunnen uploaden.

HOE WERKT EEN UPLOAD?

Een bestandupload begint met een <input> van het type "file" in een formulier. Dit wordt door een webbrowser vertaald naar een bestandsveld waarmee de gebruiker een bestand op zijn PC kan uitkiezen.

Wanneer het formulier wordt ingestuurd zal de browser automatisch een upload naar de webserver starten. Als dit lukt zal de webserver het bestand op een tijdelijke plek opslaan, en aan PHP doorgeven waar het geuploade bestand staat.

In PHP moet je een aantal stappen doorlopen:

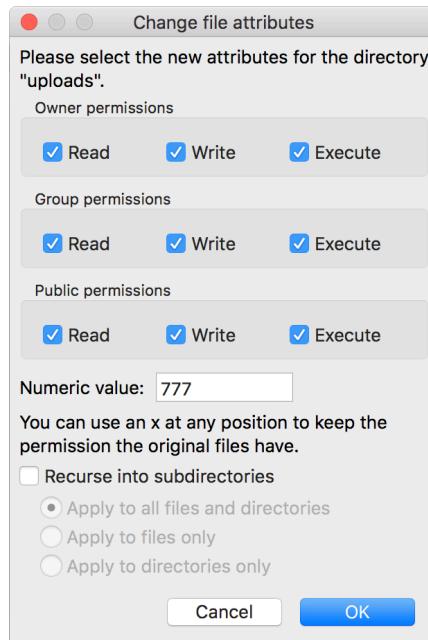
- Het bestand controleren (bijvoorbeeld op type, grootte, etc.)
- Bepalen in welke map het bestand moet komen te staan
- Bepalen hoe het bestand moet gaan heteren
- Het bestand met de juiste naam in de juiste map plaatsen

Vaak zul je de plaats en naam van het bestand ook in een database willen opslaan, maar dat is natuurlijk niet verplicht.

DE UPLOAD-MAP

Normaliter heeft PHP op je website geen schrijf-toegang, dit zou een beveiligingsrisico kunnen zijn, maar voor uploads moeten we een uitzondering kunnen maken.

Maak binnen je map **MPHP4** een nieuwe map met de naam **uploads**. Om deze map van de juiste rechten te kunnen voorzien kun je het beste een FTP client gebruiken, bijvoorbeeld FileZilla. Log met FileZilla in op je website op <ftp://ict-lab.nl>, blader naar de net gemaakte map **uploads**, rechts-klik op de map en kies "File permissions". Stel de permissions in zoals aangegeven:



HET UPLOAD-FORMULIER

Maak een nieuw bestand met de naam **foto.php**, in dit bestand gaan we de bestaande foto tonen voor een user, en een formulier om een nieuwe foto te uploaden.

Open het bestand **home.php** en voeg weer een headercel en een datacel toe met de juiste link naar **foto.php** inclusief het ID van de user (zie bij de pijlen):

```

<?php
// start de tabel
echo "<table>";

// start een tabelrij voor de kopjes
echo "<tr>";

// maak de cellen voor de kopjes
echo "<th>ID</th>";
echo "<th>Geslacht</th>";
echo "<th>Voornaam</th>";
echo "<th>Achternaam</th>";
echo "<th>Geboortedatum</th>";
echo "<th>Lid sinds</th>";
echo "<th></th>";
echo "<th></th>"; ←

// sluit de tabelrij voor de kopjes
echo "</tr>";

// loop door alle rijen data heen
while ($row = mysqli_fetch_array($result)) {
    // start een tabelrij
    echo "<tr>";

    // maak de cellen voor de gegevens
    echo "<td>" . $row['id'] . "</td>";
    echo "<td>" . $row['gender'] . "</td>";
    echo "<td>" . $row['first_name'] . "</td>";
    echo "<td>" . $row['last_name'] . "</td>";
    echo "<td>" . $row['birth_date'] . "</td>";
    echo "<td>" . $row['member_since'] . "</td>";
    echo "<td><a href='lid_bewerk.php?id=" . $row['id'] . "'>bewerk</a></td>";
    echo "<td><a href='lid_verwijder.php?id=" . $row['id'] . "'>verwijder</a></td>";
    echo "<td><a href='foto.php?id=" . $row['id'] . "'>foto</a></td>"; ←

    // sluit de tabelrij
    echo "</tr>";
}

// sluit de tabel
echo "</table>";

?>

```

Nu kunnen we in **foto.php** de code toevoegen om het ID uit de URL te lezen, die we daarna aan het formulier gaan toevoegen. Voeg de volgende PHP code toe helemaal bovenaan de pagina (dus nog voor de **doctype**):

```
// Lees het ID uit de URL
$id = $_GET['id'];
```

Voeg nu het upload-formulier toe in de body van de pagina:

```
<form action="foto_verwerk.php" method="post" enctype="multipart/form-data">

    <input type="hidden" name="id" value=<?php echo $id; ?>>

    <p>
        <label for="foto">Foto:</label>
        <input type="file" name="foto" id="foto" required="required">
    </p>

    <p>
        <input type="submit" name="submit" id="submit" value="uploaden">
        <button onclick="history.back();return false;">Annuleren</button>
    </p>

</form>
```

Let op dat de **enctype** van het formulier goed wordt ingesteld, anders weet de browser mogelijk niet hoe hij het formulier met het bestand moet versturen!

DE UPLOAD VERWERKEN

Maak een leeg PHP bestand aan met de naam **foto-verwerk.php**. In dit bestand gaan we de volgende stappen doorlopen:

1. Controleren van de upload
2. Controleren van het bestandstype
3. Bepalen van de map en de bestandsnaam
4. Het bestand op de juiste plek zetten
5. De gebruiker terugsturen naar de foto-pagina

Controleren van de upload

Als PHP van de webserver doorkrijgt dat er een of meerdere bestanden zijn geupload wordt alle info over de upload(s) per bestand opgeslagen in de array **\$_FILES**. In ons formulier hebben we de <input> de naam **foto** gegeven, dus alle info zit in de array **\$_FILES['foto']**. Deze kun je als test uitlezen door de inhoud van de array op het scherm te dumpen:

```
// Lees de informatie over de upload
$bestand = $_FILES['foto'];

// TEST: Dump de info op het scherm
echo "<pre>";
var_dump($bestand);
echo "</pre>";
```

Dit geeft bijvoorbeeld het volgende te zien in de browser:

```
array(5) {
    ["name"]=>
        string(11) "pasfoto.jpg"
    ["type"]=>
        string(10) "image/jpeg"
    ["tmp_name"]=>
        string(14) "/tmp/php5ebb8T"
    ["error"]=>
        int(0)
    ["size"]=>
        int(900136)
}
```

Hieraan zie je dat de informatie over een upload altijd 5 elementen bevat:

- **Name:** de naam van het originele bestand
- **Type:** het MIME type van het bestand
- **Tmp_name:** de map en naam van het tijdelijke bestand
- **Error:** De foutcode (0 = geen fouten)
- **Size:** de bestandsgrootte in bytes

Om de upload te controleren kijk je dus in eerste instantie of de array **\$_FILES** wel info bevat over je upload, en daarna of de foutcode “0” is (vergeet niet de testcode te verwijderen!):

```
// Controleer of de upload geslaagd is
if (isset($_FILES['foto']) && $_FILES['foto']['error'] == 0) {
```

Controleren van het bestandstype

Als de controle van de upload goed gaat kun je controleren of de upload van het juiste type is. Dit is natuurlijk niet verplicht, maar in dit geval willen we alleen JPG bestanden accepteren.

Het bestandstype kun je het snelst controleren aan de hand van het MIME type. Ieder soort bestand heeft een of meerdere mogelijke MIME types, JPG bestanden kunnen een van deze drie hebben:

- image/jpg
- image/jpeg
- image/pjpeg

We controleren nu dus of het geuploade bestand van het juiste type is:

```
// Controleer het bestandstype
if ($_FILES['foto']['type'] == "image/jpg" ||
    $_FILES['foto']['type'] == "image/jpeg" ||
    $_FILES['foto']['type'] == "image/pjpeg") {
```

Bepalen van de map en de bestandsnaam

We weten uit de array **\$_FILES** al waar het bestand nu tijdelijk (!) is opgeslagen, maar we moeten nog bepalen waar het bestand terecht moet komen en hoe het moet gaan heten.

Als eerste hebben we de **fysieke** locatie (de werkelijke plek op de disk van de webserver) nodig van de map waar het bestand naartoe moet, in ons geval de map **uploads**. Gelukkig schiet PHP ons daar ook te hulp, dit keer met een “magic constant”. De constante **__DIR__** geeft je altijd de fysieke map op de server waar het

huidige PHP bestand zich bevindt. We weten dat de map **uploads** in de zelfde map staat als **foto_verwerk.php**, dus we kunnen met **DIR** de map van deze pagina opvragen en daar dan de mapnaam **uploads** aan “vastplakken”:

```
// Wat is de fysieke locatie van de uploads-map?
$map = __DIR__ . "/uploads/";
```

We willen in dit geval per lid maar één foto bewaren, dus dan is het het makkelijkste om het bestand het ID van de user als bestandsnaam te geven, dan kunnen we het bestand altijd snel en makkelijk terugvinden. Het ID zat als hidden veld in het formulier, dus dat kunnen we simpel gebruiken:

```
// Maak de bestandsnaam
$bestand = $_POST['id'] . '.jpg';
```

Nu is alles gecontroleerd en klaar voor de volgende stap:

Het bestand op de juiste plek zetten

Om een upload in PHP op de juiste plek te zetten gebruik je de eenvoudige functie **move_uploaded_file()**:

```
// Verplaats de upload naar de juiste map met de juiste naam
move_uploaded_file($_FILES['foto']['tmp_name'], $map . $bestand);
```

Hiermee is de upload verwerkt en kunnen we de gebruiker terugsturen naar de foto-pagina (met natuurlijk het ID van de user weer in de URL):

```
// Stuur de gebruiker terug naar de foto-pagina
header("Location:foto.php?id=" . $_POST['id']);
```

De hele pagina ziet er nu zo uit:

```
<?php
// Controleer of de upload geslaagd is
if (isset($_FILES['foto']) && $_FILES['foto']['error'] == 0) {

    // Controleer het bestandstype
    if ($_FILES['foto']['type'] == "image/jpg" ||
        $_FILES['foto']['type'] == "image/jpeg" ||
        $_FILES['foto']['type'] == "image/pjpeg") {

        // Wat is de fysieke locatie van de uploads-map?
        $map = __DIR__ . "/uploads/";

        // Maak de bestandsnaam
        $bestand = $_POST['id'] . '.jpg';

        // Verplaats de upload naar de juiste map met de juiste naam
        move_uploaded_file($_FILES['foto']['tmp_name'], $map . $bestand);

        // Stuur de gebruiker terug naar de foto-pagina
        header("Location:foto.php?id=" . $_POST['id']);
    } else {
        echo "Het bestand is van het verkeerde type.";
    }
} else {
    echo "Er ging iets fout bij het uploaden.";
}
```

DE FOTO TONEN

Als laatste willen we de foto-pagina nog aanpassen dat deze de foto van een lid toont, als deze er is natuurlijk.

Open **foto.php** en voeg direct onder het formulier een PHP blok toe. Hierin gaan we eerst controleren of er een foto bestaat voor dit lid, hiervoor gebruiken we de functie **file_exists()** in combinatie met de al eerder gebruikte “magic constant” **__DIR__**:

```
<?php
// Is er al een foto voor dit lid?
if (file_exists(__DIR__ . '/uploads/' . $id . '.jpg')) {
```

Als het bestand inderdaad bestaat genereren we de HTML code in de pagina om het plaatje te tonen, het hele blok ziet er dan zo uit:

```
<?php
// Is er al een foto voor dit lid?
if (file_exists(__DIR__ . '/uploads/' . $id . '.jpg')) {

    echo "<p><img src='uploads/" . $id . ".jpg' alt='foto'></p>";
}

?>
```