Subject: IsNullInstance() for Callback
Posted by hojtsy on Fri, 10 Mar 2006 23:48:16 GMT
View Forum Message <> Reply to Message

I think it would be good to check if a given Callback is Null or not. This would need a IsNullInstance() method for Callback.

Subject: Re: IsNullInstance() for Callback
Posted by mirek on Sat, 11 Mar 2006 06:35:34 GMT
View Forum Message <> Reply to Message

hojtsy wrote on Fri, 10 March 2006 18:48I think it would be good to check if a given Callback is Null or not. This would need a IsNullInstance() method for Callback.

I am not sure. To test whether Callback is empty or not, you can call operator bool() - just like with function pointer.

IsNull would imply that Callback is sort of value (or Value..). That is something that we had decided to avoid lately. The trouble with this is that for Value, you expect comparison, hash-value etc. But for

Callback1<Foo>

that creates the requirement for Foo comparison and GetHashValue - something not very nice in most cases.

MIrek

Subject: Re: IsNullInstance() for Callback
Posted by hojtsy on Sat, 11 Mar 2006 16:02:38 GMT
View Forum Message <> Reply to Message

How about adding a IsValid() method which would return the same value as operator bool(). I like producing self-describing code, but using this bool operator needs more code comments.

Subject: Re: IsNullInstance() for Callback
Posted by mirek on Sat, 11 Mar 2006 17:46:43 GMT
View Forum Message <> Reply to Message

hojtsy wrote on Sat, 11 March 2006 11:02How about adding a IsValid() method which would return the same value as operator bool(). I like producing self-describing code, but using this bool

operator needs more code comments.

Hm, would you comment

void (*f)() = NULL;
.....

if(f) {
}

?

Mirek

Subject: Re: IsNullInstance() for Callback
Posted by hojtsy on Sat, 11 Mar 2006 19:54:48 GMT
View Forum Message <> Reply to Message

I see the analogy. But I would write that code differently.typedef void (*VoidFunctionPointer)();
VoidFunctionPointer operationPointer = NULL;
.....
if(NULL == operationPointer) {
}This version is longer and but readers have a chance of understanding what the if() does withouth seeing the definition of the pointer variable, which could be pages away or in a different file. With this descriptive coding style code comments could be shorter, but yes I would put comments around any function pointer usage if the code is to be read by anyone else than myself. This area is not that well understood by the avarage programmers.
Now let's see how can I turn back this analogy to Callback. The best version I could come up with is:Callback operationCallback;
.....
if((bool) operationCallback) {
I know that (bool) is unnecessary, but implicit conversions just make the situation less understandable for average programmers. With seeing just the if line some of them will start to think that operationCallback is bool, or maybe int. This way I can convey the info that operationCallback is not bool, and they should go and check it's type. But it is still quite cryptic. The best notation would be something which could be read aloud as it's own explanation. Such notation is:
if(operationCallback.IsValid()) {
Now try to read out aloud this one, and then the one without IsValid.
Adding this one method will only have minimal effect on the readablity of a big application, so its not that important. I just wanted to explain my point of view.
