

6, Relaties 1:1 en 1:N

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

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 el land wordt getoond.

[image-1615887892142.png](#)

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](#)

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 verwijst je vanuit de *country* tabel, via de relatie *hoofdstad* naar de *Name* (naam) van de hoofdstad.

Nu moet de relatie *hoofdstad* alleen nog worden geprogrammeerd. 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 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:

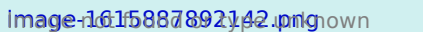
```
SELECT *
FROM Country
LEFT JOIN City
ON Country.Capital = City.ID
```

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 1

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


Maak hiervoor twee aanpassingen:

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.

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](#)

1:N relatie

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 uitgaat zien:

[image-1594550267859.png](#)

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.

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 hierboven 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*) is de relatie beschreven:

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

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

Let even op de hoofd en kleine letters!

tabel naam in database	Countrylanguage
------------------------	-----------------

model naam	CountryLanguage
relatiennaam in Country.php	Countrylanguages

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 afdruckt, 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 3

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

image-1616504705123.png

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 juiste variable in om de taal af te drukken
    echo .....;
    echo " <br/>";
}
```

Opdracht 4

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

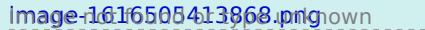
Image 1616505080100.png

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.

Opdracht 5

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

Jammer dat de percentage niet gesorteerd zijn!

Pas de relatie aan in het Country model

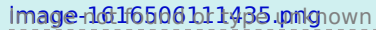
```
public function getCountrylanguages()
{
    return $this->hasMany(CountryLanguage::className(), ['CountryCode' => 'Code'])
        ->orderBy(['Countrylanguage.Percentage' => SORT_ASC]);
}
```

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 6

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

Image-1616506111435.png

Tip: gebruik Google om te zoeken hoe je de relatie in het model verandert zodat je aflopend kunt sorteren.

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.

Revision #57

Created 10 July 2020 07:55:19 by Max

Updated 25 November 2021 12:46:38 by Max