Doctrine : méthodes de base

Doctrine fournit un ensemble de méthodes prédéfinies permettant de récupérer aisément des informations provenant de la base de données.

Récupérer un objet à partir de son id : find et show

La méthode find prend en paramètre l'id et permet de créer et hydrater un objet. Par exemple si l'on souhaite récupérer un étudiant en fonction de son id :

```
$etudiant = $this->getDoctrine()
        ->getRepository(Etudiant::class)
        ->find($id);
```

La méthode show peut être définie dans le contrôleur et permet également de récupérer un objet à partir de son id.

Il faut d'abord installer le composant nécessaire à l'utilisation du composant :

```
>composer require sensio/framework-extra-bundle
```

```
/*
    * @Route("/etudiant/{id}", name="etudiant_show")
    */
    public function show(Etudiant $unEtudiant)
    {

        return $this->render('etudiant/consulter.html.twig', [
            'etudiant' => $unEtudiant,]);
    }
```

Vous voyez au passage dans les annotations, une autre façon de déclarer les routes.

Récupérer une liste d'objets : findAll

A partir de l'entité, il s'agit avec la méthode findAll de récupérer tous les objets correspondant à tous les enregistrements de la table. Par exemple la méthode ci-dessous permet de récupérer la liste des étudiants (d'où le s dans le nom de la variable $etudiants).

```
public function listerEtudiant(){
        $repository = $this->getDoctrine()->getRepository(Etudiant::class);
        $etudiants = $repository->findAll();
        return $this->render('etudiant/lister.html.twig', [
            'pEtudiants' => $etudiants,]);
```

```
    }
```

Récupérer des objets en fonction d'un paramètre ; findByXxx et findOneByXxx

Il s'agit de suffixer la méthode find par le nom d'une des propriétés de l'entité. Par exemple, si l'on souhaite récupérer la liste des étudiants en fonction de la ville:

```
/**
    * @Route("/etudiant/listerParVille/{ville}",
name="listerEtudiantsParVille")
    */
    public function listerParVille($ville){

        $etudiants = $this->getDoctrine()
        ->getRepository(Etudiant::class)
        ->findByVille($ville);

        return $this->render('etudiant/lister.html.twig', [
            'pEtudiants' => $etudiants,]);

    }
```

Cette méthode peut aussi être utilisée avec des propriétés complexes. Par exemple,si l'on souhaite récupérer la liste des étudiants par maison (Maison étant une autre entité contenant ses propres propriétés).

```
/**
    * @Route("/etudiant/listerParMaison/{idMaison}",
name="listerEtudiantsParMaison")
    */
    public function listerParMaison($idMaison){

        $maison = $this->getDoctrine()
        ->getRepository(Maison::class)
        ->findOneByCode($idMaison);

        $etudiants = $this->getDoctrine()
        ->getRepository(Etudiant::class)
        ->findByMaison($maison);

        return $this->render('etudiant/lister.html.twig', [
            'pEtudiants' => $etudiants,]);

    }
```

Vous pouvez voir aussi dans cette méthode l'utilisation de la méthode findOneByMaison. Cette méthode est utilisée lorsque la requête ne renvoie qu'un et un seul enregistrement.

Récupérer des objets selon plusieurs critères : findBy et findOneBy

Ces méthodes permettent de récupérer un seul(findOneBy) ou une liste d'objets (findBy) en fonctions de plusieurs critères. Par exemple si l'on souhaite récupérer un étudiant selon son nom et son prénom :

```
/**
    * @Route("/etudiant/consulterParNomPrenom/{nom}/{prenom}",
name="consulterEtudiantParNomPrenom")
    */
    public function consulterParNomPrenom($nom,$prenom){
        $repository = $this->getDoctrine()->getRepository(Etudiant::class);
        $etudiant = $repository->findOneBy(
            ['nom' => $nom,'prenom' => $prenom ]
        );

        return $this->render('etudiant/consulter.html.twig', [
            'etudiant' => $etudiant,]);

    }
```

Attention si plusieurs étudiants peuvent avoir le même nom et le même prénom, la requête peut donc potentiellement ramener plusieurs enregistrements. Il faudrait alors utiliser la méthode findBy et non findOneBy.

Méthodes spécifiques : EntitéRepository

Si les méthodes de base ne suffisent pas à récupérer les objets attendus, il est possible d'implémenter des méthodes spécifiques. Ces méthodes sont à positionner dans la classe repository de l'entité correspondante. Par exemple si l'on souhaite récupérer une liste d'étudiants dont la date de naissance est supérieure à une certaine date qui sera passée en paramètre, on ajoutera dans la classe EtudiantRepository la méthode ci-dessous :

```
  public function consulterEtudiantParDateNaissSup($dateNaiss): array
    {
         $dateNaiss = new \DateTime(date($dateNaiss));
         $qb = $this->createQueryBuilder('e')
            ->andWhere('e.dateNaiss > :pDateNaiss')
            ->setParameter('pDateNaiss', $dateNaiss)
            ->orderBy('e.nom', 'ASC')
            ->getQuery();

        return $qb->execute();

    }
```

La fonction est appelée alors dans le contrôleur de la même façon que les méthodes de base.

```
/**
    * @Route("/etudiant/consulterEtudiantsDateNaissSuperieur/{dateNaiss}",
name="consulterEtudiantsDateNaissSuperieur")
    */
    public function consulterEtudiantsDateNaissSuperieur($dateNaiss){
        $etudiants = $this->getDoctrine()
        ->getRepository(Etudiant::class)
        ->consulterEtudiantParDateNaissSup($dateNaiss);

        return $this->render('etudiant/lister.html.twig', [
            'pEtudiants' => $etudiants,]);

    }
```

Tests : http://localhost/webstudent/public/etudiant/consulterEtudiantsDateNaissSuperieur/1980-07-30