EntityManager Per Session Pattern

By Roger Keays, 28 January 2007

A recent webapp I was working on had a fairly simple requirement for a paged table of data from the database. Because of the large amount of data available it couldn't just all be loaded into memory at once, which means a bit more work on the UI side. So rather than write all the plumbing to manage paging and queries I thought I'd give OpenJPA's Large Result Sets (LRS) a try. A LRS is a List which looks and behaves like all the data is already loaded, but is actually fetching it on the fly using database cursors. It all worked wonderfully, except for a few.. erm.. slight problems using the EntityManager-per-session pattern, which is required to keep the LRS open.

In retrospect I suppose it's fairly obvious, but using a LRS with one em-per-session has fairly serious scalability problems. Each session in this app held a LRS open which meant consuming one database connection per session. This would be fine if you were only expect a handful of concurrent sessions, but when testing this application under load, it couldn't handle more than about 150 sessions.

Further problems were unearthed when the LRSs sometimes mysteriously closed themselves leaving no rhyme or reason as to why. Under some circumstances the problem was reproduceable and was rectified by changing the transaction isolation level. It seems that a concurrency issue caused the database to close cursors after transaction commits occured in a problematic order. The only reliable solution was to place a per-session mutex on the entire JSF Render Response phase, so that each user could only render pages containing the dataTable sequentially.

Finally, there was the issue of memory usage. While it's great that a LRS lazy-loads the data, how much sense does it make to keep this in the session scope? Well, the answer to that depends on how long your users' sessions are, and how much variation there is on that average. Generally though, you'll end up with a whole lot of data in memory that nobody is viewing any longer and it'll be there until the session times out.

So, in summary, I found three problems using the em-per-session pattern:

1. Scalability issues, since each session requires a new connection and we need to handle 500+ simultaneous sessions.
2. Concurrency issues in which, depending on the transaction isolation level and SQL statement order, a transaction commit can cause the LRS to be closed.
3. Memory usage, since the partial results are all stored in the session until the session expires.

Needless to say, the app is now using one EntityManager per request. By comparison, this version only uses 3 - 5 database connections to handle 150 sessions and the response times are a lot quicker even with caching disabled and the database being hit for each request. It seems that cursors have their own set of performance problems too (it never ends!).

Looks like EntityManager-per-request wins again.

About Roger Keays

Roger Keays is an artist, an engineer, and a student of life. He has no fixed addressand has leftfootprints on 40-something different countries around the world. Roger is addicted to surfing. His other interests are music, psychology, languages, the proper use of semicolons, and finding good food.