Subject: Re: Lions and tiger and objects, oh my!
Posted by David Fanning on Mon, 03 Nov 2008 14:48:52 GMT
View Forum Message <> Reply to Message

Mike writes:

> I haven't looked at your code in much detail yet, but I'm curious
> about CatEventHandler. Does it function like IDL's
> generic_class_event, generic_class_notify_realize and
> generic_kill_notify? Is it better/worse/different? Can you provide a
> comparison?

I've never used these routines, nor can I find them, but
I imagine they are designed to do exactly the same thing:
throw widget events and callbacks into the object system.

So, in that sense, generic_class_event is probably
similar to CatEventHandler (not it's real name, see
below). However, CatEventHandler is *much* more, as
it is designed to deal with objects and not just widget
events. So the "event" structure is repackaged so that
the relevant fields in the structure are objects and not
widget identifiers. In other words, event.id is an object
reference to the object that caused the event, not a
widget identifier of the widget that caused the event.
Other fields are similarly modified.

> Also, is it really called CatEventHandler? I don't see
> it anywhere in my copy of the sources...

No, it is called CatEventDispatcher. I call it by some
other name to throw would-be users completely off the
track :-(

The routines you are looking for are found in the "utilities"
directory of the Catalyst distribution:

  CatEventDispatcher
  CatKillNotify
  CatRealizeNotify

There is debugging code in CatEventDispatcher that
can be uncommented. This is extremely useful at times
for figuring out where your events are going. Since
all objects in the Catalyst system have "names" it
is easy to see which object is sending, and which object
is receiving, an event.

I am getting *very* close to an "official" release of this
material, but there are still changes going on almost daily.
Last night, for example, I fixed a long-time vexing problem
with CatAtom, the fundamental object in the system, that
was giving me some trouble with object deep-copying. Now,
basically, any object in the system can make an identical
copy of itself by simply calling the COPY method on itself.
Pretty slick, and used in the right situations, enormously
useful. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")