

# 09, Form en drop down

*We gaan onze koffie applicatie uitbreiden.*

*Om het invoeren van gegevens eenvoudiger te maken gaan we drop down menu's toevoegen.*

*We gaan naar twee soorten drop-downs kijken; die met vaste waarden en die met dynamische waarden (uit de database).*

*Bij het invoeren van een bestelling gaan we allereerst de status via een drop down invoeren.*

*Daarna gaan we waarde voor de medewerker ook uit een drop down halen. Deze waarde moet uit de database komen.*

## Drop down in Bestelling

We gaan kijken naar de bestelling tabel en CRUD.

Als we de status in de database als een enum hebben aangemaakt dan weet Yii dat de status uit beperkt aantal waarden kan hebben en als het goed is maakt Yii dan vanzelf een drop down in het form.

Klopt dat heb jij een drop down?

Indien niet, dan kun je de code aanpassen in het form aanpassen.

In de view `_form.php` van Bestellingen moet de volgende regel staan:

```
<?= $form->field($model, 'status')->dropDownList([ 'besteld' => 'Besteld', 'klaar' => 'Klaar', 'betaald' => 'Betaald', '' => '', ], ['prompt' => '']) ?>
```

Deze regel laat een drop down menu in het formulier zien. Kijk eens goed naar de parameter van `dropDownList`.

```
[ 'besteld' => 'Besteld', 'klaar' => 'Klaar', 'betaald' => 'Betaald', '' => '', ], ['prompt' => '']
```

De eerste parameter is een associative array waarin elk element bestaat uit een key en een value.

De key is de waarde die het element uit het form krijgt (de value) en deze waarde wordt door Yii in de database gezet. De key komt dus overeen met de waarde in de database. De value van het associatieve array is wat de gebruiker op het scherm ziet.

Deze waarde kan je dus veranderen. Je kunt dus bijvoorbeeld 'betaald' veranderen in 'afgerekend'. Het enige dat dan gebeurt is dat je iets ander op het scherm ziet.

In dit voorbeeld is de key en de value gelijk (de waarde op het scherm is hetzelfde als de waarde in de database).

Dus samengevat, de key is de waarde in de database en de value is wat de gebruiker op het scherm ziet.

Stel we willen in het form van bestelling de medewerker veranderen. Dan moeten we dus de *foreign key* die naar medewerker verwijst veranderen. Het juiste id van de medewerker moet in de tabel bestelling worden ingevuld.

De waarde wordt dus het id, maar dat is niet wat je de gebruiker wilt laten zien.

Dus als we voor de medewerkers een drop down willen maken dan hebben we een associative array nodig dat er zo uit ziet:

```
[ '1'=> 'Ayoub', '2'=> 'Brahim', '3'=> 'Carla', '4'=> 'Diego', '5'=> 'Eisa' ]
```

De keys zijn de id's die als foreign keys in de bestelling tabel staan en de namen zijn de namen van de medewerkers.

## Opdracht 8a

Maak nu een statische drop down met de waarden zoals in het voorbeeld (Ayoub, Brahim, Carla, ....).

De waarden worden dus (nog) niet uit de database gehaald.

### Inleveren

1. Schermafdruk yii-08a-jouw-naam met het form waarin je de drop down (opengeklapt) laat zien. Maak een schermafdruk van je gehele browser.

## Opdracht 8b, Dynamische drop down

In opdracht 8a hebben we een drop down van medewerkers gemaakt, maar we willen de lijst van medewerkers natuurlijk uit de database halen.

*In deze opdracht leg ik stap-voor-stap uit wat je moet doen omdat oor elkaar te krijgen. Lees alles aandachtig door en sla geen stappen over!*

Data uit de database halen doen we in de controller.

Open de `BestellingController`. Vanuit deze code wordt de create view aangeroepen en vanuit de create view wordt de `_form.php` aangeroepen. Waarom dit in twee stappen gaat leggen we later uit.

We veranderen de code in `Bestelling controller`:

```
use app\models\Medewerker;

..

..

public function actionCreate()
{
    $model = new Bestelling();
    $medewerkers = Medewerker::find()->all();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    }

    return $this->render('create', [
        'model' => $model,
        'medewerkers' => $medewerkers
    ]);
}
```

De eerste regel `use app\models\Medewerker;` zetten we bovenaan in de `BestellingController`. Hiermee vertellen we Yii dat vanuit de *BestellingController* gebruik willen gaan maken van het model *Medewerker*.

In de function *actionCreate* die wordt aangeroepen als we een nieuwe bestelling willen maken, voegen we een object toe waarin alle medewerkers zitten. In regel 9 halen we alle medewerkers op en in regel 17 sturen we het resultaat naar de view `create.php` (van *bestelling*).

We kunnen de medewerkers ook ophalen via een sql-query. Je mag regel 9 ook vervangen in de volgende twee regels:

```
// dit is hetzelfde als
// $medewerkers = Medewerker::find()->all();
$sql="select * from medewerker";
$medewerkers = Yii::$app->db->createCommand($sql)->queryAll();
```

In de view create.php van bestelling passen we het laatste code aan

```
<?= $this->render('_form', [
    'model' => $model,
    'medewerkers' => $medewerkers
]) ?>
```

Hiermee geven we het object medewerkers weer door aan de view *\_form* van bestelling.

In het view *\_form* zetten we nu bovenaan in het PHP-gedeelte.

```
dd($medewerkers);
```

Dit is de debugfunctie en hiermee controleren we of we inderdaad alle medewerkers naar de view hebben gestuurd.

We hebben nu als het goed is een lijst van medewerkers in de *\_form* maar we moeten het ombouwen naar een associatieve array.

Pas hiervoor de code aan in de view *\_form* van bestelling.

```
<?php

use yii\helpers\Html;
use yii\widgets\ActiveForm;
use yii\helpers\ArrayHelper;

$medewerkerList = ArrayHelper::map($medewerkers,'id','naam');
dd($medewerkerList);
?>

...

...

...
```

Deze code laat het eerst stukje van de view *\_form* zien.

Met de functie `ArrayHelper::map` zetten we het object `$medewerkers` om in een associatieve array. Met `dd()` laten we dat zien. Probeer maar! Als het goed is, zie je het volgende:

```
[
  1 => 'Ayoub'
  2 => 'Brahim'
  3 => 'Carla'
  4 => 'Diego'
  5 => 'Eisa'
]
```

En dit is precies wat we nodig hebben om de Drop Down te maken.

Pas de regel in de view `_form` van bestelling waarin de user het id van de medewerker moet intypen aan. Verander deze regel in:

```
<?= $form->field($model, 'medewerker_id')->dropDownList($medewerkerList, ['prompt' => ''])-
>label('Medwerker') ?>
```

Als het goed is heb je hiermee een werkend menu gekregen.

image-1616605065315.png

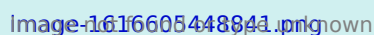
Gelukt? Wordt de lijst van medewerkers in de drop down uit de database gehaald?

## Inleveren

1. schermafdruck yii-08b-jouw-naam met het form waarin je de drop down van de medewerkers (opengeklapt) laat zien. Maak een schermafdruck van je gehele browser.

## Opdracht 8c

Maak nu een drop down voor de bestelling zodat de medewerker uit een lijstje van koffiesoorten kan kiezen bij het opnemen van de bestelling.

image-1616605448841.png

Gebruik hiervoor het stappenplan dat hieronder is beschreven.

## Het stappenplan voor een drop down

Schrijf eerst het volgende op:

In welk model komt de drop down? Dit noemen we de target.  
(in opdracht 2 hierboven is dat *Bestelling*)

Waar komt de informatie voor de Drop Down uit? Dit noemen we de source.  
(in opdracht 2 hierboven is dat *Menu*)

## Voer nu de stappen 1-6 uit

### 1. Zet in de target controller welke source model we willen gebruiken.

```
// in controller van target  
use app\models\<SourceModelnaam>;
```

### 2. Haal in de target controller in de createAction() de informatie uit de source op.

```
// sourceName bedenk jij zelf en daarin staat de informatie die in de drop down moet komen.  
$sourceName = source::find()->all();
```

### 3. Geef vanuit de target controller in de createAction() de informatie door aan de view.

```
return $this->render('create', [  
    'sourceName' => $sourceName,  
    ....  
]);
```

### 4. Open de view create in de target (view) en geef de informatie door aan \_form view.

```
<?= $this->render('_form', [  
    'sourceName' => $sourceName,  
    ...  
]);
```

### 5. Open de view \_form in de target (view) en zet de informatie om in een associative array

```
// in dit voorbeeld worden id en naam gebruikt:  
// id is het veld dat moet worden opgeslagen  
// naam is het veld dat in de drop te zien is  
$sourceNameList = ArrayHelper::map($sourceName,'id','naam');
```

### 6. Pas het invoerveld aan in de \_form van de target (view).

```
// veld_in_de_db is het veld van de target dat in dit invoerveld wordt weggeschreven.  
<?= $form->field($model, 'veld_in_de_db')->dropDownList($$sourceNameList, ['prompt' => '']) ?>
```

Als je alles correct gedaan hebt dan moet je nog 1 regel aanpassen in de code om de functionaliteit te gebruiken. Yii controleert automatisch of alle databasekolommen zijn ingevuld in het formulier. Dit controleren wordt *validatie* genoemd. Deze regels staan in het model. Omdat je geen Bestelling ID opgeeft bij het maken van een bestelling moet je deze requirement verwijderen op regel 33 van model Bestelling.

```
return [  
    ['naam', 'medewerker_id', 'menu_id', 'status'], 'required'],  
    ...  
    ...  
]
```

## Inleveren

1. schermafdruck yii-08c-jouw-naam met het form waarin je de drop down van de producten (opengeklapt) laat zien. Maak een schermafdruck van je gehele browser.
2. de *BestellingController.php* en zet jouw naam bij de aanpassingen die je hebt gedaan.

## Opdracht 8d

De insert en update lijken op elkaar en gebruiken hetzelfde form (*\_form.php*).

We hebben nu de insert aangepast, maar de update nog niet. Controleer maar! We moeten nu de update dus ook nog aanpassen.

### aanpassen update view

Weet je nog dat de *-update* view niet rechtstreeks door de *actionCreate* van de *BestellingController* wordt aangeroepen?

Hoe zit dat nu?

[image-1616610519616.png](#)

De *actionCreate* en *actionUpdate* vanuit de controller gaan bieden naar hun 'eigen' view maar via deze eigen view komen beide op hetzelfde form uit. Dat komt omdat een *update* en *create*

hetzelfde form gebruiken.

Als we nu dus een update uitvoeren van de bestelling (pennetje vanuit de gridview) dan krijgen we een foutmelding. Dat komt omdat de form verwacht dat er gegevens voor de drop down worden meegestuurd.

Om dit te fixen kopiëren we de code in de controller die we in de *actionCreate* hebben gemaakt dus naar de *actionUpdate*.

Deze twee functies zijn bijna hetzelfde. Bij de create wordt alleen een nieuw object gemaakt en bij de update wordt een bestaand object ingeladen.

Nu moeten we de objecten nog doorgeven van de *update* view naar de *\_form* view. Net zoals we dat ook deden bij de *create*.

Zorg ervoor dat de update weer werkt en dat de objecten op de juiste manier worden doorgegeven. Lees hiervoor de uitleg die hierboven staat.

Pas de *actionUpdate* in de *BestellingController* aan.

Pas de update view van de *BestellingController* aan

Zorg dat de update van een Bestelling goed werkt..

## Inleveren

1. de *BestellingController.php* en zet jouw naam bij de aanpassingen die je hebt gedaan.

## Opdracht 8e

### De final touch

Laten we nog wat labels aanpassen. Dit zijn de teksten die boven de invoervelden in het form staan.



Pas de labels in het form aan zodat er Medewerker, Klantnaam, Bestelling en Status Bestelling komt te staan:

[Image-1616611684520.png](#)

Zoek zelf uit hoe dit moet. Tip gebruik de zoektermen:

**How to change label text of the ActiveForm?**

# Inleveren

1. schermafdruck yii-08e-jouw-naam met het bestellingen form waarin je de aangepaste labels laat zien. Maak een schermafdruck van je gehele browser.

--

---

Revision #1

Created 22 June 2022 12:49:17 by Max

Updated 22 June 2022 12:49:17 by Max