Subject: Bug in sigleton classes in fairbase/base
Posted by Bertram Kopf on Tue, 27 Oct 2009 13:11:47 GMT
View Forum Message <> Reply to Message

Hi,
in the "base" directory there is the attempt to provide a couple of singleton classes (e.g. FairRootManager, FairRun, FairRunSim, etc.). Due to the fact that the relevant constructors are defined as public, these singletons are not properly implemented. For those classes, it is possible to create more than only one object and this can therefore cause some troubles. For singletons, it is a must to define the constructors in the protected or private region.
Therefore I would like to ask the relevant developers to fix this bug as soon as possible.

Thanks in advance,
Bertram.

Subject: Re: Bug in sigleton classes in fairbase/base
Posted by Mohammad Al-Turany on Tue, 27 Oct 2009 13:32:38 GMT
View Forum Message <> Reply to Message

Hi Bertram,

Thanks for having time to look into this! I know that in the design pattern book, there is this nice statement that the ctor of a singleton should be privet, and it is better to put it privet I agree. but I do not agree that you can create these classes more than once, there is protection in the ctor which prevent this, so I am wondering how could you create any of these more than once! did you try it?

the only reason that we kept this attempt to write singletons as it is that if we put the ctor as privet then you have to call always the instance so nobody can call new FairRun for example directly and this to be changed in all macros. Anyway I agree again that in the holy book of design pattern it is written that it should be privet but I do not see any problem of the protection we have and the main thing is to have a singleton that cannot be crated more than once and this is the case now. So when I will have time I will change it but now it has absolutely no priority to do this.

regards

Mohammad

Subject: Re: Bug in sigleton classes in fairbase/base
Posted by Bertram Kopf on Tue, 27 Oct 2009 14:09:52 GMT
View Forum Message <> Reply to Message

Hi Mohammad,
Mohammad Al-Turany wrote
... but I do not agree that you can create these classes more than once, there is protection in the ctor which prevent this, so I am wondering how could you create any of these more than once! did you try it?

I didn't look closer into the ctor. You are right, there is a protection against it. If you try to create a second object , then the application will crash.
But if you would like to call the function "Instance()" without creating the object before, one gets back 0 pointer. And this definitely is not the idea of a singleton.

Mohammad Al-Turany wrote
the only reason that we kept this attempt to write singletons as it is that if we put the ctor as privet then you have to call always the instance so nobody can call new FairRun for example directly ...
Mohammad

But exactly this should be avoided in a singleton. The only way to get access to a singleton object should be possible via a static instance function. The singleton itself has to take care of calling its own ctor.
Therefore I would prefer to follow the recommendations of object oriented design patterns.

Cheers,
Bertram.

Subject: Re: Bug in sigleton classes in fairbase/base
Posted by Mohammad Al-Turany on Tue, 27 Oct 2009 14:27:50 GMT
View Forum Message <> Reply to Message

Hi Bertram,

As I said before, it is nicer to have the ctor privet, but it is not a really a bug that change behavior of the singleton. in the compiled code you always use the instance and this is working, all these singleton classes are some kind of managers that have to be created from the macro (RunAna and RunSim) or by the framework itself (IO manager) and then the user uses the instance.

Quote:But exactly this should be avoided in a singleton. The only way to get access to a singleton object should be possible via a static instance function. The singleton itself has to take care of calling its own ctor.
Therefore I would prefer to follow the recommendations of object oriented design patterns.

The design pattern book is simply some recommendations for implementations, and does not present the object oriented standard, even C++ itself is not 100% object oriented (and this make it usable in contrary to JAVA or smaltalk) . Anyway when I have time I will change it, but not now.

regards

Mohammad
