
Subject: Re: How to use callback with 3 parameters

Posted by [mirek](#) on Fri, 11 Jan 2008 10:11:18 GMT

[View Forum Message](#) <> [Reply to Message](#)

malya wrote on Fri, 11 January 2008 03:47Please post me a small example.
Thanks!

Suprisingly, callback3 was somehow forgotten, while callback4 is there...

Well, it is now fixed. Quick fix

```
template <class OBJECT_, class METHOD_, class T1, class T2, class T3>
struct CallbackMethodActionArg3Pte : public CallbackAction {
    Ptr<OBJECT_> object;
    METHOD_ method;
    T1 arg1;
    T2 arg2;
    T3 arg3;
    void Execute() { if(object) (object->*method)(arg1, arg2, arg3); }
```

```
    CallbackMethodActionArg3Pte(OBJECT_ *object, METHOD_ method, T1 arg1, T2 arg2, T3 arg3)
    : object(object), method(method), arg1(arg1), arg2(arg2), arg3(arg3) {}
};
```

```
template <class Object, class R, class O, class A, class B, class C, class T1, class T2, class T3>
Callback pteback3(Object *object, R (O::*method)(A, B, C), T1 arg1, T2 arg2, T3 arg3) {
    return Callback(new CallbackMethodActionArg3Pte<Object, R (O::*)(A,B,C), T1, T2, T3>
        (object, method, arg1, arg2, arg3));
}
```

```
template <class OBJECT_, class METHOD_, class T1, class T2, class T3>
struct CallbackMethodActionArg3 : public CallbackAction
{
    OBJECT_ *object;
    METHOD_ method;
    T1 arg1;
    T2 arg2;
    T3 arg3;
```

```
    void Execute() { (object->*method)(arg1, arg2, arg3); }
```

```
    CallbackMethodActionArg3(OBJECT_ *object, METHOD_ method, T1 arg1, T2 arg2, T3 arg3)
    : object(object), method(method), arg1(arg1), arg2(arg2), arg3(arg3) {}
};
```

```
template <class Object, class R, class O, class A, class B, class C, class T1, class T2, class T3>
Callback callback3(Object *object, R (O::*method)(A, B, C), T1 arg1, T2 arg2, T3 arg3)
```

```
{
    return Callback(
        new CallbackMethodActionArg3<Object, R (O::*)(A, B, C), T1, T2, T3>(object, method, arg1,
        arg2, arg3));
}
```

```
template <class Object, class R, class O, class A, class B, class C, class T1, class T2, class T3>
Callback callback3(const Object *object, R (O::*)(A, B, C) const, T1 arg1, T2 arg2, T3
arg3) {
    return Callback(new CallbackMethodActionArg4<Object, R (O::*)(A, B, C) const, T1, T2, T3>
        (object, method, arg1, arg2, arg3));
}
```

```
template <class X, class T1, class T2, class T3, class HC = X>
struct CallbackActionCallArg3 : public CallbackAction {
    X      x;
    T1     arg1;
    T2     arg2;
    T3     arg3;
    void    Execute() { x(arg1, arg2, arg3); }
```

```
    CallbackActionCallArg3(X x, T1 arg1, T2 arg2, T3 arg3)
    : x(x), arg1(arg1), arg2(arg2), arg3(arg3) {}
};
```

```
template <class R, class A, class B, class C, class T1, class T2, class T3>
Callback callback3(R (*fn)(A, B, C), T1 arg1, T2 arg2, T3 arg3) {
    return Callback(
        new CallbackActionCallArg3<R (*)(A, B, C), T1, T2, T3, uintptr_t>(fn, arg1, arg2, arg3));
}
```

```
template <class A, class B, class C, class T1, class T2, class T3>
Callback callback3(Callback3<A, B, C> cb, T1 arg1, T2 arg2, T3 arg3) {
    return Callback(
        new CallbackActionCallArg3<Callback4<A, B,C,D>, T1, T2, T3>(cb, arg1, arg2, arg3));
}
```

add to Core/Callback.h.

Note: In fact, the reason why this was unnoticed until now is that for "higher order callbacks", it is usually better to pass the struct as single parameter. We were reluctant to extend the number of parameters beyond 2, which is the maximum that was ever needed for CtrlLib or any application we did.

Mirek