# - QueryBuilder-

Etend les possibilités de la classe Repository

C’est dans les classes Repository :

Exemple : Dans ThemeRepository, ajouter une méthode getAllThemes(pour l’exemple)

**public function** getAllThemes() {  
 // Recupérer l'instance de QueryBuilder. 't' est l'alias THEME car on est dans ThemeRepository  
 $qb = $this->createQueryBuilder('t');  
  
 $qb->select("t.name, count(p) as numerofPosts")  
 // 1ere argument est la variable qui stocke l'association  
 // le 2e argument représente l'alias de la 2e entité. En l'occurence c'est Post  
 ->innerJoin("t.posts", "p")  
 ->groupBy("t.id");  
  
 **return** $qb->getQuery();  
 }

A la fin de la méthode, faire un return de getQuery().

Puis dans l’action de notre Controller, appeler le getAllThemes via le repository. Voir code:

$repository = $this->getDoctrine()  
 ->getRepository("AppBundle:Theme");  
  
 $themeList = $repository->findAll();  
// Pour les tests du QueryBuilder  
$list = $repository->getAllThemes()->getArrayResult();

Voir doc à partir de la page 100

Installation de oro/doctrine-extensions

Utiliser certaines fonctions sql dans QueryBuilder car Doctrine ne le gère pas

composer require oro/doctrine-extensions

voir la doc :

<https://github.com/oroinc/doctrine-extensions>

pour la suite de l’installation pour la modification du fichier config.yml

Exemple 2 : Ajout d’une autre méthode dans le Repository avec QueryBuilder dans l’entité Post

**public function** getPostsGroupedByYear() {  
 $qb = $this->createQueryBuilder('p');  
  
 $qb->select('YEAR(p.createdAt) as yearPublished,  
 COUNT(p.id) as numberOfPosts ')  
 ->groupBy('YEAR(yearPublished)');  
  
 **return** $qb->getQuery()->getArrayResult();  
}