Compte-rendu de projet

Framework Symfony et React

Calculateur de temps de plongée

Projet réalisé par

Julien Marquet

Titouan Allain

Supervisé par

Christophe Vignaud

Fabien Canu

Semestre 1

Année 2020 - 2021

# Index

1. Symfony
   1. Entités
   2. API
      1. URL des API
      2. Description des API
         1. api\_profondeur
         2. api\_profondeur\_by\_depth
         3. api\_temps
         4. api\_temps\_by\_depth\_and\_time
         5. api\_defaultParam
   3. Méthodes CRUD
      1. Liste des entités CRUD-able
      2. Routes génériques des méthodes CRUD
         1. Créer
         2. Lire
         3. Éditer
         4. Supprimer
      3. Exemples génériques des méthodes CRUD
         1. Créer
            1. readAll()
            2. read($id)
            3. readSelector()
         2. Lire
            1. New(Request $request)
         3. Éditer
            1. checkEditAvailable($id)
            2. edit(Request $request, Entite $entite)
            3. editSelector()
         4. Supprimer
            1. delete($id)
            2. deleteSelector()
2. React
3. Ressources
   1. PhpDocumentor
   2. Github
   3. ISEN

# Symfony

## Entités

Le cahier de charge initiale stipulait la nécessité de 3 entités pour réaliser la fonction première de l’application du projet, le calcul du temps de plongée en fonction d’une table de plongée (TablePlongee), d’une profondeur (Profondeur) et d’un temps donné (Temps).

Pour faciliter l’utilisateur dans l’interface de calcul de temps de plongée en front-end, nous avons rajouter une entité de paramètre par défaut (DefaultParam) pour permettre au service front-end de donner à l’utilisateur une configuration par défaut avec des paramètres tel que la respiration moyenne, la vitesse de descente et les vitesses de remonter.

Diagram

Description automatically generated

## API

La mise en place de ces API permettent un partage de ressources de base de données en back-end pour notamment le serveur front-end. Les données seront renvoyées par format JSON dès lors que l’API est sollicitée par une route de Symfony.

### URL des API

*Attention : les liens cliquables ne fonctionneront uniquement si vous avez le serveur Symfony en cours d’exécution sur votre environnement de développement en local.*

<http://localhost:8000/api/profondeur>

[http://localhost:8000/api/profondeur/depth/{depth}](http://localhost:8000/api/profondeur/depth/%7bdepth%7d)

[http://localhost:8000/api/temps](http://localhost:8000/api/tempsr)

<http://localhost:8000/api/defaultParam> [http://localhost:8000/api/temps/depth/{depth}/time/{time}](http://localhost:8000/api/temps/depth/%7bdepth%7d/time/%7btime%7d)

### Description des API

* @Route("/profondeur", name="api\_profondeur")

API pour avoir toutes les entrées de l’entité Profondeur.

* @Route("/profondeur/depth/{depth}", name="api\_profondeur\_by\_depth")

API pour avoir la première entrée de l’entité Profondeur qui à la bonne profondeur choisie.

* @Route("/temps", name="api\_temps")

API pour avoir toutes les entrées de l’entité Temps.

* @Route("/temps/depth/{depth}/time/{time}", name="api\_Temps\_by\_Depth\_and\_Time")

API pour avoir la première entrée de Temps en fonction d’une valeur de temps et d’une valeur de profondeur.

* @Route("/defaultParam", name="api\_defaultParam")

API pour avoir toutes les entrées de l’entité DefaultParam.

## Méthodes CRUD

### Liste des entités CRUD-able

Les entités ayant les fonctionnalités CRUD :

* DefaultParam
* TablePlongee
* Profondeur
* Temps

### Routes génériques des méthodes CRUD

Les routes, ci-dessous, devront être précédées par le nom de l’entité choisie. Les entités sont à écrire comme suit : temps, profondeur, table\_plongee, defautlParam.

#### Créer

* /new

#### Lire

* /read/all
* /read/{id}
* /read

#### Éditer

* /edit/check/{id}
* /edit/{id}
* /edit

#### Supprimer

* /delete/{id}
* /delete

### Exemples génériques des méthodes CRUD

#### Créer

##### new(Request $request)

Le paramètre $request, provenant du composant HttpFoundation, permet de faire une requête sur l’entité afin de récupérer son formulaire.

Fonction créant un objet de la classe de l’entité choisie pour l’ajouter en base de données, avec pour valeur de champs ceux provenant du formulaire de cette même entité.

Si le formulaire est bien valide, le manageur d’entité de Doctrine nous insère cet objet selon le formulaire en base de données.

#### Lire

##### readAll()

Fonction demandant la récupération de toutes le entrée de l’entité choisie, grâce à Doctrine et la fonction de findAll() du Repository de l’entité. Les données récupérées sont ensuite affichées comme paramètre du twig readAll.html.twig.

##### read($id)

Le paramètre $id est un entier qui permet de définir l’ID de l’entrée à lire.

Fonction demandant la récupération de l’entré d’ID défini par $id de l’entité choisie. La donnée récupérée est ensuite afficher comme paramètre du twig read.html.twig. Dans le cas où l’entrée demandée n’existe pas dans la base, un message d’erreur apparait.

##### readSelector()

Fonction renvoyant sur le twig selector.html.twig avec comme paramètre de twig crud\_method égale à « read » et entity égale au nom de l’entité choisie.

#### Editer

##### checkEditAvailable($id)

Le paramètre $id est un entier qui permet de définir l’ID de l’entrée à éditer.

Fonction faisant une vérification si l’entrée d’ID défini de l’entité choisi existe pour renvoyer à la route d’édition. Dans le cas échéant, la fonction nous renvoie sur le twig d’erreur avec un message d’erreur.

NB : Cette fonction est présente suite à la sécurité de la fonction edit(Request $request, Entity $entity), *voir fonction suivante*, qui fait, par ses paramètres une vérification de la présence de l’entrée choisie. Dans le cas où il n’existe pas, l’utilisateur reçoit un message d’erreur de Symfony. Cette fonction contourne donc cette vérification.

##### edit(Request $request, Entity $entity)

Les paramètre Request $request et Entity $entity servent respectivement à désigner une requête au niveau des formulaires et à pointer sur l’entrée de l’entité choisie d’ID mit en paramètre de la route.

Fonction faisant appel au formulaire de l’entrée sélectionnée permettant ainsi à l’utilisateur de renseigner/modifier des valeurs. Dans le cas d’un formulaire valide, la fonction exécute les modifications en base de donnée et renvoie sur le twig d’affichage de toutes les entrées de l’entité readAll.html.twig.

##### editSelector()

Fonction renvoyant sur le twig selector.html.twig avec comme paramètre de twig crud\_method égale à « edit » et entity égale au nom de l’entité choisie.

#### Supprimer

##### delete($id)

Le paramètre $id est un entier qui permet de définir l’ID de l’entrée à supprimer.

Fonction permettant de supprimer de la base de données l’entrée d’entité et d’ID choisis.

##### deleteSelector()

Fonction renvoyant sur le twig selector.html.twig avec comme paramètre de twig crud\_method égale à « delete » et entity égale au nom de l’entité choisie.

# React

# Ressources

## PhpDocumentor

Une documentation, générée par PhpDocumentor, des fichiers PHP du dossier /src, de la partie Symfony, est disponible dans le dossier resources de ce projet.

Vous pouvez ouvrir la page d’index de cette documentation par le fichier `resources/.phpdoc/build/index.html` depuis le répertoire courant du projet.

## Github

Ce projet a été réaliser à l’aide de github pour le partage des fichiers sources et des ressources. Il a été contribuer par Julien Marquet et Titouan Allain. Les liens du dépôt git du projet ainsi que celui des auteurs de ce dernier sont ci-dessous :

* Dépôt du projet : <https://github.com/dr-lux/project-cir3-framework-2020-2021>
* Julien Marquet : <https://github.com/psyKomicron>
* Titouan Allain : <https://github.com/dr-lux>

## ISEN

Ce projet est à but pédagogique dans le cadre d’un cours sur les utilisations des frameworks Symfony et React. Il nous est proposé par l’établissement de l’ISEN Yncrea Ouest Brest (<https://www.isen.fr>).