XML（eXtensible Markup Language）と XHTML

1. XML 入門

XML とは eXtensible Markup Language の略で「拡張可能なマークアップ(印を付けた)言語」という意味を持つ。マークアップとは，本体となる文字列の中に，特定の機能を持った文字(タグ)を使って意味や機能を書き込むことを意味する。その際に重要なことは，本文の文字と機能を示す記号が，同じ種類の文字によって書かれているということである。これによって，データ本体だけではなく，その意味や機能も同時にひとつの文書中で表現することが可能となる。

従来，一般的なアプリケーションが使用するデータは，そのアプリケーションのみに対応するものであった。また，データ交換用に用いられる CSV(Comma Separated Value)はデータの構造を持たず，出現順序のみに依存する体系を持っているため，効果的な運用をするのは困難な場合があった。しかし，データが XML 文書になっていれば，記述の整合性のチェックやデータ構造を変換したいときなどに XML に対応した汎用のツールが使用できる。このことは，データが単一のアプリケーションだけに限定されたものではなく，プラットフォームの枠を越えて，多くのソフトウェアで，また多くの業務で共通して利用可能になったことを意味する。

また，アプリケーション側にとっても大きなメリットがある。たとえば，各アプリケーションで取り扱う機能が増えてグラフィックスや数式，音声などといったデータを取り扱う場合，利用者の要求に応じた形でこれを処理するためには，多くの機能を開発し提供する必要がある。しかし，データが XML 文書形式であれば，アプリケーション開発者は，独自にグラフィックスや数式や音声を表現するためのデータ形式を考え出す必要はなく汎用のものを使用すればよいだけとなる。XML でそれらのデータを記述する方法が，SVG や MathML や VoiceXML といった仕様として存在している。これらをデータ形式として流用すれば，簡単で短期間に，しかも完成度の高いものを実現することができる。

また，タグを自由に記述することができるということは，データの表現力を豊かにすることにもなった。たとえば，「赤いきつねと緑のたぬき」という表現がある場合，「赤い」「緑の」という修飾部と「きつね」「たぬき」という被修飾部の関係が重要となるが，従来の HTML などではこれを表現することが困難であった。そのため，「赤いたぬき」「緑のきつね」といった関係ないものも検索されることとなっていた。これが XML を使用して，修飾部および被修飾部それぞれに適切なタグを付与することで，確実に求める情報にアクセスすることができるようになる。XML 用の検索言語も，すでに XML の関連規格として XML でコンテンツを記述した Web サイトを対象に検索を実行する検索言語 XQL(XML Query Language)の検討が進んでいる。

このような考え方は昔から存在していた。しかし，データ量が増加することや，文字以外のデータは変換が必要になるなど，コンピュータ処理を行う上での効率が低下することから，今までは実現のための努力はほとんどはらわれてこなかったのである。

コンピュータの処理能力に余裕が生まれてきた現代においては，かつてのようにデータを１ビットでも圧縮して記録し，処理することよりも「扱いやすい」「わかりやすい」ことが重要視されるようになってきた。また，コンピュータが取り扱う範囲が増え，一部の天才的職人に依存してはニーズを満たせないほどコンピュータの利用は増加の一途をたどっている。XML は，こういう時代の背景に合致する技術として注目を集めている。

XML 規格の基本仕様は，あくまで，文書の構造を規定したものにすぎない。XML 文書は，人間も読むことができるファイルではあるが，それだけで何かができるというものではない。そこで，実際に使用するにあたっては様々な仕様が必要となる。

XML 文書の書き方を定めた XML 1.0 を中心に，さまざまな XML に関する仕様が存在し，その全体で XML の利用範囲を広げている。たとえば，XML 文書をほかのデータ形式にコンバートするスクリプト言語の XSLT，XML 文書間のハイパーリンクを記述するための XLink，HTML を XML 文法で記述した XHTML などがこれにあたる。

2. XHTML

XHTML は，eXtensible Hyper Text Markup Language の略である。XML と HTML を混ぜたようたような名前の通り HTML の代わりを XML のサブセットにさせようという仕様を意味している。すなわち，XML 文法に従って HTML を記述し直したものであり，Web ページを記述するという用途に特化した XML のサブセット仕様と言うことができる。

XHTML は HTML と同様に Web ブラウザで表示できると同時に XML 文書でもあるため,XSLT によって変換したり，XML パーサで整合性をチェックするといった XML 文書としての柔軟性も持ち合わせる。また，後述の SVG や MathML と組み合わせることで，HTML における IMG タグのように外部に構成要素が配置される形ではなく,処理を行う統一的な枠組みの中に位置づけることができる。

現在では，HTML の代わりとして XHTML で記述することも多くなってきている。現実問題としては XHTML とするための違いとしては，以下のようにすること程度で十分ともいえる。

1）タグの開始と終了関係をきちんとする。
2）終了タグの存在しないものについては，〈img src="a.gif" /〉 のように />で閉じる
3）入れ子構造を許さない
4)（これは XHTML の要件ではないが）タグ名は小文字で記述する

3. XML データ構造の例

以下は，データ構造についても記述した XML データの例である。ここでデータ構造については DTD という形式で記述されている。ここで２～８行目が DTD であるが，このデータ構造については記述をしないこともできる。データ構造を記述した XML データを「妥当な XML」，記述しない場合を「整形式 XML」と呼ぶことがある(妥当な XML は整形式 XML のサブセットとなる)。

```
 1: <?xml version='1.0'?>
 2: <!DOCTYPE 図書館目録[
 3: <!ELEMENT 図書館目録 (図書DATA)+>
 4: <!ELEMENT 図書DATA (書名,定価,著者)>
 5: <!ELEMENT 書名 (#PCDATA)>
 6: <!ELEMENT 定価 (#PCDATA)>
 7: <!ELEMENT 著者 (#PCDATA)>
 8: ]>
 9: <図書館目録>
10:     <図書DATA>
11:         <書名>XML入門</書名>
12:         <定価>1200</定価>
13:         <著者>情報太郎</著者>
14:     </図書DATA>
15:     <図書DATA>
16:         <書名>Webサービス入門</書名>
17:         <定価>3500</定価>
18:         <著者>知識次郎</著者>
19:     </図書DATA>
20: </図書館目録>
```

4. XML 宣言と文字コード

XML ファイルの第一行目には，<?xml version=“1.0” encoding=“Shift-JIS” ?>というような行が書かれる。これは XML 宣言文と呼ばれるものである。この行自体は省略可能であるが，省略した場合(や記述した場合でも encoding の指定をしなかった場合)には文字コードがユニコード(UTF-8)と解釈されてしまうため注意が必要である。Shift-JIS で記述する場合には上記のように記述する必要がある。なお，メモ帳などを使用して作成した場合，保存時に文字コードを UTF-8 に指定することもできる。

5. XML と Excel2003

Excel2003 以降では，従来の Excel2002(Excel XP)までとは異なり，XML データをそのままで処理することが可能となった。図書データや名簿など表形式で管理するのが適当なデータについては XML と Excel2003 を併用することが非常に便利である。その方法は簡単で，単純に XML データを「プログラムから開く」で Excel を指定して開けばよいだけである。その後，XML データとして読み込むかどうかを聞いてくることになる。

ただし，Excel2003 で最初から XML データを作成する方法は，まだ使いやすい形とはなっていない。将来的には Excel 上での作成も簡単に可能になることが予想されるほか，Visual Studio や Infopath などのソフトウェアを使用するという方法もあるが，現状ではメモ帳との併用がもっとも簡単である。具体的には，最初にメモ帳などを使用して２件分だけデータをタグ付けした形 XML で記述しておき，これを読み込むとよいだろう(１件だけしかデータを入力していない場合には正しく解釈されない)。読み込む XML データは Shift-JIS でもユニコード(UTF-8)でも問題がない。自動的に正しく文字コードの解釈を行ってくれる。

XML データを読み込んだ場合には，ワークシート上に青色の太枠でかこまれた表が表示される。この青枠内が XML データの表示部である。この青枠の最下段にカーソルを持っていくと，アスタリスク(*)が自動的に表示される。これが XML データの終わりを示しており，データを最後尾に追加する場合にはこの行の前に行を追加して入力していけばよい。他の Excel ファイルや CSV ファイル，テキストファイルなどから読み込んでくることも可能であるので，他のシステムからのデータの利用など応用範囲は非常に広いと考えられる。

また，上記のように行を追加することは全く問題ないが，タグを新規で作成できないのと同様に Excel2003 上では列(フィールド/XML タグ)を追加することはできないと思っていた方がよい。Excel2003 上では正しく追加されたように見えるが，うまく XML データに反映されていないことがあるためである。

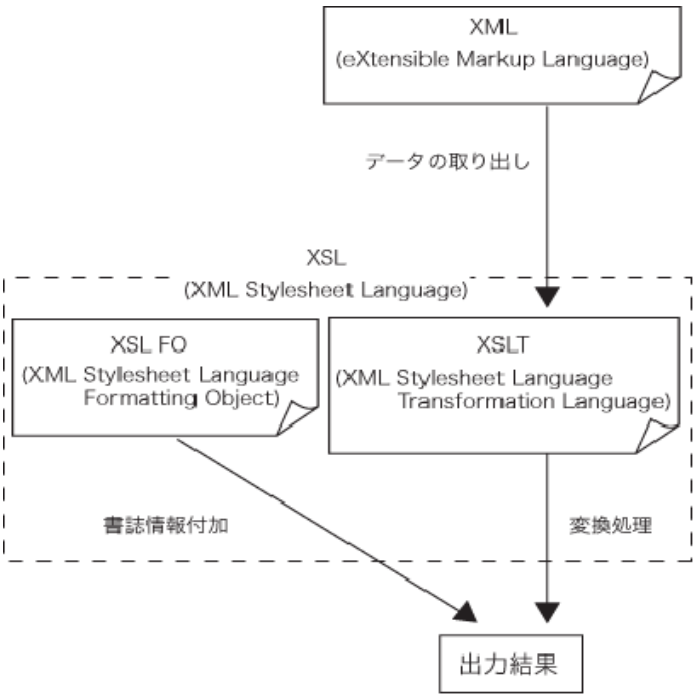
データの保存に関しては，Excel2003 の保存において XML データとして保存すればよい。この際，XML スプレッドシートというのは全く違うものであるため注意が必要となる。また，XML データを読み込んだ場合でも，そのファイル名は保持されておらず新規に名前をつける必要があることとなる。また CSV ファイルとして書き込む時などと同様に何度も Excel ブックファイルではないことの注意(Warning)が出るのでうっとうしいということもある。

さらに，作成されたファイルを次章で述べるような XSL との組み合わせで使用する場合には注意が必要である。すなわち，Excel2003 の XML データとして保存した場合には XML 宣言部に下記のように standalone="yes"属性が自動的につけられてしまうため，これをメモ帳などで取り除く必要がある。面倒だが...

〈?xml version="1.0" encoding="UTF-8" standalone="yes"?〉

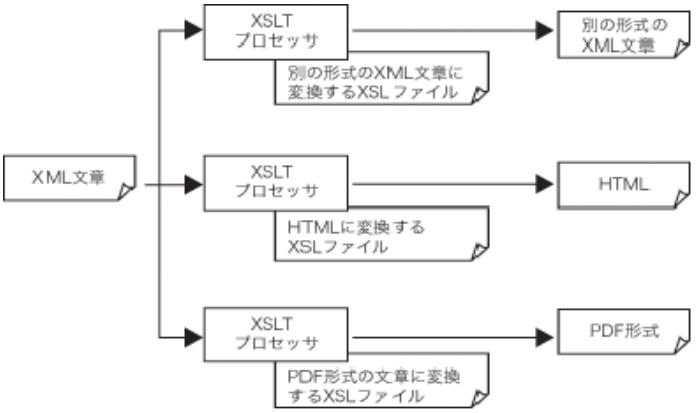
6. XML と XSL

XML はデータそのものを表現するだけであるため，その出力を整形するためには，以下のように XSL と組み合わせて使用されることとなる。



ただし，XSLFO を実際に使用している例は少なく，XML を XSLT プロセッサによって HTML に変換し，CSS を使用してスタイルの指定をすることが多い。すなわち，変換したい対象ごとに作成された XSL ファイルを元に，これを XSLT プロセッサを使って XML で記述されたデータに適用することで求める形のファイルを作成しようとするものである。たとえば，XML を HTML や XHTML に変換するという手続きを書いた XSL ファイルを用意すれば，XML ファイルを HTML/XHTML で書かれたファイルのように表示することができる。なお，この XSL ファイル自身も XML の規約に基づいて書かれている。

注意すべきことがらとしては，前にも書いたが XML 宣言において standalone="yes" が指定されている XML データファイルは XSL での元データとしては使用できないことがあるため，Excel で入力したデータなどを使用する場合には，表示する前にあらかじめメモ帳などでこの部分を削除する必要があることがあげられる。



7. HTML/XHTML ファイルから XSL ファイルへ

HTML を作成できる人であれば，XML を HTML に変換する XSLT を作成することは比較的簡単である。具体的には，まず HTML ファイルを作成し，その中でコンテンツをあらわす部分を，あたかも差し込み印刷のように指定することで実現できる。

XML データを XSL を用いて HTML に変換する場合，以下の２つの方法がある。

1）XML ファイルに直接，適用する XSL ファイル名を指定する

［具体例］

〈?xml version="1.0" encoding="UTF-8" ?〉　　のうしろに

〈?xml-stylesheet type="text/xsl" href="student04.xsl"?〉　　という行を追加する方法

2）適用パターンを指定する XML ファイルを作成し，この中で適用される XML ファイルと適用する XSL ファイルを指定する

［具体例］

XML データファイルとは別に以下のようなファイルを作成する。ここで，XML データファイルは aaa.xml, bbb.xml, ccc.xml の３つとなり，XSL ファイルは ddd.xsl である。

```
〈?xml version="1.0" encoding="UTF-8" ?〉
〈!DOCTYPE gakusei［
〈!ENTITY data1 SYSTEM "aaa.xml" 〉
〈!ENTITY data2 SYSTEM "bbb.xml" 〉
〈!ENTITY data3 SYSTEM "ccc.xml" 〉
］〉
〈?xml-stylesheet type="text/xsl" href="ddd.xsl"?〉
〈gakusei〉
&data1;
&data2;
&data3;
〈/gakusei〉
```

上記の準備をした後，XSL ファイルを作成することとなる。HTML ファイルからのごく簡単な変換手順の例を示すと以下のようになる。

1）単純に最初と最後に以下を挿入する

```
〈?xml version=“1.0” encoding=“Shift-JIS” ?〉
〈xsl:stylesheet version=“1.0”
      xmlns:xsl=“http://www.w3.org/1999/XSL/Transform”〉
〈xsl:template match=“/”〉
     ※※　ここに，もともとの HTML ファイルがはいる
〈/xsl:template〉
〈/xsl:stylesheet〉
```

2）HTML 中でコンテンツを示す部分を以下のように XML 中のタグをもとに取り込めるように指定する

データ１件の貼り込み	〈xsl:value-of select=″タグ名″ /〉
繰り返し	〈xsl:for-each select=″タグ名″〉
並べ替え	〈xsl:sort select=″タグ名″ order=″ascending″ /〉
条件判断	〈xsl:if test=″学年=2″〉
リンク先の挿入	〈xsl:attribute name=″href″〉

3）たとえば，具体例としては以下を参照してほしい。

http://www.slis.keio.ac.jp/student.xml	XML と XSL を呼び出しているファイル
http://www.slis.keio.ac.jp/slis2004.xml	XML データファイル
http://www.slis.keio.ac.jp/student.xsl	XSL ファイル

5. XML ファイルと XSL ファイルの指定

HTML を作成できる人であれば，XML を HTML に変換する XSLT を作成することは比較的簡単である。具体的には，まず HTML ファイルを作成し，その中でコンテンツをあらわす部分を，あたかも差し込み印刷のように指定することで実現できる。

XML データを XSL を用いて HTML に変換する場合，以下の２つの方法がある。

1）XML ファイルに直接，適用する XSL ファイル名を指定する

［具体例］

〈?xml version="1.0" encoding="UTF-8" ?〉　　のうしろに

〈?xml-stylesheet type="text/xsl" href="student04.xsl"?〉　　という行を追加する方法

2）適用パターンを指定する XML ファイルを作成し，この中で適用される XML ファイルと適用する XSL ファイルを指定する

［具体例］

XML データファイルとは別に以下のようなファイルを作成する。ここで，XML データファイルは aaa.xml, bbb.xml の２つとなり，XSL ファイルは ddd.xsl である。

```
〈?xml version="1.0" encoding="UTF-8" ?〉
〈!DOCTYPE gakusei［
〈!ENTITY data1 SYSTEM "aaa.xml" 〉
〈!ENTITY data2 SYSTEM "bbb.xml" 〉
］〉
〈?xml-stylesheet type="text/xsl" href="ddd.xsl"?〉
〈gakusei〉
 &data1;
 &data2;
〈/gakusei〉
```

6. HTML/XHTML ファイルから XSL ファイルへ

上記の準備をした後，XSL ファイルを作成することとなる。HTML ファイルからのごく簡単な変換手順の例を示すと以下のようになる。

1）単純に最初と最後に以下を挿入する

```
〈?xml version=“1.0” encoding=“Shift-JIS” ?〉
〈xsl:stylesheet version=“1.0”
      xmlns:xsl=“http://www.w3.org/1999/XSL/Transform”〉
〈xsl:template match=“/”〉

      ※※　ここに，もともとの HTML ファイルがはいる

〈/xsl:template〉
〈/xsl:stylesheet〉
```

2）HTML 中でコンテンツを示す部分を以下のように XML 中のタグをもとに取り込めるように指定する

データ１件の貼り込み	〈xsl:value-of select=″タグ名″ /〉
繰り返し	〈xsl:for-each select=″タグ名″〉
並べ替え	〈xsl:sort select=″タグ名″ order=″ascending″ /〉
条件判断	〈xsl:if test=″学年=2″〉
リンク先の挿入	〈xsl:attribute name=″href″〉

3）基本形

たとえば，具体例として Table で表示されていた XHTML を XSL に変更した例を示す。

［Original］（関係する〈table〉タグの部分のみ。また表頭部分は省略）

```
〈table  border="2"〉
    〈tr〉
        〈td〉Webページを用いた図書館の情報発信〈/td〉
〈td〉細野公男〈/td〉
〈td〉慶應義塾大学出版会〈/td〉
    〈/tr〉
    〈tr〉
        〈td〉図書館情報学基本論文集〈/td〉
        〈td〉上田修一〈/td〉
        〈td〉剄草書房〈/td〉
    〈/tr〉
〈/table〉
```

ここでは，各行に「タイトル」「著者名」「出版社」という３つの内容が記述されている。つまり〈tr〉～〈/tr〉の間を単位として繰り返しが指定されている。

このような繰り返し単位がどの範囲なのかを最初に見つけることが必要となる。繰り返し単位がみつかったら，その間を１セットだけ残し，前後を繰り返し処理を指定する〈xsl:for-each〉タグではさみこむ。ここで，開始タグの select 属性には，XML ファイル中で各図書に対して付与されたタグを指定する。

［XSL化後］（関係する〈table〉タグの部分のみ。また表頭部分は省略）

```
〈table  border="2"〉
    〈xsl:for-each  select="/bookdb/book"〉                ← 繰り返し開始
    〈tr〉
        〈td〉〈xsl:value-of  select="booktitle"  /〉〈/td〉   ← タイトルを示すタグ
        〈td〉〈xsl:value-of  select="author"  /〉〈/td〉      ← 著者名を示すタグ
        〈td〉〈xsl:value-of  select="publish"  /〉〈/td〉     ← 出版社を示すタグ
    〈/tr〉
    〈/xsl:for-each〉                                      ← 繰り返し終了
〈/table〉
```

たとえば,上記の例では以下のようなXMLファイルを使用することを前提としている。つまり，select 属性で指定した部分には，

"XML ファイル全体として指定したタグ/各図書を意味するために指定したタグ"

を指定する。

```
〈?xml version="1.0"    encoding="UTF-8" ?〉
    〈bookdb〉
     〈book〉
        〈author〉細野公男〈/author〉
        〈booktitle〉Web ページを用いた図書館の情報発信〈/booktitle〉
        〈publish〉慶應義塾大学出版会〈/publish〉
          〈url〉http://www.slis.keio.ac.jp/ 〈/url〉
          〈price〉 1200 〈/price〉
          〈pubyear〉2001〈/pubyear〉
       〈/book〉
                                  以下〈book〉タグの繰り返し
     〈/bookdb〉
〈/?xml〉
```

ついで，各データ内容には各図書の内容に関するタグのうち，表示したい内容のタグの名前を〈xsl:value-of select="　"〉の中に指定する。これによって，各図書のデータが繰り返し表示されることとなる。

4）並べ替えと条件判断

下のように，〈xsl:for-each〉の次に〈xsl:sort〉タグを使用して指定する。ここで，select 属性には並べ替えをする対象，order には「大きい順か小さい順か」を指定する。

```
［並べ替えと条件判断を行っている例］
    〈table border="2"〉
        〈xsl:for-each select="/bookdb/book"〉              ← 繰り返し開始
      〈xsl:sort select="price" order="ascending" /〉
      〈xsl:if test="pubyear=2001"〉
        〈tr〉
          〈td〉〈xsl:value-of select="booktitle" /〉〈/td〉   ← タイトルを示すタ
グ
          〈td〉〈xsl:value-of select="author" /〉〈/td〉      ← 著者名を示すタグ
          〈td〉〈xsl:value-of select="publish" /〉〈/td〉     ← 出版社を示すタグ
        〈/tr〉
      〈/xsl:if〉
        〈/xsl:for-each〉                                   ← 繰り返し終了
    〈/table〉
```

また，条件判断については，同じように〈xsl:for-each〉の内側に〈xsl:if test="..."〉のように指定する。ここで，〈xsl:sort /〉タグは終了タグが存在しないが，〈xsl:if〉タグは終了タグが存在していることに注意が必要である。

5）リンク先の指定

実は，XSL を用いて指定をする場合，XML ファイルに蓄積したリンク先をデータとして使用することはできない。つまり，リンク先を指定しようとしても，以下のようには指定できないのである(※印参照)。この例のような指定では〈xsl:value-of select〉タグ全体が別のタグに含まれてしまうような指定になってしまい，正しく判断されないのである。

```
［並べ替えと条件判断を行っている例］
    〈table border="2"〉
        〈xsl:for-each select="/bookdb/book"〉
        〈tr〉
            〈td〉〈xsl:value-of select="booktitle" /〉〈/td〉
            〈td〉〈xsl:value-of select="author" /〉〈/td〉
            〈td〉〈xsl:value-of select="publish" /〉〈/td〉
※          〈td〉〈a href="〈xsl:value-of select="url"〉"〉リンク先〈/a〉〈/td〉
        〈/tr〉
        〈/xsl:for-each〉
    〈/table〉
```

そこで，このような時には〈xsl:attribute name="href"〉のようなタグを使用して指定することになる。

つまり，〈a href="URL"〉という指定自体が

〈a〉〈xsl:attribute name="href"〉URL〈/xsl:attribute〉

のように指定されることになり，この URL 部分に〈xsl:value-of select="url"〉のような指定をする。

```
［並べ替えと条件判断を行っている例］
    〈table border="2"〉
        〈xsl:for-each select="/bookdb/book"〉
        〈tr〉
            〈td〉〈xsl:value-of select="booktitle" /〉〈/td〉
            〈td〉〈xsl:value-of select="author" /〉〈/td〉
            〈td〉〈xsl:value-of select="publish" /〉〈/td〉
※          〈td〉〈a〉〈xsl:attribute name="href"〉
※               〈xsl:value-of select="url"〉〈/xsl:attribute〉リンク先〈/a〉〈/td〉
        〈/tr〉
        〈/xsl:for-each〉
    〈/table〉
```