Subject: Re: Value: no deep copy support?
Posted by mirek on Thu, 21 Oct 2010 21:23:04 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 21 October 2010 14:17it was sort of retorical question anyway. RawValue / RichValue / ValueArray in any case do represent data 'container' and not only the information it has. and as a container one wants to have access modify access to the container.. (int 123 is information, int& 123 references a *container* which happens to have 123 as information).
it's difficult to explain actually.

actually, i dont see much of the point to have Value reference anything else than the intrinsic datatypes, if they are immutable. maybe i need some time to wrap my mind around Value fully

maybe i need to think of it as a reference to a bit of very volatile readonly data..that is created somewhere in the dust, and if it ceases to be of importance it becomes dust again..
you actually never need to reference the same data packet inside to rely on its information. if its content (or better the information it represents) is about to be altered, this info is created. so Value is not about the place to store sth. but about to represent some information, speeded up by ref count. sorry, trying to understand by describing for the purpose of referencing a container maybe one should use Ptr<T>..

you asked about the const thing:

http://www.ultimatepp.org/srcdoc$Core$UserValue$en-us.html
Quote:
Note that the function ValueTo returns a constant reference. This is consistent with default Value behaviour, according to which data once put into Value should never change afterwards. This is necessary because, upon copying, multiple variables of type Value can hold the same data "packet" and changing its contents would affect all these copies, which is normally undesirable. If you break the const-ness using a mutable member or by a const_cast, you should not forget that.

this actually adds to understand Value as a container for sth. ("packet", "contents").

Sorry, but what I read in that paragraph is "do not do this. If you insist to do, be prepared to shot your leg off". And it is more about mutable members used to caching information than anything else.

Quote:
BTW: maybe you could help me with the ValueArray understanding in another thread here in Core++

I guess you are using bad approach here. Why do not you just check for what Value, ValueArray and ValueMap are actually used?

Here are some good examples:

http://www.ultimatepp.org/reference$ArrayCtrl$en-us.html
http://www.ultimatepp.org/reference$Display$en-us.html
http://www.ultimatepp.org/reference$SQL_Sqlite3$en-us.html
http://www.ultimatepp.org/reference$Switch$en-us.html
http://www.ultimatepp.org/reference$Chameleon$en-us.html