Doctrine ORM

About me

Name: Aivaras Spaičys
Skype: xz33rox (Aivaras Spaičys)
Email: spaivaras@gmail.com
Occupation: Software engineer @ NFQ Technologies

Slides and source: http://prodigy.lt/kaunasPHP

What is Doctrine project?

- It’s a set of PHP libraries mainly focused around persistence services for data management, such as:
  - *Database Abstraction Layer*
  - *Object Relational Mapper 2*
  - *MongoDB Object Document Mapper*
  - *Migrations*
  - *Etc.*

Why should I care ?

- Its open source (MIT)
- Has a big community interest
- Development started around 2006
- Comes out the box with Symfony2
- Easily integrated with Zend 2 or pure PHP projects
- Flexible: you can write own data types, can interrupt DB sync operations, etc.
- Lets you **forget** about database and concentrate more on your code design

Doctrine ORM

- Your custom objects (Entities) instead of MySQL result set
- Uses Java style annotations, yaml or xml for mapping information
- Data can be manipulated using DQL, QueryBuilder or native SQL
- Data persistence and DB sync are separated processes
- Tools for validating, creating code, database schema operations

- Bytecode Cache is highly recommended
- Metadata and Query cache **is a must!**
- Some cache mechanism like **eAccelerator** can break annotations parser
- Doctrine will not help from poor decisions :)

Installing Doctrine ORM

- Composer.json

```
{
    "require": {
        "doctrine/orm": "2.*",
        "symfony/yaml": "2.*"
    },
    "autoload": {
        "psr-0": {"": "/"}
    }
}
```

Installing Doctrine ORM

- Bootstrap.php

```php
<?php
use Doctrine\ORM\Tools\Setup;
use Doctrine\ORM\EntityManager;

require_once "vendor/autoload.php";

// Create a simple "default" Doctrine ORM configuration for Annotations
$isDevMode = true;
$config = Setup::createAnnotationMetadataConfiguration(array(__DIR__."/Entity"), $isDevMode);

$conn = array(
    'driver' => 'pdo_mysql',
    'user' => 'dbuser',
    'password' => 'dbpass',
    'dbname' => 'doctrine',
);

// obtaining the entity manager
$entityManager = EntityManager::create($conn, $config);
```

First entity!!

- Entity/Author.php

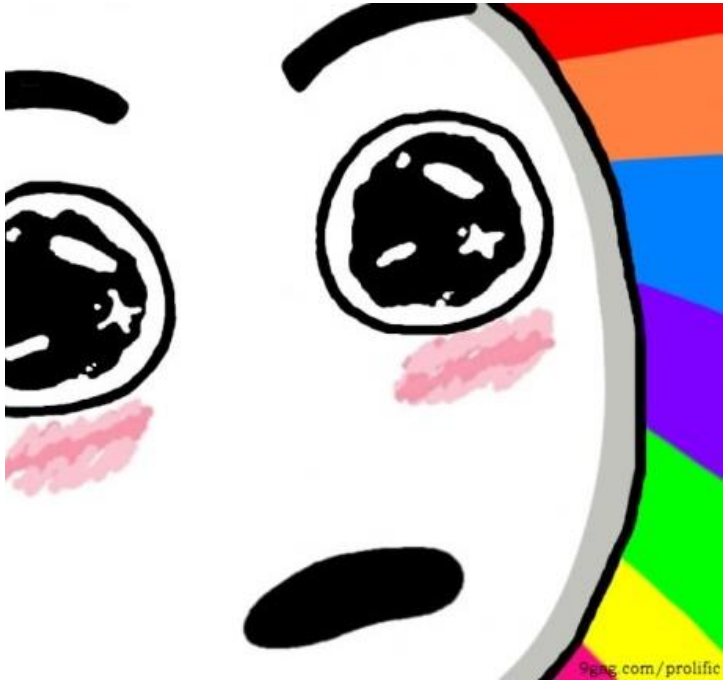
```php
<?php
namespace Entity;

/**
 * @Entity
 * @Table(name="authors")
 **/
class Author {

   /**
    * @Id
    * @Column(type="integer")
    * @GeneratedValue
    **/
   protected $id;

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

First entity!!

- There are more built-in types:
  - *string*
  - *text*
  - *Integer*
  - *float*
  - *boolean*
  - *date*
  - *etc.*
- There are more options for column annotation:
  - *unique*
  - *nullable,*
  - *precision,*
  - *etc.*

First entity!!

- Generating method stubs
  - *php vendor/bin/doctrine orm:generate-entities .*
- Creating database schema
  - *php vendor/bin/doctrine orm:schema-tool:create*
- Or updating it from mapped entities
  - *php vendor/bin/doctrine orm:schema-tool:update --force*



Relations? No problem!

- Entity/Author.php

```
/**
 * @OneToMany(targetEntity="Comment", mappedBy="author")
 */
protected $comments;
```

- Entity/Comment.php

```
/**
 * @ManyToOne(targetEntity="Author", inversedBy="comments")
 */
protected $author;
```

Relations? No problem!

- Relations can be unidirectional or bidirectional
- All basic relations are supported:
  - *OneToOne*
  - *ManyToOne*
  - *OneToMany*
  - *ManyToMany*
- Relations can handle transitive persistence (cascade operations)
- Each operation can be configured independently
  - *Persist*
  - *Remove*
  - *Merge*
- Collections can be filtered no matter were they loaded or not

Working with entities

- Objects are fetched via semi transparent objects called **Repositories**
- By default each Entity has a repository object with default methods:
  - *find(), findAll(), findByVariable(), findBy(array(…))*
- Data is manipulated thru entity object itself
- If more advanced data manipulation is required physical repositories can be created with custom methods using DQL, QueryBuilder or native sql

Working with entities

```php
$author = new Entity\Author();
$author->setName("Aivaras");

$entityManager->persist($author);
$entityManager->flush();
```

```php
$author = $entityManager->getRepository('Entity\Author')
                        ->find($authorId);
```

```php
$author = $entityManager->getRepository('Entity\Author')
                        ->findByName("Aivaras");
```

Custom repository

- Repository/CommentRepository.php

```php
namespace Repository;
use Doctrine\ORM\EntityRepository;

class CommentRepository extends EntityRepository
{
    public function getBadComments($max = 10)
    {
        $dql = "SELECT c,a FROM Entity\Comment AS c
                INNER JOIN c.author AS a
                WHERE c.text LIKE :filter";

        return $this->getEntityManager()->createQuery($dql)
                    ->setParameter('filter', "%bad word%")
                    ->setMaxResults($max)
                    ->getResult();
    }
}
```

Custom repository

- Entity/Comment.php

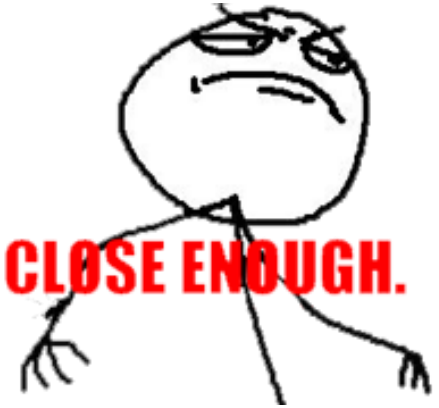
```php
<?php
namespace Entity;

/**
 * @Entity(repositoryClass="Repository\CommentRepository")
 * @Table(name="comments")
 **/
class Comment {
```

- List_comments.php

```php
<?php
require_once "bootstrap.php";

$comments = $entityManager->getRepository('Entity\Comment')
                          ->getBadComments(5);

foreach ($comments as $comment) {
   echo "Bad Comment: " . $comment->getText() . "\n";
   echo "Author: " . $comment->getAuthor()->getName() . "\n\n";
}
```

What's next? I am hyped

- There are quite a lot of things I didn’t covered, for example:
  - Lifecycle callback
  - Persist and Flush internals
  - Custom types
  - Migrations
  - And so on…



- More to read on:
  - http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/tutorials/getting-started.html
  - http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/index.html
  - http://symfony.com/doc/master/book/doctrine.html

Questions?