

Re: pooled connection myth

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2005-03/1770.html>

From: John C. Bollinger (jobollin_at_indiana.edu)

Date: 03/17/05

Date: Wed, 16 Mar 2005 18:32:04 -0500

David McDivitt wrote:

> *You said the magic words: gives back the connection to the pool. There is no
> pool. Each time a connection is closed it is gone. To use a connection, a
> new one must be instantiated from the driver with user name and password.*

You are making a rather large assumption: that the connection object you obtain from the pool is an instance provided by the driver you described. In fact, I would expect this to not be the case. I would expect the pool to provide a Connection object that `_wraps_` the low-level Connection and, among other things intercepts `close()` invocations as signals to release the underlying connection back to the pool.

> *What you describe is part of the myth regarding pooled connections. When the
> framework pools, it pools drivers only. In other words, all connections made
> to your Oracle go through the same instance of the Oracle driver. If you had
> Oracle, DB2, Sybase, and SQL Server for instance, connections would be
> pooled with regard to the major data source, but individual connections
> would not be pooled.*

I think you're all wet. It is `_possible_` for a DataSource or other resource pool to operate as you describe, but it is certainly also possible for them to provide true pooling — of Connection objects, for instance — in such a way that the existence and proper use of the pool is transparent to users of the pooled objects (once one is obtained).

> *To modify your connection manager, try instantiating a connection object at
> class level and after the first connection request, continue returning the
> same connection object from the class level. That one connection object
> should support multiple threads. Add a timer thread to the connection
> manager to read a dummy table each sixty seconds, and if it fails, null the
> connection, and recreate on the next request. Finally, put a close method on
> the connection manager to stop the timer thread to be used when the app
> server stops. This is connection pooling. Anything less is not connection
> pooling.*