Mettre en place l'environnement pour développer l'appli

Outils requis

PHP

Depuis Windows

Télécharger la dernière version de PHP en .zip (x64 Thread Safe) sur le site de PHP For Windows. Puis extraire le contenu de l'archive vers C:\php\ qu'il faut désormais au PATH.

Pour cela on ouvre le terminale : Windows+R > cmd > OK. Et on entre la commande :

```
C:\Users\user> set PATH=%PATH%;C:\php
```

Enfin, on teste l'ajout de PHP dans le PATH en ouvrant Powershell (Windows+R > powershell) et en affichant le version de PHP installée :

```
PS C:\Users\user> php -v
PHP 7.2.7 (cli) (built: Jun 19 2018 23:14:45) ( ZTS MSVC15 (Visual C++ 2017) x64 )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
```

Si un message disant que php n'est pas une commande apparaît, se reconnecter à la session de travail peut résoudre ce problème.

Depuis Ubuntu

Afin d'être sûr d'avoir la dernière version de PHP installée, on ajoute le PPA d'ondrej en effectuant les commandes suivantes (on part du principe que PHP5 est installé sur la machine, si ce n'est pas le cas, exécuter tout de même ces commandes) puis on installe PHP :

```
sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get install software-properties-common python3-software-properties -y
sudo apt-get purge php5-common -y
sudo apt-get update
sudo apt-get autoremove -y
sudo LC_ALL=fr_FR.UTF-8 add-apt-repository ppa:ondrej/php
sudo apt-get update
sudo apt-get install php php-mysql php-mbstring php-zip php-xml php-curl -y
```

Comme pour Windows, on teste l'installation en tapant `php -v` dans le terminal.

Composer

Pour Windows

Se rendre sur le site de Composer et télécharger le Windows Installer en haut de la page. Puis l'exécuter et installer Composer en faisant attention d'indiquer le PHP installé précédemment. Un message proposant de modifier la configuration de PHP pourra apparaître, accepter alors la proposition de Composer.

A la fin de l'installation, aller dans C:\php\ et faire Clique-droit > Modifier sur le fichier `php.ini`. Depuis la fenêtre Bloc-notes, rechercher avec CTRL+F la ligne `extension=pdo_mysql` et la décommenter (retirer le ";") et faire de même pour `extension=curl`. Puis sauvegarder le fichier (avec CTRL+S) et fermer la fenêtre Bloc-notes.

On peut tester l'installation en tapant `composer -v` dans un Powershell.

Pour Ubuntu

Suivre la procédure indiquée sur le site de Composer.

Attention, cela installe Composer dans le dossier courant. Pour l'installer globalement dans le système, essayer `sudo apt install composer`

Sinon, depuis le dossier *~/* (le dossier *home* de l'utilisateur courant), créer un répertoire *bin/* : `mkdir bin`. Depuis ce dossier, effectuer la procédure d'installation de Composer ci-dessus. Ensuite, dans le fichier *~/.bashrc*, ajouter en bas la ligne suivante :

```
alias composer='php ~/bin/composer.phar'
```

Puis recharger le .bashrc avec `source ~/.bashrc` et tester l'installation en entrant `composer -v`.

MariaDB

Pour Windows

Se rendre sur le site de MariaDB et depuis la page de téléchargement de la dernière version, choisir la première se terminant par "winx64.msi". Puis exécuter l'installation en cochant "Use UTF8 as default server's character set" et en laissant le reste tel quel.

Pour tester l'installation, ouvrir un Powershell et entrer :

```
PS C:\Users\user> mysql -u root -p
```

Si la commande est introuvable, rajouter "C:\Program Files\MariaDB xy.z\bin" au PATH (comme indiqué pour l'installationd de PHP) où xy.z est la numéro de la version de MariaDB installée.

Un redémarrage de la session Windows peut être nécessaire.

Pour Ubuntu

Installer MariaDB depuis les dépôts apt :

```
sudo apt update
sudo apt install mariadb-server
```

Bien indiquer un mot de passe root lors de l'installation.

Puis on teste le bon déroulement de l'installation :

```
mysql -u root -p
```

Si cela ne fonctionne pas alors que tout s'est bien passé, exécuter ***partie à compléter***

git

Pour Windows

Télécharger et installer git depuis la page du projet Suivre la procédure d'installation par défaut.

Tester depuis un Powershell : `git --version`. En cas de commande non interprétée, vérifier que "C:\Program Files\Git\cmd" est bien inclus dans le PATH (depuis cmd, `echo %PATH%`) et relancer la session.

Pour Ubuntu

Installer git depuis les dépôts apt :

```
sudo apt update
sudo apt install git
```

Puis `git --version` pour vérifier.

Récupération du projet et préparation

Depuis le terminal (Powershell, bash, zsh, etc. Dépendemment de l'OS), cloner le dépôt :

```
git clone https://github.com/simtrami/clochette.git
```

Pour récupérer les sous-modules au passage (comme escpos-php) :

```
git clone --recurse-submodules https://github.com/simtrami/clochette.git
```

Un dossier *clochette* va être créé dans le dossier courant, s'y déplacer avec `cd clochette`. Il s'agit alors d'installer les dépendances et de configurer le serveur PHP de développement ainsi que le client SQL. Exécuter alors `composer install`. Les valeurs des champs à remplir sont indiqués sur le Slack du projet.

Plus tard, pour récupérer les submodules (comme escpos-php) si l'option `--recurse-submodules` n'a pas été passée :

```
git submodule update --init --recursive
```

Générer le secret

Pour le dernier champ `secret (ThisTokenIsNotSoSecretChangeIt):`, ouvrir un nouveau terminal (Ubuntu) ou Git Bash (Windows) et exécuter :

```
cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 40 | head -n 1
```

Copier et coller la chaîne de caractères retournée pour remplir le champ du token secret de l'application Symfony.

Tester l'installation

Si aucune erreur n'a été retournée, lancer le serveur PHP intégré au projet et tester l'application depuis le navigateur. Pour cela, exécuter `php bin/console server:run` depuis le répertoire racine du projet et ouvrir le lien affiché dans le terminal, le site doit s'afficher dans le navigateur.

L'environnement local de développement est prêt !

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
```

Pour mettre à jour les index Algolia (en supprimant les existants avant)

```
php bin/console search:clear
php bin/console search:import
```

Rajouter l'option `--indices=` suivie de la liste des indices auxquels affecter ces commandes (séparés par ",").