記事
Toshihiko Minamoto · 2021年5月4日 2m read

ObjectScriptエラー処理のスニペット

ObjectScriptには、エラー（ステータスコード、例外、SQLCODEなど）を処理する方法が少なくとも3つあります。 ほとんどのシステムコードにはステータスが使用されていますが、例外は、いくつかの理由により、より簡単に処理することができます。
レガシーコードを使用している場合、さまざまな手法の変換にいくらか時間が掛かりますが、
参考として、次のスニペットをよく使用しています。 皆さんのお役にも立てればと思います。

<

```
pre>///SQLCODEのステータス:
set st = $$$ERROR($$$SQLError, SQLCODE, $g(%msg))  //埋め込みSQL
set st = $$$ERROR($$$SQLError, rs.%SQLCODE, $g(rs.%Message)) //動的SQL
///SQLCODEの例外:
throw ##class(%Exception.SQL).CreateFromSQLCODE(SQLCODE,%msg) //埋め込みSQL
throw ##class(%Exception.SQL).CreateFromSQLCODE(rs.%SQLCODE,rs.%Message) //動的SQL
throw:(SQLCODE'=0)&&(SQLCODE'=100) ##class(%Exception.SQL).
CreateFromSQLCODE(SQLCODE,%msg) //クエリに成功する場合、またはデータがない場合はスローしない
///ステータスの例外:
$$$ThrowOnError(st)
///例外のステータス:
set st = err.AsStatus()
///カスタムエラーステータスの作成:
set st = $$$ERROR($$$GeneralError,"Custom error message")
///カスタム例外をスロー:
$$$ThrowStatus($$$ERROR($$$GeneralError,"Custom error message"))
///ステータス付きのSOAPエラーの処理:
try {
  //SOAPリクエストコード
} Catch err {
  If err.Name["ZSOAP" {
  Set st = %objlasterror
  } Else {
  Set st = err.AsStatus()
  }
}
return st ///カスタム例外クラスを定義 Class App.Exceptions.SomeException Extends
%Exception.AbstractException
{
Method OnAsStatus() As %Status
{
  return $$$ERROR($$$GeneralError,"Custom error message")
}
}
///カスタム例外のスローとキャッチ
try {
  throw ##class(App.Exceptions.SomeException).%New()
} catch err {
  if err.%ClassName(1) = ##class(App.Exceptions.SomeException).%ClassName(1) {
  //この種の例外に特有の処理
  } }</pre></body></html>
```

#Code Snippet #エラーハンドリング #ObjectScript #ベストプラクティス #Caché #InterSystems IRIS

ソースURL:
https://jp.community.intersystems.com/post/objectscript%E3%82%A8%E3%83%A9%E3%83%BC%E5%87%A6%E7%90%86%E3%81%AE%E3%82%B9%E3%83%8B%E3%83%9A%E3%83%83%E3%83%88