Subject: ~Link
Posted by masu on Wed, 23 Jan 2013 11:26:04 GMT
View Forum Message <> Reply to Message

Hi,

I made a little test with Link utility class and heap leaks have been detected when I use the following code:

```
struct LinkTest : Link<LinkTest> {
};

CONSOLE_APP_MAIN
{
 LinkTest lt;
 lt.InsertPrev();
 lt.InsertNext();
 lt.Dump();
}
```

This comes from the fact that the whole list is not destroyed when the destructor of lt is called at the end of the programm, but instead only lt is removed and the other heap allocated members in the list remain.

I propose to call DeleteList() in ~Link instead of calling UnLinkAll(). If one wants to unlink a single object obj within the list an obj.UnlinkAll() can be used.

It just does not feel the U++ way to me to have to call lt.DeleteList() at the end of the program.

Regards
Matthias

Subject: Re: ~Link
Posted by mirek on Thu, 24 Jan 2013 18:52:01 GMT
View Forum Message <> Reply to Message

masu wrote on Wed, 23 January 2013 06:26Hi,

I made a little test with Link utility class and heap leaks have been detected when I use the following code:

```
struct LinkTest : Link<LinkTest> {
};

CONSOLE_APP_MAIN
{
 LinkTest lt;
 lt.InsertPrev();
```

```
 lt.InsertNext();
 lt.Dump();
}
```
This comes from the fact that the whole list is not destroyed when the destructor of lt is called at the end of the programm, but instead only lt is removed and the other heap allocated members in the list remain.

I propose to call DeleteList() in ~Link instead of calling UnLinkAll(). If one wants to unlink a single object obj within the list an obj.UnlinkAll() can be used.

It just does not feel the U++ way to me to have to call lt.DeleteList() at the end of the program.

Regards
Matthias

Uhm, that would be very bad idea indeed.

Link is only just simple utility class used mostly in U++ libraries. It is not a container and in fact, 99% of uses are NOT allocating nodes on heap, but are linking nodes that exist independently (that is the original purpose of Link). E.g. see DisplayPopup. All that code would be broken - and we would need another class for the original purpose.

InsertAfter/InsertBefore are utility methods for very special uses. Anyway, if you really want to use Link the way you want, you can use LinkOwner.

Mirek

Subject: Re: ~Link
Posted by masu on Thu, 24 Jan 2013 22:27:18 GMT
View Forum Message <> Reply to Message

Thanks, Mirek.
I was not aware of LinkOwner, but it is exactly what I need.

Matthias
