Doctrine 2

Camada de persistência para PHP

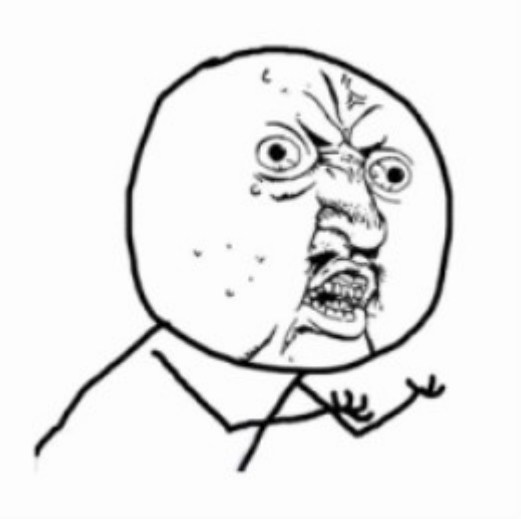
Fabio B. Silva :
fabiosilva.info / fabio.bat.silva@gmail.com / @FabioBatSilva

Quem é esse cara ai ?

- **Fabio B. Silva**
- **Desenvolvedor php**
- **Desenvolvedor java**
- **Doctrine core developer**
- **@FabioBatSilva**
- **github.com/FabioBatSilva**
- ***Alcoolatra nas horas vagas***

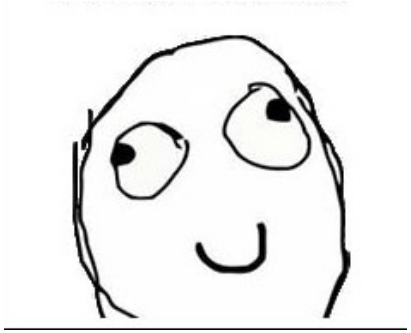
Doctrine ORM
Camada de persistência para PHP

Espera ai, o que é esse tal ORM ?



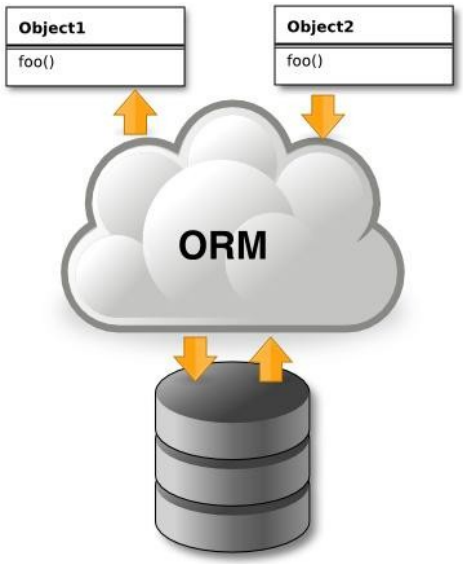
Espera ai, o que é esse tal ORM ?

O ORM (Object-relational mapping) é uma camada intermediária entre um banco de dados relacional e objetos. De um lado, você tem as tabelas e seus relacionamentos, do outro os objetos de uma linguagem OO.



O que um ORM faz ?

- Mapeia dados do DB para Objetos
- Mapeia relacionamentos
- Lida com conversão de tipos
- Normalmente é cross database

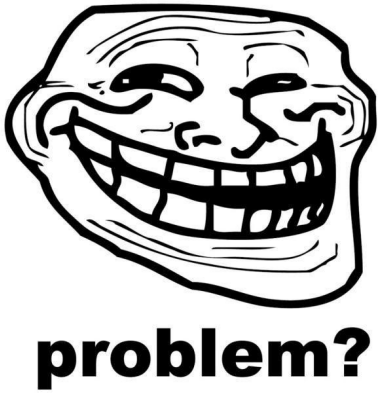


Quais problemas ele resolve ?

- Agiliza o desenvolvimento
- Diminui a quantidade de código
- Ótimo na maioria das situações
- Te ajuda a usar o poder dos objetos
- Faz parte de um bom design de software

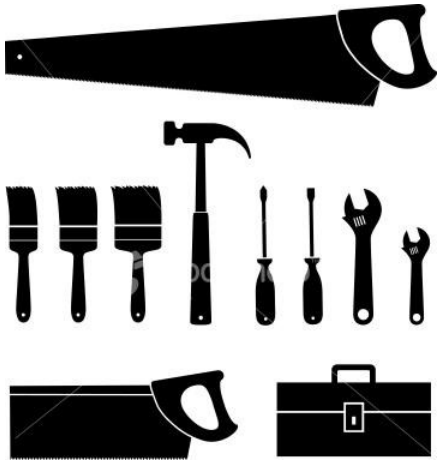
Quais problemas você pode ter ?

- Curva de aprendizado
- Requer conhecimentos em OOP
- Performance
- Não resolve todos os problemas



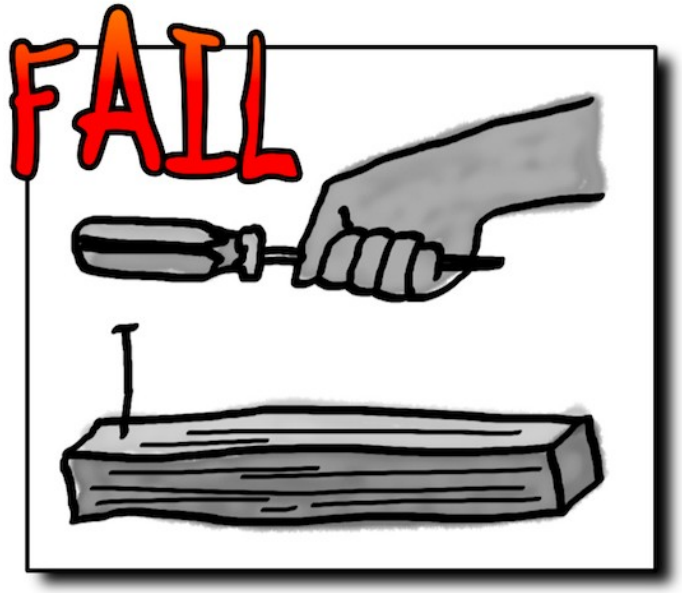
Em quais casos devo usar ?

- Se você gosta de OOP
- Se estiver cansado de SQL
- Se quiser algo ágil e de qualidade



Em quais casos NÃO devo usar ?

- Se não conhece OOP
- Se estiver desenvolvendo o facebook
- Se sua aplicação for realmente pequena



ORM PHP

- Doctrine
- Propel
- RedBeanPHP
- PHPActiveRecord

ORM PHP

Esqueça os outros, vamos falar de doctrine !

Doctrine 1 : *De onde viemos*

- PHP 5.2.3 +
- Baseado no Active Record
- Suporte a Migrations
- Command line
- Cache
- Fácil de usar

Doctrine 1 : *De onde viemos*

```php
class User extends Doctrine_Record {

    public function setTableDefinition()
    {
        $this->setTableName('user');

        $this->hasColumn('id', 'integer', 4, array(
                'type'          => 'integer',
                'length'        => 4,
                'unsigned'      => true,
                'primary'       => true,
                'autoincrement' => true
        ));

        $this->hasColumn('email', 'string', 255, array(
                'type'      => 'string',
                'email'     => true,
                'unique'    => true,
                'unsigned'  => false,
                'notnull'   => true,
                'notblank'  => true,
        ));
```

Doctrine 1 : *De onde viemos*

```
$user         = new User();
$user->name = "FabioBatSilva";

$user->save();


$user = Doctrine_Query::create()
              ->from('User')
              ->where('name = ?', 'FabioBatSilva')
              ->fetchOne();

$user->remove();
}
```

Doctrine 1 : *Problemas*

- Lento
- ActiveRecord
- Difícil de testar
- Alto consumo de memória
- ***Mesmo assim o melhor ORM disponível na época***

Doctrine 2 : Onde estamos

- PHP 5.3 +
- Totalmente Rescrito
- Otimizado
- Componentes
  - DBAL
  - Common
  - ORM
- Versão atual 2.2 (2.3 deve sair em julho)

DBAL

- Database Abstraction Layer
- Multi Plataforma
- SQL Query Builder
- Schema-Manager

Common

- Annotations
- Collections
- Events
- Cache

Common

Annotatinons

```php
/**
 * @Annotation
 * @Target({"CLASS","PROPERTY"})
 */
final class MyAnnot
{
    /** @var array<string> */
    public $value;
}

/** @MyAnnot({"Foo"}) */
class MyClass
{
    /** @MyAnnot({"Bar"}) */
    public $myProperty;
}

$reader = new AnnotationReader();
$class  = new ReflectionClass('MyClass');
$prop   = $class->getProperty('myProperty');

$classAnnots = $reader->getClassAnnotations($class);
$propAnnots  = $reader->getPropertyAnnotations($prop);
```

Common

Collections

```php
$collection = new ArrayCollection;

$collection->add('foo');
$collection->add('bar');

$collection->contains('foo');
// true

$collection->count();
// 2

$collection->remove(0);

$collection->count();
// 1
```

Common

Cache

```php
$cache = new ApcCache;

$cache->save('test_key', 'Some Value');

$cache->contains('test_key');
// true

$cache->fetch('test_key');
// "Some Value"

$cache->delete('test_key');
```

ORM

- Baseado no JPA (Hibernate / EJB / Nhibernate)
- Construído sobre componentes
- Data Mapper
- Objetos Simples
- DQL
- Otimizado
- Performance

Classe Simples

Nada de estender classes do ORM

```php
/**
 * @Entity
 * @Table(name="users")
 */
class User
{
    /**
     * @Id
     * @GeneratedValue
     * @Column(type="integer")
     */
    protected $id;

    /**
     * @Column(type="string", length=255)
     */
    protected $name;

    // getters and setters
}
```

EntityManager

Ponte central de acesso as funcionalidades do ORM

```php
$user = new User;
$user->setName('FabioBatSilva');

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

EntityManager

Ponte central de acesso as funcionalidades do ORM

```php
$user = $em->find('MyProject\Entity\User', 123);

$user->setName('Fabio');
$em->persist($user);
$em->flush();
```

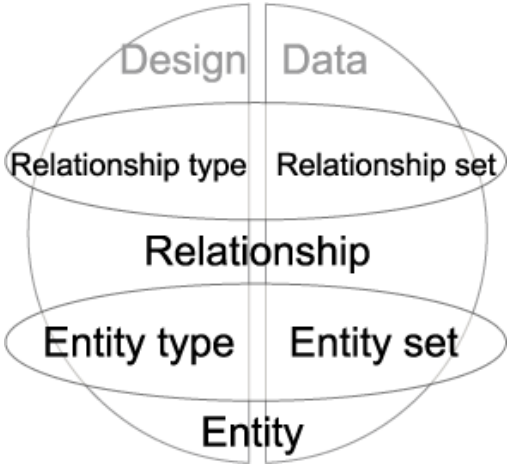
EntityManager

Ponte central de acesso as funcionalidades do ORM

```php
$user = $em->find('MyProject\Entity\User', 123);

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

Relacionamentos

- OneToOne
- ManyToOne
- OneToMany
- ManyToMany



Relacionamentos

OneToOne

```php
/**
 * @Entity
 * @Table(name="address")
 */
class Address
{
    // other properties

    /**
     * @OneToOne(targetEntity="User")
     * @JoinColumn(name="user_id", referencedColumnName="id")
     */
    protected $user;
}
```

Relacionamentos

ManyToOne

```php
/**
 * @Entity
 * @Table(name="articles")
 */
class Article
{
    // other properties

    /**
     * @ManyToOne(targetEntity="User")
     * @JoinColumn(name="user_id", referencedColumnName="id")
     */
    protected $user;
}
```

Relacionamentos

OneToMany

```php
/**
 * @Entity
 * @Table(name="users")
 */
class User
{
    // other properties

    /**
     * @OneToMany(targetEntity="Article", mappedBy="user")
     */
    protected $articles;
}
```

Relacionamentos

OneToMany

```php
/**
 * @Entity
 * @Table(name="users")
 */
class User
{
    // other properties

    /**
     * @OneToMany(targetEntity="Article", mappedBy="user")
     */
    protected $articles;
}
```

Mapeamento

Mapeie o lado inverso apenas quando for usa-lo

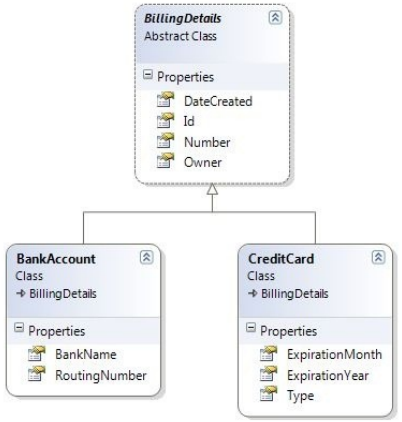
Relacionamentos

ManyToMany

```php
/**
 * @Entity
 * @Table(name="users")
 */
class User
{
    // other properties

    /**
     * @ManyToMany(targetEntity="Group")
     * @JoinTable(name="users_groups",
     *  joinColumns={
     *    @JoinColumn(name="user_id", referencedColumnName="id")
     *  },
     *  inverseJoinColumns={
     *    @JoinColumn(name="group_id", referencedColumnName="id")
     *  }
     * )
     */
    public $groups;
}
```

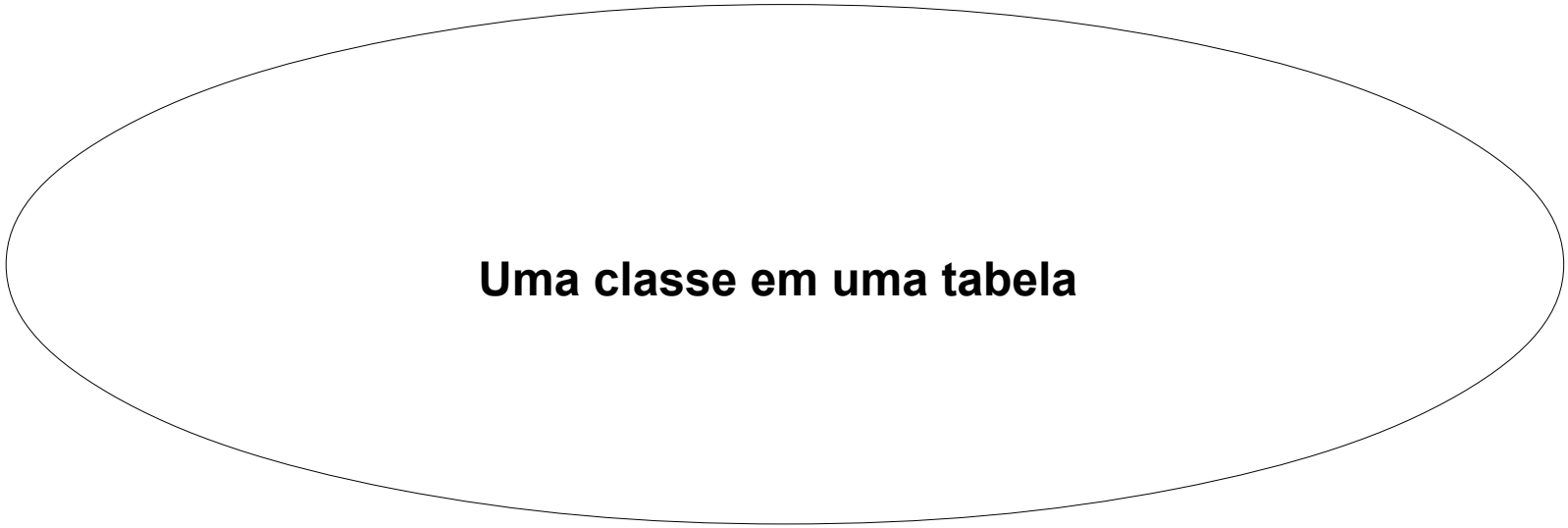
Herança

- Concrete Table Inheritance
- Single Table Inheritance
- Class Table Inheritance



Herança

- Concrete Table Inheritance
- Single Table Inheritance
- Class Table Inheritance



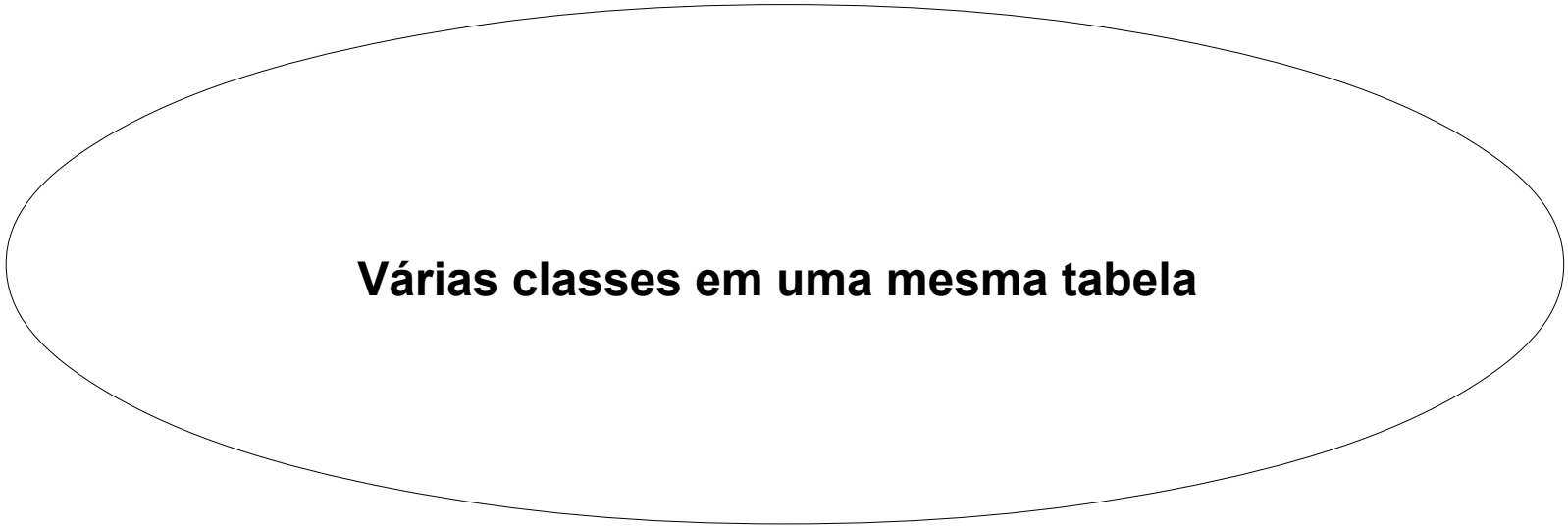
Herança

Concrete table inheritance

```php
/** @MappedSuperclass */
abstract class AbstractContentItem
{
    /** @Id @Column(type="integer") @GeneratedValue */
    protected $id;

    /** @ManyToOne(targetEntity="Directory") */
    protected $parent;

    /** @column(type="string") */
    protected $name;
}

/** @Entity @Table(name="directory") */
class Directory extends AbstractContentItem
{
    /** @Column(type="string") */
    protected $path;
}

/** @Entity @Table(name="file") */
class File extends AbstractContentItem
{
    /** @Column(type="string") */
    protected $extension;
}
```

Herança

- Concrete Table Inheritance
- Single Table Inheritance
- Class Table Inheritance



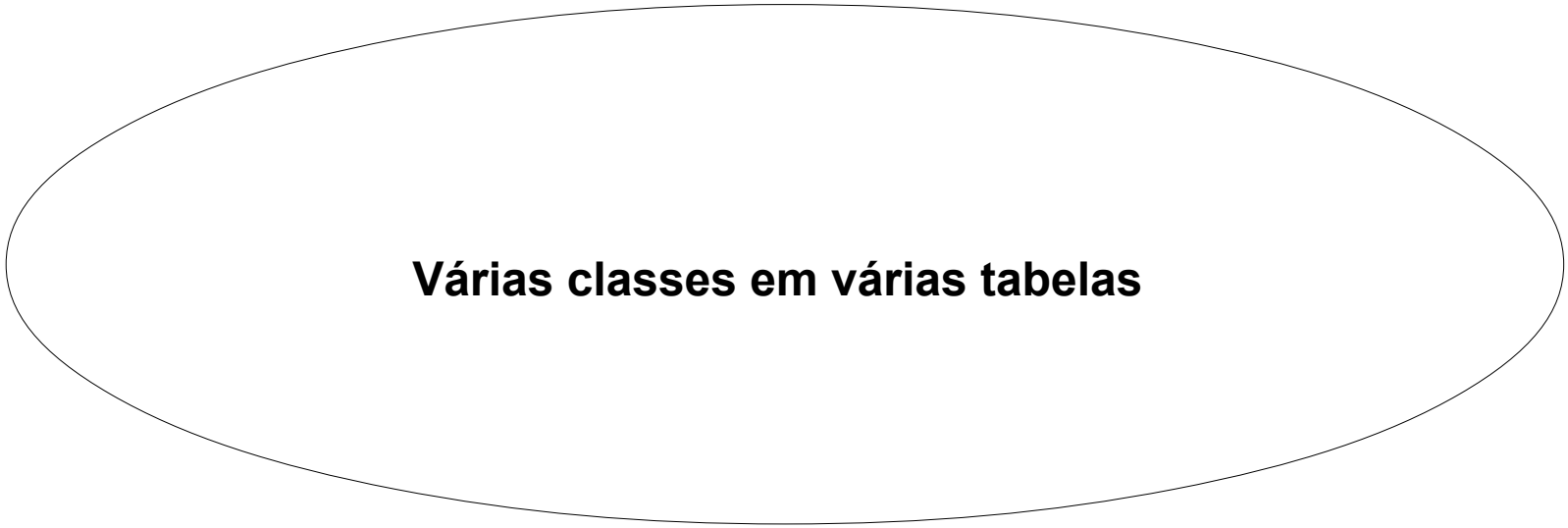
Herança

Single table inheritance

```php
/**
 * @Entity @Table(name="contracts")
 * @InheritanceType("SINGLE_TABLE")
 * @DiscriminatorColumn(name="discr", type="string")
 * @DiscriminatorMap({
 *     "fix"   = "FixContract",
 *     "flex"  = "FlexContract"
 * })
 */
abstract class Contract
{
    /** @Id @column(type="integer") @GeneratedValue */
    protected $id;
}
/** @Entity */
class FixContract extends Contract
{
    /** @column(type="integer") */
    protected $fixPrice = 0;
}
/** @Entity */
class FlexContract extends Contract
{
    /** @column(type="integer") */
    protected $hoursWorked = 0;

    /** @column(type="integer") */
    protected $pricePerHour = 0;
}
```

Herança

- Concrete Table Inheritance
- Single Table Inheritance
- Class Table Inheritance



Herança

Class table inheritance

```php
/**
 * @Entity
 * @Table(name="persons")
 * @InheritanceType("JOINED")
 * @DiscriminatorColumn(name="discr", type="string")
 * @DiscriminatorMap({
 *      "person"    = "Person"
 *      "employee"  = "Employee"
 * })
 */
class Person
{
    /** @Id @Column(type="integer") @GeneratedValue */
    protected $id;

    /** @Column */
    protected $name;
}

/** @Entity @Table(name="employees") */
class Employee extends Person
{
    /** @Column(type="integer") */
    protected $salary;

    /** @Column(type="string", length=255) */
    protected $department;
}
```

DQL
Doctrine Query Language

- DQL : Doctrine Query Language
- Usa Classes e propriedades invés de tabelas e colunas
- Normalmente retorna uma lista de entidades
- Parseado para SQL nativa
- Cross database

DQL

```
$dql   = "SELECT u FROM MyProject\Entity\User u";
$query = $em->createQuery($dql);
$users = $query->getResults();

foreach ($users as $user) {
    echo $user->getName();
}
```

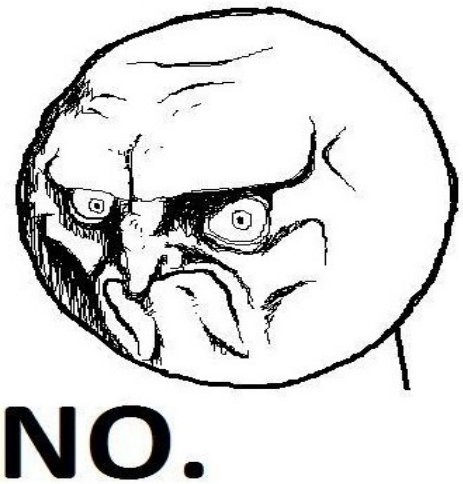
DQL

```
$dql   = "SELECT u FROM MyProject\Entity\User u";
$query = $em->createQuery($dql);
$users = $query->getResults();

foreach ($users as $user) {

    echo $user->getName();

    foreach ($user-getArticles() as $article) {
        echo $article->getName();
    }

}
```

DQL

```
$dql   = "SELECT u FROM MyProject\Entity\User u";
$query = $em->createQuery($dql);
$users = $query->getResults();

foreach ($users as $user) {

    echo $user->getName();

    foreach ($user-getArticles() as $article) {
        echo $article->getName();
    }

}
```

Você esta fazendo isso errado !!!



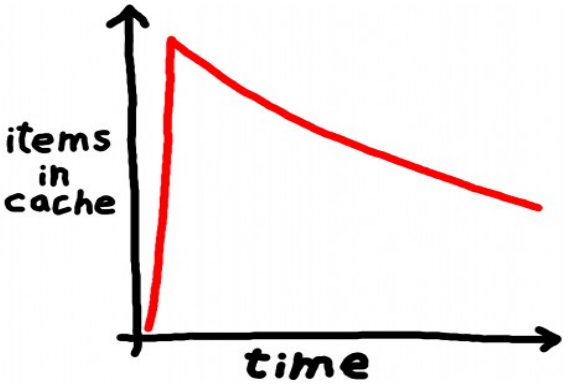
DQL

```php
$dql   = "SELECT u, a FROM MyProject\Entity\User u JOIN u.articles a";
$query = $em->createQuery($dql);
$users = $query->getResults();

foreach ($users as $user) {

    echo $user->getName();

    foreach ($user-getArticles() as $article) {
        echo $article->getName();
    }

}
```

Cache

Melhor amigo da performance

- Metadata Cache
- Query Cache
- Result Cache



Cache

Melhor amigo da performance

- Metadata Cache
- Query Cache
- Result Cache

```php
/** @var Doctrine\ORM\Configuration */
$config->setMetadataCacheImpl(new ApcCache());
```

Cache dos mapeamentos das entidades

Cache

Melhor amigo da performance

- Metadata Cache
- Query Cache
- Result Cache

```php
/** @var Doctrine\ORM\Configuration */
$config->setQueryCacheImpl(new ApcCache());
```

Cache dos parser de DQL para SQL nativa

Cache

Melhor amigo da performance

- Metadata Cache
- Query Cache
- Result Cache

```php
/** @var Doctrine\ORM\Configuration */
$config->setResultCacheImpl(new ApcCache());

$query = $em->createQuery('SELECT u from MyProject\Entity\User u');
$query->useResultCache(true, 3600, 'query_cache_id');
```

Cache dos resultados de consultas

Conclusão

- Simplifica as coisas
- Maduro e estável
- Cresce a cada dia
- Muita fácil de integrar com Symfony2
- Doctrine é PHODA !!

Doctrine 2

Camada de persistência para PHP

Perguntas ???

Fabio B. Silva :
fabiosilva.info / fabio.bat.silva@gmail.com / @FabioBatSilva