Subject: Re: Saving an application state
Posted by robert.dimeo on Fri, 25 Jun 2004 11:39:06 GMT
View Forum Message <> Reply to Message

Many thanks to all of you for your good ideas. This has been very helpful.

Rob

JD Smith <jdsmith@as.arizona.edu> wrote in message
news:<pan.2004.06.25.01.36.40.142685@as.arizona.edu>...

```
>  David is right, I do have a preferred method I've written about, and it's
>  fairly easy.  It essentially requires you to hide all the unnecessary
>  widget info behind a single pointer.  I usually use self.wInfo.  Accessing
>  said data is then a bit more tedious ((*self.wInfo).wButton instead of
>  self.wButton), but it gives you two things:
>
>  1. Your object can live and run without a GUI.  Good for batch scripts
>     and command line processing.
>
>  2. You can save your object without the unnecessary (and harmful)
>     transient widget data by simply "detaching" them before saving.
>
>  A simplified version of my save method is:
>
>  pro myClass::Save,file
>      detwInfo=self.wInfo ; detach
>      self.wInfo=ptr_new() ; a null pointer save, self,FILENAME=filename
>      self.wInfo=detwInfo  ; reattach
>  end
>
>  which works nicely, unless you have trouble saving (no permissions, etc.),
>  in which case some error checking is a good idea:
>
>  pro myClass::Save,file
>      detwInfo=self.wInfo ; detach
>      self.wInfo=ptr_new() ; a null pointer catch, serr
>      if serr eq 0 then save,self,FILENAME=file catch,/CANCEL
>      self.wInfo=detwInfo         ; reattach if serr then message,'Error
>      saving to file: '+file
>  end
>
>  Anyway, this is all you really need.  Restoring is pretty simple, but you
>  have to be a bit careful to make sure your restored object doesn't
>  overwrite the class definition of a changed class (which in principle
>  means compile that class before restoring).  My RESTORE_OBJECT routine
>  takes care of that (or you can do it yourself).
>
```

> Other fun tricks you can do with SAVE/RESTORE on objects include
> overwriting the existing "self" pointer in-place... a *very* easy way to
> implement "recover from file", which I call "transmogrification".
>
> David is right that SAVE very diligently follows the data structures
> thoroughly through all pointers and composited objects, and saves
> everything, but if you're careful to detach what you don't want or need,
> you'll be in fine shape. There's lots to read on the NG on this topic.
>
> JD