Subject: Re: GUI and non-GUI objects in Catalyst
Posted by David Fanning on Thu, 03 Dec 2009 00:00:26 GMT
View Forum Message <> Reply to Message

Jean-Paul Davis writes:

.
>
> I'm thinking about writing my first Catalyst application, and I'm
> looking for advice from others more experienced with Catalyst on the
> best design approach to handle the following. I will have a
> complicated object (call it "AnObject" for the sake of argument)
> containing data (including other objects) and methods (including read/
> write). I want this object to have a GUI method as an optional way to
> interact with the data in the object, but I do not want to create a
> GUI unless it's needed; i.e., creating an instance of AnObject should
> not by itself automatically create a GUI.
>
> The examples in Catalyst are full "applications" in the sense that the
> application itself is a class derived from a top-level base widget
> class. I could use this application template and simply not call the
> GUI method in order to avoid creating/mapping a GUI, but it seems
> inelegant to use the TLB-widget class as the basis for a non-GUI
> object class. I could make the GUI a separate object that contains an
> AnObject, but that introduces a level of referencing between the
> interaction (GUI) object and the data being manipulated. Or I could
> reverse this so AnObject contains the GUI object if needed, although I
> haven't though through the data referencing implications. Would one
> of these approaches be the most appropriate, or are there other,
> better approaches?

I often build GUI methods for objects that I sometimes, but not always, want to interact with via a graphical user interface. My NCDF_DATA object is a good example:

http://www.dfanning.com/programs/ncdf_data__define.pro

I often use that without a GUI interface. But if I need one, I can just call the BROWSE method on the object. That, in fact, is exactly what the (extremely!) short NCDF_BROWSER program does:

http://www.dfanning.com/programs/ncdf_browser.pro

I can't think, off-hand, of an instance where I have used the Catalyst Library to build the GUI as in this example, but I can't think why it wouldn't work in principle. Catalyst does assume an object hierarchy, and since it is for building "applications" it pretty much does assume a GUI of some sort. But I think the widget part of the library can pretty much stand alone without difficulty.

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.")

Subject: Re: GUI and non-GUI objects in Catalyst
Posted by Jean-Paul Davis on Fri, 04 Dec 2009 23:49:18 GMT
View Forum Message <> Reply to Message

David,

Thanks for the thoughts... It does sound like the easiest way to build an object that doesn't necessarily need a GUI when used by other applications, but that can be used by itself as a GUI application, and that takes full advantage of the Catalyst object heirarchy, is to go ahead and follow the Catalyst application template (wherein my object is derived from the Catalyst top-level base widget class), and don't worry about carrying around extra data and methods that don't matter unless a GUI is generated.

If I understand your last sentence, an alternative is to NOT derive my application from a widget class, but instead have it contain individual objects for all the needed widgets (including a TLB). I'll probably go with the first approach to learn Catalyst faster/better.

Jean-Paul

Subject: Catalyst Object Widget Hierarchy
Posted by Jean-Paul Davis on Wed, 16 Dec 2009 23:38:54 GMT
View Forum Message <> Reply to Message

David,

Thought I'd keep this discussion here since my next question is so closely related: what is the purpose of the WidgetBase::ADD and WidgetAtom::ADD methods? When creating an object widget hierarchy

using Catalyst, I see in your examples that you simply create the individual widget objects from within the top-level object's INIT or GUI method, using named variables for the object references only when needed as the parent argument to a child widget or as a property of the top-level object. Would there ever be any reason to "ADD" child widget objects to parent widget objects?

I know you've been asked before, but do you think there's even a remote chance that someone (you, Burridge, or even someone else) might ever write a book on how to use Catalyst?

Jean-Paul
