



**Benoît Roche**  
@rocheb83

Benoît Roche / @rocheb831



**DOCTRINE : STRUCTURE DE LA BD**



**Doctrine : présentation**

Doctrine appartient à la famille des ORM (Object Relational Mapping)

Doctrine assure la persistance des données traitées sous forme de classes dans les bases de données

Doctrine2 est donc une librairie permettant de gérer les interactions entre une application et une (ou plusieurs) base de données.

Benoît Roche / @rocheb832



DOCTRINE : STRUCTURE DE LA BD



Doctrine est totalement découplé de Symfony et son utilisation est optionnelle

On peut utiliser un autre ORM comme propel

On peut aussi utiliser des requêtes SQL brutes

Doctrine2 se compose de plusieurs couches :

- ✓ DBAL,
- ✓ ORM et
- ✓ Entité



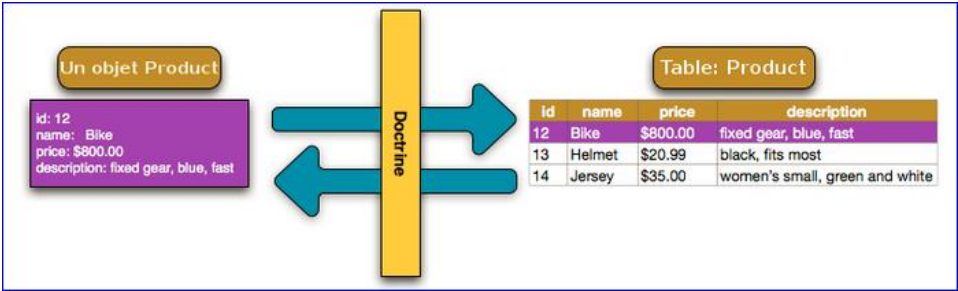
DOCTRINE : STRUCTURE DE LA BD




Un petit exemple


L'objectif de Doctrine est d'associer :

- ✓ des classes PHP avec des tables de la base,
- ✓ des propriétés de ces classes PHP avec des colonnes des tables





DOCTRINE : STRUCTURE DE LA BD



sur un exemple : *Application gestion des employés*

**Classe employe**  
*attributs : id, nom, prenom, date de naissance, salaireM*

➡

**entité employe**

➡

**table employe**

**Classe projet**  
*attributs : id, nomProjet, dureePrevue*

➡

**entité projet**


➡

**table projet**


Et le lien entre les deux ???

Benoît Roche / @rocheb83

5



DOCTRINE : STRUCTURE DE LA BD



**Une vue d'ensemble :**

Base de données

Requêtes SQL ↑   ↓ Résultats

DBAL

ORM

Entité A   Entité B   Entité C   Entité...

Application Symfony2

**DBAL (Database Abstraction Layer)**

**ORM (Object Relational Mapping)**

**Entite on Entity**

Benoît Roche / @rocheb83

6



## DOCTRINE : STRUCTURE DE LA BD



### Une vue d'ensemble :

#### *La couche DBAL (Database Abstraction Layer)*

- ✓ C'est la couche de plus bas niveau.
- ✓ Ne comporte aucune logique applicative et son rôle est d'envoyer des requêtes vers une base de données et de récupérer les résultats.
- ✓ Elle utilise PHP PDO mais elle est plus complète.

Benoît Roche / @rocheb83

7



## DOCTRINE : STRUCTURE DE LA BD



### Une vue d'ensemble :

#### *La couche Entité*

- ✓ Les entités sont les classes d'une application correspondant à des tables en base de données.
- ✓ L'entité est le reflet applicatif d'une table de la base de données, ses propriétés étant équivalentes à des colonnes.
- ✓ On peut donc récupérer, ajouter, modifier et supprimer des données en base sans avoir à écrire une seule ligne de code SQL.
- ✓ Toutes ces actions pouvant être effectuées au travers des objets Entité.

Benoît Roche / @rocheb83

8



Une vue d'ensemble :

*La couche ORM (Object Relational Mapping)*

- ✓ Elle est l'intermédiaire entre l'application et la couche DBAL.
- ✓ Son rôle est de convertir les données tabulaires reçues depuis le DBAL en entités,
- ✓ Elle transforme les interactions avec les différents objets mis à disposition du développeur en requêtes SQL à transmettre
- ✓ au DBAL (notamment pour les mises à jour).
- ✓ Elle établit une correspondance entre la base de données relationnelle et la POO.



**De Symfony à la BD ?  
Ou  
De la BD à Symfony :**

Les deux sont possibles .....

MAIS ....



Il est préférable d'aller de Symfony à la BD

Pourquoi ?

**Pour ne pas perdre en intégrité par une mauvaise interprétation des contraintes**



## DOCTRINE : STRUCTURE DE LA BD



### Configuration :

*Il est impératif de disposer du bundle Doctrine*

*Le bundle Doctrine est en standard dans les vendors de symfony*



### Vendor ??? :

- Répertoire généré par Composer qui contient tous les paquets (librairies ou bundles) dont votre projet dépend.
- Composer est chargé de le générer pour chaque installation/déploiement du projet.
- On ne versionne donc pas ce dossier

Benoît Roche / @rocheb83

11



## PARTIE 1 : LA STRUCTURE DE LA BASE

Benoît Roche / @rocheb83

12



DOCTRINE : STRUCTURE DE LA BD



Les étapes de création :



- 1. Configuration des paramètres
- 2. Création de la base de données
- 3. Générer l'entité employe
- 4. Génération de la table correspondant à l'entité employe



DOCTRINE : STRUCTURE DE LA BD




Les étapes de création :


Configuration des paramètres

Modifier le fichier **app/config/parameters.yml**

```
parameters.yml x
Source History
1 # This file is auto-generated during the compo
2 parameters:
3   database_driver: pdo_mysql
4   database_host: 127.0.0.1
5   database_port: 3306
6   database_name: testsymfony
7   database_user: root
8   database_password: null
9   mailer_transport: smtp
10  mailer_host: 127.0.0.1
11  mailer_user: null
12  mailer_password: null
13  locale: fr
14  secret: ThisTokenIsNotSoSecretChangeIt
```



DOCTRINE : STRUCTURE DE LA BD



Les étapes

**Création de la base de données:**

✓ Dans la console de commande Symfony :

Matching Tasks:

- doctrine:cache:clear-metadata : Clears all metadata cache for an entity
- doctrine:cache:clear-query : Clears all query cache for an entity manager
- doctrine:cache:clear-result : Clears result cache for an entity manager
- doctrine:database:create : Creates the configured databases**
- doctrine:database:drop : Drops the configured databases
- doctrine:ensure-production-settings : Verify that Doctrine is properly configured
- doctrine:generate:crud : Generates a CRUD based on a Doctrine entity

Usage:

`doctrine:database:create [--connection="..."]`

The `doctrine:database:create` command creates the default connections database:

Command: `console doctrine:database:create`


☐ Keep this dialog opened

Refresh Commands Run Cancel

*run command Doctrine database create*

Benoît Roche / @rocheb83

15



DOCTRINE : STRUCTURE DE LA BD



Les étapes

**Création de la base de données**

Vérification de l'existence de la bd à l'aide de Mysql Workbench

Options File

SCHEMAS

Filter objects

- atlantik
- bdemploye
- clicom
- coucou
- dbcours**
- employepoo
- exolecteur
- isenemploye
- port
- requetemagique
- testmwb
- testsymfony**
- wordpress



Benoît Roche / @rocheb83

16



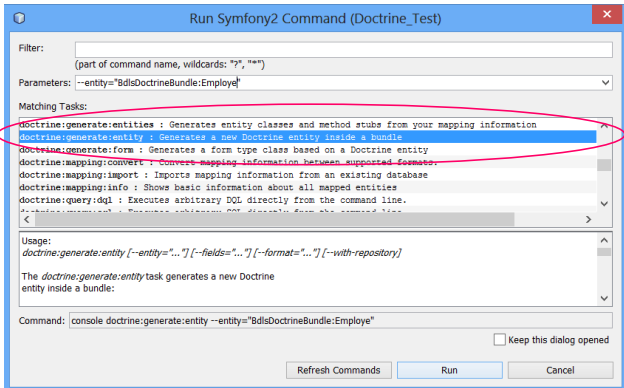


DOCTRINE : STRUCTURE DE LA BD



Les étapes de création :

Générer l'entité Employe  
On va gérer une entité (Entity) dans le bundle DoctrineBundle



Il existe une commande symfony ...



Benoît Roche / @rocheb83

17



DOCTRINE : STRUCTURE DE LA BD



Générer l'entité Employe

run command `Doctrine generate entity--entity="BdlDoctrineBundle:Employe"`

Choisir les annotations par défaut pour les Entités

Determine the format to use for the mapping information.  
Configuration format (yaml, xml, php, or annotation) [annotation]:

Saisir le nom des propriétés

Sauf le champ id !!!

boolean, integer, smallint, bigint, var, decimal,  
date, time, decimal, float, blob, guid.  
New field name (press <return> to stop adding fields): |



Pour chaque champ, on saisit le type (string,integer,...)

les annotations car  
✓ Leur utilisation est intuitive  
✓ les configurations sont souvent plus rapides .

New field name (press <return> to stop adding fields): id  
Field "id" is already defined.

Benoît Roche / @rocheb83

18



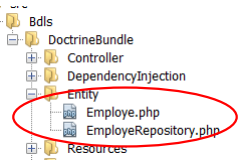
DOCTRINE : STRUCTURE DE LA BD



Générer l'entité Employee

Une fois l'entité créée, on voit une nouvelle entrée dans le bundle :

Le fichier correspondant à la classe employee, employee.php est créé



```
namespace Bdl\DoctrineBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * Employee
 *
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="Bdl\DoctrineBundle\Ent:
 */
class Employee
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;
```



Mais RIEN n'a encore été généré dans la base...



DOCTRINE : STRUCTURE DE LA BD



Générer l'entité Employee

On voit dans l'annotation que le colonne Id a été déclarée clé primaire et qu'elle est en numérotation auto:


```
/**
 * @var integer
 *
 * @ORM\Column(name="id", type="integer")
 * @ORM\Id
 * @ORM\GeneratedValue(strategy="AUTO")
 */
private $id;
```




@ORM\Id : clé primaire  
@ORM\GeneratedValue(strategy="AUTO") : autoincrément

Le contrôle se fait à l'insertion dans la BD





DOCTRINE : STRUCTURE DE LA BD



### Générer l'entité Employe

Pour que Doctrine crée la table dans la base, il va falloir mettre à jour le schéma de la BD :  
C'est la commande schema:update qui va comparer l'état actuel de notre bdd avec nos entités et générer les ordres SQL de créations des tables afin qu'elles correspondent !

Filter:

(part of command name, wildcards: "?", "\*\*")

Parameters: --dump-sql

Matching Tasks:

doctrine:schema:update : Executes (or dumps) the SQL needed to update

doctrine:schema:validate : Validates the doctrine mapping files

generate:bundle : Generates a bundle

generate:controller : Generates a controller

init:acl : Mounts ACL tables in the database

router:debug : Displays current routes for an application

router:dump-apache : Dumps all routes as Apache rewrite rules

Usage:

doctrine:schema:update [--complete] [--dump-sql] [--force] [--em[="..."]]


The doctrine:schema:update command generates the SQL needed to

Permet de générer les ordres sql de mise à jour de la BD


"C:\wamp\bin\php\php5.4.16\php.exe" "E:\Wampsites\Doctrine\_test\app\co  
CREATE TABLE Employe (id INT AUTO\_INCREMENT NOT NULL, nomemp VARCHAR(3  
Done.

Benoît Roche / @rocheb83

21



DOCTRINE : STRUCTURE DE LA BD



### Générer l'entité Employe

Pour que Doctrine crée la table dans la base, il va falloir mettre à jour le schéma de la BD :  
Ici on n'a généré que le sql, on va utiliser l'option force pour créer réellement la table dans la BD :

Run Symfony2 Command (Doctrine\_Test)

Filter:

(part of command name, wildcards: "?", "\*\*")

Parameters: --force

Matching Tasks:

doctrine:schema:update : Executes (or dumps) the SQL needed to update

doctrine:schema:validate : Validates the doctrine mapping files

generate:bundle : Generates a bundle

generate:controller : Generates a controller

init:acl : Mounts ACL tables in the database

router:debug : Displays current routes for an application

router:dump-apache : Dumps all routes as Apache rewrite rules

Doctrine\_Test (generate:bundle) × Doctrine\_Test (doctrine:generate:entities) × D

"C:\wamp\bin\php\php5.4.16\php.exe" "E:\Wampsites\Doctrine\_test\app\co  
Updating database schema...  
Database schema updated successfully! "1" queries were executed  
Done.

Permet de mettre à jour la BD

testsymfony  
└─ Tables  
   └─ employe  
      ├─ Columns  
      │  ├─ id  
      │  ├─ nomemp  
      │  ├─ prenomemp  
      │  └─ salaire  
      └─ Indexes

Benoît Roche / @rocheb83

22

Slam4

11



DOCTRINE : STRUCTURE DE LA BD



Modifier l'entité Employe : rajouter une colonne

en fonction du *mapping* que Doctrine connaît :

- Modifier la classe employé en ajoutant un attribut par exemple ville avec comme valeur par défaut Toulon

```
/**
 * @var string
 *
 * @ORM\Column(name="ville", type="string", length=35 ,options={"default"="Toulon"})
 */
private $ville;
```

- Enregistrer maintenant dans la bdd :

**doctrine:schema:update --dump-sql --force**

- générer les entités

```
Command: console doctrine:generate:entities "BdlsDoctrineBundle:Employe"
```

Il ne manque plus que les getter et les setter que l'on peut générer par une commande symfony ou par netbeans



DOCTRINE : STRUCTURE DE LA BD



Modifier l'entité Employe : rajout de 2 index à la table mappée

Dans les annotations de la table, rajouter une annotation correspondant aux index que l'on veut créer :

index sur la colonne ville,  
Index sur la colonne nomemp

```
/**
 * Employe
 *
 * @ORM\Table("Employe", indexes={@ORM\Index(name="ind_nomemp", columns={"nomemp"}),@ORM\Index(name="ind_ville", columns={"ville"})})
 * @ORM\Entity(repositoryClass="Bdls\DoctrineBundle\Entity\EmployeRepository")
 */
class Employe {
```

Enregistrer maintenant dans la bdd :  
**doctrine:schema:update --dump-sql --force**



On vérifie dans la structure de la table Employe :



## DOCTRINE : STRUCTURE DE LA BD



### De la même façon, on pourra :

- ✓ Créer des clés relations entre entity, de type
  - ✓ **OneToOne**
    - ✓ Elle signifie qu'à UN objet d'une Entity correspond UN objet d'une autre Entity.
    - ✓ Exemple : UN Gardien surveille UN Gymnase(et vice versa)
  - ✓ **ManyToOne** :
    - ✓ Elle signifie qu'à Plusieurs objets d'une Entity correspond UN objet d'une autre Entity
    - ✓ Exemple : PLUSIEURS Employes Travaillent sur UN projet
  - ✓ **ManyToMany**
    - ✓ Elle signifie qu'à Plusieurs objets d'une Entity Correspondent PLUSIEURS objets d'une autre Entity.
    - ✓ Exemple : PLUSIEURS Employes s'inscrivent à PLUSIEURS Seminaires

Benoît Roche / @rocheb83

25



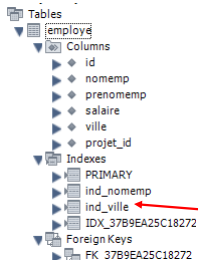
## DOCTRINE : STRUCTURE DE LA BD



### De la même façon, on pourra :

- ✓ Exemple : ManyToOne : On met l'annotation dans l'entity pour créer l'attribut qui jouera le rôle de "clé étrangère" (Entity Employe et Entity Seminaire créés)

```
/**
 * @ORM\ManyToOne(targetEntity="Projet")
 */
private $projet;
```




The doctrine:schema:update command generates the SQL needed

Command: console doctrine:schema:update --dump-sql --force


La clé étrangère a été créée dans la table projet.

Benoît Roche / @rocheb83

26



DOCTRINE : STRUCTURE DE LA BD




De la même façon, on pourra :

✓ Exemple : ManyToMany : On met l'annotation dans l'entity pour créer l'attribut qui jouera le rôle de "clé étrangère"

```
/**
 *
 * @ORM\ManyToMany(targetEntity="Seminaire")
 */
private $lesSeminaires;
```

Ne pas oublier d'instancier la collection dans le constructeur :  
\$lesSeminaires=new....



Tables

employe

employe\_seminaire

projet

seminaire

Une table employe\_seminaire a été créée

Column Name	Datatype	P.	N.	U.	B.	U.	Z.	AI	Default
employe_id	INT(11)	✓	✓						
seminaire_id	INT(11)	✓	✓						

Benoît Roche / @rocheb83

27



DOCTRINE : STRUCTURE DE LA BD



De la même façon, on pourra :

✓ Exemple : ManyToMany : On peut faire mieux en personnalisant la table de jointure : .

```
/**
 * @ORM\ManyToOne(targetEntity="Projet")
 */
private $projet;
```


```
/**
 * @ORM\ManyToMany(targetEntity="Seminaire", mappedBy="lesinscrits",
 * @ORM\JoinTable(name="Inscription",
 *     joinColumns={
 *         @ORM\JoinColumn(
 *             name="idemploye",
 *             referencedColumnName="id"
 *         )
 *     },
 *     inverseJoinColumns={
 *         @ORM\JoinColumn(
 *             name="idseminaire",
 *             referencedColumnName="id"
 *         )
 *     }
 * )
 */
private $lesseminaires;
```

Et il y a beaucoup d'autres options : cascade,...





Benoît Roche / @rocheb83

28



DOCTRINE : STRUCTURE DE LA BD






Exercice:


Il est temps de mettre en pratique tout ceci....


[Voir : exercices Doctrine](#)

Benoît Roche / @rocheb83

29







Fin du cours, merci

Question/Réponses

Benoît Roche

@rocheb83

Benoît Roche / @rocheb83

30