Doctrine ORM: Acesse o banco com Mapeamento Objeto-Relacional

- ORM: uma ferramenta que auxilia a mapear um modelo orientado a objetos para um modelo relacional no banco de dados.
- Migrations
- Annotations
- A forma mais comum de se mapear uma entidade no Doctrine é, sem dúvida, através de anotações, mas não é a única. Existem algumas pessoas que defendem que anotações "sujam" o nosso código de domínio, com informações de infraestrutura (como o banco de dados). Por isso, existe a possibilidade de mapear nossas entidades utilizando arquivos de configuração, em XML, por exemplo. Para utilizar um XML, ao invés de anotações, por exemplo, você precisaria mudar o código da sua EntityManagerFactory para buscar os dados no arquivo correto.
- Para o Doctrine, uma entidade é uma instância de uma classe em PHP que pode ser mapeada para o banco de dados.
- O Doctrine mapeia todas as alterações em memória antes (usando persist()), e envia de uma só vez ao banco (usando flush()), otimizando (e muito) a performance da aplicação. An entity can be made persistent by passing it to the EntityManager#persist($entity) method. By applying the persist operation on some entity, that entity becomes MANAGED, which means that its persistence is from now on managed by an EntityManager. As a result the persistent state of such an entity will subsequently be properly synchronized with the database when EntityManager#flush() is invoked. Invoking the persist method on an entity does NOT cause an immediate SQL INSERT to be issued on the database. Doctrine applies a strategy called transactional write-behind, which means that it will delay most SQL commands until EntityManager#flush() is invoked which will then issue all necessary SQL statements to synchronize your objects with the database in the most efficient way and a single, short transaction, taking care of maintaining referential integrity. Example:

```
<?php
$user = new User;
$user->setName('Mr.Right');
$em->persist($user);
$em->flush();
```

- An entity can be removed from persistent storage by passing it to the EntityManager#remove($entity) method. By applying the remove operation on some entity, that entity becomes REMOVED, which means that its persistent state will be deleted once EntityManager#flush() is invoked. Just like persist, invoking remove on an entity does NOT cause an immediate SQL DELETE to be issued on the database. The entity will be deleted on the next invocation of EntityManager#flush() that involves that entity. This means that entities scheduled for removal can still be queried for and appear in query and collection results. Example:

```
<?php
$em->remove($user);
$em->flush();
```

- To query for one or more entities based on several conditions that form a logical conjunction, use the findBy and findOneBy methods on a repository as follows:

```
<?php
// $em instanceof EntityManager
// All users that are 20 years old
$users = $em->getRepository('MyProject\Domain\User')->findBy(array('age' => 20));
// All users that are 20 years old and have a surname of 'Miller'
$users = $em->getRepository('MyProject\Domain\User')->findBy(array('age' => 20,
'surname' => 'Miller'));
// A single user by its nickname
$user = $em->getRepository('MyProject\Domain\User')->findOneBy(array('nickname' =>
'romanb'));
```

- findBy() pode ter 4 parâmetros
  - $criteria: Critério de busca. Array vazio significa sem critério, ou seja, sem filtro, buscando todos os registros;
  - $orderBy: Critério de ordenação. Um array onde as chaves são os campos, e os valores são 'ASC' para ordem crescente e 'DESC' para decrescente;
  - $limit: Numéro de resultados para trazer do banco;
  - $offset: A partir de qual dado buscar do banco. Muito utilizado para realizar paginação de dados.
- repository->findAll()

- Collections

- Sempre que temos uma relação no Doctrine na qual um dos lados está no plural (ou seja, é uma coleção), nós definimos o tipo dele como uma coleção do Doctrine. Para isso, criaremos o construtor dessa classe e definiremos que $this->telefones receberá uma nova coleção do Doctrine. Além do ORM, o Doctrine oferece uma biblioteca de manipulação de conjuntos de dados. Um dos mais comuns, e mais simples de entender, é o ArrayCollection, uma coleção do Doctrine que se comporta como um array e oferece algumas funcionalidades interessantes.

- Migrations
- inversedBy
- mappedBy

- DQL: Doctrine Query Language
- Caso o relacionamento em questão for ser utilizado todas as vezes, faz todo sentido utilizar o fetch="EAGER". Em nosso exemplo, toda listagem de alunos contém seus telefones, logo, é compreensível e lógico que eles sejam sempre buscados ansiosamente, ganhando assim performance em todo sistema.
    - O que queremos, então, é que o Doctrine, quando buscarmos um $aluno, traga também os telefones dele. Com o parâmetro fetch, é possível informarmos qual o formato de busca que o Doctrine deverá utilizar. Por padrão, esse parâmetro recebe o valor LAZY (preguiçoso), ou seja, ele só busca os telefones se isso realmente for necessário. Mudaremos esse valor para EAGER (ansioso), de modo que o Doctrine trará os telefones imediatamente junto com a busca por $aluno. Assim, quando chamarmos o método getTelefones(), os dados já estarão inicializados e não precisaremos executar uma query adicional para isso. Para testarmos se o número de SQLs foi reduzido, voltaremos ao Prompt de Comando e executaremos php commands\relatorio-cursos-por-aluno.php. Dessa vez, ao invés de 5 queries, teremos apenas 3. Isso porque o Doctrine, quando traz os dados de aluno, traz também os telefones (se eles existem no aluno).
- Para buscarmos os cursos por aluno, precisaremos do $entityManager. Como o repositório irá trabalhar com o Doctrine, não faz sentido ele ter acesso ao $entityManager, ou mesmo a outras operações do Doctrine? Faz sentido, e isso é possível por meio da herança da extensão EntityRepository. um repositório base que nos é fornecido pelo próprio Doctrine.

- The QueryBuilder - Doctrine Object Relational Mapper (ORM) (doctrine-project.org)

Dúvidas e assuntos a pesquisar

- Getting Started with Doctrine - Doctrine Object Relational Mapper (ORM) (doctrine-project.org)
- Basic Mapping - Doctrine Object Relational Mapper (ORM) (doctrine-project.org)
- Association Updates: Owning Side and Inverse Side - Doctrine Object Relational Mapper (ORM) (doctrine-project.org)