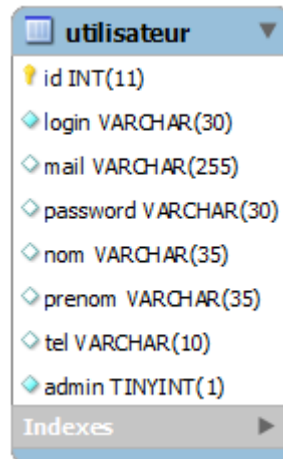


# Micro-ORM

## -- Mappage naturel

- Table  $\leftrightarrow$  Classe
- membre  $\leftrightarrow$  Champ

```
class Utilisateur
extends Base{
    private $id;
    private $login;
    private $mail;
    private $password;
    private $nom;
    private $prenom;
    private $tel;
    private $admin;
}
```



## -- Mappage avec annotations

```
/**
 * @Table(name="Users")
 */
class Utilisateur extends Base{
    /**
     * @Id
     */
    private $id;
    private $login="";
    /**
     * @Column(name=""pwd")
     */
    private $password="";
    private $mail="";
    private $tel;
}
```

Attention, les accesseurs (getters and setters) doivent être implémentés pour le bon fonctionnement des associations.

## -- Relations

## -- ManyToOne

**Exemple** : Chaque utilisateur a un groupe.

Ajout des annotations dans la classe Utilisateur :

```
class Utilisateur extends BaseObject{  
  
    private $login;  
    private $password=null;  
    private $mail;  
    /**  
     * @ManyToOne  
     * @JoinColumn(name="idGroupe",className="Groupe",nullable=true)  
     */  
    private $groupe;
```

### JoinColumn

- **name** : nom de la clé étrangère correspondant au groupe dans l'utilisateur ⇒idGroupe
- **className** : Classe du membre associé

### Chargement

Les membres **manyToOne** sont chargées automatiquement avec les objets qui les contiennent. Si un utilisateur est chargé, le groupe de l'utilisateur l'est également.

```
$aUser=DAO::getOne("Utilisateur",1);  
echo $aUser->getGroupe();
```

## -- OneToMany

**Exemple** : Dans chaque groupe, on a plusieurs utilisateurs.

Ajout des annotations dans la classe Groupe :

```
class Groupe extends BaseObject{  
  
    private $libelle;  
    /**  
     * @OneToMany(mappedBy="groupe",className="Utilisateur")  
     */  
    private $utilisateurs;
```

- **mappedBy** : fait référence au membre groupe de la classe utilisateur
- **className** : Classe du membre associé

## Chargement

Les membres **oneToMany** ne sont pas chargés automatiquement avec les objets qui les contiennent. Il est nécessaire de le faire explicitement dans le code pour qu'ils le soient :

```
$aGroupe=DAO::getOne("Groupe",1);
DAO::getOneToMany($aGroupe,"utilisateurs");
var_dump($aGroupe->getUtilisateurs());
```

## -- Déclaration des ManyToMany

**Exemple** : Les groupes disposent de droits sur les modules.  
Un groupe aura donc une collection de modules représentant ses droits.

```
class Groupe extends BaseObject{
    ...
    /**
     * @ManyToOne(targetEntity="Module", inversedBy="groupes")
     * @JoinTable(name="droit")
     */
    private $modules;
}
```

- **targetEntity** : Classe métier des objets de la collection cible
- **inversedBy** : nom du membre collection dans la classe targetEntity

## JoinTable

- name : nom de la table association correspondant à la CIM.

## Chargement

Les membres **ManyToMany** ne sont pas chargés automatiquement avec les objets qui les contiennent. Il est nécessaire de le faire explicitement dans le code pour qu'ils le soient :

```
$aGroupe=DAO::getOne("Groupe",1);
DAO::getManyToMany($aGroupe,"modules");
var_dump($aGroupe->getModules());
```

From:  
<http://slamwiki2.kobject.net/> - **Broken SlamWiki 2.0**

Permanent link:  
<http://slamwiki2.kobject.net/slam4/micro-framework/orm?rev=1458179723>

Last update: **2019/08/31 14:42**

