Inhoudsopgave

[Installatie onderdelen 2](#_Toc21359735)

[XAMPP 2](#_Toc21359736)

[Composer 2](#_Toc21359737)

[Nieuw Symfony project 2](#_Toc21359738)

[PhpStorm (editor) 6](#_Toc21359739)

[GIT 8](#_Toc21359740)

[Extra symfony packages installeren 9](#_Toc21359741)

[FosUser installatie: 9](#_Toc21359742)

[Database maken met behulp van doctrine in symfony. 12](#_Toc21359743)

[Bootstrap opmaak gebruiken in symfony. 14](#_Toc21359744)

[hoofdstuk ‘opmaak van de “twig” schermen voor new. Edit etc.’. 16](#_Toc21359745)

[hoofdstuk ‘doctrine en entities aanmaken’. 17](#_Toc21359746)

[Creating an Entity Class 18](#_Toc21359747)

[Indien je een fout maakt bij het invoeren van een entity dan: 24](#_Toc21359748)

[Extra velden FosUser 25](#_Toc21359749)

[Rollen aan FOS-USER gebruikers toekennen met FOS\_USER COMMAND\_LINE 28](#_Toc21359750)

[Opmaak van de fosuser velden regelen 28](#_Toc21359751)

[Programma schrijven 29](#_Toc21359752)

[Een eenvoudige controller aanmaken. 29](#_Toc21359753)

[Crud Controller (Create, Read, Update en Delete) 29](#_Toc21359754)

[Een homepage maken 32](#_Toc21359755)

[VichUploaderBundle gebruiken. 34](#_Toc21359756)

[CKEditor gebruiken voor bijvoorbeeld een description veld. 37](#_Toc21359757)

[Last user Activity opslaan in een veld van de User entiteit. 38](#_Toc21359758)

[Mail : Swiftmailerfunctie gebruiken. 40](#_Toc21359759)

[Pagination 43](#_Toc21359760)

[Samengestelde Forms 46](#_Toc21359761)

[Globale session parameter 48](#_Toc21359762)

[Wegschrijven van gegevens in de database 49](#_Toc21359763)

[Extra: Overzetten naar andere computer van symfony project 50](#_Toc21359764)

[Foutmeldingen: 51](#_Toc21359765)

# Installatie onderdelen

### XAMPP

Je moet XAMPP geïnstalleerd hebben (meestal is dat al gebeurd). Hiermee kun je namelijk een virtuele server maken met SQL en PHP. Symfony 4 heeft namelijk ook PHP 7.1 of hoger nodig en de laatste XAMPP versie heeft dat wel. XAMPP kun je voor zowel Linux als Windows installeren maar wij gebruiken alleen Windows. Downloaden kan via <https://www.apachefriends.org/download.html> en je neemt dan de hoogste versie voor windows (meestal onderaan).

### Composer

Composer moet geïnstalleerd zijn.

Meestal is composer al geïnstalleerd op je computer. Je kunt dit controleren door bij de prompt het commando “composer” te typen. Je krijgt dan als het goed is een reactie met de opties en commando’s van composer. Als je dit zie kun je stap 1 overslaan.

Wel is het verstandig om composer naar de laatste versie te brengen je kunt dit doen door het commando “composer self-update” te geven. Zo ben je verzekerd dat je altijd de laatste update hebt van composer.

1. Is composer niet geïnstalleerd dan kun je naar getcomposer.org gaan en daar de file downloaden (als je windows gebruikt de ‘windows installer’ versie en run de file die gedownload is (composer-setup.exe). Test na installatie door middel van “composer” te typen bij de command prompt of dat hij geïnstalleerd is.

**Extra info:**

Met composer kun je precies bijhouden welke aanvullende packages (laravel, symfony, bootstrap etc.) je allemaal gebruikt in je project. Als je een project begint dan ga je in de juiste folder staan en type je in die folder vanaf de command prompt “composer create-project <welk package je gebruikt> <naam van je project>”.

Composer gaat dan een project folder creeren met het door jou aangegeven package en zet daar allerlei files klaar die je gebruikt voor dat project. In de root van je project staan 2 belangrijke files waar je precies kan zien welke packages er gebruikt worden en welke versies (sommige packages zoals symfony maken ook weer gebruik van andere packages maar die zie je daar allemaal). Deze 2 files zijn composer.json en composer.lock.

Later kun je nog meer packages (fos\_user etc.) toevoegen aan je project. Al die packages worden in de **vendor** folder geïnstalleerd.

In dit verhaal zullen we een project op basis van symfony 4 gaan installeren en later packages toevoegen zodat je van allerlei functies uit die packages (bootstrap, crud, doctrine, fos\_user etc.) gebruik kan maken.

### Nieuw Symfony project

Ga in de command prompt in de folder staan waar je jouw project wil neerzetten (bijv. D:xampp\htdocs)

D:

CD\ of cd..

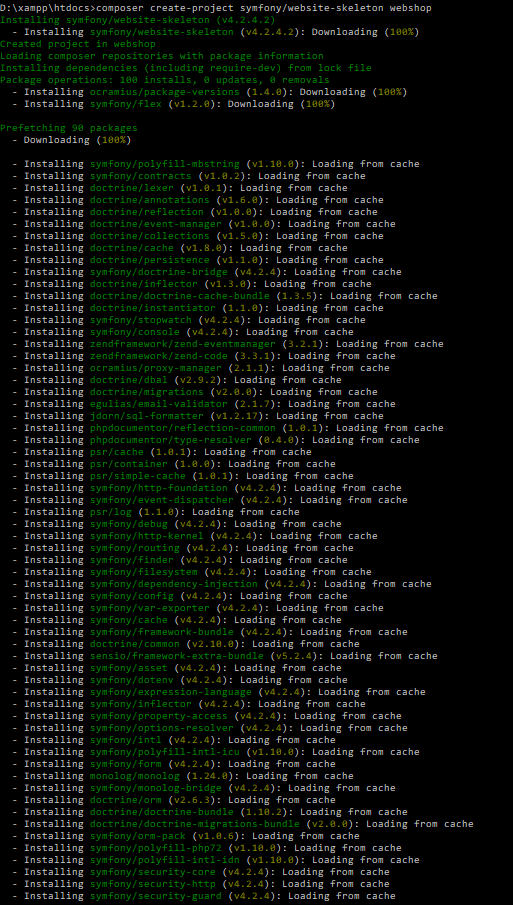
Cd xammpp\htdocs

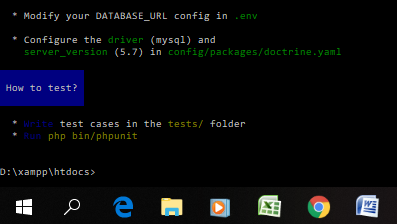
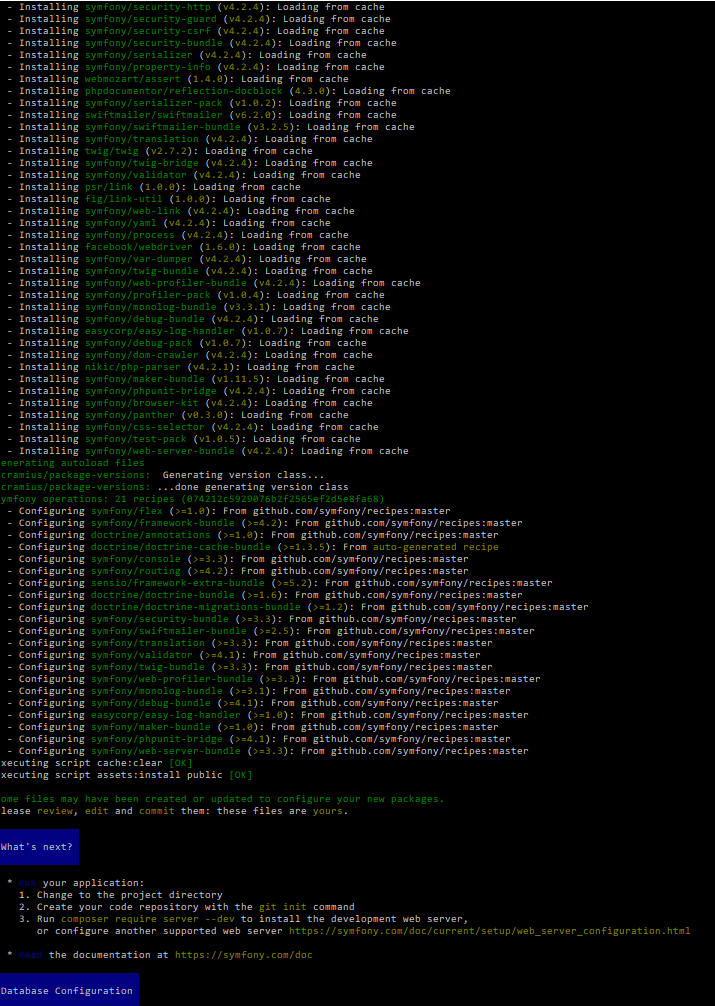
Maak het project aan met symfony 4(in ons voorbeeld noemen we het project “webshop”).

composer create-project symfony/website-skeleton webshop

**!!!!!LETOP of je website-skeleton moet gebruiken of alleen skeleton (dit is een compactere versie, normaal is website-skeleton).**

Als het goed is gaat composer nu allerlei zaken installeren in de folder “webshop”. Hieronder volgt de output. Zoals je kan zien installeer je door gebruik te maken van het hele symfony-skeleton ook allerlei andere packages waar symfony standaard gebruik van kan maken. Enkele die je kunt zien in de output en waar je gebruik van kan maken zijn Flex, annotations, doctrine, debug, routing, twig





Nu alles geïnstalleerd is (en je bij xampp de SQL en PHP heb gestart) kun je nu de server voor jouw project starten.

Ga dan in de folder van jouw project staan in de command prompt (in ons voorbeeld webshop)

cd webshop

en type in die folder

php bin/console server:run

Je ziet dan als het goed is de melding:



Je kan dan via je browser naar 127.0.0.1:8000 gaan en dan zie je de welkom melding van symfony.

Als je gebruik gaat maken van een database dan moet je die ook aanmaken. Dit staat in dit document beschreven aan het begin van het hoofdstuk over fos\_user

### PhpStorm (editor)

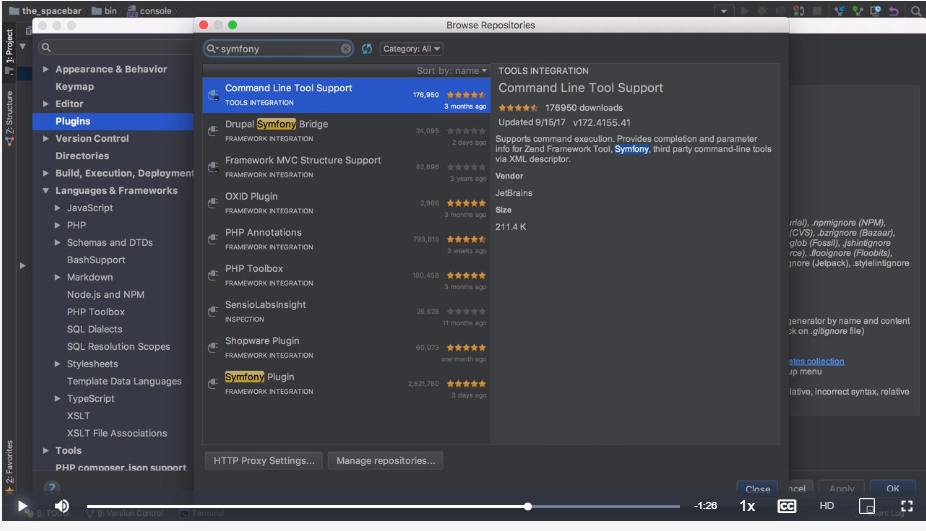
Je moet een editor hebben om al je files te editten. Ik ga er even vanuit dat dit PHP storm is. Als je een andere gebruik moet je kijken op internet of er mogelijkheden tot integratie zijn. In dit geval bespreek ik alleen de PHP Storm integratie.

Installeer PhpStorm via <https://www.jetbrains.com/phpstorm/download/#section=windows>

Let op bij het installeren dat je geen additionele zaken installeer die je niet nodig hebt.

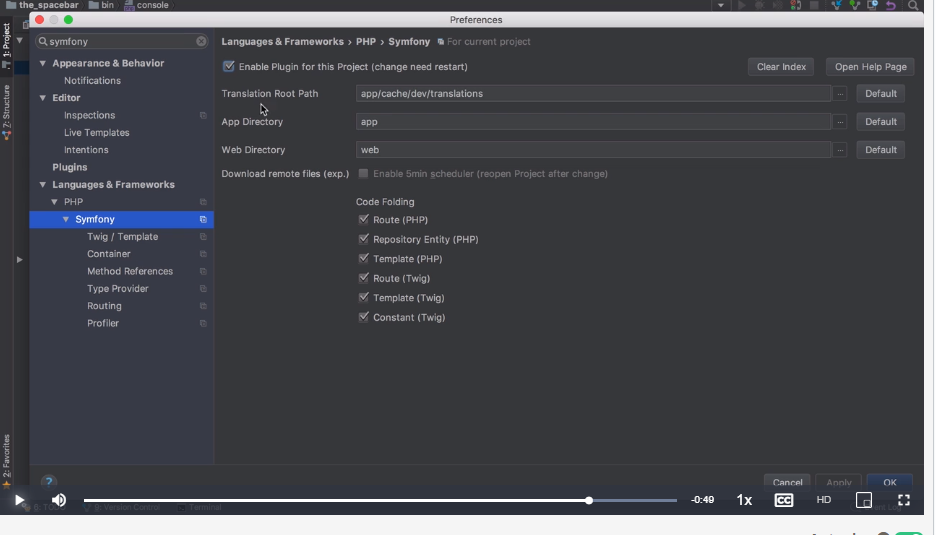
De volgende zaken kun je in PhpStorm aanpassen zodat het beter samenwerkt met symfony:

Ga naar ‘preferences’(In Nederlands zal dit anders heten instellingen of settings) boven in het menu en vervolgens ‘Plugins’. Vervolgens naar ‘Browse repositories’ .



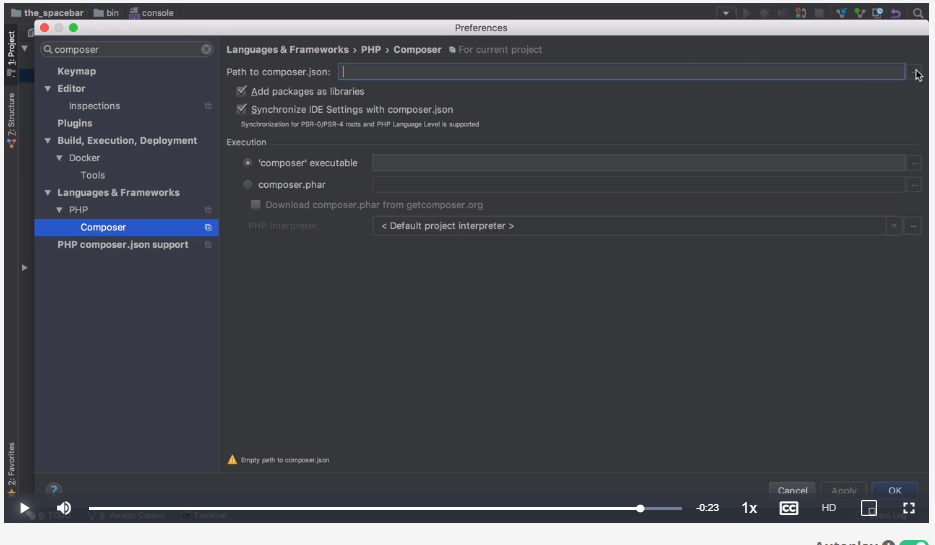
Zoek dan naar symfony en installeer als eerste de “PHP annotations” en “PHP toolbox”. Daarna pas de “symfony plugin”. Dit doe je door op de install button te klikken voor alle drie. Als er bij 1 geen install button verschijnt is hij waarschijnlijk al geïnstalleerd. Na installatie PHPStorm even opnieuw opstarten.

Ga weer naar preferences en zoek naar symfony. Als het goed is zie je dan onderstaand scherm als je op symfony klik en zorg er dan voor dat her een vinkje staat bij ‘enable plugin for this project…….’



Onderstaande geldt voor oude PhpStorm versies de nieuwe zet het automatisch al goed.

Ga dan weer naar preferences en zoek naar composer. Vul dan bij path to composer.json het path in van je projectfile. Die vind je eenvoudig door rechts te klikken waar in het plaatje de muis staat. Dan klik je in ons geval op de webshop folder (of als hij daar al staat op enter. Druk op OK en we zijn klaar voor gebruik met PhpStorm.



### GIT

Login via de browser op je GitHub account.

* Maak hier een nieuwe repository aan.
* Kopieer de link van je repository (Ctrl-C)
* Geef commando via de terminal in unit4 folder op je laptop
  + Git clone <gekopieerde link van je repository>
* Als er al codes stonden in je Github repository (dus hij was er al en je hebt niets aan hoeven te maken in stap 1) kun je de code vanuit Github naar je computer halen. Zo heb je de laatste status.
  + Git pull origin (alleen als er al een repository was)
* Vervolgens de volgende commando’s uitvoeren. LET OP bij sommige staat dat je het commando alleen de eerste keer moet gebruiken na het aanmaken van een nieuwe repository.
  + Git init (alleen de eerste keer)
  + Git status
  + Git add .
  + Git status
  + Git commit –m “tekst waarom je nieuwe dump maakt”
  + Git remote add origin <link van je repository> (alleen de eerste keer)
  + Git push (het kan zijn dat je nooit gepusht hebt kopieer dan wat er op je terminal komt)

Bij fouten kijk goed naar de melding. In het begin had ik problemen met pushen en toen werkte wel het commando

git push –f origin master (alleen uitvoeren als niets meer lukt)

# Extra symfony packages installeren

Op school heb je met bepaalde extra packages leren werken die nog niet geïnstalleerd zijn (bijvoorbeeld bootstrap, fosuser) Die wil je waarschijnlijk ook installeren. CRUD systeem is als het goed is al een onderdeel van bestaande Doctrine package dat gelijk bij installatie van Symfony al is geïnstalleerd.

### FosUser installatie:

Installeer Fos\_User mbv composer:

composer require friendsofsymfony/user-bundle

Op het eind krijg je dan nog een error met iets van db\_driver en fos\_user maar dat is normaal.

Nu moet je de user entity maken dit is het class object dat de user objecten gebruikt. Doe dit in de file webshop\src\Entity\User.php. FosUser maakt namelijk gebruik van de User entity. Als die niet bestaat dan aanmaken en onderstaande er in zetten.

<?php

// src/Entity/User.php

namespace App\Entity;

use FOS\UserBundle\Model\User as BaseUser;

use Doctrine\ORM\Mapping as ORM;   
/\*\*

\* @ORM\Entity

\* @ORM\Table(name="fos\_user")

\*/

class User extends BaseUser

{

/\*\*

\* @ORM\Id

\* @ORM\Column(type="integer")

\* @ORM\GeneratedValue(strategy="AUTO")

\*/

protected $id;

public function \_\_construct()

{

parent::\_\_construct();

// your own logic

}

}

Vervolgens moeten we ook aan symphony vertellen dat we fosuser als security package gaan gebruiken. Dit doe je door webshop/config/packages/security.yaml aan te passen.

Edit the /webshop/config/packages/security.yaml file and change its content with the following instead:

De inhoud van de file kun je helemaal overschrijven met onderstaande.

# config/packages/security.yaml

security:

encoders:

FOS\UserBundle\Model\UserInterface: bcrypt

role\_hierarchy:

ROLE\_ADMIN: ROLE\_USER

ROLE\_SUPER\_ADMIN: ROLE\_ADMIN

# https://symfony.com/doc/current/security.html#where-do-users-come-from-user-providers

providers:

fos\_userbundle:

id: fos\_user.user\_provider.username\_email

firewalls:

dev:

pattern: ^/(\_(profiler|wdt)|css|images|js)/

security: **false**

main:

pattern: ^/

form\_login:

provider: fos\_userbundle

csrf\_token\_generator: security.csrf.token\_manager

logout: **true**

anonymous: **true**

# Easy way to control access for large sections of your site

# Note: Only the \*first\* access control that matches will be used

access\_control:

- { path: ^/login$, role: IS\_AUTHENTICATED\_ANONYMOUSLY }

- { path: ^/register, role: IS\_AUTHENTICATED\_ANONYMOUSLY }

- { path: ^/resetting, role: IS\_AUTHENTICATED\_ANONYMOUSLY }

- { path: ^/admin/, role: ROLE\_ADMIN }

Nu Fosuser gedefinieerd is als standard user provider moet je fosuser nog configureren. Dit kan door een nieuwe yaml file te maken: webshop/config/packages/fos\_user.yaml.

# config/packages/fos\_user.yaml

fos\_user:

db\_driver: orm # other valid values are 'mongodb' and 'couchdb'

firewall\_name: main

user\_class: App\Entity\User

from\_email:

address: "test@domain.com"

sender\_name: "test@domain.com"

As required by FOSUserBundle the PHP and Twig templating engines must be enabled, so be sure to enable it in the /webshop/config/packages/framework.yaml file:

Dit toevoegen aan de file. **LETOP!! Framwork: staat als het goed is al bovenaan dus die moet je niet meekopieren.**

framework:

templating:

engines: ['twig', 'php']

You need to enable the FOSUserBundle routes as well, so create the routes file for FOSUserBundle with the file  /project/config/routes/fos\_user.yaml and add the following content:

fos\_user:

resource: "@FOSUserBundle/Resources/config/routing/all.xml"

This will register the default routes of FOSUserBundle like the login, logout, register etc.

FosUser maakt gebruik van een database. Je moet een database aanmaken. We doen dit met doctrine.

### Database maken met behulp van doctrine in symfony.

Als eerste moet je de .env file aanpassen die in de hoofdfolder staat van je project. Daarin staat onderstaande regel:

DATABASE\_URL="mysql://db\_user:db\_password@127.0.0.1:3306/db\_name"

Je kunt deze aanpassen met een eigen usernaam(db\_user) en passwoord(db\_password) en je zelf gekozen naam voor de database.

Stel we willen een usernaam ‘root’ en geen password en de database gaan we ‘webshop’ noemen dan moet je de .env file aanpassen als volgt:

DATABASE\_URL=mysql://root@127.0.0.1:3306/webshop

Vervolgens kun je de database creeren met het commando op de command prompt

php bin/console doctrine:database:create

Heb een paar keer een foutmelding gehad dat hij templating niet kon vinden in config/packages/framework.yaml. Nadat ik toen dat met tab op de juiste plaats zette of opnieuw kopieerde pakte hij het creeren van de DB wel.

Controleer in XAMPP bij de SQL Admin dat de database is aangemaakt.

Vervolgens kun je extra tabellen (entities) met velden maken. Zie daarvoor het hoofdstuk **‘doctrine en entities aanmaken’.** Dat hoef je nu nog niet te doen.

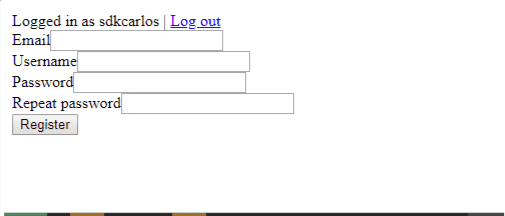
Fos User heeft al wel zijn entity aangemaakt (User.php) en deze entiteit willen we nu toevoegen aan het database schema.

php bin/console doctrine:schema:update --force

This will create the fos\_user table in your database and you will be able to register users. With this step, the installation and setup of FOSUserBundle finishes.

**Kopieren van fosuser views naar je projectfolder zodat je ze zelf kan editten**. What you need to do is copy all the templates files from vendor/friendsofsymfony/user-bundle/Resources/views and put them in a directory you create in /templates/bundles/FOSUserBundle. You can then edit the files in your newly created directory which will override the default templates.

As last step, you can test FOSUserBundle in your project accesing the web project, for example at the register route e.g http://127.0.0.1:8000/register, where you will find the form of FOSUserBundle:



**Extra Informatie:**

Met het onderstaande commando forceer je een ‘harde’ database update met je entiteiten.

php bin/console doctrine:schema:update –force

In veel documentatie zie je echter date en voorzichtere method gebaseerd op een migratie of gedeeltelijke migratie wordt gebruikt. Dit gebeurt in 2 stappen. De eerste stap is voorbereiden waarbij er in de folder migrations een migratie file wordt gecreëerd.

php bin/console make:migration

De tweede stap is de werkelijke migratie (of gedeeltelijk) uitvoeren. Hiervoor zijn allerlei opties mogelijk maar dat is een complex verhaal. De voorbereide migratie in zijn geheel uitvoeren kan met:

php bin/console doctrine:migrations:migrate

### Bootstrap opmaak gebruiken in symfony.

Hiervoor kun je als voorbeeld onderstaande file gebruiken (Let Op, je moet hem wel aanpassen met je eigen menu’s en linken), welke je in webshop\templates\base.html.twig moet plakken als bron. Dat is een voorbeeld. Hieronder staat de file uit je mail.

<!doctype **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="utf-8"**>  
 <**meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"**>  
 <**meta name="description" content=""**>  
 <**meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors"**>  
 <**meta name="generator" content="Jekyll v3.8.5"**>  
 <**title**>{% **block** title %}Welcome!{% **endblock** %}</**title**>  
 *<!-- Custom styles for this template -->* <**link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous"**>  
 <**link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/timepicker/1.3.5/jquery.timepicker.min.css"**>  
  
 {% **block** stylesheets %}  
 *<!-- <link rel="stylesheet" href="/css/style.css"> -->* {% **endblock** %}  
</**head**>  
<**body style="color**:**white**; **background-image**: **url**(**/images/Achtergrond.jpg**); **min-height**: 100**vh**; **padding-bottom**: 5**rem**; **background-size**: 100%; **background-repeat**: **no-repeat**; **background-attachment**: **fixed**;**"**>  
<**header style="margin-bottom**: 20**px"**>  
 <**nav class="navbar navbar-expand-md navbar-dark " style="background-color**: **#3b5998"**>  
 <**button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation"**>  
 <**span class="navbar-toggler-icon"**></**span**>  
 </**button**>  
 <**div class="collapse navbar-collapse" id="navbarCollapse"**>  
 {% **if** is\_granted(**'ROLE\_ADMIN'**) %}  
 <**ul class="navbar-nav mr-auto"**>  
 <**li class="nav-item "**>  
 <**a class="nav-link text-white" href="**{{ path(**'homepage'**) }}**"**>Home <**span class="sr-only"**>(current)</**span**></**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'reserveringenvrij'**) }}**"**>Reserveringen</**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'kamers'**) }}**"**>Kamers</**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'contact'**) }}**"**>Contact</**a**>  
 </**li**>  
 <**li class="nav-item dropdown"**>  
  
 <**a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink" data-toggle="dropdown"** >  
 Administratie  
 </**a**>  
 <**div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink"**>  
 <**a class="dropdown-item" href="**{{ path(**'user\_index'**) }}**"**>User</**a**>  
 <**a class="dropdown-item" href="**{{ path(**'reservering\_index'**) }}**"**>Reservering</**a**>  
 <**a class="dropdown-item" href="**{{ path(**'betaal\_index'**) }}**"**>Betaal</**a**>  
 <**a class="dropdown-item" href="**{{ path(**'kamer\_index'**) }}**"**>Kamer</**a**>  
 <**a class="dropdown-item" href="**{{ path(**'soort\_index'**) }}**"**>Soort</**a**>  
 <**a class="dropdown-item" href="**{{ path(**'image\_index'**) }}**"**>Image</**a**>  
 <**a class="dropdown-item" href="**{{ path(**'seizoen\_index'**) }}**"**>Seizoen</**a**>  
 </**div**>  
 </**li**>  
 </**ul**>  
 {% **else** %}  
 <**ul class="navbar-nav mr-auto"**>  
 <**li class="nav-item "**>  
 <**a class="nav-link text-white" href="**{{ path(**'homepage'**) }}**"**>Home <**span class="sr-only"**>(current)</**span**></**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'reserveringen'**) }}**"**>Reserveringen</**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'kamers'**) }}**"**>Kamers</**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'contact'**) }}**"**>Contact</**a**>  
 </**li**>  
 </**ul**>  
 {% **endif** %}  
 <**div class="navbar navbar-inverse" role="navigation"**>  
 <**div class="blockMain "**>  
 {% **if** app.user %}  
 {% **set** mainMenu = [  
 {**'path'**: **'fos\_user\_security\_logout'**, **'name'** : **'Logout'**}  
 ] %}  
 {% **else** %}  
 {% **set** mainMenu = [  
 {**'path'**: **'fos\_user\_security\_login'**, **'name'**: **'Login'** },  
 {**'path'**: **'fos\_user\_registration\_register'**, **'name'**: **'Registreren'** }  
 ] %}  
 {% **endif** %}  
 <**div class="navbar-collapse collapse"**>  
 <**a class="nav-link text-white"**>{{ app.user }}</**a**>  
 <**ul class="nav navbar-nav"**>  
 {% **for** item **in** mainMenu %}  
 <**li class="nav-item"**{{ app.request.get(**'\_route'**) == item[**'path'**] ? **' class="active"'** : **''** }}>  
 <**a class="nav-link text-white" href="**{{ path(item[**'path'**]) }}**"**>{{ item[**'name'**] }}</**a**>  
 </**li**>  
 {% **endfor** %}  
 </**ul**>  
 </**div**>*<!--/.navbar-collapse -->* </**div**>  
 </**div**>  
 </**div**>  
 </**nav**>  
</**header**>  
<**div style="padding**: 20**px**;**" class="container"**>{% **block** login\_register %} {% **endblock** %}{% **block** body %}{% **endblock** %}</**div**>  
*<!-- FOOTER -->*{% **block** footer %}  
<**div style="margin-top**: 20**px**;**"**></**div**>  
<**footer class="bg-dark" style="position**: **absolute**; **bottom**: 0; **width**: 100%; **height**: 5**rem"**>  
 <**br**>  
</**footer**>  
{% **endblock** %}  
</**main**>  
<**script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"**></**script**>  
<**script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"**></**script**>  
<**script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"**></**script**>  
{% **block** javascripts %}{% **endblock** %}  
</**body**>  
</**html**>

**Een ander voorbeeld voor base.html.twig met ook ROLE\_USER naast ROLE\_ADMIN**

<!doctype **html**>  
<**html lang="en"**>  
<**head**>  
 <**meta charset="utf-8"**>  
 <**meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"**>  
 <**meta name="description" content=""**>  
 <**meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap contributors"**>  
 <**meta name="generator" content="Jekyll v3.8.5"**>  
 <**title**>{% **block** title %}Welcome!{% **endblock** %}</**title**>  
 *<!-- Custom styles for this template -->* <**link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous"**>  
 <**link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/timepicker/1.3.5/jquery.timepicker.min.css"**>  
  
 {% **block** stylesheets %}  
 *<!-- <link rel="stylesheet" href="/css/style.css"> -->* {% **endblock** %}  
</**head**>  
<**body style="min-height**: 100**vh**; **padding-bottom**: 5**rem**; **background-size**: 100%; **background-repeat**: **no-repeat**; **background-attachment**: **fixed**;**"**>  
<**header style="margin-bottom**: 20**px"**>  
 <**nav class="navbar navbar-expand-md navbar-dark " style="background-color**: **#3b5998"**>  
 <**button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation"**>  
 <**span class="navbar-toggler-icon"**></**span**>  
 </**button**>  
 <**div class="collapse navbar-collapse" id="navbarCollapse"**>  
 {% **if** is\_granted(**'ROLE\_ADMIN'**) %}  
 <**ul class="navbar-nav mr-auto"**>  
 <**li class="nav-item "**>  
 <**a class="nav-link text-white" href="**{{ path(**'homepage'**) }}**"**>Home <**span class="sr-only"**>(current)</**span**></**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'category\_index'**) }}**"**>Forum</**a**>  
 </**li**>  
 <**li class="nav-item dropdown"**>  
  
 <**a class="nav-link dropdown-toggle" href="#" id="navbarDropdownMenuLink" data-toggle="dropdown"** >  
 Approvals  
 </**a**>  
 <**div class="dropdown-menu" aria-labelledby="navbarDropdownMenuLink"**>  
 *{# <a class="dropdown-item" href="{{ path('category\_index') }}">User approval</a>#}* <**a class="dropdown-item" href="**{{ path(**'topicnotapproved\_index'**) }}**"**>Topic approval</**a**>  
 </**div**>  
 </**li**>  
 </**ul**>  
 {% **elseif** is\_granted(**'ROLE\_CUSTOMER'**) %}  
 <**ul class="navbar-nav mr-auto"**>  
 <**li class="nav-item "**>  
 <**a class="nav-link text-white" href="**{{ path(**'homepage'**) }}**"**>Home <**span class="sr-only"**>(current)</**span**></**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'category\_index'**) }}**"**>Forum</**a**>  
 </**li**>  
 </**ul**>  
 {% **else** %}  
 <**ul class="navbar-nav mr-auto"**>  
 <**li class="nav-item "**>  
 <**a class="nav-link text-white" href="**{{ path(**'homepage'**) }}**"**>Home <**span class="sr-only"**>(current)</**span**></**a**>  
 </**li**>  
 <**li class="nav-item"**>  
 <**a class="nav-link text-white" href="**{{ path(**'category\_index'**) }}**"**>Forum</**a**>  
 </**li**>  
 </**ul**>  
 {% **endif** %}  
 <**div class="navbar navbar-inverse" role="navigation"**>  
 <**div class="blockMain "**>  
 {% **if** app.user %}  
 {% **set** mainMenu = [  
 {**'path'**: **'fos\_user\_security\_logout'**, **'name'** : **'Logout'**}  
 ] %}  
 {% **else** %}  
 {% **set** mainMenu = [  
 {**'path'**: **'fos\_user\_security\_login'**, **'name'**: **'Login'** },  
 {**'path'**: **'fos\_user\_registration\_register'**, **'name'**: **'Registreren'** }  
 ] %}  
 {% **endif** %}  
 <**div class="navbar-collapse collapse"**>  
 <**a class="nav-link text-white"**>{{ app.user }}</**a**>  
 <**ul class="nav navbar-nav"**>  
 {% **for** item **in** mainMenu %}  
 <**li class="nav-item"**{{ app.request.get(**'\_route'**) == item[**'path'**] ? **' class="active"'** : **''** }}>  
 <**a class="nav-link text-white" href="**{{ path(item[**'path'**]) }}**"**>{{ item[**'name'**] }}</**a**>  
 </**li**>  
 {% **endfor** %}  
 </**ul**>  
 </**div**>*<!--/.navbar-collapse -->* </**div**>  
 </**div**>  
 </**div**>  
 </**nav**>  
</**header**>  
<**div style="padding**: 20**px**;**" class="container"**>{% **block** login\_register %} {% **endblock** %}{% **block** body %}{% **endblock** %}</**div**>  
*<!-- FOOTER -->*{% **block** footer %}  
  
{% **endblock** %}  
</**main**>  
<**script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"**></**script**>  
<**script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"**></**script**>  
<**script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"**></**script**>  
{% **block** javascripts %}{% **endblock** %}  
</**body**>  
</**html**>

### hoofdstuk ‘opmaak van de “twig” schermen voor new. Edit etc.’.

Om bootstrap forms te krijgen voor je TWIG files moet je de regel met grootste letters toevoegen aan de twig.yaml file in de packages folder:

Config\packages\twig.yaml

twig:  
 default\_path: '%kernel.project\_dir%/templates'  
 debug: '%kernel.debug%'  
 strict\_variables: '%kernel.debug%'  
 **form\_themes: ['bootstrap\_4\_layout.html.twig']** #- '@FOSCKEditor/Form/ckeditor\_widget.html.twig'

# hoofdstuk ‘doctrine en entities aanmaken’.

Als je met behulp van doctrine een entity/class aanmaakt krijg je default al een ID veld.

Je moet altijd van te voren al een tabel(dit is dus ook de entity) bedacht hebben met de velden en ook het type van dat veld (bijv. int, string of float)

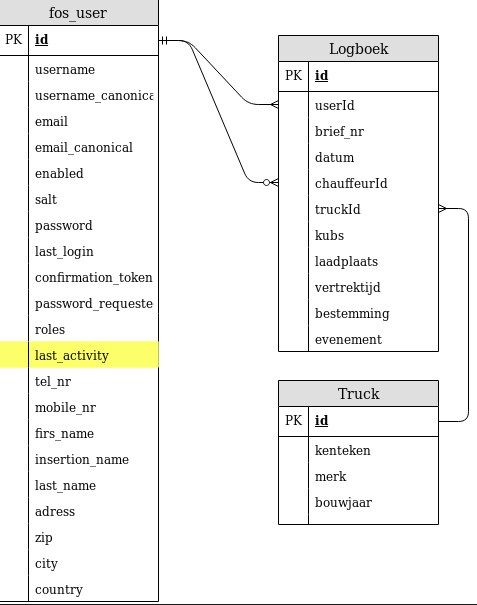
Als voorbeeld maken we entities aan voor onderstaand schema.

**Let op:**

De volgorde van het definiëren van de tabellen is van belang bij het maken van de entities en de relaties. Denk goed na over de volgorde van het aanmaken van de tabellen. De fos\_user tabel bestaat al want die heb je met fos\_user installatie al gemaakt (alleen de velden vanaf tel\_nr t/m country zouden toegevoegd moeten worden).

Als je de tabel logboek eerst aanmaak dan kun je alleen de Many to one relatie leggen met fos\_user en nog niet de manytoone naar truck. Je moet dan later nog bij het aanmaken van truck de One to many aanmaken naar logboek.

We kiezen er daarom om om eerst de tabel Truck te maken en vervolgens Logboek en dan gelijk alle relaties te definieren.



### Creating an Entity Class

Suppose you're building an application where products need to be displayed. Without even thinking about Doctrine or databases, you already know that you need a Product object to represent those products. Dit object gebruikt dan weer dezelfde structuur als de database tabellen

You can use the make:entity command to create this class and any fields you need. The command will ask you some questions - answer them like done below:

Let op ID hoef je niet op te geven die wordt automatisch gemaakt.

**het kan gebeuren dat je een fout maakt bij het aanmaken van entiteiten.**

**Dit kun je oplossen door de entiteit te verwijderen uit de map entity en uit de map repository.**

**als je de twee bestanden hebt verwijderd kun je de entiteit opnieuw aanmaken.**

|  |
| --- |
| php bin/console make:entity  Class name of the entity to create or update:  > Truck  New property name (press <return> to stop adding fields):  > kenteken  Field type (enter ? to see all types) [string]:  > string  Field length [255]:  > 10  Can this field be null in the database (nullable) (yes/no) [no]:  > no  Add another property? Enter the property name (or press <return> to stop adding fields):  > merk  Field type (enter ? to see all types) [string]:  > string  Field length [255]:  > 50  Can this field be null in the database (nullable) (yes/no) [no]:  > no  Add another property? Enter the property name (or press <return> to stop adding fields):  > bouwjaar  Field type (enter ? to see all types) [string]:  > integer  Can this field be null in the database (nullable) (yes/no) [no]:  > no  Add another property? Enter the property name (or press <return> to stop adding fields): >  >  (press enter again to finish) |

Vervolgens gaan we logboek aanmaken en gelijk de relaties leggen:

php bin/console make:entity

Class name of the entity to create or update:

> Logboek

to stop adding fields):

> userId

Field type (enter ? to see all types) [string]:

> relation

What class should this entity be related to?:

LET HIER OP. Je denkt dan fos\_user als je naar de table kijkt alleen maakt fos\_user gebruik van de User entity (kijk ook maar in de Entity folder daar staat de User.php file)

>User

What type of relationship is this?

------------ -------------------------------------------------------------------

Type Description

------------ -------------------------------------------------------------------

ManyToOne Each Logboek relates to (has) one User.

Each User can relate to (can have) many Logboek objects

OneToMany Each Logboek can relate to (can have) many User objects.

Each User relates to (has) one Logboek

ManyToMany Each Logboek can relate to (can have) many User objects.

Each User can also relate to (can also have) many Logboek objects

OneToOne Each Logboek relates to (has) exactly one User.

Each User also relates to (has) exactly one Logboek.

------------ -------------------------------------------------------------------

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:

Vanuit de table zie je dat het een many to one is en niet null mag zijn.

>ManyToOne

Is the Logboek.userId property allowed to be null (nullable)? (yes/no) [yes]:

>no

Do you want to add a new property to User so that you can access/update Logboek objects from it - e.g. $user->getLogboeks()? (yes/no) [yes]:

>yes

New field name inside User [logboeks]:

>logboeks

Do you want to activate orphanRemoval on your relationship?

A Logboek is "orphaned" when it is removed from its related User.

e.g. $user->removeLogboek($logboek)

NOTE: If a Logboek may \*change\* from one User to another, answer "no".

Do you want to automatically delete orphaned App\Entity\Logboek objects (orphanRemoval)? (yes/no) [no]:

>no

updated: src/Entity/Logboek.php

updated: src/Entity/User.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> brief\_nr

Field type (enter ? to see all types) [string]:

> integer

Can this field be null in the database (nullable) (yes/no) [no]:

> no

updated: src/Entity/Logboek.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> datum

Field type (enter ? to see all types) [string]:

> date

Can this field be null in the database (nullable) (yes/no) [no]:

> no

updated: src/Entity/Logboek.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> chauffeurId

Field type (enter ? to see all types) [integer]:

> relation

What class should this entity be related to?:

> User

What type of relationship is this?

------------ -------------------------------------------------------------------

Type Description

------------ -------------------------------------------------------------------

ManyToOne Each Logboek relates to (has) one User.

Each User can relate to (can have) many Logboek objects

OneToMany Each Logboek can relate to (can have) many User objects.

Each User relates to (has) one Logboek

ManyToMany Each Logboek can relate to (can have) many User objects.

Each User can also relate to (can also have) many Logboek objects

OneToOne Each Logboek relates to (has) exactly one User.

Each User also relates to (has) exactly one Logboek.

------------ -------------------------------------------------------------------

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]: > ManyToOne

Is the Logboek.chauffeurId property allowed to be null (nullable)? (yes/no) [yes]:

> no

Do you want to add a new property to User so that you can access/update Logboek objects from it - e.g. $user->getLogboeks()? (yes/no) [yes]:

> yes

A new property will also be added to the User class so that you can access the related Logboek objects from it.

New field name inside User [logboeks]:

> logboeks

Do you want to activate orphanRemoval on your relationship?

A Logboek is "orphaned" when it is removed from its related User.

e.g. $user->removeLogboek($logboek)

NOTE: If a Logboek may \*change\* from one User to another, answer "no".

Do you want to automatically delete orphaned App\Entity\Logboek objects (orphanRemoval)? (yes/no) [no]:

> no

Not generating User::getLogboeks(): method already exists

Not generating User::addLogboek(): method already exists

Not generating User::removeLogboek(): method already exists

updated: src/Entity/Logboek.php

no change: src/Entity/User.php

Omdat je eerder al tijdens het maken van deze entity een relatie heb gelegd tussen class Logboek en class User (bij field UserID) zie je dat hij sommige files niet creert of update omdat de relatie er al was.

Add another property? Enter the property name (or press <return> to stop adding fields):

> truckId

Field type (enter ? to see all types) [integer]:

> relation

What class should this entity be related to?:

> Truck

What type of relationship is this?

------------ --------------------------------------------------------------------

Type Description

------------ --------------------------------------------------------------------

ManyToOne Each Logboek relates to (has) one Truck.

Each Truck can relate to (can have) many Logboek objects

OneToMany Each Logboek can relate to (can have) many Truck objects.

Each Truck relates to (has) one Logboek

ManyToMany Each Logboek can relate to (can have) many Truck objects.

Each Truck can also relate to (can also have) many Logboek objects

OneToOne Each Logboek relates to (has) exactly one Truck.

Each Truck also relates to (has) exactly one Logboek.

------------ --------------------------------------------------------------------

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:

> ManyToOne

Is the Logboek.truckId property allowed to be null (nullable)? (yes/no) [yes]:

> no

Do you want to add a new property to Truck so that you can access/update Logboek objects from it - e.g. $truck->getLogboeks()? (yes/no) [yes]:

> yes

A new property will also be added to the Truck class so that you can access the related Logboek objects from it.

New field name inside Truck [logboeks]:

>logboeks

Do you want to activate orphanRemoval on your relationship?

A Logboek is "orphaned" when it is removed from its related Truck.

e.g. $truck->removeLogboek($logboek)

NOTE: If a Logboek may \*change\* from one Truck to another, answer "no".

Do you want to automatically delete orphaned App\Entity\Logboek objects (orphanRemoval)? (yes/no) [no]:

> no

updated: src/Entity/Logboek.php

updated: src/Entity/Truck.php

Add another property? Enter the property name (or press <return> to stop adding fields):

> kubs

Field type (enter ? to see all types) [string]:

> integer

Can this field be null in the database (nullable) (yes/no) [no]:

>no

updated: src/Entity/Logboek.php

Add another property? Enter the property name (or press <return> to stop adding fields):

**> laadplaats**

**Field type (enter ? to see all types) [string]:**

**> string**

**Field length [255]:**

**> 50**

**Can this field be null in the database (nullable) (yes/no) [no]:**

**> no**

**updated: src/Entity/Logboek.php**

**Add another property? Enter the property name (or press <return> to stop adding fields):**

**> vertrektijd**

**Field type (enter ? to see all types) [string]:**

**> time**

**Can this field be null in the database (nullable) (yes/no) [no]:**

**> no**

**updated: src/Entity/Logboek.php**

**Add another property? Enter the property name (or press <return> to stop adding fields):**

**> bestemming**

**Field type (enter ? to see all types) [string]:**

**> string**

**Field length [255]:**

**> 50**

**Can this field be null in the database (nullable) (yes/no) [no]:**

**> no**

**updated: src/Entity/Logboek.php**

**Add another property? Enter the property name (or press <return> to stop adding fields):**

**> evenement**

**Field type (enter ? to see all types) [string]:**

**> string**

**Field length [255]:**

**> 50**

**Can this field be null in the database (nullable) (yes/no) [no]:**

**> yes**

**updated: src/Entity/Logboek.php**

**Add another property? Enter the property name (or press <return> to stop adding fields):**

**>**

**Success!**

**Next: When you're ready, create a migration with make:migration (wij doen dit echter met onderstaand commando)**

(press enter again to finish)

Om die entities/classes te activeren in de database moet je weer onderstaand commando geven.

php bin/console doctrine:schema:update --force

### Indien je een fout maakt bij het invoeren van een entity dan:

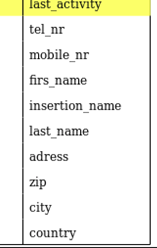
kun je dit oplossen door de entiteit te verwijderen uit de map entity en uit de map repository.

Als je de twee bestanden hebt verwijderd kun je de entiteit opnieuw aanmaken.

### Extra velden FosUser

Als je extra velden wil maken bij FosUser User en je wilt die ook terug zien bij bijvoorbeeld registreren dan moet je het Default Form van FosUser Overschrijven.

In ons voorbeeld maken we gebruik van het ERD dat eerder genoemd is voor de fos\_user. De velden onder het last\_activity veld zijn geen standaard fos\_user velden. Die gaan we dan toevoegen.



Stel je wilt bovenstaande velden toevoegen aan de standard User van FosUser.

Dan moet je er voor zorgen dat je deze velden toevoeg aan de bestaande User.php entity zodat ze ook aan de fosuser database worden toegevoegd.

Dit doe je door de entiteit aan te passen **(LET OP: maak anders eerst een kopie van User.php naar User.old omdat je hier niet als je een fout maakt bij invoer het User.php bestand niet zo maar kan verwijderen omdat je dan oude zaken weggooi, als je een User.old heb gemaakt kun je die weer hernoemen naar User.php bij het maken van een fout)**

Velden toevoegen doe je door de bestaande User.php te updaten:

php bin/console make:entity

Class name of the entity to create or update:

> User

New property name (press <return> to stop adding fields):

> first\_name

Field type (enter ? to see all types) [string]:

> string

Field length [255]:

> 40

Can this field be null in the database (nullable) (yes/no) [no]:

> no

Add another property? Enter the property name (or press <return> to stop adding fields):

**En dat doe je dan voor alle velden die je toe wil voegen aan User**

Als je klaar bent met velden toevoegen aan User.php kun je de User.php file nog even bekijken.

Vervolgens moet je zorgen dat de velden in de Doctrine Database komen:

php bin/console doctrine:schema:update --force

Als je nu het standard registration Form gebruikt zul je zien dat de extra velden nog niet aanwezig zijn. Je moet FosUser vertellen dat je extra velden gebruikt.

De eerste stap is dat je dan een eigen Form gaat maken waarbij je gebruik maakt van de bestaande FosUser zaken en welke je toe wilt voegen.

Maak daarom een nieuwe file onder Form folder met naam RegistrationType.php .

**Kopieer onderstaande in en vul de functie buildForm aan met de extra veldnamen. In de file hieronder is dat al voor ‘ first\_name ‘ gedaan** (gebruik de juiste namen zoals je bij de entiteit updaten heb gebruikt, die kun je ook in de database zien).

De functie getParent zorgt er voor dat je de bestaande FosUser velden ook ziet.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | <?php  *// src/App/Form/RegistrationType.php*  namespace App\Form;  use Symfony\Component\Form\AbstractType;  use Symfony\Component\Form\FormBuilderInterface;  class RegistrationType extends AbstractType  {  public function buildForm(FormBuilderInterface $builder, array $options)  {  $builder->add('first\_name');  }  public function getParent()  {  return 'FOS\UserBundle\Form\Type\RegistrationFormType';  }  public function getBlockPrefix()  {  return 'app\_user\_registration';  }  } |

**NOTE**

If you don't want to reuse the fields added in FOSUserBundle by default, you can omit the getParent method and configure all fields yourself.

Now that you have created your custom form type, you must declare it as a service and add a tag to it. The tag must have a name value of form.type and an alias value that is the equal to the string returned from the getName method of your form type class. The alias that you specify is what you will use in the FOSUserBundle configuration to let the bundle know that you want to use your custom form.

Hieronder staat hoe dat je dat doet. Als het goed is is er al een service: gedefinieerd en hoef je alleen vanaf regel 3 te kopiëren en plakken:

* [YAML](https://symfony.com/doc/master/bundles/FOSUserBundle/overriding_forms.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | *# app/config/services.yml*  services:  app.form.registration:  class: App\Form\RegistrationType  tags:  - { name: form.type, alias: app\_user\_registration } |

 Let op dat de alias die je hier gebruikt hetzelfde is als in de functie getBlockPrefix in je Form RegistrationType.php zoals hierboven ook het geval is.

* [XML](https://symfony.com/doc/master/bundles/FOSUserBundle/overriding_forms.html)

Finally, you must update the configuration of the FOSUserBundle so that it will use your form type instead of the default one. Below is the configuration for changing the registration form type in YAML.

Als het goed is is er al een fos\_user dus alleen vanaf regel 4 kopieren.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | *# app/config/packages/fos\_user.yaml*  fos\_user:  *# ...*  registration:  form:  type: App\Form\RegistrationType |

**NOTE**

**Aanvullende Info:**

De originele FosUser velden welke je als eerste in het form krijgt staan in PHP file:

Vendor\friendsofsymfony\user-bundle\form\type\RegistrationFormType.php

Hier kun je de labelnamen eventueel veranderen van die velden.

### Rollen aan FOS-USER gebruikers toekennen met FOS\_USER COMMAND\_LINE

Zoek op internet op:

“FOSUserBundle Command Line Tools”

Hier kun je dan bestaande gebruikers via promote een rol toekennen . Bijvoorbeeld de user testuser een ROLE\_ADMIN geven:

php bin/console fos:user:promote testuser ROLE\_ADMIN

### Opmaak van de fosuser velden regelen

De opmaak van de fosuser velden kun je doen in de standaard layout file van fosuser “layout.html.twig”.

Deze file is te vinden Vendor\friendsofsymfony\user-bundle\resources\views

Om gebruik te maken van de base.html.twig file waar je bootstrap in heb staan moet je de onderstaande 2 regels bovenaan toevoegen in layout.html.twig

{% extends 'base.html.twig' %}  
{% block body %}

En onderaan in die file:

{% endblock %}

Hierdoor vervang je de body van base.html.twig

# Programma schrijven

### Een eenvoudige controller aanmaken.

Een controller is het feitelijke programma dat je schrijft.

Een controller kun je eenvoudig aanmaken met het make:controller commando. Stel dat je een DefaultController wilt maken:

php bin/console make:controller Default

Het programma maakt dan automatisch in de controller folder een DefaultController.php aan en daarnaast maakt hij ook automatisch een folder ‘default’ aan onder templates waar een index.html.twig (gebruikt voor opmaak) komt te staan.

php bin/console make:controller Homepage

maakt evenzo een HomepageController.php en een index.html.twig (gebruikt voor opmaak) onder templates\homepage. De naam van de file moet altijd eindigen op Controller dus hij maakt er zelf HomepageController van in dit voorbeeld

### Crud Controller (Create, Read, Update en Delete)

In ons voorbeeld willen we een CRUD systeem bouwen voor de verschillende tabellen/entities.

Crud is als het goed is al bij het installeren een onderdeel van je symfony 4 project package.

Crud maakt automatisch een:

* controller aan in Controller folder met <naam>Controler.php waar al code in staat met de verschillende routes **(hoofdroute->index ; en ook voor new, show, en edit. Vanuit de TWIG files hieronder worden de functies aangeroepen uit de controller (dit gebeurt door de parameter name die bij iedere Route staat in deze controller)**
* Form aan in Form folder met <naam>Type .php voor de velden van de databasetabel die zichtbaar moeten worden.
* Verschillende TWIG template files in templates<naam> folder.
  + Index.html.twig om alle gegevens van de database te laten zien bij hoofdroute van de controller
  + New.html.twig om een nieuw object toe te voegen en als submitted en valid is ook op te slaan
  + Show.html.twig om een geselecteerd item/object te laten zien
  + Edit.html.twig om een bestaand item te ditten
  + \_form.html.twig welke wordt aangeroepen bij new, edit en show
  + \_delete\_form.html.twig welke wordt aangeroepen bij edit en show.

php bin/console make:crud User

maakt een CRUD systeem voor de betreffende entiteit. Hieronder de files die dan aangemaakt worden:

created: src/Controller/UserController.php

created: src/Form/UserType.php

created: templates/User/\_delete\_form.html.twig

created: templates/User/\_form.html.twig

created: templates/User/edit.html.twig

created: templates/User/index.html.twig

created: templates/User/new.html.twig

created: templates/User/show.html.twig

php bin/console make:crud Truck

maakt een CRUD systeem voor de betreffende entiteit. Hieronder de files die dan aangemaakt worden:

created: src/Controller/TruckController.php

created: src/Form/TruckType.php

created: templates/Truck/\_delete\_form.html.twig

created: templates/Truck/\_form.html.twig

created: templates/Truck/edit.html.twig

created: templates/Truck/index.html.twig

created: templates/Truck/new.html.twig

created: templates/Truck/show.html.twig

php bin/console make:crud Logboek

maakt een CRUD systeem voor de betreffende entiteit. Hieronder de files die dan aangemaakt worden:

created: src/Controller/LogboekController.php

created: src/Form/LogboekType.php

created: templates/Logboek/\_delete\_form.html.twig

created: templates/Logboek/\_form.html.twig

created: templates/Logboek/edit.html.twig

created: templates/Logboek/index.html.twig

created: templates/Logboek/new.html.twig

created: templates/Logboek/show.html.twig

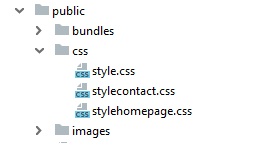
### Een homepage maken

Maak een controller homepage. Als het goed is krijg je dan een HomepageController. Let op dat de route op “/ “ staat dat is de hoofdroute van de website.

HomepageControler.php ziet er dan bijvoorbeeld zo uit.

**<?php  
  
namespace** App\Controller;  
  
**use** Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
**use** Symfony\Component\Routing\Annotation\Route;  
  
**class** HomepageController **extends** AbstractController  
{  
 */\*\*  
 \** ***@Route****("/", name="homepage")  
 \*/* **public function** index()  
 {  
 **return** $this->render(**'homepage/index.html.twig'**, [  
 **'controller\_name'** => **'HomepageController'**,  
 ]);  
 }  
}

Vaak zit hier ook css opmaak en images bij. De CSS en Image file zet je dan in de public folder neer. Hieronder een voorbeeld:



De bijbehorende index.html.twig kan er dan zo uit zien (je vervangt dan het Block body (en title, footer) van de base.html.twig file:

{% **extends 'base.html.twig'** %}  
  
{% **block** title %}Hotel La Caserai!{% **endblock** %}  
  
{% **block** body %}  
<**style**>  
 .**example-wrapper** { **margin**: 1**em auto**; **max-width**: 800**px**; **width**: 95%; **font**: 18**px**/1.5 **sans-serif**; }  
 .**example-wrapper code** { **background**: **#F5F5F5**; **padding**: 2**px** 6**px**; }  
</**style**>  
  
<**div class="example-wrapper"**>  
 <**h1**>{{ **"Hotel La Caserei"** }}</**h1**>  
  
</**div**>  
 {% **block** stylesheets %}  
 <**link rel="stylesheet" href="/css/stylehomepage.css"**>  
 {% **endblock** %}  
<**div class="container"**>  
 <**div id="slider"**>  
 <**figure**>  
 <**img alt="Hotel1" src="../images/slidefoto1.jpg"**>  
 <**img alt="Kamer1" src="../images/slidefoto kamer.jpg"**>  
 <**img alt="Hotel2" src="../images/Slidefoto 3.jpg"**>  
 <**img alt="Hotel3" src="../images/Slidefoto 4.jpg"**>  
 <**img alt="Kamer2" src="../images/slidefoto kamer2.jpg"**>  
 </**figure**>  
 </**div**>  
 <**section class="section"**>  
 <**h2 class="kleur"**>La Caserai</**h2**>  
 <**br**>  
 <**div**>  
 <**p class="intro"**>  
 Het kenmerkende zeilvormige silhouet van Hotel La Caserai is meer dan alleen een prachtig hotel, het is een symbool van het moderne Dubai.<**br**><**br**>  
  
 Maar voor al het wonder dat deze verbluffende structuur biedt wanneer je het eindelijk persoonlijk ziet, is het de service die ervoor zorgt dat Hotel La Caserai zo buitengewoon is.  
 <**br**><**br**>  
 Herhaaldelijk uitgeroepen tot 's werelds meest luxueuze hotel, biedt deze schitterende bestemming u de beste service en ervaringen overal - tot en met een optionele Rolls-Royce met chauffeur, helikoptervluchten vanaf een iconisch helikopterplatform, toegang tot het privéstrand, luxe vrijetijdsbesteding op een adembenemend terras met zwembaden en cabanas  
 <**br**><**br**>  
 De suite-only accommodatie van Hotel La Caserai biedt discrete check-in in uw suite, een privéreceptie op elke verdieping en een groot aantal persoonlijke butlers, elk een warme boodschap van onze ongeëvenaarde gastvrijheid.  
 <**br**><**br**>  
 Kom en ervaar het zelf.</**p**>  
 <**br**>  
 <**br**>  
 <**br**>  
 <**br**>  
 <**hr**>  
 <**h2 class="kleur"**>La Caserai</**h2**>  
 <**br**>  
  
 <**p**>- 16 luxueuze duplex-suites</**p**>  
 <**p**>- Vijf zwembaden (drie buiten, twee binnen) en een privéstrand</**p**>  
 <**p**>- Talise Spa</**p**>  
 <**p**>- Sinbad's Kids Club</**p**>  
 <**p**>- Negen wereldklasse restaurants en bars</**p**>  
 <**p**>- Gratis toegang tot Wild Wadi Waterpark ™ voor de duur van uw verblijf</**p**>  
 <**p**>- Talise Fitness bij La Caserai</**p**>  
 </**div**>  
 </**section**>  
</**div**>  
{% **endblock** %}  
{% **block** footer %}  
  
{% **endblock** %}

### VichUploaderBundle gebruiken.

VichUploaderBundle maakt het mogelijk om images/bestanden te uploaden naar de server en slaat tevens de naam van de image (folder\naam) op in de database.

Eerst installeren met:

composer require vich/uploader-bundle

en dan Y kiezen

In tegenstelling tot de documentatie bij symfony hoef je verder niets toe te voegen voor configuraties (APPkernel).

In de config/bundles.php zie je dat de bundle al geactiveerd is. Er moet onderstaande in staan:

Vich\UploaderBundle\VichUploaderBundle::**class** => [**'all'** => **true**],

Onder config/packages zie je als het goed is een file met naam **vich\_uploader.yaml** die kunnen we ook al aanpassen door het commentaar weg te halen zodat onderstaande er staat.

**vich\_uploader**:  
 **db\_driver**: orm  
  
 **mappings**:  
 **product\_images**:  
 **uri\_prefix**: /images/products  
 **upload\_destination**: **'%kernel.project\_dir%/public/images/products'**

product\_images is een gewone naam die je kan aanpassen maar dan moet je wel in de entiteit later bij imageFile wel dezelfde mapping naam invullen (zie verderop).

De andere regels geven aan waar je de images gaat opslaan.

**Entity aanpassen voor een upload veld:**

Stel dat we een entity location hebben dan moeten we de vichloader in ieder geval bekend maken voor die class .

Onderstaande is een voorbeeld voor de file location.php. De 3 regels (2x use en 1x@) welke groter gemaakt zijn toevoegen aan de entity file (onder src\entity) waar je de uploader wilt toepassen , dit kunnen meerdere entities zijn. In dit geval location.php

namespace App\Entity;  
  
use Doctrine\Common\Collections\ArrayCollection;  
use Doctrine\Common\Collections\Collection;  
use Doctrine\ORM\Mapping as ORM;

use **Symfony\Component\HttpFoundation\File\File**;

use Vich\UploaderBundle\Mapping\Annotation as Vich;

*/\*\**

*\* @ORM\Entity*

*\* @Vich\Uploadable*

*\*/*

class Location

{

*// ...Hier staat al mogelijk van alles in*

}

Dan dezelfde entity aanpassen met de velden **image** (dit komt in de database als veld terug en bevat de naam/url van de image) en **imageFile** (dit is de file zelf welke in de mapping definities worden opgeslagen). Let er op bij mapping dezelfde naam staat als in de vich\_uploader.yaml.

Eventueel bestaande image zaken kan je overschrijven. Ik neem lengte 255 omdat een url best lang kan zijn met foldernamen.

*/\*\**

*\* @ORM\Column(type="string", length=255)*

*\* @var string*

*\*/*

private $image;

*/\*\**

*\* @Vich\UploadableField(mapping="product\_images", fileNameProperty="image")*

*\* @var File*

*\*/*

private $imageFile;

*/\*\**

*\* @ORM\Column(type="datetime")*

*\* @var \DateTime*

*\*/*

private $updatedAt;

*// ...*

En vervolgens de bijbehorende functies/methods (get/set) toevoegen in die file:

public function setImageFile(File $image = null)

{

$this->imageFile = $image;

*// VERY IMPORTANT:*

*// It is required that at least one field changes if you are using Doctrine,*

*// otherwise the event listeners won't be called and the file is lost*

if ($image) {

*// if 'updatedAt' is not defined in your entity, use another property*

$this->updatedAt = new \DateTime('now');

}

}

public function getImageFile()

{

return $this->imageFile;

}

public function setImage($image)

{

$this->image = $image;

}

public function getImage()

{

return $this->image;

}

Als dit gebeurd is kun je de database eigenlijk weer opnieuw maken met deze velden:

php bin/console doctrine:schema:update --force

Dan moeten we er nu nog voor zorgen dat we in de form zelf ook de vichuploader functie gaan gebruiken bij image.

De forms staan onder src\Forms\...... In dit geval LocationType.php

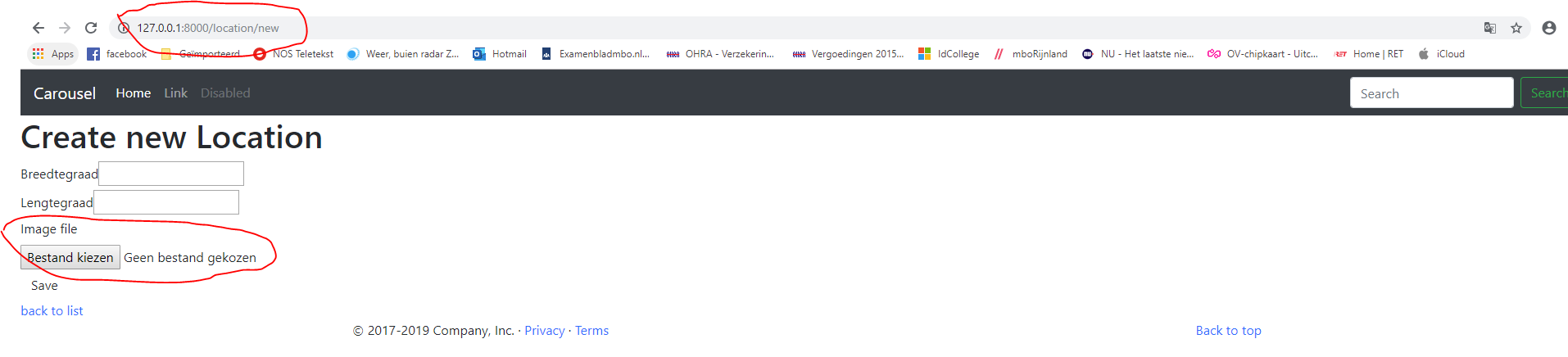
Dit doe je door er in ieder geval voor te zorgen dat dit in de file staat (was bij mij al automatisch)

**use** Vich\UploaderBundle\Form\Type\VichImageType;

en daarnaast moet je de eventuele bestaande add->image vervangen door een VichUpload image.

**public function** buildForm(FormBuilderInterface $builder, **array** $options)  
{  
 $builder  
 ->add(**'breedtegraad'**)  
 ->add(**'lengtegraad'**)  
 *# ->add('image')***->add ('imageFile', VichImageType::class)** ;  
}

Nu zou het moeten werken



Om het plaatje zelf te laten zien in de tabel kun je (als voorbeeld) bij de index.html.twig onderstaande tabel kolom invoegen. Dit was voor een ‘image’ controller

<**td**><**img src=**{{ vich\_uploader\_asset(image, **'imageFile'**) }} **width="300px" height="200px"**></**td**>

### CKEditor gebruiken voor bijvoorbeeld een description veld.

Installeer de bundle:

composer require friendsofsymfony/ckeditor-bundle

klik y

Installeer CKeditor

php bin/console ckeditor:install

Installeer het in de web directory (als je Flex gebruikt)

php bin/console assets:install public

Configureer TWIG (twig: staat er waarschijnlijk al en dan hoef je alleen de onderste 2 regels toe te voegen)

*# Symfony 4: config/packages/twig.yaml*

twig:

form\_themes:

- '@FOSCKEditor/Form/ckeditor\_widget.html.twig'

Inde Form file waarvan je het veld met CKeditor wilt gaan gebruiken (bijv. EventType.php Form met een veld description):

**use** FOS\CKEditorBundle\Form\Type\CKEditorType;

en

Let er op dat description de naam van het veld in de tabel van de database moet zijn.

Stel dat het veld omschrijving heet waarvoor je dit doet moet je in onderstaande code ‘description’ aanpassen naar ‘omschrijving’

->add(**'description'**, CKEditorType::**class**, **array**(  
 **'config'** => **array**(  
 **'uiColor'** => **'#ffffff'** ),  
 ))

### Last user Activity opslaan in een veld van de User entiteit.

Zorg dat je al last\_activity toevoegt aan de FosUser database. Zie hiervoor het eerdere hoofdstuk extra velden toevoegen aan FosUser.

Zie hiervoor het hoofdstuk extra velden toevoegen aan FosUser.

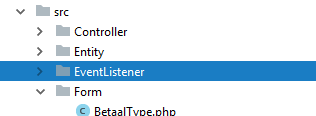
Heb je al andere velden volgens dat hoofdstuk uitgevoerd dan hoef je alleen maar. Het stukje php bin/console make:entity voor User uit te voeren en veld last-activity toe te voegen en daarna het schema updaten met scheme update –force.

Om te voorkomen dat je bij een grote applicatie met veel gebruikers heel veel naar de database geschreven wordt om voor iedere user zijn last\_activity veld te updaten kun je een timer inbouwen die na een update een timer start. Gedurende deze timer doet hij verder geen updates van last\_activity. In dit stuk hieronder is 2 minuten de delay.

Voeg dit toe aan src\Entity\User.php (om het overzichtelijk te houden onder functie setLastActivity.

public function isActiveNow()  
{  
 // Delay during which the user will be considered as still active  
 $delay = new \DateTime('2 minutes ago');  
  
 return ( $this->getLastActivity() > $delay );  
}

Maak onder de folder src een folder EventListener aan.



Maak in de folder Eventlistener een php file aan ActivityListener.php en zet daar onderstaande in

<?php  
namespace App\EventListener;  
use Symfony\Component\Security\Core\Authentication\Token\Storage\TokenStorage;  
use Symfony\Component\Security\Core\SecurityContext;  
use Symfony\Component\HttpKernel\Event\FilterControllerEvent;  
use Symfony\Component\HttpKernel\HttpKernel;  
use Doctrine\ORM\EntityManager;  
use App\Entity\User;  
/\*\*  
 \* Listener that updates the last activity of the authenticated user  
 \*/  
class ActivityListener  
{  
 protected $securityContext;  
 protected $userManager;  
 public function \_\_construct(TokenStorage $securityContext, $userManager)  
 {  
 $this->securityContext= $securityContext;  
 $this->userManager= $userManager;  
 }  
 /\*\*  
 \* Update the user "lastActivity" on each request  
 \* @param FilterControllerEvent $event  
 \*/  
 public function onCoreController(FilterControllerEvent $event)  
 {  
 // Check that the current request is a "MASTER\_REQUEST"  
 // Ignore any sub-request  
 if ($event->getRequestType() !== HttpKernel::*MASTER\_REQUEST*) {  
 return;  
 }  
 // Check token authentication availability  
 if ($this->securityContext->getToken()) {  
 $user = $this->securityContext->getToken()->getUser();  
 if ( ($user instanceof User) && !($user->isActiveNow()) && ($user->isCredentialsNonExpired()) ) {  
 $user->setLastActivity(new \DateTime());  
 $this->userManager->getManager()->flush($user);  
 }  
 }  
 }  
}

Nu alleen nog config\services.yaml aanpassen door onderstaande toe te voegen:

# voor activity listener in config\services.yaml  
# please note that last definitions always \*replace\* previous ones  
App.activity\_listener:  
 class: App\EventListener\ActivityListener  
 arguments: ['@security.token\_storage', '@doctrine']  
 tags:  
 - { name: kernel.event\_listener, event: kernel.controller, method: onCoreController }

### Mail : Swiftmailerfunctie gebruiken.

Installeer de bundle (dit is meestal al gedaan, kijk of je swiftmailer al onder vendor zit). ALLEEN ALS HET NOG NIET GEINTALLEERD IS!!!.

composer require symfony/swiftmailer-bundle

Je kunt een GMAIL adres gebruiken hiervoor. Daar gaan we in dit voorbeeld van uit. (username = [enricokarate@gmail.com](mailto:enricokarate@gmail.com), password = k??????????????l

Pas de .env in de root van je project aan:

*# .env file username is your full Gmail or Google Apps email address password je paswoord*

MAILER\_URL=gmail://enricokarate@gmail.com:password@localhost

Onderstaande hoeven in principe niet

#MAIL\_DRIVER=smtp  
#MAIL\_HOST=smtp.gmail.com  
#MAIL\_PORT=587  
#MAIL\_USERNAME=xxxxxxxxxxx  
#MAIL\_PASSWORD=xxxxxxxxxx  
#MAIL\_ENCRYPTION=ssl

LET OP: als je steeds dezelfde mails stuurt met hetzelfde onderwerp kan Google misschien die mails blokkeren. Daarnaast kunnen mails ook in je spam box komen.

Zorg er ook voor dat je google account berichten van onbekende apps\servers accepteert.

Dit doe je door in te loggen op je google mailaccount en dan:

Instellingen

Accounts en import 🡪 Accountinstellingen wijzigen 🡪Andere instellingen van Googl eAccount

Beveiliging 🡪 naar beneden scrollen tot “toegang voor minder veilige apps” . Die moet je op aan zetten

Let op er zijn meerder switmailer.yaml files!!

**Pas config\packages\swiftmailer.yaml aan** door de spool functie uit te zetten mbv # (Dit betekent dat de mails niet eerst in een buffer/spooler worden klaargezet maar direct worden verstuurd:

**# config\packages\swiftmailer.yaml**

**swiftmailer**:  
 **url**: **'%env(MAILER\_URL)%'** *# spool: { type: 'memory' }*

In je controller code kun je dan een mailfunctie maken. Waarbij je dan eventueel een twig file kan aanroepen waar je de variabelen aan meegeef welke de info bevatten voor het mailbericht.

Voorbeeld van de code in de controller:

**<?php  
  
namespace** App\Controller;  
  
**use** Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
**use** Symfony\Component\Routing\Annotation\Route;  
**use** Symfony\Component\HttpFoundation\Session\SessionInterface;  
  
**class** AfbetalenController **extends** AbstractController  
{  
 **private $session**;  
  
 **public function** \_\_construct(SessionInterface $session)  
 {  
 $this->**session** = $session;  
 }  
  
   
  
 */\*\*  
 \** ***@Route****("/afbetalen", name="afbetalen")  
 \*/* **public function** index( \Swift\_Mailer $mailer)  
 {  
 *#bijvoorbeeld een shopping cart variabele van de sessie* $getCart = $this->**session**->get(**'Cart'**);  
 *#gegevens van een specifieke kolom van de shoppingcart hierboven* $specifiekekolom = *array\_column*($getCart, **'omschrijving'**);  
 *# gewoon zelf een string meegeven van bijvoorbeeld een naam* $naamstring = **"Giovanni van Ravesteijn"**;  
 *# bijvoorbeeld de naamvan de gebruiker. Let wel er moet iemand ingelogd zijn!!! anders foutmelding met deze code* $usersessie = $this->getUser();  
 $naamusersessie = $usersessie->getUsername();  
 dump($getCart);  
 $message = (**new** \Swift\_Message(**'Dit is 6e mail'**))  
 ->setFrom(**'windyenemiel@gmail.com'**)  
 ->setTo(**'windyenemiel@hotmail.com'**)  
 ->setBody(  
 $this->renderView(  
 *// templates/hello/email.txt.twig* **'mail/email.txt.twig'**,  
 [**'variabele1'** => $getCart , **'variabele2'** => $specifiekekolom, **'variabele3'** => $naamstring, **'variabele4'** => $naamusersessie]  
 )  
 )  
 ;  
 *# dump($message);* $mailer->send($message);  
  
 **return** $this->redirectToRoute(**'homepage'**);  
 }  
}

Hieronder nog voorbeeld van de bijbehorende TWIG file waarin de variabelen worden gebruikt.

*{# templates/emails/email.txt.twig #}*Hi met variabele drie {{ variabele3 }}! Je bestelling was succesvol  
Hi met variabele vier {{ variabele4 }}! Je bestelling was succesvol  
  
  
You ordered the following items  
  
  
Id Naam Omschrijving Prijs Aantal Totaal Bedrag  
{% **for** product **in** variabele1 %}  
{{ product.id }} {{ product.naam }} {{ product.omschrijving }} € {{ product.prijs }} {{ product.aantal }} € {{ product.prijs \* product.aantal }}  
  
 {% **else** %}  
  
 U heeft niets besteld  
  
 {% **endfor** %}  
  
Thanks!  
  
Wil je ook een specfieke waarde uit een array index 0 (in dit geval van kolom cart) dan variabele 2 is {{ variabele2[0] }}

### Pagination

Installeer knp pagination

composer require knplabs/knp-paginator-bundle

Creeer config file paginator.yaml onder config/packages/ met onderstaande inhoud

# config/packages/paginator.yaml

knp\_paginator:

    page\_range: 5                       # number of links showed in the pagination menu (e.g: you have 10 pages, a page\_range of 3, on the 5th page you'll see links to page 4, 5, 6)

    default\_options:

        page\_name: page                 # page query parameter name

        sort\_field\_name: sort           # sort field query parameter name

        sort\_direction\_name: direction  # sort direction query parameter name

        distinct: **true**# ensure distinct results, useful when ORM queries are using GROUP BY statements

        filter\_field\_name: filterField  # filter field query parameter name

        filter\_value\_name: filterValue  # filter value query paameter name

    template:

        pagination: '@KnpPaginator/Pagination/sliding.html.twig'     # sliding pagination controls template

        sortable: '@KnpPaginator/Pagination/sortable\_link.html.twig' # sort link template

        filtration: '@KnpPaginator/Pagination/filtration.html.twig'  # filters template

Zorg er dan voor dat je controller de query results van hetgeen je wilt uitdraaien op het scherm nog een keer door de pagination code haalt. LET OP DIT IS EEN VOORBEELD, alleen het gele gaat het om!!!Hieronder een voorbeeld.

<?php

namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;

use Symfony\Component\HttpFoundation\Request;

use App\Entity\Appointments;

// Include paginator interface

use Knp\Component\Pager\PaginatorInterface;

class DefaultController extends AbstractController

{

    // Include the paginator through dependency injection, the autowire needs to be enabled in the project

    public function index(Request $request, PaginatorInterface $paginator , AppointmentRepositroy $appointmentRepository)

    {

        // Retrieve the entity manager of Doctrine

        $em = $this->getDoctrine()->getManager();

        // Find all the data on the Appointments table, filter your query as you need

        $allAppointmentsQuery = $appointmentRepository->createQueryBuilder('p')

            ->where('p.status != :status')

            ->setParameter('status', 'canceled')

            ->getQuery();

        // Paginate the results of the query

        $jouwnieuwevariabele = $paginator->paginate(

            // Doctrine Query, not results

            $allAppointmentsQuery,

            // Define the page parameter

            $request->query->getInt('page', 1),

            // Items per page

            5

        );

        // Render the twig view

        return $this->render('default/index.html.twig', [

            'appointments' => $jouwnieuwevariabele

        ]);

    }

}

Creeer dan de paginator in je twigtemplate. Let op dit is een voorbeeld en gaat alleen om gele stuk

{# ./templates/default/index.html.twig #}

<!DOCTYPE html>

<html>

    <head>

        <meta charset="UTF-8">

        <title>My Application</title>

    </head>

    <body>

        <ul>

        {% for appointment in appointments %}

            <li>

                {{ appointment.title }}

            </li>

        {% endfor %}

        </ul>

        <div class="navigation">

            {{ knp\_pagination\_render(appointments) }}

        </div>

    </body>

</html>

Je kunt nu controleren of het allemaal werkt.

Als het werkt kun je de layout van de paginator wat mooier te maken kun je in de file

# config/packages/paginator.yaml

Bijvoorbeeld de bootstrap 4 paginator opmaak zetten als je bootstrap 4 gebruikt.

Vervang dan

pagination: '@KnpPaginator/Pagination/sliding.html.twig'     # sliding pagination controls template

door

pagination: '@KnpPaginator/Pagination/twitter\_bootstrap\_v3\_pagination.html.twig'     # sliding pagination controls template

### Samengestelde Forms

Dit hoofdstuk is nog niet af niet gebruiken!!!!!!!!!!!!!!!!!

In de kamercontroller.php (ook in namespace nog zaken toevoegen

*/\*\*  
 \** ***@Route****("/new", name="kamer\_new", methods={"GET","POST"})  
 \*/* **public function new**(Request $request): Response  
 {  
 $kamer = **new** Kamer();  
*# $image1 = new Image();  
# $kamer->getImages()->add($image1);* $form = $this->createForm(KamerType::**class**, $kamer);  
 $form->handleRequest($request);  
  
 **if** ($form->isSubmitted() && $form->isValid()) {  
 $entityManager = $this->getDoctrine()->getManager();  
 $entityManager->persist($kamer);  
 *# $entityManager->persist($image1);* $entityManager->flush();  
  
 **return** $this->redirectToRoute(**'kamer\_index'**);  
 }  
  
 **return** $this->render(**'kamer/new.html.twig'**, [  
 **'kamer'** => $kamer,  
 **'form'** => $form->createView(),  
 ]);  
 }

In de form KamerType

**<?php  
  
namespace** App\Form;  
  
**use** App\Entity\Kamer;  
**use** Symfony\Component\Form\AbstractType;  
*# use Symfony\Component\Form\Extension\Core\Type\CollectionType;***use** Symfony\Component\Form\FormBuilderInterface;  
**use** Symfony\Component\OptionsResolver\OptionsResolver;  
  
**class** KamerType **extends** AbstractType  
{  
 **public function** buildForm(FormBuilderInterface $builder, **array** $options)  
 {  
 $builder  
 ->add(**'prijs'**)  
 ->add(**'soortid'**)  
 ->add(**'prijs'**)  
 ;  
  
 *# $builder->add('images', CollectionType::class, [  
 # 'entry\_type' => ImageType::class,  
 # 'entry\_options' => ['label' => false],  
 # ]);* }  
  
 **public function** configureOptions(OptionsResolver $resolver)  
 {  
 $resolver->setDefaults([  
 **'data\_class'** => Kamer::**class**,  
 ]);  
 }  
}

### Globale session parameter

Je kunt voor symfony gebruik maken van de session parameter waar je andere variabelen in op kan slaan.

Onderstaand hetgeen je dan op moet nemen in je programma/functie

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| // haal de gegevens van de huidige sessie parameter op |
|  |
| $session = $this->get('request\_stack')->getCurrentRequest()->getSession();  //zet variabelen in de $session parameter  $session->set(‘<geef een unieke naam>’, $variabelenaam);  Voorbeelden waarbij vrijekamers een array is en checkin, checkuit date:  $session->set(‘checkin’, $checkindatum);  $session->set(‘checkuit’, $checkuitdatum);  $session->set(‘kamersvrij’, $vrijekamers);  //haal variabelen gegeens uit de $session parameter  $variabelenaam = $session->get(‘naam ’, $variabelenaam);  Voorbeelden:  $checkindatum = $session->get(‘checkin’);  $checkuitdatum = $session->get(‘checkuit’);  $vrijekamers = $session->get(‘kamersvrij’, array())  $session->set(‘kamersvrij’, $vrijekamers);  // variabelen uit de session parameter halen , is netjes programmeren maar hoeft niet perse  Voorbeelden |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| $session->remove('checkin');  $session->remove('checkuit');  $session->remove('kamersvrij'); |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

### Wegschrijven van gegevens in de database

Let op het voorbeeld gebruikt verzonnen namen , je moet altijd de namen van je eigen database gebruiken en de variabelen uit je programma.

Voorbeeld om een reservering met checkindatum, checkuitdatum,kamernummer en klant op te slaan:

// haal de doctrine database op

$em = $this->getDoctrine()->getManager();

// maak een nieuw object voor reservering aan

$reservering = new Reservering();

// zet nu de waarden van de variabelen in het nieuwe object

$reservering->setKamernummer($kamers['nummer']);

$reservering->setCheckinDatum($checkindate);

$reservering->setCheckuitDatum($checkuitdate);

$reservering->setUser($this->getUser());

// schrijf de gegevens in de database

$em->persist($reservering);

$em->flush();

### Extra: Overzetten naar andere computer van symfony project

* Kopieer de folder van het project op computer 1 naar een USB stick (Xampp\htdocs\<projectnaam>)
* Zoek op computer 1 via Xampp-panel MySql->admin naar de database van het project
* Klik op de Database en kies dan in het menu : Exporteren . Geef eventueel template een naam en druk op starten. Er wordt dan een file met de naam <database>.sql ergens op computer 1 gezet. Ik denk in de download folder. Anders moet je het bestand openen en via Save-As kun je zien waar het staat. Zet dit bestand ook op de USB stick.
* Haal na het kopieren de USB stick er uit en koppel die op computer 2
* Kopieer de folder van USB naar computer2 in de xampp\htdocs folder
* Ga naar Xampp SQL Admin panel en kijk of er al een database met die name aanwezig is.
* Zo nee maak een database aan (zo ja benader expert) met de exacte naam van de oorspronkelijke database. Dit doe je via linkermenu “nieuw” of op tabblad databse te klikken. Type de nieuwe naam en kies bij collatie voor utf8mb4\_unicode\_ci
* Importeer nu de database welke je via exporteren in een file had gezet en op de usb stick staat. Ga hiervoor op de datbase staan ; klik importeren; ga bij bestand kiezen naar het bestand op je USB stick <database>.sql en als je die hebt geselecteerd druk je op starten. De databse wordt nu ingelezen.

# Foutmeldingen:

Algemeen:

Mbv een commando **dump($variabele);** kun je zien wat de waarde van een variabele is.

Soms zie je dat niet door een redirect die gelijk komt.

Je kunt dat tijdelijk voor debuggen even wel zichtbaar te maken door even te stoppen voor de redirect zodat je de zaken kan opvragen. Door in onderstaande file de intercept parameter op true te zetten (default staat hij op false)

|  |  |
| --- | --- |
| 1  2  3 | *# config/packages/dev/web\_profiler.yaml*  web\_profiler:  intercept\_redirects: true |

CASE MISMATCH

Na het creeren van de Entities kreeg ik foutmeldingen:

Case mismatch between loaded and declared class names: "App\Entity\kamer" vs "App\Entity\Kamer".

Dan is er doordat je relaties gelegd heb iets misgegaan met de hoofdletter / kleine letter van kamer (of andere velden).

In de PHP file kijken van de entitiy met relatie en dan de hoofdletter/kleine letter aanpassen.

Voorbeeld:

/\*\*  
 \* @ORM\ManyToOne(targetEntity="App\Entity\kamer", inversedBy="reserverings")  
 \* @ORM\JoinColumn(nullable=false)  
 \*/  
private $kamerid;

Moet worden;

/\*\*  
 \* @ORM\ManyToOne(targetEntity="App\Entity\**K**amer", inversedBy="reserverings")  
 \* @ORM\JoinColumn(nullable=false)  
 \*/  
private $kamerid;

Blijkbaar dus case sensitive.

Catchable Fatal Error: Object of class App\Entity\Soort could not be converted to string

Fatal error met string:

Foutmeldingen later toen moest je bij de Soort(of de andere welke in de foutmelding staat) entity de onderstaande functie toevoegen

public function \_\_toString()  
{  
 return strval($this->id);  
}

Hallo,

<https://stackoverflow.com/questions/56551896/issues-with-symfony-4-3-1-and-fosuserbundle-2-1-2>

|  |  |
| --- | --- |
| [https://cdn.sstatic.net/Sites/stackoverflow/img/apple-touch-icon@2.png?v=73d79a89bded](https://stackoverflow.com/questions/56551896/issues-with-symfony-4-3-1-and-fosuserbundle-2-1-2) | [php - Issues with Symfony 4.3.1 and FosUserBundle 2.1.2 - Stack Overflow](https://stackoverflow.com/questions/56551896/issues-with-symfony-4-3-1-and-fosuserbundle-2-1-2)  For a new project, I use Symfony 4.3 and FosUserBundle 2.1. I installed with success this bundle but when I try to use it, I've got somes issues.  stackoverflow.com |

Deze link geeft je de oplossing voor het FOS User probleem wat in SF 4.3.1 is gekomen.

Deze problemen zijn ontstaan door het toevoegen van functionaliteiten in FOS-User bundle, de listing in deze stack lost dat op.

Groeten,

Henry

Moet nog plaatje van vichuploader image manier bij

Nog de bootstrap base.html.twig toevoegen

En eventueel mail over probleem met nieuwe symfony en globale variabele.