Base Language - Bug #5901

at least some 4GL warnings need to process differently when the current scope has a catch block

12/15/2021 03:49 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#2 - 12/15/2021 03:58 PM - Greg Shah

In #5765-41, Ovidiu was testing the find-* methods of buffer. He found this:

> It looks like there are 4 cases, depending on the presence of the no-error option and catch statements:

	no option	no-error
simple	Error messages displayed Method returns no Procedure continues	No error displayed Method returns no Procedure continues
catch statement	No error displayed Method does not return Procedure **abends**	No error displayed Method returns no Procedure continues

> Question: are there cases when some (error) message is simply displayed and not caught by the catch statement? The answer is yes. The Error and SysError types will catch the event. Switching to AppError will cause the catch statement to be ignored.
>
> I think the issue might be fixed in ErrorManager. If the procedure/block has a catch block, regardless if the filter matched the current event, the recordOrShowError() APIs should act as a throw and abend processing of the current procedure. The method being executed (in this case find-first) is not aware of the active catch statements, so it must just notify the ErrorManager and let it decide in which of the four (actually only 3 because if no-error is present, the outcome is the same regardless of the catch statement) cases from the table we're in and act accordingly. It still need to make distinction when the error message is just displayed in simple case (for methods) and when the procedure is abended (for statements). For example, find-first method as opposed to find first statement.

In #5765-42, Marian pointed out:

> I think this case falls under the handle method error handling - https://docs.progress.com/bundle/openedge-abl-reference-117/page/Error-handling-for-handle-method-calls.html
>
> This is what I see in #5308 as well when setting the format attribute to an invalid value, there instead of recordOrShowError the recordOrThrowError is used instead... none of those approaches are actually right :(
>
> What is needed there is some kind of 'warning' method in ErrorManager that needs to implement the same logic as outlined in PSC documentation, did made a quick test procedure in testcases repository - ui/error-handling/FillInInvalidFormat.w
> 1. for direct assignment, without no-error nor catch block the messages are being displayed directly
> 2. when a catch block is used (one that can catch SysError) one warning message is being shown 4058 and the 148 is being catch
> 3. when using a no-error option then both messages are being hidden and recorded into the error-status system handle but as warning only (error-status:error is false)

And I noted in #5765-43:

The ErrorManager.recordOrShowError() is meant to be that warning method. But we did not know about the catch block behavior, so we didn't implement it.

The dangerous part here is that ErrorManager.recordOrShowError() is used from a very large number of places (900+ locations in 98 files). To the degree that those other locations don't have this catch block behavior, we can introduce regressions. For example, we use these this in the int64(date) constructor. This is not a handle method case but it very well could have the same behavior. Without testing these, we can't know. All we know is that our previous non-catch block testing showed a "warning" instead of an error. We've seen that a lot.

I guess the question here is: how likely is it that these other warning cases also work the same way when a catch block is present?

The intention of this task is to:

1. Determine if it is safe to modify ErrorManager.recordOrShowError() to add the behavior for catch.
2. If it is safe, go ahead and implement it.
3. If it is not safe, then add a conditional path to ErrorManager.recordOrShowError() that can be enabled in those places where it is known to be safe. This is not optimal but is a safe fallback plan in case we can't make the unconditional change.

#3 - 12/15/2021 04:45 PM - Ovidiu Maxiniuc

A note on catch: if the exception thrown does not match the one caught by the statement, the error is "swallowed". That is, the effects from table from note-2 still occur, but no special code is executed. For example, the procedure containing the following code:

```
res = bh:find-first("bad predicate").
message res.

catch e as Progress.Lang.AppError:
   message "Caught an application error.".
end.
```

will abend without printing anything. I would expect the catch to NOT interfere with normal return of no value of the find-first method in this case, since that is generated by an unrelated (sibling) SysError.

Having an un-matching error, my logical reasoning was it would completely ignore the catch statement. Yet, they have chosen to implement otherwise. I think it is a good thing, though, because it seems to me to be easier to implement. The ErrorManager does not need to know in advance the exact error which will be thrown. If a catch statement is present in procedure, it should switch to show-errors-as-errors2 mode and stop execution of the procedure on all recordOrShowError calls from (all?) methods. At this moment, if a matching error if identified, the associated catch block is executed. Otherwise the procedure is abended.