Doctrine

The Goal

Kill the magic

Doctrine Packages

- Common
  - Shared code
  - Cache drivers
  - Annotation parser
- DBAL abstraction on top of PDO
  - Mysql, pgsql, etc.
- ORM

What is ORM for

- Object Oriented first
- Transactions
- DDD
- Fast prototyping

What is ORM not for

- Dynamic data structures
  - like ORO entity Extend
- Reporting
  - Raw SQL

Entity

Lightweight persistent domain object

- Define database first
- Then mappings
- At ORO we use Anemic models
  - no behaviour at entities

Lifecycle Callbacks

- Don’t put business logic to it, but database related behaviour
- Use Entity Listeners instead of Doctrine Event listeners

ORM Structure

- Entity Manager - central access point, covers most of cases, internally all covered with transactions
- UnitOfWork
- Metadata Drivers
- DQL
- Repositories
- Hydrators

DBAL Data API

Connection interface, Connection

- query (returns result)
- exec (returns number of affected rows)
- beginTransaction, commit, rollback
- fetchAll, fetchAssoc, fetchArray, fetchColumn
- executeUpdate
- delete

Metadata formats

- Annotations
- Yaml
- Xml

Inheritance

- Mapped Superclass
    - Single Table
    - Class Table

Prefer aggregation over inheritance

Example:

- BaseProductPrice mapped superclass
- ProductPrice
- CombinedProductPrice

Query API & Query Builder API

- No difference from performance perspective
- DQL Query is more readable
- Query builder easier to use for dynamic queries

Repositories

- One mapped to Entity, but you can create custom not mapped
- Organize queries for reuse
- Avoid defining as a service
    - use Managers instead

Native Query & ResultSetMapping

- Execute raw SQL
- Hydrate to entities if needed
  - Better to use arrays when possible

Batch operations

- Flush every N entries
  - Flush is a transaction, it takes time
  - N should not be too small or too big
  - Test or Profile to know

Cache Drivers

Doesn’t depend to doctrine

Used at ORO everywhere

- fetch
- contains
- save
- delete

Manual invalidation

Specific drivers contains more effective methods

Cache Drivers

- Filesystem
  - default
- Redis
  - fast
  - easy to scale
  - used at OroCloud
- Memcache
  - fast
  - hard to scale
- Cache without price

Result cache

```php
# Use
return $qb->getQuery()
    ->useResultCache(true, 3600, 'some_unique_prefix')
    ->getResult();

# Invalidate
$cache = $this->getEntityManager()
    ->getConfiguration()
    ->getResultCacheImpl();

if ($cache) {
    $cache->delete('some_unique_prefix');
}
```

Oro Doctrine Extension

Mysql & Postgres support. Repository link

Functions

Date & Time

- date
- time
- timestamp
- convert_tz

String

- md5
- group_concat
- concat_ws
- cast
- replace
- date_format

Numeric

- timestampdiff
- dayofyear
- dayofmonth
- dayofweek
- week
- day
- hour
- minute
- month
- quarter
- second
- year
- sign
- pow
- round

Types

- Extra Types
- MoneyType
- PercentType
- ObjectType
- ArrayType

CLI

Schema and cache configuration validation

- doctrine:ensure-production-settings
    - Verify that Doctrine is properly configured for a production environment

Mapping Validation

- doctrine:mapping:info | grep ProductPrice

Data

- doctrine:fixtures:load

Database

- doctrine:database:create
- doctrine:database:drop

CLI

Should not Be used with ORO

- doctrine:schema:*

Hot Fixes during development

- doctrine:query:dql
- doctrine:query:sql
- doctrine:database:import

Hot Fixes on Cache, can be useful during development

- doctrine:cache:*

Toolbar

- Know what executed
- Query optimizations
- Similar & Duplicated queries
- Know Hydration price

>> Demonstration

Notice

- Avoid composite primary keys
    - Use just unique indices + incremental PK instead
- Avoid eager loading
- Don’t inject Repository, inject @doctrine instead
- Avoid inheritance, use aggregation instead