NHibernate Session.Flush() Sending Update Queries When No Update Has Occurred

Asked 16 years, 3 months ago Modified 11 years, 10 months ago Viewed 11k times

35

I have an NHibernate session. In this session, I am performing exactly 1 operation, which is to run this code to get a list:

```
public IList<Customer> GetCustomerByFirstName(string customerFirstName)
{
return _session.CreateCriteria(typeof(Customer))
    .Add(new NHibernate.Expression.EqExpression("FirstName",
customerFirstName))
    .List<Customer>();
}
```

I am calling `Session.Flush()` at the end of the `HttpRequest`, and I get a `HibernateAdoException`. NHibernate is passing an update statement to the db, and causing a foreign key violation. If I don't run the `flush`, the request completes with no problem. The issue here is that I need the flush in place in case there is a change that occurs within other sessions, since this code is reused in other areas. Is there another configuration setting I might be missing?

Here's the code from the exception:

```
[SQL: UPDATE CUSTOMER SET first_name = ?, last_name = ?, strategy_code_1 = ?,
strategy_code_2 = ?, strategy_code_3 = ?, dts_import = ?, account_cycle_code =
?, bucket = ?, collector_code = ?, days_delinquent_count = ?,
external_status_code = ?, principal_balance_amount = ?, total_min_pay_due = ?,
current_balance = ?, amount_delinquent = ?, current_min_pay_due = ?, bucket_1 =
?, bucket_2 = ?, bucket_3 = ?, bucket_4 = ?, bucket_5 = ?, bucket_6 = ?,
bucket_7 = ? WHERE customer_account_id = ?]
```

No parameters are showing as being passed.

c# .net nhibernate

Share
Improve this question
Follow

edited Feb 14, 2013 at 6:25

NullUserException
85.4k ● 30 ● 211 ● 237

asked Aug 29, 2008 at 17:49

Mark Struzinski
33.5k ● 35 ● 108 ● 137

3 Answers

Sorted by: Highest score (default)

46

Always be careful with NULLable fields whenever you deal with NHibernate. If your field is NULLable in DB, make sure corresponding .NET class uses Nullable type too. Otherwise, all kinds of weird things will happen. The symptom is usually will be that NHibernate will try to update the record in DB, even though you have not changed any fields since you read the entity from the database.

The following sequence explains why this happens:

1. NHibernate retrieves raw entity's data from DB using ADO.NET
2. NHibernate constructs the entity and sets its properties
3. If DB field contained NULL the property will be set to the defaul value for its type:
    - properties of reference types will be set to null
    - properties of integer and floating point types will be set to 0
    - properties of boolean type will be set to false
    - properties of DateTime type will be set to DateTime.MinValue
    - etc.
4. Now, when transaction is committed, NHibernate compares the value of the property to the original field value it read form DB, and since the field contained NULL but the property contains a non-null value, **NHibernate considers the property dirty, and forces an update of the enity.**

Not only this hurts performance (you get extra round-trip to DB and extra update every time you retrieve the entity) but it also may cause hard to troubleshoot errors with DateTime columns. Indeed, when DateTime property is initialized to its default value it's set to 1/1/0001. When this value is saved to DB, ADO.NET's SqlClient can't convert it to a valid SqlDateTime value since the smallest possible SqlDateTime is 1/1/1753!!!

The easiest fix is to make the class property use Nullable type, in this case "DateTime?". Alternatively, you could implement a custom type mapper by implementing IUserType with its Equals method properly comparing DbNull.Value with whatever default value of your value type. In our case Equals would need to return true when comparing 1/1/0001 with DbNull.Value. Implementing a full-functional IUserType is not really that hard but it does require knowledge of NHibernate trivia so prepare to do some substantial googling if you choose to go that way.

Share

Improve this answer

edited Mar 27, 2012 at 16:46

answered Sep 16, 2009 at 17:09

Andriy Volkov

18.9k ● 9 ● 69 ● 84

Follow

In my case problem was with `Guid`. I used `Guid?` instead and problem solved. – Afshar Nov 26, 2014 at 10:36

16

I have seen this once before when one of my models was not mapped correctly (wasn't using nullable types correctly). May you please paste your model and mapping?

Share Improve this answer Follow

answered Aug 29, 2008 at 18:35

Ryan Duffield
19k ● 6 ● 41 ● 42

@Ryan (34965) Thanks for the hint. It took a ton of refactoring, but the issue was with nullable types. Once I fixed that, everything fell into place. Thanks! – Mark Struzinski Aug 30, 2008 at 3:00

I couldn't understand why (guys, please add links next time :)) googling got this: zvolkov.com/blog/post/2009/07/09/... – Yonatan Karni Sep 7, 2009 at 16:56

1 There are also other reason why this may happen. Have a look at nhforge.org/blogs/nhibernate/archive/2008/10/20/... for an automated test to detect these scenarios. – John Rayner Sep 17, 2009 at 1:00

0

I also experienced this problem in NH 2.0.1 when trying to hide the inverse ends of many-to-many bags using access="noop" (hint: this doesn't work).

Converting them to access="field" + adding a field on the class fixed the problem. Pretty hard to track them down though.

Share Improve this answer Follow

answered Jul 18, 2009 at 4:20

Richard Dingwall
2,732 ● 1 ● 31 ● 32