Creating A Subroutine To Process All Components In An Assembly

Submitted by NXJournaling on Fri, 02/24/2012 - 07:58

Occasionally you will want to process all the parts in a given assembly. The journal in this tutorial will recursively step through your assembly structure allowing you to process each part. If the code is run as provided, it will open a listing window and print the name of each component and the current active arrangement if it is a subassembly. This journal was developed on NX 7.5 native; therefore if you are running Teamcenter (or other PLM) some changes will need to be made.

```
'Journal to recursively walk through the assembly structure
' will run on assemblies or piece parts
' will step through all components of the displayed part
'NX 7.5, native
'NXJournaling.com February 24, 2012

Option Strict Off

Imports System
Imports NXOpen
Imports NXOpen.UF
Imports NXOpen.Assemblies

Module NXJournal

    Public theSession As Session = Session.GetSession()
    Public ufs As UFSession = UFSession.GetUFSession()
    Public lw As ListingWindow = theSession.ListingWindow

    Sub Main()
    Dim workPart As Part = theSession.Parts.Work
    Dim dispPart As Part = theSession.Parts.Display

    lw.Open
    Try
        Dim c As ComponentAssembly = dispPart.ComponentAssembly
        'to process the work part rather than the display part,
        '  comment the previous line and uncomment the following line
        'Dim c As ComponentAssembly = workPart.ComponentAssembly
        if not IsNothing(c.RootComponent) then
            '*** insert code to process 'root component' (assembly file)
            lw.WriteLine("Assembly: " & c.RootComponent.DisplayName)
            lw.WriteLine(" + Active Arrangement: " & c.ActiveArrangement.Name)
            '*** end of code to process root component
            ReportComponentChildren(c.RootComponent, 0)
        else
            '*** insert code to process piece part
            lw.WriteLine("Part has no components")
        end if
    Catch e As Exception
        theSession.ListingWindow.WriteLine("Failed: " & e.ToString)
    End Try
    lw.Close

    End Sub

'**************************************************************
```

Search

User Login

Username *

Password *

- Create new account
- Request new password

Log in

RSS Feed

Main Menu

- Home
- Forums
- Tutorials
  - Getting started with journaling in NX
  - Create Geometry
  - User Interaction
  - Export Parasolid
  - Expressions: create an expression with units
  - Expressions: creating expressions
  - Expressions: query existing expressions
  - NXOpen.Features.Feature and the Feature Collection
  - Process file in directory
  - Read a text file
  - Report length of Tube Feature
  - Reset Component Position
  - Sort a list of NX objects
  - Start journal from custom button
  - Units
  - Using VectorArithmetic.Vector3
  - Wrangling the WCS
  - Write to a text file
  - Processing an Assembly
- Journals
- Submit
- Resources

```
    Sub reportComponentChildren( ByVal comp As Component, _
        ByVal indent As Integer)

        For Each child As Component In comp.GetChildren()
            '*** insert code to process component or subassembly
            lw.WriteLine(New String(" ", indent * 2) & child.DisplayName())
            '*** end of code to process component or subassembly
            if child.GetChildren.Length <> 0 then
                '*** this is a subassembly, add code specific to subassemblies
                lw.WriteLine(New String(" ", indent * 2) & _
                    "* subassembly with " & _
                    child.GetChildren.Length & " components")
                lw.WriteLine(New String(" ", indent * 2) & _
                    " + Active Arrangement: " & _
                    child.OwningPart.ComponentAssembly.ActiveArrangement.Name)
                '*** end of code to process subassembly
            else
                'this component has no children (it is a leaf node)
                'add any code specific to bottom level components
            end if
            reportComponentChildren(child, indent + 1)
        Next
    End Sub
'**************************************************************
    Public Function GetUnloadOption(ByVal dummy As String) As Integer
        Return Session.LibraryUnloadOption.Immediately
    End Function
'**************************************************************

End Module
```

So how does it work?

First, we grab the ComponentAssembly of the displayed part. The ComponentAssembly object gives us access to the components and allows us to do things such as add components, move components, query components, set attributes, etc. If the RootComponent property is Nothing, then we are dealing with a piece part (no components). If the RootComponent property indicates this part is an assembly, we call the ReportComponentChildren subroutine; this is where the heavy lifting takes place. Near the end of the ReportComponentChildren subroutine a call is made to the subroutine: ReportComponentChildren.

To understand recursion, you must first understand recursion

When a subroutine or function calls itself, it is called recursion. Recursion is very useful when dealing with tree type structures such as a directory listing or NX assembly. If you wanted a hand written list of all the folders on your C drive, you could do the following:

1. Open the C drive, write down the name of all the first folder
2. Open folder: if subfolders exist, write down the name of the first folder
3. No subfolders? move up a level and open the next folder
4. Repeat until you run out of folders

The beauty of this approach is it works if you only have a handful of folders or if you have hundreds of folders nested dozens of levels deep - you don't need to know anything about the structure, other than the basic layout, before you start. It may look strange for a subroutine to call itself, but as far as the computer is concerned it is a copy. So subroutine A calls itself, now subroutine A' is running which kicks off A''. A'' finishes returning control to A' which eventually finishes returning control to A. Each copy has its own copy of local variables to use so they won't overwrite each other's information. If my explanation left you scratching your head, you may want to try Wikipedia's definition.

Conclusion

A journal was presented that recursively steps through the assembly structure. Feel free to copy this code and modify it for your purposes. Some examples would be:

Recent Comments

- Whether the assembly is fully 3 days 6 hours ago
- An alternative way could be 4 days 11 hours ago
- Our initial plan was to use 4 days 11 hours ago
- re: edit part attribute 4 days 12 hours ago
- re: sketch plane 4 days 12 hours ago
- re: long run time 4 days 12 hours ago
- run journal in batch mode 6 days 6 hours ago
- re: run journal in batch mode 6 days 11 hours ago
- Thank you for your response. 6 days 15 hours ago
- re: PDF export 1 week 2 days ago

- assigning an attribute to every component in the assembly
- writing out the assembly structure to a text file or spreadsheet
- report the quantity of each component and/or the number of unique components

I hope you find this journal useful. If you experience any problems or could do some testing for a Teamcenter version, please let me know!

Tags: assembly recursive RootComponent GetChildren

Log in or register to post comments

Comments

Recursion

uwe.a - Sun, 05/20/2012 - 01:45

Hi,
I want to walk /process recursive from buttom up to the top level of a deeper fragmented assembly structure . -> I want to fix all compoments of a imported assembly.
thanks ind ad

Log in or register to post comments

Recursion

NXJournaling - Mon, 05/21/2012 - 16:05

If you want to process all the files in the assembly, recursively walking through an assembly works best from the top down. If you are starting somewhere down the tree, loop through the component's .RootComponent property until it is equal "nothing" - when that happens, you are at the top node.

If you are starting somewhere down the assembly tree and only want to process the files in its particular branch, you can get the .RootComponent each time to cycle up a level in the assembly. You will miss other "branches" of the assembly tree with this strategy.

Log in or register to post comments

Teamcenter - Same Process

christopherdomingues - Fri, 09/21/2012 - 07:04

I'd like to know if is possible create this same subroutine to run at Teamcenter?
I need to perform a task over files inside an assembly.
My journal works individually, but i need a way to run this determined task (just some clicks) over whole assembly components.

Christopher Prazeres
Brazil - Designer
551297174168
christopherdomingues@hotmail.com

Log in or register to post comments

RE: Teamcenter - Same Process

carlo.daristotile - Thu, 11/21/2013 - 03:36

Hello Christopher,
Did you find a solution to running this in Teamcenter?

Carlo Tony Daristotile

Log in or register to post comments

Teamcenter Assembly

NXJournaling - Fri, 09/21/2012 - 07:57

I currently do not have a Teamcenter install to test with. When you run the code, what errors do you get?

Log in or register to post comments

Write To A Text File

smitty239 - Tue, 02/12/2013 - 04:55

How would you go about writing this to a text file instead of sending it to the listing window ?

Log in or register to post comments

Re: Writing To A Text File

NXJournaling - Wed, 02/13/2013 - 12:01

Both .NET and NXOpen provide ways of writing to text files, here are 2 links for using .NET methods:
http://www.dotnetperls.com/streamwriter-vbnet
http://www.homeandlearn.co.uk/net/nets8p4.html

In the NXOpen API, the listing window itself provides for a method of writing to a text file. Have a look at the **SelectDevice** method of the **ListingWindow** object in the NXOpen API help file.

It is a question that I've seen come up multiple times, I'll add it to my queue of "future article ideas".

Log in or register to post comments

Components Obtained From A Function ???

carlo.daristotile - Wed, 10/23/2013 - 06:41

Hello I have a component returned from a function.
But then I cannot apply any component functions to the component.
Can anyone help with this.
Thanks

dim device_component(ConnectorDevice_s1.length - 1) as Component '''works
dim ConnectorDevice_s1() as nxopen.routing.electrical.ConnectorDevice '''works
for i as integer = 0 to ConnectorDevice_s1.length - 1 '''works
device_component(i) = ConnectorDevice_s1(i).OwningComponent '''works component returned from function
device_component(i).DisplayName() '''fails!!!!!!!!!!!!!!!!!!!!
next

Carlo Tony Daristotile

Log in or register to post comments

Re: Component Errors

NXJournaling - Thu, 10/24/2013 - 07:06

There are two problems with the code snippet above:

- device_component(i).DisplayName() cannot stand on its own; you need to use the value in some way (ie assign the value to a variable or use it as part of a larger expression)
- the loop variable is never incremented, probably the result of posting a small portion of your code (which I appreciate), but I thought I'd mention it just in case.

Is there an error message shown when you run your code? If so, what exactly does it say?

Log in or register to post comments

"OBJECT REFERENCE NOT SET TO AN INSTANCE OF AN OBJECT"

carlo.daristotile - Thu, 10/24/2013 - 12:11

HELP
I updated the code.
I still get an error "OBJECT REFERENCE NOT SET TO AN INSTANCE OF AN OBJECT"

```
dim ConnectorDevice_s1() as nxopen.routing.electrical.ConnectorDevice =
ConnectorDevice_Collection1.toarray()
dim device_component(ConnectorDevice_s1.length - 1) as Component
dim lineouts(ConnectorDevice_s1.length - 1) as string
dim lineOut as string = ""
for i as integer = 0 to ( ConnectorDevice_s1.length - 1 )
device_component(i) = ConnectorDevice_s1(i).OwningComponent '''works component returned
from function
lineOut = device_component(i).DisplayName '''fails!!!!!!!!!!!!!!!!!!!!
lineouts(i)= lineOut
next
```

Carlo Tony Daristotile

Log in or register to post comments

UNABLE TO CAST OBJECT OF TYPE

carlo.daristotile - Thu, 10/24/2013 - 12:20

Please HELP

'''fails!! UNABLE TO CAST OBJECT OF TYPE 'NXOpen.Routing.PartDefinitionShadow' to type 'NXOpen.Routing.Electrical.ElectricalPartDefinitionShadow'

```
dim ConnectorDevice_Collection1 as nxopen.routing.electrical.ConnectorDeviceCollection
ConnectorDevice_Collection1 = workpart.routemanager.ConnectorDevices
dim ConnectorDevice_s1() as nxopen.routing.electrical.ConnectorDevice =
ConnectorDevice_Collection1.toarray()
dim ElectricalPartDefinitionShadow1(ConnectorDevice_s1.length - 1) as
NXOpen.Routing.Electrical.ElectricalPartDefinitionShadow
dim lineouts(ConnectorDevice_s1.length - 1) as string
dim lineOut as string = ""
for i as integer = 0 to ( ConnectorDevice_s1.length - 1 )
ElectricalPartDefinitionShadow1(i) = ConnectorDevice_s1(i).GetPartDefinition() '''fails!! UNABLE
TO CAST OBJECT OF TYPE
lineouts(i)= lineOut
next
```

Carlo Tony Daristotile

Log in or register to post comments

Re: Cast Object Error

NXJournaling - Thu, 10/24/2013 - 14:09

I don't see an obvious reason why it shouldn't work. Perhaps this is a good one to take up with the GTAC help desk...

Log in or register to post comments

Re: Cast Object Error: "GTAC Reply"

carlo.daristotile - Tue, 10/29/2013 - 07:33

Hello,
This is the analysis I got from GTAC. I still dont see the error in my Journal.

The error states 'UNABLE TO CAST OBJECT OF TYPE' PartDefinitionShadow to type ElectricalPartDefinitionShadow.
In the documentation we can see the hierarchy.
PartDefinitionShadow Inheritance Hierarchy
Object
MarshalByRefObject
NXRemotableObject
TaggedObject
NXObject
RootObject
RouteObject
ItemDefinition
PartDefinitionShadow
ElectricalPartDefinitionShadow Inheritance Hierarchy
Object
MarshalByRefObject
NXRemotableObject
TaggedObject
NXObject
RootObject
RouteObject
ItemDefinition
PartDefinitionShadow
ElectricalPartDefinitionShadow
Note that ElectricalPartDefinitionShadow inherits from PartDefinitionShadow. What the journal is trying to do is cast an object down the inherintance tree. I.E. make an onject inherit from itself. One can usually cast to the same level or up. But not down.

Art Schumacher
Sr. Application Engineer
art.schumacher@siemens.com

'Brief Summary of My Journal
dim ConnectorDevice_s1() as nxopen.routing.electrical.ConnectorDevice = ConnectorDevice_Collection1.toarray()
dim EleclPartDefShad1(ConnectorDevice_s1.length - 1) as NXOpen.Routing.Electrical.ElectricalPartDefinitionShadow
for i as integer = 0 to (ConnectorDevice_s1.length - 1)
'"fails!! UNABLE TO CAST OBJECT OF TYPE 'NXOpen.Routing.PartDefinitionShadow' to type 'NXOpen.Routing.Electrical.ElectricalPartDefinitionShadow'
EleclPartDefShad1(i) = ConnectorDevice_s1(i).GetPartDefinition()
Next

Carlo Tony Daristotile

Log in or register to post comments

Re: GTAC Answer

NXJournaling - Tue, 10/29/2013 - 08:31

I understand what the GTAC representative is saying, however the question becomes "where is this NXOpen.Routing.PartDefinitionShadow object coming from?".

If your actual code matches what you have posted here, I don't see where this PartDefinitionShadow is coming from. According to the .net API reference docs (7.5, 8, & 8.5), a "ConnectorDevice" object is part of the Routing.Electrical namespace and the ".GetPartDefinition" method of this object should return an "ElectricalPartDefinitionShadow" object.

Log in or register to post comments

Cycle Through Each Component As Make Displayed/Work Part ?

carlo.daristotile - Fri, 11/01/2013 - 04:32

Is it possible to
Cycle through each component as Make Displayed part ?
or
Cycle through each component as Work Displayed part ?

Carlo Tony Daristotile

Re: Cycling Components

NXJournaling - Fri, 11/01/2013 - 05:49

The short answer is: yes that is possible.

I'd suggest opening an assembly, start the journal recorder and make one of the components the displayed/work part as desired. Stop the recording and examine the code, incorporate the code as needed into the journal above.

Cycle Through 1st Level Components And Make Displayed Part

carlo.daristotile - Fri, 11/01/2013 - 08:11

It works

```
'Journal to walk through the assembly structure
' will run on assemblies or piece parts
' will step through all "components of 1st level" of the displayed part and make them displayed part

Option Strict Off

Imports System
Imports NXOpen
Imports NXOpen.UF
Imports NXOpen.Assemblies
Imports System.IO
Imports NXOpenUI
Imports System.Windows.Forms

Module NXJournal

Public theSession As Session = Session.GetSession()
Public ufs As UFSession = UFSession.GetUFSession()
Public lw As ListingWindow = theSession.ListingWindow

Sub Main()
Dim workPart As Part = theSession.Parts.Work
Dim dispPart As Part = theSession.Parts.Display
'lw.Open
Try
Dim c As ComponentAssembly = dispPart.ComponentAssembly
'to process the work part rather than the display part,
' comment the previous line and uncomment the following line
'Dim c As ComponentAssembly = workPart.ComponentAssembly
if not IsNothing(c.RootComponent) then
'*** insert code to process 'root component' (assembly file)
'lw.WriteLine("Assembly: " & c.RootComponent.DisplayName)
'lw.WriteLine(" + Active Arrangement: " & c.ActiveArrangement.Name)
'*** end of code to process root component
'lw.WriteLine("")
'lw.WriteLine("comps")
```

```
'lw.WriteLine("")
dim comps() as component = c.RootComponent.getchildren()
Dim part2 As Part = CType(theSession.Parts.FindObject(c.RootComponent.DisplayName),
Part)
for i as integer = 0 to ( comps.length - 1 )
'lw.writeline( comps(i).name )
'------------------------------------------------------------
Dim component1 As Assemblies.Component = comps(i)
Dim components1(0) As Assemblies.Component
components1(0) = component1
Dim errorList1 As ErrorList
errorList1 = component1.DisplayComponentsExact(components1)
errorList1.Clear()
Dim part1 As Part = CType(theSession.Parts.FindObject(comps(i).displayname), Part)
part1.Preferences.Modeling.CutViewUpdateDelayed = True
Dim partLoadStatus1 As PartLoadStatus
Dim status1 As PartCollection.SdpsStatus
status1 = theSession.Parts.SetDisplay(part1, True, False, partLoadStatus1)
workPart = theSession.Parts.Work
dispPart = theSession.Parts.Display
partLoadStatus1.Dispose()
MessageBox.Show(comps(i).displayname)
'------------------------------------------------------------
'Dim part2 As Part = CType(theSession.Parts.FindObject(c.RootComponent.DisplayName),
Part)
workPart.Preferences.Modeling.CutViewUpdateDelayed = True
Dim partLoadStatus2 As PartLoadStatus
Dim status2 As PartCollection.SdpsStatus
status2 = theSession.Parts.SetDisplay(part2, True, False, partLoadStatus2)
workPart = theSession.Parts.Work
dispPart = theSession.Parts.Display
partLoadStatus2.Dispose()
'------------------------------------------------------------
next
workPart.Preferences.Modeling.CutViewUpdateDelayed = True
Dim partLoadStatus6 As PartLoadStatus
Dim status6 As PartCollection.SdpsStatus
status6 = theSession.Parts.SetDisplay(part2, True, False, partLoadStatus6)
workPart = theSession.Parts.Work
dispPart = theSession.Parts.Display
Dim nullAssemblies_Component As Assemblies.Component = Nothing
dispPart.Preferences.Modeling.CutViewUpdateDelayed = True
Dim partLoadStatus7 As PartLoadStatus
theSession.Parts.SetWorkComponent(nullAssemblies_Component,
PartCollection.RefsetOption.Current, PartCollection.WorkComponentOption.Visible,
partLoadStatus7)
workPart = theSession.Parts.Work
partLoadStatus7.Dispose()
MessageBox.Show(c.RootComponent.DisplayName)
else
'*** insert code to process piece part
lw.WriteLine("Part has no components")
end if
Catch e As Exception
theSession.ListingWindow.WriteLine("Failed: " & e.ToString)
End Try
'lw.Close

End Sub

'**************************************************************
Public Function GetUnloadOption(ByVal dummy As String) As Integer
Return Session.LibraryUnloadOption.Immediately
```

```
End Function
'*************************************************************

End Module
```

Carlo Tony Daristotile

Verify Each Component Is Loaded?

carlo.daristotile - Tue, 02/25/2014 - 04:11

How do you verify each component is loaded/opened?
Thanks

Carlo Tony Daristotile

Re: Verify Load Status

NXJournaling - Tue, 02/25/2014 - 05:46

I've been doing some work on a class that gathers information about the open assembly. It isn't finished yet, but I think it is far enough along to help answer your question. The LoadComponent function in the code listing below (last function in the listing), checks to see if the given component is fully or partially loaded; if it is not, it will attempt to load the component and check the "load status" variable to see if it was successful.

```
Option Strict Off
Imports System
Imports System.Collections.Generic
Imports System.windows.forms
Imports NXOpen
Imports NXOpen.UF
Imports NXOpen.Assemblies

Module Module1

    Sub Main()

        Dim theSession As Session = Session.GetSession()

        If IsNothing(theSession.Parts.Display) Then
            MessageBox.Show("Active Part Required", "Error", MessageBoxButt
            Return
        End If

        Dim lw As ListingWindow = theSession.ListingWindow
        lw.Open()

        Dim myAsmInfo As New NXJ_Assembly_info
        myAsmInfo.Part = theSession.Parts.Work

        lw.WriteLine("All Components")
        lw.WriteLine("--------------")
        For Each tempComp As Assemblies.Component In myAsmInfo.AllComponent
            lw.WriteLine(tempComp.DisplayName)

        Next

        lw.WriteLine("")
        lw.WriteLine("All Unique Parts")
        lw.WriteLine("----------------")
        For Each tempPart As Part In myAsmInfo.AllUniqueParts
            lw.WriteLine(tempPart.FullPath)
        Next
```

```
        lw.WriteLine("")
        lw.WriteLine("Parts that could not be loaded")
        lw.WriteLine("------------------------------")
        For Each temp As String In myAsmInfo.NotLoaded
            lw.WriteLine(temp)
        Next


    End Sub

End Module


Public Class NXJ_Assembly_info

#Region "Private Variables"

    Private Const Version As String = "0.1.1"

    Private _theSession As Session = Session.GetSession()
    Private _theUfSession As UFSession = UFSession.GetUFSession

    Private _components As New List(Of Assemblies.Component)
    Private _uniqueParts As New List(Of Part)
    Private _allComponents As New List(Of Assemblies.Component)
    Private _allUniqueParts As New List(Of Part)
    Private _notLoaded As New List(Of String)

    Private lg As LogFile = _theSession.LogFile

#End Region

#Region "Properties"

    Private _isTCRunning As Boolean
    Public ReadOnly Property IsTCRunning() As Boolean
        Get
            Return _isTCRunning
        End Get
    End Property

    Private _thePart As Part = Nothing
    Public Property Part() As Part
        Get
            Return _thePart
        End Get
        Set(ByVal value As Part)
            _thePart = value
            'Me.GetInfo()
            Me.GetAllInfo()
        End Set
    End Property

    Public ReadOnly Property AllComponents() As List(Of Component)
        Get
            Return _allComponents
        End Get
    End Property

    Public ReadOnly Property AllUniqueParts() As List(Of Part)
        Get
            Return _allUniqueParts
        End Get
    End Property

    Public ReadOnly Property Components As List(Of Component)
        Get
            Return _components
        End Get
    End Property

    Public ReadOnly Property UniqueParts As List(Of Part)
```

```
        Get
            Return _uniqueParts
        End Get
    End Property

    Public ReadOnly Property NotLoaded As List(Of String)
        Get
            Return _notLoaded
        End Get
    End Property

#End Region

    Public Sub New()

        lg.WriteLine("")
        lg.WriteLine("~ NXJournaling.com: NXJ_Assembly_info object created
        lg.WriteLine("  ~~ Version: " & Version & " ~~")
        lg.WriteLine("  ~~ Timestamp of run: " & DateTime.Now.ToString & "
        lg.WriteLine("NXJ_Assembly_info Sub New()")

        'determine if we are running under TC or native
        _theUfSession.UF.IsUgmanagerActive(_isTCRunning)
        lg.WriteLine("IsTcRunning: " & _isTCRunning.ToString)

        lg.WriteLine("exiting Sub New")
        lg.WriteLine("")


    End Sub

    Private Sub GetAllInfo()

        'get all component info from assembly (all levels)
        lg.WriteLine("Sub GetAllInfo()")

        Try
            Dim c As ComponentAssembly = Part.ComponentAssembly
            If Not IsNothing(c.RootComponent) Then
                '*** insert code to process 'root component' (assembly file
                lg.WriteLine("  part has components")
                '*** end of code to process root component
                lg.WriteLine("  calling GetAllComponentChildren")
                GetAllComponentChildren(c.RootComponent)
            Else
                '*** insert code to process piece part, part has no compone
                lg.WriteLine("  part has no components")
            End If
        Catch ex As NXException
            lg.WriteLine("Sub GetAllInfo error: " & ex.ErrorCode)
            lg.WriteLine("  " & ex.Message)
        End Try

        lg.WriteLine("exiting Sub GetAllInfo()")

    End Sub

    Private Sub GetAllComponentChildren(ByVal comp As Component)

        For Each child As Component In comp.GetChildren()
            lg.WriteLine(child.DisplayName)
            '*** insert code to process component or subassembly

            If Me.LoadComponent(child) Then

                _allComponents.Add(child)

                Dim tempPart As Part = child.Prototype.OwningPart
                If Not _allUniqueParts.Contains(tempPart) Then
                    _allUniqueParts.Add(tempPart)
                End If

            Else
```

```
            'component could not be loaded
        End If
        '*** end of code to process component or subassembly
        If child.GetChildren.Length <> 0 Then
            '*** this is a subassembly, add code specific to subassembl

            '*** end of code to process subassembly
        Else
            'this component has no children (it is a leaf node)
            'add any code specific to bottom level components

        End If
        Me.GetAllComponentChildren(child)
    Next
End Sub


Private Sub GetInfo()

    'get top level component info from assembly (no recursion)
    lg.WriteLine("Sub GetInfo()")

    Try
        Dim c As ComponentAssembly = Part.ComponentAssembly
        If Not IsNothing(c.RootComponent) Then
            '*** insert code to process 'root component' (assembly file
            lg.WriteLine("  part has components")
            '*** end of code to process root component
            lg.WriteLine("  calling GetComponentChildren")
            Me.GetComponentChildren(c.RootComponent)
        Else
            '*** insert code to process piece part, part has no compone
            lg.WriteLine("  part has no components")
        End If
    Catch ex As NXException
        lg.WriteLine("Sub GetInfo error: " & ex.ErrorCode)
        lg.WriteLine("  " & ex.Message)
    End Try

    lg.WriteLine("exiting GetInfo()")

End Sub

Private Sub GetComponentChildren(ByVal comp As Component)

    For Each child As Component In comp.GetChildren()
        '*** insert code to process component or subassembly
        _components.Add(child)
        Dim tempPart As Part = child.Prototype.OwningPart
        If Not _uniqueParts.Contains(tempPart) Then
            _uniqueParts.Add(tempPart)
        End If
        '*** end of code to process component or subassembly
        If child.GetChildren.Length <> 0 Then
            '*** this is a subassembly, add code specific to subassembl

            '*** end of code to process subassembly
        Else
            'this component has no children (it is a leaf node)
            'add any code specific to bottom level components

        End If
    Next
End Sub

Private Function LoadComponent(ByVal theComponent As Component) As Bool

    lg.WriteLine("Sub LoadComponent()")

    Dim thePart As Part = theComponent.Prototype.OwningPart

    Dim partName As String = ""
    Dim refsetName As String = ""
```

```
        Dim instanceName As String = ""
        Dim origin(2) As Double
        Dim csysMatrix(8) As Double
        Dim transform(3, 3) As Double

        Try
            If thePart.IsFullyLoaded Then
                'component is fully loaded
            Else
                'component is partially loaded
            End If
            lg.WriteLine("  component: " & theComponent.DisplayName & " is
            lg.WriteLine("  return: True")
            lg.WriteLine("exiting Sub LoadComponent()")
            lg.WriteLine("")
            Return True
        Catch ex As NullReferenceException
            'component is not loaded
            Try
                lg.WriteLine("  component not loaded, retrieving part infor
                _theUfSession.Assem.AskComponentData(theComponent.Tag, part
                lg.WriteLine("  component part file: " & partName)

                Dim theLoadStatus As PartLoadStatus
                _theSession.Parts.Open(partName, theLoadStatus)

                If theLoadStatus.NumberUnloadedParts > 0 Then
                    If theLoadStatus.NumberUnloadedParts > 1 Then
                        lg.WriteLine("  problem loading " & theLoadStatus.N
                    Else
                        lg.WriteLine("  problem loading 1 component")
                    End If

                    Dim allReadOnly As Boolean = True
                    For i As Integer = 0 To theLoadStatus.NumberUnloadedPar
                        lg.WriteLine("part name: " & theLoadStatus.GetPartN
                        lg.WriteLine("part status: " & theLoadStatus.GetSta
                        If theLoadStatus.GetStatus(i) = 641058 Then
                            'read-only warning, file loaded ok
                        Else
                            '641044: file not found
                            allReadOnly = False
                            If Not _notLoaded.Contains(partName) Then
                                _notLoaded.Add(partName)
                            End If
                        End If
                        lg.WriteLine("status description: " & theLoadStatus
                        lg.WriteLine("")
                    Next
                    If allReadOnly Then
                        lg.WriteLine("  'read-only' warnings only")
                        lg.WriteLine("  return: True")
                        Return True
                    Else
                        'warnings other than read-only...
                        lg.WriteLine("  return: False")
                        lg.WriteLine("exiting Sub LoadComponent()")
                        lg.WriteLine("")
                        Return False
                    End If
                Else
                    lg.WriteLine("  component(s) loaded successfully")
                    lg.WriteLine("  return: True")
                    lg.WriteLine("exiting Sub LoadComponent()")
                    lg.WriteLine("")
                    Return True
                End If

            Catch ex2 As NXException
                lg.WriteLine("  Load error: " & ex2.Message)
                lg.WriteLine("  error code: " & ex2.ErrorCode)
                lg.WriteLine("  return: False")
                lg.WriteLine("exiting Sub LoadComponent()")
```

```
                lg.WriteLine("")
                If ex2.Message.ToLower = "file not found" Then
                    If Not _notLoaded.Contains(partName) Then
                        _notLoaded.Add(partName)
                    End If
                End If
                Return False
            End Try
        Catch ex As NXException
            'unexpected error
            lg.WriteLine("  Error in Sub LoadComponent: " & ex.Message)
            lg.WriteLine("  return: False")
            lg.WriteLine("exiting Sub LoadComponent()")
            lg.WriteLine("")
            Return False
        End Try

    End Function

End Class
```

Log in or register to post comments

Internal Error

DHuskic - Sat, 07/12/2014 - 15:04

I received an internal error when attempting to un-hide everything in a Nx 7.5 assembly after running this code in Nx 7.5, I later used the same assembly in Nx 8.5 and attempted the same process with success. Any ideas?

DHuskic
Nx 9 VB

Log in or register to post comments

Re: Internal Error

NXJournaling - Sat, 07/12/2014 - 17:33

I was using NX 8.5 when I wrote the code, but I can't see anything obvious that would cause errors in 7.5. The code writes updates to the log file, check the log after you run the code, it may give you clues as to what is causing the error.

Log in or register to post comments

Replace Component

mattfitt - Sat, 02/04/2017 - 02:55

Hi
Do you know what the command to replace a child component after you identify it as not loaded?
Thanks

Log in or register to post comments

Re: Replace Component

NXJournaling - Mon, 02/06/2017 - 11:36

There are a number of reasons why a component may not be loaded. Perhaps it would be better to see if the component in question could be loaded, rather than replacing it?

If you need a code sample of replacing a component (or just about anything else), I suggest recording a journal while performing the operation. The returned code will show you what commands are necessary; with a little tweaking, the code can be made more general (usually it only works on objects involved at the time of recording).

Log in or register to post comments

Output To Excel

ian.eldred - Wed, 07/09/2014 - 05:16

At the end of this tutorial, you left us with the 'homework' to output the BoM to excel. I have done that and posted the example here. It also generates a screenshot and reports the quantity of each part.

Log in or register to post comments

PDF Batch All In Folder

pfillion - Thu, 03/05/2015 - 03:59

Paul Fillion

Log in or register to post comments

Measure Bodies

Junfanbl - Thu, 06/04/2015 - 09:25

So what would I put in to make this script measure bodies for each piece part in the assembly, while saving a text file to the hard drive for each part? I tired different things but I am not getting the desired result.

Log in or register to post comments

Moving To Layers

Germfask - Wed, 01/11/2017 - 07:27

I would like to use this journal as a base for a customer requirement that I have.

The requirement is that all purchase parts be on a specific range of layers and all manufactured parts be on another set of layers. The parts can be identified by the first character of the part number.

Any recommendations on how to approach this? I was thinking about adding a conditional statement to determine the part file name and then move to the appropriate layer, and then index the layer number by one.

Log in or register to post comments

Re: Moving To Layers

NXJournaling - Wed, 01/11/2017 - 12:39

Sounds like a good plan. If you are working in native NX, you can get the component part file name with something like:

```
{component variable}.Prototype.OwningPart.Leaf
```

So if you were working on the file: "C:\CAD_data\rest\of\the\path\12345.prt", the above command would return "12345". Then you can use the .net string functions to read the first character.

Log in or register to post comments

Use Code In A Public Class

tim-91 - Wed, 01/03/2018 - 00:56

Hello! First off thank you for all the information about journaling with NX! I'm currently working on a program that analyses assemblies and parts to predict the cost of a product. The program is able to process parts already. Now I need to cycle trough all parts of my assembly. I use a Public Class that contains Public functions and subs. But I get errors if I change the program from a module to a Class. The Error Message: System.Reflection.TargetException: The non-static method requires target. Or is there a way to integrate a Module in a Class and call Functions from the Class?

Greetings
Tim Schröder

```
Imports Microsoft.VisualBasic
Imports System
Imports NXOpen
Imports NXOpen.UF
Imports NXOpen.Assemblies


Public Class NXJournal

    Public theSession As Session = Session.GetSession()
    Public ufs As UFSession = UFSession.GetUFSession()
    Public lw As ListingWindow = theSession.ListingWindow
    ' class members
    'Public theSession As Session
    'Public theUFSession As UFSession
    Public theUI As UI
    'Public lw As ListingWindow
    'Public workPart As Part

    'Zählervariable
    'Dim z As Integer

    Dim workPart As Part = theSession.Parts.Work
    Dim dispPart As Part = theSession.Parts.Display
    Dim z As Integer = 0

    Public Sub Main(ByVal args() As String)

        lw.Open

        Try
            Dim c As ComponentAssembly = dispPart.ComponentAssembly
            'to process the work part rather than the display part,
            '  comment the previous line and uncomment the following line
            'Dim c As ComponentAssembly = workPart.ComponentAssembly
            If Not IsNothing(c.RootComponent) Then
                '*** insert code to process 'root component' (assembly file
                lw.WriteLine("Assembly: " & c.RootComponent.DisplayName)
                lw.WriteLine(" + Active Arrangement: " & c.ActiveArrangemer
                '*** end of code to process root component
                reportComponentChildren(c.RootComponent, 0)

            Else
                '*** insert code to process piece part
                lw.WriteLine("Part has no components")

            End If
        Catch e As Exception
            theSession.ListingWindow.WriteLine("Failed: " & e.ToString)
        End Try
        lw.writeLine("Test" & z)
        lw.Close

    End Sub

    '**************************************************************
    Public Sub reportComponentChildren(ByVal comp As Component,
        ByVal indent As Integer)

        For Each child As Component In comp.GetChildren()
            '*** insert code to process component or subassembly
```

```
            lw.WriteLine(New String(" ", indent * 2) & child.DisplayName())
            If child.DisplayName Like "03-*" Then
                    'test2.testsub(z)
            End If
            '*** end of code to process component or subassembly
            If child.GetChildren.Length <> 0 Then
                '*** this is a subassembly, add code specific to subassembl
                lw.WriteLine(New String(" ", indent * 2) &
                    "* subassembly with " &
                    child.GetChildren.Length & " components")
                lw.WriteLine(New String(" ", indent * 2) &
                " + Active Arrangement: " &
                child.OwningPart.ComponentAssembly.ActiveArrangement.Name)
                            '*** end of code to process subassembly
            Else
                'this component has no children (it is a leaf node)
                'add any code specific to bottom level components
            End If
            reportComponentChildren(child, indent + 1)
        Next
    End Sub
    '************************************************************
    'Sub GetUnloadOption(ByVal dummy As String) 'As Integer
    'Return Session.LibraryUnloadOption.Immediately
    'End Sub
    '************************************************************

End Class
```

Log in or register to post comments

Re: Module To Class

NXJournaling - Wed, 01/03/2018 - 05:15

Why do you want to convert your code from a module to a class? What benefit do you hope to gain by doing so?

Log in or register to post comments

Hi,

tim-91 - Wed, 01/03/2018 - 06:48

Hi,
I just want to integrate the code in my program that uses a Public Class. I figured out, that it isn't possible to call a Function from a Module that is contained in a Public Class. Is there a way to change this code, to run in a Class instead of a Module? This would save me a lot of work!
I'm currently working on my thesis and I've never been working with Visual Basic before. I don't really know exactly the difference between a Module and a Class. The program I'm working on is based on a Class that contains Functions and Subs. Or is there a way to integrate this Module in my program?

Log in or register to post comments

Re: Module Vs. Class

NXJournaling - Wed, 01/03/2018 - 08:21

"I figured out, that it isn't possible to call a Function from a Module that is contained in a Public Class."

You will need to create an instance of the class in your module, then you can call its public methods and properties. If you are going to be using VB, I suggest reading up on modules and classes. Each have their use and trying to force one to work like the other may be more trouble than it is worth.

Below is a quick and dirty (i.e. probably not the best) way to get your journal to run as a class.

```
Imports Microsoft.VisualBasic
Imports System
Imports NXOpen
Imports NXOpen.UF
Imports NXOpen.Assemblies


Public Class NXJournal

    public shared theSession As Session
    public shared lw As ListingWindow

    'Zählervariable
    'Dim z As Integer

    'Dim workPart As Part = theSession.Parts.Work
    'Dim dispPart As Part = theSession.Parts.Display
    Dim z As Integer = 0

    public sub New()
        theSession = Session.GetSession()
        lw = theSession.ListingWindow
    end sub

    Public shared Sub Main(ByVal args() As String)
        dim theProgram as new NXJournal

        NXJournal.lw.Open

        Try
            Dim c As ComponentAssembly = NXJournal.theSession.Parts.Disp
            'to process the work part rather than the display part,
            '  comment the previous line and uncomment the following lin
            'Dim c As ComponentAssembly = workPart.ComponentAssembly
            If Not IsNothing(c.RootComponent) Then
                '*** insert code to process 'root component' (assembly j
                NXJournal.lw.WriteLine("Assembly: " & c.RootComponent.Di
                NXJournal.lw.WriteLine(" + Active Arrangement: " & c.Act
                '*** end of code to process root component
                reportComponentChildren(c.RootComponent, 0)

            Else
                '*** insert code to process piece part
                NXJournal.lw.WriteLine("Part has no components")

            End If
        Catch e As Exception
            NXJournal.lw.WriteLine("Failed: " & e.ToString)
        End Try
        NXJournal.lw.writeLine("Test" & theProgram.z.ToString)
        NXJournal.lw.Close

    End Sub

    '**************************************************************
    Public shared Sub reportComponentChildren(ByVal comp As Component,
        ByVal indent As Integer)

        For Each child As Component In comp.GetChildren()
            '*** insert code to process component or subassembly
            NXJournal.lw.WriteLine(New String(" ", indent * 2) & child.D
            If child.DisplayName Like "03-*" Then
                      'test2.testsub(z)
            End If
            '*** end of code to process component or subassembly
            If child.GetChildren.Length <> 0 Then
                '*** this is a subassembly, add code specific to subasse
                NXJournal.lw.WriteLine(New String(" ", indent * 2) &
                    "* subassembly with " &
                    child.GetChildren.Length & " components")
```

```
                NXJournal.lw.WriteLine(New String(" ", indent * 2) &
                " + Active Arrangement: " &
                child.OwningPart.ComponentAssembly.ActiveArrangement.Nar
                            '*** end of code to process subassembly
            Else
                'this component has no children (it is a leaf node)
                'add any code specific to bottom level components
            End If
            reportComponentChildren(child, indent + 1)
        Next
    End Sub
    '************************************************************
    'Sub GetUnloadOption(ByVal dummy As String) 'As Integer
    'Return Session.LibraryUnloadOption.Immediately
    'End Sub
    '************************************************************

End Class
```

Log in or register to post comments

Hi! It's Working Now! Thank

tim-91 - Thu, 01/04/2018 - 06:57

Hi! It's working now! Thank you so much for your help. It's not easy to get useful information about programing with NX exapt on this site!

Log in or register to post comments

Recent Content

Creating sketch on a user selected plane (or face) and sketch origin on a user selected point fathmi
Long run time when working with loaded assembly Rafal
Error/questions about a nongui running NX vb.script obe0009
Edit part attribute in catagory Frank F
Resources NXJournaling
NX crash with pdf export journal Frank F
User information on each custom button Click fathmi
Get Curve from centerline2dBuilder hebmwo
Part write permissions GervaldNik
passing argument to Main() JXB