Fluent NHibernate Architecture Question

Asked 16 years, 3 months ago Modified 16 years, 3 months ago Viewed 619 times

1

I have a question that I may be over thinking at this point but here goes...

I have 2 classes Users and Groups. Users and groups have a many to many relationship and I was thinking that the join table group_users I wanted to have an IsAuthorized property (because some groups are private -- users will need authorization).

Would you recommend creating a class for the join table as well as the User and Groups table? Currently my classes look like this.

```
public class Groups
{
    public Groups()
    {
        members = new List<Person>();
    }
    ...
    public virtual IList<Person> members { get; set; }
}

public class User
{


    public User()
    {
       groups = new Groups()
    }
    ...
    public virtual IList<Groups> groups{ get; set; }

}
```

My mapping is like the following in both classes (I'm only showing the one in the users mapping but they are very similar):

```
HasManyToMany<Groups>(x => x.Groups)
.WithTableName("GroupMembers")
.WithParentKeyColumn("UserID")
.WithChildKeyColumn("GroupID")
.Cascade.SaveUpdate();
```

Should I write a class for the join table that looks like this?

```
public class GroupMembers
{
    public virtual string GroupID { get; set; }
    public virtual string PersonID { get; set; }
    public virtual bool WaitingForAccept { get; set; }
}
```

I would really like to be able to adjust the group membership status and I guess I'm trying to think of the best way to go about this.

nhibernate architecture fluent

Share Follow

edited Sep 16, 2008 at 1:35
jfs
16.7k ● 13 ● 63 ● 90

asked Sep 16, 2008 at 1:33
Ryan Lanciaux
5,976 ● 2 ● 39 ● 49

2 Answers

Sorted by: Highest score (default)

1

I generally only like to create classes that represent actual business entities. In this case I don't think 'groupmembers' represents anything of value in your code. To me the ORM should map the database to your business objects. This means that your classes don't have to exactly mirror the database layout.

Also I suspect that by implementing GroupMembers, you will end up with some nasty collections in both your user and group classes. I.E. the group class will have the list of users and also a list of groupmembers which references a user and vice versa for the user class. To me this isn't that clean and will make it harder to maintain and propagate changes to the tables.

I would suggest keeping the join table in the database as you have suggested, and add a List of groups called waitingtoaccept in users and (if it makes sense too) add List of users called waitingtoaccept in groups.

These would then pull their values from your join-table in the database based on the waitingtoaccept flag.

Share Follow

edited Sep 16, 2008 at 3:10

answered Sep 16, 2008 at 3:04
dpollock
175 ● 1 ● 4

Yes, sure you need another class like UserGroupBridge. Another good side-effect is that you can modify user membership and group members without loading potentially

1

heavy User/Group objects to NHibernate session.

Cheers.

Share Follow

answered Sep 16, 2008 at 2:53

dimarzionist

18.7k ● 4 ● 24 ● 23