

## Re: Advice on mod\_lisp-based web application.

---

*Source:* <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2007-04/msg01158.html>

---

- *From:* Tim X <[timx@xxxxxxxxxxxxxxxxxxx](mailto:timx@xxxxxxxxxxxxxxxxxxx)>
  - *Date:* Tue, 24 Apr 2007 20:35:58 +1000
- 

Blaine <[Blaine.M.Nelson@xxxxxxxxxx](mailto:Blaine.M.Nelson@xxxxxxxxxx)> writes:

I've got a lisp simulation program that serves its interface over html. After all kinds of trouble with IT folks at my organization – paperwork, having to migrate from Portable Allegro Serve to mod\_lisp/Apache, I have a beta version on our intranet.

The beta works fine for one user, but, understandably, it doesn't work for multiple users. It's understandable because 1) multiple concurrent users isn't (at lease wasn't) a requirement and 2) I'm just a noob.

My kludgy and hopefully short term solution strategy is to allow only one user at a time. There are only a few (less than five) potential users, so it's not as bad a solution as it sounds.

What I was hoping for is some advice on implmentation. At the very least, it seems, I have two desiderata:

1. Persistence. My server needs to be able to maintain connection with a single, unique user while that user is using the simulation, and politely block other users.

and

2. Termination. After some pre-specified idle period, terminate the "session" (not sure what word I should be using here), thus allowing a new user

Re: Advice on mod\_lisp-based web application.

access.

As for the Persistence issue, I was thinking I could use remote-ip-address key/value pair that comes with every modlisp request to identify the user.  
Will this work, or will I have to do something with cookies?

As for the termination issue, I've been tinkering around with a timeout. One of the options when you start a modlisp server is a timeout. When that timeout is reached, a timeout condition is signaled. Presumably, I could write code that could handle that condition, offering the user a choice to continue using the simulation, and in the event that the user doesn't respond, close out the user's "session". I'm not sure this is an appropriate use of the condition system, and furthermore, I don't know where I would put the handling code (a handler-bind form, I believe).

My guess is I have more problems than just Persistence and Termination. Any comments on either of these or the other problems that I'm not yet aware will be much appreciated.

Thanks in advance,  
Blaine

PS A while back I wrote a little noob modlisp install guide that may provide some details that may help answer my questions:  
<http://www.blaino.com/guide/modlisp-pcl-guide.html>

I think you may need to re-think your paradigm. It seems to me that you are approaching the problem from the wrong direction. HTTP is essentially a "stateless" protocol rather than a stateful connected protocol. The use of cookies etc, can simulate a stateful interaction, but its not like other more connection oriented protocols, wehre both ends are always connected and communicating over the same connection. While Apache and other servers to provide some support for maintaining connections and connection pooling, this is more related to improved performance in establishing a connection rather than maintaining a continuous bidirectional communication channel.

Re: Advice on mod\_lisp-based web application.

Re: Advice on mod\_lisp-based web application.

You said your applicaiton would not work with more than one user. I'd concentrate on fixing that issue. Approach the whole client/server interaction as one which is stateless and then use cookies (or even REMOTE\_USER) as a way of connecting states and maybe have your application save/load past states of your lisp simulation before/after dealing with a client request. This will allow you to avoid the whole issue of trying to restrict access to a specific user in a serial type interaction model and as a side effect, you will be able to support multiple users at the same time (ignoring performance/resource issues of course).

Once you assume a stateless interaction model in which you cannot know the order or sequence of requests you will recieve, lots of problems just vanish. Most of the time, problems I've seen people having with web based applications are due to incorrect assumptions about the order/sequence of interactions you will get from the same and different clients. this is an understandable mistake because from an end user perspective, it does appear to be a well defined sequence/series of interactions.

Tim

—

tcross (at) rapttech dot com dot au

.