

TD Symfony

Étape 7 : Une page d'accueil dynamique

Dans l'étape précédente, nous avons créé nos entités et les tables en base de données pour la page d'accueil (avec Maker et Doctrine)

Nous allons voir comment « injecter » ces données dans notre template twig.

Commençons par créer nos catégories en base de données. Récupérer le script d'insertion sql sur le git : https://github.com/laurentbedu/td_symfony_5/tree/TD-7/docs/sql

Récupérer également les images sur git :

https://github.com/laurentbedu/td_symfony_5/tree/TD-7/public/img

Nous modifions notre HomeController pour récupérer les catégories et les passer au template.

```
src > Controller > 🐘 HomeController.php > ...
1  <?php
2
3  namespace App\Controller;
4
5  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
6  use Symfony\Component\HttpFoundation\Response;
7  use Symfony\Component\Routing\Annotation\Route;
8  use App\Entity\Category;
9
10 class HomeController extends AbstractController
11 {
12     /**
13      * @Route("/home", name="home")
14      * @Route("/", name="root")
15      */
16     public function index(): Response
17     {
18         $categories = $this->getDoctrine()
19             ->getRepository(Category::class)
20             ->findAll();
21         return $this->render('home/index.html.twig', [
22             'categories' => $categories,
23         ]);
24     }
25 }
```

Nous modifions également notre template. Nous ne gardons qu'une « card » et bouclons sur notre variable categories.

```

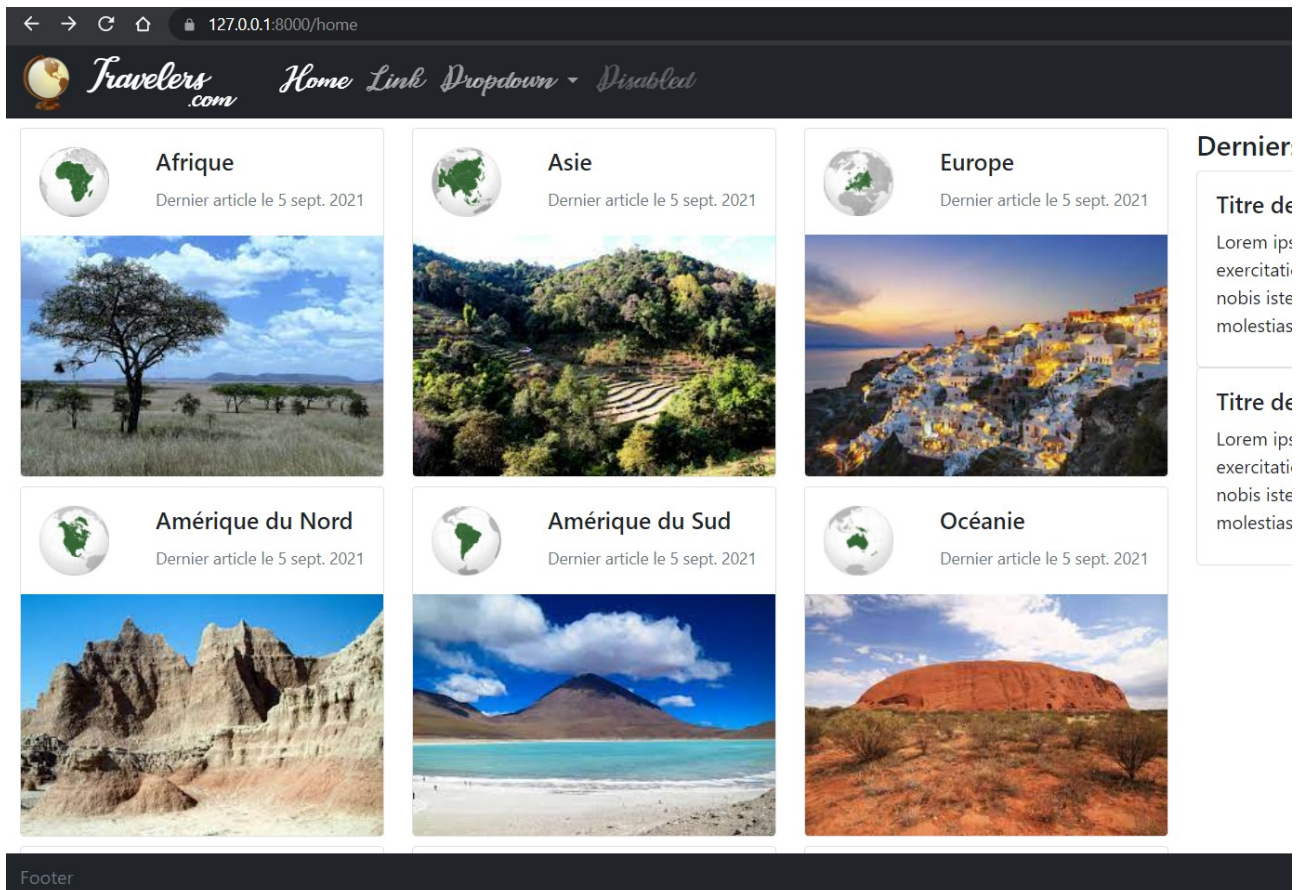
templates > home > index.html.twig
8
9 {% block content %}
10 <div class="container-fluid">
11   <div class="row pt-2">
12     <div class="col-12 col-lg-8">
13       <div class="row">
14         {% for category in categories %}
15           <div class="col-12 col-lg-4">
16             <div class="card category_card ml-1 mb-2" onclick="window.location='categories/{{ category.id }}/{{ category.title }}'">
17               <div class="card-body">
18                 <div class="d-flex justify-content-between">
19                   <div class="category_globe">
20                     
21                   </div>
22                   <div>
23                     <h5 class="card-title">{{ category.title }}</h5>
24                     <p class="card-text"><small class="text-muted">Dernier article le 5 sept. 2021</small></p>
25                   </div>
26                 </div>
27               </div>
28             <div class="category_landscape">
29               
30             </div>
31           </div>
32         {% endfor %}
33       </div>
34     </div>
35   <div class="col-12 col-lg-4 order-first order-lg-last">
36

```

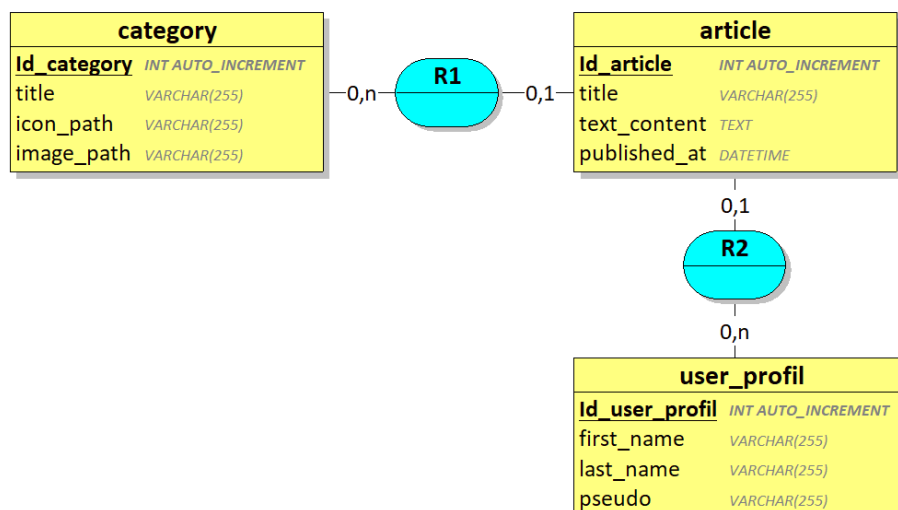
Un peu de css ...

```
public > css > # base.css > body
1  body{
2      padding-top: 62px;
3      margin-bottom: 62px;
4  }
5
```

Voilà la résultat



Nous avons besoin d'afficher les articles par date avec le nom de leur auteur. Nous allons devoir modifier la base de données. Voici le MCD correspondant :



Un user_profil (correspondant à un utilisateur que nous créerons plus tard) peut avoir écrit plusieurs articles, mais un article ne possède qu'un seul auteur.

Nous créons l'entité UserProfile, modifions l'entité Article (published_at) et nous ajoutons la relation OTM/MTO entre ces 2 entités.

Commençons par créer notre entité UserProfile

```
\td_symfony> php bin/console make:entity
```

[...]

```
updated: src/Entity/UserProfil.php
```

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

```
Success!
```

Nous modifions ensuite Article

```
\td_symfony> php bin/console make:entity Article
```

```
Your entity already exists! So let's add some new fields!
```

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

```
> published_at
```

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

```
>
```

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

```
> yes
```

```
updated: src/Entity/Article.php
```

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

```
>
```

```
Success!
```

Dans le TD6 nous avons réalisé une relation OTM/MTO en faisant le MTO en premier, cette fois-ci nous allons commencer par OTM. Nous allons modifier l'entité UserProfile

```
\td_symfony> php bin/console make:entity UserProfile
```

```
Your entity already exists! So let's add some new fields!
```

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

```
> articles
```

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

```
> OneToMany
```

```
What class should this entity be related to?:
```

```
> Article
```

```
A new property will also be added to the Article class so that it can be accessed from it.
```

```
New field name inside Article [userProfil]:
```

```
>
```

```
Is the Article.userProfil property allowed to be null (null if yes):
```

```
>
```

```
updated: src/Entity/UserProfil.php
```

```
updated: src/Entity/Article.php
```

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

```
>
```

```
Success!
```

Nous préparons la migration, puis la réalisons avec doctrine.

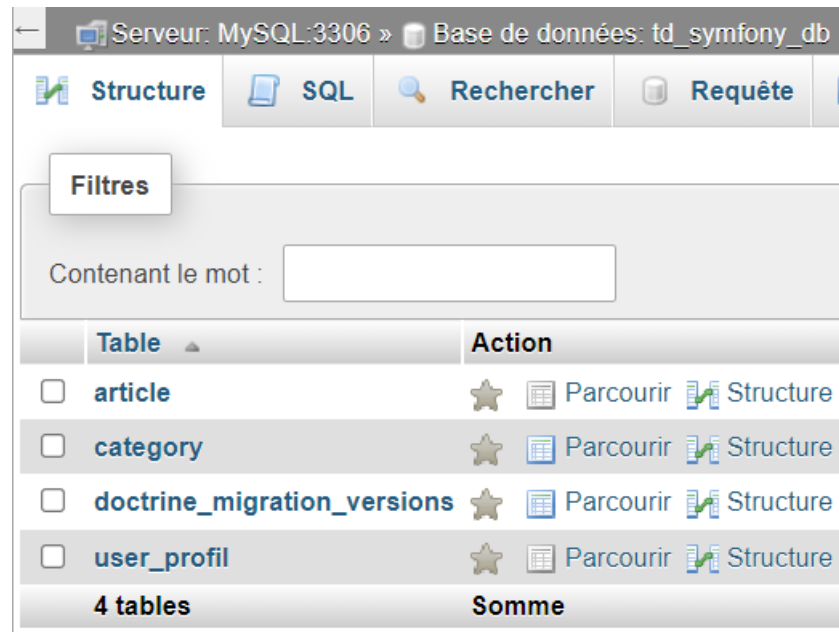
```
\td_symfony> php bin/console make:migration
```

```
\td_symfony> php bin/console doctrine:migrations:migrate
```

```
[notice] Migrating up to DoctrineMigrations\Version20210922032041
```

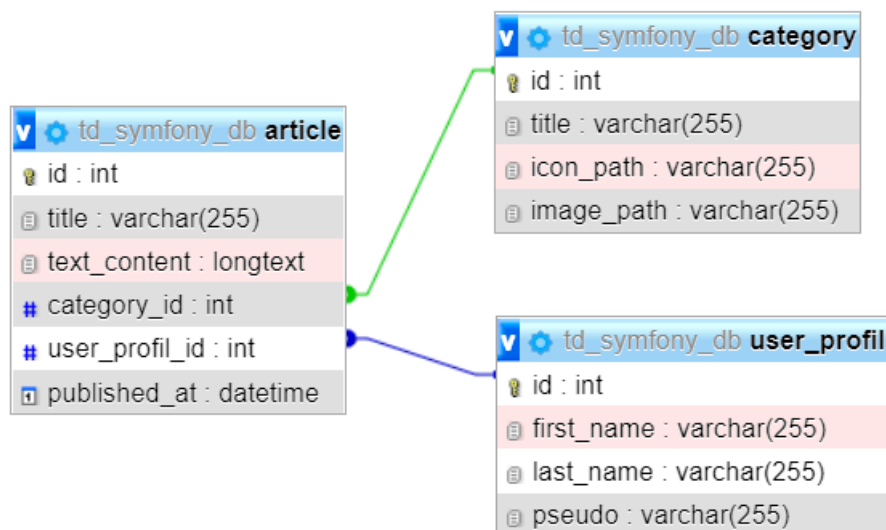
```
[notice] finished in 544.6ms, used 20M memory, 1 migrations executed, 4 sql queries
```


Un petit coup d'œil dans phpMyAdmin



The screenshot shows the phpMyAdmin interface for a MySQL server. The top navigation bar includes 'Structure', 'SQL', 'Rechercher', and 'Requête'. Below this is a 'Filtres' section with a search box labeled 'Contenant le mot :'. The main content area displays a table of database tables:

Table	Action
<input type="checkbox"/> article	★ Parcourir Structure
<input type="checkbox"/> category	★ Parcourir Structure
<input type="checkbox"/> doctrine_migration_versions	★ Parcourir Structure
<input type="checkbox"/> user_profil	★ Parcourir Structure
4 tables	Somme



Nous allons créer 2 user_profil en base de données et 4 articles pour avant de continuer le développement de notre page d'accueil afin de pouvoir visualiser le résultat.

Récupérez les scripts sur git :

https://github.com/laurentbedu/td_symfony_5/tree/TD-7/docs/sql

Une fois les données insérées (user_profil et article), nous allons pouvoir continuer notre page d'accueil.

Nous allons récupérer la date du dernier article publié pour chaque catégorie. Nous modifions l'entité Category en triant ses articles par date de publication descendante.

```
src > Entity > Category.php > Category
34
35     private $image_path;
36
37     /**
38      * @ORM\OneToMany(targetEntity=Article::class, mappedBy="category")
39      * @ORM\OrderBy({"published_at" = "DESC"})
40      *
41     private $articles;
```

On crée également une méthode permettant de récupérer le dernier article publié.

```
src > Entity > Category.php > Category > getLastPublishedArticle
119
120     public function getLastPublishedArticle() : ?Article
121     {
122         if(!$this->getArticles()->isEmpty() && $this->getArticles()->first()->getPublishedAt()){
123             return $this->getArticles()->first();
124         }
125         return null;
126     }
127
128
129 }
```

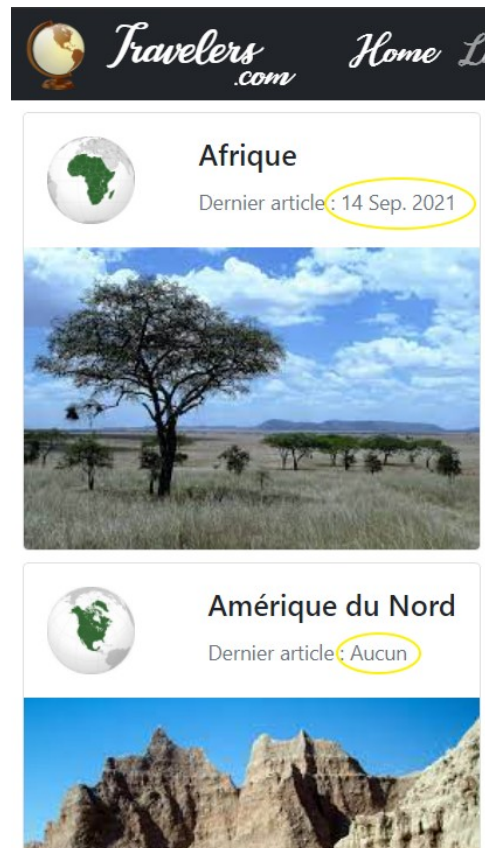
Nous modifions ensuite notre entité Article en y ajoutant un méthode permettant de convertir la date de publication en string afin de l'afficher dans notre template

```
src > Entity > Article.php > Article > getPublishedDateString
109
110     public function getPublishedDateString() : ?string
111     {
112         return $this->getPublishedAt() ? $this->getPublishedAt()->format("d M. Y") : null;
113     }
114
115
116 }
117
```

Il nous reste à modifier le template

```
templates > home > index.html.twig
23     <h5 class="card-title">{{ category.title }}</h5>
24     <p class="card-text">
25         <small class="text-muted">
26             Dernier article :
27             {{ category.getLastPublishedArticle() ?
28                 category.getLastPublishedArticle().getPublishedDateString() ?? 'Aucun' : 'Aucun' }}
29         </small>
30     </p>
```

Voici ce que ça donne



Nous allons maintenant récupérer les derniers articles publiés pour les afficher dans la partie droite de notre Home page. Nous avons besoin de ne récupérer que les articles publiés (`published_at != null`) en les triant pare date de publication décroissante. Nous créons dans la classe `ArticleRepository` une méthode pour faire ce travail.

```
src > Repository > ArticleRepository.php > ArticleRepository > findAllPublished

15 class ArticleRepository extends ServiceEntityRepository
16 {
17     public function __construct(ManagerRegistry $registry)
18     {
19         parent::__construct($registry, Article::class);
20     }
21
22     public function findAllPublished($maxResults = 5)
23     {
24         $qb = $this->createQueryBuilder('art');
25         return $qb->where($qb->expr()->neq('art.published_at', '?1'))
26             ->setParameter(1, new \DateTimeImmutable())
27             ->orderBy('art.published_at', 'DESC')
28             ->setMaxResults($maxResults)
29             ->getQuery()
30             ->getResult();
31     }
}
```


Puis nous récupérons les articles pour les passer au template dans notre contrôleur grâce à cette méthode

```
src > Controller > 🐘 HomeController.php > 🏠 HomeController > 🏠 index

14      /**
15      * @Route("/home", name="home")
16      * @Route("/", name="root")
17      */
18      public function index(): Response
19      {
20          $categories = $this->getDoctrine()
21                      ->getRepository(Category::class)
22                      ->findAll();
23
24          $articles = $this->getDoctrine()
25                      ->getRepository(Article::class)
26                      ->findAllPublished();
27
28          return $this->render('home/index.html.twig', [
29              'categories' => $categories,
30              'articles' => $articles,
31          ]);
32      }
```

Nous modifions enfin le template :

```
templates > home > 📄 index.html.twig

42      <div class="col-12 col-lg-4 order-first order-lg-last">
43          <h4>Derniers Articles</h4>
44          {% for article in articles %}
45              <div class="card article pb-2 mb-2" onclick="window.location='/article/{{ article.id }}/{{ article.title }}'">
46                  <div class="card-body overflow-hidden">
47                      <div class="d-flex justify-content-between">
48                          <h5 class="card-title">{{ article.title }}</h5>
49                          <span>Le {{ article.getPublishedDateString() }} par <a href="#">{{ article.userProfil.pseudo }}</a></span>
50                      <span>
51                          
52                          {{ article.getCategory().title }}
53                      </span>
54                  </div>
55                  <p class="card-text">
56                      {% if article.textContent|length > 200 %}
57                          {{ article.textContent|slice(0, 200) ~ '...' }}
58                      {% else %}
59                          {{ article.textContent }}
60                      {% endif %}
61                  </p>
62              </div>
63          </div>
64          {% endfor %}
65      </div>
66  </div>
67
```

Résultat :

Derniers Articles

Nullam dictum felis

Le 20 Sep. 2021 par [LarryDebug](#)

Asie

Curabitur commodo dolor eu odio finibus, id elementum mauris blandit. Nam at libero augue. Curabitur vel orci vel ipsum elementum vulputate sit amet sed massa. Fusce in mollis sapien. Suspendisse sed ...

Aenean faucibus urna massa

Le 19 Sep. 2021 par [LarryDebug](#)

Europe

Vivamus dictum libero quam, vitae convallis lorem pretium ullamcorper. Fusce ullamcorper gravida ex quis euismod. Vestibulum tempor viverra neque id bibendum. Aliquam at suscipit nulla, eget auctor li...

Sed varius tortor a diam.

Le 18 Sep. 2021 par [JDoe](#)

En France

In imperdiet, lacus in venenatis vestibulum, mauris nulla dapibus risus, quis sagittis sapien massa eu ante. Fusce quis hendrerit nisl. Quisque sed metus massa. Morbi non feugiat dolor, at dignissim o...

Code complet sur git : https://github.com/laurentbedu/td_symfony_5/tree/TD-7