# PHP 3

Herhalen PHP basis, OOP, PDO en MVC en klaarstomen voor Laravel.
Gegeven aan klas OITAOO8B (in 2019)

- 1.0 Wat weet ik? - test jezelf
- 1.1 Opdracht Arrays en Loops
- 1.2 Assiociative Arrays
- 1.2.1 Assiociative Arrays
- 1.3 Herhaling
- 2.0 Hoofdstedenspel
- 2.1 Extra - oefeningen
- 3 Functions
- 3.1 van functie naar object (oop)
- 4.1 Calculator+ deel 1 (OOP)
- 4.2 Calculator+ deel 2 (cookies)
- 5.0 PDO en Sessies
- 5.1 PDO class - MVC
- 5.1 CRUD - Read
- 5.2 CRUD - Create
- 5.3 CRUD - Update
- 5.4 CRUD - Delete
- 7.1 Bankrekeningnummer - check

- 7.2 Winst of verlies bij 3 x 6
- 8 Hoofdstedenspel deel-2
- 9 Corona simulatie in PHP
- 10 The Challenge ***
- Alle PHP Videos
- Complete CRUD in PHP
- CRUD Challenge
- PHP exec python script

# 1.0 Wat weet ik? - test jezelf

In deze les staat telkens een opdracht, als je die niet goed kunt maken wordt er verwezen naar een pagina met extra uitleg.

Maak alle opgaven en als je de stof nog niet helemaal begrijpt, bekijk dan de extra uitleg.

# 1 Forms

- Maak een html form en vraag de gebruiker om zijn <naam> en geboortejaar. Post het form en bereken de <leeftijd> van de gebruiker. Post het form naar een PHP script en antwoord met "Welcome <naam>, you are <leeftijd> years old."

Mocht je dit moeilijk vinden  kijk dan voor meer uitleg over forms:

https://www.roc.ovh/link/210

Weet je het verschil tussen GET en POST? Kun je beide technieken toepassen?

https://www.guru99.com/php-forms-handling.html

# 2 Arrays

- Maak een array en zet daar de waarden "rood", "blauw" en "groen" in. Druk de waarden af.

Moeilijk? Voor meer uitleg over arrays:

https://www.roc.ovh/link/168

# 3 Assiociative Arrays

- Maak een assiociative array waarbij de key de namen "Yuba", "Anna" en "Ayoub" bevat. De waarde van elke element is het behaalde cijfer voor het vak PHP. Yuba heeft een 5, Anna een 8 en Ayoub een 6.

Is dit lastig lees dan eer uitleg over Assiociative Arrays op:

https://www.roc.ovh/link/170

# 4.1 For Loops

We onderscheiden for loops en while loops. Ze doen allemaal ongeveer hetzelfde. Ze voeren een stuk code meerdere maken uit.

- Maak een for loop waarin alle waarden 100, 99, 98,....2,1,0 worden afgedrukt. Dus alle getallen van 100 terugtellend naar 0.

Lastig:

https://www.roc.ovh/link/169

# 4.2 While Loops

- Maak een do-while loop waarmee je alle even getallen onder de 100 afdrukt, dus 0, 2, 4, 6, ...,98, 100.

Voor meer uitleg over de while-do en do-while:

https://www.roc.ovh/link/207

--

# 1.1 Opdracht Arrays en Loops

*Oefenen met arrays en loops.*

We weten hoe arrays en loops werken. Met deze combinatie kun je veel handige dingen coderen.

Eerst een voorbeeld:

```
<?php
$directors = array( "Alfred Hitchcock", "Stanley Kubrick", "Martin Scorsese", "Fritz Lang" );
for($i=0; $i<count($directors); $i++){
echo $directors[$i]."<br>";
}
?>
```

Een paar zaken zijn belangrijk:

1. Het eerste array element is element 0, de loop begint dus met 0!

2. De count() van het array is het aantal elementen, in dit voorbeeld 4.

3. De loop stopt bij $i=4 dus de laatste iteratie is met $i=3. Even checken....bestaat element $directors[3]? En is het inderdaad het laatste element?

Nu jij:

1. Maak een array met de dagen van de week: maandag, dinsdag,... etc. Maak een loop en druk de dagen onder elkaar af.

2. Gebruik het array uit de vorige opdracht maar druk nu het volgende af: "Dag 1 is maandag", "Dag 2 is dinsdag", etc.

3. Druk nu de dagen van de week af in omgekeerde volgorde: dus begin met zondag, dan zaterdag, etc.

4. Gebruik nu een loop om de tafel van 7 af te drukken, dus
   1 X 7 = 7
   2 X 7 = 14
   3X 7 = 21
   , ...
   Vervang de 7 voor een variabele en zet deze boven in je code. Druk nu de tafel van 127 af.

# 1.2 Assiociative Arrays

*Je gaat leren wat een assiociative array is en hoe je de inhoud hiervan kunt afdrukken in een tabel.*

Associative arrays zijn belanrijk want resultaten van een query uit de database worden door PDO vaak als associcative array terug gegeven. Hier gaan we later (in PHP PDO) ook mee oefenen. Ook zal dit terug komen in Laravel.

We hebben in de vorige les 'gewone' arrays gezien, een 'gewoon' array kun je aanspreken via bijvoorbeeld:

```
$array[0]
```

Dit is het eerste element van het $array. De 0 is de index, dat geeft aan welk element je aanspreekt.

Bij een associative array heb je ook een index, maar de index is geen nummer maar een naam, we noemen dat een sleutel. Bijvoorbeeld:

```
$array['PHP']
```

Dit is het element in het $array met de index 'PHP'. De index in een assiociative array wordt ook wel key genoemd.

> Een index in een 'gewoon' array heet **key** in een assiociative array.

We kunnen op deze manier een heel array opbouwen, bijvoorbeeld:

```
<?php


$array['PHP']=5.5;
$array['Java']=6.5;
$array['Engels']=7.0;
$array['Nederlands']=6.5;
$array['Veilig Programmeren']=8.0;
```

Dit zouden bijvoorbeeld de cijfers voor de verschillende vakken kunnen zijn.

Deze kun je ook op een andere manier toekennen:

```
<?php

$array=array("PHP" => 5.5, "Java"=>6.5, ....);
```

> Vraag 1: Maak nu de onderstaande code af zodat alle vakken, PHP, Java, Engels, Nederlands, en Veilig programmeren allemaal afgedrukt worden.

```
<?php

$array=array("PHP" => 5.5, "Java"=>6.5, MAAK HIER DE CODE AF);

echo "PHP:" . $array["PHP"]."<br>";
echo "Java:". $array["Java"]."<br>";
MAAK HIER DE CODE AF

?>
```

Nu werkt dit best een beetje onhandig want je zou deze vakken het liefst in een loop afdrukken. Net zoals de tafels die we in de vorige hoofdstukken in een loop maakte.

Maar we kunnen ook door een associative array met een loop. Hoe werkt dat?

De standaard manier is als volgt:

```
<?php
foreach ($array as $key => $value) {
....
}
```

In dit voorbeeld is $array het assiociative array, en is $key de key (zeg maar de index) en $value de waarde.

# Opdracht 1

Druk met behulp van een loop (foreach zoals hierboven is uitgelegd) het assiociatieve array helemaal af.

Als dt klaar is dan hebben we dus code gemaakt die het volgende afdrukt:

```
PHP:5.5
Java:6.5
Engels:7.0
Nederlands:6.5
Veilig Programmeren:8.0
```

Verander deze code zodat de gegevens in een tabel worden afgedrukt. Dus ongeveer zo:

| Vak | Cijfer |
| --- | --- |
| PHP | 5.5 |
| Engels | 7.0 |
| Nederlands | 6.5 |
| Veilig Programmeren | 8.0 |

# PDO

Je gaat leren hoe PDO de inhoud van een tabel aan jou terug geeft en hoe je deze in een HTMl tabel kan afdrukken.

Kijk eens naar de volgende code, dit is een voorbeeld van ene resultset zoals je die van PDO zou kunnen krijgen als je een query op de database uitvoerd.

```
<?php
$resultSet = array(
    array(
        'id' => "1",
        'name' => "Mo"
    ),
   array(
        'id'   => "2",
        'name' => "Angel"
    ),
   array(
        'id'   => "3",
        'name' => "Do"
    ),
    array(
        'id'   => "4",
        'name' => "Anouar"
```

```
    ),
 array(
        'id'   => "5",
        'name' => "Keyana"
    )
)
```

Wat je ziet is een 'gewoon' array van assiociative arrays. Weet je nog dat we arrays van arrays konden maken? Dat doen we nu ook in de volgende vorm (schematisch):

```
[ [assiociative array], [assiociative array], [assiociative array], [assiociative array], [assiociative array] ]
```

We hebben dus een array genaamd, $resultSet en die heeft 5 elementen, $resultSet[0], $resultSet[1],....$resultSet[5].

> Opdracht 1: maak een for-loop en doe een print_r van alle 5 de elementen van de $resultSet.

# Opdracht 2

Vervang nu de print_r zodat je nu ook de resultaten netjes in een tabel krijgt, dus de output moet er als volgt uit zien.

| Id | Naam |
|---|---|
| 1 | Mo |
| 2 | Angel |
| 3 | Do |
| 4 | Anouar |
| 5 | Keyana |

Dit zou dus de inhoud van een tabel kunnen zijn.

> We hbben nu geleerd hoe we de return value van een PDO object in een tabel kunnen weergeven!

--

Besproken in de les ... to be continued

```php
<?php

echo "<pre>";

$a = [1, 3, 5];
$b = [2, 4, 6];

$combine=[ $a, $b];
print_r($combine);

echo $combine[1][0];
echo $combine[1][1];
echo $combine[1][2];

$deref=$combine[1];
echo $deref[1];

echo "<hr>";
for($i=0; $i<3; $i++) {
  echo $combine[1][$i]."<br>";
}
echo "<hr>";

$array1=array('PHP' => 5.5, 'Java'=>6.5, 'Engels'=>7.0, 'Nederlands'=>6.5, 'Veilig Programmeren'=>8.0);
$array2=array('PHP' => 6.0, 'Java'=>7.0, 'Engels'=>7.5, 'Nederlands'=>7.0, 'Veilig Programmeren'=>8.5);
$combine=[ $array1, $array2, $array2 ];
print_r($combine);
echo $combine[1]['PHP'];
echo $combine[1]['Java'];
echo $combine[1]['Engels'];
echo "<hr>";


for($i=0; $i<count($combine); $i++) {
  echo "Cijfers persoon $i:<br>";
  foreach( $combine[$i] as $key => $value ) {
    echo "Vak: ".$key." cijfer: ".$value."<br>";
  }

}
```

# 1.2.1 Assiociative Arrays

*Herhaling Indexed array en  Assiociative Array*

*Arrays zijn er in meerdere soorten. In deze les gaan we het hebben over associatieve arrays.*

We hebben het al uitgebreid gehad over 'gewone' of 'eenvoudige' arrays in PHP. Deze 'gewone' arrays heten ook wel een indexed array omdat je de elementen via de index (0,1,2,3,4,5.....) kunt opzoeken.

Er bestaat nog een type array: het assiociative array (in Python heet dit een 'Dictionary'). Het wordt ook wel een key-value array genoemd. Het assiociative array lijkt veel op JSON. JSON is een bestandsformaat waarin je data kan opslaan. Je zou het een soort database kunnen noemen.

| | Andere namen | Hoe benader je een element? | Voorbeeld |
|---|---|---|---|
| **Indexed Array** | gewoon array, 1 dimensionaal array, eenvoudig array | via de index, die is altijd 0,1,2,3,4.... | $myArray[13] |
| **Assiociative Array** | key-value array, Dictionary (Python), hash table | via de key, dit moet een unieke string zijn waarvan je zelf de waarde bepaald. | $myArray['james']; |

In het onderstaande filmpje wordt eerst nog een keer herhaald hoe een 'gewoon' array ook alweer werkt en vanaf 3'00'' in het filmpje wordt uitgelegd hoe een associative array in PHP werkt.

https://www.youtube.com/embed/5lJLecI0BTA

In dit filmpje wordt aan de hand van een ander voorbeeld hetzelfde uitgelegd in het Engels.

# Indexed Array

Bij een indexed array heb je als index een nummer. Elk array element heeft een nummer, te beginnen met 0.

```
$age[0] = "35";
$age[1] = "37";
$age[2] = "43";
```

Je kunt dit array ook op een ander manier maken, weet je nog?

```
$age = array(35,37,43);
```

Het bovenstaande array heeft drie elementen: $age[0], $age[1] en age{2]. Het getal tussen de vierkante haken heet de index.

# Associative Array

Stel dat je het array dat hierboven staat wilt gebruiken om de leeftijd van personen vast te leggen. $age[0] is de leeftijd van Peter, $age[1] is de leeftijd van Ben en $age[2] is de leeftijd van Joe.

```
$age[0] = "35";
$age[1] = "37";
$age[2] = "43";
```

Dit is wel wat onhandig want je moet nu onthouden dat de 0 bij Peter hoort, de 1 bij Ben en de 2 bij Joe. Dat kan anders!

## Associative Array aanmaken

Met een assiociative array kun je de index vervangen door een key. En de key bestaat uit een string.

Bijvoorbeeld:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

Je kunt het array ook op een andere manier maken

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

## Associative Uitlezen

```
echo $age['Ben'];
```

In plaats van de index bij een indexed array, gebruik je nu de key die je zelf hebt gemaakt.

## Associative Veranderen

```
$age['Ben']=38;
```

Dit gaat ook hetzelfde als bij een indexed array. Hier gebruik je de key die jezelf hebt gemaakt voor het array.

### Associative Uitbreiden

```
$age['Mirna']=28;
```

Je kunt een assiociative array eenvoudig uitbreiden door een nieuwe key te gebruiken. Je moet wel zeker weten dat de key nieuw is anders overschrijf je de oude waarde. Als in het voorbeeld $age['Mirna'] al bestond dan heb je de oude waarde nu overschreven met de nieuwe waarde. Een key kan dus maar één keer voorkomen.

# Opgave 1, theorievragen

a) In het (Nederlandse) filmpje over associative arrays staat 'ma' => 8 waar staat de 'ma' voor en waar staat de 8 voor? Dus wat betekent de 'ma' (Maarten, of wat?) en wat is de 8; 8 peren, 8 kilo, 8-wat?

b) Waaruit bestaat een index bij een index array; uit een string of een getal?

c) Waaruit bestaat een key bij een assiociative array; uit een string of uit een getal?

d) Stel een indexed array bestaat uit drie elementen. Wat zijn de indexen die dan worden gebruikt in dit array?

e) Kan dezelfde key bij een assiociative array meer dan één keer voorkomen?

f) Kan de key bij een assiociative array uit een getal zijn?

# Opgave 2

Op deze pagina staan van alle maanden van het jaar de normale gemiddelde temperatuur in Nederland vermeld. Zo kun je zien dat januari de koudste maand is met gemiddeld 3.1 graden en de warmste maand is juli met 17.9 graden.

a) Maar een assiociative array waarin van alle maanden van het jaar (januari, februari, etc.) de normale gemiddelde temperatuur staan. Je hoeft alleen het array te maken.

b) Gebruik het assiociative array dat je hebt gemaakt en druk de temperatuur af van de maand april.

# Opgave 3

Begin met de volgende code

```
<?php
$places = array(
"Japan" => "Tokyo",
   "Mexico" => "Mexico City",
   "USA" => "New York City",
   "India" => "Mumbai",
   "Korea" => "Seoul",
   "China" => "Shanghai",
   "Nigeria" => "Lagos",
   "Argentina" => "Buenos Aires",
   "Egypt" => "Cairo",
   "UK" => "London");
?>
```

- Maak code die het volgende afdrukt:
  *de hoofdstad van China is Shanghai*
  Hierbij moet de naam Shanghai worden opgezocht uit het array. Je code mag de naam Shanghai dus niet bevatten.

- Bestudeer: https://www.w3schools.com/php/php_arrays_associative.asp en zoek uit hoe je een associative array kunt uitprinten met een loop. Maak dan code die met het array van hierboven het volgende uitprint. Gebruik daarvoor een loop.

```
De hoofdstad van Japan is Tokyo
De hoofdstad van Mexico is Mexico City
De hoofdstad van USA is New York City
De hoofdstad van India is Mumbai
De hoofdstad van Korea is Seoul
De hoofdstad van China is Shanghai
De hoofdstad van Nigeria is Lagos
De hoofdstad van Argentina is Buenos Aires
De hoofdstad van Egypt is Cairo
De hoofdstad van UK is London
```

# Opgave 4

Gebruik het array dat je hebt gemaakt bij opgave 2a en druk de temperatuur van alle maanden af. Gebruik hiervoor een loop en zorg ervoor dat de output er als volgt uit ziet.

De normale gemiddelde temperatuur van januari in Nederland is 3.1 graden.
De normale gemiddelde temperatuur van februari in Nederland is 3.3 graden.
De normale gemiddelde temperatuur van maart in Nederland is 6.2 graden.
De normale gemiddelde temperatuur van april in Nederland is 9.2 graden.
De normale gemiddelde temperatuur van mei in Nederland is 13.1 graden.
De normale gemiddelde temperatuur van juni in Nederland is 15.6 graden.
De normale gemiddelde temperatuur van juli in Nederland is 17.9 graden.
De normale gemiddelde temperatuur van augustus in Nederland is 17.5 graden.
De normale gemiddelde temperatuur van september in Nederland is 14.5 graden.
De normale gemiddelde temperatuur van oktober in Nederland is 10.7 graden.
De normale gemiddelde temperatuur van november in Nederland is 6.7 graden.
De normale gemiddelde temperatuur van december in Nederland is 3.17graden.

--

# 1.3 Herhaling

Maak onderstaande opdrachten en bewaar alle oplossingen in een php file.

## Opdracht 1a

In de code staan 2 maal ??? (drie vraagtekens). Verander deze vraagtekens zodat
de getallen 1 tot en met 100 worden afgedrukt.

```
for($i= ??? ; $i< ??? ; $i++) {
    echo $i." ";
}
```

## Opdracht 1b

Doe nu hetzelfde als bij 1a maar gebruik nu een while-loop.

## Opdracht 2

Er zit een klein foutje in de onderstaande code, haal dit foutje eruit. Je kunt
de code gewoon kopieren en uitvoeren.

```
$results=[ 3,6,8,4,6,7,8,5,5,6]

foreach($results as $item) {
    echo $item;
    echo <br>;
}
```

## Opdracht 3

In de array $results van de vorige opgave staan alle cijfers van een leerling.
Bereken het gemiddelde van deze cijfers.

# Opdracht 4

Maak nu zelf een array waarin alle cijfers van 1 tot en met 1000 voorkomen.
Dus het array ziet er zo uit $array=[1,2,3,4,5.....,1000]

Maak een php programma dat eerst het array vult met alle cijfers van 1 t/m 1000
en maak daarna een loop waarin je alle cijfers uit het array optelt.

Dus geen functie array_sum() gebruiken.

# Opdracht 5

Bestudeer de code, en pas het zodanig aan dat je het gemiddelde resultaat
van alle leerlingen berekend.

```
$results=[ 'Jori' => '8', 'peter' => '6', 'Sid'=>'7', 'sarah-lin' =>'9',
    'Djab' => '5', 'Lin' => '6', 'maria'=> '7', 'Tjeerd'=> '4'];

foreach ($results as $item) {
    echo $item."
";
}
```

# Opdracht 6

Verander de loop uit opdracht 5 in een loop waarin je de key en de value van het array
kan afdrukken

foreach($results as $key => $value) { ..... }

Druk alle resultaten van alle leerlingen netjes af. Het liefst in een table

# Opdracht 7

Gebruik het array uit opdracht 5.

Zoals je ziet beginnen niet alle namen netjes met hoofdletter. Gebruik de
de functie ucfirst om de namen allemaal te laten beginnen met een hoofdletter.

# Opdracht 8

```
$results= [ 'Jori' => ['6','7','6'], 'Peter' => ['7','7','8'],
            'Sid'=> ['6','5','5'], 'Sarah-lin' =>['9','4','6'],
            'Djab' => ['7','7','7'], 'Lin' => ['6','5','6'],
            'Maria'=> ['6','7','7'], 'Tjeerd'=> ['7','7','6'] ];

foreach($results as $key => $value)
    echo $key.": ";
    var_dump($value); echo "
";
}
```

Er zit een klein foutje in de bovenstaande code, haal de fout eruit en probeer
te begrijpen wat er gebeurt.

# Opdracht 9

De code laat van heeft van alle leerlingen een array met cijfers. Bereken per
leerling het gemiddelde cijfer en druk dat per leerling af.

# Opdracht 10

Maak een form en vraag de gebruiker serie cijfers in te vullen. Scheidt deze cijfers door een spatie. Post het form,
zet de cijfers in een array en bereken het gemiddelde.

Voorbeeld, post 5.5  6.0 6.5 en je PHP code zal uitereken dat je gemiddled een 6 staat.

Hint: https://stackoverflow.com/questions/1209447/how-can-i-turn-a-list-of-items-into-a-php-array
Op deze pagina wordt beschreven hoe je een regel met woorden/getallen omzet in een array.

# 2.0 Hoofdstedenspel

*We gaan de herhalingslessen afsluiten met het maken van een eenvoudig leerspel. Later, als we hebben geleerd hoe we een database kunnen gebruiken gaan we dit spel uitbreiden.*

*Praktijkopdrachten worden individueel uitgevoerd. Je moet dus je eigen code maken. Praktijkopdrachten worden apart beoordeeld (los van het huiswerk).*

*We gaan een programma maken waarmee je de hoofdsteden van Europa kunt leren. We gaan in deze les oefenen met onder meer html forms, loops, arrays en assiociative arrays. We leren ook een groot probleem op te delen in kleinere deel-problemen.*

# Doel

Het uiteindelijke doel is om een programma te maken dat alle Europese landen en hoofdsteden kent. Het programma kiest een willekeurig land en vraagt aan jou de hoofdstad van dat land te noemen. Je kunt kiezen uit een lange lijst van hoofdsteden die alfabetisch zijn gesorteerd.

Een uitgewerkt voorbeeld is te vinden op: http://www.softwaredeveloper.ovh/max/hoofdsteden/

# Plan van aanpak - stapjes

Bij een dergelijk probleem dienen we het probleem in kleinere (deel)probleempjes op te delen. Dit kan op verschillende manieren. Zo lees je in de opdracht dat je een willekeurig land moet kiezen. We zullen dus moeten onderzoeken hoe we een willekeurig item uit een lijst kunnen kiezen. We zien ook dat we lijst moeten maken waaruit we een hoofdstad moeten kiezen. We moeten dus uitzoeken hoe we zo'n lijst moeten maken. We zien ook dat we een lijst moeten sorteren. Zo zijn er dus allemaal deelproblemen te definiëren. Deze opdracht is opgedeeld in 6 kleinere deelprobleempjes. Los deze problemen ieder afzonderlijk op en kom zo stapje voor stapje tot de uiteindelijke oplossing. Let op als je een stap niet in één keer kunt oplossen, kun je dit stapje zelf ook weer opdelen in kleinere stapjes.

## Stap 1 - form met drop down

Maak een form in HTML waarbij je een drop down maakt.
Check deze pagina en probeer te begrijpen hoe het werkt:

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_elem_select

```
<form action="/action_page.php">
  <select name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <br><br>
  <input type="submit">
</form>
```

Maak nu zelf een form met PHP waarbij je de waarden uit het array $cars haalt.

```
array=$cars('Volvo','Saab','Fiat','Audio');
```

De regels 3,4,5 en 6 uit het bovenstaande voorbeeld worden dus vervangen door PHP code.

# Stap 2 - random array element selecteren

We gaan het array iets uitbreiden.

```
array=$cars('Volvo','Saab','Fiat','Audio','BMW','Porsch','Hyundai','Opel','Kia','Tesla','VW');
```

Maak nu een PHP programma die één willekeurig automerk uit het array oppikt.

Dus je maakt een programma dat

`echo $cars[$random];`

uitvoert.

Zoek zelf uit hoe de rand() werkt waarmee je een random (willekeurig) getal kan berekenen.

Als eind resultaat heb je een php programmaatje dat elke keer als je een reload uitvoert een ander automerk laat zien.

# Stap 3 -  willekeurig land met hoofdstad

Gebruik het array zoals hieronder is aangegeven.

```
$ceu = array(
"Italy"=>"Rome", "Luxembourg"=>"Luxembourg", "Belgium"=> "Brussels", "Denmark"=>"Copenhagen",
"Finland"=>"Helsinki", "France" => "Paris", "Slovakia"=>"Bratislava", "Slovenia"=>"Ljubljana",
```

```
"Germany" => "Berlin", "Greece" => "Athens", "Ireland"=>"Dublin", "Netherlands"=>"Amsterdam",
"Portugal"=>"Lisbon", "Spain"=>"Madrid", "Sweden"=>"Stockholm", "United Kingdom"=>"London",
"Cyprus"=>"Nicosia", "Lithuania"=>"Vilnius", "Czech Republic"=>"Prague", "Estonia"=>"Tallin",
"Hungary"=>"Budapest", "Latvia"=>"Riga", "Malta"=>"Valetta", "Austria" => "Vienna",
"Poland"=>"Warsaw") ;
```

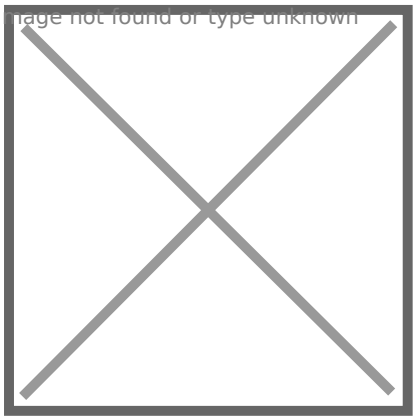
Maak een php programma dat een willekeurig land laat zien en dat dan het volgende afdrukt:

> The Capital of Finland is Helsinki

Elke keer als je de pagina ververst zie je dus een ander land.

# Stap 4 - Hoofdstad raden

Maak nu een programma dat een willekeurig land uit het array landen $ceu selecteert en de gebruiker via een form vraagt:




Controleer het antwoord en toon 'Correct' of 'Incorrect' afhankelijk van of het gegeven antwoord juist is.

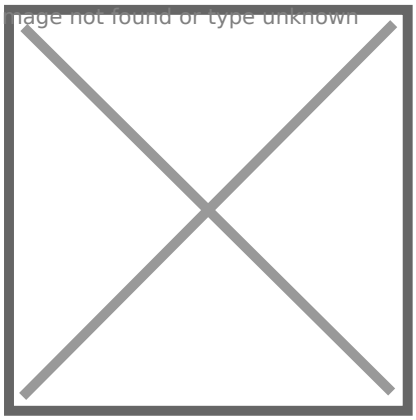
# Stap 5 - drop down

We gaan nu ons programma uit de vorige stap uitbreiden.

Maak een form dat een willekeurig land uit het array pikt. Stel je kiest 'Sweden'. Laat dan een form zien die vraagt wat de hoofdstad van Sweden is.

Zet alle hoofdsteden in een *drop down list* zodat de gebruiker een hoofdstad kan uitkiezen.

Image not found or type unknown



De gebruiker kiest dan een hoofdstad en als je op de knop *check my answer* drukt dan bepaald het PHP programma of het goed is.

Het programma antwoord:

Indien het antwoord juist is:

```
The Capital of Sweden is Stockholm, your answer was correct.
```

Indien het antwoord niet goed is:

```
The Capital of Sweden is Stockholm, your answer Riga was wrong.
```

# Stap 6 - sorteren

Als laatste stap zorgen we ervoor dat de drop down lijst met hoofdsteden wordt gesorteerd.

Hiervoor zijn diverse strategieën te bedenken. Lees en bestudeer de volgende pagina en kies jouw strategie.

https://www.bitdegree.org/learn/php-sort-array

Denk hier ook aan het opdelen in kleine stapjes. Voordat je sorteren gaat toevoegen in jouw programma, kun je eerst met eenvoudige kleine stukjes code testen hoe het sorteren werkt.

# Tips

Als je met een form werkt dan is het handig om te beginnen met een action method GET.

Voorbeeld staat in deze les. Op deze manier kun je op de URL (internet adres bovenaan in je browser) zien welke variabelen er worden meegegeven.

Je vraagt bijvoorbeeld wat de hoofdstad is van France (Paris). Stel je antwoord is Paris. Je post jouw antwoord en in de code die wordt uitgevoerd heb je nu jouw antwoord (Paris), maar is dat voldoende om te controleren of je het goed hebt? Nee dus je moet ook weten wat de vraag was, dus van welk land moest je de hoofdstad benoemen?Je zult dat dus ook mee moeten geven via de

form variabelen aan jouw (action) script.

https://www.maxdata.ovh/cnt.php?id=php1-hoofdsteden1

Als we straks met PDO hebben geleerd hoe we een database kunnen benaderen dan gaan we dit 'spel' uitbreiden.

--

# 2.1 Extra - oefeningen

## Extra oefenstof

*De volgende opgaven staan in oplopende moeilijkheidsgraad en dienen als extra oefenmateriaal. Bij de moeilijkere opgave zul je soms zelf op zoek moeten gaan naar antwoorden. Er worden zaken gevraagd die niet allemaal in de les zijn behandeld. net zoals in het echte leven van een Software Developer!*

## Opgave 1

```
<?php $maanden = array ('Januari', 'Februari', 'Maart', 'April', 'Mei', 'Juni', 'Juli', 'Augustus',
                    'September', 'Oktober', 'November', 'December'); ?>
...
...
```

Maak deze code af zodat de output er zo uit ziet en gebruik daarbij een loop.

```
Maand 1 is Januari.
Maand 2 is Februari.
Maand 3 is Maart.
Maand 4 is April.
Maand 5 is Mei.
Maand 6 is Juni.
Maand 7 is Juli.
Maand 8 is Augustus.
Maand 9 is September.
Maand 10 is Oktober.
Maand 11 is November.
Maand 12 is December.
```

## Opgave 2

Maak een drop-down menu met de jaartallen 1940 tot en met het huidige jaar. Het huidige jaar kan worden opgevraagd met de PHP-functie `date('Y')`. We gaan nu niet in op de werking hiervan, maar hiermee kan worden voorkomen dat het formulier ieder jaar moet worden aangepast. Vul `date('Y')`

dus in plaats van 2019 in.

Gebruik een loop.

Pas het drop-down menu aan, zodat de jaartallen in aflopende volgorde worden weergegeven en dat de geboortejaren voor mensen jonger dan 18 worden weggelaten.

# Opgave 3

```
<?php
$eten = array(
    'fruit' => array('geel' => array('banaan', 'citroen'), 'oranje' => array('sinaasappel', 'mandarijn')),
    'groente ' => array('asperge', 'broccoli', 'courgette')
);
...
...
```

Maak het stukje code af en druk daarmee alle groenten onder elkaar af, zoals:

- asperge
- broccoli
- courgette

Gebruik de HTML <UL> om een *unnumbered list* te krijgen zoals hierboven aangegeven.

# Opgave 4

Gebruik het stukje code van opgave 3. Maak een extra stuk code. De code zorgt ervoor dat alle fruit van een bepaalde kleur afgedrukt wordt. Maak daarvoor een *functie* die als parameter het array en de kleur mee krijgt, dus

`drukGeelFruitAf($eten,$kleur);`

# Opgave 5

Maak een functie die twee parameters mee krijgt, $string en $nummer. De output van de functie is de return value is het karakter van de string op de positie n, bijvoorbeeld:

```
myFunct("Hallo",2) -> "a"
myFunct("Hallo",4) -> "l
myFunct("Hallo",9) -> ""
```

# Opgave 6

Maak nu een functie die letter n uit een string haalt. Bijvoorbeeld:

```
myFunct("Hallo",2) -> "Hllo"
myFunct("Hallo",4) -> "Halo
myFunct("Hallo",9) -> "Hallo"
```

# Opgave 7

## logfile

Maak de onderstaande code af. De functie stuurt de regel tekst in $string naar een logfile. De naam van de logfile wordt gedefinieerd in de constante LOGFILE. Gebruikt deze voor de logfile naam.

```
<?

define("LOGFILE", "logfile.txt");

myLog("Oops er ging iets mis in de code op regel 5");

function myLog($string) {
  .....
}
```

Elke keer als je de functie aanroept, wordt er een nieuwe regel aan de logfile toegevoegd. Deze regel begint met de datum en tijd en dan volgt de regel tekst. De log file zou er zo kunnen uitzien:

```
...
17-09-2019 21:35 Oops er ging iets mis in de code
17-09-2019 21:36 Oops er ging weer iets is in de code op regel 12
17-09-2019 21:36 User heeft 12 keer proberen aan te loggen met user id admin, inloggen geblokeerd
...
```

# Opgave 8

## Page teller

Maak een teller die bijhoud hoe vaak een pagina wordt bezocht. Elke keer als een pagina wordt bezocht dan roep jet de functie teller() aan. De functie teller leest een getal uit een file. De file name is gedefineerd in een CONSTANTE (net als LOGFLE bij opgave 7). In de file staat gewoon een nummer, bijvoorbeeld 12. Dit nummer wordt ingelezen, met één opgehoogd en terug geschreven naar de file. In de file staat nu dus het nummer 13.

# Opgave 9

Maak en functie die als parameter een string krijgt. Deze string is een regel text en deze regel wordt omgezet zodat elk woord met een hoofdletter begint. Dus:

```
myFunct("HALLO, Ik ben een goede programmeur!") -> Hallo, Ik Ben Een Goede Programmeur!
```

Let op je mag *geen* gebruik maken van de functie php ucfirst()

# Opgave 10

Maak een functie die een aan de hand van de geboortedatum de leeftijd van een persoon bepaald. Dus input is de geboortedatum en output is de leeftijd. De geboortedatum wordt in drie verschillende variabelen aan de functie gegeven zodat je deze variabelen later makkelijk via een form (apart) kunt opvragen.

Dus:

```
(dit voorbeeld is van 19 september 2019)
myFunct(23,8,2010) -> 9 jaar
myFuntt(23,12,2010) -> 8 jaar
```

# Opgave 11a

Wat doet de php functie scandir("c:")?

Voer de functie uit en maak een lijst van alle directories in je C-Drive, de output ziet er bijvoorbeeld zo uit:

```
Config.Msi
DRIVERS
Documents and Settings
Intel
PerfLogs
Program Files
Program Files (x86)
ProgramData
Recovery
Reflect_Install.log
....
```

# Opgave 11b

De lijst die je bij opgave 6 afdrukt bevat directories en files. Met de fuctie is_dir() kun je bepalen of een item een directory (map) is af dat het bijvoorbeeld een file is. Pas de code aan en Druk het woord Directory af voor elke directory.

Zet de code in een *functie*. De functie krijgt als parameter de directory die moet worden getoond. In het voorbeeld is dat "C:".

# Opgave 11c

Als je een directory anders dan . of .. tegenkomt dat voer je opnieuw de functie die je hebt gecreëerd uit. Dus de functie roept zichzelf aan. Je noemt dit recursief. Als het goed is worden nu alle directories en files afgedrukt. Filter de output nu zodanig dat je alleen alle files ziet. Directories worden dus niet afgedrukt.

# Opgave 11d

## Files zoeken

Je hebt nu een functie die alle files op jouw harde schijf afdrukt.  Maak nu een zoekfunctie die zoekt naar een bepalade file naam. Maak daarbij gebruik van een wild card. mySearch('Te*') zoekt dus alle documenten op die met Te beginnen. Maar de seacht is case insensensitive, dus mySearch('Te*') vindt alle documenten die met te, TE, Te of tE begint.

--

# 3 Functions

*In deze les leren we wat functions zijn waarom je ze gebruikt en hoe je ze maakt. In deze lessen zullen ook onderdelen van de vorige lessen weer terugkomen.*

In de volgende twee lessen wordt uitgelegd wat functions en hoe je ze kan gebruiken.

## Wat is een function?

Wat is een functie en hoe maak je een functie? Waarom zou je een functie willen gebruiken? De antwoorden op deze vragen worden in deze film gegeven.

https://www.youtube.com/embed/XfnH3AEF5Z8

## Nog een voorbeeld van een function

In deze les wordt een iets complexere functie gemaakt. In deze functie worden de even getallen uit een array gefilterd.

https://www.youtube.com/embed/INHM0S_8GHA

## Parameters

Dat wat je meegeeft aan een function heten parameters. In de vorige voorbeelden (filmpjes) zagen we dat we telkens precies één parameter meegaven. In het eerste voorbeeld was dat een getal en in het tweede voorbeeld was dat een array.

Je kunt ook minder of meer parameter meegeven, bijvoorbeeld nul:

```
function welkom() {
    $tekst = 'Welkom op mijn website';
    return $tekst;
}
```

En je kunt ook meer parameters meegeven, bijvoorbeeld drie:

```
function welkom2($tekst, $naam, $hoofdletters = FALSE ) {
    //bepaal tekst
    if ($tekst == 1) {
        $uitvoer = 'Welkom '. $naam;
    }
    elseif ($tekst == 2) {
        $uitvoer = 'Tot ziens, ',$naam;
    }
    //bepaal hoofdletters
    if ($hoofdletters == TRUE) {
        $uitvoer = strtoupper($uitvoer);
    }
    //geef resultaat
    return $uitvoer;
}
```

De derde parameter is wat vreemd want de waarde wordt al bepaald in de functie, althans zo lijkt het. Dit is een zogenaamde *optionele* parameter. Je hoeft hem niet mee te geven en als je hem niet meegeeft dan is die 'by default' false.

Dus

> **welkom2(2, 'Mark', False )**
> is hetzelfde als
> **welkom2(2, 'Mark')**

# Apart bestand

Als je veel functies hebt dan kun je die in een apart bestand zetten dat houd de boel overzichtelijk en maakt samenwerking ook makkelijker, omdat iedereen aan zijn eigen set functions kan werken.

Stel je zet de functie uit het laatste voorbeeld in een apart bestand met de naam functie.php, dan zou de volgende code gewoon werken.

```
<?php
include_once('functie.php');
echo welkom2(1,'Ayoub',true);
?>
```

# Opdracht 1

Maak een php file, functions.php met de functie welkom2 uit één van de voorbeelden van hierboven. Maak een file opdracht1.php en include de functions.php file.

```
$myArray=[''Nouaman','Aart','Samil','Rainee','Diego','Omer','Wessel','Jari','Max','Brian','Kikiya'];
```

Gebruik het bovenstaande array en roep voor elk van de namen de welkom2 functie twee maal aan, één keer om de welkom-boodschap af te drukken en één keer om de Tot-ziens-boodschap af te drukken.

Gebruik een loop en roep vanuit de loop de functie op de juiste manier aan.

Dus de output ziet er zo uit:

```
Welkom Nouaman
Tot ziens Nouaman

Welkom Aart
Tot ziens Aart

Welkom Samil
Tot ziens Samil

....

Welkom Kikiya
Tot ziens Kikiya
```

# Opdracht 2a

Maak een functie die iedereen met een onvoldoende uitprint. Geef het assiociative array mee als parameter van de functie. Noem de functie *resultaten()*

```
$cijferPHP=['Ayoub' => 6, 'Roberto' => 7, 'Seman' => 4, 'Osmani' => 2,
        'Klaas' => 3, 'Sid' => 8, 'Kendal' => 3, 'Ayas' => 9,
        'Kianne' => 8, 'Droppie' => 7, 'Ken' => 6];
```

# Opdracht 2b

Pas de functie uit opdracht 2a aan, zodat deze een assiociative array als parameter krijgt en ook een assiociative array terug geeft waarin alle studenten met een voldoende staan. Het resultaat is dus het volgende array:

```
$cijferPHP=['Ayoub' => 6,'Roberto' => 7, 'Sid' => 8, ,
        'Ayas' => 9, 'Kianne' => 8, 'Droppie' => 7, 'Ken' => 6];
```

# Opdracht 2c

Pas de functie uit opdracht 2b aan, zodat deze een assiociative array terug geeft waarin alle studenten met een voldoende of met een onvoldoende staan. Doe dat op de volgende manier:

```
resultaten($cijferPHP, true); // geef assiociative array met voldoende terug (zoals bij 2a)
resultaten($cijferPHP, false); // geef assiociative array met onvoldoendes terug
```

--

# 3.1 van functie naar object (oop)

*In deze les gaan we leren wat een global variabele is en we gaan een functie maken waarbij we gebruik gaan maken van global variabelen. In sommige gevallen kan dat functioneel zijn. Zeker bij grotere programma's heeft het gebruik van global variabele nadelen en leidt snel tot fouten. Hoe we dit probleem oplossen is met object oriented programmeren en we gaan onze eerste stukje OOP maken.*

## Functies en globale variabelen

In het volgende filmpje wordt een functie gemaakt en wordt een voorbeeld gegeven van het gebruik van *globale vari*abele in een functie. Een *globale* variabele heeft een globale scope en kan *overal* in de code worden gebruikt. Dit in tegenstelling tot een variabele in een functie; die heeft een *lokale* scope en kan alleen in de functie worden gebruikt.

https://www.youtube.com/embed/9CCGc4xZy_0

In de video wordt de onderstaande besproken. We gaan deze code aanpassen.

```
<?php

global $som;
global $aantal;

$som=0;
$aantal=0;

function berekenGemiddelde($cijfer){
  global $som, $aantal;
  $som=$som+$cijfer;
  $aantal++;
  $gemiddelde=$som/$aantal;
  return($gemiddelde);
```

```
}

echo "Gemiddelde: ".berekenGemiddelde(6)."<br>";
echo "Gemiddelde: ".berekenGemiddelde(7)."<br>";
echo "Gemiddelde: ".berekenGemiddelde(8)."<br>";

?>
```

## Opgave 1

De functie *berekenGemmidelde($cijfer)*, moet worden gesplits in twee nieuwe functies.

*cijferToevoegen($cijfer)*, met deze functie voegen we een cijfer toe.

*gemiddeldeBerekenen()*, met deze functie wordt het gemiddelde berekend

Met deze twee nieuwe functies moet het gemiddelde cijfer worden berekend van alle cijfers uit het volgende array.

```
$cijfersPHP=[5,5,6,5,7,6,7,5,8,7,8,6];
```

Gebruik een for-loop en voeg elke cijfer uit het array toe aan de lijst van gemiddelden met de functie *cijferToevoegen($cijfer).*

De functie *cijferToevoegen($cijfer)* wordt dus voor elk cijfer uit het array cijferPHP een keer aangeroepen.

Als alle cijfers zijn toegevoegd, druk je het gemiddelde af met de functie *gemiddeldeBerekenen()*.

Zet de nieuwe code in de file opdracht1.php en lever deze in.

# OOP

Om de basis van OOP uit te leggen ga ik eerst aan de hand van een voorbeeld met een hond een paar nieuwe begrippen uitleggen. Als we die begrippen dan kennen ga ik in het tweede filmpje uitleggen hoe we het voorbeeld van het gemiddelde kunnen omzetten naar een object waarbij we de functies en de variabelen combineren in één object.

### Begrippen

Class, Object, Properties en Methods worden nog een keer duidelijk gemaakt in:

> https://slides.com/jamescandan/oop-php-for-beginners-1-2/fullscreen#/5

https://www.youtube.com/embed/tZJcsFp9ttU

Dus een class is een sjabloon van een object en met het commando new <class_naam> maken we een nieuw object van de sjabloon. In een class heten de variabelen proeprties en de fucnties heten methods.

## Code

Nu gaan we ons gemiddelde in een class zetten.

https://www.youtube.com/embed/z7mB1jRRy_Y

De code die we in het filmpje uitleggen:

```
<?php

class MijnGemiddelde {
  private $som;
  private $aantal;

  public function __construct() {
    $this->reset();
  }

  public function addNumber($number) {
    $this->som=$this->som+$number;
    $this->aantal++;
  }

  public function reset() {
    $this->som=0;
    $this->aantal=0;
  }

  public function gemiddelde() {
    if ($this->aantal) {
      return($this->som/$this->aantal);
    } else {
```

```
      return(0);
    }
  }

}

$gemiddeldePHP = new MijnGemiddelde;
$gemiddeldeJava = new MijnGemiddelde;

$gemiddeldePHP->addNumber(8);
$gemiddeldePHP->addNumber(6);
echo "Gemiddelde PHP: ".$gemiddeldePHP->gemiddelde();

echo "<hr>";

$gemiddeldeJava->addNumber(5);
$gemiddeldeJava->addNumber(7);
echo "Gemiddelde Java: ".$gemiddeldeJava->gemiddelde();

echo "<hr>";

$gemiddeldeJava->reset();
echo "Gemiddelde Java: ".$gemiddeldeJava->gemiddelde();

?>
```

In de volgende opdrachten wordt het voorbeeld van hierboven aangepast en uitgebreid.

Maak alle opdrachten en zet de aangepaste code in opdracht2.php.

# Opdracht 2

Maak een nieuw object $*gemiddeldeLinux*.

Voeg de cijfers 8 en 7 toe aan het object $*gemiddeldeLinux*.

Druk het gemiddelde van Linux af op dezelfde manier zoals in het voorbeeld de andere gemiddelde zijn afgedrukt.

Zet de aangepaste code in opdracht2.php.

# Opdracht 3

Pas de method gemiddelde aan, zodat het gemiddelde cijfer op één decimaal afgerond wordt gereturned.

zet de aangepaste code in opdracht2.php.

# Opdracht 4

Voeg aan de class *MijnGemiddelde* een method toe waarmee je cijfers kunt toevoegen die 2x meetellen.

Dus stel je hebt een 5 en een 8 gehaald, maar de 8 telt twee maal mee. Dan is het gemiddelde 5+8+8=21 en 21/3=7

Maak nu een method en noem die *gewogenCijfer($cijfer, $gewicht).* Je roept de method dan aan op de volgende manier.

```
$gemiddeldePHP = new MijnGemiddelde;
$gemiddeldePHP = $gemiddeldePHP->gewogenCijfer(5,1); // voeg het cijfer 5 toe, deze telt 1x mee
$gemiddeldePHP = $gemiddeldePHP->gewogenCijfer(8,2); // voeg het cijfer 8 toe, deze telt 2x mee


echo "Gemiddelde PHP: ".$gemiddeldePHP->gemiddelde(); // het resultaat zou 7 moeten zijn
```

Pas de class *MijnGemiddelde* aan en maak een de nieuwe method *gewogenCijfer($cijfer, $gewicht)*

Zet de aangepaste code in opdracht2.php.

Lever de file opdracht2.php (waarin de uitwerking van opdracht 2, 3 en 4 zit) in.

--

# 4.1 Calculator+ deel 1 (OOP)

## Intro

We gaan een calculator maken met een object. Hierbij zullen we ook een begin maken met het MVC-model. Dit model is een standaard voor het ontwikkelen van complexere web sites en wordt ook in Laravel gebruikt. Dit is dus een eerste stapje richting Laravel.

In het eerste deel gaan gebruik maken en oefenen met:

1. het maken van een form met een keuze lijst
2. het aanmaken van een class
3. het instantiëren van een object aan de hand van de class
4. het maken van methods (=functies in een class)
5. het aanroepen van methods
6. het opstellen van een switch-case constructie

## Drie files

We gaan drie files maken:

```
form.html
result.php
includes/calc.php
```

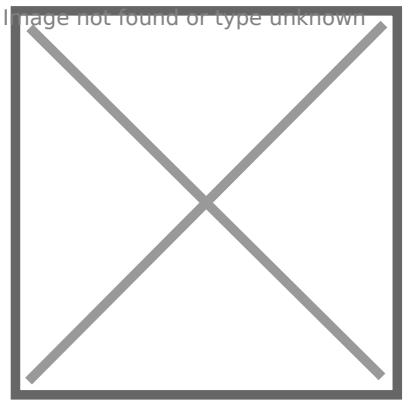
## Form.html

We beginnen met de form.html. Maak een HTML-pagina met een form. Het form heeft drie velden plus een submit knop.

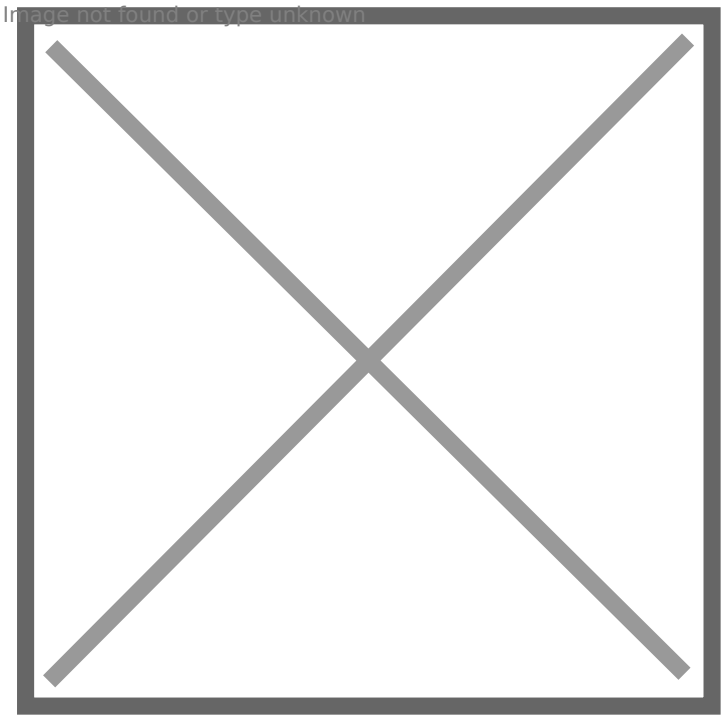
We gaan ook een drop down box maken in het form, lees hier hoe dat werkt:

> https://www.w3schools.com/html/html_form_elements.asp

Het form in eenvoudige vorm ziet er als volgt uit:




We zetten nu het form in een tabel, zodat het er iets fraaier uit ziet. Gebruik hiervoor `cellpadding="10"`

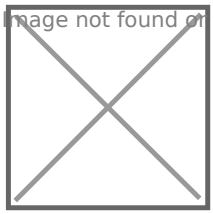



Het form ziet er op deze manier iets fraaier uit, zorg ervoor dat als je de submit knop in drukt dat je dan de pagina result.php laat zien.

# Resultaat.php

Zet in resultaat.php code, zodat je de form- variabellen kun zien, dus je ziet het volgende als (bijvoorbeeld als je op submit drukt):

De output ziet er bijvoorbeeld als volgt uit:




We gaan dit niet netjes maken, we doen dit alleen om te controleren of het form goed werkt en of de variabelen goed worden verstuurd. Gebruik je in het form POST of GET? Zet in het form expliciet welke methode je gebruikt, we kunnen het dan later evetueel nog aanpassen.

Zet in de output ook een link terug naar het form: `<a href=.....>Back to form</a>`

# Includes/class.php

Nu gaan we onze eerste class maken!

We gaan in dit voorbeeld niet op de meest eenvoudigste manier een programma maken (een calculator in php kun je echt veel eenvoudiger maken), maar we gaan dit op deze manier doen om OOP te leren. Het voorbeeld is eenvoudig, maar de manier waarop we het doen is best ingewikkeld.

Ready, voor je eerste PHP OOP-ervaring?

We gaan naar de file calc.php en maken daar de class Calc

```
<?php

class Calc {
    private $number1;
    private $number2;

    public function setNumber1($number){
        $this->number1 = $number;
    }

    // maak hier een tweede public method die number2 initialiseerd

}
?>
```

# Public function $setNumber2

Maak zelf de tweede public function op de plaats waar nu het commentaar staat. We maken $number1 en $number2 'private', hetgeen betekent dat ze van buiten de class niet beschikbaar zijn. Dit zorgt ervoor dat je als gebruiker van deze class niet precies hoeft te weten hoe alle variabelen heten (dit is een vorm van *encapsulation*).

In de file resultaat.php gaan we nu de class instantiëren; dit betekent dat we een object gaan aanmaken aan de hand van de blauwdruk van de class die we zojuist hebben gemaakt.

We doen dit met: `$myCalcObject = new Calc;`

# Include

Om new Calc te kunnen aanroepen moet de de file waarin de class is gedefinieerd wel worden included:

```
include "includes/calc.php";
```

'Include' plaatst als het ware inhoud van de file `calc.php` in `resultaat.php`

In plaats van `include` te gebruiken zou je ook alle code in éé'n grote  file kunnen plaatsen. Dit doen we **niet** omdat je dan snel het overzicht verliest. Door de code op te delen en op een logische plaats in de juiste folder te plaatsen houd je het overzicht. In Laravel zul je zien dat je een zeer uitgebreide folder structuur hebt. Alles staat dan op een logische plaats. Onderdeel van het leren van Laravel is weten waar wat staat en hoe de verschillende files met elkaar verband houden. Maar daarover later meer.

# Methods aanroepen vanuit resultaat.php

We kunnen nu de methods (functions) vanuit resultaat.php aanroepen die we in de class hebben aangemaakt:

```
$myCalcObject->setNumber1(...);
```

Op de plaats van de puntjes zet je de variabele die uit het form komt en die we zojuist nog hebben afgedrukt.

Op dezelfde manier geven we de object property `$number2` ook een waarde. We maken een tweede method (functie) waarmee we de property (variabele) $number2 de juiste waarde geven: `$myCalcObject->setNumber2(...);`

Het object 'weet' nu welke twee getallen we hebben verstuurd met het formulier. Deze twee getallen staan in de class properties $number1 en $number2

# Nieuwe method: executeCalculation

We gaan nu een method maken die het resultaat berekent `echo "Resultaat is: ".$myCalcObject->executeCalculation();`

Laten we weer kleine stapjes maken en eerst doen alsof we alleen kunnen optellen.

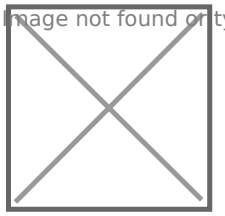
We maken dan een method om de twee getallen op te tellen en return-en het resultaat.

```
    public function executeCalculation() {
        // return op de plaats van de puntjes ($number1 pus $number2).
□□//let op dat je elementen van dit object met $this-> aanspreekt.
□□return ..... ;
    }
```

In de code van resultaat.php, voegen we dan een regel toe:

```
echo "Result is: ".$myCalcObject->executeCalculation();
```

Als het goed is dan zie je dit als je het form post:




# Stap 2, Switch control stucture

Nu gaan we de *executeCalculaiton* method zodanig aanpassen dat de andere bewerkingen, substitude (aftrekken), multiply (verenigvuldigen), divide (delen) en addition (optellen) ook worden ondersteund. We passen hiervoor eerst de method *executeCalculation* aan, zodat de functie (add, subst, mult of div) wordt meegegeven. We kunnen dit met if-then-else doen, maar het is netter om een switch-case constructie te gebruiken. Deze kun je ook iets eenvoudiger uitbreiden (wat we natuurlijk nog gaan doen).

Lees de documentatie: https://www.php.net/manual/en/control-structures.switch.php

Bekijk ook de voorbeelden bij deze documentatie. In example #2 wordt een switch gemaakt aan de hand van een string ($i), de cases zijn 'apple', 'bar' of 'cake'. Wij maken onze switch aan de hand van de variabelen die we meegeven bij het aanroepen van de method (functie) executeCalculation en onze cases zijn: 'add', 'subst', 'mult' of 'div'.

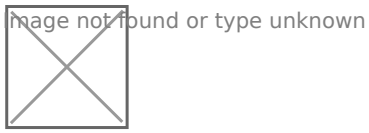
Maak de method *executeCalcultion* af:

1. zorg ervoor dat de functie ( 'add', 'subst', 'mult' of 'div') wordt meegegeven als parameter.
2. Maak een switch aan de hand van deze parameter en zet bereken bij elke case het juiste resultaat.
3. return het resultaat.

Klaar!

Je hebt nu de eerste milstone bereikt, gefeliciteerd!

In schema ziet de flow van deze mini applicatie er als voglt uit:




**Laat het resultaat zien aan je docent voordat je verder gaat met deel 2.**

# 4.2 Calculator+ deel 2 (cookies)

1. In deze les leren we:
2. onze class gaan we uitbreiden
3. we leren wat stateless (en statefull) is
4. we leren met cookies te werken: creëren, uitlezen en debuggen
5. we gaan een string omzetten met explode zoals we dat eerder hebben gedaan in de oefeningen.
6. we gaan alle elementen van een array afdrukken in een loop
7. we gaan het gemiddelde berekenen van een array

We hebben een calculator die bestaat uit (1) een form, (2) een resultaten pagina en (3) een class.

Stel dat we nu een serie getallen willen invoeren en dat we het gemiddelde willen uitrekenen. Hoe kunnen we dit dan doen?

We kunnen ons object gebuiken en een array gebruiken waar we elke keer een element aan toe voegen. Dit hebben we geoefend in de opgaven die we eerder hebben gemaakt. Weet je nog dat we de functie array_push gebruikte?

We zouden in de lijst van het form een functie kunnen maken die we 'avg' noemen. Hiermee kunnen we dan het gemiddelde berekenen. We willen dit als volgt maken:

Elke keer als we een getal invoeren en de functie `avg` kiezen dan wordt het getal aan een array toegevoegd en wordt het gemiddelde van alle getallen berekend.

Dus

- je voert 5 in met de nieuwe functie avg en drukt op submit,
- je ziet als resultaat 5 en het gemiddelde is 5
- je voert nu 15,
- je ziet als resultaat 5,15 en het gemiddelde 10

- je voert nu 1 in en je ziet als reultaat 5,15,1 en het gemiddelde 7

Een webserver is stateless:

> *Stateless is the polar opposite of stateful, in which any given response from the server is **independent of any sort of state**.*

Dit betekent dat de webserver geen dingen onthoud, althans niet vanzelf. Elke keer als je de webserver iets vraagt weet hij niets van de vorige keer dat je er was. En toch weten webserver zoals Facebook precies wie jij bent en wat jij allemaal hebt gedaan. Hoe kan dat?

Dat komt omdat Facebook (en andere sites) deze gegevens opslaat en koppelt aan jouw als gebruiker. je logt aan en Facebook weet dan wie je bent. Facebook zoekt dan alle gegevens die bij jou horen op en gebruikt die.

> Vraag: hoe weet een site als Facebook bij elke keer alsje wat vraagt wie jij bent; je stuurt immers niet elke keer je userID en password mee?

Dus - terug naar ons gemiddelde- als we willen dat er een array wordt opgebouwd dan moeten we de status van het array bijhouden. Dit kan met een database maar dan moeten we ook een log-in maken. In een database moeten de gegevens immers aan jouw worden gekoppeld. Waarom? Nou stel dat er gelijktijdig met jou een andere gebruiker ook een gemiddelde wil berekenen, en je zou de gegevens niet per gebruiker opslaan, dan zouden jullie elkaar gegevens gebruiken.

Als dit verhaal niet duidelijk is, vraag dan om uitleg. Dit is een belangrijk onderwerp om te begrijpen en om inzicht te krijgen hoe een webserver werkt.

Er zijn andere methodes om (beperkt) gegevens vast te houden. Dit zijn sessie-varieblen en cookies.

Cookies worden op de client (browser) opgeslagen en sessie-variabelen worden op de server opgeslagen. Voor onze toepassing gaan we cookies gebruiken. We gebruiken de cookie als een soort mini-database-je.

We gaan terug naar onze class Cals en voegen aan de switch-case constructie een case toe. Noem die 'avg' (average/gemiddelde). Als deze functie wordt aangeroepen dan maken we een cookie waarin we de waarde van $number1 zetten.

```
switch($function) {
case('add'):
...
case('avg'):
    setcookie("myCalc", $this->number1, time()+60);
    break;
```

```
...
}
```

Hiermee maken we dus een cookie met de naam myCalc, geven het de waarde $this->number1 en we laten dit cookie verlopen na 1 minuut.

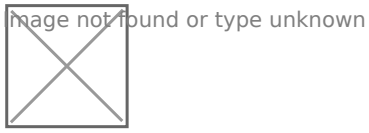
Voeg de optie avg (average) ook to aan het form.

`<option value="avg">average</option>`

## Testen van cookie

Ga nu naar het form, vul een getal in en kies de optie avg. Post het form en als het goed is zou het cookie moeten worden gezet.

Zoek op internet op hoe je met joiw browser je cookies kunt zien. Bij Firefox moet je onder het developer menu kiezen voor 'Storage Inspector'. Je krijgt dan het volgende te zien:




Je ziet (in geel gearceerd) dat het cookie is aangemaakt en dat het de waarde 12 heeft.

We gaan nu niet alleen de waarde opslaan, maar ook de vorige waarde bewaren. Opgelet!

In de class definitie bij de case("avg") wordt nu het cookie gezet, maar voordat we dat doen gaan we eerst kijken of er al een cookie eerder is gezet met `$_COOKIE["myCalc"]` kunnen we de bestaande cookie opvragen, maar we weten niet zeker of die al is gezet. De allereerste keer is het cookie immers nog niet gezet. We moeten dus testen of de variabele `$_COOKIE["myCalc"]` bestaat. Dat doen we met de functie `isset()`.

Dus met isset testen we of het cookie al bestaat. Als dat zo is dan plakken we de nieuwe waarde aan de bestaande waarde.

`$value=$_COOKIE["myCalc"].'-'. $this->number1;`

We gebruiken '-' (min-teken) als koppel teken om de verschillende waarden te scheiden. Dit doen we omdat dit makkelijk leesbaar is. Let op dat de nieuwe cookie waarde nu in `$value` staat en nog niet in het cookie.

Als met de isset functie wordt bepaald dat er nog geen cookie is gezet dan stellen we deze in met:

`$value= $this->number1;`

Ook hier geldt dat de nieuwe cookie waarde nu in de variabele $value staat en nog niet in het cookie.

We moeten dus het cookie nog vullen met de waarde `$value`.

```
setcookie("myCalc", $value, time()+60);
```

Check of het geheel nu werkt.


In dit voorbeeld heeft de cookie de waarden 1,2 3 en 5.

We zijn er bijna!

Het enige dat we nog moeten doen is de waarden afdrukken en het gemiddelde berekenen.

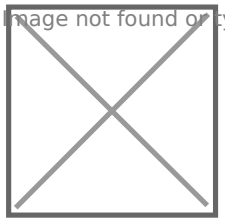
Laten we de `case("avg")` de waarde van de cookie als return waarde terug geven aan de `results.php`.

In de result pagina moeten we nu onderscheid maken tussen het geval als de avg functie gekozen is en als er een andere functie wordt gekozen.

```
<?php
...
if ( $_GET['function'] == 'avg' ){
    $result = $myCalcObject->executeCalculation($_GET['function']);
  // zet de variabele $result om in een $myArray, gebruik de explode functie
// maak een loop en druk alle elementen van het $myArray af.
    // druk het gemiddelde van het array af; gebruik array_sum en count
}else{
    echo "Result is: ".$myCalcObject->executeCalculation($_GET['function']);
}
...
?>
```

Maak de code af, lees het commentaar en vervang deze door code.

De output moet er ongeveer zo uit zien (nadat je 2 maal de functie avg hebt aangeroepen).




Je kunt nu de output nog een beetje mooier maken, gebruik een table en gebruik wat CSS.

De opdracht is klaar.

# 5.0 PDO en Sessies

In deze les wordt een demo gegeven van hoe je met PDO en sessie variabelen een login kunt maken.
Je leert gebruik maken van PDO en van PHP sessies.

De wachtwoorden worden in deze les niet-encypted opgeslagen. Op deze manier kan je goed zien wat er gebeurt.

## Sessions

Een sessie is wordt gestart met het php commando `session_start()`. Hiermee krijgt jouw browser sessie een stukje geheugen op de web server waar sessie variabelen worden opgeslagen.

Elke keer als je `session_start()` aanroept wordt er eerst gekeken of er al een sessie bestaat, als die bestaat dan worden alle sessie variabelen door de server naar de browser gestuurd. Bestaat de sessie nog niet, dan wordt er een sessie opgestart en wordt er een stukje geheugen op de server voor jou gereserveerd.

Binnen een sessie kan je met `$_SESSION['naam']='xyz'` de sessie variabele naam de waarde 'xyz' geven.

Met `session_destroy()` kun je de sessie beëindigen. Alles sessie variabele worden dan weg gegooid.

Als je een keer bent aangelogd dan zet je in een sessie variabele een indicator dat je bent aangelogd. Je kunt bijvoorbeeld jouw userid in een sessie variabele zetten. Dit userid wordt dan elke keer door de server nar jouw browser gestuurd,

## Login

Je maakt dus een form waarin je een gebruikersnaam en wachtwoord aan de gebruiker opvraagt. Met PDO voer je een query uit en controleer je of de combinatie gebruikersnaam en wachtwoord klopt. Als dat zo is dan start je een sessie en zet je (bijvoorbeeld) het userid in een sessie variabele. Je hoeft dan geen wachtwoord te onthouden en telkens opnieuw aanloggen. De sessie variabele is een betrouwbare indicator dat je bent aangelogd.

## Database

Maak een database test en draai het volgende SQL script:

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL,
  `naam` varchar(100) NOT NULL,
  `wachtwoord` varchar(100) NOT NULL,
  `rol` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `users` (`id`, `naam`, `wachtwoord`, `rol`) VALUES
(1, 'max', 'max', 1),
(2, 'piet', 'piet', 1);

ALTER TABLE `users`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `users`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
COMMIT;
```

De database heeft de tabel users waarin het id, naam, wachtwoord en rol (id) van de gebruiker staan.

# DB

De verbinding van de database wordt in **db.php** gedefinieerd.

```
<?php

$dsn = "mysql:host=localhost;dbname=test";
$user = "root";
$passwd = "";

try {
  $pdo = new PDO($dsn, $user, $passwd);
} catch (Exception $e) {
  echo 'Caught exception: ',  $e->getMessage(), "\n";
}



?>
```

# Landing page

De **index.html** is alleen maar een landing page die een zeer eenvoudig menu laat zien.

```
<html>
<body>

<a href='index.php'>Landing page</a> -
<a href="login.php">Login</a> -
<a href="logout.php">Log out</a> -
<a href='page.php'>Just another page</a>
<hr>

Landing page
```

# Login page

Deze pagina laat een laat een form zien waarmee je kunt inloggen. Als je het form post dan wordt er gecontroleerd aan de hand van de database of de opgegeven gebruikersnaam bestaat en of het wachtwoord klopt.

Als je bent aangelogd dan wordt er een sessie variabele gezet. Aan de hand van deze sessie variabele kan je later zien of je bent aangelogd. Zolang de sessie blijft bestaan zal de server elke keer de sessie variabelen meesturen naar jouw browser.

Maak de file **login.php**

```
<!DOCTYPE html>
<html>
<body>

  <a href='index.php'>Landing page</a> -
  <a href="login.php">Login</a> -
  <a href="logout.php">Log out</a> -
  <a href='page.php'>Just another page</a>
<hr>

<?php

if ( isset($_GET['naam']) &&  isset($_GET['wachtwoord']) ) {
```

```
  $naam = $_GET['naam'];
  $ww=$_GET['wachtwoord'];

  include "db.php";
  echo "Trying to login with ".$_GET['naam'];

  $stmt = $pdo->prepare("SELECT * FROM users where naam = :naam and wachtwoord=:wachtwoord");
  $stmt->bindParam(':naam', $naam, PDO::PARAM_STR);
  $stmt->bindParam(':wachtwoord', $ww, PDO::PARAM_STR);
  $stmt->execute();
  $result = $stmt->fetch();

  if ($result ) {

    echo "<pre>";
    var_dump($result);
    echo "</pre>";

    session_start([ 'cookie_lifetime' => 86400,]);
    $_SESSION['id'] = $result['id'];
    $_SESSION['naam'] = $result['naam'];

    echo "<br>Succesfully loggin in!";
    echo "<br><a href='page.php'>Just another page</a>";
    exit;
  }
}

?>

<h2>Login</h2>

<form action="login.php">
  <label for="naam">Naam:</label><br>
  <input type="text" id="naam" name="naam" value=""><br>
  <label for="wachtwoord">Wachtwoord:</label><br>
  <input type="text" id="wachtwoord" name="wachtwoord" value=""><br><br>
  <input type="submit" value="Submit">
</form>
```

```
</body>
</html>
```

# Logout

Als je de sessie beëindigt met sessie_destroy() dan is de sessie beëindigd en weet je browser niet meer of je bent aangelogd.

Maak een **logout.php**

```
<!DOCTYPE html>
<html>
<body>

  <a href='index.php'>Landing page</a> -
  <a href="login.php">Login</a> -
  <a href="logout.php">Log out</a> -
  <a href='page.php'>Just another page</a>
<hr>

<?php
session_start();
session_destroy();
?>
Uitgelogd
```

# Page

Tenslotte maken we nog een pagina die gewoon laat zien of we aangelogd zijn of niet.

Maak de pagina **page.php**

```
<!DOCTYPE html>
<html>
<body>

  <a href='index.php'>Landing page</a> -
  <a href="login.php">Login</a> -
  <a href="logout.php">Log out</a> -
```

```
  <a href='page.php'>Just another page</a>
  <hr>

<?php

session_start();

if (isset($_SESSION['id'])) {
  echo  "<br>You are logged in and your user id is    ". $_SESSION['id'];
  echo  "<br>and your naam id is ". $_SESSION['naam'];
} else {
    echo  "<br>You are NOT logged in";
}

?>
```

# 5.1 PDO class - MVC

In de vorige lessen hebben we een calculator gemaakt en hebben hier een cookie als mini-database-je gebruikt. We gaan nu een 'echte' database verbinding maken. Als het goed is hebben we al eerder met PDO gewerkt. We gaan even kort herhalen hoe het ook alweer zat met PDO en dan gaan we een eigen class maken met PDO. Met die class kunnen we dan eenvoudig gegevens opslaan en opvragen.

We gaan dus:

1. een stukje PDO herhalen
2. en een PDO-database class maken

## PDO 'old style'

PDO wordt gebruikt om een verbinding met een database te maken en om via deze verbinding gegevens op te slaan, te wijzigen, te verwijderen of op te vragen (insert, update, delete, select).

Vorig jaar heb je PDO gebruikt op de (ongeveer) de volgende manier:

```
 <?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Gebruik deze code om te controleren of PDO goed werkt en of je de username en password van je database nog weet.

> Gebruik je root als username? Waarom is dit onverstandig?

In PDO heb je een aantal stappen die je moet doorlopen om een query uit te voeren:

1. De verbinding maken; dit hoeft maar één keer.
2. De query klaarzetten (prepare).
3. De query uitvoeren (execute).
4. De resultaten uitlezen (fetch).

Sommigen queries hebben geen resultaten, dan is stap 4 niet nodig.

> Vraag: welk soort queries hebben geen resultaat zoals in stap 4 wordt bedoeld?

## PDO Object-style

We gaan nu een PDO-object maken waarin we deze stappen proberen samen te voegen.

Omdat het maken van de verbinding maar één keer nodig is, zetten we dit in de __construct van de class.

> Vraag: weet je nog wat de __construct method (functie) doet?

In

https://www.youtube.com/watch?v=7MVOpcKVr_g

kun je nog een keer bekijken hoe de __construct ook alweer werkt.

De PDO class ziet er uit zoals hieronder. Bekijk de code en zorg dat je alles begrijpt. Vul de code aan op de plaats waar ... staan.

```
<?php

// Define DB Params
define("DB_HOST", "localhost");
define("DB_USER", "");
define("DB_PASS", "");
define("DB_NAME", "");

class DB{
```

```
    protected $dbh;       // dit is de verbinding met de databas
    protected $stmt;      // dit is het huidige statement
    protected $resultSet; // hierin komen de resutlaten te staan van een query

    public function __construct(){
        $this->dbh = new PDO(....); // zoek op hoe je de verbinding maakt en vul de juiste code op de plaats van de puntjes
        $this->resultSet=[];
    }

    public function execute($query){
        $this->stmt = $this->dbh->prepare($query);
        // haal nu het resultaat binnen in $result door de query nu uit te voeren (je moet dus de method execute uitvoeren op $this->stmt)
        .....
        if (! $result) {
            die('<pre>Oops, Error execute query '.$query.'</pre><br><pre>'.'Result code: '.$result.'</pre>');
        }
        $this->resultSet = $this->stmt->fetchAll(PDO::FETCH_ASSOC);
        // return nu het het aantal rows dat de query terug geeft. Dit is dus het aantal array elementen dat je terug krijgt.
        .....
    }

    public function getRow(){
        // lees het array waarin de resultaten staan ($this->resultSet) uit, regel voor regel.
        // Elke keer als deze functie wordt aangeroepen return je de volgende regel uit de resultSet.
        // Gebruik de array_shift functie en zoek op hoe je deze moet gebruiken.
        // Let op wat er gebeurt als je een lege resultSet hebt.
        .....
        .....
        .....
        .....
        .....
    }
}
?>
```

We gaan onze PDO class testen maar daarvoor hebben we wat testdata nodig. We gaan dus testdata maken.

# Aanmaken testdata

Maak nu een test-database en een testtabel.

Creëer een database 'flights' en maak een tabel met testdata met het volgende script via PHPMyAdmin

```
CREATE TABLE `airlines` (
  `code` char(2) NOT NULL,
  `airline` varchar(40) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

INSERT INTO `airlines` (`code`, `airline`) VALUES
('AA', 'American Airlines Inc.'),
('AS', 'Alaska Airlines Inc.'),
('B6', 'JetBlue Airways'),
('DL', 'Delta Air Lines Inc.'),
('EV', 'Atlantic Southeast Airlines'),
('F9', 'Frontier Airlines Inc.'),
('HA', 'Hawaiian Airlines Inc.'),
('IA', 'AIRLINE'),
('MQ', 'American Eagle Airlines Inc.'),
('NK', 'Spirit Air Lines'),
('OO', 'Skywest Airlines Inc.'),
('UA', 'United Air Lines Inc.'),
('US', 'US Airways Inc.'),
('VX', 'Virgin America'),
('WN', 'Southwest Airlines Co.');
```

# MVC, stap 1

Maak nu een nieuw project met de volgende files en folder structuur:

```
/includes/db.php
crud.php
view.html
```

De db.php bevat de class DB zoals je die hierboven heb gemaakt. Met de crud.php file gaan we een CRUD-pagina maken en met de view.html gaan de data aan de gebruiker laten zien.

De /includes/db.php regelt dus de connectie naar de database, de crud.php is de pagina waar de 'controle' plaats vind; wat gebeurt er met de data. Ten slotte wordt er in de view.html pagina de

data getoond. We hebben hier dus te maken met vier functies Model (de connectie naar de database), View (de html), en Control (de PHP-code die ales 'regelt'). We hebben het hier over het MVC-model.

In de volgende les gaan we onze CRUD-pagina maken in deze MVC-structuur.

Handige resource: https://www.webcarpenter.com/blog/52-PDO-Tutorial-for-MySQL-Developers

---

# 5.1 CRUD - Read

Wat moet goed je weten voor deze les:

- Hoe werken arrays.
- Hoe werken assiociative arrays.
- Hoe werken tables.

Wat ga je leren deze les?

- Je gaat werken met PHP objecten
- Je gaat leren werken met methods en properties van een object
- Je gaat inheritance toepassen

## CRUD

CRUD is een groep functionaliteiten die je voor je examen moet kunnen

CRUD staat voor Create, Read, Update en Delete. We gaan een pagina maken waar je deze functies kunt uitvoeren. We gaan dit testen op de airlines table in de database flights. Dat betekent dat webeginnen met het maken van een overzicht van alle flights (R), Dit is de R(ead) van de CRUD.

Voordat we beginnen moeten we controleren of we alles uit de vorige lessen goed hebben.

Controleer of je de volgende structuur hebt:

```
/includes/db.php
crud.php
view.html
```

In /incudes/db.php heb je een db class gemaakt. Deze zorgt ervoor dat er een verbinding wordt gemaakt met de SQL database.

crud.php en view.html zijn nu nog leeg.

Je hebt een tabel `airlines` gemaakt in een nieuwe databse `flights`.

Remember: *een object is een variabele in een object!*

In de `crud.php` file maken we een class airlines. In de __construct() maken we een DB object. Het airlines object heeft nu property die verwijst naar een DB object.

> Vraag 1: Hoe heet de property (variabele) in crud.php die het object DB bevat?

```
<?php

include_once "includes/db.php";

class airlines .... {
    public function getAllAirlines() {
        return $this->executeSQL("select * from airlines");
    }
}
?>
```

Bekijk bovensstaande code en let vooral op de regel waar de class airlines wordt gemaakt. Zie je de puntjes? Daar moet nog wat komen te staan. De class airlines maken we namelijk een 'kind' van de class 'DB'. Hierbij erft het kind, alle eigenschappen van de moeder.

Weet je nog hoe dat zat met classes en ínherentence'?

Als je het nog eens wilt bekijken dan kun je het volgende video filmpje bekijken:

Inheritance in PHP eplained

Vul nu de juiste code in op de plaats van de puntjes zodat de airlines class alle eigenschappen van de DB class erft.

> Vraag 2: welke statement gebruik je in PHP om een child class te maken?

Remember: *een method is een functie in een object!*

Nu gaan we onze code testen en dat doen we door vanuit view de code aan te roepen.

Zet dan de volgende code in view.php

```
<?php
include "crud.php";

$airlines = new Airlines;
echo $airlines->getAllAirlines();
```

```
?>
```

Bekijk wat er gebeurt. Welk getal wordt er afgedrukt en waarom?

## Rows

In de class DB staat een method getRow. Roep deze aan vanuit de view.php.

`$row=$airlines->getRow();`

Druk de regel af,wat zie je? Waarom wertk dat niet? Gebruik de PHP functie `var_dump($row)` om de regel af te drukken. Zie je nu wat er wordt afgedrukt?

We gaan nu in driestappen alle regels netjes afdrukken.

Stap 1, we gaan in een loop alle regels afdruken met var_dump

Dat ziet er ongeveel zo uit:

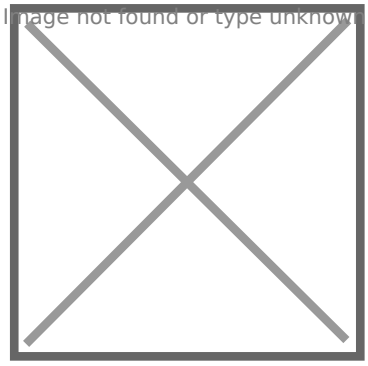
```
array(2) { ["code"]=> string(2) "AA" ["airline"]=> string(22) "American Airlines Inc." }
array(2) { ["code"]=> string(2) "AS" ["airline"]=> string(20) "Alaska Airlines Inc." }
array(2) { ["code"]=> string(2) "B6" ["airline"]=> string(15) "JetBlue Airways" }
array(2) { ["code"]=> string(2) "DL" ["airline"]=> string(20) "Delta Air Lines Inc." }
....
....
```

Om de regels één voor één op te halen moet je net zo lang de method getRow() aanroepen todat je ..... totdat je wat? Kijk in de code voor het antwoord.

Stap 2, druk de regels af als twee losse variabele

```
AA American Airlines Inc.
AS Alaska Airlines Inc.
B6 JetBlue Airways
DL Delta Air Lines Inc.
....
....
```

Stap  3, zet nu alles netjes in een table:




Eventueel kun je de table met CSS nog wat mooier maken. Je kunt nog een nette kopregel toevoegen.

Op zich zijn het opfraien van de GUI geen keiharde voor de back-end programmeur, maar als je met een paar eenvoudige stappen de output wat kunt 'opleuken' dat toont dat toch beter.

De R van CRUD is nu klaar.

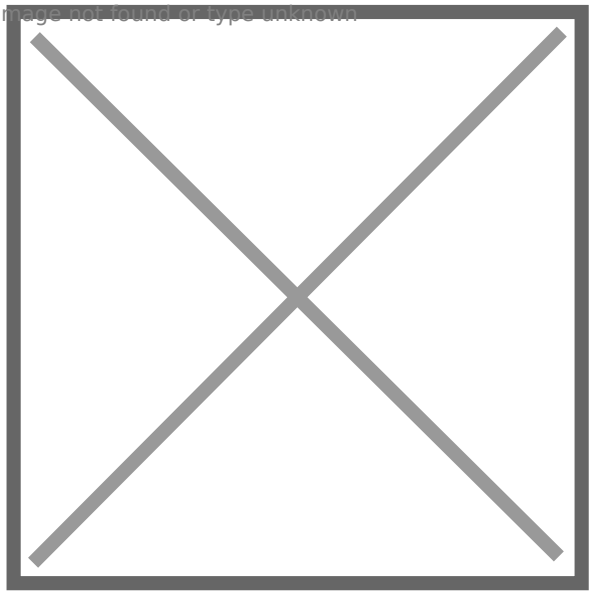
---

# 5.2 CRUD - Create

We hebben de R in de vorige les gemaakt. We gaan nu de C van create maken. Daarmee kunnen we een Airline toevoegen.

We maken hiervoor een file insert.html en in deze file maken we een form waarin de gebruiker wordt gevraagd om de gegevens van de nieuwe ailine in te typen. Een nieuwe airline bestaan slechts uit twee velden: de airline code en de airline name. Het form ziet er dus ongeveer als volgt uit:

Image not found or type unknown



Een eenvoudig form met twee velden en een button. De button post de velden naar de file insert.php. Deze insert.php voert dan de juiste query uit gebrukt daarvoor het object DB.

We hebben dus een rijtje van drie files die elkaar aanroepen: het form vraagt de waarde aan de gebruiker, de php file ontvangt de waarden uit het form en stuurt deze in de juiste vorm naar insert.php. Insert.php roeptde crud.php aan waar de juiste query wordt uitgevoerd.

| File | Functie | MVC |
|---|---|---|
| insert.html | Formulier, GUI naar gebruiker | View (=html, GUI) |
| insert.php | De code waar wordt bepaald wat er wordt gedaan. Voor een insert is dat echter vrij eenvoudig. | Control (=code) |
| crud.php | De vertaling naar een query naar de database. CRUD maakt zelf weer gebruik van DB om een verbinidng naar de DB te maken. Dit is allemaal nodig om met de Database te communiceren. | Model (=Database) |

Waarom deze 'moeilijke' scheiding in allemaal verschillende file, zul je misschien afvragen? Dit zullen we in de les verder uitleggen, maar het komt er op neer dat je alle functies zoveel mogelijk wilt scheiden. Dit komt allemaal ten goede aan code die goed onderhoudbaar is. Stel je maar eens voor dat je over en nergens een SQL query kun tegenkomen en de datanase tabelnamen veranderen, dan weet je niet waar je dit allemaal moet aanpassen.

OK we gaan de code aanpassen, we beginnen met crud.php.

```
<?php
...
    public function insertNewAirline($code, $name) {
        return $this->executeSQL("insert into airlines (code, airline) values ('$code', '$name')");
    }
...
```

We voegen een nieuwe method to die een rij in de database toevoegd. De rows in de tabel in dit voorbeeld heten code en airline en de tabel heet airlines. Dat kan in jou database anders heten. De method verwacht twee parameters, $code en $name. Maar een file insert.php en zorg deze method wordt aangeroepen en dan dat de geposte variabelen uit het form als parameters worden meegegeven.

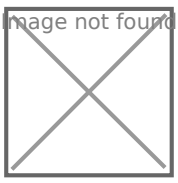
Let op dat als je een method aanroept dat je dan wel het jusite object eerst moet instantiëren. Kijk daarvoor nog eens terug hoe je dat hebt gedaan in view.php

Als de insert is gelukt vanuit insert.php dan ga je vanuit insert.php terug naar view.hp. Je kunt daarvoor de PHP functie header("") gebruiken. Deze stuurt een header naar je browser en daarin kun je dan aangeven welke pagina moet worden geladen. Zoek zelf op hoe dit werkt.

Als het goed werkt dan zul je zien (via de view.php) dat je een regel heb toegevoegd.

Je kunt nu ergens in je view.php een <a href....> plaatsen zodat je vanuit deze pagina naar de insert kunt springen.

Image not found or type unknown



Wat gebeuert er nu als je een Airline code voor een tweede keer probeert in te voeren?

Je krijgt een foutmelding, waar komt die vandaan?

# Opdracht/Huiswerk

1. Zorg ervoor dat de input wordt gecontroleerd. Een airlinie code moet altijd uit twee hoofdletter bestaan en de Airline naam mag niet leeg zijn.
2. Pas je code nu zodanig aan dat je een nette foutmelding krijgt als je een Airline code voor een tweede keer probeert toe te voegen.
   Je zult hiervoor de control (code) moeten aanpassen. Welke file is dat? Bedenk eerst hoe je dit wilt gaan aanpakken en overleg met de docent.

**Stuur de file insert.php op via teams als controle voor het uitvoeren van het huiswerk. Dit wordt afgetekent.**

--

# 5.3 CRUD - Update

We hebben nu een Create en een Read. We gaan nu de update maken.

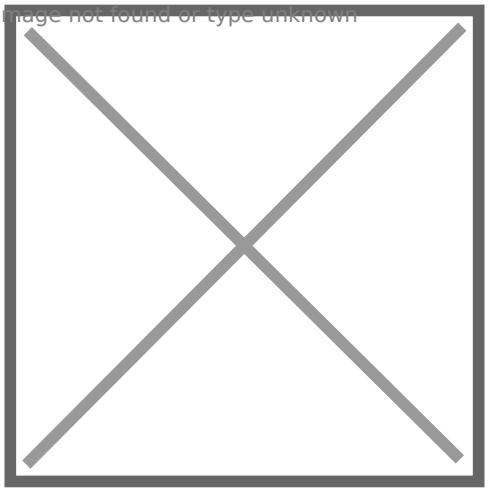
We maken een linkje op de view pagina waarmee je de betreffende regel kan gaan updaten. In dit geval wordt de link in een aparte kolom geplaatst. De link wordt weegegeven door een variabele die we later gaan invullen voor nu kunnen we de variabele boven in de loop even leeg initialiseren.

```
<a href=\"".$editLink."\" role=\"button\">Edit</a>
```

> Vraag: zie je de \" in de string hierboven, wat betekend dat?

De view pagina ziet er nu ongeveer zo uit:

Image not found or type unknown



Wat moet er nu in de edit link komen? Twee dingen de control die moet worden uigevoerd en op een of andere manier een verwijzing naar de huidige regel De control moet immers weten welke regel je wilt gaan editen!

Noem je edit control control.php. De link moet nu wijzen naar deze control, maar hij zal oo moeten meegeven welke regel je wilt veranderen. Wat is de key van de regel? Geeft deze key mee volgens de GET methode. Weet je nog hoe dat werkt? Indien niet lees het dan nog eens na op:

https://www.tutorialspoint.com/php/php_get_post.htm

In de while-loop van de view.php weet je telkens wat de key is want die druk je immers ook af. Zorg er dus voor dat de link 'Edit' die je net hebt gemaakt een link krijgt naar de control.php file met als parameter de key.

Maak nu de edit.php en druk de parameter, de key af. **Ga pas door als je dit voor elkaar hebt.**

We moeten nu een form maken waarin we het record (de regel / row) laten zien en kunnen veranderen. We willen niet dat de key wordt veranderd. Die wordt wel getoond op de edit pagina, maar alleen de naam van de airline kan worden gewijzigd.

We maken dus een form en de twee waarden, airline code en airline naam dienen ingevuld te zijn.

Omdat er bijna geen code in de edit.php zal staan, kunnen we deze ook edit.html noemen. Het is immers vrijwel volledig een view pagina waar vrijwel geen code in staat.

In de edit.html staat een submit knop (net zoals in de pagina waar je een regel toevoegd) en die post de nieuwe naam waarde naar de control edit.php. In de edit.php wordt de juiste query uitgevoerd. In dit geval een update query:

Dit werkt bijna hetzelfde als bij de insert, alleen de query is net iets anders. Werk dit verder zelf uit.

# Site structuur

We hebben nu de files zoals aangegeven in de de onderstaande tabel. Deze tablel geeft overzicht van hoe de website in elkaar zit.

1. Maak de tabel verder af.

| File | Wat | Type (MVC) | Aangeroepen door | Roept aan |
|---|---|---|---|---|
| view.php | Laat lijst zien met alle airlines | View/Control | Landing page | Edit, Insert |
| insert.html | Nieuwe airline | View | view.php | |
| insert.php | | | | |
| edit.html | | | | |
| edit.php | | | | |
| crud.php | | | | |
| db.php | | | | |

# 5.4 CRUD - Delete

We zijn nu bijna klaar. Alleen de delete nog.Voor de delete wordt alleen op hoofdlijnenn aangegeven wat je moet doen. Dit omdat de delete vrij eenvoudig is en alle technieken die we nodig hebben hebben we al gebruikt in voorgaande lessen.

1. We maken een extra kolom in de view.html waarin we naast de edit link een delete link zetten.
2. Deze link roept een control aan (delete.php) en geft via de GET methode de key van de regel mee zodat de delete.php weet welke regel moet worden verwijderd.
3. In de delete.php roepen we een nieuwe te maken method aan in de crud.php met als parameter de regel (key) die moet worden verwijderd.
4. In de CRUD maken we een extra method die met behulp van de pararameter die is meegegeven uit crud.php, de jusite query uitvoerd. let op dat je de jusite query uitvoerd want met een SQL delete statemnet kun je vrij eenvoudig de hele tabel leegmaken. Wellich weil je eerst de query laten afdrukken zodat je kunt controleren of de query juist is.
5. Denk aan security; de parameter wordt via GET meegegven. Zorg ervoor dat je de input eerst goed valideerd. Welke parameter wil je absoluut niet? Waarom niet en hoe houd je die tegen?
6. Eventuuel kun je nog een bevestiging opvragen: "Are you sure you want to delete....."

---

# 7.1 Bankrekeningnummer - check

Image result for iban nummer
Image not found or type unknown

Een bankrekeningnummer in Nederland bestaat niet uit zomaar willekeurige nummers. Niet elk nummer van 9 cijfers is een geldig en goed bankrekeningnummer. Een bankrekeningnummer bestaat een aantal velden, bijvoorbeeld:

## NL 69 INGB **0123456789**

De eerste twee tekens NL, staan voor het land, 69 is een controlenummer en INGB is de bank (in dit geval ING). Dan volgt er een een rekeningnummer van 10 cijfers. Deze opdracht gaat voer deze laatste 10 cijfers (in donkerrood weergegeven).

Om typefouten te voorkomen, wordt er in Nederland de elfproef gedaan met rekeningnummers. Dit betekent dat de nummers in een rekeningnummer vermenigvuldigd worden met 10 terug naar 1 en als de som van die getallen gedeeld door 11 een geheel getal is, dan is het een geldig rekeningnummer.
Bijvoorbeeld:

Rekeningnummer: 12 34 56 789
Som: 1*9 + 2*8 + 3*7 + 4*6 + 5*5 + 6*4 + 7*3 + 8*2 + 9*1 = 165

165/11 is een geheel getal want 11 x 15 = 165 en 165 % 11 = 0, omdat de rest van de deling 165/11, 0 is.

12 34 56 789 is dus een geldig rekeningnummer.

**Opdracht**

Maak een functie die een rekeningnummer meekrijgt (alleen maximaal 10 cijfers) en die true of false terug geeft. True als het een goed bankrekeningnummer is en false als het een niet goed bankrekeningnummer is.

Let op netjes inspringen en lever de code in een .php file aan in Teams.

--

# 7.2 Winst of verlies bij 3 x 6

## Hoe lang duurt het om 3 x 6 te gooien met dobbelstenen?

Image result for 3 x 6 dobbelstenen
Image not found or type unknown

Iemand biedt stelt jou het volgende voor? Jij betaalt 1 euro voor elke worp met drie dobbelstenen en als je 18 gooit dan krijg je 150 euro. Je mag dit zo vaak doen als je wilt. Stel je betaald 100 euro en gooit 100 keer. Wat verdien je dan? Of verlies je, en zo ja hoeveel?

$ogen = gooi($n)

Maak een functie gooi($n) waarbij n het aantal dobbelstenen is waarmee je gaat gooien en de functie returned het totaal aantal ogen dat je gooit. Gebruik de rand() functie van php om een dobbelsteen na te doen.

$gewonnen = wedden($aantalworpen, $winstbij, $winst)

Maak een functie wedden($aantalworpen, $winstbij, $winst) waarbij je het aantal keer dat je gooit, de gewenste uitkomst en de uitbetaling meegeeft. Bijvoorbeeld als je wilt weten hoeveel geld je krijgt als 1000 keer gooit en 100 euro krijgt als je 18 hebt gegooid dan roep je de functie als volgt aan: wedden(1000, 18, 100). De computer 'gooit' nu 1000 keer en elke keer als er 18 wordt gegooid dan 'verdien' je 150 Euro. De return value is de hoeveelheid geld dit je hebt verdient, bijvoorbeeld 1200 euro.

En bij welke winstuitkering zou jij deze weddenschap willen aangaan?

**Opdracht**

Lever een .php file in met de functie *wedden*. Spring netjes in!

Video uitleg over functies: https://youtu.be/XfnH3AEF5Z8

Video uitleg over stap 1 en 2 van deze opdracht: https://youtu.be/xyxGQMVXteY

## Stap 1

Maak een de functie gooi($n).

> *parameter* $n is het aantal dobbelstenen
> *return value*: totaal aantal ogen van de $n dobbelstenen

De functie gooit dus met drie dobbelstenen (of met één dobbelsteen drie maal) en geeft het totaal aantal ogen terug.

(note: de *return value* is de waarde die met return() door een functie wordt teruggegeven)

Voorbeelden

```
echo gooi(1);         // dit drukt 1, of 2, of 3, .. , of 6 af.
$a=2; echo gooi($a);  // dit drukt 2, of 3, of 4, of 5, ...., of 12 af.
$b= gooi(3); echo $b; // dit drukt 3, of 4, of 5, of, ....., of 18 af.
```

Tip: Gebruik de PHP rand() functie om een willekeurig getal tussen de 1 en de 6 te bedenken ( https://www.w3schools.com/php/func_math_rand.asp). Op die manier maak je een 'elektronische' dobbelsteen.

In https://youtu.be/xyxGQMVXteY wordt stap 1 en stap 2 nog ene keer uitgelegd.

# Stap 2

Gebruik de variabele $aantalworpen en geef deze de waarde 100. Gooi vervolgens $aantalworpen keer en druk telkens af hoeveel de uitkomst van de worp is.

```
$aantalworpen = 100;


// voer deze code $aantalworpen keer uit
echo "De uitkomst van een worp met 3 dobbelstenen is: ".gooi(3);
```

Tip: zorg er eerst voor dat regel 4 (de regel die met echo begint) 100x wordt uitgevoerd. Maak daarna de code, zodat regel 4 $aantalworpen keer wordt uitgevoerd.

# Stap 3

(a) Verander de code uit stap 2, zodat je alleen de echo regel afdrukt als je 18 hebt gegooid.

(b) Houd in een aparte teller ($wins) bij hoeveel keer je 18 hebt gegooid.

(c) Druk aan het eind van je code af hoeveel keer je hebt 'gewonnen':

```
echo "Je hebt $wins keer gewonnen.";
```

(d) Maak nu een nieuw variabele en zet die boven aan in je main code. De main code is alle code die *niet* in de functie staat. Noem deze variabele $winsPerKeer en zet daar 100 in. Dit betekent dat telkens als je 18 gooit, je $winsPerKeer Euro wint; in dit geval dus 100.

(e) Verander nu de regel die je bij stap (c) hebt toegevoegd en zet daarbij hoeveel euro je gewonnen hebt.

De output ziet er afhankelijk van hoe vaak je hebt gewonnen, als volgt uit:

```
De uitkomst van een worp met 3 dobbelstenen is: 18
De uitkomst van een worp met 3 dobbelstenen is: 18
De uitkomst van een worp met 3 dobbelstenen is: 18
Je hebt 300 euro keer gewonnen.
```

# Stap 4

a) maak nu van de 18, dat is het aantal ogen waarbij je winst, een variabele. Noem deze $winstbij.

b) Maak nu van de code die (nog) niet in de functie staat een de functie:

```
wedden($aantalworpen, $winstbij, $winstPerKeer);
```

De functie returned het gewonnen bedrag (in het voorbeeld hierboven de 300).

--

--

# 8 Hoofdstedenspel deel-2

We gaan ons eerder gemaakte hoofdstedenspel uitbreiden.

## Opdracht

Met het hoofdstedenspel dat we eerdere hebben gemaakt hadden we alle landen en hoofdsteden uit Europa in een Assiociative Array staan. We gaan nu gebruik maken van een SQL-database.

Download de *world* database.sql en importeer deze in een nieuwe database die we *world* noemen.

De database inhoud wordt in meer detail hier beschreven.

De quiz wordt nu uitgebreid. Je kunt als gebruiker eerst kiezen welk continent je wilt 'spelen'. Als je een continent hebt gekozen dan wordt er willekeurig het land gekozen en er worden er ook willekeurig 6 hoofdsteden getoond waaruit de speler de juisten hoofdstad moet kiezen. De steden worden getoond in een form met zogenaamde radio buttons.

De landen en hoofdsteden worden uit de database gehaald.

Eén spel bestaat uit 10 vragen.

Na 10 vragen wordt er getoond hoeveel vragen de gebruiker goed had.

## Optie

Voeg een scorebord toe. Per continent wordt er een top 10 van scores bijgehouden. Bij gelijke scores wordt de meest recent behaalde score bovenaan gezet. Dus als er 10 spelers met allemaal 10 punten zijn dan komt het meest recent gespeeld spel bovenaan. Een speler die in de top 10 van het gekozen continent komt kan zelf een naam kiezen. Deze naam wordt dan in de score list getoond.

--

# 9 Corona simulatie in PHP

*In deze les gaan we code van een ander lezen en aanpassen. In het kader van Covid-19, (Corona) heb ik een virus-simulatie gemaakt. Dit is een* ***zeer vereenvoudigde weergave van de werkelijkheid*** *en je moet dit niet zien als een simulatie van de echte wereld. Dus alle getallen die je zien moet je niet betrekken op hetgeen er nu in de echte wereld afspeelt.*

## De simulatie ziet er in de browser als volgt uit.


In het 'grid' (het grote gedeelte bovenin) zie je de bevolking. Elke karakter is een persoon. Een - is een gewond persoon, een oranje s is een ziek persoon, een groene o is een immuun persoon en een rode X is een overleden persoon. Met de knoppen kun je één, drie of zeven dage vooruit springen. In de grafiek zie je het aantal zieke personen.

# Installeer de simulatie

Zet de twee php files in één directory in je document root en test de simulatie uit.

# MVC

MVC is een principe waarin je bepaalde zaken scheidt. M staat voor **Model**, dat is de database of eigenlijk de opslag en toegang tot data. De V staat voor **View,** dat is eenvoudig gezegd de GUI, wat de gebruiker ziet: daar zit dus vaak veel HTML en JavaScript bij. Later zullen we ook libraries als bootstrap gaan gebruiken voor de GUI. De C tenslotte, staat voor **Control** en dat is alle logica, zeg maar de 'brains' van het systeem.

# Opdracht 1

Beschrijf van de hieronder genoemde onderdelen van de software of ze bij M, V of C horen.

(a) Regel 141 t/m 145

(b) De functie *printPopulation* (regel 57-77)

(c) De functie *passTime* (regel 41-55)

(d) Regel 214 t/m 230 (de table)

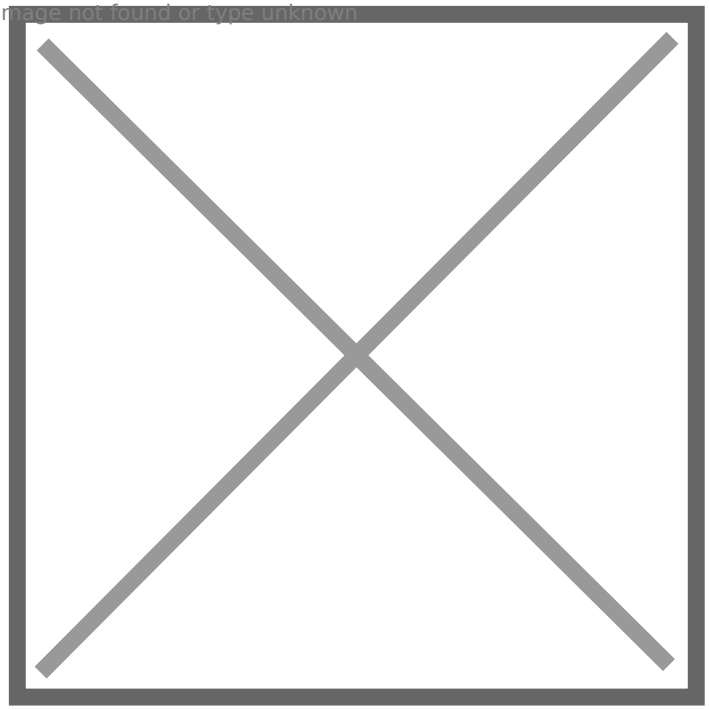
# Opdracht 2

De knop "Next 7 >>>" die je in de screen afdruk ziet springt een week vooruit in de simulatie. Deze knop is nog niet geïmplementeerd. Pas de code aan zodat je deze knop wel hebt en zorg ervoor dat de knop ook werkt.

# Opdracht 3

In de 'grid' staat elke oranje s voor een ziek persoon, een groene o staat voor iemand die immuun is geworden, - staat voor iemand die nog niet besmet is en ene rode X staat voor iemand die is overleden.

Image not found or type unknown



Verander de code zodat iemand die niet is besmet niet wordt weergegeven, zie voorbeeld hierboven.

# Opdracht 4

We gaan functionaliteit uitbreiden zodat de simulatie-parameters kunnen worden aangepast.

Op regel 3 t/m 8 worden de parameters van de simulatie gedefinieerd.

(a) Zet deze definities in een aparte file, parameters.php en `include 'parameters.php'` in de corona.php file.

(b) Maak een form (editparams.php) om de parameters aan te passen. In dit form worden de waardes uit de parameters.php ingelezen (met een include). Deze parameters worden gebruikt als default values voor het form. Als je het form post dan wordt er een nieuwe parameters.php file geschreven. Hierin staan de aangepaste waarden. Na het aanpassen van de waarden wordt de simulatie (corona.php) vanzelf aangeroepen.

(c) Maak een extra knop op de corona.php pagina die je kunt gebruiken om naar de editparams.php pagina te gaan.

(d) Maak een extra knop op de editparams.php. Indien je deze knop indrukt worden alle waarden in het form weer op de standaard waarden ingesteld. De standaard waarden zijn de waarde zoals ze in de code staan:

```
population=4200;
$popWidth=120;
$contactPerPerson=10; // number of contacts per person
$changeToGetSick=5;   // when you contact someone change to get infected
$changeToDie=3;       // when you are infected, change to die
$daysSick=10;         // days sick
```

# Opdracht 5

Pas de code aan, zodat de simulatie begint met 100 mensen die immune zijn.

Pas de editparams.php aan, zodat dit aantal immune personen kan worden aangepast.

--

# 10 The Challenge ***

*Snel zoeken in password hash lijst.*

## Wat ga je leren?

- met hash functies werken
- PHP en files lezen en doorzoeken
- een programmeer-opdracht opdelen in stapjes; een plan maken.

## Inleiding

Er is een zeer lange lijst met password hashes. Je kan daarin zoeken of jouw password is gehacked. Dit kan je online doen, maar je wilt je super-goede-geheime password waarschijnlijk niet zomaar naar een of andere site sturen. We gaan dus onze eigen lokale password validation maken. En de uitdaging is wie de snelste code kan maken. Hoe snel kan jij een hash vinden in een half miljard password hashes. Ik kan je vertellen dat dat op een i5 laptop met SSD onder de paar seconden moet kunnen.

## Have I been powned?

Kijk op https://haveibeenpwned.com/Passwords je kunt daar je password invullen en controleren of het een 'bekend' (dus gehacked) wachtwoord is. Deze wachtwoorden zijn onder hackers bekend en zijn dus eigenlijk niet (meer) veilig. Maar als je de controle wilt uitvoeren zonder jouw wachtwoord over het internet te sturen dan zal je zelf iets moeten maken. En ja, de site heeft ene secure connectie via SSL, maar wat als de programmeur stiekem ene lijstje bijhoud van alle wachtwoorden die worden geprobeerd?

## Case

We gaan dus onze eigen versie maken van 'Have I been powned'. Daarvoor moet je het grote bestand met hashes downloaden. Deze staat op de genoemde site. Kies de versie in het format SHA-1 ordered by hash.

De file ziet er als volgt uit:

```
000000005AD76BD555C1D6D771DE417A4B87E4B4:4
00000000A8DAE4228F821FB418F59826079BF368:3
00000000DD7F2A1C68A35673713783CA390C9E93:630
00000001E225B908BAC31C56DB04D892E47536E0:5
00000006BAB7FC3113AA73DE3589630FC08218E7:2
00000008CD1806EB7B9B46A8F87690B2AC16F617:4
0000000A0E3B9F25FF41DE4B5AC238C2D545C7A8:15
0000000A1D4B746FAA3FD526FF6D5BC8052FDB38:16
0000000CAEF405439D57847A8657218C618160B2:15
0000000FC1C08E6454BED24F463EA2129E254D43:40
```

Elke regel bestaat uit een password hash, een dubbele punt en een getal. Dit laatste getal geeft aan bij hoeveel hacks het password is gevonden. Het enige dat je moet is dus een password hash zoeken.

# Hints

Onder Linux kun je met het wc (word count) commando vrij snel bepalen hoeveel regels het bestand heeft. Je zult er snel achter komen dat gewoon de file doorlopen en regel voor regel kijken of je de juiste hash hebt gevonden niet werkt. Dit ligt natuurlijk wel aan de snelheid van jouw computer/laptop. Probeer maar eens in te schatten hoelang het duurt voordat je bijvoorbeeld 10% van de file heb doorzocht. Ik heb een algoritme genaakt dat op elke eenvoudige laptop de hash binnen 1 seconden vind.

Als je gaat nadenken over een strategie bedenk dan hoe jij het handmatig zou aanpakken?

Deel nu het probleem op in stapjes en test elke stapje. Wat zijn je stapjes en hoe wil je het aanpakken.

## Maak het probleem eenvoudiger.

Als je eer nog niet uitkomt, maak het probleem dan eenvoudiger. Bijvoorbeeld je hebt 20 hashes:

```
1200
1201
2011
2045
2234
3400
4000
4001
4010
```

```
4098
4099
5332
8020
8100
8201
8245
8376
8898
8999
9010
```

Stel je zoekt de hash van het een wachtwoord en de hash is 9500. Ga jij nu 20x maal kijken of de hash gelijk is of zie je di sneller? En stel je zoekt naar de hash 8020, ga je dan alle hashes vergelijken of die je dit op een snellere manier. Probeer een plan te maken hoe je zelf in zo min mogelijk stappen bepaald of een hash in de lijst staat. Deze strategie kun je dan mogelijk ook toepassen op het grote bestand.

Maar een plan en bespreek dat eventueel met je docent voordat je begint. Een goed plan is het halve werk!

--

# Alle PHP Videos

## PHP For loop

Basis uitleg van hoe een for loop werkt.

Een lange lijst tekst wordt in de browser afgedrukt met echo. Deze code wordt vereenvoudigd met een for-loop.

https://youtu.be/dlgq69XKiR0

## Waarom count(*) en hoe af te ronden?

Waarom gebruik je count($array) als je een array doorloopt in een loop en hoe rond je getallen af? Dit voorbeeld gaat verder met de code van de vorige video (PHP For Loop).

https://youtu.be/kvrKIa68Rv0

## Wat is het verschil tussen een gewoon array en een assiociative array?

Een gewoon één dimensionaal array is eigenlijk een assiociative array met een automatisch gegenereerde key.  In dit fimpje worden de twee arrays met elkaar vergeleken.

https://youtu.be/5lJLecI0BTA

## Assiociative arrays en arrays en loops.

Hoe bereken ik de gemiddelden van de cijfers per persoon uit de onderstaande datastructuur?

```
$results= [ 'Jori' => ['6','7','6','9'], 'Peter' => ['7','7','8'],
        'Sid'=> ['6','5','5'], 'Sarah-lin' =>['9','4','6'],
        'Djab' => ['7','7','7'], 'Lin' => ['6','5','6'],
        'Maria'=> ['6'], 'Tjeerd'=> ['7','7','6','9'] ];
```

https://youtu.be/VWBbfVtpDoY

## Wat is een function (functie) en waarom gebruik je die?

Basis uitleg van een functie in PHP. Er wordt een vergelijking gemaakt met een snoep/frisdrank-automaat en een functie. Dan wordt uitgelegd hoe je een functie maakt en waarom je ene functie maakt.

https://youtu.be/XfnH3AEF5Z8

## Hoe filter je met een functie alle even waarden uit een array?

In dit voorbeeld gaan we een wat ingewikkelder functie maken. Deze functie krijgt als parameter een array mee en de retun value is ook een (ander) array.

https://youtu.be/INHM0S_8GHA

## Van functies naar objecten (OOP) (1/3)

Het gebruik van global variabelen heeft nadelen maar zijn in sommige gevallen toch handig. Wanneer is dit het geval? En wat is de oplossing?

https://youtu.be/9CCGc4xZy_0

## Definities OOP (2/3)

Definities aan de hand van object hond.

https://youtu.be/tZJcsFp9ttU

## Class en object in PHP 'MijnGemiddelde' (3/3)

Eerste class in PHP

https://youtu.be/z7mB1jRRy_Y

--

# 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
Image not found or type unknown

# 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
Image not found or type unknown

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
Image not found or type unknown

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
Image not found or type unknown

# 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
Image not found or type unknown

# 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
Image not found or type unknown

# 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).

--

# CRUD Challenge

Voor deze challange gebruik je kennis die je in de PHP modules en in de BNB challenge hebt geleerd. Je gaat werken met forms en je gaat gegevens in een database opslaan en opvragen.

De bedoeling is dat je een CRUD gaat maken. Wat is een CRUD? Dat leggen we zo uit. Eerste het voorbeeld, dat kan je vinden op:

https://stampwerk.nl

Weer een te laat-1.jpg
Image not found or type unknown

Je ziet hier een overzicht van studenten die te laat zijn. Met de groene knop onderaan de pagina kan je een nieuwe te laat melding aanmaken, probeer maar!

# Wat is een CRUD?

CRUD staat voor Create, Read, Update en Delete.

crud.jpg
Image not found or type unknown

Deze vier functies zijn de basisfuncties die je op een tabel uit de database kan uitvoeren. Stel je hebt een tabel student, je kunt een student toevoegen (Create), je kunt een overzicht krijgen van studenten (Read), je kunt de gegevens van een student aanpassen (Update) en als laatste kun je een student ook weer verwijderen.

De challenge is dat jij een CRUD gaat maken voor te laat meldingen. Maak daarvoor eerst een tabel in de database waarin je te laat meldingen kan registreren.

Als je naar het voorbeeld kijkt dan zie je dat je van een te laat melding de volgende gegevens wilt vastleggen:

- naam van de student
- klas
- aantal minuten te laat
- de reden van het te laat komen.

# Stap 1, de database

Maak een database en maak een tabel zoals je dat ook in de BNB challenge hebt gedaan. Zet de velden in de database en geef de velden het juiste datatype. Vergeet niet een primary key toe te voegen!

# Stap 2, Read

- Zet een paar regels in de database en begin met het maken van een overzichtspagina (zie voorbeeld).
- Zet alle gegevens netjes in een tabel, netjes onder elkaar en vergeet niet de kopregels (headers) toe te voegen.
- Je mag voor de lay-out bootstrap gebruiken.

# Stap 3, Create

- Maak dan de knop '*Weer eentje te laat*' (zie voorbeeld)
- Zorg dat er na het klikken op deze knop een nieuwe pagina met een formulier getoond wordt.
- Voeg een knop '*Invoegen*' toe waarmee je de gegevens aan de database toevoegt.
- Met behulp van dit formulier moet je een een nieuwe te laat melding kunnen toevoegen aan de database en dus ook aan de overzichtspagina.
- Als je foute gegevens invoerd, zoals -300000 bij het aantal minutne te laat dan verschijnt er een nette melding naar de gebruiker. Bij voorkeur controleer je alle invoervelden client side. Zoek zelf op hoe dat moet.

# Stap 4, Delete

- Voeg een kolom in de tabel op de overzichtspagina toe en plaats op elke regel een knop 'V*erwijder*' (zie voorbeeld).
  Als deze knop wordt ingedrukt dan wordt de te laat melding verwijderd uit de

overzichtspagina en dus ook uit de database.

- Voordat je daadwerkelijk een te laat melding verwijderd, vraag je eerst een bevestiging aan de gebruiker.

# Stap 5, Update

- Dit is een extra uitdaging, met update kun je de gegevens in de database updaten.
- Voeg een kolom in de tabel op de overzichtspagina toe en plaats op elke regel een knop 'update' (zie voorbeeld).
- Als deze knop wordt ingedrukt dan wordt er een nieuwe pagina getoond met een formulier waarin de gegevens staan van de rij waar je op geklikt hebt. Het formulier dat je gebruikt om de update uit te voeren mag dus niet leeg zijn.
- In dit formulier kunnen de gegevens worden aangepast.
- Voeg een knop update toe waarmee je de gegevens in de database update.

  Tip:

  Het is lastig om in het formulier de oude waarden te laten zien. Je kan dit als volgt oplossen.

  Als je het formulier opent geef je het id mee van de regel die je wilt updaten. Dat kan je het eenvoudigst via een $_GET doen. Controleer of je het ID in het formulier ook daadwerkelijk kan lezen.

  Als je het ID hebt dan voer je een query uit en haal je alle gegevens op die bij dit id horen. Die gegevens gebruik je daarna om de values van het formulier te vullen. Op deze manier zie je in het formulier de oude waarden en hoef je niet alle waarden opnieuw in te voeren.

# *Inleveren*

- Eén screenshot van jouw Read (overzichtspagina) scherm.
- Eén screenshot van jouw Create (pagina met formulier om een item toe te voegen) scherm.
- Eén screenshot van jouw Update (pagina met formulier om een item aan te passen) scherm.
- ***Alle*** PHP files, sql-export en (eventueel) CSS file die je hebt gebruikt.

# *Puntentelling*

(je moet **meer dan 80 punten** of meer halen voor deze challenge).

| | |
|---|---|
| 10 | Database met tabel maken en de juiste gegevens vastleggen in de tabel. Denk ook om de datatypes. |
| 10 | Read, een overzicht van alle te-laat meldingen uit de database maken. |
| 10 | Het (Read) overzicht netjes met CSS styles formateren (mag ook via Bootstrap). |
| 20 | Create, een nieuwe te laat melding kunnen maken. |
| 10 | Invoercontrole. Bij een nieuw te laat melding kan je verkeerde zaken invoeren, zoals bij minuten te laat een woord, of een negatief getal. Je kunt ook een naam invoeren van een student met een lengte langer dan in de database is toegstaan. Zorg ervoor dat er naar de gebruiker een nette melding komt als je iets verkeerds invoert. |
| 20 | Update, je kunt een te laat melding uit het overzicht aanpassen. |
| 10 | Delete je kunt een te lat melding uit het overzicht verwijderen. |
| 10 | Als je een te laat melding verwijderd wordt er eerst om een bevestiging gevraagd. |

# PHP exec python script

```
<?php
error_reporting(E_ALL);
$command = 'python3 /home/uploads/testing.py';
$command = escapeshellcmd($command);
$shelloutput = exec($command,$output, $ret_code);
echo "<h1>";
echo $shelloutput;
echo $output;
echo $ret_code;
echo "</h1>";
?>
```