Programmation Web Avancé

Chapitre 3 : Object-Relational Mapping

Guillaume-Olivier LAFFITTE

guillaume.laffitte@stratow.com

ORM ?

Un mapping objet-relationnel est une interface entre une application et une base de données relationnelle pour simuler une base de données orientée objet.

Base de données relationnelle

- Dans une base de données relationnelle l'information est organisée dans des tableaux à deux dimensions
- Il peut y avoir plusieurs tables connectées implicitement par les valeurs qu'elles contiennent.
- Les liens existants entre les informations sont stockés dans les champs des enregistrements sous forme de clé étrangère.

Base de données relationnelle

personne			
ID	**NOM**	**PRENOM**	**ADRESSE**
1	Depuis	Pierre	1

bureau			
ID	**NOM**	**RESPONSABLE**	**ADRESSE**
1	EDF	1	2

adresse			
ID	**NUMERO**	**RUE**	**VILLE**
1	53	Rue de la paix	Paris
2	147	Place de la mairie	Paris

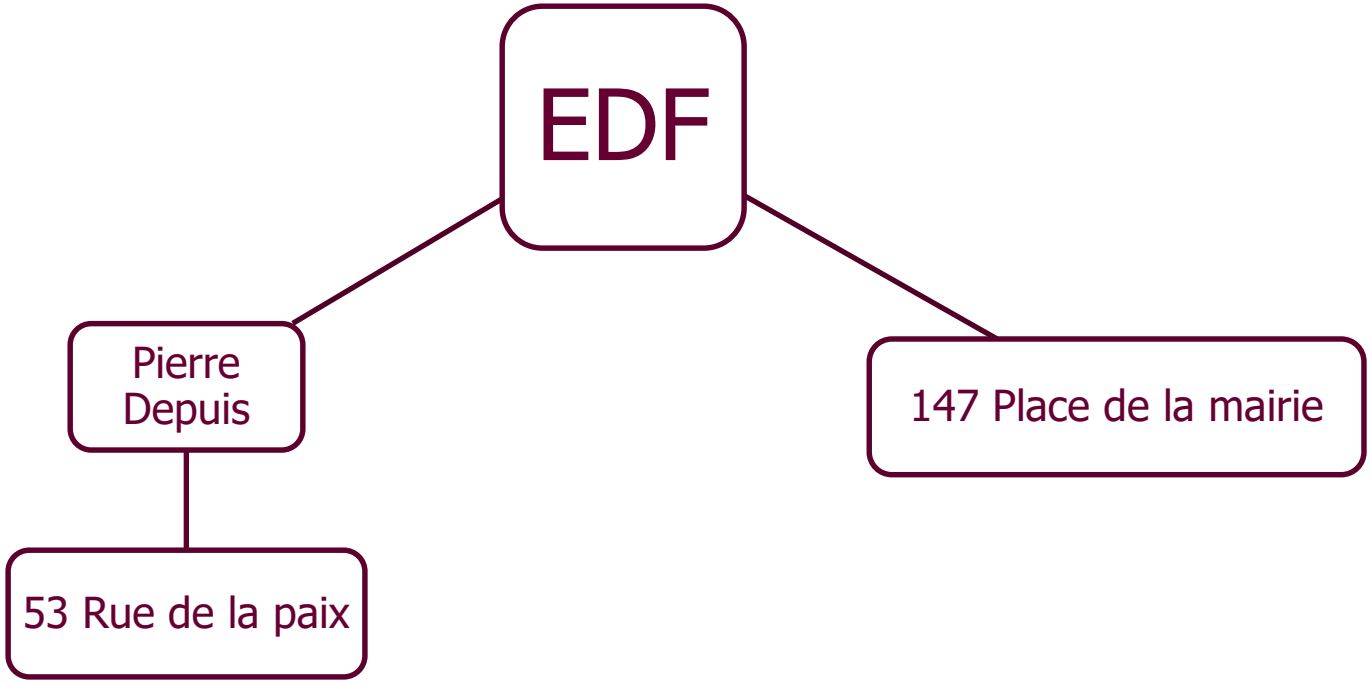
Base de données relationnelle

```
test=> SELECT * FROM personne;
 id |  nom   | prenom | adress
----+--------+--------+--------
  1 | Depuis | Pierre |      1
(1 row)
```

```
test=> SELECT * FROM personne INNER JOIN adresse ON personne.adress = adresse.id;
 id |  nom   | prenom | adress | id | numero |      rue       | ville
----+--------+--------+--------+----+--------+----------------+-------
  1 | Depuis | Pierre |      1 |  1 |     53 | Rue de la paix | Paris
(1 row)
```

```
test=>
test=> SELECT * FROM bureau
INNER JOIN personne ON bureau.responsable = personne.id
INNER JOIN adresse ON personne.adress = adresse.id;
 id | nom | responsable | adresse | id |  nom   | prenom | adress | id | numero |      rue       | ville
----+-----+-------------+---------+----+--------+--------+--------+----+--------+----------------+-------
  1 | EDF |           1 |       2 |  1 | Depuis | Pierre |      1 |  1 |     53 | Rue de la paix | Paris
(1 row)
```

Objets



Doctrine

Entité

```php
<?php

namespace Stratow\StratowBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Table(name="users")
 * @ORM\Entity
 */
class Users
{

    /**
     * @ORM\Column(name="id", type="string", length=128)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @ORM\Column(name="username", type="string", length=255, nullable=false, unique=true)
     */
    private $username;

    /**
     * @ORM\Column(name="password", type="string", length=255, nullable=true)
     */
    private $password;


    //setter et getter
}
```

personne		
ID	**USERNAME**	**PASSWORD**
1	root	123456

Entité

Type Doctrine	Type PHP	Type en base de données (MySQL)
string	string	VARCHAR
integer	integer	INT
boolean	boolean	BOOLEAN
date	\DateTime	DATETIME
datetime	\DateTime	TIMESTAMP
blob	stream resource	BLOB

Créer la BDD

- doctrine:database:create
  - Créé la BDD configurée
- Doctrine:schema:create
  - Crée le model de donnée de votre application
- doctrine:schema:update
  - Met a jour le model de donnée

- doctrine:generate:entities
  - Ajoute les Setter et Getter à vos entités
- Doctrine:generate:entity
  - Génération automatique d'une entité

EntityManager

- L'entity manager est un registre des entités chargées en mémoire
- Pour sauvegarder un objet en base il faut l'ajouter à l'entity manager puis synchroniser l'entity manager

EntityManager

- $entityManager->persist($entity)
  - Cette méthode signale à Doctrine que l'objet doit être enregistré. Elle ne doit être utilisée que pour un nouvel objet et non pas pour une mise à jour.

- $entityManager->flush()
  - Met à jour la base à partir des objets signalés à Doctrine. Tant qu'elle n'est pas appellée, rien n'est modifié en base.

- $entityManager->clear($nomEntity = null)
  - Annule tous les persist() en cours. Si le nom d'une entité est précisé (son namespace complet ou son raccourci), seuls les persist() sur les entités de ce type seront annulés.

EntityManager

- $entityManager->detach($entity)
  - Annule le persist() effectué sur l'entité en argument. Au prochain flush(), aucun changement ne sera donc appliqué à l'entité.

- $entityManager->refresh($entity)
  - Rafraîchit l'entité donnée en argument pour la mettre dans l'état où elle se trouve en base de données. Cela écrase et annule donc tous les changements qu'il a pu y avoir sur l'entité depuis le dernier flush().

- $entityManager->remove($entity)
  - Signale à Doctrine qu'on veut supprimer l'entité en argument de la base de données. Effectif au prochain flush().

Ajout et suppression

- Ajout d'un objet en base

```
$em = $this->container->get('doctrine')->getManager();

$user = new Users();

$user
    ->setUsername('root')
    ->setPassword('azertyuiop')
;

$em->persist($user);
$em->flush();
```

- Supression d'un objet

```
$this->em->remove($status);
$this->em->flush();
```

Recherche

- Trouver toutes les personnes

```
$personnes = $this->container->get('doctrine')
    ->getRepository(Personnes::class)
    ->findAll()
;
```

- Trouver les adresses avec un critère

```
$adresses = $this->container->get('doctrine')
    ->getRepository(Adresses::class)
    ->findByVille('Paris')
;
```

Recherche

- Trouver une personne avec un critère

```
$personne = $this->container->get('doctrine')
    ->getRepository(Personnes::class)
    ->findOneById(1)
;
```

- Tri et limite

```
$adresses = $this->container->get('doctrine')
    ->getRepository(Adresses::class)
    ->findBy(
        array('ville' => 'Paris'),
        array('id' => 'desc'),
        1000
    );
;
```

QueryBuilder

```php
$qb = $em->createQueryBuilder();

$qb
    ->select('p')
    ->from('Personnes', 'p')
    ->where('p.id = ?1')
    ->orderBy('p.name', 'ASC')
    ->setParameter(1, 100)
;
```

```php
// Execute Query
$result = $query->getResult();
$single = $query->getSingleResult();
$array = $query->getArrayResult();
$scalar = $query->getScalarResult();
$singleScalar = $query->getSingleScalarResult();
```

Repository

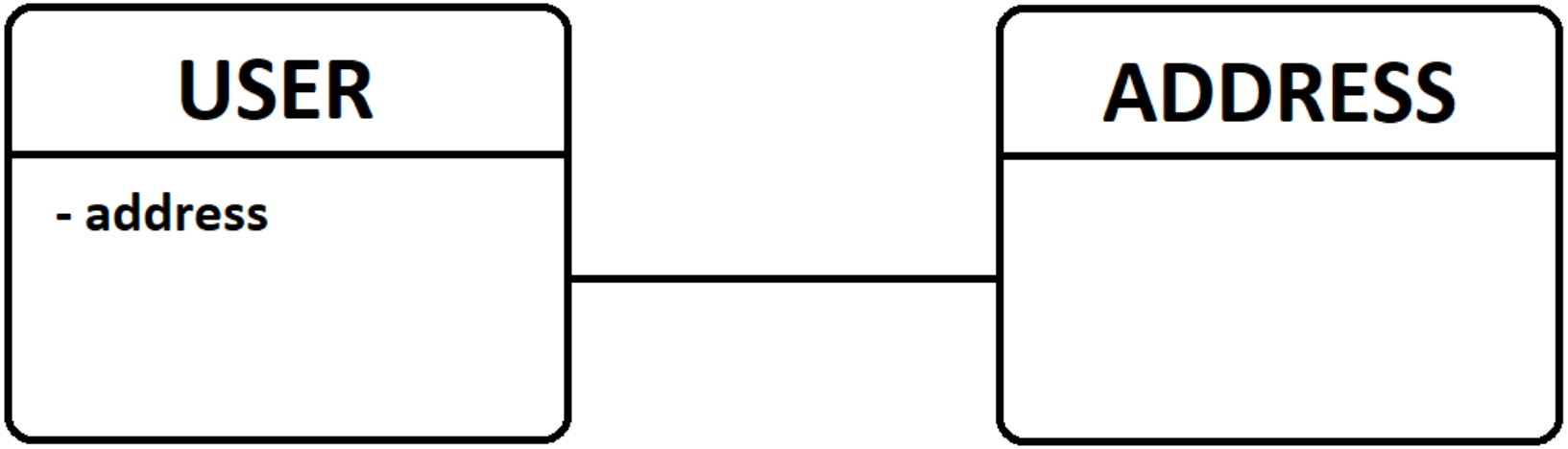
```php
namespace Stratow\StratowBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Table(name="users")
 * @ORM\Entity(repositoryClass="Test\TestBundle\Repository\UsersRepository")
 */
class Users
{
```

```php
<?php

namespace Test\TestBundle\Repository;

use Doctrine\ORM\EntityRepository;

class UsersRepository extends EntityRepository
{

    public function findMagic($securityProfileArray)
    {

        return $this->createQueryBuilder('obj')
            ->orderBy('obj.name', 'ASC')
            ->getQuery()
            ->getResult()
        ;

    }
}
```

Relations entre Entités

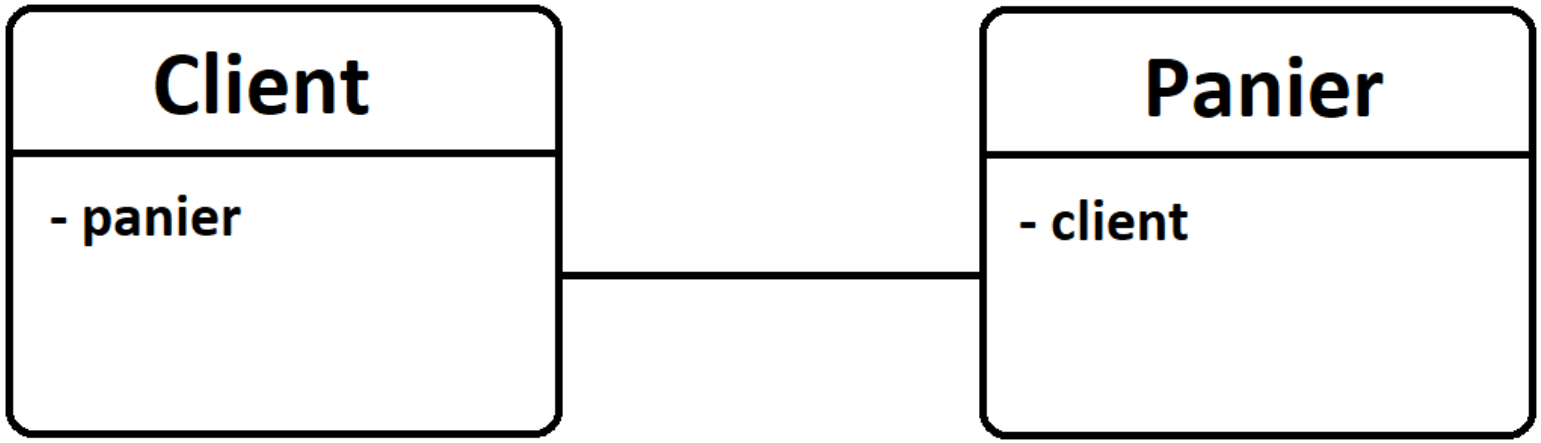
- 4 types de relation
  - OneToOne
  - OneToMany
  - ManyToOne
  - ManyToMany

Relation OneToOne




```php
/**
 * @ManyToOne(targetEntity="Address")
 * @JoinColumn(name="address_id", referencedColumnName="id")
 */
private $address;
```

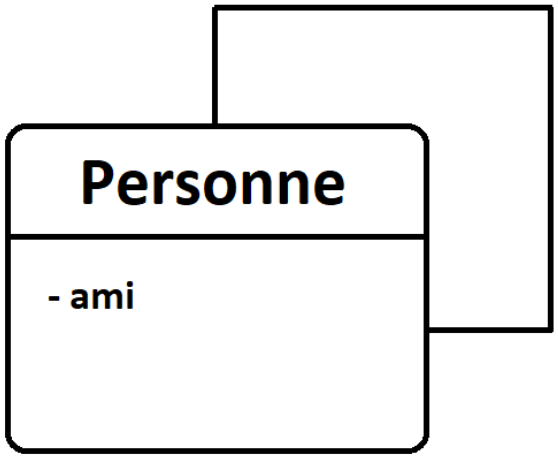
Relation OneToOne




```php
/**
 * @OneToOne(targetEntity="Panier", mappedBy="client")
 */
private $panier;
```

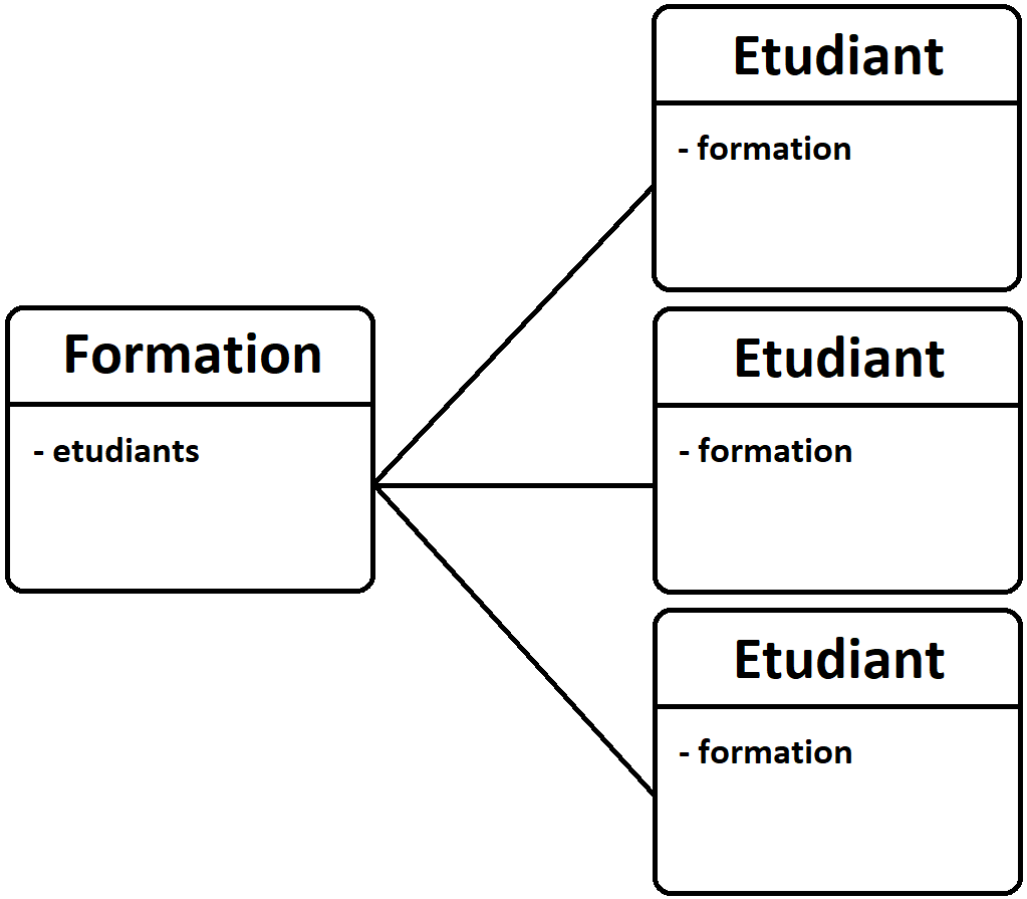
```php
/**
 * @OneToOne(targetEntity="Client", inversedBy="panier")
 * @JoinColumn(name="customer_id", referencedColumnName="id")
 */
private $client;
```

Relation OneToOne




```php
class Personne
{

    /**
     * @OneToOne(targetEntity="Personne")
     * @JoinColumn(name="ami_id", referencedColumnName="id")
     */
    private $ami;

    /* ... */
}
```

Relation ManyToOne / OneToMany

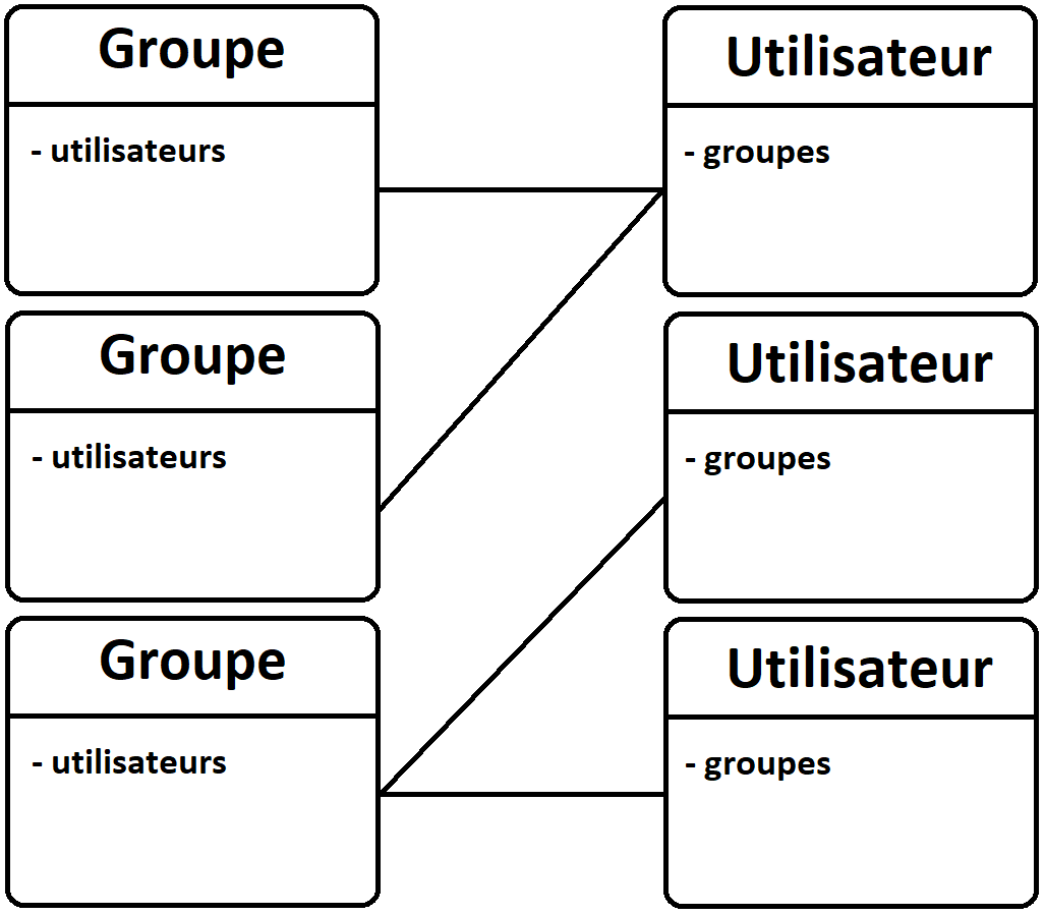


Relation ManyToOne / OneToMany

```php
use Doctrine\Common\Collections\ArrayCollection;

/** @Entity */
class Formation
{
    /**
     * @OneToMany(targetEntity="Etudiant", mappedBy="formation")
     */
    private $etudiants;

    public function __construct() {
        $this->etudiants = new ArrayCollection();
    }

    /* ... */
}
```

```php
/** @Entity */
class Etudiant
{
    /**
     * @ManyToOne(targetEntity="Formation", inversedBy="etudiants")
     * @JoinColumn(name="formation_id", referencedColumnName="id")
     */
    private $formation;

    /* ... */
}
```

Relation ManyToMany



Relation ManyToMany

```php
/** @Entity */
class Groupe
{
    /**
     * @ManyToMany(targetEntity="Utilisateur", mappedBy="groupes")
     */
    private $utilisateurs;

    public function __construct() {
        $this->utilisateurs = new \Doctrine\Common\Collections\ArrayCollection();
    }
}
```

```php
/** @Entity */
class Utilisateur
{
    /**
     * @ManyToMany(targetEntity="Groupe", inversedBy="utilisateurs")
     * @JoinTable(name="groupes_utilisateurs")
     */
    private $groupes;

    public function __construct() {
        $this->groupes = new \Doctrine\Common\Collections\ArrayCollection();
    }
}
```