Rendering a template

```php
class VinylController extends AbstractController
{
    #[Route('/', name: 'app_homepage')]
    public function homepage(): Response
    {
        $tracks = [
            ['song' => 'Gangsta\'s Paradise', 'artist' => 'Coolio'],
            ['song' => 'Waterfalls', 'artist' => 'TLC'],
            ['song' => 'Creep', 'artist' => 'Radiohead'],
            ['song' => 'Kiss from a Rose', 'artist' => 'Seal'],
            ['song' => 'On Bended Knee', 'artist' => 'Boyz II Men'],
            ['song' => 'Fantasy', 'artist' => 'Mariah Carey'],
        ];

        return $this->render( view: 'vinyl/homepage.html.twig', [
            'title' => 'PB & Jams',
            'tracks' => $tracks,
        ]);
    }
```

Navigeren naar een route (1)

Gebruik de path() functie om URL links te maken

```php
#[Route('/', name: 'blog_index')]
public function index(): Response
{
    // ...
}
```

```twig
<a href="{{ path('blog_index') }}">Homepage</a>
```

Navigeren naar een route (2)


```
#[Route('/smartphone/details/{id}', name: 'app_smartphone_details')]
```

```
{% for smartphone in smartphones %}
    <tr>
        <td>{{ i }}</td>
        <td>{{ smartphone.vendor.name}}</td>
        <td>{{ smartphone.type}}</td>
        <td class="text-center"><a href="{{ path('app_smartphone_details',{id:smartphone.id }) }}"><i class="bi-search"></i></a></td>
        <td class="text-center"><a href="{{ path('app_smartphone_edit',{id:smartphone.id }) }}"><i class="bi-pencil-square"></i></a></td>
        <td class="text-center"><a href="{{ path('app_smartphone_delete',{id:smartphone.id}) }}"><i class="bi-trash3"></i></a></td>
    </tr>
    {% set i=i+1 %}
{% endfor %}
```

Tonen producten

```
{% for product in products %}
    <div class="col-md-4">
        <div class="card mb-4">
            <img src="https://dummyimage.com/200x150" alt="{{ product.name }}"
                 class="card-img-top">
            <div class="card-body">
                <h5 class="card-title">{{ product.name }}</h5>
                <p class="card-text">{{ product.description }}</p>
                <div class="d-flex justify-content-between align-item-center">
                    <a href="{{ path('app_product_add',{id:product.id}) }}"
                       class="btn btn-dark">buy</a>
                    <span class="h5 mt-auto">{{ product.price }} €</span>
                </div>
            </div>
        </div>
    </div>
{% endfor %}
```

```
#add product to cart
#[Route('/product/add/{id}', name: 'app_product_add')]
```

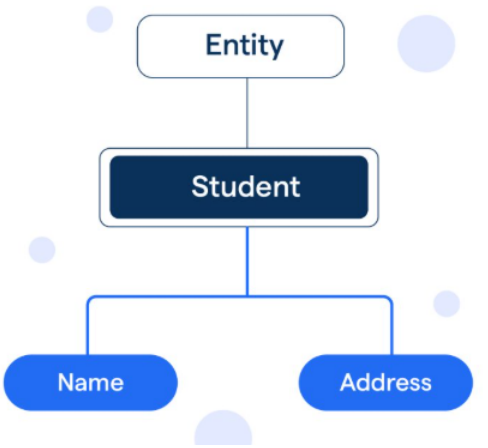
Maken van een nieuwe entity 1

- Een nieuwe entity aanmaken kun je met een commando doen op de terminal:

```
symfony console make:entity
```

- Of

```
symfony console make:entity [EntityName]
```



Migratie aanmaken

- Heb je al je entities aangemaakt met alle velden (properties) die je wilt hebben moet je ze naar de database migreren. Hiervoor moet je eerst een migratie aanmaken.
- Voer hiervoor het volgende commando uit:
- `symfony console make:migration`

- Dit commando zorgt ervoor dat voor alle wijzigingen in de entities worden toegevoegd aan een migratie bestand in de map /migrations.

Migratie uitvoeren

- Nadat je het migratiebestand hebt aangemaakt moet je de daadwerkelijke migratie ook nog uitvoeren. Dit doe je met het commando:
- `symfony console doctrine:migrations:migrate`

- Na het uitvoeren van dit commando zullen alle nog niet gemigreerde migratiebestanden gemigreerd worden naar de database.

- Belangrijk: check altijd even met PHPMyAdmin of je migratie goed gelukt is. Zijn de tabellen toegevoegd? Zijn jouw wijzigingen doorgevoerd?

Aanmaken database

- Ga nu de gastenboek database configureren in het `.env` bestand

```
DATABASE_URL="mysql://root:@127.0.0.1:3306/gastenboek?serverVersion=10.4.32-MariaDB&charset=utf8mb4"
```

```
$ php bin/console doctrine:database:create
```

	Database	Collatie	Actie
☐	cijfersysteem	utf8mb4_general_ci	Controleer rechten
☐	fietsenmaker	utf8mb4_general_ci	Controleer rechten
☐	gastenboek	utf8mb4_general_ci	Controleer rechten

Databaseserver

- Server: 127.0.0.1 via TCP/IP
- Servertype: MariaDB
- Serververbinding: SSL wordt niet gebruikt
- Serverversie: 10.4.32-MariaDB - mariadb.org binary distribution
- Protocolversie: 10
- Gebruiker: root@localhost
- Karakterset van server: UTF-8 Unicode (utf8mb4)

Hoe create ik een entity in Symfony

- Hieronder een code voorbeeld voor het aanmaken van een entiteit.

```php
public function create(EntityManagerInterface $entityManager){
    //creating a new instance of the entity Ingredient
    $ingredient = new Ingredient();
    //set name on entity
    $ingredient->setName( name: 'Test');
    //persist the entity into the entitymanager
    $entityManager->persist($ingredient);
    //flush changes stored in the entitymanager and actually store them in the database
    $entityManager->flush();
}
```

Hoe maak ik een form in Symfony?

- In Symfony bestaat er een commando die je kunt uitvoeren op de terminal om een form te maken.

```
PS C:\xampp\htdocs\sd22-p07-symfony-oefen-b3m-mkleijwegt> symfony console make:form

 The name of the form class (e.g. OrangePopsicleType):
 >
```

- Het commando is:
  - `symfony console make:form`

- Op de terminal zal er gevraagd worden hoe het formulier moet heten. Vaak gebruik je de naam van de entity + type. Bijvoorbeeld: CustomerType, StudentType, IngredientType.

- Ook wordt er gevraagd voor welke entity het formulier is bedoeld. Geef dan de naam op van de entity. Bijvoorbeeld: User, Customer, Student.

Het maken van een formulier voor de Smartphone tabel

1 En dan nu het formulier maken

```
PS C:\xampp\htdocs\sd-p7-symfony-smartphone4u> php bin/console make:form

 The name of the form class (e.g. GrumpyGnomeType):
 > SmartphoneType

 The name of Entity or fully qualified model class name that the new form will be bound to (empty for none):
 > Smartphone

 created: src/Form/SmartphoneType.php


  Success!


 Next: Add fields to your form and start using it.
 Find the documentation at https://symfony.com/doc/current/forms.html
PS C:\xampp\htdocs\sd-p7-symfony-smartphone4u>
```

1 Open SmartphoneType.php. Hier komt de html per input veld te staan.
Bij de velden staan geen input types (bv <input type="text">)
Symfony 'gokt' aan de hand van de entity Smartphone wat voor input element je nog hebt.

```php
class SmartphoneType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add( child: 'vendor')
            ->add( child: 'type')
            ->add( child: 'memory')
            ->add( child: 'color')
            ->add( child: 'price')
            ->add( child: 'decription')
            ->add( child: 'picture')
        ;
    }
```

Rendering Forms

```php
class SmartphoneController extends AbstractController
{
    #[Route('/smartphone/add', name: 'app_smartphone')]
    public function index(): Response
    {
        $smartphone = new Smartphone();
        $form=$this->createForm( type: SmartphoneType::class, $smartphone);

        return $this->render( view: 'smartphone/index', [
            'controller_name' => 'SmartphoneController',
            'form' => $form,
        ]);
    }
}
```

```twig
{% extends 'base.html.twig' %}

{% block title %}Hello SmartphoneController!{% endblock %}

{% block body %}
<h1>Insert Smartphone</h1>
    {{ form(form) }}
{% endblock %}
```

```php
#[Route('/ingredient-add', name: 'app_add_ingredient')]
public function addIngredient(Request $request, EntityManagerInterface $entityManager): Response
{
    $form = $this->createForm( type: IngredientType::class);
    $form->handleRequest($request);
    if($form->isSubmitted() && $form->isValid()) {
        $ingredient = $form->getData();
        $entityManager->persist($ingredient);
        $entityManager->flush();

        $this->addFlash( type: 'success', message: 'Ingredient is toegevoegd');

        return $this->redirectToRoute( route: 'app_ingredient');
    }
    return $this->render( view: 'ingredient/add.html.twig', [
        'form' => $form,
    ]);
}
```

```php
#[Route('/ingredient-update/{id}', name: 'app_update_ingredient')]
public function updateIngredient(Request $request, EntityManagerInterface $entityManager, int $id): Response
{
    $ingredient = $entityManager->getRepository(Ingredient::class)->find($id);
    $form = $this->createForm( type: IngredientType::class, $ingredient);
    $form->handleRequest($request);
    if($form->isSubmitted() && $form->isValid()) {
        $ingredient = $form->getData();
        $entityManager->persist($ingredient);
        $entityManager->flush();

        $this->addFlash( type: 'success', message: 'Ingredient is bijgewerkt');

        return $this->redirectToRoute( route: 'app_ingredient');
    }
    return $this->render( view: 'ingredient/update.html.twig', [
        'form' => $form,
    ]);
}
```

```php
namespace App\Controller;

use App\Entity\Smartphone;
use App\Form\SmartphoneType;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class SmartphoneController extends AbstractController
{
    #[Route('/smartphone/add', name: 'app_smartphone')]
    public function index(Request $request): Response
    {
        $smartphone = new Smartphone();
        $form=$this->createForm( type: SmartphoneType::class, $smartphone);

        $form->handleRequest($request);
        if ($form->isSubmitted() && $form->isValid()) {
            $smartphone = $form->getData();
            dd($smartphone);
        }
```

Processing Forms