Project Versions

latest

Search

Go

Table Of Contents

Extra Lazy Associations

Enabling Extra-Lazy Associations

Previous topic

Working with Indexed Associations

Next topic

Composite and Foreign Keys as Primary Key

This Page

Show Source

Extra Lazy Associations

New in version 2.1.

In many cases associations between entities can get pretty large. Even in a simple scenario like a blog. where posts can be commented, you always have to assume that a post draws hundreds of comments. In Doctrine 2.0 if you accessed an association it would always get loaded completely into memory. This can lead to pretty serious performance problems, if your associations contain several hundreds or thousands of entities.

With Doctrine 2.1 a feature called **Extra Lazy** is introduced for associations. Associations are marked as **Lazy** by default, which means the whole collection object for an association is populated the first time its accessed. If you mark an association as extra lazy the following methods on collections can be called without triggering a full load of the collection:

- **`Collection#contains($entity)`**
- **`Collection#containsKey($key)`** (available with Doctrine 2.5)
- **`Collection#count()`**
- **`Collection#get($key)`** (available with Doctrine 2.4)
- **`Collection#slice($offset, $length = null)`**

For each of the above methods the following semantics apply:

- For each call, if the Collection is not yet loaded, issue a straight SELECT statement against the database.
- For each call, if the collection is already loaded, fallback to the default functionality for lazy collections. No additional SELECT statements are executed.

Additionally even with Doctrine 2.0 the following methods do not trigger the collection load:

- **`Collection#add($entity)`**
- **`Collection#offsetSet($key, $entity)`** - ArrayAccess with no specific key **`$coll[] =$entity`**, it does not work when setting specific keys like **`$coll[0] = $entity`**.

With extra lazy collections you can now not only add entities to large collections but also paginate them easily using a combination of **`count`** and **`slice`**.

Enabling Extra-Lazy Associations

have to switch to extra lazy as shown in these examples:

PHP XML YAML

```
<?php
namespace Doctrine\Tests\Models\CMS;

/**
 * @Entity
 */
class CmsGroup
{
    /**
     * @ManyToMany(targetEntity="CmsUser", mappedBy="groups", fetch="EXTRA_LAZY")
     */
    public $users;
}
```