Subject: Re: Destroying objects
Posted by JD Smith on Mon, 16 Feb 2004 21:11:56 GMT
View Forum Message <> Reply to Message

On Mon, 16 Feb 2004 11:18:44 -0700, David Fanning wrote:

> Andy Meigs writes:
>
>> Starting to work on objects as well using Martin Schlutz's mgs_objects base
>> class system-- mighty frustrating when 'errors' (probably not of Martin's,
>> but of my ignorance in how to use his code) occur several superclasses from
>> the class you have written.
>
> Yes, well, one of the things we *have* done right in our
> library is implement excellent error handling. The problem
> we had originally was cascading error messages. Something
> would happen way down deep, and the message would propagate
> up the entire object chain. It sounded like one of those
> fire alarms you can't turn off. :-(
>
> Very disconcerting to new users of the library.
>
> We figured out a way to indicate that the error was already
> "handled" so objects further up the chain could just pass it
> silently. Users get one error, and the traceback points to
> the right place 99.9% of the time. Note that this is a
> different technique from handing the *error* silently, which
> is what the iTool library does. Using the latter method, you can
> be frustrated for weeks on end, since it takes four or five
> times as long to realize you *have* an error than it does
> to fix it. :-)

I wonder if you can expand on this error handling technique a bit.
One feature I like to have is the ability for errors to "just work",
regardless of if the object is being used on the command-line, or via
its GUI. In the latter case, errors should be trapped and displayed
in a pop-up. In the former case, they should call MESSAGE to print to
the command line and exit. I've managed it so far by having a
super-class which implements Error, Warning, and other methods, which
checks to see if you are in a command-line invocation or not, and uses
message or dialog_message as appropriate. Suitable use of CATCH
allows deep errors in other routines further down the stack (ones you
didn't write, for instance) to be handled in the same way, but that
requires some care to "set the trap" when necessary. This is somewhat
similar to the way XManager can catch and disregard errors (as it does
by default), but with reporting, and valid even when XManager isn't
running.

JD