交通・電力・環境システム研究室 プログラミング演習 2008 年度　第 4 回

XML など

2008. 6. 6. 高木　亮

1　XML ファイルの書き方

1-1　XML は Extensible Markup Language の略であり，目的に応じ様々な「マークアップ言語」を定義して利用することができる仕様である。「マークアップ言語」はコンピュータ言語の一種で，文章の構造（段落など）や見栄え（フォントやそのサイズなど）に関する指定を文章とともにテキストファイルに記述するための言語である。文章に対するそれらの指定をマークアップ（markup）と呼び、マークアップを記述するための文字列をタグ（tag）と呼ぶ。

ここでは，この XML をデータ記述形式として用いることを考えてみよう。

1-2　たとえば，論文における参考文献リストを作るためのデータ形式を定義してみよう。通常，論文等における参考文献は箇条書きリストになっており，ひとつまたは複数の参考文献エントリがそれに含まれる。簡単のため，参考文献エントリは，学会誌論文のエントリと書籍のエントリとに分けられるものとしよう。

学会誌論文エントリ，書籍エントリとも，タイトルおよび著者名が必要であろう（著者については，1 名または複数名存在することを考慮する必要がある）。

学会誌論文エントリの場合，タイトル・著者名のほか，掲載誌名，巻数，号数，開始・終了ページ数，発行年月の情報が必要になる。例をあげる：

Title:	Proposal of an Energy-saving Electric Railway System
Author:	Dr Ryo Takagi
Author:	Mr Mitsuharu Oshiba
Journal:	Journal of Rail and Rapid Transit
Volume & number:	103 - 2
Pages:	27‐30
Year & month:	2008 - 6

一方，書籍エントリの場合，タイトル・著者名のほか，出版社，出版社所在地，発行年月の情報が必要になる。例をあげる：

Title:	Energy-saving Railways
Author:	Dr Ryo Takagi
Publisher:	Denkisha Kenkyukai
Address:	Tokyo, Japan
Year & month:	2008 - 6

1-3　以上のデータを XML で表現することを考えてみよう。

XML では，自分の好きなタグを定義できる。たとえば，Author（著者）を示すために，以下のようなタグを利用することができる:

```
<author>Dr Ryo Takagi</author>
```

タグ名は，<author>のように<>で括ることになっている。タグは<author>で開始，</author>で終了となるので，このタグのなかには Dr Ryo Takagi という「テキスト要素」が入っているということになる。

上記の代わりに，タグの「属性」を利用する方法もある。この例では，例えば以下のように name 属性なるものを指定させればよかろう:

```
<author name="Dr Ryo Takagi"></author>
```

この場合，タグ名が<author>である点は先ほどの例と同じだが，タグに name という「属性」が追加され，Dr Ryo Takagi というのはその属性の値として記述される。このタグには要素がないが，このような「空要素タグ」は:

```
<author name="Dr Ryo Takagi" />
```

と記述してもいいことになっている。

この<author>タグを変更し，より多くの属性を用いて次のように定義してもいい:

```
<author atitle="Dr" firstname="Ryo" familyname="Takagi" />
```

このケースに関していえば，前者（データは要素で記述）がいいか後者（データは属性で記述）がいいかは基本的には趣味の問題であるといえる。両者の主な違いとして,同一名称の属性は 1 タグあたり 1 つのみ指定できることがあげられる。たとえば，<author>タグの例において複数の atitle 属性を与えることはできない。そのような必要がある場合は，属性ではなく要素を利用せざるを得ない。

1-4　では，これにならって学会誌論文エントリの他のデータを定義していこう。

<author>タグと同様に，<title>タグ，<journal>タグ，<volume_number>タグ，<pages>タグ，そして<year_month>タグを定義すればよい。

1-2 に例示した学会誌論文エントリデータは，例えば次のように表現できる:

```
<title name="Proposal of an Energy-saving Electric Railway
  System" />
<author atitle="Dr" firstname="Ryo" familyname="Takagi" />
<author atitle="Mr" firstname="Mitsuharu"
  familyname="Oshiba" />
<journal name="Journal of Rail and Rapid Transit" />
<volume_number volume="103" number="2" />
<pages start="27" end="30" />
<year_month year="2008" month="6" />
```

同様に，書籍エントリについても，<author>タグと同様に，<title>タグ，<publisher>タグ，<address>タグ，そして<year_month>タグを定義すればよい。1-2 で例示した書籍エントリデータは，例えば次のように表現できる:

```
<title name=”Energy-saving Railways” />
<author atitle=”Dr” firstname=”Ryo” familyname=”Takagi” />
<publisher name=”Denkisha Kenkyukai” />
<address name=”Tokyo, Japan” />
<year_month year=”2008” month=”6” />
```

1-5　1-4 で示したタグ群は,「学会誌論文エントリ（または書籍エントリ）をひとつ定義するために必要なデータ」を定義するために必要なものだ。

さて，これらを利用し，学会誌論文エントリを定義するためのタグ<journal_paper_entry>または書籍エントリを定義するためのタグ<book_entry>を定義することを考えよう。それは簡単で，これらタグには属性はなく，データはすべて要素で指定され，その要素というのが 1-4 で定義したタグ群そのものである，とすればよいわけである。学会誌論文エントリであれば:

```
<journal_paper_entry>
  <title name=”Proposal of an Energy-saving Electric
    Railway System” />
  <author atitle=”Dr” firstname=”Ryo”
    familyname=”Takagi” />
  <author atitle=”Mr” firstname=”Mitsuharu”
    familyname=”Oshiba” />
  <journal name=”Journal of Rail and Rapid Transit” />
  <volume_number volume=”103” number=”2” />
  <pages start=”27” end=”30” />
  <year_month year=”2008” month=”6” />
</journal_paper_entry>
```

そして，書籍エントリであれば:

```
<book_entry>
  <title name=”Energy-saving Railways” />
  <author atitle=”Dr” firstname=”Ryo”
    familyname=”Takagi” />
  <publisher name=”Denkisha Kenkyukai” />
  <address name=”Tokyo, Japan” />
  <year_month year=”2008” month=”6” />
</book_entry>
```

とすればいい。

このように，XML のタグは別なタグの要素になり得る。従って，階層構造のデータを容易に定義し，記述することが可能となる。

1-6　さて，これでエントリの記述ができるようになったから，参考文献リストの記述の設計に移ろう。すでに述べたように，参考文献エントリは参考文献リスト中に複数存在する。そこで，「参考文献リスト」を表現するためのタグ<references_list>を定義する場合，そのなかには要素として「参考文献エントリ」を定義するタグ<reference_entry>が複数あり，それぞれの<reference_entry>タグのなかには要素として<journal_paper_entry>タグもしくは<book_entry>タグのいずれかがひとつだけ入っていると考えればよいだろう。

1-7　以上をまとめれば，参考文献リストを定義するひとつの XML ファイルが作成できることになる。その例として，references-list.xml ファイルを添付する。これを実際に入力し，セーブし，ウィンドウズのエクスプローラを用いて Internet Explorer で開いてみよ。

なお，1 行目は XML 宣言と呼ばれるものである。また，3〜10 行目および 12〜21 行目はそれぞれコメント行である。

2　TinyXML…XML ファイルを読むための簡易ライブラリ

2-1　この references-list.xml をプログラムで読むためには，ライブラリを用いるのが簡単である。ライブラリとしてはいくつかのものが存在するが，ここでは簡易なライブラリとして知られている TinyXML を用いる。

2-2　まずは，TinyXML をダウンロードする。http://www.grinninglizard.com/tinyxml/ というウェブサイトを訪れ，ここに Download the latest source release on Sourceforge. と書かれているリンクをたどり，Sourceforge ウェブサイトにある最新のソースファイル一式を入手する。

ファイルはアーカイブになっており，ファイル名は tinyxml_2_5_3.tar.gz である。これを MSYS のホームディレクトリにダウンロードする。

2-3　MSYS のコンソールを開き，以下のコマンドを入力する。これによりアーカイブが解凍され，tinyxml というディレクトリがホームの直下にできているはずである:

```
gzip -dc tinyxml_2_5_3.tar.gz | tar xfp -
```

こうしてできたディレクトリのなかにある tinyxml.cpp, tinyxml.h,

tinyxmlerror.cpp および tinyxmlparser.cpp を，一連の作業をしているディレクトリにコピーする。

2-4 コピーしたファイルのうち，tinyxml.h を編集する。オリジナルのファイルの 27 行目のすぐ下に，以下の 3 行を追加する:

```
#ifndef TIXML_USE_STL
#define TIXML_USE_STL
#endif
```

このようにすることで, TinyXML ライブラリは C++ の STL 標準ライブラリを利用するようになる。

2-5 準備ができたら，reflist1.cpp および Makefile-reflist1（どちらも添付）を入力し，コンパイルしてみよう。コンパイルのやり方は，前回資料の 4-11 を参照のこと。できあがった実行ファイルを実行する場合，1-7 で作成した references-list.xml ファイルが同じディレクトリ(フォルダ)にあることを確認すること。実行結果は長くなるので，MSYS コンソール上で

```
./reflist1 > reflist1.log
```

のように実行すると，出力が reflist.log というファイルに保存される。筆者の環境でこのコマンドを実行した結果生成された reflist.log を添付する。

2-6 プログラム reflist1.cpp は，XML ファイルを読み込み，その内容を標準出力に出力するだけのプログラムである。

まず，main()関数(148～170 行目)を見てみよう。このプログラムでは，TiXmlDocument クラスのインスタンス doc が 160 行目で宣言される。コンストラクタには XML ファイル名がわたされるが，このときはファイルを読みに行かない。実際にファイルを読みに行くのは 161 行目においてである。ファイルの読み取りに失敗した場合，162～166 行目で処理されるが，成功すれば 168 行目で dump_to_stdout()関数が実行され，読み込んだ内容が標準出力に出力される。

その dump_to_stdout()関数の定義は 87～144 行目にある。

読み込まれた XML 文書は，コンピュータのメモリ上で多数の「ノード」に分割されて保存されている。このうち，main()関数のなかで宣言・初期化した TiXmlDocument クラスのインスタンス doc は，文書全体を指し示す最上位の「ドキュメントノード」である。すべてのノードは，このドキュメントノードを最上位とするツリーのどこかに配置され，すべてのノードは木によってリンクされている。

あるノードのなかには，テキスト要素，子タグ要素，その他の要素が含まれ得るほか，属性(attributes)も含まれ得る。例えば，この文書のドキュメントノードに

は，属性はなく，3 つのノードがその直下に含まれている。それらのうちの 1 番目は references-list.xml の 1 行目にある XML 宣言(Declaration)要素，2 番目は同じく 3〜10 行目にあるコメント要素(Comment), そして 3 番目は同じく 11 行目以降の<references-list>タグに対応する。3 つのうち最後のものだけが孫ノードをもっている。

main()関数はまず doc ノードについて dump_to_stdout()関数を呼び出す。dump_to_stdout()関数は，doc ノードが持つ 3 つの子ノードについて出力をすませた後，インデントレベルをひとつ増やし，孫ノードを持つ 3 番目の子ノードについて dump_to_stdout()関数を再帰呼び出ししている。

孫ノードがさらにひ孫ノードを持っていれば, dump_to_stdout()関数はさらに再帰呼び出しされる。このようにして，dump_to_stdout()関数は親→子→孫→…，とノードの木をたどり，すべてのノードの出力を順次行っていく。

2-7　このように，TinyXML ライブラリを使うことで，プログラムは XML ファイルに記述されたデータを容易に読み込むことができる。

2-8　このプログラムで, さまざまな XML ファイルを読み込み, 出力させてみること。フォーマットの定義を変えてもよい。

当然のことだが，疑問・質問は随時高木まで（今年度は多忙なので，アポイントメントをとること！）。

以上

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2
 3 <!--
 4 ********************************************************************************
 5 Tag reference_list: a tag expressing a list of references.
 6   Attribute: none.
 7   Subdata: one or more of the following can be present.
 8     (1) Tag reference_entry: Specifies a reference entry.
 9 ********************************************************************************
10   -->
11 <references_list>
12 <!--
13 ********************************************************************************
14 Tag reference_entry: a tag expressing a reference entry.
15   Attribute: none.
16   Subdata: one of the following tags must be present. For details on how to
17     write these tags, refer to the text document.
18     (1) Tag journal_paper_entry: Specifies a journal paper.
19     (2) Tag book_entry: Specifies a book.
20 ********************************************************************************
21   -->
22   <reference_entry>
23     <journal_paper_entry>
24       <title name="Proposal of an Energy-saving Electric Railway System" />
25       <author atitle="Dr" firstname="Ryo" familyname="Takagi" />
26       <author atitle="Mr" firstname="Mitsuharu" familyname="Oshiba" />
27       <journal name="Journal of Rail and Rapid Transit" />
28       <volume_number volume="103" number="2" />
29       <pages start="27" end="30" />
30       <year_month year="2008" month="6" />
31     </journal_paper_entry>
32   </reference_entry>
33   <reference_entry>
34     <book_entry>
35       <title name="Energy-saving Railways" />
36       <author atitle="Dr" firstname="Ryo" familyname="Takagi" />
37       <publisher name="Denkisha Kenkyukai" />
38       <address name="Tokyo, Japan" />
39       <year_month year="2008" month="6" />
40     </book_entry>
41   </reference_entry>
42 </references_list>
```

```
 1 // -*- C++ -*-
 2
 3 #include <iostream>
 4 #include <iomanip>
 5 #include <cstring>
 6 #include <cstdlib>
 7 #include "tinyxml.h"
 8
 9 using std :: cout ;
10 using std :: cerr ;
11 using std :: endl ;
12
13
14 void
15 printUsage
16 ( const char * argzero )
17 {
18   cerr << "Usage:" << endl << " " << argzero << endl
19        << " No command line arguments allowed." << endl ;
20 }
21
22
23 // Copied from TinyXML tutorial ----
24
25 // Indents per space
26 const unsigned int NUM_INDENTS_PER_SPACE = 2 ;
27
28 const char *
29 getIndent
30 ( unsigned int numIndents )
31 {
32   static const char * pINDENT="                                      + " ;
33   static const unsigned int LENGTH = strlen ( pINDENT ) ;
34   unsigned int n = numIndents * NUM_INDENTS_PER_SPACE ;
35   if ( n > LENGTH )
36     n = LENGTH ;
37
38   return & pINDENT [ LENGTH - n ] ;
39 }
40
41
42 // same as getIndent but no "+" at the end
43 const char *
44 getIndentAlt
45 ( unsigned int numIndents )
46 {
47   static const char * pINDENT="                                        " ;
48   static const unsigned int LENGTH = strlen ( pINDENT ) ;
49   unsigned int n = numIndents * NUM_INDENTS_PER_SPACE ;
50   if ( n > LENGTH )
51     n = LENGTH ;
52
53   return & pINDENT [ LENGTH - n ] ;
54 }
55
56
57 int
58 dump_attribs_to_stdout
59 ( TiXmlElement * pElement ,
60   unsigned int indent )
61 {
62   if ( ! pElement )
63     return 0 ;
64
65   TiXmlAttribute * pAttrib = pElement -> FirstAttribute () ;
66   int i = 0 ;
67   int ival ;
68   double dval ;
```

```
 69   const char * pIndent = getIndent ( indent ) ;
 70   cout << endl ;
 71   while ( pAttrib )
 72   {
 73     cout << pIndent << pAttrib -> Name () << ": value=["
 74          << pAttrib -> Value () << "]" ;
 75     if ( pAttrib -> QueryIntValue ( & ival ) == TIXML_SUCCESS )
 76       cout << " int=" << ival ;
 77     if ( pAttrib -> QueryDoubleValue ( & dval ) == TIXML_SUCCESS )
 78       cout << " d=" << std :: setprecision ( 1 ) << dval ;
 79     cout << endl ;
 80     ++ i ;
 81     pAttrib = pAttrib -> Next () ;
 82   }
 83   return i ;
 84 }
 85
 86
 87 void
 88 dump_to_stdout
 89 ( TiXmlNode * pParent ,
 90   unsigned int indent = 0 )
 91 {
 92   if ( ! pParent ) return ;
 93
 94   TiXmlNode * pChild ;
 95   TiXmlText * pText ;
 96   int t = pParent -> Type () ;
 97   cout << getIndent ( indent ) ;
 98   int num ;
 99
100   switch ( t )
101   {
102   case TiXmlNode :: DOCUMENT :
103     cout << "Document" ;
104     break ;
105
106   case TiXmlNode :: ELEMENT :
107     cout << "Element [" << pParent -> Value () << "]" ;
108     num = dump_attribs_to_stdout ( pParent -> ToElement () , indent + 1 ) ;
109     switch ( num )
110     {
111     case 0 :
112       cout << " (No attributes)" ;
113       break ;
114     case 1 :
115       cout << getIndentAlt ( indent ) << "1 attribute" ;
116       break ;
117     default :
118       cout << getIndentAlt ( indent ) << num << " attributes" ;
119       break ;
120     }
121     break ;
122   case TiXmlNode :: COMMENT :
123     cout << "Comment: [" << pParent -> Value () << "]" ;
124     break ;
125   case TiXmlNode :: UNKNOWN :
126     cout << "Unknown" ;
127     break ;
128   case TiXmlNode :: TEXT :
129     pText = pParent -> ToText () ;
130     cout << "Text: [" << pText -> Value () << "]" ;
131     break ;
132   case TiXmlNode :: DECLARATION :
133     cout << "Declaration" ;
134     break ;
135   default :
136     break ;
```

```
137   }
138   cout << endl ;
139   for ( pChild = pParent -> FirstChild () ; pChild != 0 ;
140         pChild = pChild -> NextSibling () )
141   {
142     dump_to_stdout ( pChild, indent + 1 ) ;
143   }
144 }
145
146
147
148 int
149 main
150 ( int argc ,
151   char * * argv )
152 {
153   if ( argc != 1 )
154   {
155     cerr << "Error: too many arguments." << endl ;
156     printUsage ( argv [ 0 ] ) ;
157     return 1 ;
158   }
159
160   TiXmlDocument doc ( "references-list.xml" ) ;
161   bool loadOkay = doc . LoadFile () ;
162   if ( ! loadOkay )
163   {
164     cerr << "Failed to load file ¥"references-list.xml¥"" << endl ;
165     exit ( 1 ) ;
166   }
167
168   dump_to_stdout ( & doc ) ;
169   exit ( 0 ) ;
170 }
```

```
### -*- makefile -*-
### Makefile for reflist1

PROGRAM       = reflist1

NICE          =

CXX           = $(NICE) g++

CXXFLAGS      = -g -O2 -Wall
LDFLAGS       = -g -O2
CPPFLAGS      =

HDRS          = tinyxml.h

LINKER        = $(NICE) g++

LIBS          =

SRCS          = reflist1.cpp ¥
                tinyxml.cpp tinyxmlerror.cpp tinyxmlparser.cpp

OBJS          = $(SRCS:%.cpp=%.o)

DEPS          = $(SRCS:%.cpp=%.dd)

MAKEFILES     = $(DEPS)

all:            $(PROGRAM)

include $(MAKEFILES)

%.dd:           %.cpp
                $(CXX) -M $(CPPFLAGS) $< | sed 's/$*.o/& $@/g' > $@

$(PROGRAM):     $(OBJS)
                $(LINKER) $(LDFLAGS) $(OBJS) $(LIBS) -o $(PROGRAM)

clean:;         rm -f $(OBJS) *~

program:        $(PROGRAM)

###
```

```
Document
+ Declaration
+ Comment: [
*********************************************************************************
Tag reference_list: a tag expressing a list of references.
  Attribute: none.
  Subdata: one or more of the following can be present.
   (1) Tag reference_entry: Specifies a reference entry.
*********************************************************************************
  ]
+ Element [references_list]
 (No attributes)
  + Comment: [
*********************************************************************************
Tag reference_entry: a tag expressing a reference entry.
  Attribute: none.
  Subdata: one of the following tags must be present. For details on how to
   write these tags, refer to the text document.
   (1) Tag journal_paper_entry: Specifies a journal paper.
   (2) Tag book_entry: Specifies a book.
*********************************************************************************
  ]
  + Element [reference_entry]
 (No attributes)
    + Element [journal_paper_entry]
 (No attributes)
      + Element [title]
        + name: value=[Proposal of an Energy-saving Electric Railway System]
        1 attribute
      + Element [author]
        + atitle: value=[Dr]
        + firstname: value=[Ryo]
        + familyname: value=[Takagi]
        3 attributes
      + Element [author]
        + atitle: value=[Mr]
        + firstname: value=[Mitsuharu]
        + familyname: value=[Oshiba]
        3 attributes
      + Element [journal]
        + name: value=[Journal of Rail and Rapid Transit]
        1 attribute
      + Element [volume_number]
        + volume: value=[103] int=103 d=1e+02
        + number: value=[2] int=2 d=2
        2 attributes
      + Element [pages]
        + start: value=[27] int=27 d=3e+01
        + end: value=[30] int=30 d=3e+01
        2 attributes
      + Element [year_month]
        + year: value=[2008] int=2008 d=2e+03
        + month: value=[6] int=6 d=6
        2 attributes
  + Element [reference_entry]
 (No attributes)
    + Element [book_entry]
 (No attributes)
      + Element [title]
        + name: value=[Energy-saving Railways]
        1 attribute
      + Element [author]
        + atitle: value=[Dr]
        + firstname: value=[Ryo]
        + familyname: value=[Takagi]
        3 attributes
      + Element [publisher]
        + name: value=[Denkisha Kenkyukai]
```

```
  69          1 attribute
  70        + Element [address]
  71          + name: value=[Tokyo, Japan]
  72          1 attribute
  73        + Element [year_month]
  74          + year: value=[2008] int=2008 d=2e+03
  75          + month: value=[6] int=6 d=6
  76          2 attributes
```