Nhibernate session per conversation example

Hello can some pros with nhibernate give me example implementation of Nhibernate session per conversation or unhaddins implementation? Or explain how to do that. Best regards Endiss

.net visual-studio nhibernate

edited **20 hours ago**
EtherDragon
846 1 11

asked **Apr 11 at 12:41**
Endiss
58 8
100% accept rate

Added tags to increase visibility – EtherDragon 20 hours ago

feedback

1 Answer

I work with NHibernate to 4 years. Previously I worked with the "open-session-per-operation" antipattern. The objects were always detached. So, to persist, I had to reattach them or copy their values to attached objects. This causes many lines of code and many "lazy initialization exeption".

Recently, I studied the "Conversation pattern" and I did an implementation over the the "Spring.Net" infrastructure. The implementation was submitted to "jira.springsource" at the Issue SPRNET-1431 (Workaround for 'conversation scope' and "session-per-conversation").

I made no "sample application", but if you're interested I can do that.

Hailton

Supplementary Answer:

I prepared the sample application and posted in SPRNET-1431 Workaround for 'conversation scope' and "session-per-conversation" as the file "Spring.Conversation.example.7z".

Below, I wrote explanations to clarify (or not) what I did.

This "sample application" is a modification of "Spring.Data.NHibernate.Northwind" contained in the version "1.3.0" of "Spring.NET" to use Conversation. Currently, the "Spring.Net" has no "conversation scope" nor implements the concept of "Extended Persistence Context" ("session per conversation strategy").

In this Sample Application the objective is demonstrate:

1. How to keep instances of objects in a mimicry of "conversation scope". Shown in `expression="@(convCustomer)['CustomerEditController']"`.
2. How to enjoy the "extended persistence context". "Lazy load errors" do not happen anymore and repeated calls to `ISession.Get` to a "same record" do not cause numerous visits to the database, more efficient use of the NHibernate cache. The modifications on "CustomerList.aspx" demonstrate this. To verify the effectiveness of Conversation, comment on "App_Code\ConversationPage.cs" the line `this.Conversation.StartResumeConversation();` then you will see the error "failed to lazily initialize a collection of role" occurring at the click of the button "+" on "CustomerList.aspx".

IMPORTANT: Never use a single conversation for the entire application (with the same duration of the "HTTP Session"). Remember, NHibernate keeps a cache of all loaded objects, if the conversation is held for a long time this cache tends to grow indefinitely (the limit is the amount of database records). That is, each conversation should be limited to a subset of the application pages, and must be discarded at the end of interaction with this subset (`IConversationState.EndConversation()`). Recommendation: Keep `<property name="EndPaused" value="true"/>` in "Spring.Conversation.Imple.WebConversationManager", so when start a conversation the others are discarded.

ADDITIONAL INFORMATION: The unit tests ("Spring.Northwind.IntegrationTests.2008") are not working. But there is no problem because it is not related to the changes made to support conversation, in fact they were already resulting in errors even before that.

List of changes in "Spring.Data.NHibernate.Northwind":

- Spring.Northwind.Web.References.2008
  - Adding links to schems to enable auto complete.
- web.config
  - module, added:

```
<add
  name="ConversationModule"
  type="Spring.Conversation.HttpModule.ConversationModule, Spring.Conversation"/>
<add
  name="ConversationModule"
  type="Spring.Conversation.HttpModule.ConversationModule, Spring.Conversation"/>
```

  - module, removed:

```
<add
  name="OpenSessionInView"
  type="Spring.Data.NHibernate.Support.OpenSessionInViewModule, Spring.Data.NHibernat
```

- web.xml
  - module configurations

```
<!--Configuration for Spring HttpModule interceptor's-->
<object
   name="HttpApplicationConfigurer"
   type="Spring.Context.Support.HttpApplicationConfigurer, Spring.Web">
  <property name="ModuleTemplates">
    <dictionary>
      <entry key="ConversationModule">
        <!-- this name must match the module name -->
        <object>
          <!--
          select "view source" in your browser on any page to
          see the appended html comment
          -->
          <property name="ConversationManagerNameList">
            <list element-type="string">
              <value>conversationManager</value>
            </list>
          </property>
        </object>
      </entry>
    </dictionary>
  </property>
</object>
```

  - conversation manager

```
<!--Conversation Manager-->
<object
    name="conversationManager"
    type="Spring.Conversation.Imple.WebConversationManager, Spring.Conversation"
    scope="session">
  <property name="SessionFactory" ref="NHibernateSessionFactory"/>
  <property name="EndPaused" value="true"/>
</object>
```

  - Customer Conversation

```
<!--
 Conversation for 'CustomerEditor.aspx', 'CustomerList.aspx',
 'CustomerOrders.aspx', 'CustomerView.aspx',  and 'FulfillmentResult.aspx'
-->
<!--
 Important: If the application had other parties
 ("management employees" for example), they should use another
 conversation.
-->
<object
    name="convCustomer"
    type="Spring.Conversation.Imple.WebConversationSpringState, Spring.Conversation"
    scope="session">
  <property name="Id" value="convCustomer"></property>
  <property name="TimeOut" value="0"></property>
  <property name="ConversationManager" ref="conversationManager"></property>
  <property name="SessionFactory" ref="NHibernateSessionFactory"/>
  <property name="DbProvider" ref="DbProvider"/>
  <!--
    Using workaround for 'conversation scope' to reference for
```

```
    'CustomerEditController'. It is not as volatile as "request scope"
    not as durable as the "session scope"
  -->
  <property name="['CustomerEditController']" ref="CustomerEditController"></propert
</object>
```

- Change "CustomerEditController" scope, remove [scope="session"] and put [singleton="false"]:

```
<object
    name="CustomerEditController"
    type="NHibernateCustomerEditController"
    singleton="false">
  <constructor-arg name="sessionFactory" ref="NHibernateSessionFactory"/>
</object>
...
```

- Change reference for `"CustomerEditController"` , remove `ref="CustomerEditController"` and put `expression="@(convCustomer)['CustomerEditController']"` (Simulating "conversation scope"):

```
<!--
  Using workaround for 'conversation scope' to reference for
  'CustomerEditController'. It is not as volatile as "request scope"
  not as durable as the "session scope"
-->
<object name="CustomerEditPage" abstract="true">
  <property
    name="CustomerEditController"
    expression="@(convCustomer)['CustomerEditController']"/>
  <property name="Conversation" ref="convCustomer"/>
</object>
```

```
<!--
  Using workaround for 'conversation scope' to reference for
  'CustomerEditController'. It is not as volatile as "request scope"
  not as durable as the "session scope"
-->
<object type="CustomerView.aspx">
  <property name="CustomerDao" ref="CustomerDao" />
  <property
    name="CustomerEditController"
    expression="@(convCustomer)['CustomerEditController']" />
  <property name="Conversation" ref="convCustomer"/>
  <property name="Results">
    <dictionary>
      <entry key="EditCustomer" value="redirect:CustomerEditor.aspx" />
      <entry key="CustomerList" value="redirect:CustomerList.aspx" />
    </dictionary>
  </property>
</object>
```

```
<!--
  Using workaround for 'conversation scope' to reference for
  'CustomerEditController'. It is not as volatile as "request scope"
  not as durable as the "session scope"
-->
<object type="FulfillmentResult.aspx">
  <property name="FulfillmentService" ref="FulfillmentService" />
  <property
    name="CustomerEditController"
    expression="@(convCustomer)['CustomerEditController']" />
  <property name="Conversation" ref="convCustomer"/>
  <property name="Results">
    <dictionary>
      <entry key="Back" value="redirect:CustomerOrders.aspx" />
    </dictionary>
  </property>
</object>
```

```
<object type="Default.aspx">
  <property name="Conversation" ref="convDefault"/>
  <property name="Results">
    <dictionary>
      <entry key="CustomerList" value="redirect:CustomerList.aspx" />
    </dictionary>
  </property>
</object>
```

```
<!--Conversation for 'Default.aspx'-->
<!--
```

```
        "convDefault" will have only one functionality: trigger the release
        of other conversations when started (StartResumeConversation())
      -->
      <object
          name="convDefault"
          type="Spring.Conversation.Imple.WebConversationSpringState, Spring.Conversation"
          scope="session">
        <property name="Id" value="convDefault"></property>
        <property name="TimeOut" value="0"></property>
        <property name="ConversationManager" ref="conversationManager"></property>
        <property name="SessionFactory" ref="NHibernateSessionFactory"/>
        <property name="DbProvider" ref="DbProvider"/>
      </object>
```

- Added "ConversationPage.cs". Base page with support for conversation.
- CustomerList.aspx
  - Allow list the "Order's" on the same page without "lazy initialization error". All objects stay attached to ISession (NHibernate).
- CustomerList.aspx.cs:
  - Added property `IList<Customer> CustomersLoadedOncePerConvList`. List loaded only once, searching the database only once per conversation.
  - Changing the `Page_InitializeControls` for consider `CustomersLoadedOncePerConvList`.
  - The method `BtnShowOrders_Click` performe implicitly a "lazy load" on `Customer.Orders`.
- Change `??? : Spring.Web.UI.Page` to `??? : Spring.Web.UI.Page` on:
  - CustomerEditor.aspx.cs
  - CustomerList.aspx.cs
  - CustomerOrders.aspx.cs
  - CustomerView.aspx.cs
  - FullfillmentResult.aspx.cs
  - Default.aspx.cs
- Dao.xml
  - Added `<entry key="connection.release_mode" value="on_close"/>` to avoid disconnection and reconnection before and after each IDbCommand execution. This is important because we will do more lazyload outside the transaction boundaries.
  - Added:

```
...
<entry key="show_sql" value="true"/>
<entry key="format_sql" value="true"/>
...
```

- Remove from "Default.aspx.cs" (they are never used):
  - `customerDao`;
  - `fulfillmentService`;
  - `CustomerDao`;
  - `Button1_Click(object sender, EventArgs e)`;
  - `ProcessCustomer()`;
- Config\Log4Net.xml.

```
...
<!--detail's about SQL's. To view sql commands on Logs\log.txt-->
<logger name="NHibernate.SQL">
  <level value="DEBUG" />
</logger>
...
<!--detail's about Conversation-->
<logger name="Spring.Conversation">
  <level value="DEBUG" />
</logger>
```

Hailton de Castro.

edited **22 secs ago**

answered **yesterday**

user1350308

26 2

Can you make some sample it will help me a lot :) – Endiss yesterday

Was this post useful to you?