

Re: EnumChildWindows and C++

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.win32.programmer.ui/2005-06/msg00008.html>

- *From:* "Kevin Cochran" <KevinCochran@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 31 May 2005 21:20:02 -0700
-

Hi John,

Thanks for the useful information. After reading your post a couple of times, I started getting a grasp on it. :)

I checked out the book you referred me to, and it got really good reviews on Amazon. I have a good stack of reading material now.

Thanks again!
Kevin

"John Carson" wrote:

```
> "Kevin Cochran" <KevinCochran@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in
> message news:BF3E7350-25E8-4DD1-96BD-4C1C531B4580@xxxxxxxxxxxxxx
>> Greetings John,
>>
>> Yep, making it static worked. Could you recommend a resource that will
>> explain in depth how to use static and non-static member functions in
>> C++? I only recently began switching to C++ from C, so any good
>> resources would be appreciated.
>
> Any C++ textbook should do it. A good one that is available both as a free
> download and as a non-free printed book is Bruce Eckel: Thinking in C++.
>
> http://mindview.net/Books/TICPP/ThinkingInCPP2e.html
>
> The EnumChildWindows function is designed to take a C-style non-member
> callback function. Generally a *public* static member function can be used
> wherever a C-style non-member function can be used.
>
> Compared to a non-member function, a static member function has two
> properties:
>
> 1. It has access rights to public, protected and private members of the
> class to which it belongs in the same way that a non-member "friend"
> function does.
```

Re: EnumChildWindows and C++

> 2. It can itself be public, protected or private.
>
> What distinguishes a static member from a non-static member is that a static
> member is not tied to an object of the class. To illustrate, suppose we have
> a class X. We may define an object of class X as:
>
> X x;
>
> We can then call an ordinary member function:
>
> x.ordinary_member();
>
> ordinary_member() has an implicit argument, which is the "this" pointer of
> x. The effect of this is that x's data members are "in scope" within the
> definition of ordinary_member.
>
> We can also call
>
> x.static_member();
>
> but static_member() does not have an implicit "this" pointer argument (this
> absence of a hidden argument makes it like a non-member function, which is
> why it is can be used in place of a non-member function in most contexts).
> For static_member() to be able to refer to x's data members, it must take a
> pointer/reference to x as an argument, e.g.,
>
> x.static_member(&x);
>
> Again, this is what you would do with a non-member function.
>
> A static member can also be called with an alternative syntax:
>
> X::static_member();
>
> This means that, as with a non-member function, you can call a static member
> without having an object of the class to bind it to (you can of course only
> call it if the static member function is public or it is being called from a
> function with the necessary access rights --- e.g., a friend function).
>
>
> ---
> John Carson
>
>
>

-
- Prev by Date: [*Re: inhibiting the dotted focus box*](#)
 - Next by Date: [*How to get selected files Filename and full path?*](#)
 - Previous by thread: [*RE: Ownerdraw tabitem no longer works when run from 64-bit Windows*](#)

Re: EnumChildWindows and C++

- Next by thread: *How to get selected files Filename and full path?*
- Index(es):
 - ◆ *Date*
 - ◆ *Thread*