# Utilizzo delle funzioni Try in Navision 2016

Le funzioni Try in C / AL consentono di gestire gli errori che si verificano nell'applicazione durante l'esecuzione del codice.

Ad esempio, con le funzioni try, è possibile fornire all'utente finale messaggi di errore più intuitivi rispetto a quelli lanciati dal sistema.

È possibile utilizzare le funzioni try per rilevare errori / eccezioni generati da Microsoft Dynamics NAV o eccezioni generate durante le operazioni di interoperabilità di .NET Framework.

Le funzioni Try rilevano errori simili a una chiamata di funzione Codeunit.Run condizionale, tranne con le seguenti differenze:

Le chiamate di funzione Try non richiedono il commit delle transazioni di scrittura nel database.  
Le modifiche al database apportate con una funzione try non vengono annullate.

**Creazione di una funzione Try**

Per creare una funzione try, aggiungi una funzione nel codice C / AL di un oggetto (come una codeunit) come di consueto, quindi imposta la **proprietà TryFunction Property** su **Sì** . Una funzione try ha le seguenti limitazioni:

Nelle codeunits di test e di upgrade, è possibile utilizzare solo una funzione try su un tipo di funzione normale, come definito dalla **proprietà FunctionType** (Test Codeunits) o **FunctionType** (Upgrade Codeunits).  
La funzione try non può avere un **valore restituito definito dall'utente** .  
**Comprensione del comportamento e dell'utilizzo della funzione Try**

Una funzione designata come funzione try ha un valore di ritorno booleano (vero o falso). Una funzione try ha la costruzione **OK: = MyTryFunction** .

Se una chiamata alla funzione try non utilizza il valore restituito, la funzione try funziona come una normale funzione e gli errori vengono esposti come al solito.

Se una chiamata alla funzione try utilizza il valore restituito in un'istruzione **OK: =** o un'istruzione condizionale **come IF-THEN** , vengono rilevati errori.

È possibile utilizzare la funzione **GETLASTERRORTEXT** per ottenere gli errori generati da Microsoft Dynamics NAV.

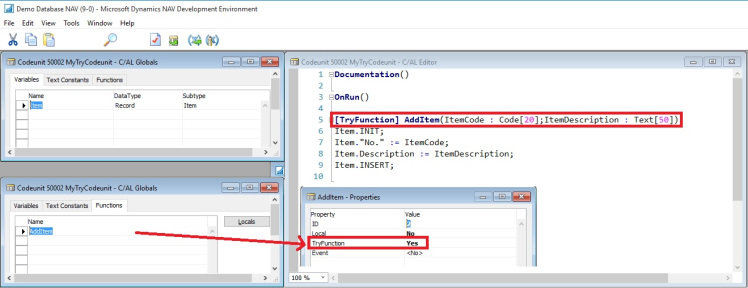
Per ottenere i dettagli delle eccezioni generate dagli oggetti .NET Framework, è possibile utilizzare la funzione **GETLASTERROROBJECT** per controllare la **proprietà Expection.InnerException** . È possibile visualizzare la codeunit **1291 - DotNet Exception Handler** .

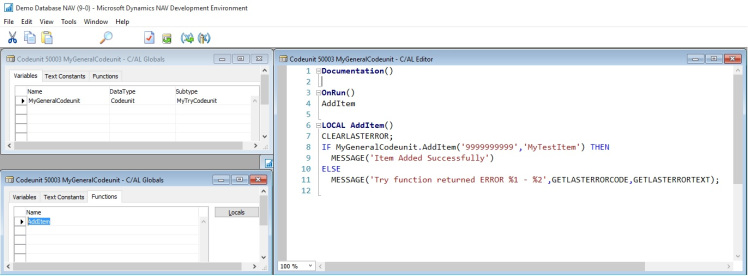
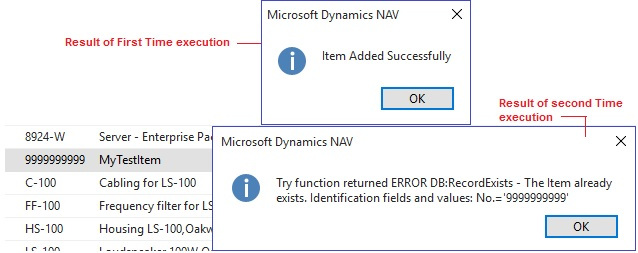
Se si chiama la funzione **GETLASTERRORTEXT** immediatamente dopo aver chiamato la funzione **CLEARLASTERROR** , viene restituita una stringa vuota.

Il risultato della funzione **GETLASTERRORCODE** non viene tradotto nella lingua locale. Il risultato della funzione **GETLASTERRORTEXT** viene tradotto nella lingua locale.

Il concetto di utilizzo di TryFunction è possibile trovare in tutto il sistema, in particolare per l' **anteprima di pubblicazione, l'applicazione di voci e la stampa di rapporti** ecc. In diverse unità di codice.

Tuttavia, per dare una semplice visione dell'uso dello stesso, sto usando il mio esempio definito di seguito.

Ho creato una codeunit con TryFunction per aggiungere un articolo alla tabella degli articoli.  


Ho creato un'altra codeunit con la funzione che chiama la funzione AddItem di codeunits sopra passando il parametro alla funzione. Testarlo se ha esito positivo o si è verificato un errore durante l'esecuzione dell'azione.   
  
Quando eseguo la seconda codeunit nella prima esecuzione, viene visualizzato il messaggio riuscito poiché l'elemento non esiste nella tabella. Quando eseguo per la seconda volta dà errore poiché l'elemento era già stato aggiunto alla tabella nella sua prima esecuzione.  


L'esempio sopra mostra anche l'utilizzo delle **funzioni ClearLastError, GetLastErrorCode e GetLastErrorText** .

Dedica del tempo alla scansione di altre codeunit per una migliore comprensione di TryFunctions.

È possibile vedere l'esempio fornito da Microsoft MSDN anche per comprendere l'utilizzo di TryFunction.

L'esempio seguente illustra l'utilizzo di una funzione try insieme alla **codeunit 1291 DotNet Exception Handler** per gestire le eccezioni di interoperabilità di .NET Framework. Il codice è in formato file di testo ed è stato semplificato per l'illustrazione. La funzione **CallTryPostingDotNet** esegue la funzione try **TryPostSomething** in un'istruzione condizionale per **rilevare le** eccezioni di interoperabilità di .NET Framework. Gli errori diversi dal tipo I **ndexOutOfRangeException** vengono rilanciati.

[TryFunction]

PROCEDURE TryPosting@1);

BEGIN

CODEUNIT.RUN(CODEUNIT::”Purch.-Post”);

END;

PROCEDURE CallTryPostingDotNet @2();

VAR

MyPostingCodeunit@1 : Codeunit 90;

MyDotNetExceptionHandler@2 : Codeunit 1291;

IndexOutOfRangeException@3 : DotNet ‘mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089′.System.IndexOutOfRangeException’

BEGIN

IF TryPostSomething THEN

MESSAGE(‘Posting succeeded.’)

ELSE BEGIN

MyDotNetExceptionHandler.Collect;

IF MyDotNetExceptionHandler.TryCastToType(IndexOutOfRangeException) THEN

MESSAGE(‘The index used to find the value was not valid.’)

ELSE

MyDotNetExceptionHandler.Rethrow;

END;

END;