

Re: CLOS persistence

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2008-01/msg00221.html>

- *From:* levy <levente.meszaros@xxxxxxxxxx>
 - *Date:* Fri, 4 Jan 2008 04:02:07 -0800 (PST)
-

On Jan 4, 6:13 am, vtail <victor.kryu...@xxxxxxxxxx> wrote:

Interesting. At that point elephant seems to be easier to use, because it has some pretty good documentation and tutorial, and is asdf-installable. I have nothing wrong with getting a package via darcs – in fact, distributing your sources via some distributed SCM is probably the best way to involve community in development – but I spent several hours yesterday trying to install all the dependencies (and it still doesn't run: (asdf:oos 'asdf:load-op 'cl-perec) reports "The name CL-PEREC does not designate any package" while doing 14: (COMPILE-FILE #P"/home/victor/.sbcl/site/cl-perec/configuration.lisp"). On the other hand, I really don't like how it's

Hmm, I don't know what causes that, maybe you could send some more details to the devel list?

misusing database backend for a hash. If cl-perec provides a better map between objects and database, it's worth a serious consideration!

Cl-perec basically maps each persistent class into a table (which is created automatically) and some persistent associations (namely the many-to-many ones) too. The class tables have one (in non default modes two) extra column for the oid and one or multiple columns per each primitive slot and the usual foreign key columns for associations. If a persistent class is abstract and would not have a column at all then it does not have a table to avoid unnecessary inserts. Persistent instances are cached within the transaction and their (slot) data is prefetched and cached upon first access or when querying. Within a transaction it is guaranteed that two instances will be eq iff their oid is the same independently of how you got the two instances (by querying or navigating, etc.)

I understand that making (good) documentation takes a lot of effort and reading cases / sources is usually enough, but having good quick tutorial + API documentation simplify things big way. Have you considered any tools that automatically extracts documentation strings

Re: CLOS persistence

(like doc/manual/docstrings.lisp from the sbcl source tree or such)?

No, not yet, but patches are always welcomed! ;-)

control different threads for using different connections etc. – am I right that cl-rdbms is thread-safe by default?

It is, we are using it in an application with more than 50 threads per node and sometimes threads may even require nested transactions.

On the other hand, CLSQL supports sqlite, which is an important backend IMHO – very easy to install and sometimes faster than Postgresql due to lower overhead. How hard it is to add sqlite backend to cl-rdbms?

I think it's quite straightforward but might take a couple of days to do correctly. Basically you need to subclass the database and transaction classes, write some generic functions to connect/disconnect, do some reflection on tables, prepare and execute a statement and specialize the SQL printer if needed.

If you look at the postgresql backend, it's only 370 LOC (based on postmodern) which is not that much after all. The oracle backend is nearly 2000 LOC not counting the generated CFFI interface.

levy

.