Function persistence in Common Lisp

Asked 16 years, 1 month ago Modified 15 years, 7 months ago

Viewed 1k times

9

Is there any persistence solution for Common Lisp, such as Elephant, that allows function persistence? Currently my app stores an identifier on the db and later searches in a function table which it is, but this method does not allow dynamically created functions to be stored.

database lisp persistence common-lisp

Share

Improve this question

Follow

edited May 20, 2009 at 5:32

Pablo Fernandez

287k ● 139 ● 401 ● 641

asked Nov 15, 2008 at 20:52

AticusFinch

2,421 ● 3 ● 26 ● 32

6 Answers

Sorted by: Highest score (default)

It's not a database persistence mechanism, but most Common Lisps have a way of writing FASLs for all kinds of objects, including functions. For example:

4

```
cl-user(1): (compile (defun hello () (format t "~&Hell
hello
nil
nil
cl-user(2): (excl:fasl-write (symbol-function 'hello)
t
cl-user(3): (excl:fasl-read "/tmp/hello.fasl")
(#<Function hello @ #x1000a964d2>)
```

You can write to a stream (here I used a file for convenience), so you can trivially capture those bytes and stuff them into a database if you wished.

Share Improve this answer

Follow

edited May 20, 2009 at 5:32

Pablo Fernandez

287k ● 139 ● 401 ● 641

answered Nov 15, 2008 at 22:04

Rich

2,893 ● 1 ● 20 ● 20

1 But you should note, that it isn't in the CL standard.
– Anton Kazennikov May 20, 2009 at 6:54

3

Pascal Bourguignon gave a standard solution on comp.lang.lisp. Basically you have to write the source form to a file and `COMPILE` it then `LOAD` it.

```
(defvar *anon*)

(defun save-anonymous-function (fname args body)
  (let ((fname (make-pathname :type "LISP" :case :comm
    (with-open-file (src fname :direction :output
```

```
                          :if-does-not-exist :create :if-ex
        (print `(defparameter *anon* (lambda ,args ,bod
      (compile-file fname)))
```

Then you'll have to read the file and store it in your database. To get it back you'll need to fetch it from the database and write it in a file before loading it.

```
(defun load-anonymous-function (fname)
   (let ((*load-verbose* nil)
         (*anon* nil)) ; to avoid modifying the global
     (load fname)
     *anon*))
```

Share Improve this answer

Follow

answered Nov 18, 2008 at 8:47

kmkaplan

31 ● 1

2

Common Cold might be what you want. It includes serializable closures and serializable continuations.

Share Improve this answer

Follow

answered Nov 16, 2008 at 19:14

Luís Oliveira

2,964 ● 1 ● 20 ● 26

On the internet archive: web.archive.org/web/20170331132917/http://… – Ehvince Jul 8 at 10:38

1

Functions are opaque objects, so you won't have much luck storing them in files or something like that. You can, however, store lists and compile them upon retrieval from the database.

This won't help you store closures, of course. That would entail storing the lexical environment along with the code, neither of which you have (portable) access to. Code that you compile from stored lists will have to rely entirely on global data or data stored in the database.

By the way, note that you can funcall symbols, so you don't need a function table for global functions.

Share Improve this answer Follow

edited Nov 15, 2008 at 21:13

answered Nov 15, 2008 at 21:08

Matthias Benkard

15.7k ● 4 ● 41 ● 47

1

You can look into saving Lisp images. This allows saving "enough information to restart a Lisp process at a later time." You can have your functions saved after you have loaded them into your image.

This may also be a bit more advanced than what you were looking for, but here is a (very) brief introduction to the process: Saving a Core Image

Share Improve this answer

Follow

answered Nov 15, 2008 at 21:23

helloandre

10.7k ● 9 ● 49 ● 64

0

Be careful that storing code may not be that good. The Zope developers learned it the hard way.

Share Improve this answer

Follow

answered Nov 26, 2008 at 14:21

Pablo Fernandez

287k ● 139 ● 401 ● 641

1 What is that supposed to mean? A little more explicitness would be much more useful... – AticusFinch Dec 28, 2008 at 14:23