Stack Overflow is a community of 9.0 million programmers, just like you, helping each other. Join them; it only takes a minute:

Sign up

What is the “owning side” in an ORM mapping?

Ask Question

What exactly does the ***owning side*** mean? What is an explanation with some mapping examples (***one to many, one to one, many to one***)?

The following text is an excerpt from the description of ***@OneToOne*** in Java EE 6 documentation. You can see the concept ***owning side*** in it.

> Defines a single-valued association to another entity that has one-to-one multiplicity. It is not normally necessary to specify the associated target entity explicitly since it can usually be inferred from the type of the object being referenced. If the relationship is bidirectional, ***the non-owning side*** must use the mappedBy element of the OneToOne annotation to specify the relationship field or property of the owning side.

java hibernate orm jpa mapping

edited Nov 13 '17 at 18:18

Willi Mentzel

6,384 11 36 60

asked May 1 '10 at 11:11

Ogrish Man

6,763 33 112 161

stackoverflow.com/questions/12493865/… – Developer May 12 '14 at 12:06

2 I was lost until I read this: javacodegeeks.com/2013/04/… – darga33 Nov 7 '16 at 9:54

The DB table with the foreign key column is treated as owning side. So the business entity representing that DB table is

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow’s Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

You can imagine that the *owning side* is the entity that has the reference to the other one. In your excerpt, you have an one-to-one relationship. Since it's a *symmetric* relation, you'll end up having that if object A is in relation with object B then also the vice-versa is true.

This means that saving into object A a reference to object B and saving in object B a reference to object A will be redundant: that's why you choose which object "owns" the other having the reference to it.

When you have got an one-to-many relationship, the objects related to the "many" part will be the owning side, otherwise you would have to store many references from a single object to a multitude. To avoid that, every object in the second class will have a pointer to the single one they refer to (so they are the owning side).

For a many-to-many relationship, since you will need a separate mapping table anyway there won't be any owning side.

In conclusion the **owning side** is the entity that has the reference to the other.

edited Jul 14 '15 at 2:24

Community ♦
1 1

answered May 1 '10 at 11:24

Jack
104k 25 175 283

6 Thank you for your clarification. – Ogrish Man May 1 '10 at 11:38

1 it might help to see bellow as well my answer for reasons for the names 'mappedBy' and 'owning side', what happens if we don't define a owning side, GOTCHAs, hope it helps – Angular University Jan 12 '14 at 22:00

3 Well, the answer is mostly correct I guess. But for Hibernate at least, even many-to-many relationships have an owning side. This has implications for the updating behavior for example. Have a close look at section 4 ("Update Hibernate Model Class") of this tutorial: viralpatel.net/blogs/… – Pfiver Mar 20 '15 at 10:25

7 This answer confuses more than it helps. What good is it to say that "You can imagine that the owning side is the entity that has the reference to the other one" when in bidirectional relationships, both Entity objects will have a reference to each other? Also, "For a many-to-many relationship, since you will need a separate mapping table anyway there won't be any owning side" is just plain incorrect: `@ManyToMany` relationships have owning sides too. Similarly, `@OneToMany` relationships can use join tables, and you still have to specify an owning side. – DavidS Jul 14 '15 at 21:46

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our , , and our . Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

Why is the notion of a owning side necessary:

The idea of a owning side of a bidirectional relation comes from the fact that in relational databases there are no bidirectional relations like in the case of objects. In databases we only have unidirectional relations - foreign keys.

What is the reason for the name 'owning side'?

The owning side of the relation tracked by Hibernate is the side of the relation that *owns* the foreign key in the database.

What is the problem that the notion of owning side solves?

Take an example of two entities mapped *without* declaring a owning side:

```
@Entity
@Table(name="PERSONS")
public class Person {
    @OneToMany
    private List<IdDocument>  idDocuments;
}

@Entity
@Table(name="ID_DOCUMENTS")
public class IdDocument {
    @ManyToOne
    private Person person;
}
```

From a OO point of view this mapping defines not one bi-directional relation, but *two* separate uni-directional relations.

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our , , and our . Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

```
  persons_id bigint NOT NULL,
  id_documents_id bigint NOT NULL,
  CONSTRAINT fk_persons FOREIGN KEY (persons_id) REFERENCES persons (id),
  CONSTRAINT fk_docs FOREIGN KEY (id_documents_id) REFERENCES id_documents (id),
  CONSTRAINT pk UNIQUE (id_documents_id)
)
```

Notice the primary key `pk` on `ID_DOCUMENTS` only. In this case Hibernate tracks both sides of the relation independently: If you add a document to relation `Person.idDocuments`, it inserts a record in the association table `PERSON_ID_DOCUMENTS`.

On the other hand, if we call `idDocument.setPerson(person)`, we change the foreign key person_id on table `ID_DOCUMENTS`. Hibernate is creating *two* unidirectional (foreign key) relations on the database, to implement *one* bidirectional object relation.

How the notion of owning side solves the problem:

Many times what we want is only a foreign key on table `ID_DOCUMENTS` towards `PERSONS` and the extra association table.

To solve this we need to configure Hibernate to stop tracking the modifications on relation `Person.idDocuments`. Hibernate should only track the *other* side of the relation `IdDocument.person`, and to do so we add **mappedBy**:

```
@OneToMany(mappedBy="person")
private List<IdDocument>  idDocuments;
```

What does it mean mappedBy ?

> This means something like: "modifications on this side of the relation are already **Mapped By** the other side of the relation IdDocument.person, so no need to track it here separately in an extra table."

Are there any GOTCHAs, consequences?

Using **mappedBy**, If we only call `person.getDocuments().add(document)`, the foreign key in `ID_DOCUMENTS` will **NOT** be linked to the new document, because this is not the owning /tracked side of the relation!

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our , , and our . Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

edited Sep 3 '15 at 19:08

answered Jan 11 '14 at 22:22

Angular University

27.9k 10 53 74

12 The best answer i find that explains the Doctrine 'mappedBy' +'inversedBy'. – Kurt Zhong Jul 7 '14 at 6:18

3 Way better than accepted answer, thank you – Ali Dehghani Jan 10 '16 at 16:59

Thanks for specifying the mappings & the reason behind the concept. – Mohnish Apr 13 '16 at 19:06

1 I don't know if things have changed, but on Hibernate 5.0.9.Final if I " only call `person.getDocuments().add(document)` , " hibernate updates the foreign key in `ID_DOCUMENTS` . – K.Nicholas Apr 14 '16 at 13:12

Can you please also add the meaning of a "mappedBy" in a ManyToMany relationship? According to what you said, the real owner is an external table (which owns two references, one per side). So what's the point to put a "mappedBy" in one of the two sides? – Jack Jul 12 '17 at 10:26

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our , , and our . Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.