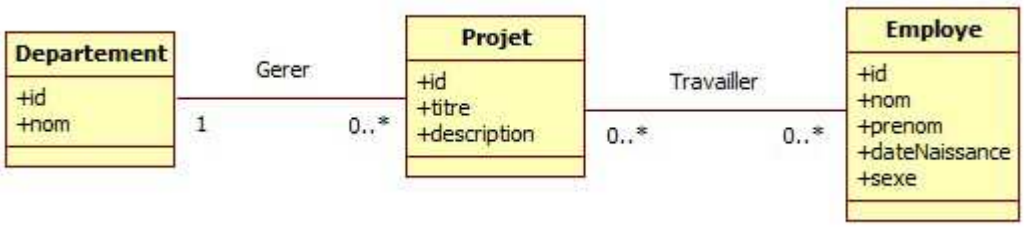
Projet **FW**

TP2 : Création de la base de données

1 – Création des entités

Notre application doit gérer des départements d'entreprise, des projets et des employés. Les employés travaillent sur des projets (« développement des compétences managériales », « norme Bâle III », « site E-commerce », « Intranet mobile » …) et ces projets sont gérés par les départements de l'entreprise (Ressources Humaines, Comptabilité, Informatique…). Ces entités peuvent être représentées par le schéma suivant (remarquez les **multiplicités**) :



Dans Symfony, une entité est représentée par une classe d'objets. Nous allons ainsi créer plusieurs fichiers dans le répertoire "Entity".

Fichier Departement.php

```
<?php
namespace ProjetBundle\Entity;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;
/**
* @ORM\Entity
*/
class Departement {
    /**
    * @ORM\GeneratedValue
    * @ORM\Id
    * @ORM\Column(type="integer")
    */
    private $id;
    /**
    * @ORM\Column(type="string",length=255)
    * @Assert\NotBlank()
    * @Assert\Length(
    *       min = 3,
    *       minMessage = "The nom must be at least {{ limit }} characters long",
    * )
    */
    private $nom;
}
```

Créer aussi les fichiers suivants en vous basant sur la description fournie (et les exemples du cours) :

Projet.php avec les colonnes :

- id, de type entier, identifiant, auto-généré
- titre, de type string (255), non nul, doit avoir au moins 3 caractères
- description, de type string (500)
- departement, avec les contraintes :
    - non nul
    - clé étrangère : @ORM\ManyToOne(targetEntity="Departement")
- employes, avec la contrainte :
    @ORM\ManyToMany(targetEntity="Employe")

Employe.php avec les colonnes :

- id, de type entier, identifiant, auto-généré
- nom, de type string (255), non nul, doit avoir au moins 3 caractères
- prenom, de type string (255), non nul, doit avoir au moins 3 caractères
- dateNaissance, de type date, non nul
- sexe, de type string (1), avec les contraintes :
    - non nulle
    - Valeurs limitées : @Assert\Choice(choices = {"M", "F"})

Créer maintenant un répertoire '**Entity'** dans votre répertoire /**VotreCompte**/public_html/symfony/src/**ProjetBundle**/ et placer à l'interieur les fichiers de vos 3 nouvelles entités.

Nous allons utiliser la console pour générer automatiquement les accesseurs (les fonctions **set()** et **get()).**

Ouvrez tout d'abord le fichier **symfony/app/config/config.yml** et vérifiez que le paramètre **auto_mapping** a bien pour valeur **true** :

```
    orm:
        auto_generate_proxy_classes: "%kernel.debug%"
        naming_strategy: doctrine.orm.naming_strategy.underscore
        auto_mapping: true
```

Il faut maintenant paramétrer l'accès à votre base de données. Aller dans le fichier **VotreCompte**/public_html/symfony/**app**/**config/parameters.yml** pour enregistrer vos identifiants de connexion à la base :

```
parameters:
    database_host: 127.0.0.1
    database_port: null
    database_name: base_XXXX
    database_user: XXXXX
    database_password: YYYYYY
    mailer_transport: smtp
    mailer_host: 127.0.0.1
    mailer_user: null
    mailer_password: null
    secret: d5a51d82c03da75c1088192905518b16b36f03e2
```

Puis ouvrez la console et tapez la commande suivante depuis le répertoire Symfony : **php app/console doctrine:generate:entities ProjetBundle**

De nouveaux fichiers d'entités sont alors générés avec l'ensemble des fonctions **set()** et **get().** Les anciens fichiers sont pour leur part renommés avec le suffixe "~" (Employe.php~, Departement.php~ et Projet.php~). Vous pouvez supprimer ces trois fichiers temporaires (ceux avec le suffixe "~"), ils ne serviront plus.

2 – Création de la base de données et des tables

A - Création des tables de la base

• Dans votre console, tapez la commande :

php app/console doctrine:schema:create.

• Le message "Database schema created successfully ! " devrait s'afficher.

Vérifier que les tables ont été créées correctement (sous https://webdev.iut-blagnac.fr/phpmyadmin/, avec vos identifiants usuels).

Pour ajouter une nouvelle entité ou effectuer des modifications sur une entité existante, alors il faudra mettre à jour les tables grâce à la commande :

php app/console doctrine:schema:update --force

B - Enregistrement d'une première donnée

Plutôt que d'effectuer une requête **INSERT INTO...**, nous allons utiliser la puissance de Symfony et de Doctrine pour ajouter des départements.

• Ouvrez le fichier **ProjetBundle/Controller/DefaultController.php**.
• Modifier le code comme suit :

```php
<?php

namespace ProjetBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use ProjetBundle\Entity\Departement;

class DefaultController extends Controller {

  public function indexAction($name) {
      // on recupere le gestionnaire de donnees
    $em = $this->getDoctrine()->getManager();

    $dept1 = new Departement();
    $dept1->setNom('Ressources Humaines');
    $em->persist($dept1);

    $dept2 = new Departement();
    $dept2 ->setNom('Comptabilité');
    $em->persist($dept2);

      // le flush fait l'equivalent d'un commit
    $em->flush();

    return $this->render('ProjetBundle:Default:index.html.twig',
                              array('name' => $name));
  }
}
```

Ouvrez votre navigateur et rendez-vous sur la page :
http://webdev.iut-blagnac.fr/~VotreCompte/symfony/web/projet/hello/martin

• Vérifiez dans votre base de données que les deux départements ont bien été créés.

• Pour éviter de créer ces départements chaque fois que vous accédez à cette URL, vous pouvez mettre tout le nouveau bloc en commentaires (toutes les instructions de la fonction indexAction(..) hormis la derniere ligne (return $this…).

Dossier à rendre sous Moodle (en **PDF UNIQUEMENT**) :

- Page de garde avec nom-prénom, groupe, date, matière, numéro et intitulé du TP
- Sommaire
- Introduction (objectif / thème du TP)
- Pour chaque nouvelle page :
  - Code « intéressant » commenté
  - Copie d'écran pour la trace d'exécution
- Conclusion (notions importantes abordées dans ce TP)