Archive

Oude lessen

- 5, Zelf een view maken
- 6, De tweede web app
- 7, Sorteren, selectie maken en menu's
- 8, Login / rollen
- Opdrachten

5, Zelf een view maken

DEZE EN VOLGENDE LESSEN WORDEN NOG AANGEPAST

Tot nu hebben we gebruik gemaakt van de GridView om data te laten zien (localhost:8080/country/index) - in deze les gaan we onze eigen view maken. Het maken van een eigen view kost meer tijd, maar je kunt meer zaken zelf aanpassen zoals de weergave (formattering) van getallen.

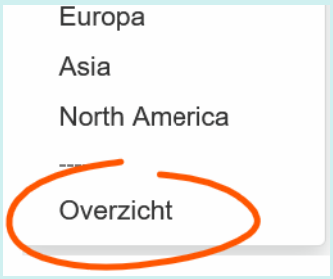
Menu - Controller - View

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

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

Opdracht 1

Maar een menu item *overzicht* onder country dat naar */country/overzicht* gaat.



Probeer het menu uit. Wat zie je en kun je verklaren wat je ziet?

Juist.... je krijgt een foutmelding want de routering verwijst naar de method/function actionOverzicht in de controller CountryController en die bestaat niet!

We gaan dus een stukje code toevoegen aan onze CountryController.

```
use yii\data\Pagination;


..
..


public function actionOverzicht()
```

```
{
    // 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 2

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

Test opnieuw wat er gebeurt als je het menu *overzicht* selecteert

What? Weer 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 3

Maak de view *overzicht* zoals hierboven is aangegeven.

Het overzicht is bijna niet geformatteerd.

Aruba - ABW - 103 000
Afghanistan - AFG - 22 720 000
Angola - AGO - 12 878 000
Anguilla - AIA - 8 000
Albania - ALB - 3 401 200
Andorra - AND - 78 000
Netherlands Antilles - ANT - 217 000
United Arab Emirates - ARE - 2 441 000

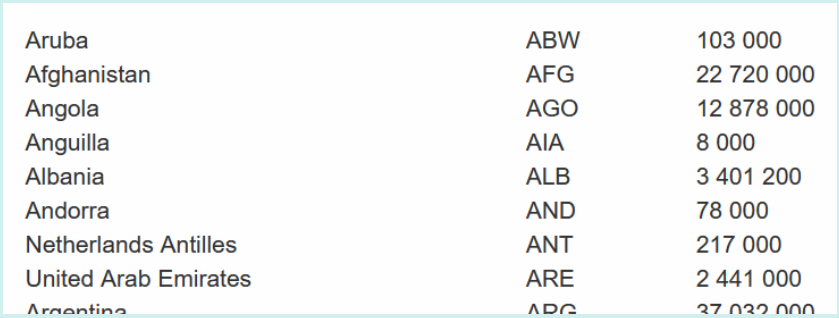
Weet je nog hoe een HTML table er uit ziet?

```
<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.

Opdracht 4

Verander de /view/country/overzicht nu zo dat het netjes in een table wordt gezet. De output komt er als volgt uit te zien:



Aruba	ABW	103 000
Afghanistan	AFG	22 720 000
Angola	AGO	12 878 000
Anguilla	AIA	8 000
Albania	ALB	3 401 200
Andorra	AND	78 000
Netherlands Antilles	ANT	217 000
United Arab Emirates	ARE	2 441 000
Argentina	ARG	37 032 000

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 een overzicht te maken maar je hebt wel veel meer controle over hoe jouw overzicht er uit komt te zien.

Sorteren en selecteren

We gaan nu een paar functies bouwen die in de Gridview widget al ingebouwd zijn. Dit gaat om sorteren en selecteren.

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

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

Dit statement zoekt gaat naar het object country, dit staat in het model en zoekt daar alle regels (rows) in op. Yii vertaald dit nar de query

```
SELECT * FROM country
```

Maar 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* geslecteerd met *Continent=Europe*, gesorteerd op *Name.* Het sql statement zou er zo uit zien:

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

Op deze manier kun je dus via de code in de controller een selectie maken en de sortering aanpassen.

Opdracht 5

- Maar een nieuw menu item en noem dit *Overzicht Europe*.
- Maak een nieuw actionOverzichtEurope 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 - , zie les over routing).
- Laat in dit overzicht alle landen van Europa zien, en 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 vangrootste land naar kleinste.

  Het overzicht kot er dus zo uit te zien:

Naam	Oppervlakte
Russian Federation	17 075 400
Ukraine	603 700
France	551 500
Spain	505 992
Sweden	449 964

Lees het commentaar in deze controller, zodat je begrijpt wat de verschillende blokken code doen.

```
use yii\data\Pagination;


..
..

public function actionOverzicht()
{
   $this->view->title = 'Search Countries';
```

```
    // dit is de query, dit is te vergelijken met select * from Country
    $query=Country::find();

    // hier worden bepaald hoeveel regels er op één pagina worden getoond en hoeveel regels er in totaal zijn
    $pagination = new Pagination([
        'defaultPageSize' => 5,
        'totalCount' => $query->count(),
    ]);

    // hier wordt de query uitgevoerd, er wordt gesorteerd en er wordt slecht één pagina gelijktijdig binnen gehaald.
    $countries = $query->orderBy('name')
        ->offset($pagination->offset)
        ->limit($pagination->limit)
        ->all();

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

View

In Yii zit standaard bootstrap(3) geïnstalleerd. In de volgende view wordt gebruik gemaakt van bootstrap(3) styles.

Dan zetten we in de file **views/country/overzicht.php** het volgende

```
<?php
use yii\helpers\Html;
use yii\widgets\LinkPager;
?>

<div class="card">
  <div class="card-body">
  <h3 class="card-title">Countries</h5>

    <table class="table" style="width: 70rem;" border=0>
```

```
    <tr>
      <th scope="col" style="width: 10rem;">Land</th>
      <th scope="col" style="width: 5rem;">Code</th>
      <th scope="col" style="width: 10rem;">Hoofdstad</th>
      <th scope="col" style="text-align: right;width: 10rem;">Inwoners</th>

    </tr>
    <?php foreach ($countries as $country): ?>
     <tr>
      <td><?= $country->Name ?></td>
      <td><?= $country->Code ?></td>
      <td><?php
       if ($country->capital) { // note: f.e. Antartica has no Capital
        echo $country->capital->Name;
       } else {
        echo "";
       }
      ?></td>
      <td style="text-align: right"><?= number_format($country->Population, 0, ',', ' '); ?></td>
     </tr>
    <?php endforeach; ?>
   </table>

  </div>
</div>

<?= LinkPager::widget(['pagination' => $pagination]) ?>
```

In deze HTML-code onderscheiden we enkele zaken:

1. We hebben een table waarin we alles afdrukken.
2. We hebben een table header waarin we de kopjes van de verschillende kolommen afdrukken.
3. We hebben een loop waarmee we door het *country* object heen itereren en alle countries afdruken.
4. We hebben een widget die ervoor zorgt dat we naar de volgende pagina kunnen springen.

Opdrachten

1. schrijf van de vier bovenstaande punten op waar je dit precies vind in de code, noteer de regels waarop het betrekking heeft. Bijvoorbeeld: 1 heeft betrekking op regel 10-32

2. wat (welk stukje code) zorgt ervoor dat het inwoneraantal netjes geformatteerd wordt afgedrukt?

3. In deze code wordt de naam van de hoofdstad afgedrukt (en niet meer de foreign key zoals in ons eerdere overzicht). Op welke regel wordt de naam van de hoofdstad afgedrukt?

4. Er staan 'landen' in de database die geen hoofdstad hebben zoals bijvoorbeeld Antartica. Verander de code, zodat er in deze gevallen een streepje (-) wordt afgedrukt.

Opmerking:

In de view zie je php-code staan tussen `<?php ?>` en tussen `<?= ?>` Dit is bijna hetzelfde: `<?= $a ?>` is een verkorte notering voor `<?php echo $a ?>`

--

6, De tweede web app

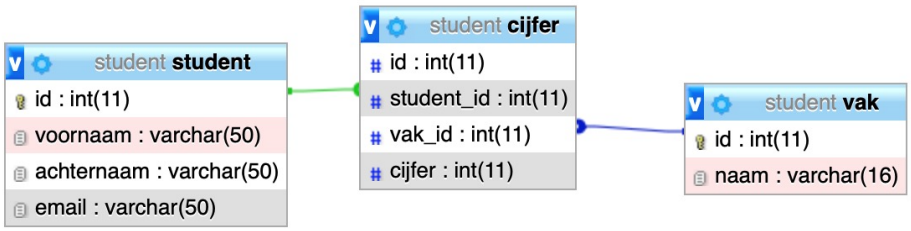
In deze les gaan we oefenen met alles wat we tot nu hebben geleerd. We zullen een database met een N:M relatie gebruiken en we zullen zien dat dat niet echt anders is als een 1:N relatie.

Stap 1, Database

In deze stap gaan we een nieuwe database maken.

Maak een nieuwe database; ga naar deze pagina en installeer de student database.

Je hebt nu een database student met drie tabellen; *student*, *vak* en een koppeltabel *cijfer.* Een student heeft meerdere vakken een één vak kan door meerdere studenten worden gevolgd.



Stap 2, Nieuw Yii project

In deze stap gaan we een nieuw Yii project maken.

```
composer create-project --prefer-dist yiisoft/yii2-app-basic student
```

Pas de **config/web.php** en de **config/db.php** aan (kijk naar naar de eerste les als je niet meer weet hoe dit precies moet).

Tip: vergeet niet de localhost in de d.php te veranderen in 127.0.01

Start de Yii PHP server.

Stap 2, Models

In deze stap gan we de models maken.

Maak met behulp van Gii zoals we dat in de eerste les hebben gedaan een model van alle *drie* de tabellen. Maak daarna met Gii een CRUD van de student tabel. Kijk nog een keer naar de eerste les

als je niet meer weet hoe dit precies moet.

Stap 3, CRUDs

In deze stap gaan we de CRUDs generen

Maak met Gii een CRUD van de student-tabel, de cijfer-tabel en de vak-tabel. Kijk nog een keer naar de eerste les als je niet meer weet hoe dit precies moet.

Stap 4, View

In deze stap gaan we onze eigen view maken waarin we per student alle (behaalde) cijfers laten zien.

In deze stap zien we een relatie, namelijk een 1:N relatie. Elke student heeft immers 0,1, of meer vakken.

In de vorige lessen hebben we hier mee geoefend, weet je nog? In Eén land werden er 1 of meer talen gesproken. In dat overzicht lieten we alle talen per land zien en nu laten we alle vakken per student zien. Technisch werkt dit op dezelfde manier.

image-1594677185236.png
Image not found or type unknown

Stap 5, Update Cijfer

In deze stap gaan we een het automatisch gegenereerde form waarmee we een cijfer kunnen aanpassen veranderen.

In de standaard CRUD van de tabel cijfer kun je cijfers updaten. Ga naar http://localhost:8080/cijfer en druk op edit (rechts). Je komt nu in het update-scherm van de tabel cijfer.

image-1594748309104.png
Image not found or type unknown

Nu zie je alleen de gegevens uit de tabel *cijfer*: je zou liever de naam van de student en de naam van het vak zien. Dat gaan we fixen!

Allereerst kijken we naar de URL: http://localhost:8080/cijfer/update?id=0

Om te bepalen welke view we moeten aanpassen volgend de routeringsregels. We gaan naar de *CijferController.php* file en zoeken naar de *actionUpdate*. We zien dat aan het einde van deze function (method) de view update wordt aangeroepen:

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

De update view staat in *views/cijfer/update.php*, maar als je die opent, dan zie je heel weinig code. Dat komt, omdat de *update* en *insert* heel veel op elkaar lijken. Het invulscherm (form) is namelijk

voor beide hetzelfde. En beide views (*update.php* en *insert.php*) roepen dus beiden *_form.php* aan. Als we in *_form.php* wijzigingen maken, dan gelden die dus voor update en voor insert.

We willen dat het scherm er ongeveer zo gaat uitzien:

image-1594750980851.png
Image not found or type unknown

We kunnen in dit form alleen het cijfer aanpassen; de naam van de student en de naam van het vak kunnen hier niet worden aangepast. Het cijfer staat overigens als 100-tal in de database. In dit voorbeeld staat 85 en dat staat voor een 8.5.

<video uitleg - met gebruik van dd>

Stap 6, De link tussen overzicht en update

Nu moeten we een link maken tussen ons overzichtsscherm en de juisten update pagina. Het moet je moet op het vak/cijfer van een student kunnen klikken en dan moet jet juiste update scherm worden getoond.

Als we naar de link kijken die wordt gebruikt om een cijfer te updaten dan zien we dat daar een id wordt meegegeven. Dit is het ID van de tabel cijfer, de primary key. Om dus naar het juiste update-scherm te kunnen springen hebben we het ID van de *cijfer* tabel dus nodig.

Ons overzicht *overzicht.php* staat in de student directory.

Om zeker te weten dat we het juiste ID gebruiken, maken we een tijdelijke aanpassing in het overzicht en drukken het ID van de cijfer tabel af.

Opdracht (1): druk na het *cijfer* in student/overzicht.php het id af van de koppeltabel waarin het cijfer voor het vak staat. Zie voorbeeld hieronder.

image-1594752066080.png
Image not found or type unknown

We hebben nu alle informatie om van het cijfer een hyperlink te maken naar het juiste update scherm.

Opdracht (2): maak van het cijfer dat wordt afgedrukt in student/overzicht.php een hyperlink (<a href=....), zodat als je op het cijfer klikt, het juiste en zonet gemaakte update scherm verschijnt.

Het skelet van de code om de hyperlink af te drukken:

```
<?php $url='/cijfer/update?id='.XXXXXX; ?>
<a href=<?=$url?> >
  <?= number_format(($cijfervak->cijfer)/10,1) ?>
</a>
```

Om het overzichtelijk te houden maak je eerst de variabele $url met de juisten URL en maak je daarna de hyperlink. De XXXXXX moet nog worden vervangen door het juiste id (van de tabel vak). Deze heb je bij de vorige opdracht al laten afdrukken.

--

7, Sorteren, selectie maken en menu's

In deze les....

Stap 1, aanmaken nieuwe overzicht

We gaan in de *cijferController*, een nieuwe function (method) *actionOverzicht* maken. Als basis kun je de *actionOverzicht* uit *studentController* kopiëren. Je moet de nieuwe functie natuurlijk wel aanpassen!

We gaan in de view *cijfer* een nieuw *overzicht.php* aanmaken. Als basis kun je de file *overzicht.php* uit student kopiëren. Ook deze nieuw gekopieerde view moet je wel aanpassen!

We drukken de volgende vier kolommen af:

```
<?= $cijfer->student->voornaam ?>
<?= $cijfer->student->achternaam ?>
<?= $cijfer->vak->naam ?>
<?= $cijfer->cijfer ?>
```

Stap 2, conditioneel afdrukken

We gaan de cijfers conditioneel anders afdrukken. De conditie is dat we cijfers lager dan 55 in het rood gaan afdrukken. De conditie is dus *cijfer<55*.

We kunnen hiervoor de volgende code gebruiken.

```
<?php if($cijfer->cijfer < 55): ?>
  <td><font color="red"><?= number_format(($cijfer->cijfer)/10,1) ?></td>
<?php else: ?>
  <td><?= number_format(($cijfer->cijfer)/10,1) ?></td>
<?php endif; ?>
```

Op regel 1 staat de conditie, dit is bijna hetzelfde als een PHP if. Het enige verschil is dat de { is vervangen door een :
Op regel 2 wordt het cijfer door 10 gedeeld en met één decimaal in het rood afgedrukt.
Op regel 3 staat de *else*, hier wordt de } weggelaten en de { is vervangen door :

Op regel 4 wordt het cijfer net zo afgedrukt als op regel 3, alleen nu niet in het rood.
Op regel 5 staat een endif, zonder } en met een ;. De ; geeft aan dat dit het eidne is van de if.

Stap 3, sortering aanpassen

De sortering wordt geregeld vanuit de controller.

De query wordt in de controller gemaakt.

```
$query=cijfer::find()
```

Voeg hier een *orderBy* aan toe.

```
$query=cijfer::find()
  ->orderBy('cijfer');
```

Stap 3, selectie aanpassen

We gaan een selectie maken op vak, dit gebeurt ook in de controller.

We voegen een *where* toe.

```
$query=cijfer::find()
  ->where( ['vak_id' => $vak_id] )
  ->orderBy('cijfer');
```

Oops, waar komt $vak_id vandaan? Hoe weet de controller waar hij op moet selecteren?

Dat weet de controller nog niet, we zullen bij het aanroepen van controller een parameter moeten meegeven, verander de definitie van de controller in:

```
public function actionOverzicht($vak_id=1)
```

$vak_id wordt nu optioneel meegegeven en is 1 als er geen waarde wordt meegegeven.

Probeer maar, als het goed is, zie je nu alleen de vakken waarvan het *vak_id* een 1 is (en dat in *Nederlands*).

Verander de URL naar bijvoorbeeld:

```
http://localhost:8080/cijfer/overzicht/?vak_id=2
```

en je ziet alleen het vak met het *vak_id* 2 (en dat is *Engels*).

Stap 4a, extra menu in overzicht

image-1594845411296.png
Image not found or type unknown

De eenvoudigste methode om via de view een vak te selecteren is met een 'eenvoudig' Bootstrap menu. Dat kan met de volgende code:

```
<div class="row">
  <div class="col-md-2">
     <ul class="nav nav-tabs">
       <li class="dropdown">
         <a href="#" data-toggle="dropdown">Vak <span class="caret"></span></a>
         <ul class="dropdown-menu" role="menu">
           <li><a href="/cijfer/overzicht/?vak_id=1" >Nederlands</a></li>
           <li><a href="/cijfer/overzicht/?vak_id=2" >Engels</a></li>
           <li><a href="/cijfer/overzicht/?vak_id=3" >Rekenen</a></li>
           <li><a href="/cijfer/overzicht/?vak_id=4" >JavaScript</a></li>
           <li><a href="/cijfer/overzicht/?vak_id=5" >PHP</a></li>
           <li><a href="/cijfer/overzicht/?vak_id=6" >HTML</a></li>
           <li><a href="/cijfer/overzicht/?vak_id=7" >SQL/Databases</a></li>
           <li><a href="/cijfer/overzicht/?vak_id=8" >Burgerschap</a></li>
           <li><a href="/cijfer/overzicht/?vak_id=9" >IT Security</a></li>
         </ul>
       </li>
     </ul>
   </div>
</div>
```

Dit is de Bootstrap manier om een menu te maken.

Zie: https://getbootstrap.com/2.3.2/components.html#buttonDropdowns voor meer uitleg.

Op zich hoef je niet precies te weten hoe het werkt, als je maar weet hoe je dit kunt aanpassen voor je eigen menu.

Je kunt dit menu ook in de menu bar van de applicatie plaatsen, daarvoor moet je de file *views/layouts/mail.php* aanpassen.

Stap 4b, tabbed menu in overzicht

image-1594845470106.png
Image not found or type unknown

Een tabbed overzicht ziet er fraaier, maar is wat lastiger. Dit kun je op de volgende manier toevoegen in je view.

```
<ul class="nav nav-tabs">
  <?php $vakArr=['Nederlands','Engels','Rekenen', 'JavaScript',
            'PHP', 'HTML', 'SQL/DB', 'Burgerschap', 'Security'] ?>
  <?php for($i=1; $i<=9; $i++): ?>
    <?php if($i==$vak_id): ?>
      <li class="active">
    <?php else: ?>
      <li>
    <?php endif; ?>
      <a href="/cijfer/overzicht/?vak_id=<?=$i?>" ><?=$vakArr[$i-1]?></a></li>
  <?php endfor; ?>
</ul>
```

Dit is vrijwel allemaal PHP code. De vakken staan in een array en zijn in de database genummerd met de ID's 1 tot en met 9. In een loop worden alle menu items afgedrukt en wordt de juiste waarde uit het array erbij afgedrukt.

Bestudeer de code en vraag om uitleg als je dit niet begrijpt.

--

8, Login / rollen

In onze tweede web app, de student database gaan we een login maken. Als je aanlogt als beheerder dan mag je de cijfers invoeren, veranderen of deleten.

models/Users.php

```
private static $users = [
  '100' => [
      'id' => '100',
      'username' => 'admin',
      'password' => 'admin',
       'authKey' => 'test100key',
       'accessToken' => '100-token',
       'role'=> 'admin',
  ],
   '101' => [
      'id' => '101',
      'username' => 'user',
      'password' => 'demo',
      'authKey' => 'test101key',
      'accessToken' => '101-token',
      'role'=> 'user',
  ],
];
```

In controller

```
public function behaviors()
{
  return [
    'access' => [
    'class' => AccessControl::className(),

    'rules' => [
          [ 'actions' => ['index','view'],
            'allow' => true,
            'roles' => ['@'] // any authenticated user, use ? for guests
```

```
          ],

          [ 'actions' => ['create','update','delete','overzicht'],
            'allow' => true,
            'roles' => ['@'],
            'matchCallback' => function ($rule, $action)
            {
              return (Yii::$app->user->identity->role == 'admin');
            }
          ],

        ],

      ],
    ];
}
```

via Init method in Controller (replaces the __construct)

```
// if user object does nit exists or it exists but is anything but admin, go to login screen
public function init() {
    if (! isset(Yii::$app->user->identity->role) || Yii::$app->user->identity->role != 'admin') {
        $this->redirect(['/site/login']);
    }
 }
```

Opdrachten

Les 1, opdracht 1

1. Bespreek met elkaar wat een framework is en probeer een omschrijving te maken.
2. Bespreek met elkaar waarom jij een framework zou willen leren gebruiken?
3. Welke (PHP) frameworks ken je?

Les 1, opdracht 2

Als jij een framework kiest om te leren, wat is dan belangrijk voor jou? Zet de volgende punten op volgorde van belangrijkheid (van meest belangrijk naar minder).

- Veel over te vinden (tutorials e.d.).
- Veel plugins voor beschikbaar.
- Wordt actief onderhouden, veel updates dus.
- Is makkelijk te leren, kun je makkelijk je examen mee halen.
- Kun je snel iets mee maken/ontwikkelen.
- Heeft heel veel mogelijkheden.
- Genereert automatisch code, die je dan kan aanpassen.
- Heeft veel ‘straight out of the box’, sorteren, selecteren, login, menu, etc.
- Is hip/modern/cool.

Huiswerk, les 1

1. Lees de Yii inleiding
2. Volg les 1 tot en met "Create New Yii project",

Je hebt dan dus XAMPP draiaen, Composer geïnstalleerd en een nieuw Yii project gemaakt.