Yii level 2

- 01 Waar zijn we gebleven?
- 01 Quiz
- 02 Opdracht
- 03 Eigen overzicht (Read)
- 04 Overzicht van kleinste landen in Europa
- 05 1:1 Relaties, hoofdsteden
- 06 Relaties, land en talen

01 Waar zijn we gebleven?

We kijken even terug wat we in Yii L1 hebben gedaan en aan het eind van deze terugblik configureren we onze bestaande Yii installatie zodat we in de vervolg stappen beschikken over twee handige debug functies.

We hebben in de vorige lessenserie over Yii hebben we het volgende behandeld:

- installatie
- CRUD maken (via GUI in Yii)
- MVC
- Routing
- Menu in Yii
- Grid view in Yii

Weet je nog wat dat allemaal is, doe dan de quiz. Wil je de quiz super goed maken of weet je niet precies meer wat al deze termen inhouden lees dan even verder.

Samenvatting Yii - level 1

Installatie YII

Voor de installatie hebben we composer nodig. Als composer is geïnstalleerd dan kunnen we een nieuw Yii project beginnen met het commando:

```
composer create-project --prefer-dist yiisoft/yii2-app-basic <naam van het nieuwe project>
```

<naam van het nieuwe project> moet je dan vervangen door de naam van jouw project (in de vorige lessen *world*).

Als je met (bijv.) VCS jouw nieuwe Yii project folder opent dan kun je de Yii webserver starten met:

```
php yii serve
```

Als Yii eenmaal is geïnstalleerd dan moet je nog een paar zaken wijzigen zoals de database naam en de manier waarop jij routing wilt gebruiken. Hiervoor moet je de volgende bestanden aanpassen:

- config/web.php
- config/db.php

Wil je meer ddetaisl weten, kijk dan nog een keer hier.

CRUD

Crud staat voor Create, Read, Update en Delete. Dat zijn de vier basishandelingen die je op een tabel in de database kunt uitvoeren. Dat kan met plain PHP, met Yii, met Laravel, en met nog veel meer systemen. Crud is vaak de basis van een web applicatie. Met Yii kun je een hele crud applicatie laten genereren. Je hoeft heir dus niets voor te coderen.

MVC

MVC staat voor model, view controller. En is een manier waarop je in Yii en Laravel je code opdeelt.

In het **model** beschrijf je hoe en waar de data in de database wordt opgeslagen.

In de view **beschrijf** je hoe de data aan de gebruiker wordt getoond Hierin zt vaak veel HTML/CSS en andere opmaak elememten.

In de **controller** beschrijf je waar de gegevens in de view vandaan komen en voer je eventueel bewerkingen of berekeningen uit op de data.

Routing

Routing is de vertaling van de URL (jouw webadres) naar een bestand op de webserver.

Dus hoe vertaald www.domain.com/rapport/edit.php?id=1 zich naar een bestand. Of stel dat je op de pagina van de genoemde URL iets wit wijzigen hoe weet je dan welk bestand je moet openen?

In Yii en Laravel is de routing anders dan bij XAMPP.

Wil je dit nog eens nalezen klik dan hier.

Menu

Yii heeft zijn eigen ingebouwde manier om een menu te maken. In de menu les van Yii Level 1 hebben we gezien hoe je een menu kan aanmaken of kan wijzigen.

Yii Grid

Yii heeft een handige grid view waarmee je data in een soort Excel blad kan zien. Je kan sorteren en zoeken net als Excel en je hoeft er bijna niets voor te coderen. In deze les van Yii level 1 hebben

we kennis gemaakt van de Yii grid view.

Opdracht debugging

Om straks de opgave goed te kunnen maken, gaan we een eenvoudige debug functie maken.

Easy Debugging

Als je de aanpassingen in dit onderdeel uitvoert, dan kun je later in de code de volgende twee functies gebruiken:

`d($objectnaam)` bekijk $object de code loopt gewoon door.
`dd($objectnaam)` bekijk $object de uitvoering van de code stopt.

Hoe krijg je dit? Eenvoudig twee stappen.

Stap 1, New file config/functions.php

```
<?php
/**
 * Debug function
 * d($var);
 */
function d($var,$caller=null)
{
    if(!isset($caller)){
        $caller = debug_backtrace(1)[0];
    }
    echo '<code>Line: '.$caller['line'].'<br>';
    echo 'File: '.$caller['file'].'</code>';
    echo '<pre>';
    yii\helpers\VarDumper::dump($var, 10, true);
    echo '</pre>';
}

/**
 * Debug function with die() after
 * dd($var);
 */
function dd($var)
{
```

```
    $caller = debug_backtrace(1)[0];
    d($var,$caller);
    die();
}
```

Stap 2, Edit config/web.php - voeg regel 3 en 4 toe.

```
<?php


/* Include debug functions */
require_once(__DIR__.'/functions.php');
```

Inleveren

Niets, maar zorg ervoor dat je de aanpassingen goed hebt uitgevoerd anders loop je in de vervolg stappen vast.

--

01 Quiz

Hoe start je in Yii de development server op?

1. yii serve
2. yii php serve
3. yii php-serve
4. php yii serve

Met welke tool installeer je een nieuw Yii project?

1. installer (msi)
2. composer
3. laravel
4. yii

In welk bestand wordt de database connectie in Yii gedefeineerd?

1. config/db.php
2. config/database.php
3. config/databas.conf
4. config/secret.php

Stel je ziet in de browser de volgende URL: localhost:8080/bestelling/overzicht
Waar verwijst overzicht naar?

1. de view
2. de controller
3. de method/function in de controller

4. de variabele in de controller

Stel je ziet in de browser de volgende URL: localhost:8080/bestelling/overzicht
Waar verwijst bestelling naar?

1. de view
2. de controller
3. de method/function in de controller
4. de variabele in de controller

Waar staat MVC voor?

1. Multiple Vhost Config(uration)
2. Modular Virtual Codebase
3. Mogelijk Verkeerde Code
4. Model View Controller
5. Model View Code

Waar staat de meeste HTML code voor de opmaak van de site?

1. In het Model
2. In de View
3. In de Controller
4. In de Configuration

In Yii heb je de GirdView widget (deze is uniek voor Yii). Je hebt deze widget al gebruikt. Waarvoor gebruik je de GridView Widget?

1. Om een Read view te maken waarin je selecties kan maken en kan sorteren.
2. Om een eenvoudig Read view te maken.

3. Om een edit view te maken.
4. In de controller om de juiste selectie voor de read view te maken,

Veel code in Yii ziet er ongeveer zo uit:

```
[
    'label' => 'Bovolkingsdichtheid',
    'attribute'=>'Population',
    'format' => 'raw',
],
```

Wat is dit voor een data-structuur?

1. Dit is een indexed array
2. Dit is een associatieve array
3. Dit is een class
4. Dit is een loop

02 Opdracht

```
['class' => 'yii\grid\ActionColumn', 'template' => '{update} {view} {delete}',],
```

Zorg ervoor dat jouw Yii World project weer draait.

In deze opdracht hebben we een refresh en gaan we nog een keer kijken naar de standaard Yii gridview. We gaan de gegenereerde grid view van country aanpassen.

Wat leren we?

- we frissen onze kennis weer wat op en leren met het MVC model te werken.
- we leren hoe we de standaard Yii grid view kunnen tweaken (aanpassen).
- we leren hoe je in PHP ene getal kan laten afronden.
- we nemen ook een klein stukje CSS-vormgeving mee.

Checklist

1. Controleer of je de views Country, City en CountryLanguage nog hebt. Heb je de models én de controllers?
2. Open jouw Yii project in VSC
3. Controleer de map models, heb je de bestanden 6 bestanden City.php, CitySearch.php, Country.php, CountrySearch.php, CountryLanguage.php en CountryLanguageSearch.php?
   Als je deze bestanden niet hebt, ga dan terug naar Yii deel 1 en doorloop nog een keer alle stappen.
4. Type `php yii serve` in het terminal window in VCS in en controleer op localhost:8080/country of you website het nog doet.

OK all fine, let's go!

Om er weer in te komen gaan we onze country view aanpassen.

In stap 1 is dit wat er moet uitkomen. Controleer de kolommen. Heb je deze kolommen al? Dat kan want de laatste opgave van Yii Level 1 ging ook over dit onderwerp. Lees de opgave toch even door want je moet het overzicht precies zo maken als in het voorbeeld inclusief het afronden op twee decimalen.

image-1655802388771.png
Image not found or type unknown

We hebben de volgende kolommen:

image-1655802427760.png
Image not found or type unknown

Maak de eerste drie kolommen *Name*, *Continent* en *Population* (in Yii L1 wordt uitgelegd hoe dat moet).

De kolom *Bevolkingsdichtheid* is een berekende kolom. De bevolkingsdichtheid is het aantal inwoners (Population) gedeeld door de oppervlakte (SurfaceArea).

```
Bevolkingsdichtheid = aantal inwoners / oppervlakte
```

Je kunt daarvoor de volgende Yii-code op de juiste plaats in de Grid view van country zetten:

```
[
  'label' => 'Bovolkingsdichtheid',
  'attribute'=>'Population',
  'contentOptions' => ['style' => 'width:30px; white-space: normal;'],
  'format' => 'raw',
   'value' => function ($data) {
      return  $data->Population/$data->SurfaceArea;
   }
 ],
```

Opdracht: Plaats de code op de juiste plaats in jouw grid view van country.

Opdracht: Test de code. Zie je dat het werkt?

Nu alleen nog de getallen afronden op **2 decimalen**. Je kunt hiervoor de php functie *number_format* gebruiken.

Opdracht: Zoek nu zelf uit hoe je de bevolkingsdichtheid kan afronden op twee decimalen.

Als laatste stap gaan we onze kolom nog wat stylen. Check de gegeven code. Zie je de styling?

Opdracht: Pas de **styling** aan zodat je kolom er als volgt, komt uit te zien:

image-1655802941138.png
Image not found or type unknown

Opdracht: Pas je jouw overzicht aan en zet je jouw voornaam in de kolom header Bevolkingsdichtheid.

Opdracht: De eerste kolom met het volgnummer moet weg.

Opdracht: De laatste kolom heeft drie icoontjes, oogje, pennetje en prullenmand. Verwijder het oogje.

Deze laatste opdracht is lastig en de opdracht is om zelf proberen uit te vinden hoe dat werkt. Als hint wordt deze code gegeven:

```
['class' => 'yii\grid\ActionColumn', 'template' => '{update} {view} {delete}',],
```

Probeer deze code uit door het toe te voegen als kolom. Bedenk dan hoe je de het oogje (het view-knopje) kan weghalen.

Uiteindelijk ziet de grid view er zo uit:

image-1656507597597.png
Image not found or type unknown

Als de grid view er zo uit ziet als hierboven, kan je de opdracht inleveren.

Inleveren

1. Screenshot van je aangepaste city view zoals het voorbeeld hierboven, maar dan met jouw naam in de kolomnaam.
2. Jouw aangepaste code (*view/country/index.ph*p).

03 Eigen overzicht (Read)

Inleiding

We gaan een eigen overzicht maken. We beginnen met het aanmaken van een menu, dan maken we een controller, en van de controller gaan we naar de view. We leren data op te halen met de controller en deze te tonen in onze eigen view.

Het ophalen van data doen we met een eenvoudige query die we in de Controller plaatsen. Vanuit de Controller gaan we via het Model naar de database. De volgorde wordt:

```
Gebruiker -> (kiest) menu -> controller -> model -> database -> (resultaat terug naar) controller -> view -> gebruiker.
```

In deze les maken we dus het hele 'rondje'. Dit is een standaard 'rondje' dat in elke applicatie vaak terug zal komen.

Wat leren we?

- we leren werken met het MVC model
- we gaan leren hoe je een eigen overzicht (de read van de CRUD) kan maken (dus niet via de Yii grid view).

Om een eigen overzicht te maken, gaan we drie opdrachten maken.

1. We gaan een menu item maken en kopellen die aan een nieuwe (eigen) method in een controller.
2. We gaan vanuit de controleer data ophalen uit de database.
3. Met de data maken we een eigen view.

Menu

Om te beginnen maken we een nieuwe method (=functie) in de **controllers/CountryControlle**r class.

De function/method returned "oops" en doet verder niets.

Deze controller noemen we actionOverzicht. Volgens de routing regels wordt de route wordt dus /country/overzicht.

Opdracht 3a

> Maak een menu item *overzicht* onder country dat naar */country/overzicht* gaat.
>
> In Yii level 1 hebben we met menu's geoefend.
>
> Image-1614620329168.png Unknown
>
> Test het menu uit. Het menu gaat naar de actionOverzicht in de CountryController en returned "oops". We krijgen dus een leeg scherm met "oops".

Inleveren

- Een schermafdruk van jouw gehele browser (url moet zichtbaar zijn) waarin je het via het menu naar jouw overzicht gaat en waar "oops" wordt afgedrukt.

Controller

We halen onze "oops" weg en gaan dus een stukje code toevoegen aan onze CountryController.

De controller komt er als volgt uit te zien.

Let op zet jouw naam op regel 3.

```
public function actionOverzicht()
{
    // zet hier jouw naam
    // dit is de query, dit is te vergelijken met select * from Country
    $countries=Country::find()->all();

    // de view wordt aangeroepen en het object $countries en $pagination wordt meegegeven.
    return $this->render('overzicht', [
        'countries' => $countries,
    ]);
```

```
}
```

Opdracht 03b

> Maak de function *actionOverzicht* in de *CountryController* zoals hierboven is aangegeven.

Inleveren

- Leg uit waarom je een foutmeldign krijgt.
- CountryController.php

View

image-1614623472969.png Image not found or type unknown

What? Een fout? Waarom nu?

Juist het menu gaat naar de actionOverzicht en de actionOverzicht voert een query uit en roept de view overzicht op. En die view bestaat (nog) niet.

We gaan dus in de folder views/country een nieuwe file overzicht.php aanmaken en plaatsen daar de volgende code in.

```
<?php
    foreach ($countries as $country) {
        echo $country->Name;
        echo " - ";
        echo $country->Code;
        echo " - ";
        echo number_format($country->Population, 0, ',', ' ');
        echo "<br>";
    }
?>
```

Opdracht 03c

> Maak de view *overzicht* zoals hierboven is aangegeven.

Het overzicht is bijna niet geformatteerd.

image-1614627165703.png
Image not found or type unknown

Weet je nog hoe een HTML table er uit ziet?

```
<table>

 <tr>
    <td> .. </td>
      <td> .. </td>
      <td> .. </td>
   </tr>

    <tr>
    <td> .. </td>
      <td> .. </td>
      <td> .. </td>
    </tr>

</table>
```

Hierboven staat een skelet van een table met twee rijen en drie kolommen.

Vind je dit lastig? Kijk dan de video.

> Verander de /view/country/overzicht nu zo dat het netjes in een table wordt gezet. De output komt er als volgt uit te zien:
>
> image-1614627534845.png
> Image not found or type unknown
> De kolommen staan nu dus netjes onder elkaar.

Je hebt nu zelf een view opgebouwd zonder gebruik te maken van de Gridview widget. Het kost meer tijd en moeite om zelf een overzicht zonder Gridview widget te maken, maar je hebt wel veel meer controle over hoe jouw overzicht er uit komt te zien.

Inleveren

- Jouw aangepaste en werkende view/country/overzicht.php

--

04 Overzicht van kleinste landen in Europa

Inleiding

We gaan verder met het overzicht dat we in de vorige les hebben gemaakt.

We gaan nu onze eigen query's toe voegen en maken een gesorteerd overzicht van alle kleine Europese landen.

Wat leren we?

- we leren hoe je met Yii op vershillende manieren data uit de database kan halen
- we leren hoe je zelf met Yii de gegevens gesorteerd kan afdrukken.
- we leren hoe we via een query een selectie kunnen maken.

Sorteren en selecteren

Via de Yii Gridview widget hebben we geleerd dat we eenvoudig kunnen sorteren en selecteren. De Yii Gridview heeft ook beperkingen. Je kunt bijvoorbeeld niet alle landen zoeken met *meer dan* X inwoners. Soms wil je ook dat de gebruiker zelf niet kan selecteren. Hij krijgt dan niet alle regels uit de tabel te zien, maar bijvoorbeeld alleen de regels met de datum van vandaag.

We gaan het sorteren en selecteren via de controller coderen.

Kijk nog eens naar de *CountryController* bij de *actionOverzicht*, daar staat:

```
$countries=Country::find()->all();
```

Dit statement kijkt naar het object *Country* dat in het model is gemaakt en zoekt dan naar alle regels uit dit object. Dit object wordt in het model gekoppeld aan de tabel *country*.

Yii vertaald dit naar de query

```
SELECT * FROM country
```

We kunnen nog veel meer, kijk maar eens naar het volgende uitgebreide voorbeeld:

`$countries = country::find()->where(['Continent' => 'Europe'])->orderBy(['Name' => SORT_ASC])->all();`

Hier worden alle *countries* geselecteerd met *Continent=Europe*, gesorteerd op *Name.*

Dit is een soort Yii query; een query in Yii-formaat.

Het SQL-statement zou er zo uit zien:

`SELECT * FROM country WHERE Continent = 'Europe' ORDER BY Name ASC`

Vind je een query eenvoudiger? OK dat kan ook.

```
$sql=SELECT * FROM country WHERE Continent = 'Europe' ORDER BY Name ASC";
$result = Yii::$app->db->createCommand($sql)->queryOne();

// Dit is hetzelfde als

$countries = country::find()->where(['Continent' => 'Europe'])->orderBy(['Name' => SORT_ASC])->all();
```

Op deze manier kun je dus via de code in de controller een selectie maken en de sortering aanpassen. In de volgende opdracht komt dit terug.

Opdracht 4a

- Maar een nieuw menu-item en noem dit *Overzicht Europe*.
- Maak een nieuwe function *actionOverzichtEurope,* zet deze in de juiste controller en koppel deze aan het menu-item.

  (let op de vertaling naar de routering; in de routering staat nooit een hoofdletter en woorden worden gescheiden door een streepje "-" , zie les 2 over routing).
- Laat in dit overzicht alle landen van Europa zien. Gebruik hiervoor het stukje code dat hierboven staat ($countries = country::find()->where.....).
- Druk de volgende kolommen af:

  Name en *Surface Area*

- Zet de twee kolommen netjes onder elkaar
- Formateer de Surface Aera netjes met spatie tussen de duizendtallen en lijn ze recht uit.
- Maak een header die **bold** is.
- Sorteer het overzicht op Surface Area van grootste land naar kleinste.

  Het overzicht komt er dus zo uit te zien:

image-1614629726011.png Image not found or type unknown

Je hebt geleerd dat je een overzicht kan maken via een (Yii-)query. Met de query kan je de selectie en sortering aanpassingen. De query wordt geplaatst in de *controller* en maakt gebruik van het model. In dit voorbeeld het model van *country*.

Inleveren

- Jouw aangepaste en werkende view/country/overzicht.php in tabel vorm, en
- Een schermafdruk yii-04a-jouw-naam van jouw gehele browser waarin je je overzicht laat zien.

Opdracht 4b

> Pas het overzicht van de vorige opdracht aan zodat je alleen alle kleine Europese landen ziet.
>
> Een land is klein als `SurfaceArea < 100000`
>
> Pas hiervoor de controller aan.
>
> Je hebt dus te maken met **twee** zoekcriteria.
>
> Zoek zelf op internet uit hoe je in een 'Yii-query' twee zoekcriteria kan combineren met *andWhere()* of gebruik de query methode om met Yii data uit de database te halen.

Inleveren

- Jouw aangepaste en werkende countryController.php

--

05 1:1 Relaties, hoofdsteden

In deze les gaan we bekijken hoe je relaties legt tussen tabellen en hoe je informatie uit andere tabellen kan afdrukken in je eigen view.

Wat leren we?

- we kijken naar relaties in tabellen met primary key en foreign key
- we leren hoe je een 1:N relatie tussen twee tabellen in Yii kan definiëren.
- we leren hoe je met behulp van een relatie gegevens bij een overzicht uit een andere tabel kan afdrukken.
- we leren hoe je met behulp van een relatie vanuit een overzicht naar een andere overzicht kan springen.

Inleiding

Een relatie in de view wordt altijd gelegd tussen één regel (row/record) van één tabel naar één of meer rijen in een andere tabel.

Omdat de relatie vanuit één regel wordt gelegd, is er dus altijd sprake van een 1:1 of 1:N relatie.

We gaan beide relaties apart bespreken en uittesten met voorbeelden.

Als uitgangspunt nemen onze eigen view die we in de vorige les hebben gemaakt. We gaan een kolom toevoegen waarin de hoofdstad van het land wordt getoond.

image-1615887892142.png
Image not found or type unknown

1:1 relatie

Bekijk de *World* database. In de *country* tabel staat de kolom *Capital*, dit is een foreign key die verwijst naar de primary key *ID* in de *city* table.

image-1615888192933.png
Image not found or type unknown

Vanuit de country table kun je via de foreign key *Capital* verwijzen naar de Name in de gekoppelde tabel city.

```
$country->hoofdstad->Name;
```

Met dit statement verwijs je vanuit de *country* tabel, via de relatie *hoofdstad* naar de N*ame* (naam) van de hoofdstad.

De relatie *hoofdstad* bestaat nog niet.

De relatie hoofdstad moet nog worden gecodeerd. Dit doen we in de *model* file van Country.

```
// models/Country.php


public function getHoofdstad()
{
    return $this->hasOne(City::className(), ['ID' => 'Capital']);
}
```

De functie *getHoofdstad* vertel jij Yii hoe de relatie tussen de tabel *Country* en *City* in elkaar zit.

Een relatie begint in Yii altijd met *get* gevolgd door de naam van de relatie. In dit geval is de naam van de relatie hoofdstad. De function heet daarom *getHoofdstad*.

Het is een 1:1 relatie; elke country heeft één hoofdstad. Dat vertellen we met *hasOne* op regel 5. Verder plaatsen we op regel 5 de primary key (*ID*) en de foreign key (*Capital).*

Een 1:1 relatie wordt dus op de volgende manier gemaakt:

```
 public function get<RELATIE NAAM>()
 {
    return $this-><RELATIE SOORT>(<RIGHT TABEL>::className(),
        ['<PRIMARY KEY RIGHT TABLE>' => '<FOREIGN KEY LEFT TABLE>']);
 }
```

In een query zou de relatie er als volgt uit zien:

```
SELECT *
FROM Country
LEFT JOIN City
ON Country.Capital = City.ID
```

De relatie is gemaakt in het Model van Country en is dan ook alleen vanuit Country Beschikbaar. Je kunt in een view van Country dan gegevens uit de relatie afdrukken door:

`$country->`**`relatie_naam`**`['`**`kolom naam`**`']`

Opdracht 05a

Lees de uitleg hierboven goed door en plaats de kolom hoofdstad in het overzicht.

Image-1615887892142.png Image not found or type unknown
Maak hiervoor twee aanpassingen:

1. 1. maak de relatie in het model van *Country*, en
   2. pas de view *overzicht* aan door daar de kolom *Hoofdstad* aan toe te voegen.

Let op! In de database zit een land zonder hoofdstad. Zolang je alleen de landen uit Europa afdrukt gaat alles goed, want in Europa hebben alle landen een hoofdstad. Als je alle landen met hun hoofdsteden van de hele wereld probeert af te drukken zul je een foutmelding krijgen omdat er dus (ten minste) één land zonder hoofdstad bestaat.

Inleveren

1. Een schermafdruk yii-05a-<jouw naam> van je country view waarin de kolommen Naam, Hoofdstad en Oppervlakte worden afgedrukt, zoals in het voorbeeld hierboven. Maak een afdruk van je gehele browserscherm.

Opdracht 05b

We gaan de Hoofdstad clickable maken. Als je naar http://localhost:8080/city gaat dan zie je het overzicht van steden. Klik op een oogje op een van de steden. Let op de URL. Je ziet http://localhost:8080/city/view?id=<NR> als url en <NR> is het ID van de stad.

Dus als je bijvoorbeeld klikt op http://localhost:8080/city/view?id=5 dan zie je informatie over Amsterdam.

Opdracht 2

Maak elke hoofdstad in het overzicht dat je bij opdracht 1 hebt gemaakt *clickable. Als je op de link klikt dan* open je het overzicht van de stad, bijvoorbeeld http://localhost:8080/city/view?id=5.

Tip: je kunt Html:a() gebruiken om een link te maken. Dit is in de vorige les uitgelegd.

Image-1615896595590.png Image not found or type unknown

Inleveren

1. Jouw country view (view/country/index.php). Plaats in je country view commentaar met jouw naam.

--

06 Relaties, land en talen

Inleding

We hebben het in de vorige les gehad over 1:1 relaties; elke land heeft precies één hoofdstad. Maar hoe zit het met een 1-op-veel (1:N) relatie?

In één land kunnen 1 of meerdere talen worden gesproken. In deze les gaan we uitvinden hoe dat werkt in Yii.

Wat leren we?

- We leren hoe we in Yii een 1:n relatie kunnen definiëren.
- We leren hoe we in Yii met de debug functie objecten kunnen inzien.

Talen in een land

In de *World* database staat een tabel *countrylanguage*. In deze tabel staat per land welke talen er worden gesproken. Per taal wordt ook aangegeven welk percentage van de bevolking deze taal spreekt.

De relatie tussen country en *countrylanguage* is 1:N. Namelijk in een land worden **1 of meer** talen gesproken.

We willen een overzicht maken dat er als volgt uit gaat zien:

In dit voorbeeld zien we dat in Indonesië 8 talen worden gesproken. We gaan ons overzicht aanpassen, zodat we per land alle talen die er worden gesproken netjes in de tabel worden weergegeven zoals hierboven in het voorbeeld is te zien.

Relatie in het model

Hoe werkt dit?

Bij de 1:1 relatie hierboven konden we via `$country->hoofdstad['Name']` de naam van de hoofdstad uit de gelinkte tabel opvragen. Als dit niet duidelijk is, lees dan het stuk over 1:1 relatie in de vorige les nog een keer goed door!

Bij een 1:N relatie werkt het hetzelfde als bij de 1:1 relatie. Dus in het model (*models/Country.php*) wordt de relatie beschreven alleen nu gebruiken we *hasMany* (in plaats van *hasOne*)

```
public function getCountrylanguages()
{
  return $this->hasMany(Countrylanguage::className(), ['CountryCode' => 'Code']);
}
```

Controleer of je het model hebt aangemaakt van *Countrylanguage* tabel.

De relatie moet je ook aanmaken, maar die wordt in het Country model aangemaakt. Elke country heeft namelijk een relatie met één of meer talen (*hasMany*).

Hoofd en kleine letters

Let even op de hoofd en kleine letters! We hebben de tabel in de database, de model file en de relatienaam. En allemaal hebben ze een andere spelling voor de hoofdletters!

Even alles op een rijtje:

tabel naam in database	Countrylanguage
model naam	CountryLanguage
relatienaam in Country.php	Countrylanguages

Relatie in de view

We kunnen nu vanuit de view (*views/country/overzicht.php*) via `$country->countryLanguages` de taal opvragen.

> Als je de Gridview widget gebruikt, dan noem je het veld (in de Widget) `country.countryLanguages`

Er is alleen één belangrijk verschil, omdat we geen 1:1 relatie hebben, maar een 1:N, krijgen we geen variabele terug maar een array. We moeten het dus met een loop door het array heen lopen.

Dit gaan we demonstreren met behulp van de debug functie die we in les 1 (optioneel) hadden toegevoegd. Als je dit nog niet hebt gedaan dan moet je dat nu doen.

Debugging

Ga nu naar je view en voeg bovenaan in de loop waarmee je het overzicht afdrukt, de regel dd($countries) toe. *dd* staat voor dump & die. Dus *dd()* laat de variabele zien en stopt met het uitvoeren van het programma.

```
<?php
....
echo "<table border=1>";
....
foreach ($countries as $country) {
...
    // hier staat al code voor eerste kolommen
...
        // voeg hier deze kolom toe
        echo "<td>";
        d($country->countrylanguages);
        echo "</td>";
...
}
...
```

Als we even de border van de tabel aanzetten (regel 3) dan is het iets makkelijker om de output te lezen.

Als je goed kijkt dan zie je in de rechter kolom een array van objecten. In deze Yii-objecten staat telkens een 'Language'.

image-1616504143444.png

Om de taal te af te drukken moeten we eerst het element array aanwijzen:

```
$country->countrylanguages[0]
$country->countrylanguages[1]
$country->countrylanguages[2]
$country->countrylanguages[3]
```

En van elk element moeten we de 'Language' hebben. Dat kun je in de dump (plaatje hierboven) goed zien.

```
$country->countrylanguages[0]->Language
$country->countrylanguages[1]->Language
```

```
$country->countrylanguages[2]->Language
$country->countrylanguages[3]->Language
```

Je weet natuurlijk niet hoeveel talen er in elk land worden gesproken. De lengte van het array is in dit voorbeeld 4, maar het kunnen er natuurlijk meer of minder zijn.

Opdracht

Let op de opdrachten hieronder volgen elkaar op. Er wordt telkens een stapje uitgevoerd.

Uiteindelijk willen we een overzicht waarin per land alle talen die daar worden gesproken, op volgorde worden afdrukt (zie voorbeeld opdracht 4d). Dit doen wel stapje-voor-stapje.

Opdracht 06a

Lees de uitleg hierboven goed door. Druk in de laatst kolom één taal af.

Image 1616504705123.png Unknown

Nu hebben we één taal per land, maar in de meeste landen wordt meer dan één taal gesproken. We gaan nu met een loop door het array $country->countrylanguages heen. Gebruik hiervoor de volgende loop.

```
foreach( $country->countrylanguages as $taal) {
// vul zelf de jusite variable in om de taal af te drukken
    echo .....;
echo " <br/>";
}
```

Inleveren

Een schermafdruk yii-06a-jouw-naam met het scherm van jouw gehele browser waarin je het landenoverzicht laat zien en waarbij je één taal afdrukt.

Opdracht 6b

Gebruik de loop zoals hierboven is aangegeven en maak het volgende overzicht.

Image-1616505080100.png Image not found or type unknown

Zorg ervoor (met CSS) dat de tabel er netjes uitgelijnd uit ziet zoals hierboven in het voorbeeld.

Je hebt nu een 1:N (één op meer) relatie gemaakt. In volgende lessen gaan we hier opnieuw mee aan de slag.

Inleveren

- Een schermafdruk yii-06b-jouw-naam.png waarin je je complete scherm laat zien waarin het landenoverzicht te zien is met alle talen zoals in het voorbeeld.

Opdracht 6c

In de Countrylanguages tabel staat naast de taal ook het percentage. Dit is het percentage van de bevolking dat deze taal spreekt. Zet dit percentage tussen haakjes achter elke taal. Voorbeeld:

Image-1616505413868.png Image not found or type unknown

Inleveren

1.
   - Een schermafdruk yii-06c-jouw-naam.png waarin je je complete scherm laat zien waarin het landenoverzicht te zien is met talen en percentages zoals in het voorbeeld.

Sorteren

Jammer dat de percentage niet gesorteerd zijn!

Pas de relatie aan in het Country model

```
public function getCountrylanguages()
{
   return $this->hasMany(CountryLanguage::className(), ['CountryCode' => 'Code'])
   ->orderBy(['Countrylanguage.Percentage' => SORT_ASC]);
}
```

Regel 4 is erbij gekomen. let op de ; !
Als je nu het overzicht opnieuw toont dan zie je dat de talen nu oplopend (van laag naar hoog) zijn gesorteerd.

Opdracht 6d

Sorteer de talen op percentage aflopend van hoog naar laag. Op die manier komt de meest gesproken van een land bovenaan.

Image-1616500111435.png Image not found or type unknown
Tip: gebruik Google om te zoeken hoe je de relatie in het model verandert zodat je **aflopend** kunt sorteren.

Nog een tip:

> Als je in de database de relaties (via SQL) goed definieert dan worden de relaties in Yii door de model generator automatisch gemaakt. Het aanmaken van relaties in de databases valt buiten deze lessen en kan lastig zijn. Het is dus belangrijk dat je snapt hoe de relaties tussen tabellen in Yii worden gemaakt.

Inleveren

1. Een schermafdruk yii-06d-jouw-naam.png waarin je je complete scherm laat zien waarin het landenoverzicht te zien is zoals in het voorbeeld.
2. De code van de view van country. Voeg in jouw code commentaar toe waarin je jouw naam zet.

--