
記事

[Toshihiko Minamoto](#) · 2021年4月22日 11m read

ObjectScript の信頼性の高いエラー処理機能とクリーンアップ機能 はじめに (および本記事を書いた動機)

ObjectScript コードのユニット (ClassMethod など) を実行する場合、そのスコープ外にあるシステムの諸部分と対話するときに適切なクリーンアップを行えないことが原因で、様々な予期せぬ副作用が発生することがあります。以下にその一部を紹介します。

- トランザクション
- ロック
- I/O デバイス
- SQL のカーソル
- システムフラグと設定
- \$Namespace
- 一時ファイル

ObjectScript のこういった重要な機能を、クリーンアップのコーディングや防御的なコーディングを適切に行わずに使用すると、普段は正常に動作しても、予期せぬかたちで、またデバッグが困難なかたちで失敗し得るアプリケーションができてしまう可能性があります。想定できるすべてのエラーケースにおいてクリーンアップコードが正常に動作することは、極めて重要です。表面的なテストではエラーを見落とす可能性が高いことを考えるとなおさらです。

この記事では、既知の落とし穴をいくつかご紹介し、信頼性の高いエラー処理とクリーンアップを実現するための2種類の対処法について説明いたします。

確実にすべてのエッジケースをテストしたい方は、私が Open Exchange に掲載している [Test Coverage Tool](#) をご覧ください！

注記: この記事は私が元々 2018 年 6 月 に InterSystems 社内で掲載したものです。
開発者コミュニティに投稿しようと思って To-Do リストに加えてから、もう 1 年半になります。
[ありきたりな言い訳で恐縮です。。。。](#)

避けたい落とし穴

トランザクション

トランザクションを自然にかつシンプルに処理する方法として、以下のように try/catch ブロックを使い、catch の中で TRollback を使います。

```
Try {
    TSTART
    // ... ???????????? ...
    TCOMMIT
} Catch e {
    TROLLBACK
    // e.AsStatus(), e.Log(), etc.
}
```

ここだけを見た場合、エラーが発生したときに TStart と TCommit の間のコードが早々と Quit を出すのではなく、エラーを投げしてくれることを考えると、このコードに問題はありません。しかし、このコードにはリスクがあります。理由は以下の 2 つです。

- もし他のデベロッパーが try ブロックに「Quit」を追加した場合、このトランザクションは開いた状態で放置される。そのような変更内容は、コードレビューの際につい見落としてしまうことがあるでしょう。現在のコンテキストにトランザクションが含まれていることが明らかでない場合は一層見落としやすくなります。
- このブロックのメソッドが外部のトランザクションの中から呼び出されることがあれば、トランザクションのすべてのレベルが TRollback によりロールバックされてしまう。

より好適なアプローチとしては、トランザクションのレベルをメソッドの初めから追跡し、最後にトランザクションのそのレベルにロールバックします。下の例をご覧ください。

```
Set tInitTLevel = $TLevel
Try {
    TSTART
    // ... ??????????????...
    // tStatus ??????????????????????????????????????
    If $$$ISERR(tStatus) {
        Quit
    }
    // ... ??????????????...
    TCOMMIT
} Catch e {
    // e.AsStatus(), e.Log(), etc.
}
While $TLevel > tInitTLevel {
    // ??????????????????????????????????????
    TROLLBACK 1
}
```

ロック

インクリメントロックを使用するコードでは、ロックが必要でなくなったら、クリーンアップコードの実行時にロックをデクリメントする必要があります。これをしないと、そのようなロックはプロセスが終了するまで保持されることになります。ロックがメソッドの外にリークすることはありませんが、そのようなロックを取得することがメソッドの副作用として確認されている場合は除きます。

I/O デバイス {#RobustErrorHandlingandCleanupinObjectScript-I/ODevices}

同じように、現在の I/O デバイス (特殊変数 \$io) の変更もメソッドの外にリークすることはありませんが、メソッドが現在のデバイスを変更することを目的としている場合は除きます (I/O リダイレクトを有効にするなど)。ファイルを操作するときは、OPEN / USE / READ / CLOSE を使うシーケンシャルファイルのダイレクト I/O よりも、%Stream パッケージを使用することをおすすめします。これ以外の場合で、I/O デバイスを使用する必要があるときは、メソッドの終わりにデバイスを元のデバイスに戻すよう注意が必要です。例えば、次のコードにはリスクがあります。

```
Method ReadFromDevice(pSomeOtherDevice As %String)
{
    Open pSomeOtherDevice:10
    Use pSomeOtherDevice
    Read x
    // ... x ??????????????????...
```

```
    Close pSomeOtherDevice  
}
```

pSomeOtherDevice がクローズする前に例外が投げられることがあれば、\$io は pSomeOtherDevice のままとなります。これにより、カスケードエラーが発生する可能性があります。

また、デバイスがクローズされると、\$io はプロセスのデフォルトデバイスにリセットされますが、メソッドが呼び出された前の同じデバイスにはリセットされない場合があります。

SQL のカーソル

カーソルベースの SQL を使用する場合にエラーが発生したら、カーソルをクローズする必要があります。

カーソルをクローズしなければ、リソースがリークする可能性があります ([ドキュメント](#)にその旨の記載あり)。

また、コードを再度実行して、カーソルをオープンしようとする、「already open」(既にオープン状態) エラー (SQLCODE -101) が発生する場合があります。

システムフラグと設定

アプリケーションコードがプロセスレベルのフラグやシステムレベルのフラグを変更する必要があるということは滅多にありません。例えば、これらの多くは [%SYSTEM.Process](#) と [%SYSTEM.SQL](#) に定義されています。その必要があるという場合は、初期値を保存してから、メソッドの終わりに再度保存しなおす必要があるので注意が必要です。

```
$Namespace {#RobustErrorHandlingandCleanupinObjectScript-$Namespace}
```

ネームスペースの変更がメソッドのスコープ外にリークするのを防ぐために、ネームスペースを変更するコードは、常に新しい \$Namespace を最初に記述する必要があります。

一時ファイル

[%Library.File:TempFilename](#) などを使って一時ファイルを作成するアプリケーションコードでは、その一時ファイルが不要になれば、その時点で削除するよう心掛ける必要があります

(ちなみに、[%Library.File:TempFilename0](#)> を使うと、主に InterSystems IRIS では、実際にファイルが作成されます)。

お薦めの対処法: Try-Catch (-Finally) {#RobustErrorHandlingandCleanupinObjectScript-RecommendedPattern:Try-Catch(-Finally)}

多くのプログラミング言語には、try/catch

構造に「finally」ブロックを追加できるという機能が備わっています。「finally」ブロックは、try/catch 文が完了した後に、例外が発生したかどうかに関係なく実行されます。ObjectScript

にこの機能はありませんが、似たようなことができます。

その一般的なパターンは、以下のとおりです。上述した潜在的な多くの問題をご確認いただけます。

```
ClassMethod MyRobustMethod(pFile As %String = "C:\foo\bar.txt") As %Status  
{  
    Set tSC = $$$OK  
    Set tInitialTLevel = $TLevel  
    Set tMyGlobalLocked = 0  
    Set tDevice = $io  
    Set tFileOpen = 0  
    Set tCursorOpen = 0  
    Set tOldSystemFlagValue = ""
```

```

Try {
    // ???????????????5 ?????????????????
    Lock +^MyGlobal(42):5
    If '$Test {
        $$$ThrowStatus($$ERROR($$GeneralError,"Couldn't lock ^MyGlobal(42)."))
    }
    Set tMyGlobalLocked = 1

    // ???????
    Open pFile:"WNS":10
    If '$Test {
        $$$ThrowStatus($$ERROR($$GeneralError,"Couldn't open file "_pFile))
    }
    Set tFileOpen = 1

    // [ ??? MyCursor ?? ]
    &SQL(OPEN MyCursor)
    Set tCursorOpen = 1

    // ?????????????????
    Set tOldSystemFlagValue = $System.Process.SetZEOF(1)

    // ????????????...
    Use tFile

    TSTART

    // [ ... ?????????????????????????????????... ]

    // ??????

    TCOMMIT
} Catch e {
    Set tSC = e.AsStatus()
}

// Finally {

// ??????: ??????
If (tOldSystemFlagValue != "") {
    Do $System.Process.SetZEOF(tOldSystemFlagValue)
}

// ??????: ???
If tFileOpen {
    Close pFile
    // pFile ??????????CLOSE ????? $io ?????????????????????
    // ????????????????? $io ?????????????
    // ?????:
    Use tDevice
}

// ??????: ???
If tMyGlobalLocked {
    Lock -^MyGlobal(42)
}

// ??????: ??????
// ?????????????????????????????????????

```



```

        TROLLBACK 1
    }
} Catch e {
    Set tSC = $$$ADDSC(tSC,e.AsStatus())
}

Quit tSC
}
}

```

以下のクラスは、今お見せした登録クラスを使って、メソッドの最後で実行するクリーンアップを簡素化する方法を示すものです。

```

Class DC.Demo.Driver
{

ClassMethod Run()
{
    For tArgument = "good","bad" {
        Do ..LogState(tArgument,"before")
        Do ..DemoRobustMethod(tArgument)
        Do ..LogState(tArgument,"after")
    }
}

ClassMethod LogState(pArgument As %String, pWhen As %String)
{
    Write !,pWhen," calling DemoRobustMethod("_$$$QUOTE(pArgument)_"):"
    Write !,$c(9),"$Namespace=", $Namespace
    Write !,$c(9),"$TLevel=", $TLevel
    Write !,$c(9),"$System.Process.SetZEOF( )=", $System.Process.SetZEOF( )
}

ClassMethod DemoRobustMethod(pArgument As %String)
{
    Set tScopeManager = ##class(DC.Demo.ScopeManager).%New( )

    Set $Namespace = "%SYS"
    TSTART
    Do tScopeManager.SetZEOF(1)
    If (pArgument = "bad") {
        // ?????????????????????? tScopeManager ?????????????
        Quit
    }
    TCOMMIT
}
}
}

```

[#エラーハンドリング](#) [#ObjectScript](#) [#Caché](#) [#InterSystems IRIS](#)

ソースURL:

<https://jp.community.intersystems.com/post/objectscript-%E3%81%AE%E4%BF%A1%E9%A0%BC%E6%80%A7>

[%E3%81%AE%E9%AB%98%E3%81%84%E3%82%A8%E3%83%A9%E3%83%BC%E5%87%A6%E7%90%86%E6%A9%9F%E8%83%BD%E3%81%A8%E3%82%AF%E3%83%AA%E3%83%BC%E3%83%B3%E3%82%A2%E3%83%83%E3%83%97%E6%A9%9F%E8%83%BD](#)