Paramétrer Symfony pour utiliser une bdd :

Fichiers de paramétrage

Paramètres de la database : `app/config/parameters.yml` Configuration de Doctrine : `app/config/config.yml`

Générer la database, les tables et les colonnes avec Doctrine :

Rafraîchir le cache :

```
php bin/console cache:clear
```

Créer la database :

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

Définir des tables puis leurs colonnes (interactif) :

```
php bin/console doctrine:generate:entity
```

- `AppBundle:table`
- `yml`
- `nomColonne`
- ...[ses paramètres]...

Générer les tables et colonnes (dans la base) :

Afficher les commandes SQL avant d'exécuter :

```
php bin/console doctrine:schema:update --dump-sql
```

Exécuter :

```
php bin/console doctrine:schema:update --force
```

Suite à la modification des fichiers de définition (.yml) :

```
php bin/console doctrine:generate:entities AppBundle
```

Si les tables sont déjà existantes, on génère les fichiers (.yml) :

```
php bin/console doctrine:mapping:import --force AppBundle yml
```

Insérer des données avec Doctrine :

Dans le controller :

Pour importer la table de la bdd :

```
use AppBundle\Entity\Table
```

Définir le Doctrine Manager :

```
$em = $this->getDoctrine()->getManager();
```

Définir une instance client et récupérer les données de la variable d'un formulaire:

```
$client = new Client();
$client->setName($tabDonneesForm['name']);
...
$client->setEmail($tabDonneesForm['email']);
```

Préparer les données à être insérées :

```
$em->persist($client);
```

Créer et exécuter la requête :

```
$em->flush();
```

Récupérer des données avec Doctrine :

Dans le controller :

Dans la fonction showIndex() :

```
$clients = $this->getDoctrine()->getRepository('AppBundle:Client')->findAll();
$data['clients'] = $clients;
```

On récupère tous les clients de la bdd que l'on met dans la variable clients (tableau) qui est transmise à la view (ajouter `$data` dans `render()`).

Pour récupérer une table uniquement (`id_client`) :

```
$client = $client_repo->find($id_client);
$client_data['id'] = $client->getID();
```

Mettre à jour des données avec Doctrine :

Dans le controller :

Dans la fonction Submit (ou analogue) :

```
$client = $client_repo->find($id_client)
$client->setName($form_data['name']);
...
$client->setEmail($form_data['email']);
$em = $this->getDoctrine()->getManager();
$em->flush();

return $this->redirectToRoute('index_clients');
```

On récupère les données entrées dans le formulaire qu'on insère avec `flush()` puis on redirige vers la page nommée (dans le controller) `index_clients`.

Commandes utiles

Pour encoder un password à insérer dans la base directement en SQL (créer l'utilisateur *admin* par exemple)

```
php bin/console security:encode-password
```