Cleaning Whitespace from XML Values

By David Adams

Technical Note 06-42

Overview

XML要素値は、XMLソースを読みやすくする目的で、前後にホワイトスペース文字を含みます。例えば以下のようなXMLを考えてみます:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
 <contact>
  <business>ACME Black Dot
   <phone>123 456 789</phone>
  </business>
</contact>
```

この場合、business 要素の値は何でしょう? "ACME Black Dot"という回答は正しくありません。DOM コマンドを使用した場合、完全な値は business タグに挟まれるすべてのテキストです。SAXコマンドを使用した場合、完全な値はbusiness要素の始まりから終わりまで、または次の要素の開始までで、どちらが先に出現したかによって変わります。XML のホワイトスペース文字を記述すると、以下のようになります:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
 <contact>
  <business>ACME Black Dot CARRIAGE_RETURN
SPACE SPACE SPACE SPACE<phone>123 456 789</phone>CARRIAGE_RETURN
SPACE SPACE</business>
</contact>
```

business要素の完全な値は以下のようになります:

DOMコマンド

```
ACME Black Dot CARRIAGE_RETURN
SPACE SPACE SPACE SPACE CARRIAGE_RETURN
SPACE SPACE
```

SAXコマンド

```
ACME Black Dot CARRIAGE_RETURN
SPACE SPACE SPACE SPACE
```

残念ながら 4th Dimension に含まれる **DOM GET XML ELEMENT VALUE** や **SAX GET XML ELEMENT VALUE** コマンドは、ホワイトスペースの取り除きに対応していません。幸いなことにこの足りない機能は簡単に実装することが可能であり、今回のテクニカルノートのサンプルデータベースにコードが含まれています。このノートは XML ホワイトスペースの基本的なルールと、値からホワイトスペースを取り除く方法について確認します。

XML Whitespace

XML仕様は明確に4つの文字をホワイトスペースと定めています。
(4th DimensionはUnicode文字参照をサポートしないので、Asciiコードを記載しています):
Tab 9
Line feed 10
Carriage return 13
Space 32

Designing a Trimming Function

XML のホワイトスペース削除は、概念的には単純な作業であり、いくつかの方法で実装が可能です。最も直接的なアプローチは **Substring** や **Delete string** を、要素値前後のホワイトスペースがなくなるまで繰り返しコールすることです。以下のようなソースを考えて見ましょう:

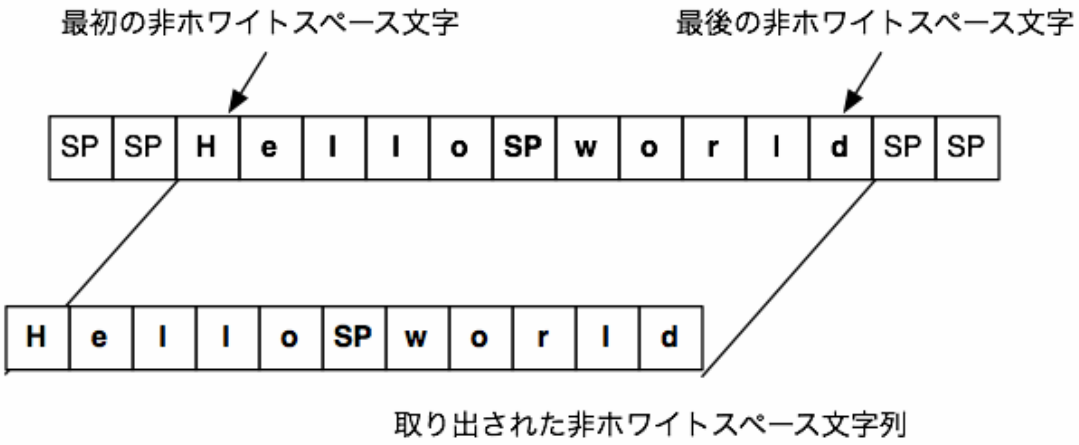
```
SPACE SPACE Hello world! SPACE SPACE
```

単純なアプローチを使用すると、文字が切り取られるたびに文字列のリサイズが発生します。この例題では 4 回です。このアプローチでは、文字列がリサイズされるたびに、4th Dimension がメモリを移動させる必要があるかもしれません。メモリの移動は起きないかもしれませんし、ほとんどの場合移動そのものが大きなパフォーマンス上の違いを生むことはありません。しかし文字列を一回だけリサイズするようなコードを書くことは難しくありません。連続的に文字列を更新する代わりに、この関数は以下の 3 ステップを行います:

1. 先頭から文字列をスキャンし、非ホワイトスペース文字を探す、
2. 最後から文字列をスキャンし、非ホワイトスペース文字を探す、
3. 非ホワイトスペース文字を取り出す

以下の図は `SPACE SPACE Hello world! SPACE SPACE` 例題の重要な部分を示して

います:



文字列の内側のスペース文字はホワイトスペースでないことに注目してください。文字列の前後にある場合のみ、これらはホワイトスペースとして扱われます。

Trimming Function Implementation

このテクニカルノートのサンプルデータベースには、*XML_CleanWhitespace* という名前の関数が含まれています。この関数は上述の 3 ステップを実装したものです。操作を単純にするため、4 つのホワイトスペースを格納するインタプロセス配列を、以下に示す *XML_InitWhitespaceCharacters* メソッドを起動時にコールすることで初期化します:

```
ARRAY STRING(1;<>XML_WhitespaceCharacters_as;4)
<>XML_WhitespaceCharacters_as{1}:=Char(Tab )
<>XML_WhitespaceCharacters_as{2}:=Char(Line feed )
<>XML_WhitespaceCharacters_as{3}:=Char(Carriage return )
<>XML_WhitespaceCharacters_as{4}:=Char(Space )
```

XML_CleanWhitespace 関数は以下のとおりです:

```
C_TEXT($0;$result_t)
C_TEXT($1;$source_t)
$source_t:=$1
$result_t:=""
C_LONGINT($firstCharacter_index)
C_LONGINT($lastCharacter_index)
```

```
$firstCharacter_index:=0
$lastCharacter_index:=0
`------------------------------------------------------
` 1) 最初の非ホワイトスペース文字を探す
`------------------------------------------------------
C_LONGINT($length)
C_LONGINT($index)
C_LONGINT($element)
$length:=Length($source_t)
$index:=0
$element:=0
C_BOOLEAN($done)
$done:=False
Repeat
  $index:=$index+1
  If ($index>$length)
    $done:=True
  Else `チェック対象文字が、ホワイトスペース配列に含まれるかテスト
    $element:=Find in array(<>XML_WhitespaceCharacters_as;$source_t$index)
    $firstCharacter_index:=$index
    $done:=True
  End if
Until ($done)

`------------------------------------------------------
` 2) 最後の非ホワイトスペース文字を探す
`------------------------------------------------------
C_LONGINT($index)
C_LONGINT($element)
$index:=Length($source_t)+1
$element:=0
C_BOOLEAN($done)
$done:=False

Repeat ` 最後の非ホワイトスペース文字を探すために、文字列を最後から検索
  $index:=$index-1
```

```
  If ($index=0)
    $done:=True
  Else
      `チェック対象文字が、ホワイトスペース配列に含まれるかテスト
    $element:=Find in array(<>XML_WhitespaceCharacters_as;$source_t$index)
    If ($element<0)
        ` 検査した文字が非ホワイトスペース文字である
      $lastCharacter_index:=$index
      $done:=True
    End if
  End if
Until ($done)

`------------------------------------------------------
` 3) 非ホワイトスペース文字列を取り出す
`------------------------------------------------------
C_LONGINT($result_length)
$result_length:=$lastCharacter_index-$firstCharacter_index+1
Case of
  ¥ ($lastCharacter_index=0)
    $result_t:=""
  ¥ ($firstCharacter_index=0)
    $result_t:=""
  ¥ ($result_length<1)
    $result_t:=""
  Else
    $result_t:=Substring($source_t;$firstCharacter_index;$result_length)
End case
$0:=$result_t
```

Additional Comments on the Implementation

XML_CleanWhitespace 関数は、<>XML_WhitespaceCharacters_as 配列の内容をチェックして、文字がホワイトスペースかどうかをテストします。コードで直接文字をテストしないのはなぜでしょう？例えば以下のようなコードを考えて見ましょう (<>Tab, <>LINE_FEED, <>CARRIAGE_RETURN, <>SPACE は起動時に初期化されているとします):

```
If ($index=0)
  $done:=True
Else

  Case of
    ¥ ($source_t$index=<>TAB)
    ¥ ($source_t$index=<>LINE_FEED)
    ¥ ($source_t$index=<>CARRIAGE_RETURN)
    ¥ ($source_t$index=<>SPACE)
    Else ` テストした文字はホワイトスペースではない
      $lastCharacter_index:=$index
      $done:=True
  End case
End if
```

上のようにハードコードされたソースの利点は読みやすいということです。他方 *XML_CleanWhitespace* のように配列を使用した場合、同じコードを他の文字列取り除きルールでも使用することができます。この方法では、配列を変更するだけで、コードはそのままにルーチンの動作を変更することができます。例えば、(制御文字など)文字列の前後から取り除きたい他の文字列リストを別な配列として宣言することも可能です。Case 文にコードを追加することで同じ結果を得ることが可能ですが、関数は一般的な文字列取り除きエンジンとしてそのままにし、新しいデータを追加するほうがより簡単です。

Summary

DOM GET XML ELEMENT VALUE, DOM Get parent XML element, DOM Get first child XML element そして **SAX GET XML ELEMENT VALUE** など、4th Dimension の XML 要素値読み込みコマンドは、文字列前後のホワイトスペースの削除を行いません。幸いなことにこの機能を加えることは簡単です。このテクニカルノートではこの取り除き機能の効果的な実装方法について説明しました。