

Re: sbcl – how to get a REPL for a created thread?

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2006-02/msg01312.html>

- *From:* Glenn.Ehrlich@xxxxxxxxxxxxxx
 - *Date:* 10 Feb 2006 17:26:08 –0800
-

Karol,

With that version of SBCL, you need to get the CVS version of Slime. There was a problem with how Slime initialized some threads inside SBCL. I reported this on the slime list about a month ago and Helmut fixed it for me. Additionally, the Slime startup sequence is slightly different, so here's the procedure for doing what you want to do.

The steps you need to do are:

1. In a shell, start SBCL up manually. You'll then need to start Swank up manually. At the SBCL REPL in your shell, you'll need to do these steps:

– Load the Swank loader lisp file:

```
(load "swank-loader.lisp") ; or wherever you place slime
```

– Have Swank load the rest of Swank in for you:

```
(swank-loader:load-swank)
```

– Optionally, load the Slime presentation code:

```
(load "present") ; because I have mine compiled
```

– Finally, you need to start Swank running within SBCL. I have this convenience function loaded in from my lisp startup file:

```
(defun start-swank (&key (port swank::default-server-port)
                    (style swank:*communication-style*))
  (dont-close t)
  "Start the SLIME swank server, listening on PORT. Multiple
connections are allowed."
  (swank:create-server :port port
                      :style style
                      :dont-close dont-close))
```

So, I just start up swank with:

Re: sbcl – how to get a REPL for a created thread?

(start-swank) ; this starts up Swank with the Slime default port.

2. In Emacs then, you need to start up Slime, but you need to use slime-connect instead of the slime command:

M-x slime-connect

It will prompt you for several parameters, like the port. If you started up Swank like I showed above, you can just keep hitting <enter>.

At this point, Slime should start up.

3. So you now have Slime with 1 Slime REPL. To get additional Slime REPLs, just do M-x slime-connect again, but you'll want to make sure that you answer N to the question about closing the current connection. If you do it correctly, then you should have a second REPL buffer and you can use the normal Emacs buffer commands or windowing to manager your REPLs. You can also use the slime connection chooser, but I find it to work a little odd.

I've used this many times in debugging socket and threading code and it works pretty cleanly.

One thing that might not be obvious is that if you use the Slime compilation commands and what not in a Lisp buffer, that those commands go to the last REPL created, unless you use the Slime connection thingy to change the default REPL.

Hope this helps.

Glenn

.