Testing if an XML Node Has Relatives

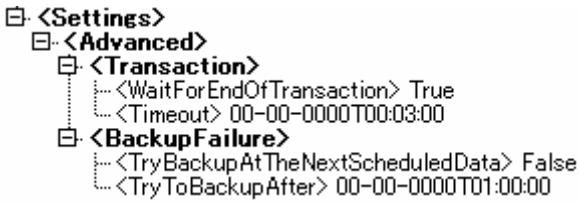
By David Adams
Technical Note 07-04

Abstract

このテクニカルノートでは、ノードの親、子、兄弟が存在するかテストする方法を説明し、4th Dimensionのコマンドを使用して実装されたメソッドを含むサンプルデータベースを提供します。

Overview

4th DimensionのXMLコマンドを使用して、要素の名前や属性、値、そしてCDATAセクションの内容など、XML要素の詳細な情報を得ることができます。ところで、カレントのノードが親や子、兄弟ノードを持つかどうかを知ることが必要になることがあります。例として、XML設定を階層リストに表示するデータベースがあるとしましょう。リストの内容を理解しやすいように視覚化するため、子要素を持つ要素を太字で表示します:



以下のコードは、この視覚化をどのようにして行うかを示します:

```
If (XmlNode_HasChildren ($xml_node))
   SET LIST ITEM PROPERTIES($list;$list_item;False;Bold;0)
End if
```

関連するノードをテストするその他の理由にはいかのようなものがあります:

■ 多くの一般的なXML木構造走査アルゴリズムでは、どこで停止するかを決定するために、子や親をテストする必要があります。

■ カスタムXML読み込みコードでは、レコードや関連するレコードをいつ作成したり保存したりするかを決定するために、兄弟ノードや子ノードが存在するかを知る必要があります。

■ XML木構造の中で異なるレベルに同じ要素名が使用されているとき、親や子、兄弟ノードをテストすることで、どの要素がカレントであるかを知ることができる場合があります。

このテクニカルノートでは親や子、兄弟ノードをテストする方法を説明し、メソッドが実装されたサンプルデータベースを提供します。

Background: DOM, SAX and Trees

コードを見る前に、XML処理について確認しましょう。

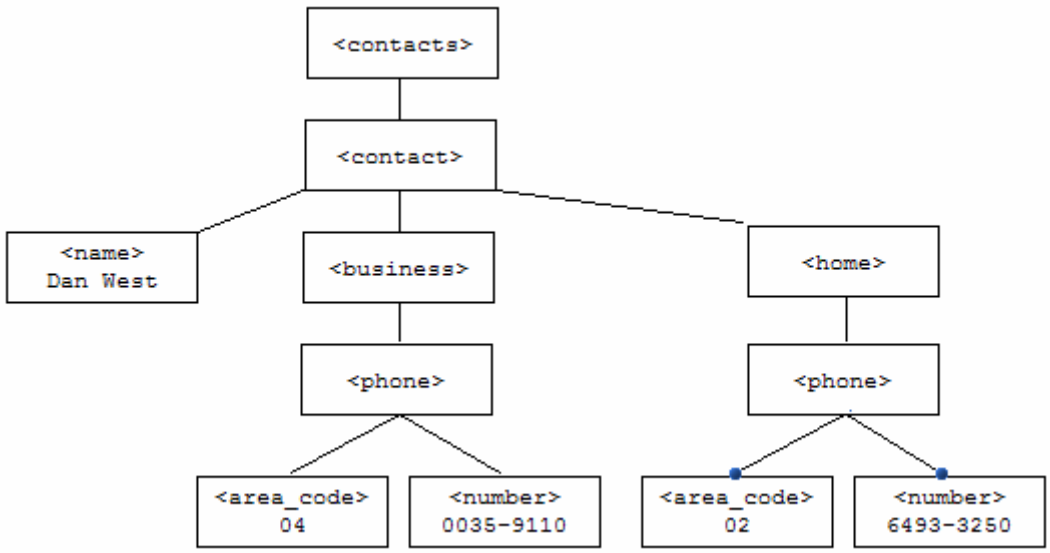
4th Dimensionには2つのXMLパーサ、DOM (Document Object Model) パーサとSAX (Simple API for XML) パーサが用意されています。両ツールともノードに関する同じ情報へのアクセスを提供します。しかしXML処理を行う際のアプローチがまったく異なっています。

SAXモデルを使用した場合、XMLドキュメントはテキストのストリームとして扱われます。この環境下では、関連するノードというコンセプトはまったく存在しません。

DOMモデルの場合、XMLドキュメントはリンクされたノードの木構造として、内部的に展開されます。例として以下のようなXMLを見てみましょう:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<contacts>
  <contact>
    <name>Dan West</name>
    <business>
      <phone>
        <area_code>04</area_code>
        <number>0035-9110</number>
      </phone>
    </business>
    <home>
      <phone>
        <area_code>02</area_code>
        <number>6493-3250</number>
      </phone>
    </home>
  </contact>
</contacts>
```

木構造として展開されると、ノードは以下のようにリンクされます:



DOM環境では、ノードの関連について論じることは意味のあることです。例えば、<business>要素は<contact>と<contacts>を祖先ノードに持ち、<name>と<home>を兄弟、<phone>、<area_code>そして<number>を子要素に持ちます。

Note *XMLパスや要素名、属性名はすべて大文字小文字を区別します。大文字小文字を区別する文字列の比較については**4D Technical Note 05-41,*** Case-Sensitive Operations in 4th Dimension*を参照してください。*

Using the Routines

このテクニカルノートで紹介するコードは、以下のように直感的に利用することができます:

```
C_STRING(16;$contacts_xmlref)
$contacts_xmlref:=DOM Find XML element($root_xmlref;"/contacts/") ` Find a node.
$hasAncestors_b:=DOM_NodeHasAncestors ($contacts_xmlref)
$hasChildren_b:=DOM_NodeHasChildren ($contacts_xmlref)
$hasSiblings_b:=DOM_NodeHasSiblings ($contacts_xmlref)
```

コードの呼び出しに必要なのは、DOMコマンドを使用してXMLノード参照を得ることです。どのような方法でノード参照を得たかは関係ありません。関連チェックルーチンは完全に一般化されています。さらにルーチンに渡した参照が無効であった場合も安全です。内部的にノードテストルーチンはカスタムエラーハンドラを使用して、エラーの表示を避けるようにしています。ヌルもしくは無効なノード参照をルーチンに渡した場合、ルーチンは**False**を返します。

The Routines

関連テストルーチンのコードとエラー管理ルーチンを以下に示します。

DOM_NodeHasChildren

*DOM_NodeHasChildren*ルーチンは、ノードが1つ以上の子ノードを持つかテストします。ノード参照が無効である場合、**False**が返されます。

```
C_BOOLEAN($0;$nodeHasChildren_b)
C_STRING(16;$1;$noderef)
$nodeHasChildren_b:=False  ` Default to False in case there are errors.
$noderef:=$1

DOM_StartCustomErrorHandling

C_STRING(16;$child_xmlref)
$child_xmlref:=DOM Get first child XML element($noderef)  ` Try to get a child.

  ` If the resulting xmlref is valid, there is at least one child.
$nodeHasChildren_b:=DOM_ReferenceIsValid ($child_xmlref)

DOM_StopCustomErrorHandling

$0:=$nodeHasChildren_b
```

DOM_NodeHasAncestors

DOM_NodeHasAncestors ルーチンは、ノードが1つ以上の祖先ノードを持つかテストします。ノード参照が無効である場合や、`#document`ノードをさす場合、**False**が返されます。`#document`要素は木構造のルートのさらに上にある仮想的な要素で、祖先ノードとしては扱いません。

```
C_BOOLEAN($0;$nodeHasAncestors_b)
C_STRING(16;$1;$noderef)
$nodeHasAncestors_b:=False  ` Default to False in case there are errors.
$noderef:=$1

DOM_StartCustomErrorHandling

C_STRING(16;$parent_xmlref)
```

```
C_TEXT($name_text)
$parent_xmlref:=""
$name_text:=""

   ` Try to get a parent.
$parent_xmlref:=DOM Get parent XML element($noderef;$name_text)
$nodeHasAncestors_b:=DOM_ReferenceIsValid ($parent_xmlref)

   ` If the resulting xmlref is valid, there is at least one ancestor.
If ($nodeHasAncestors_b)   ` There appear to be ancestors.
   If ($name_text="#document")   ` #document is an artificial node above the tree.
      $nodeHasAncestors_b:=False
   End if
End if

DOM_StopCustomErrorHandling

$0:=$nodeHasAncestors_b
```

DOM_NodeHasSiblings

*DOM_NodeHasSiblings*ルーチンは、ノードが1つ以上の兄弟ノードを持つかテストします。ノード参照が無効である場合、**False**が返されます。

```
C_BOOLEAN($0;$nodeHasSiblings_b)
C_STRING(16;$1;$noderef)
$noderef:=$1
$nodeHasSiblings_b:=False   ` Default to False in case there are errors.

DOM_StartCustomErrorHandling

C_STRING(16;$sibling_xmlref)
   ` Try to get an earlier sibling.
$sibling_xmlref:=DOM Get previous sibling XML element($noderef)

   ` If the resulting xmlref is valid, there is at least one sibling.
$nodeHasSiblings_b:=DOM_ReferenceIsValid ($sibling_xmlref)
If (Not($nodeHasSiblings_b))   ` There was no previous sibling, is there a next sibling?
      ` Try to get an earlier sibling.
   $sibling_xmlref:=DOM Get Next sibling XML element($noderef)
      ` If the resulting xmlref is valid, there is at least one sibling.
   $nodeHasSiblings_b:=DOM_ReferenceIsValid ($sibling_xmlref)
End if

DOM_StopCustomErrorHandling

$0:=$nodeHasSiblings_b
```

DOM_ReferenceIsValid

*DOM_ReferenceIsValid*ルーチンは4D Technical Note #06-40, *Enhancing the DOM XML Reading Functions*から引用されています。変更されたコードは以下になります:

```
C_BOOLEAN($0;$nodrefIsValid_b)
C_STRING(16;$1;$noderef)
$noderef:=$1
$nodrefIsValid_b:=True

DOM_StartCustomErrorHandling

C_TEXT($elementName_t)   ` Line below should throw an error if the element is not valid.
$elementName_t:=""
DOM GET XML ELEMENT NAME($noderef;$elementName_t)

Case of
   ¥ (DOM_Error#0)   ` There was an error of some kind.
      $nodrefIsValid_b:=False
   ¥ ($elementName_t="")   ` Elements must have names to be operated on safely.
```

```
      $nodrefIsValid_b:=False
   Else
      $nodrefIsValid_b:=True
End case

DOM_StopCustomErrorHandling

$0:=$nodrefIsValid_b
```

DOM_StartCustomErrorHandling

*DOM_StartCustomErrorHandling*ルーチンは既存のエラー情報を格納し、カスタムエラーハンドラをインストールします。

```
If (Undefined(Error))
   Error:=0
End if

C_STRING(80;DOM_PreviousErrorHandler_s)
C_LONGINT(DOM_PreviousValueOfError_l)
C_LONGINT(DOM_Error)
DOM_PreviousErrorHandler_s:=Method called on error
DOM_PreviousValueOfError_l:=Error
DOM_Error:=0   ` Assign a value in error method, if run.
Error:=0
ON ERR CALL("DOM_ErrorTrappingRoutine")
```

DOM_StopCustomErrorHandling

*DOM_StopCustomErrorHandling*ルーチンは、カスタムエラーハンドラをクリアし、*DOM_StartCustomErrorHandling*で格納されたエラー情報をロードします。

```
ON ERR CALL(DOM_PreviousErrorHandler_s)   ` Restore original error handler.
Error:=DOM_PreviousValueOfError_l   ` Restore original value of Error.
```

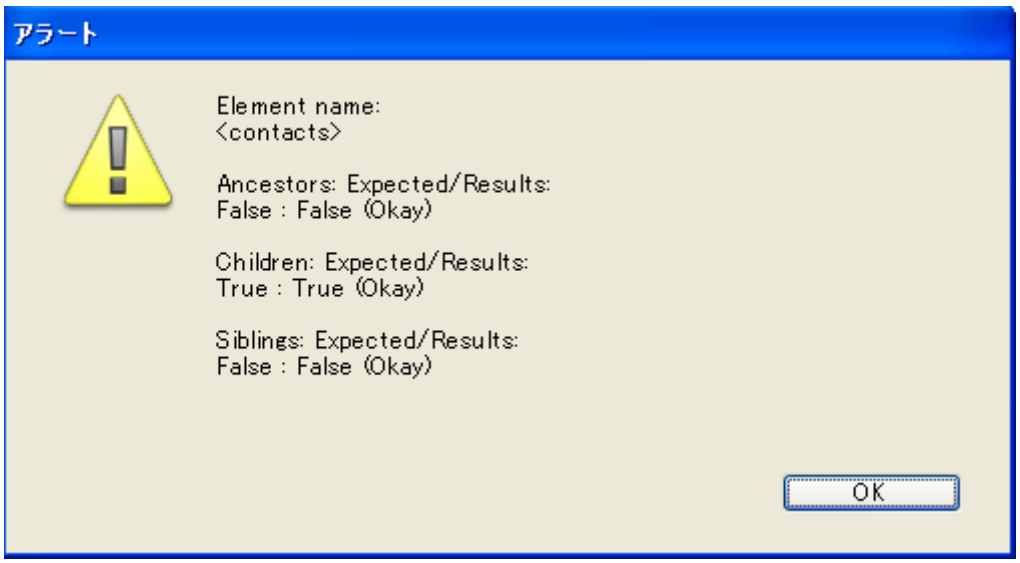
DOM_ErrorTrappingRoutine

*DOM_ErrorTrappingRoutine*メソッドは*DOM_ReferenceIsValid*と*DOM_StartCustomErrorHandling*によってインストールされ、無効なノード読み込みを試行した場合に生成されるエラーをトラップします:

```
DOM_Error:=Error
```

The Sample Database

サンプルデータベースには、今までに説明したコードと、簡単なテストルーチン*Test_NodeHasRelatives*が含まれています。テストコードは*DOM_NodeHasAncestors*、*DOM_NodeHasChildren*、そして*DOM_NodeHasSiblings*メソッドを、有効なノードや無効なノードなど様々な条件でテストするようにデザインされています。テストを行うごとに、コードは以下に示すようなダイアログに結果を表示します:



Summary

XMLを処理する際に、しばしばノードが親や子、兄弟ノードを持っているかどうかをテストすることが必要になることがあります。4th DimensionのXML読み出しコマンドには、ノードの関係を返すコマンドがありませんが、既存のコマンドを使用して簡単に実装することができます。このテクニカルノートでは、これを実行するためのコードを説明し、サンプルデータベースに完全な実装済みコードを含めました。