

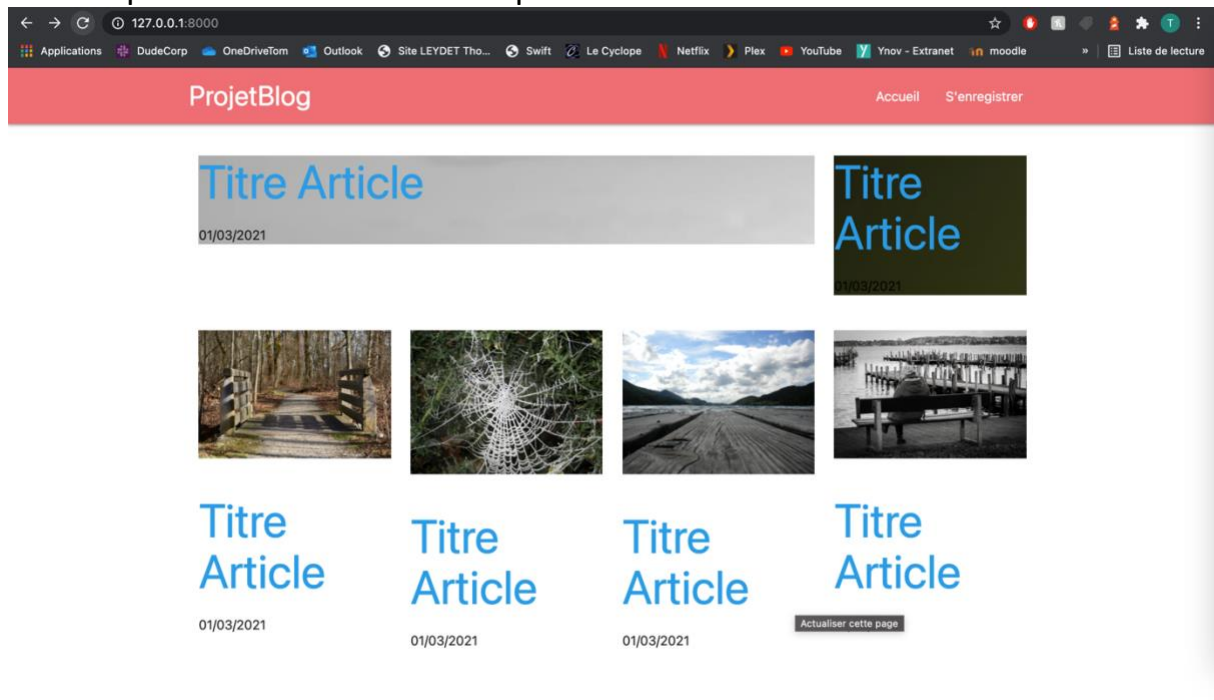
Compte Rendu ProjetBlog :

La première étape de ce projet va être la création des contrôleurs. Une fois les contrôleurs créés, grâce au moteur de template twig on aura un dossier « blog » dans lequel on va trouver les fichiers suivants :



Ces fichiers ont été générés lorsque nous avons créé le contrôleur « BlogController ».

Voici la première version du site qui a été créé :



Ainsi qu'un exemple de vue d'un article.

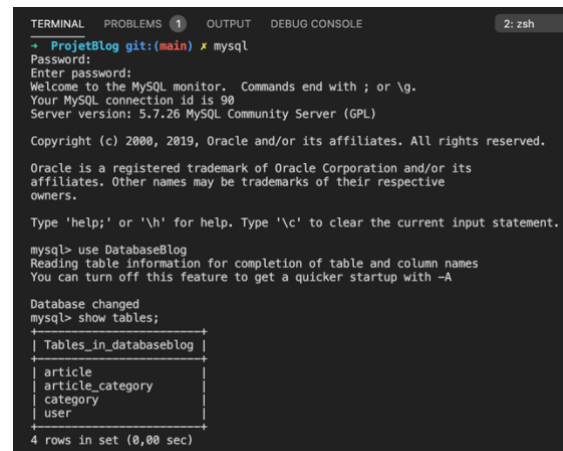
- Une ou plusieurs catégories (categories: Category[])
- Une date de publication et de dernière modification (publicationDate: datetime, lastUpdateDate: datetime)
- Et un booléen pour savoir si l'article est publié ou pas (isPublished: boolean)

Je crée mes entités avec la commande « php bin/console make:entity »

J'arrive à la partie qui m'a pris le plus de temps (juste pour un port pas bon). La base de données dans le .env

DATABASE_URL="mysql://root:root@127.0.0.1:8889/DatabaseBlog"

Une fois le problème réglé, j'ai pu créer mes tables : article, category, article_category.



```

TERMINAL  PROBLEMS  1  OUTPUT  DEBUG CONSOLE  2: zsh
+ ProjetBlog git:(main) x mysql
Password:
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 90
Server version: 5.7.26 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use DatabaseBlog
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_databaseblog |
+-----+
| article                 |
| article_category        |
| category                |
| user                    |
+-----+
4 rows in set (0,00 sec)

```

Il faut maintenant générer les Getter et Setter pour Category
php bin/console make:entity --regenerate

J'ai maintenant mes entités.

Les Formulaires : des classes

J'ai créé un premier formulaire à savoir le formulaire pour l'entité Article. La première chose à retenir c'est que chaque formulaire que j'ai créé est rattaché à une entité.

Ensuite, "php bin/console make:form" pour créer le formulaire.
Une fois créé, il faut l'afficher dans la vue

Ici, je crée un objet de type Article. Ensuite j'utilise la méthode `createForm()` de la classe `AbstractController` dont le contrôleur hérite pour créer le formulaire. En premier paramètre : l'EntityType sur lequel le formulaire est basé, puis j'envoie l'objet qui va contenir les données. Enfin j'envoie le formulaire à la vue.

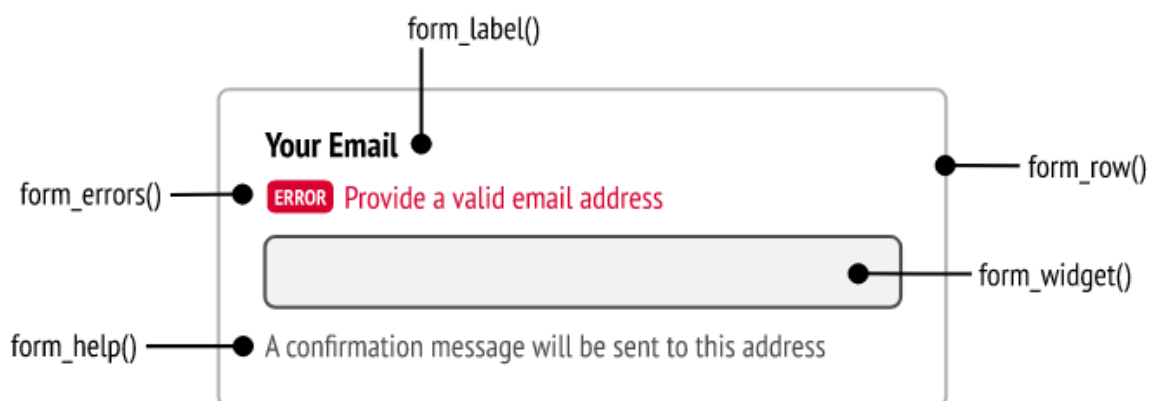
```
class BlogController extends AbstractController
{
    public function index()
    {
        return $this->render('blog/index.html.twig');
    }

    public function add()
    {
        $article = new Article();
        $form = $this->createForm(ArticleType::class, $article);

        return $this->render('blog/add.html.twig', [
            'form' => $form->createView()
        ]);
    }
}
```

Maintenant je dois donc afficher le formulaire :

Je me suis aidé de cet exemple :



Pour pouvoir personnaliser mon formulaire comme je le souhaite.

```
ArticleType.php | article_form.html.twig x | .env | User.php | edit.html.twig | index
templates > includes > ./ article_form.html.twig
1  {{ form_start(form) }}
2      <div class="form-group">
3          {{ form_label(form.picture, 'Image à la une') }}
4          {{ form_widget(form.picture, {'attr': {'class': 'form-control-file'}}) }}
5      </div>
6      <div class="form-row">
7          <div class="form-group col-md-6">
8              {{ form_label(form.title, 'Titre') }}
9              {{ form_widget(form.title, {'attr': {'class': 'form-control', 'placeholder': 'Titre'}}) }}
10             <small class="text-danger">{{ form_errors(form.title) }}</small>
11          </div>
12          <div class="form-group col-md-6">
13              {{ form_label(form.categories, 'Catégories') }}
14              {{ form_widget(form.categories, {'attr': {'class': 'selectpicker form-control'}}) }}
15          </div>
16      </div>
17      <div class="form-group">
18          {{ form_label(form.content, 'Contenu') }}
19          {{ form_widget(form.content, {'attr': {'class': 'form-control', 'rows': 12}}) }}
20      </div>
21      <button type="submit" class="btn btn-primary">Enregistrer</button>
22  {{ form_end(form) }}
```

J'ai rajouté cette commande qui me permet d'afficher un message d'erreur si le champ « title » est vide.

```
10      <small class="text-danger">{{ form_errors(form.title) }}</small>
```

127.0.0.1:8000/add

Symfony Blog Home Login

Ajout d'un article

Image à la une Aucun fichier choisi

Titre

Ce champ ne peut pas être vide.

Catégories

Contenu

ENREGISTRER

Is published

Une fois que j'ai mon formulaire, j'aimerais enregistrer un article. Car pour l'instant j'envoie qu'un texte.

127.0.0.1:8000/add

Applications DudeCorp OneDriveT

Le formulaire a été soumis...

Ici, j'ai eu un problème avec l'entity manager car je n'arrivais pas à enregistrer, modifier, supprimer l'entité en base de données.

L'authentification :

Je créer un user :

```
→ ProjetBlog git:(main) php bin/console make:user

The name of the security user class (e.g. User) [User]:
> User

Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:
> yes

Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:
> email

Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed by
rver).

Does this app need to hash/check user passwords? (yes/no) [yes]:
> yes

created: src/Entity/User.php
created: src/Repository/UserRepository.php
updated: src/Entity/User.php
updated: config/packages/security.yaml

Success!

Next Steps:
- Review your new App\Entity\User class.
- Use make:entity to add more fields to your User entity and then run make:migration.
- Create a way to authenticate! See https://symfony.com/doc/current/security.html
→ ProjetBlog git:(main) ✖ symfony server:start
```

La classe User hérite de UserInterface, une interface créée par Symfony.

Je génère la classe User :

```
→ ProjetBlog git:(main) ✖ php bin/console make:entity

Class name of the entity to create or update (e.g. DeliciousElephant):
> User

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

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

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

Field length [255]:
>

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

Not generating User::getUsername(): method already exists
updated: src/Entity/User.php

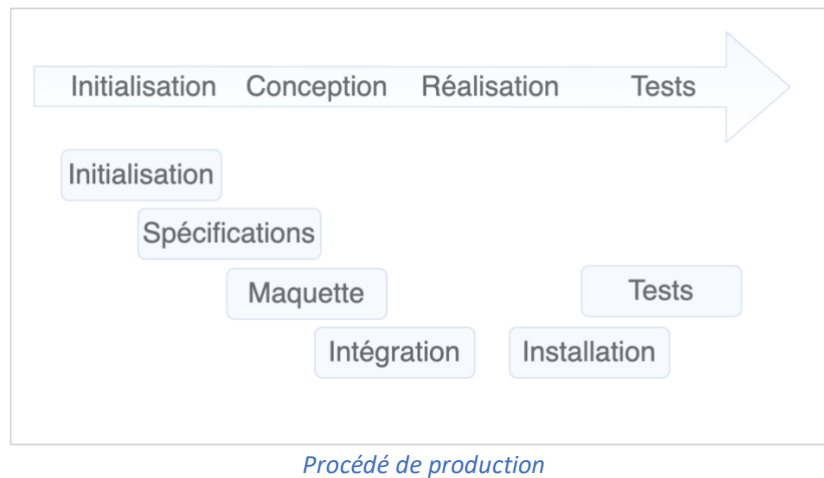
Add another property? Enter the property name (or press <return> to stop adding fields):
>

Success!
```

J'inscris un utilisateur :

Étant donné que j'ai déjà mon entité User, j'ai d'abord créé un formulaire d'inscription où l'utilisateur va devoir renseigner ces informations en fonction de mon entité User, puis j'ai créé un contrôleur pour traiter le formulaire et enregistrer l'utilisateur si tous les champs sont bons.

Cependant dans ma page register je n'ai pas le résultat attendu.



Pour conclure, je n'ai pas terminé ce projet dans les temps mais je compte bien vous envoyer la version terminée d'ici la fin du module.

Ce projet m'a beaucoup servi dans mon travail personnel, j'ai été sur toute la longueur du projet en autonomie étant donné les circonstances, ce qui m'a permis de progresser sur différents points.