Subject: Re: Speaking of RESTORING
Posted by Pavel A. Romashkin on Wed, 27 Dec 2000 16:50:00 GMT
View Forum Message <> Reply to Message

I think that in that recent thread somebody did point this one out. I guess it indicates people do read manuals, even though they complain about them, huh?

Cheers,
Pavel

David Fanning wrote:
>
> Folks,
>
> While we are on the topic of restoring SAVE files,
> have you noticed the new RESTORED_OBJECTS keyword
> on the RESTORE command? You can get a list of saved
> object references. This should make it much easier to
> re-compile the object methods. :-)
>
> Cheers,
>
> David

Subject: Re: Speaking of RESTORING
Posted by John-David T. Smith on Fri, 29 Dec 2000 17:29:36 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Folks,
>
> While we are on the topic of restoring SAVE files,
> have you noticed the new RESTORED_OBJECTS keyword
> on the RESTORE command? You can get a list of saved
> object references. This should make it much easier to
> re-compile the object methods. :-)

That's been around for a while, since 5.2 maybe (?). In fact, that's how the original version of resolve_object worked (though I think it was called restore_object then).

Not to belabor this point, but the issue at hand is that you need to have defined the class *before* you restore the object (which has the side effect of defining it using a potentially out of date definition). Hence the whole

resolve_object, CLASS=class formalism. And hence Craig's efforts to parse the save file for class information prior to restoring it (to obviate apriori knowledge of the object's class).

By the way, there are two ways to get the object (suppose only one), in the SAVE file:

restore, RESTORE_OBJECTS=obj

or simply

restore

with the variable name of the object reference known. This is most convenient inside of an object method itself, when the object SAVE'd is called "self". The current object ref "self" will be overwritten (think of self as a pass-by-reference extra argument to all methods). This can obviously be dangerous, so be careful.

JD
