

Re: Exception Handling – help!

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.clr/2004-11/0017.html>

From: David Levine (*noSpamdlevineNNTP2_at_wi.rr.com*)

Date: 11/03/04

Date: Wed, 3 Nov 2004 05:24:29 -0600

>> *and it does not support calling
>> methods, only evaluating simple expressions. To call a user-filter you
>> would
>> have to write the code directly in IL.*
>
> *I don't agree :-)* I tried out examples where the "when" clause of a catch
> contains a method call. Something like:
>

Thanks for the info. I was under the impression (incorrect, obviously) that it did not. I don't use VB so I got my info from reading about it, not actual experimentation.

>
>> *Since the stack has not yet been unwound then any methods
>> invoked during the user filter would have to be on top of the stack,
>> which
>> implies that those routines also have access to the stack frame in which
>> the
>> original exception occurred*
>
> *are you sure these routines have access to the faulting frame? I think
> rather not. They are in completely different frames.*
>

I did not mean that you necessarily had access to them within those languages, but that since it was executing on top of those stack frames then low-level tools could access them. For example, the filter could call a low-level tool that could walk the stack, setup a frame, extract the parameters passed to a method, and log them. Not for the weak of heart, but possible.

>> *If another exception occurs within a user filter then the runtime would
>> have
>> to treat this as a nested exception.*
>

- > *I tried examples in VB and I guess they do the following:*
- >
- > *if an exception occurs in a filter (i.e. in pass 1), then it starts a new*
- > *exception propagation. It looks for a suitable handler. Upon propagation,*
- > *if*
- > *this exception reaches our filter, then the 1st pass of this second*
- > *exception*
- > *finishes – further the second pass is executed. Finally, our filter is*
- > *considered as it would have returned 0 (i.e. "continue search"). Please*
- > *someone contradict me if I'm wrong!!*
- >
- > *In this situation, we return to the 1st pass of the initial exception,*
- > *looking for another catch or filter.*
- >

Are you saying that if the 2nd exception's stack walk reaches the original filter it terminates the that stack walk and treats that as if the filter had returned ExecuteHandler? ...interesting. I should install VB and try it to verify actual behavior. Have you tried this with multiple nested exceptions? I would really like someone from MSFT to describe how this is implemented.