What is the difference between inversedBy and mappedBy?

I am developing my application using Zend Framework 2 and Doctrine 2.

While writting annotations, I am unable to understand the difference between `mappedBy` and `inversedBy` .

When should I use `mappedBy` ?

When should I use `inversedBy` ?

When should I use neither?

Here is an example:

```
 /**
 *
 * @ORM\OneToOne(targetEntity="\custMod\Entity\Person", mappedBy="customer")
 * @ORM\JoinColumn(name="personID", referencedColumnName="id")
 */
protected $person;

/**
 *
 * @ORM\OneToOne(targetEntity="\Auth\Entity\User")
 * @ORM\JoinColumn(name="userID", referencedColumnName="id")
 */
protected $user;

/**
```

I did a quick search and found the following, but I am still confused:

- example 1
- example 2
- example 3

php doctrine2

edited May 23 '17 at 12:02
Community ♦
1 1

asked Sep 19 '12 at 11:33
Developer
13.8k 18 62 106

4 Answers

- **mappedBy** has to be specified on the *inversed side* of a (bidirectional) association
- **inversedBy** has to be specified on the *owning side* of a (bidirectional) association

from doctrine documentation:

- ManyToOne is always the owning side of a bidirectional assocation.
- OneToMany is always the inverse side of a bidirectional assocation.
- The owning side of a OneToOne assocation is the entity with the table containing the foreign key.

See https://www.doctrine-project.org/projects/doctrine-orm/en/latest/reference/unitofwork-associations.html

edited Apr 16 at 18:38
jake stayman
736 6 17

answered Sep 19 '12 at 13:32
Andreas Linden
9,834 7 38 59

1 Oddly the Doctrine documenter decided to leave out the yaml example of a many-to-one bidirectional mapping, probably the most commonly used! – Peter Wooster Jan 21 '13 at 21:46

yourself. – 11mb Dec 16 '15 at 14:20

2 @AndreasLinden widely used doesn't mean best practice. Something using comments to write code on the fly cannot ever been considered best practice, it's not php native, and even not included by default in all frameworks. Having all the info about an entity in one place is an anti-argument. Since when grouping all your code into one place is a good thing ? It is a pain to write, a pain to maintain, and reduce the organization in your project. best practice ? Duh. – JesusTheHun Nov 29 '16 at 15:26

@JesusTheHun you're comparing apples and pears. "all code" is very very different to "all info about an entity" ;) – Andreas Linden Dec 1 '16 at 13:02

The answers above were not sufficient for me to understand what was going on, so after delving into it more I think I have a way of explaining it that will make sense for people who struggled like I did to understand.

inversedBy and mappedBy are used by the **INTERNAL DOCTRINE** engine to **reduce the number of SQL queries** it has to do to get the information you need. To be clear if you don't add inversedBy or mappedBy your code will still work but will not be **optimized**.

So for example, look at the classes below:

```
class Task
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
     */
    private $task;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="dueDate", type="datetime")
     */
    private $dueDate;

    /**
     * @ORM\ManyToOne(targetEntity="Category", inversedBy="tasks", cascade=
{"persist"})
     * @ORM\JoinColumn(name="category_id", referencedColumnName="id")
     */
    protected $category;
}

class Category
{
    /**
     * @var int
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="name", type="string", length=255)
     */
    private $name;

    /**
     * @ORM\OneToMany(targetEntity="Task", mappedBy="category")
     */
    protected $tasks;
}
```

These classes if you were to run the command to generate the schema (for example, `bin/console` like I was... **Category is NOT referring TO THE CLASS NAME**, its referring to the property on the Task class called 'protected $category'.

Like wise, on the Tasks class the property $category mentions it is inversedBy="tasks", notice this is plural, **this is NOT THE PLURAL OF THE CLASS NAME**, but just because the property is called 'protected $tasks' in the Category class.

Once you understand this it becomes very easy to understand what inversedBy and mappedBy are doing and how to use them in this situation.

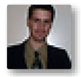
The side that is referencing the foreign key like 'tasks' in my example always gets the inversedBy attribute because it needs to know what class (via the targetEntity command) and what variable (inversedBy=) on that class to 'work backwards' so to speak and get the category information from. An easy way to remember this, is the class that would have the foreignkey_id is the one that needs to have inversedBy.

Where as with category, and its $tasks property (which is not on the table remember, just only part of the class for optimization purposes) is MappedBy 'tasks', this creates the relationship officially between the two entities so that doctrine can now safely use JOIN SQL statements instead of two separate SELECT statements. Without mappedBy, the doctrine engine would not know from the JOIN statement it will create what variable in the class 'Task' to put the category information.

Hope this explains it a bit better.

edited Jan 4 '16 at 0:55

answered Jan 4 '16 at 0:48


Joseph Astrahan
2,727 28 68

4 This is very well explained and thanks for your effort. I came from Laravel Eloquent to doctrine and this was hard for me to understand the logic here. Well done. `Category is NOT referring TO THE CLASS NAME, its referring to the property on the Task class called 'protected $category'` all I needed. It didn't only solved my problem but it helped me to understand. The best answer IMO :-) – The Alpha Jun 19 '16 at 15:57

Thanks, took me a while to figure this out. Glad it helped :). – Joseph Astrahan Jun 21 '16 at 17:08

I too came from Eloquent, this helped me greatly. my only sticking point now is how to set the setter/getter for this, I'm still learning the ropes on it – Eman May 25 '17 at 3:01

In bidirectional relationship has both an owning side and an inverse side

mappedBy : put into The inverse side of a bidirectional relationship To refer to its owning side

inversedBy : put into The owning side of a bidirectional relationship To refer to its inverse side

AND

mappedBy attribute used with the OneToOne, OneToMany, or ManyToMany mapping declaration.

inversedBy attribute used with the OneToOne, ManyToOne, or ManyToMany mapping declaration.

Notice : The owning side of a bidirectional relationship the side that contains the foreign key.

there two reference about inversedBy and mappedBy into Doctrine Documentation : First Link,Second Link

answered Nov 20 '13 at 16:12
ahmed hamdy
3,896 1 36 42

1 +1 for the links with some examples. – Marcos Jul 24 '14 at 8:22

5.9.1. Owning and Inverse Side

For Many-To-Many associations you can chose which entity is the owning and which the inverse side. There is a very simple semantic rule to decide which side is more suitable to be the owning side from a developers perspective. You only have to ask yourself, which entity is responsible for the connection management and pick that as the owning side.

Take an example of two entities Article and Tag. Whenever you want to connect an Article to a Tag and vice-versa, it is mostly the Article that is responsible for this relation. Whenever you add a new article, you want to connect it with existing or new tags. Your create Article form will probably support this notion and answered Sep 25 '16 at 8:20

Micheal Mouner Mikhail Youssif
586 3 12