

# DAOs and Connection sharing

**Source:** <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2004-07/4252.html>

---

**From:** janne ([janne\\_1976\\_at\\_hotmail.com](mailto:janne_1976_at_hotmail.com))

**Date:** 07/30/04

Date: 30 Jul 2004 01:06:24 -0700

I have been using a DAO –based design to implement my applications with the following principles:

- business / persistence logic is handled by 3 layers; Actions, business logic classes and DAOs
- logic in a use case is handled by an Action (Struts is not used; but for the sake of discussion you might as well assume it is)
- Action gets a Connection from a DataSource; and passes it to each business logic class it uses
- When persistence is needed, business logic class calls a DAO and passes it the Connection. All database activities therefore share the same connection and transaction.
- After Action is finishing, Action either commits (if no exceptions were thrown) or rolls back (in case of exceptions) the connection, and closes it.

This works fine, but there's coupling between actions–business–persistence with the passed Connections. I have seen suggestions that an alternative approach could be used:

- Connection objects are not passed as method arguments. Instead each DAO fetches Connections from DataSource directly
- I would still use Action to commit / rollback the whole transaction

The basis for this, according to some, is that when different objects in same thread request connections from a datasource, they in fact get the same shared connection. Which is a requirement for this to work, of course.

Does this work, and if it does, can you point to a specification that describes DataSource's correct behaviour? I have not been able to find any.

Hibernate and other alternatives are all very nice, but I want to concentrate on this kind of approach now...