« Day 2 »

La semaine dernière, nous avons manipulé un contrôleur et une page twig (vue). Nous allons maintenant travailler sur le modèle avec doctrine http://www.doctrine-project.org/.

1. Dans votre terminal Cloud9 :
   a. `composer require symfony/orm-pack`
   b. `composer require symfony/maker-bundle –dev`

2. Créer le répertoire « Entity » dans **src.** Pour chaque objet nécessaire à votre CV : Formations, Expériences, Loisirs.
   a. Créer un fichier PHP et une classe dédiée dans « Entity »
   b. Créer des attributs qui caractérisent votre classe (protected)
   c. Créer des accesseurs de type « public »
   d. Avec les annotations ORM de doctrine vous allez indiquer comment synchroniser vos attributs avec la base de données, Prenez votre temps. Regarder le site Doctrine.

```php
use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="AppBundle\Repository\FormationRepository")
 * @ORM\Table(name="app_user")
 *
 * @UniqueEntity("name")
 */
class Formation
{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var string
     *
     * @Groups({"user"})
     *
     * @ORM\Column(type="string", name="name")
     */
    private $name;
```

3. Vérifier que votre configuration à la base de données (MySQL ou autre) est bonne dans le fichier **.env** (si le fichier n'est pas visible dans cloud9 => activer voir les fichiers cacher)

```
# .env (or override DATABASE_URL in .env.local to avoid committing your changes)

# customize this line!
DATABASE_URL="mysql://db_user:db_password@127.0.0.1:3306/db_name"

# to use sqlite:
# DATABASE_URL="sqlite:///%kernel.project_dir%/var/app.db"
```

On oublie pas d'installer MySQL et PhpMyAdmin sur cloud9 !!!!

4. A l'aide de la console Symfony dans votre shell : « php bin/console –help » :

a. Synchroniser le schéma avec votre modèle dans un sens ou l'autre avec les commandes doctrines : https://symfony.com/doc/3.4/doctrine.html
Quel est l'intérêt de doctrine ? Pourquoi utiliser le langage DBAL ?

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

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

Vous avez maintenant une association équivalente entre votre modèle et votre base de données.

Pour accélérer la création d'entity on peut utiliser

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

```
Restricting packages listed in "symfony/symfony" to "3.4.*"
Nothing to install or update
Generating autoload files
ocramius/package-versions:  Generating version class...
ocramius/package-versions: ...done generating version class
Executing script cache:clear [OK]
Executing script assets:install public [OK]


a454007@FR-PC0U8W5J MINGW64 ~/repos/uga/my_cv (master)
$ php bin/console make:entity

 Class name of the entity to create or update (e.g. GentlePizza):
 > Formation
F[Ko[Kr[Km[Ka[Kt[Ki[Ko[Kn[K

 created: src/Entity/Formation.php
 created: src/Repository/FormationRepository.php

 Entity generated! Now let's add some fields!
 You can always add more fields later manually or by re-running this command.

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

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


 Field length [255]:
 >

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

 updated: src/Entity/Formation.php

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

5. Créer une méthode dédiée dans un contrôleur qui sauvegarde un objet dans votre base :

```
$form = new Formation();
$form->setName('Ma formation');
$eManager = $this->getDoctrine()->getManager();
$eManager->persist($form);
$eManager->flush();
```

Vous pouvez aussi temporairement créer vos données directement avec Phpmyadmin.

6. Pour récupérer vos objets dans un contrôleur, il faut soit :

a. Utiliser le fichier « repository » défini dans la classe de votre modèle avec **@ORM\Entity** puis dans ce fichier créer une méthode dédiée pour récupérer les données en DBAL et enfin l'appeler de votre contrôleur (Méthode à préférer)

```php
use Doctrine\ORM\EntityRepository;

/**
 * Formation repository.
 */
class FormationRepository extends EntityRepository
{
    /**
     * Find ...
     */
    public function findAllFormations()
    {
        $qBuilder = $this
            ->getEntityManager()
            ->createQueryBuilder();

        $qBuilder->select('f');
        $qBuilder->from('AppBundle:Formation', 'f');

        $result = $qBuilder->getQuery()->getResult();

        return $result;
    }
```

b. Utiliser les « magic queries » directement dans votre contrôleur findAll et findOneBy

De la façon suivante : **$this->getDoctrine()->getRepository('ApiBundle:User')->findOneBy(['username' => $username]);**

7. Enfin, vous pouvez faire suivre le résultat à votre vue :

```php
 * @param Request $request the request
 *
 * @return array
 */
public function indexAction(Request $request)
{
    $forms = $this->getDoctrine()->getRepository("AppBundle:Formation")->findAll();

    return array(
        'formations' => $forms,
    );
}
```

8. Et l'afficher par exemple :

```twig
<div class="col-md-9 col-sm-9 col-xs-12">
    <ul>
    {% for formation in formations %}
        <li class="padding-top-10">
            {{ formation.getName() }}
        </li>
    {% endfor %}
    </ul>
</div>
```

BONUS

Mettre à jour composer avec « api-platform/core »

Utiliser l'annotation @ApiResource sur chaque entité de votre modèle pour générer une API de votre modèle. https://api-platform.com/