

XSLT2.0 の新機能紹介

奥井康弘

1. はじめに

2007年1月23日、W3CがXSLT2.0をW3C勧告として公開した。

XSLTは、もともとXMLを印刷・表示用のフォーマット・オブジェクトと呼ばれるXMLタグに変換する手段として開発が始まったのだが、XMLが文書データ以外の様々なデータ表現に利用されるようになり、汎用的なデータ記述言語として位置づけられるようになったため、XMLデータ変換記述言語として切り出されたのである。そして、1999年11月16日にXSLT1.0がW3C勧告となった。

XMLがデータベースから抽出したデータを格納したり、XMLデータそのものをデータベースに格納できるようになるにつれ、XMLデータがさらに表示・印刷とは無関係な汎用データ表現として用いられるようになった。

それでもスタイルシートという言葉が表しているように、XSLT1.0は、まだまだXMLデータ変換スクリプトというよりは、XMLを順次処理してHTMLなどに置き換える単純な構造をしていた。そして、本格的なデータ加工はXMLの木構造を扱うDOM APIやSAX APIを使ったプログラミングに任せていたのである。

XSLTのメリットは、本格的なプログラムを書かなくても、XMLデータとXSLT記述データをXSLTプロセッサを使えば、簡単にXMLデータ変換を行えるという手軽さにある。XSLT1.0もかなりの程度XMLデータ変換機能を実現できるので、なるべくXSLTでXMLデータ変換を行いたいのだが、今一步のところでは機能が足りないためもどかしい思いをした開発者も多いはずである。

XSLT2.0はこのような問題を解決することができる新しい機能を多く備えている。本稿では、それらの一部を紹介する。

2. XSLT2.0 規格の主な特徴

XSLT2.0は大幅な機能拡張を行っているために、少

ない紙面ですべての新機能を説明することができないが、以下の2つの機能のみ紹介する。

- XSLT内で作成した一時的な作業ツリーへのデータ操作が可能になった

- 複数の出力ファイルを指定できるようになった

2.1 一時的な作業ツリーへのデータ操作

XSLT1.0では、XMLデータを読み込んで作成されるソースツリーから、最終的な出力データを表す結果ツリーへの変換を行う際、内部で一時的なツリーを作成し、それを変数にバインドすることができた。このような一時的なツリー構造を「結果ツリーフラグメント (Result Tree Fragment)」と呼んでいたが、これに対して、XPathの機能を使ってデータをたどる操作は行えなかった。唯一文字列化操作だけが行えた。これでは、内部で一時的なツリーを作業ツリーとみなして、それを操作することができず、XSLTの大きな制約となっていた。

XSLT2.0では、これが行えるようになったのである。これと同時にXSLT2.0規格では、「結果ツリーフラグメント (Result Tree Fragment)」という用語ではなく、「一時ツリー (Temporary Tree)」という用語が用いられるようになった。

一時ツリーのデータをXPath式で操作できることを示すために、以下のようなXMLデータを考えてみよう。

(XMLデータ: grocery.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<grocery>
  <article><name>りんご</name><price>60</price><quantity>5</quantity></article>
  <article><name>バナナ</name><price>100</price><quantity>1</quantity></article>
  <article><name>みかん</name><price>300</price><quantity>2</quantity></article>
</grocery>
```

これは、果物の購入リストであるが、この購入金額を計算するためには、個々の品物の値段 (price 要素)

と個数 (quantity 要素) を掛け合わせ、それらを合計するというデータ操作が必要である。

このとき、値段と個数を掛け合わせた値を個々の品物に対して計算し、一時的な XML データを作った後、それらを合計したい。つまり、XSLT 処理時の値の格納場所として一時的な XML データのツリーを利用するという方法である。しかし、XSLT1.0 では、そのような一時的な XML データの値を XPath 式で取り出して計算することができなかった。(XPath 式を用いることができないため、テンプレートの再帰呼び出しなど複雑な処理が必要だった)

この制限がなくなったため、XSLT2.0 では次のような XSLT 記述が行えるようになった。

(XSLT スタイルシート : grocery.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="UTF-8"
  indent="yes"/>
<xsl:template match="/" >
  <html><xsl:apply-templates/></html>
</xsl:template>
<xsl:template match="grocery" >
  <h1> 今日のお買い物 </h1>
  <table frame="border" width="100%">
    <thead><tr><th> 品物 </th><th> 値段 </th><th> 個数
</th></tr></thead>
    <tbody>
      <xsl:for-each select="article">
        <tr>
          <td><xsl:value-of select="name"/></td>
          <td><xsl:value-of select="price"/></td>
          <td><xsl:value-of select="quantity"/></td>
        </tr>
      </xsl:for-each>
    </tbody>
  </table>
  <p> 合計 : <xsl:value-of select="sum($total/subtotal)"
/> 円 </p> ← XSLT2.0 で可能となった指定
</xsl:template>
<xsl:variable name="total" >
  <xsl:for-each select="//article" >
```

```
<subtotal><xsl:value-of select="price * quantity" /></
subtotal>
</xsl:for-each>
</xsl:variable>
</xsl:stylesheet>
```

price 要素の値と quantity 要素の値を掛け合わせた結果を subtotal 要素に格納して、一時的なツリーを total という変数にバインドしているのだが、この変数 total に対して、下位の subtree 要素へアクセスするための "\$total/subtotal" という書き方が許されたので、それを数値合計の関数である sum() に渡して合計金額を得ることができる。

(結果の HTML データ)

```
<html>
<h1> 今日のお買い物 </h1>
<table frame="border" width="100%">
  <thead>
    <tr>
      <th> 品物 </th><th> 値段 </th><th> 個数 </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> りんご </td><td>60</td><td>5</td>
    </tr>
    <tr>
      <td> バナナ </td><td>100</td><td>1</td>
    </tr>
    <tr>
      <td> みかん </td><td>300</td><td>2</td>
    </tr>
  </tbody>
</table>
<p> 合計 : 1000 円 </p>
</html>
```

一時ツリーへのデータ操作が可能になったことで、XSLT はプログラミング言語としても十分使えるものとなった。これは大きな飛躍であり、XSLT を記述する開発者にとって朗報である。

2.2 複数の出力ファイル指定

XSLT1.0 では、XML データに対し変換をかけた結果、一つの出力ファイルを生成した。これに対し、XSLT2.0 では、複数の出力ファイルを生成できるよう

になっている。

次のような XML データを考えよう。

(XML データ：bookcatalog2.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <book genre=" お料理 "> 肉ジャガのおいしい作り方 </
book>
  <book genre="XML">DOM 完全解説 </book>
  <book genre=" 音楽 "> ジャズとは何か </book>
  <book genre="XML">XSLT 完全解説 </book>
  <book genre=" 音楽 "> クラシックの楽しみ </book>
  <book genre=" お料理 "> ジンギスカン最高！ </book>
</catalog>
```

書籍に関する情報を記述した XML データであるが、ジャンルに分けて別々の出力ファイルを生成するために新機能である `xsl:result-document` を使用することができる。`xsl:result-document` の内部に書かれたテンプレートで生成される XML データがそれぞれのファイルとして出力される。

(XSLT 記述：result-document-2.0.xsl)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" encoding="UTF-8"
indent="yes" name="HTML-format"/>
<xsl:template match="/" >
  <xsl:apply-templates/>
</xsl:template>
<xsl:template match="catalog" >
```

```
  <xsl:result-document href="XML 関連書籍 .html"
format="HTML-format">
    <html>
      <h1>XML 関連書籍 </h1>
      <xsl:for-each select="book[@genre='XML']">
        <p><xsl:value-of select="."/></p>
      </xsl:for-each>
    </html>
  </xsl:result-document>
  <xsl:result-document href=" 音楽 関連書籍 .html"
format="HTML-format">
    <html>
      <h1> 音楽関連書籍 </h1>
```

```
  <xsl:for-each select="book[@genre=' 音楽 ']">
    <p><xsl:value-of select="."/></p>
  </xsl:for-each>
</html>
</xsl:result-document>
<xsl:result-document href=" お料理関連書籍 .html"
format="HTML-format">
  <html>
    <h1> お料理関連書籍 </h1>
    <xsl:for-each select="book[@genre=' お料理 ']">
      <p><xsl:value-of select="."/></p>
    </xsl:for-each>
  </html>
</xsl:result-document>
</xsl:template>
</xsl:stylesheet>
```

これによって、次の3つのファイルが出力される。

(結果ファイルその1：XML 関連書籍 .html)

```
<html>
  <h1>XML 関連書籍 </h1>
  <p>DOM 完全解説 </p>
  <p>XSLT 完全解説 </p>
</html>
```

(結果ファイルその2：お料理関連書籍 .html)

```
<html>
  <h1> お料理関連書籍 </h1>
  <p> 肉ジャガのおいしい作り方 </p>
  <p> ジンギスカン最高！ </p>
</html>
```

(結果ファイルその3：音楽関連書籍 .html)

```
<html>
  <h1> 音楽関連書籍 </h1>
  <p> ジャズとは何か </p>
  <p> クラシックの楽しみ </p>
</html>
```

3. まとめ

XSLT2.0 では、これまで XSLT1.0 を書いてきた方にとっては、これまでやりたかったことができるようになったという喜びと同時に、新たに覚えるべきことも多い。しかし、何事も覚えてしまえば後は自由自在にそれを操って仕事が楽しくなるはずである。是非、XSLT2.0 にチャレンジしていただきたい。