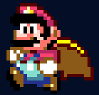
DOCTRINE 2 & ZF2

+



MARCO PIVETTA

Doctrine core team

Zf2 contributor

Modules ~~developer~~ time waster

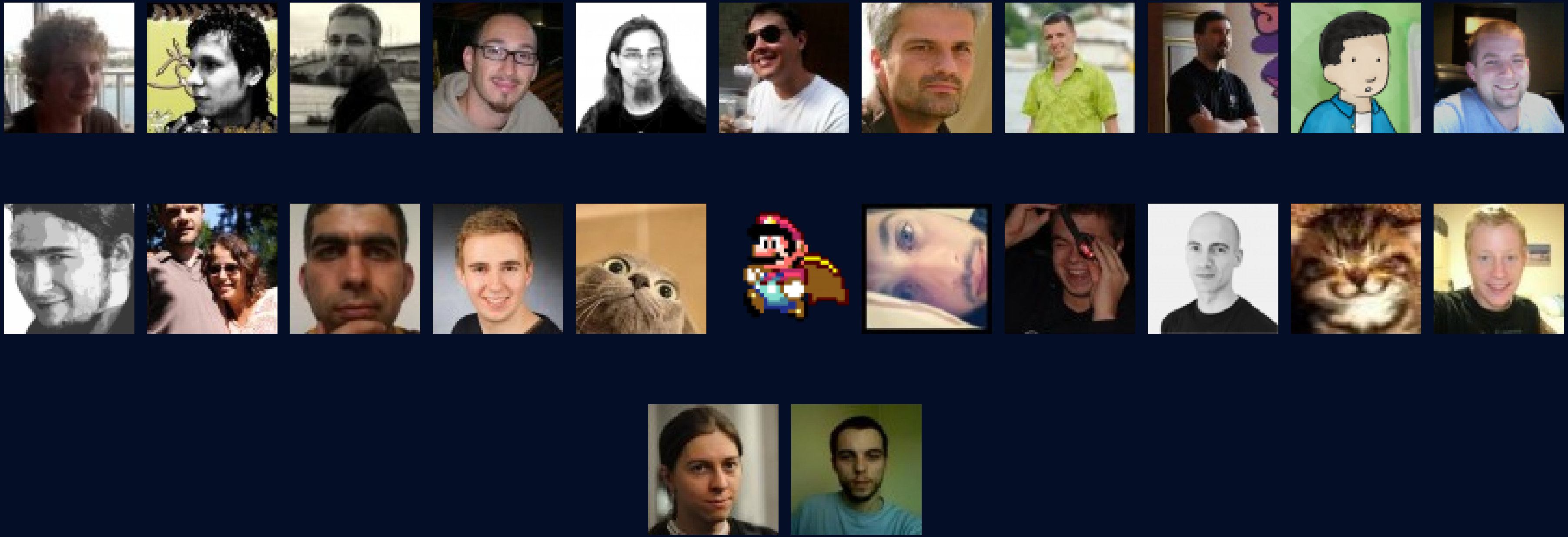
@Ocramius - Ocramius

MAIN LIBRARIES

BjyAuthorize, AssetManager, ZeffMu, ZfrRest, OcraDiCompiler, OcraServiceManager, OcraCachedViewResolver, DoctrineModule, DoctrineORMModule, DoctrineMongoODMModule, VersionEyeModule

DOCTRINE PROJECT

An incubator for persistence-oriented libraries



WHAT IS DOCTRINE ORM?

Doctrine ORM is an Object Relational Mapper

It is inspired by Hibernate and the JPA (JSR-317)

It is based on a DBAL (DataBase Abstraction Layer)

Allows developers to save and load POPO with SQL

An ORM gives you the impression that you are working with a "virtual" database (graph) composed by objects

Simple put:

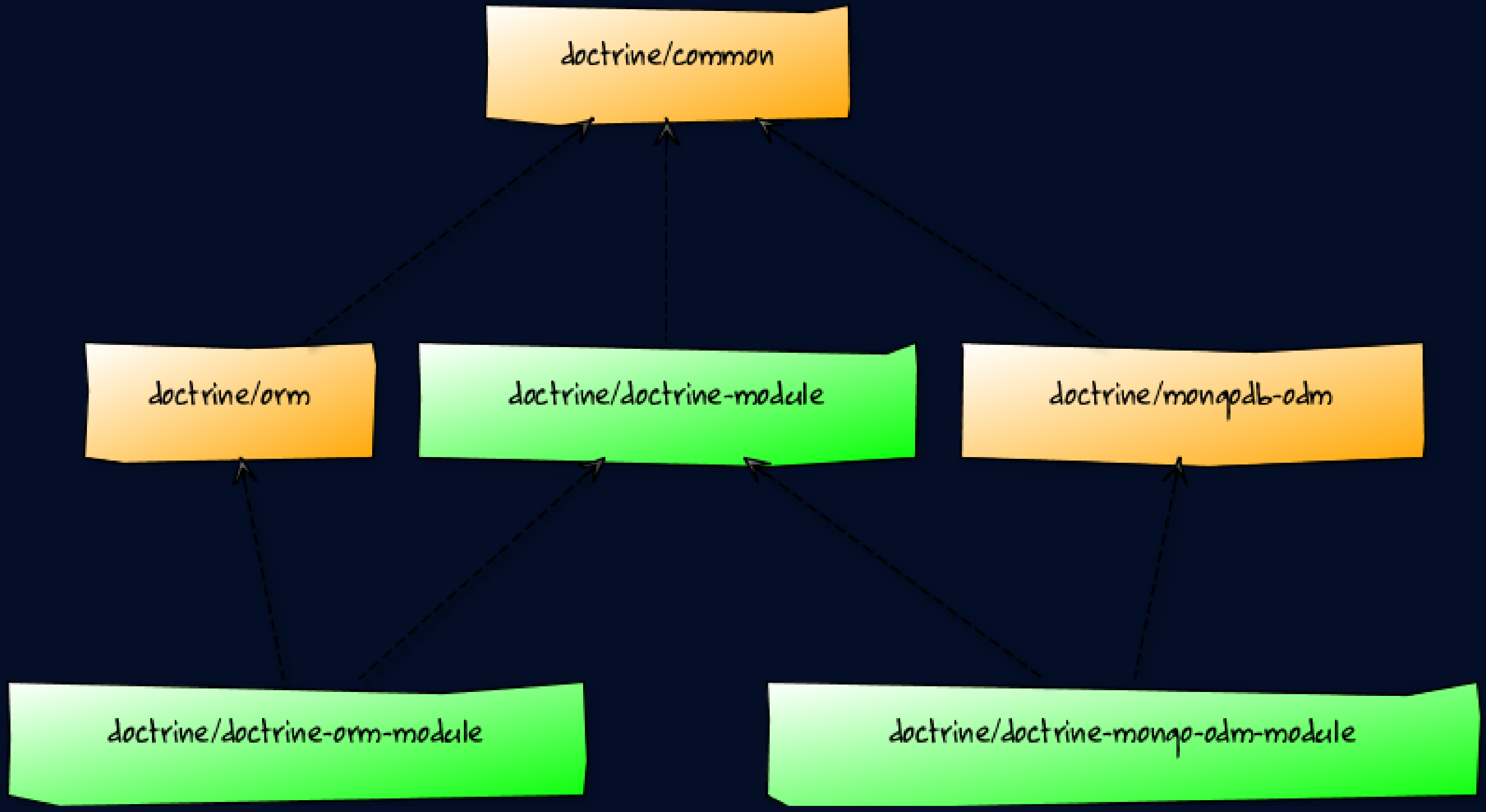
FORGET THE DATABASE!

THE MODULES!

DoctrineModule
basic common functionality

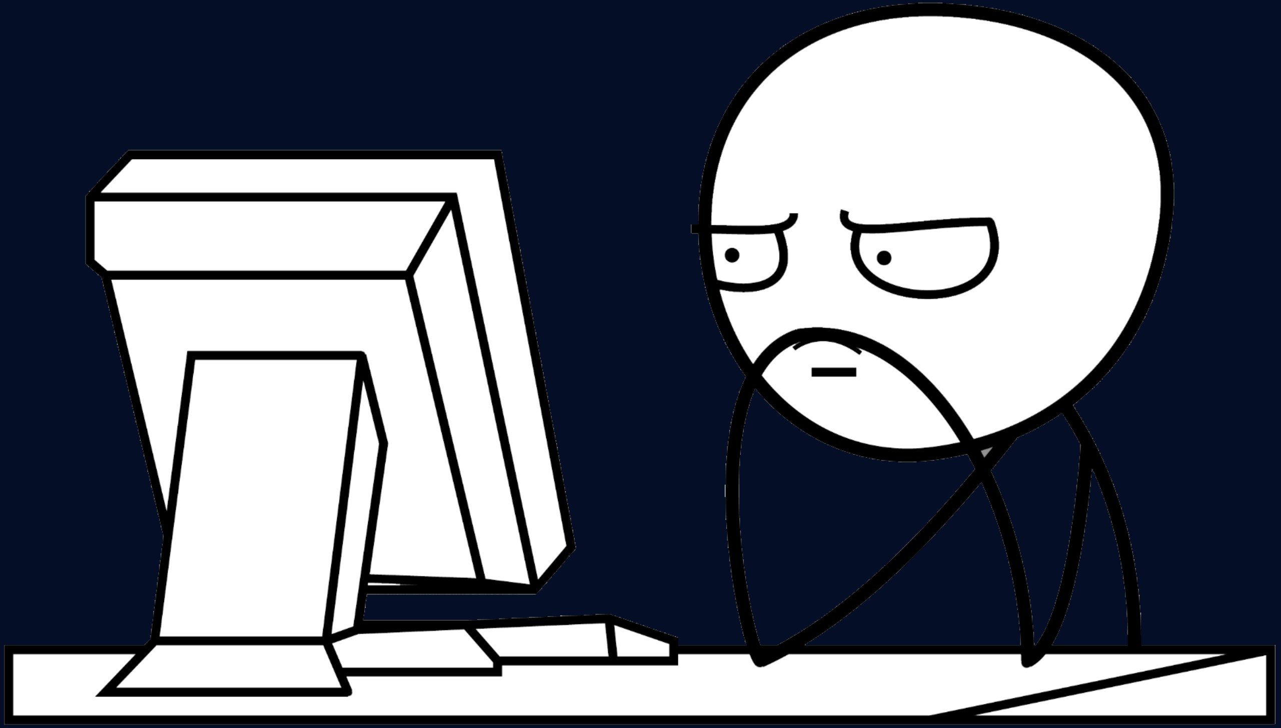
DoctrineORMModule
ORM/SQL Connection

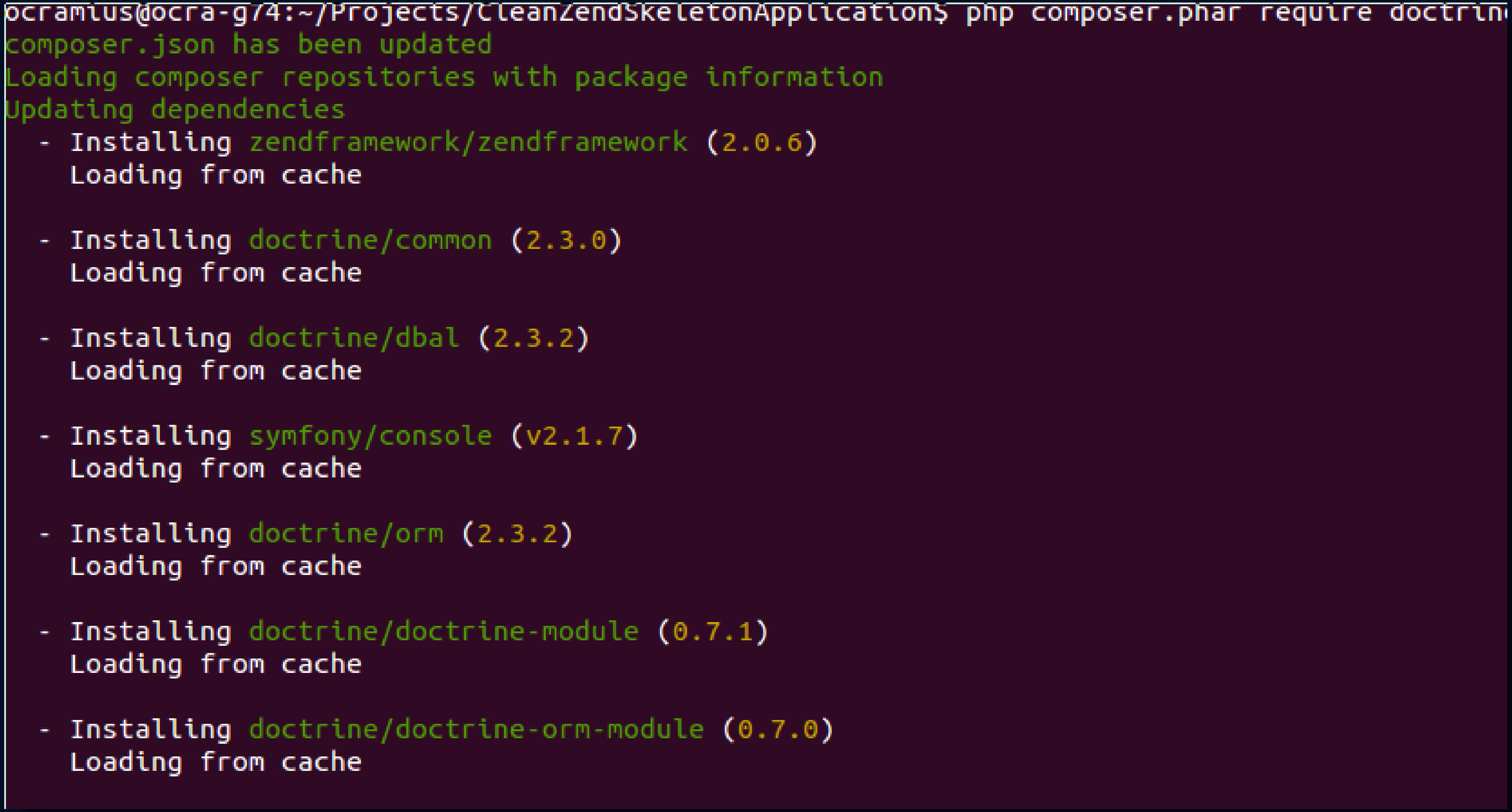
DoctrineMongoODMModule
ODM/MongoDB Connection


doctrine/common
doctrine/orm
doctrine/doctrine-module
doctrine/mongodb-odm
doctrine/doctrine-orm-module
doctrine/doctrine-mongo-odm-module

INSTALLATION!

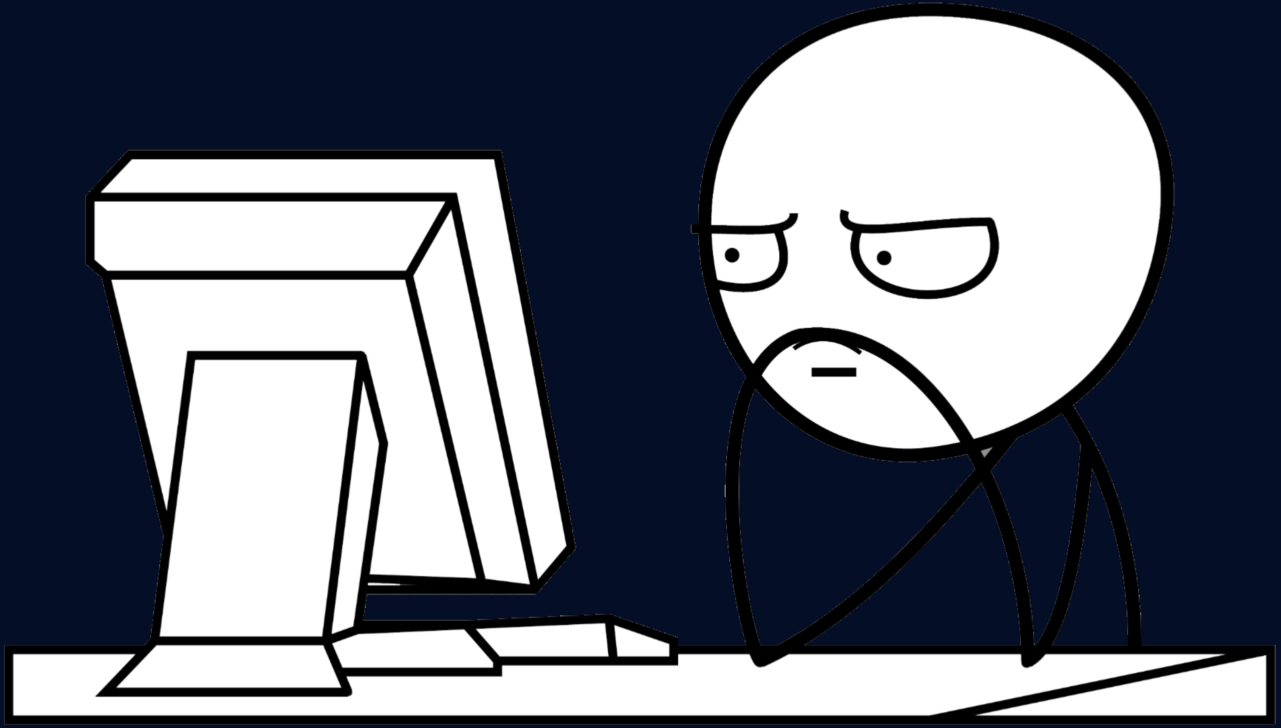
```
php composer.phar require doctrine/doctrine-orm-module:0.7.*
```

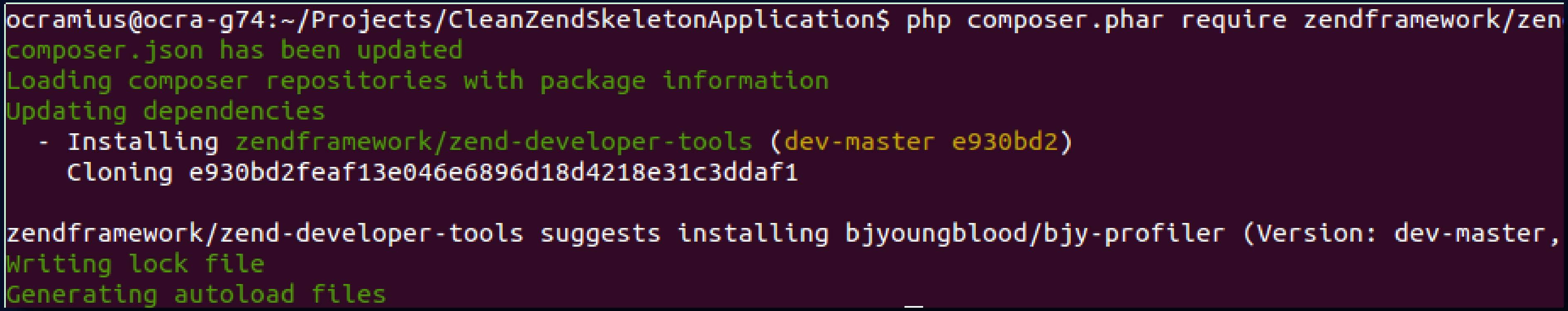




```
ocramius@ocra-g74:~/Projects/CleanZendSkeletonApplication$ php composer.phar require doctrin
composer.json has been updated
Loading composer repositories with package information
Updating dependencies
  - Installing zendframework/zendframework (2.0.6)
    Loading from cache

  - Installing doctrine/common (2.3.0)
    Loading from cache

  - Installing doctrine/dbal (2.3.2)
    Loading from cache

  - Installing symfony/console (v2.1.7)
    Loading from cache

  - Installing doctrine/orm (2.3.2)
    Loading from cache

  - Installing doctrine/doctrine-module (0.7.1)
    Loading from cache

  - Installing doctrine/doctrine-orm-module (0.7.0)
    Loading from cache
```

```
php composer.phar require zendframework/zend-developer-tools:dev-m
aster
```




```
ocramius@ocra-g74:~/Projects/CleanZendSkeletonApplication$ php composer.phar require zendframework/zen
composer.json has been updated
Loading composer repositories with package information
Updating dependencies
  - Installing zendframework/zend-developer-tools (dev-master e930bd2)
    Cloning e930bd2feaf13e046e6896d18d4218e31c3ddaf1

zendframework/zend-developer-tools suggests installing bjyoungblood/bjy-profiler (Version: dev-master,
Writing lock file
Generating autoload files
```

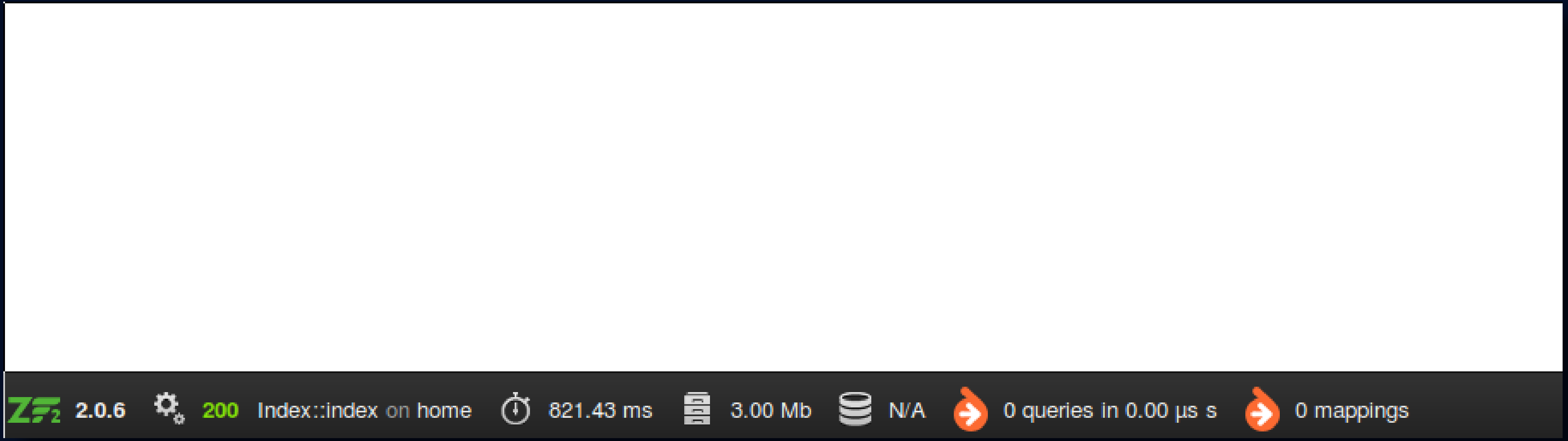
```
cp vendor/zendframework/zend-developer-tools/config/zenddevelopert
ools.local.php.dist config/autoload/zdt.local.php
```

ENABLING THE MODULES

```
config/application.config.php
```

```php
return array(
    'modules' => array(
        'ZendDeveloperTools',
        'Application',
        'DoctrineModule',
        'DoctrineORMModule',
    ),
    // [...]
);
```

You should see:



```
2.0.6   200   Index::index on home   821.43 ms   3.00 Mb   N/A   0 queries in 0.00 µs s   0 mappings
```

WRITE YOUR FIRST ENTITY

```
module/Application/src/Application/Entity/User
```

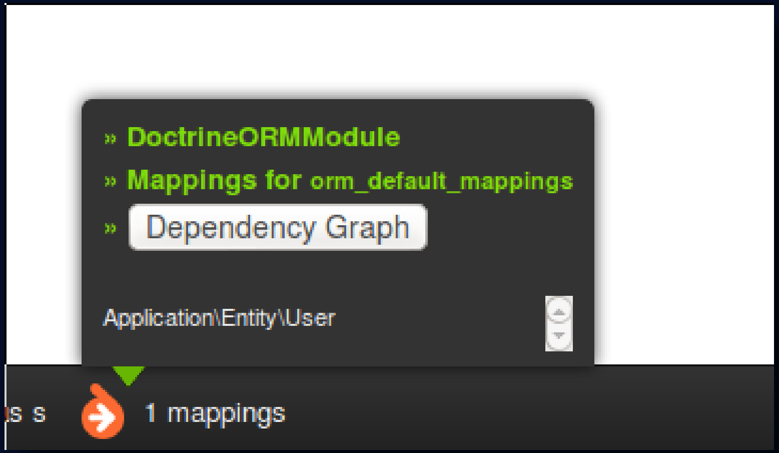
```
namespace Application\Entity;
use Doctrine\ORM\Mapping as ORM;
/** @ORM\Entity */
class User {
    /**
    * @ORM\Id
    * @ORM\GeneratedValue(strategy="AUTO")
    * @ORM\Column(type="integer")
    */
    protected $id;

    /** @ORM\Column(type="string") */
    protected $fullName;

    // getters/setters
}
```

CONFIGURE MAPPINGS

```
module/Application/config/module.config.php
```

```
return array(
'doctrine' => array(
  'driver' => array(
    'application_entities' => array(
      'class' =>'Doctrine\ORM\Mapping\Driver\AnnotationDriver'
      'cache' => 'array'
      'paths' => array(__DIR__ . '/../src/Application/Entity')
    ),

    'orm_default' => array(
      'drivers' => array(
        'Application\Entity' => 'application_entities'
      )
))), // [...]
```

You should see:



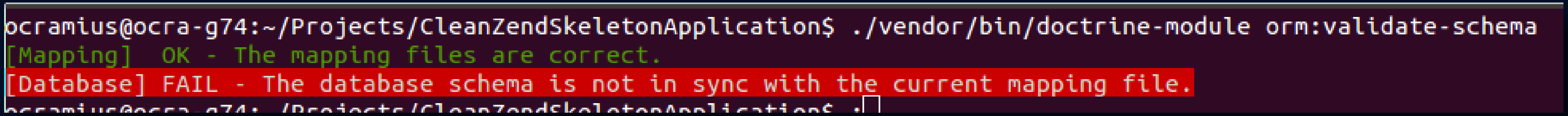
CONFIGURE THE CONNECTION

```
config/autoload/doctrine.local.php
```

```
return array(
  'doctrine' => array(
    'connection' => array(
      'orm_default' => array(
        'driverClass' =>'Doctrine\DBAL\Driver\PDOMySql\Driver',
        'params' => array(
          'host'     => 'localhost',
          'port'     => '3306',
          'user'     => 'username',
          'password' => 'password',
          'dbname'   => 'database',
)))));
```

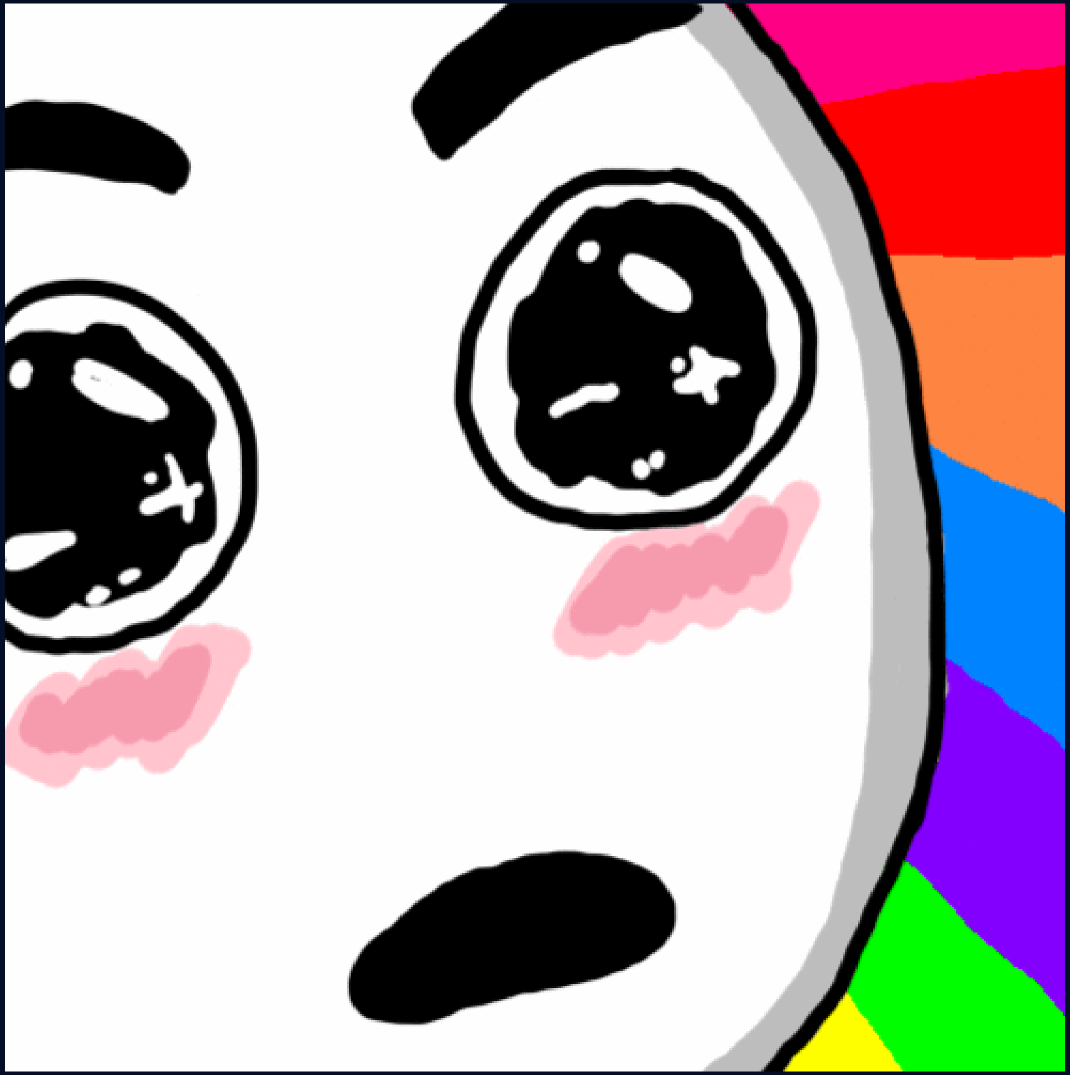
VALIDATE MAPPINGS

```
./vendor/bin/doctrine-module orm:validate-schema
```

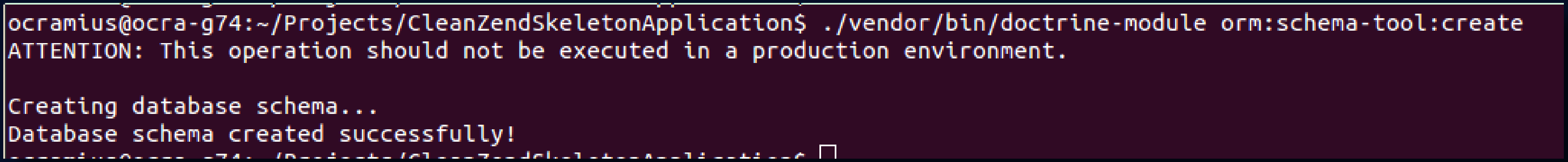


```
ocramius@ocra-g74:~/Projects/CleanZendSkeletonApplication$ ./vendor/bin/doctrine-module orm:validate-schema
[Mapping]  OK - The mapping files are correct.
[Database] FAIL - The database schema is not in sync with the current mapping file.
ocramius@ocra-g74:~/Projects/CleanZendSkeletonApplication$ 
```

GENERATE THE DATABASE



```
./vendor/bin/doctrine-module orm:schema-tool:create
```



```
ocramius@ocra-g74:~/Projects/CleanZendSkeletonApplication$ ./vendor/bin/doctrine-module orm:schema-tool:create
ATTENTION: This operation should not be executed in a production environment.

Creating database schema...
Database schema created successfully!
```

TEST IT!

```
module/Application/src/Application/Controller/IndexController.php
```

```php
public function indexAction() {
    $objectManager = $this
        ->getServiceLocator()
        ->get('Doctrine\ORM\EntityManager');

    $user = new \Application\Entity\User();
    $user->setFullName('Marco Pivetta');

    $objectManager->persist($user);
    $objectManager->flush();

    die(var_dump($user->getId())); // yes, I'm lazy
}
```

EXAMPLES

PERSISTING AN OBJECT

```
$user = new User();
$user->setFullName('Marco Pivetta');

$objectManager->persist($user); // $user1 is now "managed"
$objectManager->flush(); // commit changes to db

var_dump($user1->getId()); // 1
```

PERSISTING MULTIPLE OBJECTS

```php
$user1 = new User();
$user1->setFullName('Marco Pivetta');
$objectManager->persist($user1);

$user2 = new User();
$user2->setFullName('Michaël Gallego');
$objectManager->persist($user2);

$user3 = new User();
$user3->setFullName('Kyle Spraggs');
$objectManager->persist($user3);

$objectManager->flush();
```

RETRIEVING AN OBJECT

```
$user1 = $objectManager->find('User', 1);

var_dump($user1->getFullName()); // Marco Pivetta

$user2 = $objectManager
    ->getRepository('User')
    ->findOneBy(array('fullName' => 'Michaël Gallego'));

var_dump($user2->getFullName()); // Michaël Gallego
```

UPDATING AN OBJECT

```
$user = $objectManager->find('User', 1);

$user->setFullName('Guilherme Blanco');

$objectManager->flush();
```

DELETING AN OBJECT

```
$user = $objectManager->find('User', 1);

$objectManager->remove($user);

$objectManager->flush();
```

ASSOCIATIONS - USER

```
/** @ORM\Entity */
class User {
    // like before

    /** @ORM\ManyToOne(targetEntity="Address") */
    protected $address;

    /** @ORM\ManyToMany(targetEntity="Projects") */
    protected $projects;

    public function __construct()
    {
        $this->projects = new ArrayCollection();
    }

    // getters/setters
}
```

ASSOCIATIONS - ADDRESS

```
/** @ORM\Entity */
class Address {
    /* @ORM\Id @ORM\Column(type="integer") @ORM\GeneratedValue(str
ategy="AUTO") */
    protected $id;

    /** @ORM\Column(type="string") */
    protected $city;

    /** @ORM\Column(type="string") */
    protected $country;

    // getters/setters etc.
}
```

ASSOCIATIONS - PROJECTS

```
/** @ORM\Entity */
class Project {
    /* @ORM\Id @ORM\Column(type="integer") @ORM\GeneratedValue(str
ategy="AUTO") */
    protected $id;

    /** @ORM\Column(type="string") */
    protected $name;

    // getters/setters
}
```

ASSOCIATIONS - PERSISTING ASSOCIATIONS

```php
$user = new User();
$user->setFullName('Marco Pivetta');
$objectManager->persist($user);

$address = new Address();
$address->setCity('Frankfurt')
$address->setCountry('Germany');
$objectManager->persist($address);

$project = new Project();
$project->setName('Doctrine ORM');
$objectManager->persist($project);

$user->setAddress($address);
$user->getProjects()->add($project);
$objectManager->flush();
```

ASSOCIATIONS - RETRIEVING ASSOCIATIONS

```php
$user = $objectManager->find('User', 1);

var_dump($user->getAddress()->getCity()); // Frankfurt
var_dump($user->getProjects()->first()->getName()) // Doctrine ORM
```

More tutorials at

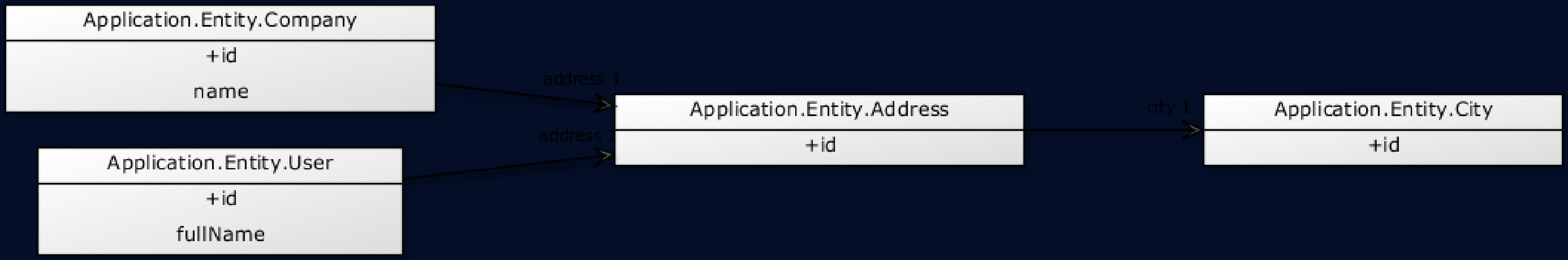
http://marco-pivetta.com/doctrine2-orm-tutorial

YO DAWG, I HEARD YOU LIKE PRESENTATIONS
SO I'VE PUT A PRESENTATION IN A PRESENTATION
memegenerator.net

DOCTRINEMODULE GOODIES

EER UML MODEL

See what your entities look like in a graph:



PAGINATOR ADAPTER

```php
use Doctrine\Common\Collections\ArrayCollection;
use DoctrineModule\Paginator\Adapter\Collection as Adapter;
use Zend\Paginator\Paginator;

// Create a Doctrine Collection
$collection = new ArrayCollection(range(1, 101));

// Create the paginator itself
$paginator = new Paginator(new Adapter($collection));

$paginator
    ->setCurrentPageNumber(1)
    ->setItemCountPerPage(5);
```

PAGINATOR ADAPTER (ORM)

```php
use DoctrineORMModule\Paginator\Adapter\DoctrinePaginator;
use Doctrine\ORM\Tools\Pagination\Paginator as ORMPaginator;
use Zend\Paginator\Paginator;

// Create a Doctrine Collection
$query = $em->createQuery('SELECT f FROM Foo f JOIN f.bar b');

// Create the paginator itself
$paginator = new Paginator(
    new DoctrinePaginator(new ORMPaginator($query))
);

$paginator
    ->setCurrentPageNumber(1)
    ->setItemCountPerPage(5);
```

OBJECT-EXISTS VALIDATOR

```
$repository = $objectManager
    ->getRepository('Application\Entity\User');

$validator = new \DoctrineModule\Validator\ObjectExists(array(
    'object_repository' => $repository,
    'fields' => array('email')
));

var_dump($validator->isValid('test@example.com'));
var_dump($validator->isValid(array(
    'email' => 'test@example.com'
)));
```

CACHE ADAPTERS

```php
$zendCache = new \Zend\Cache\Storage\Adapter\Memory();

$cache = new \DoctrineModule\Cache\ZendStorageCache($zendCache);
```

```php
$doctrineCache = new \Doctrine\Common\Cache\ArrayCache();
$options = new \Zend\Cache\Storage\Adapter\AdapterOptions();

$cache = new \DoctrineModule\Cache\DoctrineCacheStorage(
    $options,
    $doctrineCache
);
```

HYDRATOR

```php
use DoctrineModule\Stdlib\Hydrator\DoctrineObject;

$hydrator = new DoctrineObject(
    $objectManager,
    'Application\Entity\City'
);

$city = new City();
$data = array('name' => 'Frankfurt');

$city = $hydrator->hydrate($data, $city);

echo $city->getName(); // prints "Frankfurt"

$dataArray = $hydrator->extract($city);
echo $dataArray['name']; // prints "Frankfurt"
```

HYDRATOR (2)

```php
use DoctrineModule\Stdlib\Hydrator\DoctrineObject;

$hydrator = new DoctrineObject(
    $objectManager,
    'Application\Entity\City'
);

$city = new City();
$data = array('country' => 123);

$city = $hydrator->hydrate($data, $city);

var_dump($city->getCountry());
// prints class Country#1 (1) {
//   protected $name => string(5) "Germany"
// }
```

FORM ELEMENT

```
$form->add(array(
    'type' => 'DoctrineModule\Form\Element\ObjectSelect',
    'name' => 'user',
    'options' => array(
        'object_manager' => $objectManager,
        'target_class'   => 'Module\Entity\User',
        'property'       => 'fullName',
        'is_method'      => true,
        'find_method'    => array(
            'name'   => 'findBy',
            'params' => array(
                'criteria' => array('active' => 1),
                'orderBy'  => array('lasName' => 'ASC'),
            ),
        ),
    ),
));
```

MORE STUFF!

Everything works with MongoDB ODM too!

CouchDB ODM/PHPCR ODM/OrientDB ODM

GOOD PRACTICES

KEEP ENTITIES SIMPLE

Think of entities as value-objects

Don't add logic to entities (hard to change later!)

Keep entities aware only of themselves + relations

USE DOCTRINE/COMMON API

If you stick with using only doctrine/common API, users of your modules can switch between ORM/ MongoDB ODM/CouchDB ODM/ PHPCR ODM/OrientDB ODM

USE DOCTRINE/COMMON API

Prefer

```
Doctrine\Common\Persistence\ObjectManager
```

over

```
Doctrine\ORM\EntityManager
```

USE DOCTRINE/COMMON API

Prefer

```
Doctrine\Common\Persistence\ObjectRepository
```

over

```
Doctrine\ORM\EntityRepository
```

USE COLLECTIONS EXTENSIVELY

Doctrine comes with a powerful collections API

OOP API for array-like data structures

USE THE CRITERIA API

Collections provide a Criteria API

Allows you to filter virtually any kind of data structure

CRITERIA API EXAMPLE

```
use Doctrine\Common\Collections\Criteria;
use Doctrine\Common\Collections\ArrayCollection;

$collection = new ArrayCollection(array($user1, $user2, $user3));
$criteria   = new Criteria();
$criteria->andWhere(
    $criteria->expr()->gt(
        'lastLogin',
        new \DateTime('-1 day')
    )
);

$recentVisitors = $collection->matching($criteria);
```

```
$recentVisitors = $em
    ->getRepository('Application\Entity\Users')
    ->matching($criteria);
```

CRITERIA API ADVANTAGES

Works in ORM Repositories, Collections, etc...

Abstracts the problem of "searching"

Same criteria for different storages (ORM, ODM, Memory, ElasticSearch, cache...)

Allows you to define your own `RecentUsersCriteria` or `InactiveUsersCriteria`...

INJECT THE OBJECT MANAGER

If you fetch the entity manager from within your services, replacing it will become very hard: Inject it instead!

INJECT THE OBJECT MANAGER

```
'factories' => array(
    'my_service' => function ($sl) {
        $objectManager = $sl->get('Doctrine\ORM\EntityManager');
        return new MyService($objectManager);
    }
),
```

```
class MyService
{
    public function __construct(ObjectManager $objectManager)
    {
        // [...]
    }
}
```

KEEP OBJECT MANAGER OUT OF CONTROLLERS

IF YOU INJECT OBJECTMANAGER IN YOUR CONTROLLERS
INSTRUCTOR
YOU'RE GONNA HAVE A BAD ARCHITECTURE
memegenerator.net

DON’T USE PERSISTENCE TO SOLVE APPLICATION PROBLEMS

Filtering data when saved to DB

Validating data when saved to DB

Saving files when records are saved to DB

Using DB-level errors to check input validity

KEEP YOUR OBJECT GRAPH CONSISTENT

An ObjectManager works under the assumption that managed objects are valid!

Assign values to your entities only when data is valid!

QUESTIONS?

FORK IT!

You can find these slides on GitHub at https://github.com/Ocramius/doctrine-orm-zf2-tutorial

THANKS FOR WATCHING!