

« 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. Créer le répertoire « Entity » dans **src/AppBundle**
2. Pour chaque objet nécessaire à votre CV : Formations, Expériences, Loisirs.
 - a. Créer 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

```
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 **app/config/config.yml** et **app/config/parameters<_*>.yml**
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 ?

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

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

```
$form = new Formation();

$form->setName('Ma formation');

$entityManager = $this->getDoctrine()->getManager();

$entityManager->persist($form);

$entityManager->flush();
```

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)

```
use Doctrine\ORM\EntityRepository;

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

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

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

        return $result;
    }
}
```

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

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

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

```
* @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 :

```
<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/>