Subject: Thoughts about resource management
Posted by Mindtraveller on Sun, 23 Jan 2011 15:28:11 GMT
View Forum Message <> Reply to Message

You of course know "everything belongs somewhere" approach. If you tried it, you most probably know it as simple and effective way of getting rid of resource management problems. Simply speaking, you equalize visibility of object and period of its life by defining resource as stack object of parent class.
This works good for simple cases where resource lives constantly, starting with constructor of its parent until parent destructor is called.
But sometimes we meet more complex scenario, when resource is destroyed and re-created many times while program is working. We of course have Ptr/Pte wrappers. And I use them havily in these cases. Of course we may use public parent member functions which manage this resource for it not to violate our general approach.
But in my opinion we should discuss "everything belongs somewhere" for complex scenarios and widen U++ manual a little. Because complex scenarios is where smart pointers live, and we have to avoid mixing them with "U++ style".

Subject: Re: Thoughts about resource management
Posted by mirek on Sun, 23 Jan 2011 22:48:28 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Sun, 23 January 2011 10:28You of course know "everything belongs somewhere" approach. If you tried it, you most probably know it as simple and effective way of getting rid of resource management problems. Simply speaking, you equalize visibility of object and period of its life by defining resource as stack object of parent class.
This works good for simple cases where resource lives constantly, starting with constructor of its parent until parent destructor is called.
But sometimes we meet more complex scenario, when resource is destroyed and re-created many times while program is working. We of course have Ptr/Pte wrappers. And I use them havily in these cases. Of course we may use public parent member functions which manage this resource for it not to violate our general approach.
But in my opinion we should discuss "everything belongs somewhere" for complex scenarios and widen U++ manual a little. Because complex scenarios is where smart pointers live, and we have to avoid mixing them with "U++ style".

I do not know, well, maybe it is I am far used and invested to U++ approach, but I believe those complex scenarios are simply the result of "not trying hard enough" to avoid such situations.

Sometimes it took me time and effort to rethink the whole design to fit U++, but in the end it always ended simpler and faster than using shared ownership.

But I agree it is not always that obvious...

Subject: Re: Thoughts about resource management
Posted by fudadmin on Mon, 24 Jan 2011 13:34:27 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Sun, 23 January 2011 10:28You of course know "everything belongs somewhere" approach. If you tried it, you most probably know it as simple and effective way of getting rid of resource management problems. Simply speaking, you equalize visibility of object and period of its life by defining resource as stack object of parent class.
This works good for simple cases where resource lives constantly, starting with constructor of its parent until parent destructor is called.
But sometimes we meet more complex scenario, when resource is destroyed and re-created many times while program is working. We of course have Ptr/Pte wrappers. And I use them havily in these cases. Of course we may use public parent member functions which manage this resource for it not to violate our general approach.
But in my opinion we should discuss "everything belongs somewhere" for complex scenarios and widen U++ manual a little. Because complex scenarios is where smart pointers live, and we have to avoid mixing them with "U++ style".

Have you read this topic ?

Subject: Re: Thoughts about resource management
Posted by Mindtraveller on Mon, 24 Jan 2011 17:14:39 GMT
View Forum Message <> Reply to Message

Thank you very much for this link. It mostly discusses the choice between refcount and pick_ approaches, which is a very interesting discussion itself. But my question was slightly different.
I was wondering if U++ approach is good for resources with dynamic lifetime.

Subject: Re: Thoughts about resource management
Posted by fudadmin on Mon, 24 Jan 2011 17:34:12 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Mon, 24 January 2011 17:14
I was wondering if U++ approach is good for resources with dynamic lifetime.

Yes, I knew it was not the exact answer. Have you studied DynamicDlg
Also, how keyboard shortcuts and color settings for TheIde are implemented?
Also, maybe, Qtf and RichEdit (table, in particular)?
Maybe to start a topic with a concrete example?

Subject: Re: Thoughts about resource management
Posted by mirek on Mon, 24 Jan 2011 23:39:28 GMT

Mindtraveller wrote on Mon, 24 January 2011 12:14Thank you very much for this link. It mostly discusses the choice between refcount and pick_ approaches, which is a very interesting discussion itself. But my question was slightly different.
I was wondering if U++ approach is good for resources with dynamic lifetime.

Well, frankly, I do not know....

Anyway, in theory, using heap is equivalent to using some global scope container. So perhaps in this sense, you could say 'it belongs to global scope'

Now that is hardly useful proof. Anyway, I believe you can always benefit to move this 'global scope container' down to some more fitting scope....