Transfer: Incrementing IDs and an Oooops

Posted At : July 21, 2008 6:53PM | Posted By : Matt Quackenbush
Related Categories: ColdBox, ColdFusion, Transfer

Transfer allows you the option of having the database handle your IDs for new records (e.g. 'identity' in MSSQL or 'auto_increment' in MySQL), which is the default behavior for Transfer, or you can have Transfer generate your new IDs for you by utilizing the 'generate' attribute on your ID tag in your transfer.xml config file, and setting it to 'true'. I personally prefer having the database handle my IDs for me, so I never really paid much attention to this particular feature of Transfer until recently.

One day last week I was doing a bit of testing on a new application. When I got to the point of testing a particular function of the application, an exception was thrown:

> [Macromedia][SQLServer JDBC Driver][SQLServer]Cannot insert explicit value for identity column in table 'tblFoo' when IDENTITY_INSERT is set to OFF.

"WTF? Why would Transfer suddenly be trying to insert an ID?" went coursing through my mind. "It's never done this to me before, and I haven't made any changes to the config. What gives?"

Here's a quote from the Persisting and Retrieving Objects section of the wiki :

> If in the configuration the ID element has a [@generate='true'] value, Transfer will generate the primary key.
>
> If [@generate='false'], or there is no 'generate' attribute, the database will attempt to retrieve the primary key value from the database.
>
> **If the ID is set() before the TransferObject is inserted, this will override both of these options, and Transfer will attempt to insert the data into the database under the set ID.**

Note the bold portion (emphasis mine). Transfer assumes that if you have explicitly set the ID on the object prior to it being persisted, you must know what you're doing, and therefore it tries to insert the ID.

The "Oooops"

The very first thought that went through my head was "but I didn't set the ID!". It was then that I realized I had indeed set it, albeit unintentionally.

```
<cffunction name= "someMethod" >
<cfargument name= "Event"
hint= "The event object"
required= "yes"
type= "coldbox.system.beans.requestContext" />
<cfscript>
// get our object
var object = getService().getObject();
// populate the object from the request collection
getPlugin( "beanFactory" ).populateBean(object);
</cfscript>
</cffunction>
```

When the BeanFactory plugin's populateBean() method receives our object, it will grab all of the variables from the request collection and look for a setter on the object that matches. So, for instance, if the request collection has "ID" in it, and the object has a SetID() method on it, the BeanFactory plugin will use it to set the ID from the request collection.

There are a number of times in an application when during the request cycle you'll want to execute multiple events/methods. This particular request that I was testing does exactly that. The first method in the request cycle has an ID, since it needs it to process that particular portion of the request. The problem lied in the fact that when I sent the request to the next method, 'ID' still existed. As a result, the object in that method was having its ID implicitly set prior to save().

While it would have taken me quite some time to find had Mark Mandel not mentioned that it sounded like I was setting the ID prior to save, it was, thankfully, a simple fix. If you're using ColdBox , you just do something like this in your controller (handler):

```
<cffunction name= "someMethod" >
<cfargument name= "Event"
hint= "The event object"
required= "yes"
type= "coldbox.system.beans.requestContext" />
<cfscript>
// do some stuff here
// prepare to send to next event/method
event.removeValue( "id" );
runEvent( "next.event" );
</cfscript>
</cffunction>
```

By removing 'ID' from the request collection in the first event/method, we don't have to worry about it being implicitly set in the next event.

NOTE: This is a really tough concept to try and explain in a blog post, so my apologies if it doesn't make sense. My hope is that it'll make just enough sense to save someone a headache.