Subject: hierarchical tree data structure & binding to TreeCtrl
Posted by solareclectic on Thu, 01 Nov 2007 07:35:28 GMT
View Forum Message <> Reply to Message

I've a few questions/ideas I'd like to bounce of those more experienced with U++...

I've more than a few applications to write that all want to have a hierarchical tree-like data structure as their primary structure. I see that TreeCtrl has the Key/Value (which, handily, take a Value) and all the requisite add/remove/count/find child/parent functions that one should expect in a Tree. At first glance, I thought I'd use whatever TreeCtrl inherits from- expecting to find a "Tree", to which Tree"Ctrl" then added all the GUI aspects; but, alas, I find it all in TreeCtrl.

Now, it just so happens that these tree data structures are, in fact, meant to be directly manipulated by the user; and it would be the simplest matter to just use the TreeCtrl directly as the data structure and add/remove my various other objects into the Key/Value "Values," using .Is() to determine the class of object that had been stuck in to a Node previously, and ValueTo<MyObject> to get them out and call their functions.

Perhaps there is an easier way, still, to call functions of their common base class with out doing the ValueTo<MyClass> bit?
I had tried making my common base class "Rich," but wasn't entirely successful, and wasn't sure it was actually necessarily any easier.

Now, to the real question... I'm looking for an obvious way to keep that Model/View/Controller separation that we've all been compelled to accept as gospel. I find no compelling reason (for my applications) or examples of TreeCtrl keeping the MVC separation with an datastructure internal to the application (the filesystem is an external datasource providing hierarchy, an other sample populate it with assorted labels - but no "linking" to internal datastructure). Can I just use TreeCtrl as my datastructure, as it provides all the functionality that I require, including the GUI stuff? Or if MVC separation must be maintained, is there a harm in using two TreeCtrls - one for the View, and one for the Model that just never gets shown?

And, a bonus question... (though for different widgets)
Since I'm taking shots a MVC separation anyway, I'm finding a similar situation with U++ nifty EditField derived widgets, which have all of the range/notNull checking, etc. that I want on to protect my Model object fields from getting set incorrectly, from the GUI or otherwise. If I want to keep, say, an Int datamember in one of my objects AND I want the user to manipulate it, why wouldn't I declare that datamember as EditSpin, instead of Int? Then when I want to present it to the user for editing, I simply have it draw itself?
Are there major performance/memory issues with this behavior?

Now I do, honestly, understand and respect the value of MVC separation, but since U++ has so conveniently packaged obviously useful behaviors together, I thought I should ask you all to see if I'm just making things harder on myself and overlooking the obvious "U++ Way" of doing things.

Thanks for any insight any of you can provide regarding best use of U++. Regardless of this answer, U++ has restored my interest in programming for FUN again.

shannon

Subject: Re: hierarchical tree data structure & binding to TreeCtrl
Posted by mirek on Sun, 11 Nov 2007 22:38:22 GMT
View Forum Message <> Reply to Message

[quote title=solareclectic wrote on Thu, 01 November 2007 03:35
Now, to the real question... I'm looking for an obvious way to keep that Model/View/Controller separation that we've all been compelled to accept as gospel.
[/quote]

Well, I am afraid, if you are looking for rigid MVC model, you are in the wrong place here...

Quote:
Can I just use TreeCtrl as my datastructure, as it provides all the functionality that I require, including the GUI stuff? Or if MVC separation must be maintained, is there a harm in using two TreeCtrls - one for the View, and one for the Model that just never gets shown?

Actually, why should your model be another TreeCtrl?

Make a real model and refill (or modify) TreeCtrl as required...

Quote:
And, a bonus question... (though for different widgets)
Since I'm taking shots a MVC separation anyway, I'm finding a similar situation with U++ nifty EditField derived widgets, which have all of the range/notNull checking, etc. that I want on to protect my Model object fields from getting set incorrectly, from the GUI or otherwise. If I want to keep, say, an Int datamember in one of my objects AND I want the user to manipulate it, why wouldn't I declare that datamember as EditSpin, instead of Int? Then when I want to present it to the user for editing, I simply have it draw itself?
Are there major performance/memory issues with this behavior?

I am not quite sure I understand the question. Anyway, maybe the answer is that U++ is desgined in a way that you usually do not need have "variable-widget" pairs (using variable to store the data, only use widget for GUI interaction). You do not need the variable, you can store the data directly in widget. You can even think about widgets as "value with possible GUI editing"...

I guess this also replies your "second TreeCtrl" question.

In short, using U++ widgets for other purposes is OK

Mirek

Subject: Re: hierarchical tree data structure & binding to TreeCtrl
Posted by Novo on Mon, 12 Nov 2007 19:56:05 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 11 November 2007 17:38I am not quite sure I understand the question. Anyway, maybe the answer is that U++ is desgined in a way that you usually do not need have "variable-widget" pairs (using variable to store the data, only use widget for GUI interaction). You do not need the variable, you can store the data directly in widget. You can even think about widgets as "value with possible GUI editing"...

Let say I have a database table with 100K records (quite small) and I'd like to show it in several widgets (ArrayCtrl). Duplicating of data would require huge amount of memory, and I'll need to synchronize data in several widgets in case of data change. Fortunately there is a virtual mode of ArrayCtrl. Having such a virtual mode in the TreeCtrl would be very handy (isn't it a model-view?).

I think that separation of concepts (like graphical data representation and data itself) is very useful. Developing something more or less complicated without that is hardly possible.

Making clear separation of concepts requires a lot of experience in software design. That is not easy. But using good designed software is real fun !

Subject: Re: hierarchical tree data structure & binding to TreeCtrl
Posted by solareclectic on Thu, 15 Nov 2007 02:02:23 GMT
View Forum Message <> Reply to Message

In short, thanks! That is just what I suspected.
shannon

Subject: Re: hierarchical tree data structure & binding to TreeCtrl
Posted by mirek on Thu, 15 Nov 2007 04:46:52 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 12 November 2007 14:56luzr wrote on Sun, 11 November 2007 17:38I am not quite sure I understand the question. Anyway, maybe the answer is that U++ is desgined in a way that you usually do not need have "variable-widget" pairs (using variable to store the data, only use widget for GUI interaction). You do not need the variable, you can store the data directly in widget. You can even think about widgets as "value with possible GUI editing"...

I think that separation of concepts (like graphical data representation and data itself) is very useful. Developing something more or less complicated without that is hardly possible.

Hey, only because you CAN do something does not mean you HAVE to do... There obviously are

scenarios where it is better not to store data in the widget:)

(This is the same as some people insisting that you cannot allocate U++ widgets on the heap. Of course you can, but unlike other toolkits, you are not required to).

Quote:
Making clear separation of concepts requires a lot of experience in software design. That is not easy. But using good designed software is real fun !

Well, I just think that sometimes, people go over the roof here. Obviously, in some cases the separation is the right thing to do, but to do it always only leads to added complexity without any real benefit to the program reliability or user experience.

Mirek

Subject: Re: hierarchical tree data structure & binding to TreeCtrl
Posted by unodgs on Thu, 15 Nov 2007 07:27:14 GMT
View Forum Message <> Reply to Message

Quote:Well, I just think that sometimes, people go over the roof here. Obviously, in some cases the separation is the right thing to do, but to do it always only leads to added complexity without any real benefit to the program reliability or user experience.
Amen

PS: That should be my 666 mesage on this forum..

Subject: Re: hierarchical tree data structure & binding to TreeCtrl
Posted by Novo on Mon, 19 Nov 2007 14:10:37 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 14 November 2007 23:46
Hey, only because you CAN do something does not mean you HAVE to do... There obviously are scenarios where it is better not to store data in the widget:)

(This is the same as some people insisting that you cannot allocate U++ widgets on the heap. Of course you can, but unlike other toolkits, you are not required to).

I think that is a little bit different. If you cannot allocate U++ widgets on the heap, that is a design decision, which won't affect performance.

If you have to allocate hundreds megabytes of RAM every time you want to show your data, that is different.

If a widget (I mean ArrayCtrl) has an API to retrieve data, one can implement an algorithm, which won't keep all data in memory. That will let to display data tables with billions of records without significant memory impact. And memory allocation/deallocation never was fast (probably not in U++). And data sorting can be moved out of the widget. IMHO it should be moved out because there are too many ways to do that.

The same with trees, although they usually much smaller than tables. But I can imagine situation when one wants to display whole gene ontology in a tree widget. And it is vvvvveryyyyyy big.

May be I'm wrong, but ...

And I do not want to say that data containers, which are built into widgets, is a bad idea. In 95% of all cases they are the best solution.

Subject: Re: hierarchical tree data structure & binding to TreeCtrl
Posted by unodgs on Mon, 19 Nov 2007 14:23:29 GMT
View Forum Message <> Reply to Message

I agree that TreeCtrl should support "virtual leafs". This and all kinds of grids are the most often used widgets to display large amount of data. That remainds me to add virtual mode to GridCtrl . Anyway this work mode can be problematic. For example - it's hard to implement "virtual rows" with resizeable heights. I tested many grids and everywhere row has fixed height . It's also hard to estimate scrollbar height. It must be updated durign fetching rows from database (as we don't know in most cases how many rows are there to fetch). I sometimes think it's better to used paging or better search criteria than virtual mode.

Subject: Re: hierarchical tree data structure & binding to TreeCtrl
Posted by Novo on Mon, 19 Nov 2007 14:34:13 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 14 November 2007 23:46
Quote:
Making clear separation of concepts requires a lot of experience in software design. That is not easy. But using good designed software is real fun !

Well, I just think that sometimes, people go over the roof here. Obviously, in some cases the separation is the right thing to do, but to do it always only leads to added complexity without any real benefit to the program reliability or user experience.

Design is more like religion. It is hard to explain why one is better than another. People usually choose one and follow it whole life ...

Let's start typing till the dead end / grave

I do not want to start another holly war here ...
