# Symfony

## Liens utiles

Conférence : https://bbb.dawan.fr/b/cle-hci-tyj-dqt (https://bbb.dawan.fr/b/cle-hci-tyj-dqt)

Doc Symfony : https://symfony.com/doc (https://symfony.com/doc)

GitHub : https://github.com/DELORD-C/Formation-Symfony/tree/04-12-2023 (https://github.com/DELORD-C/Formation-Symfony/tree/04-12-2023)

LiveShare : https://prod.liveshare.vsengsaas.visualstudio.com/join?96DC5C76E7464EC6CA1B9E64B460FA3BF16E (https://prod.liveshare.vsengsaas.visualstudio.com/join?96DC5C76E7464EC6CA1B9E64B460FA3BF16E)

Principe Solid : https://fr.wikipedia.org/wiki/SOLID_(informatique) (https://fr.wikipedia.org/wiki/SOLID_(informatique))

Doc Doctrine : https://symfony.com/doc/current/doctrine.html (https://symfony.com/doc/current/doctrine.html)

## Prérequis

- PHP : https://php.net (https://php.net)
- Composer : https://getcomposer.org (https://getcomposer.org)
- Npm : https://nodejs.org/en/download/current (https://nodejs.org/en/download/current)
- Git : https://git-scm.com/download/win (https://git-scm.com/download/win)
- IDE
- Client Symfony : https://symfony.com/download (https://symfony.com/download)

> Pour installer le client symfony sur windows, le télécharger, décompresser l'archive, puis placer symfony.exe dans C:/symfony (créer un dossier)

## Extensions VSCode

- Live Share
- Twig
- Symfony for VSCode

# Commandes utiles

Vérifier les prérequis

```
symfony check:requirements
```

Installer un projet symfony

```
symfony new my_project_directory --version="7.0.*@dev" --webapp
```

Installer le fcertificat local https

```
symfony server:ca:install
```

Lancer le serveur

```
symfony serve
```

Forcer l’arrêt du serveur

```
symfony server:stop
```

Lister les routes

```
php bin/console debug:router
```

Créer la database

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

Créer une entité

```
php bin/console make:entity
```

Créer une migration

```
php bin/console make:migration
```

Appliquer les migrations

```
php bin/console doctrine:migrations:migrate
```

Installer File Loader

```
npm i file-loader
```

# Webpack

1. Installer Webpack

```
composer require symfony/webpack-encore-bundle
```

2. Installer les dépendances npm

```
npm install
```

3. Compiler les assets

```
npm run dev
npm run watch // Compiler en continue
```

# Bootstrap

1. Installer Bootstrap

```
npm install bootstrap
```

2. Ajouter bootstrap au point d'entrée

```
// assets/app.js
import 'bootstrap/dist/css/bootstrap.css';
require('bootstrap');
```

3. Compiler avec npm

```
npm run dev
npm run watch // Compiler en continue
```

# Mettre en route un projet

1. Cloner le repository
2. Mettre à jour le fichier .env (idéalement créer un .env.local)
3. Installer les dépendances composer

```
composer install
```

4. Si le projet utilise webpack (dossier assets / webpack.config.js) OU si il y a un fichier package.json
   - il faut installer les dépendances npm

     ```
     npm install
     ```

   - il faut compiler les assets

     ```
     npm run dev
     ```

5. Démarrer le serveur

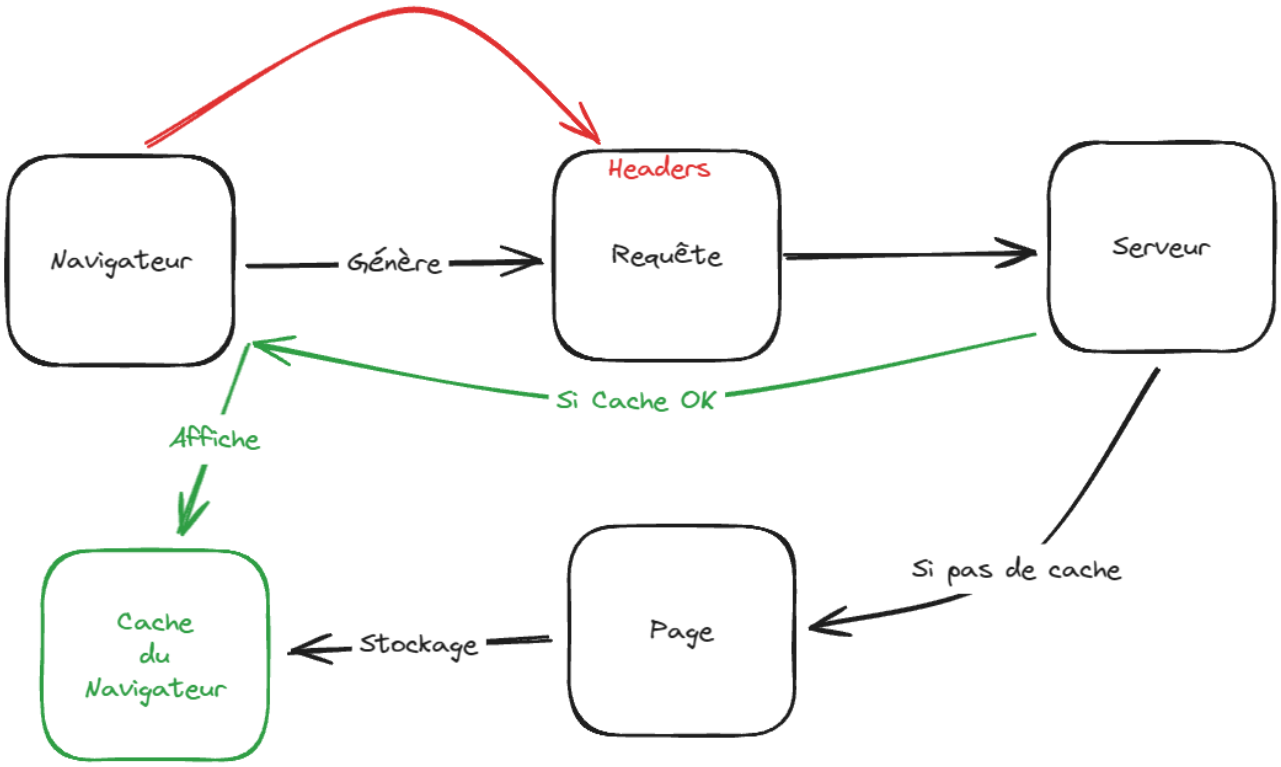
```
symfony serve
```

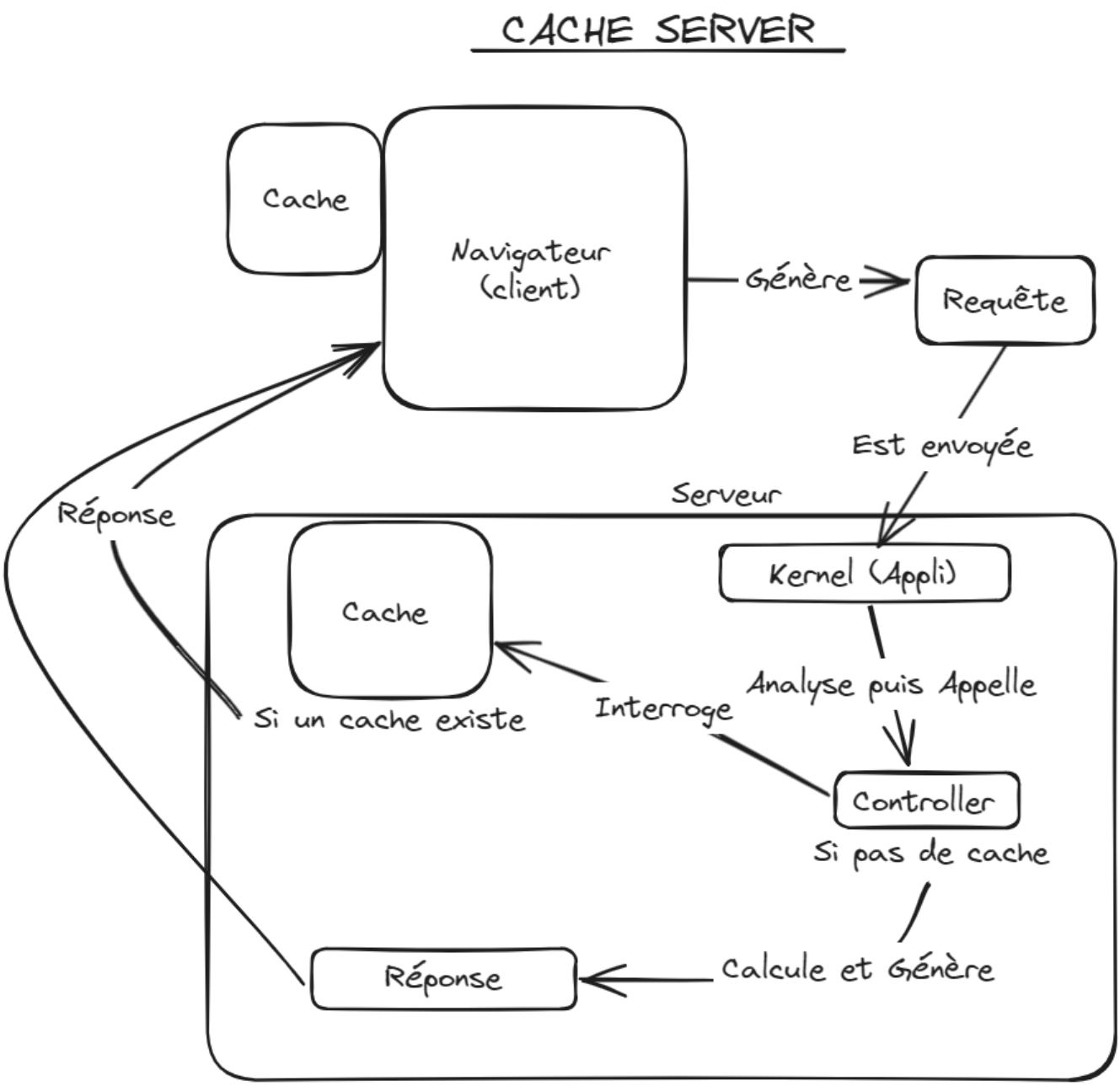
## Cache

https://symfony.com/doc/current/http_cache.html (https://symfony.com/doc/current/http_cache.html)

# CACHE DU NAVIGATEUR

Informations stockées localement sur la machine du client






# Exercices

## Exercice 1

Créer une page https://localhost:8000/random (https://localhost:8000/random) qui affiche un nombre aléatoire en 0 et 1000

Créer une page https://localhost:8000/random/10 (https://localhost:8000/random/10) qui affiche un nombre aléatoire en 0 et 10 (10 est un paramètre et peut changer)

Créer une page https://localhost:8000/random/10/1000 (https://localhost:8000/random/10/1000) qui affiche un nombre aléatoire en 10 et 1000 (10 et 1000 sont des paramètres et peuvent changer)

## Exercice 2

Utiliser le template base.html.twig sur toutes les pages en utilisant l'héritage.

Ajouter sur chaque page une navbar avec des liens vers chaque page

TOUTES VOS PAGES DOIVENT UTILISER RENDER

## Exercice 3

Pistes :

- https://symfony.com/doc/current/doctrine.html#persisting-objects-to-the-database (https://symfony.com/doc/current/doctrine.html#persisting-objects-to-the-database)
- https://symfony.com/bundles/SensioFrameworkExtraBundle/current/annotations/converters.html (https://symfony.com/bundles/SensioFrameworkExtraBundle/current/annotations/converters.html)

Ajouter les pages suivantes

/post/list => Liste de tous les posts sous forme de tableau
/post/delete/{post} => Supprime un post
/post/edit/{post} => Edition d'un post

Sur la page list, des boutons doivent permettre d'accéder aux autres pages.

## Exercice 4

Créer une entité Review qui sera un avis rédigé concernant un film.
L'entité doit-avoir les champs suivants :

- movie (le film critiqué) : string
- body : text
- createdAt (attention à modifier l'entité pour affecter une valeur par défaut)
- rating (note entre 1 et 10)

Créer une migration et l'appliquer.

Créer ensuite un Controller et un CRUD pour cette entité (penser à créer le type de formulaire)

## Exercice 5

Installer bootstrap et de l'appliquer à toutes les pages
*CF Doc

# Exercice 6

Ajouter des titres H1 sur chaque page

Appliquer du style sur toutes les pages (bootstrap / tailwind ...)

Appliquer du style sur les formulaires

# Exercice 7

Créer un système de commentaires pour les Review.

Créer l'entité ReviewComment

- body
- createdAt
- review (la relation ManyToOne avec les Review)

Migrer

Créer / Modifier le(s) controlleurs
Créer / Modifier le(s) templates

## Bonus

Changer le nom des entités Comment et ReviewComment pour
Post\Comment et Review\Comment

# Exercice 8

Créer une entité Like

- comment (relation avec Commentaire)

Migrer

Faire en sorte que l'on puisse liker un commentaire (Post), à chaque fois cela doit créer une entité Like.

Sur la page de lecture d'un post, le nombre de like associé à chaque commentaire doit apparaitre.

# Exercices 9

Relier toutes les entités à l'entité User :

- Un post doit avoir un auteur
- Un commentaire doit avoir un auteur
- Un review doit avoir un auteur
- Un like doit avoir un auteur

Un utilisateur ne peut liker qu'une seule fois un commentaire

Ajouter la possibilte d'annuler son like "déliker"

> ⚠️ Si on ajoute un champ obligatoire à une entité, et si on avait des entités de ce type présentes dans notre base de donnée, les migrations vont planter.

> Supprimer tous vos post/review/commentaires avant de faire votre/vos migrations

## Exercice 10

Doc Voter : https://symfony.com/doc/current/security/voters.html (https://symfony.com/doc/current/security/voters.html)

Doc Securité : https://symfony.com/doc/current/security.html (https://symfony.com/doc/current/security.html)

Créer un Voter pour les Post, afin d'appliquer la même stratégie que sur les Review, appliquer ce voter dans le controller à l'aide des annotations.

Idem pour les deux types de commentaire

Bloquer l'accès à la méthode LikeToggle aux utilisateurs connectés

Affecter le Front pour suivre cette politique (ne pas afficher les boutons lorsque l'utilisateur n'a pas accès)

## Exercice 11

Créer un CRUD pour les User sur les routes

/admin/user/create
----------/list
----------/update
----------/delete

Dans le update, il doit être possible de modifier les rôles (avec une selection, un champ libre, des cases à cocher au choix)

On ne doit pas pouvoir modifier un mot de passe utilisateur

Bloquer l'accès à toute cette partie aux admin uniquement

## Exercice 12

Donner la possibilité aux propriétaires d'un Post de supprimer n'importe quel commentaire sur celui-ci

## Exercice 13

https://symfony.com/doc/current/translation.html (https://symfony.com/doc/current/translation.html)

Si on change la valeur de default_locale en `fr` dans config/packages/translation.yaml, toute la page /post/list doit changer de langue (Menu, Titre, Titres de colonnes, Boutons)

## Exercice 14

https://symfony.com/doc/current/session.html#making-the-locale-sticky-during-a-user-s-session (https://symfony.com/doc/current/session.html#making-the-locale-sticky-during-a-user-s-session)

Mettre en place un système qui permet à l'utilisateur de changer la langue du site avec par exemple un bouton dans le menu

Bonus: Par défaut, utiliser la langue du navigateur

## Exercice Bonus

Grâce aux api https://open-meteo.com/ (https://open-meteo.com/) et/ou https://geocode.maps.co/ (https://geocode.maps.co/), Créer dans la navbar un affichage de la météo pour un lieux donné.

Utiliser JavaScript et la méthode fetch pour faire des calls API.

https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch
(https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch)

Bonus :

Utiliser l'api geolocation de js pour récupérer la position de l'utilisateur
https://developer.mozilla.org/fr/docs/Web/API/Geolocation_API
(https://developer.mozilla.org/fr/docs/Web/API/Geolocation_API)