データマッピングを用いたプログラム言語からのXMLアクセス

6W−04

西谷 康仁, 高木 渉
† (株) 日立製作所ソフトウェア事業部

1 はじめに

XML[1]が電子データ交換の標準フォーマットとして広まりつつある。

電子データ交換を目的とするデータ処理においては、データのフォーマットがXMLであることは重要ではない。XMLの構造を解析することより、XML中に存在するデータそのものを扱うことが目的である。

XMLを扱うプログラムを作成するには、JavaやC++等の言語から、Document Object Model[2]やSimple API for XML[3]といったAPIを用いることが出来る。これらのAPIでは、XMLの構造をツリーとして扱ったり、XMLをパースして現れたタグやデータをイベントとしてアプリケーションに通知することによりXMLを処理する。これにはポインタでつながれた動的データ構造を扱ったり、状態遷移を扱うことが必要で、アプリケーションにとって本質的でないコードの量が多くなり、コードの保守性が損なわれる。

我々は、XMLとCOBOLのデータ構造をマッピングすることで、COBOLでXMLを容易に扱うことの出来る処理系を開発した。XMLのデータがCOBOLのデータ構造として扱えれば、アプリケーションにとって本質的なロジックの開発に専念することができる。COBOLは、電子データ交換を実現するためのシステム構築において、現在でも需要の多いプログラミング言語であり、COBOLでXMLを扱えるようにすることで、XMLをデータ交換フォーマットとして用いるシステムの構築が短期間で可能となる。

本稿では、我々が開発したCOBOLでXMLを扱うことの出来る処理系について述べる。

2 データマッピング

データマッピングとは、XMLのデータ構造と、COBOLのデータ構造を対応づけることである。データマッピングを行なうことにより、アプリケーションでは通常のCOBOLデータを扱うのと同様にXMLを扱うことが出来る。

XMLのデータ構造を定義するスキーマとしては、

```
DTD:
<!DOCTYPE invoice [
<!ELEMENT invoice (order-no,item*)>
<!ELEMENT order-no (#PCDATA)>
<!ELEMENT item (item-name,price)>
<!ELEMENT item-name (#PCDATA)>
<!ELEMENT price (#PCDATA)>]>

DDF:
<Interface interfaceName="sample">
 <BaseElement elemName="invoice">
  <Group elemName="invoice">
   <Item elemName="order-no"/>
   <Array max="10">
    <Group elemName="item">
     <Item elemName="item-name"/>
     <Item elemName="price" type="numeric"/>
    </Group>
   </Array>
  </Group>
 </BaseElement>
</Interface>
```

図1: DTD、DDFの例

DTD(Document Type Definition)を用いる。また、データ定義用ファイル(以下, DDF)に、COBOLのデータ構造に従って、各COBOLデータ項目とDTDの要素とのデータマッピングを指定する。

DTD、DDFの例を図1に示す。DDF自体もXMLによる記述になっている。

以下データマッピングを指定するためのDDFの各要素について説明する。

(1) Interface
DDFによって定義されるXMLとCOBOLの間のインタフェース名を指定する。複数のインタフェースを定義することで、一つのCOBOLプログラムで複数の異なるXMLを扱うことが出来る。

(2) BaseElement
BaseElementは、XML中でアクセス対象となる要素を指定する。XMLへのアクセスは、BaseElementに指定した要素の単位で行われる。XMLの入出力は、BaseElement単位でプログラムで制御する。例えば、XMLドキュメント中に繰り返し回数が不明な要素がある場合に、その要素をBaseElementに指定し、要素が無くなるまで読み込みを繰り返すようなプログラムを作成することが出来る。

(3) Group
子要素を持つXMLの要素は、COBOLの集団項

XML access from programming language using data mapping.
Yasunori NISHITANI, Wataru TAKAGI
†Software Division, Hitachi Ltd.

目に対応付ける。

(4) Item
それ以上子要素を持たない末端のXMLの要素を、COBOLのデータ項目に対応付ける。XML上では単なる文字列である要素の内容を、COBOLデータ項目に対応付けたときの型を、Itemのパラメータ(属性値)として指定する。

(5) Array
XMLのDTD中で'*'や'+'の記号で繰り返しが指定された要素は、COBOLの表(配列)に対応付ける。

(6) AttrItem
XMLで、要素の属性として与えられるデータは、DDFのAttrItem要素を用いて、Itemと同様にCOBOLデータ項目に対応付ける。

以上のマッピングの他に、XMLドキュメントの入出力に関する様々な情報(入出力データ情報)を、COBOLのデータ項目に対応させることができる。本機能を用いると、XMLのDTD中で'?'付きで宣言された要素がXMLドキュメント上に実際に存在したかどうかや、入力の際の桁あふれ、不正文字等を検出することが出来る。また、DDFでArray要素で対応づけた繰り返し要素の入出力数をCOBOLデータ項目として参照できる。

3 処理系の構成

本処理系の構成を図2に示す。

本処理系は、データマッピング情報からXMLアクセスのためのCOBOLソースを生成するcblxmlコマンドと、実行時のXML入出力動作を補助するランタイムから成る。

(1) XMLアクセス用COBOLソース生成コマンド
cblxmlコマンドは、DTDと対応するDDFを入力し、COBOLデータの構造が記述されたCOPY登録集(コンパイル対象のソースに取り込まれるソース)と、与えられたデータマッピングに従ってXMLをアクセスするためのCOBOLソース(XMLアクセスルーチン)を生成する。ユーザは生成されたCOPY登録集をユーザプログラム中にCOPY文で取り込み、XMLアクセスルーチンを用いてXMLの入出力を行う。XMLアクセスルーチンは、以下の4つからなる。

- OPEN
  XMLドキュメントをオープンする。XMLドキュメントとして、ファイルまたはメモリ上に存在する文字列を指定できる。

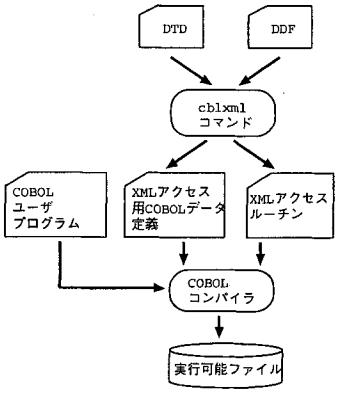

図 2: 処理系の構成

- READ
  XMLドキュメントを、BaseElement単位で入力する。
- WRITE
  XMLドキュメントに、BaseElement単位で出力する。
- CLOSE
  XMLドキュメントを閉じる。

これらのインタフェースは、COBOLの標準的なデータ入出力形式であるレコードの入出力インタフェースと似せてある。

(2) ランタイム
ランタイムは、前述のXMLアクセスルーチンから呼び出され、実際のXMLの入出力処理を行う。XMLデータとCOBOLのデータ型との間の変換もランタイムで行っている。

4 おわりに

COBOLでレコードを入出力するのと同様にXMLデータにアクセスできる処理系を開発した。XMLによるデータ交換を実現するシステムをCOBOLで短期間で構築することを可能にした。本処理系は、既存システムをXMLを利用してインターネットに接続する場合等にも適用されている。

参考文献

[1] W3C Working Group. Extensible Markup Language (XML) 1.0 (Second Edition). Oct 2000. http://www.w3.org/TR/REC-xml.

[2] W3C Working Group. Document Object Model Level 2 Core Specification. Nov 2000. http://www.w3.org/TR/DOM-Level-2-Core.

[3] Simple API for XML. http://www.saxproject.org/.