élao
Agence Web - iPhone & iPad Apps - Experts Symfony

Doctrine 2 Sheets

Symfony Doctrine Commands

ensure-prodution-settings	*Verify Doctrine for production*
data:load	*Load data fixtures to your database*
database:create	*Create the configured databases*
database:drop	*Drop the configured databases*
generate:entities	*Generate entity classes and method stubs*
generate:entity	*Generate a new entity in a bundle*
generate:proxies	*Generates proxy classes*
generate:repositories	*Generate repository classes*
mapping:import	*Import mapping info from database*
mapping:info	*Show basic info about mapped entities*
schema:create	*Processes the schema and create it or generate the SQL*
schema:drop	*Drop the database schema or generate the SQL*
schema:update	*Update the database schema or generate the SQL*

Symfony Doctrine DBAL Configuration

```
doctrine:
    dbal:
        default_connection:   default
        connections:
            default:
                dbname:               database
                host:                 localhost
                port:                 1234
                user:                 user
                password:             secret
                driver:               pdo_mysql
                driver_class:         MyNamespace\MyDriverImpl
                options:
                    foo: bar
                path:                 %kernel.data_dir%/data.sqlite
                memory:               true
                unix_socket:          /tmp/mysql.sock
                wrapper_class:        MyDoctrineDbalConnectionWrapper
                charset:              UTF-8
                logging:              %kernel.debug%
                platform_service:     MyOwnDatabasePlatformService
            conn1:
                # ...
        types:
```

Symfony Doctrine ORM Configuration

```
doctrine:
   orm:
      auto_generate_proxy_classes:    true
      proxy_namespace:                Proxies
      proxy_dir:                %kernel.cache_dir%/doctrine/orm/Proxies
      default_entity_manager:         default # Required
      entity_managers:          # At least one has to be defined
         default:
            # The name of a DBAL connection (default used if not set)
            connection:                   conn1
            mappings: # Required
               HelloBundle: ~
            class_metadata_factory_name: Doctrine\ORM\Mapping\ClassMetadataFactory
            # Cache drivers have to be array, apc, xcache or memcache
            metadata_cache_driver:          array
            query_cache_driver:             array
            result_cache_driver:
               type:                        memcache
               host:                        localhost
               port:                        11211
               instance_class:              Memcache
               class:             Doctrine\Common\Cache\MemcacheCache
         em2:
            # ...
```

Entity Repository (Method Doctrine\ORM\EntityRepository)

```
find($id)                                                    @return Entity
$em->getRepository('MyProject\Domain\User')->find($id);

findAll()                                                    @return  Array
$em->getRepository('MyProject\Domain\User')->findAll();

findOneBy(array $criteria)                                   @return Entity
$em->getRepository('MyProject\Domain\User')->findOneBy(array('nickname'=>
'dude'));

findBy(array $criteria)                                      @return  Array
$em->getRepository('MyProject\Domain\User')->findBy(array('age' => 20));
```

Doctrine Query Langage

DQL FUNCTIONS

- *ABS(arithmetic_expression)*
- *CONCAT(str1, str2)*
- *CURRENT_DATE()*
- *CURRENT_TIME()*
- *CURRENT_TIMESTAMP()*
- *LENGTH(str)*
- *LOCATE(needle, haystack [, offset])*
- *LOWER(str)*
- *MOD(a, b)*
- *SIZE(collection)*
- *SQRT(q)*
- *SUBSTRING(str, start [, length])*
- *TRIM(str)*

AGGREGATE FUNCTIONS

- *AVG*
- *COUNT*
- *MIN*
- *MAX*
- *SUM*

OTHERS EXPRESSION

- *ALL / ANY / SOME*
- *BETWEEN a AND b*
- *NOT BETWEEN a AND b*
- *IN(x1, x2, ...) / NOT IN (x1, x2, ...)*
- *LIKE a / NOT LIKE a*
- *IS NULL / IS NOT NULL*
- *EXISTS / NOT EXISTS*
- *INSTANCE OF*

HYDRATION MODES

- *Query::HYDRATE_OBJECT* or *getSingleResult() / getResult()*
- *Query::HYDRATE_ARRAY* or *getArrayResult()*
- *Query::H YDRATE_SCALAR* or *getScalarResult()*
- *Query::HYDRATE_SINGLE_SCALAR* or *getSingleScalarResult()*

Examples

```
$em ->createQuery($query)
    ->setParameter(':p1','xx')
    ->setParameters(array(':p1' => 'xx', ':p2' => 'xy')
    ->getResult(HYDRATE_MODE);
```

```
SELECT u FROM MyProject\Model\User u
SELECT DISTINCT u.id FROM Article a JOIN a.user u
SELECT a FROM Article a JOIN a.user u ORDER BY u.name ASC
SELECT u.username, u.name FROM User u
SELECT u, a FROM ForumUser u JOIN u.avatar a
SELECT u, p FROM User u JOIN u.phonenumbers p
SELECT u FROM ForumUser u ORDER BY u.id ASC
SELECT COUNT(u.id) FROM Entities\User u
SELECT u FROM ForumUser u WHERE u.id = ?1
SELECT u FROM ForumUser u WHERE (u.name = :name OR u.name = :name2) AND
u.id = :id
SELECT u FROM User u WHERE ((u.id + 5000) * u.id + 3) < 10000000
SELECT u.id, a.id as article_id FROM User u LEFT JOIN u.articles a
SELECT u, a, p, c FROM User u JOIN u.articles a JOIN u.phonenumbers p JOIN
a.comments c
SELECT u.name FROM User u WHERE u.id BETWEEN ?1 AND ?2
SELECT u FROM User u WHERE u.id IN (1, 2)
SELECT CONCAT(u.id, u.name) FROM User u WHERE u.id = ?1
SELECT u FROM User u WHERE EXISTS (SELECT p.phone FROM Phone p WHERE p.user =
u.id)
SELECT u.id FROM User u WHERE :groupId MEMBER OF u.groups
SELECT u FROM User u WHERE SIZE(u.phonenumbers) > 1
SELECT u FROM User u WHERE u.phonenumbers IS EMPTY
SELECT u FROM Model\CompanyPerson u WHERE u INSTANCE OF Model\CompanyEmployee
```

élao

Agence Web - iPhone & iPad Apps - Experts Symfony

Doctrine 2 Sheets

Classes annotations / Main

@Entity
- *repositoryClass*

@HasLifecycleCallbacks

@Table
- *name*
- *indexes*

Properties annotations / Main

@Column

type	string, integer, smallint, bigint, boolean, decimal, date, time, datetime, text, object, array, float
name	column name *(string)*
length	column length *(integer)*
unique	unique key *(true or false)*
nullable	column can be null *(true or false)*

@Id

@Index

name	index name *(string)*
columns	related columns *(strings array)*

@GeneratedValue *with @Id*

strategy	AUTO, SEQUENCE, TABLE, IDENTITY, NONE

Properties annotations / Associations

@OneToOne

targetEntity	FQCN of the referenced target entity *(string)*
inversedBy	field in the entity that is the inverse side *(string)*
cascade	*(persist, remove, merge, detach, all)*
fetch	fetch type *(LAZY or EAGER)*
orphanRemoval	remove orphan *(true or false)*

@OneToMany

targetEntity	FQCN of the target entity *(string)*
cascade	*(persist, remove, merge, detach, all)*
orphanRemoval	remove orphan *(true or false)*
mappedBy	property on the targetEntity that is the owning side *(string)*

@ManyToOne

targetEntity	FQCN of the target entity *(string)*
cascade	*(persist remove, merge, detach, all)*
fetch	fetch type *(LAZY or EAGER)*

@ManyToMany

targetEntity	FQCN of the target entity *(string)*
mappedBy	property on the targetEntity that is the owning side *(string)*
inversedBy	field in the entity that is the inverse side *(string)*
cascade	*(persist, remove, merge, detach, all)*
fetch	fetch type *(LAZY or EAGER)*

@JoinTable *with @OneToMany or @ManyToMany*

name	Database name of the join-table
joinColumns	An array of @JoinColumn
inverseJoinColumns	An array of @JoinColumn

@JoinColumn *with @ManyToOne or @OneToOne*

name	Column name that holds the foreign key
referencedColumnName	Name of the primary key identifier used for join
unique	is this relation exclusive between the affected entities
nullable	related entity is required
onDelete	Cascade Action (Database-level)
onUpdate	Cascade Action (Database-level)

@OrderBy

Class annotations / Inheritance

@DiscriminatorColumn

@DiscriminatorMap

@InheritanceType

@MapperSuperclass

Method annotations / Callbacks

require @HasLifecycleCallbacks

@PostLoad, @PostPersist, @PostRemove, @PostUpdate, @PrePersist, @PreRemove, @PreUpdate

Associations Example / One-To-One Bidirectional

```php
<?php
    /** @Entity */
    class Customer
    {
        /**
         * @OneToOne(targetEntity="Cart", mappedBy="customer")
         */
        private $cart;
    }

    /** @Entity */
    class Cart
    {
        /**
         * @OneToOne(targetEntity="Customer", inversedBy="cart")
         * @JoinColumn(name="customer_id", referencedColumnName="id")
         */
        private $customer;
    }
```

Associations Example / One-To-Many Bidirectional

```php
<?php
 /** @Entity */
 class Product
 {
    /**
    * @OneToMany(targetEntity="Feature", mappedBy="product")
    */
    private $features;

    public function __construct() {
       $this->features = new \Doctrine\Common\Collections\ArrayCollection();
    }
 }

 /** @Entity */
 class Feature
 {
    /**
    * @ManyToOne(targetEntity="Product", inversedBy="features")
    * @JoinColumn(name="product_id", referencedColumnName="id")
    */
    private $product;
 }
```

Associations Example / One-To-Many Self Referencing

```php
<?php
 /** @Entity */
 class Category
 {
    /**
    * @OneToMany(targetEntity="Category", mappedBy="parent")
    */
    private $children;

    /**
    * @ManyToOne(targetEntity="Category", inversedBy="children")
    * @JoinColumn(name="parent_id", referencedColumnName="id")
    */
    private $parent;

    public function __construct() {
       $this->children = new \Doctrine\Common\Collections\ArrayCollection();
    }
 }
```

Associations Example / Many-To-Many Bidir - Ordered

```php
<?php
 /** @Entity */
 class User
 {
    /**
    * @ManyToMany(targetEntity="Group", inversedBy="users")
    * @OrderBy({"name" = "ASC"})
    * @JoinTable(name="users_groups",
    *     joinColumns={@JoinColumn(name="user_id", referencedColumnName="id")},
    *     inverseJoinColumns={@JoinColumn(name="group_id", referencedColumnName="id")}
    * )
    */
    private $groups;

    public function __construct() {
       $this->groups = new \Doctrine\Common\Collections\ArrayCollection();
    }
 }

 /** @Entity */
 class Group
 {
    /**
    * @ManyToMany(targetEntity="User", mappedBy="groups")
    */
    private $users;

    public function __construct() {
       $this->users = new \Doctrine\Common\Collections\ArrayCollection();
    }
 }
```

Inheritance Example

```php
<?php
 namespace MyProject\Model;
 /**
  * @Entity
  * @InheritanceType("JOINED")
  * @DiscriminatorColumn(name="discr", type="string")
  * @DiscriminatorMap({"person" = "Person", "employee" = "Employee"})
  */
 class Person
 {
 }

 /** @Entity */
 class Employee extends Person
 {
 }
```