# 情報科学演習 資料 10
# XSLT スタイルシートの基本構造と反復処理

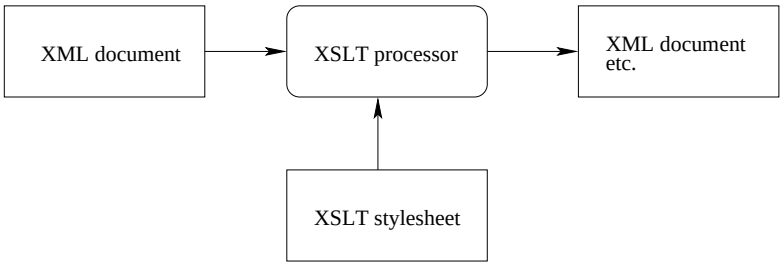
令和 8 年 6 月 29 日

## 目 次

# 1　XSLT（復習）

XSLT (eXtensible Stylesheet Language Transformation) は XML 文書を変換・加工するための言語の一つであり，XML 文書からのデータ抽出や Web ページ用文書の作成等に使うことができる。

XSLT を利用するには，XML 文書 (XML document) をどのように変換するかを書いた XSLT スタイルシートを作り，XML 文書と XSLT スタイルシート (XSLT stylesheet) を XSLT プロセッサ (XSLT processor) で処理する。




この資料で用いる「XML 文書例」として前回資料の XML 文書例[1] と，その構造を木で表現した図を以下に再掲する。

```
1   <?xml version="1.0" ?>
2   <ref>
3    <book isbn="1111">
4     <title>Title1</title>
5     <author>Author1</author>
6    </book>
7    <book>
8     <title>Title2</title>
9     <author>Author2</author>
10    <year>2020</year>
11   </book>
12   <web>
13    <title>Title3</title>
14   </web>
15  </ref>
```

```
/
|
+- ref
    |
    +- book
    |   |
    |   +- @isbn (1111)
    |   |
    |   +- title (Title1)
    |   |
    |   +- author (Author1)
    |
    +- book
    |   |
    |   +- title (Title2)
    |   |
    |   +- author (Author2)
    |   |
    |   +- year (2020)
    |
    +- web
        |
        +- title (Title3)
```

# 2　XSLT スタイルシートの基本構造

XSLT スタイルシート（以下，単にスタイルシートと呼ぶことがある）は，XML 文書をどのような形に変換するかを，XSLT で記したものである。

次のスタイルシート[2] は，`References:` と出力した後，続いて変換対象となる XML 文書から，XML 宣言やタグを除いた文字データ部分をすべて出力する[3]。

---

[1] 実習用コンピュータの /pub/eis/xml/data_model.xml
[2] 実習用コンピュータの /pub/eis/xml/allref.xsl
[3] 例えば実習用コンピュータ上で次を実行:　`xsltproc /pub/eis/xml/allref.xsl /pub/eis/xml/data_model.xml`

```
1  <?xml version="1.0"?>
2  <xsl:stylesheet version="1.0"
3                  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4    <xsl:output method="text" encoding="UTF-8" />
5    <xsl:template match="/">
6  References:
7      <xsl:value-of select="/" />
8    </xsl:template>
9  </xsl:stylesheet>
```

XSLT スタイルシート自体も整形式の XML 文書なので，スタイルシートにも XML の用語（XML 宣言や要素，属性等）や規則が当てはまる。それに加え，XSLT には独自の規則や役割を持つ要素があるので，上記のスタイルシートを元にこれらを説明する。

1 行目: `<?xml version="1.0"?>` : XML 宣言

2-3 行目:
```
<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

XSLT スタイルシートには xsl:stylesheet 要素が必要である。この要素の内容（この次の行から終了タグ `</xsl:stylesheet>` の前まで）は，XSLT の規則に従って記述しなければならず，ここに記述される xsl: で始まる名前の要素には，XSLT に固有の意味や役割がある。

xsl:stylesheet 要素の子要素を，XSLT ではトップレベル要素 (top-level element) という。トップレベル要素として記述できる要素は限られている。この資料では xsl:output と xsl:template 要素のみを取り上げる。

4 行目: `<xsl:output method="text" encoding="UTF-8" />`

xsl:output 要素は，変換結果の出力形式を定めるものである。ここでは出力形式を text 形式[4]とし，出力の文字コードを UTF-8 としている。

5 行目: `<xsl:template match="/">`

xsl:template 要素は，変換元の XML 文書の中の

1. どの部分に関して，
2. 何を出力するか，

を具体的に記述する，スタイルシートの中核をなす要素である。

1. を指定するのが xsl:template 要素の match 属性であり，ここでの属性値 / はルートノード（変換元の XML 文書全体）を指している[5]。

2. は xsl:template 要素の内容として記述する。この記述をテンプレートという。

[4] text 形式は変換結果からタグを除去 (テキストノードの内容のみ出力) し，文字データのエスケープ (変換) を行わないで出力する形式である。text の他には xml と html がある。

[5] 一つのスタイルシート内に match 属性値の異なる xsl:template 要素を複数記述することが可能であるが，この資料で扱うスタイルシートでは，属性 match=”/” の xsl:template 要素を一つだけ記述する。

テンプレートには，

a) 変換対象の XML 文書とは無関係に出力したい文字データ等

b) 変換対象の XML 文書に基づいて出力を行うための要素

を記述する。

上記のスタイルシートでは，a) に相当するのが “改行 `References:`　改行 空白” 等であり，テンプレート内に記述した文字データは，基本的には空白文字（空白や改行，タブ文字等）を含めそのまま出力される。ただし，出力の振る舞いは，前述した xsl:output 要素の method 属性の指定によって変わる。

b) に該当するものは xsl:value-of 要素である。

7 行目: `<xsl:value-of select="/" />`

前の資料で扱ったとおり，xsl:value-of 要素を用いると，変換対象となる XML 文書から，select 属性で指定した値（ロケーションパスで指定した文字列値など）を取り出すことができる。

# 3　XSLT の反復構造

本章では，先の XML 文書例から

```
Title1 Title2
```

のような出力を得るための XSLT スタイルシートの書き方を紹介する。ここで Title1 と Title2 は XML 文書例においてロケーションパス /ref/book/title で指定されるノードセットの文字列値であるが，xsl:value-of 要素を用いて `<xsl:value-of select="/ref/book/title" />` で出力されるのは，最初のノードの文字列値である Title1 のみである[6]。ノードセット内の全ノードの文字列値を出力するには，ノードセットの各ノードに対する「反復処理」が必要である。

## 3.1　XSLT における相対ロケーションパスの基点

XSLT における相対ロケーションパスの基点は，**カレントノード (current node)** と呼ばれる，現在処理中のノードである。ロケーションパスではカレントノードを .（ドット）で表す。

XSLT スタイルシートでは，テンプレート内でロケーションパスを使えるが，テンプレートでの初期のカレントノードは xsl:template 要素の match 属性によって決まる。

例えば，`<xsl:template match="/">` で始まるテンプレート内の初期カレントノードはルートノードである。従って，このテンプレートでは / と . は等価であり，`<xsl:value-of select="/" />` と`<xsl:value-of select="." />` は，どちらもルートノード (/) の文字列値を出力する。また，ロケーションパス/ref/book と ./ref/book および ref/book が表すノードセットは同じものである。

ただし，テンプレート内にカレントノードを変更する XSLT の要素があれば，その中ではカレントノードに応じて相対ロケーションパスの基点が変わる。次に紹介する xsl:for-each 要素はカレントノードを変更する要素である。

[6] XSLT Version 1.0 の場合

### 3.2 xsl:for-each 要素による反復

XSLT スタイルシートのテンプレート内に xsl:for-each 要素を置くことで，テンプレート内で反復処理を指定することができる。次に示すのは xsl:for-each を使ったスタイルシートの例である。

```
1  <?xml version="1.0" ?>
2  <xsl:stylesheet version="1.0"
3                  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4    <xsl:output method="text" encoding="UTF-8" />
5    <xsl:template match="/">
6  Books:
7      <xsl:for-each select="/ref/book">
8        <xsl:value-of select="title" />
9        <xsl:value-of select="author" />
10     </xsl:for-each>
11   </xsl:template>
12 </xsl:stylesheet>
```

これを XML 文書例に適用すると[7]，次の出力が得られる。

```
Books:
    Title1Author1Title2Author2
```

以下では，上記のスタイルシートのうち，xsl:for-each の動作に関わる部分について説明する。

5 行目: `<xsl:template match="/">`

出力を具体的に記述する部分（テンプレート）の始まり。

属性 match=”/” の指定により，ルートノード (/) がカレントノードになるため，テンプレート内の初期の相対ロケーションパスの基点は / になる。

7 行目: `<xsl:for-each select="/ref/book">`

xsl:for-each 要素は，select 属性で指定されたノードの集まり（ノードセット）の中の各ノードを逐次カレントノードにしながら，要素の内容（終了タグ `</xsl:for-each>` の前まで）に記述されたものを繰り返し出力する要素である。

XML 文書例を処理する場合，/ref/book は二つのノードに対応するので，xsl:for-each 要素の内容は 2 回反復される。

なお，この開始タグが現れた段階でのカレントノードは直前の xsl:template 要素により / であるため，select 属性値を絶対ロケーションパス /ref/book に代え，相対ロケーションパスの ref/book や ./ref/book としても同じ動作になる。

8-9 行目: `<xsl:value-of select="title" />`
`<xsl:value-of select="author" />`

xsl:for-each 要素による繰り返しの対象部分。XML 文書例に対しては，次の動作が起きる。

[7] カレントディレクトリに XSLT スタイルシート for_each.xsl と XML 文書 data_model.xml があるとすれば `xsltproc for_each.xsl data_model.xml` を実行する。

**反復 1 回目** 第 1 の /ref/book 要素 (/ref/book[1]) がカレントノードとなる。

このとき，xsl:value-of の select 属性値である title と author（相対ロケーションパスでの指定）は，絶対ロケーションパスでの/ref/book[1]/title および /ref/book[1]/author に等しい。

従って，xsl:value-of 要素によって，それらの文字列値である Title1 と Author1 が出力される。

**反復 2 回目** 第 2 の /ref/book 要素 (/ref/book[2]) がカレントノードとなる。

反復 1 回目と同様に，xsl:value-of によって，/ref/book[2]/title および /ref/book[2]/author の文字列値である Title2 と Author2 が出力される。

10 行目: `</xsl:for-each>`

xsl:for-each 要素の終了タグ。xsl:for-each 要素が終わると，カレントノードは以前のもの（このスタイルシートでは xsl:template で指定された / ）に戻る。そのため，この後に相対ロケーションパスを含むテンプレート記述があれば，その起点は再度 / になる。

## 3.3 データの並べ換えと空白文字の出力

第 3.2 節の XSLT スタイルシートにおいて，xsl:for-each 要素を

```
<xsl:for-each select="/ref/book">
  <xsl:sort order="descending"/>
  <xsl:value-of select="title" />
  <xsl:text> </xsl:text>
  <xsl:value-of select="author" />
  <xsl:text>
</xsl:text>
</xsl:for-each>
```

に変更して，XML 文書例を処理すると，出力は次のとおりとなる。

```
Books:
    Title2 Author2
    Title1 Author1
```

この出力を得るために行っているテンプレート内の処理について，以下に記す。

### 3.3.1 xsl:sort 要素による並べ換え

xsl:for-each 要素は，特に指定しなければ，XML 文書に現れた順にカレントノードを変更して反復処理を行うが，xsl:for-each の内容の始めに xsl:sort 要素を置くことで，処理の順序を変更できる。

例えば，先の `<xsl:sort order="descending"/>` は，出力される文字列値が辞書の逆順（降順）になるように，xsl:for-each が処理するノードの順序を並べ換えるものである。

xsl:order 要素の動作を指定する属性は幾つかあり，order 属性は並べ換えの順を指定するものである。属性値には ascending （昇順）か descending （降順）を指定する (order 属性を略すと昇順になる)。

他の属性としては，並べ換えに使うキーを指定する select 属性があり，`<xsl:sort select="author" order="descending"/>` とすれば，author ノードの文字列値が降順になるように処理の順序が変更される。

### 3.3.2 xsl:text 要素を使った空白文字の出力

第 3.2 節の実行結果において，空白や改行無しに Title1 や Author1 などが続いて出力されたのは，テンプレート内で，空白文字（ブランクや改行）のみからなる文字データをタグの外に書いても，それらが出力されないためである[8]。

テンプレートに記述した空白や改行を出力したければ，xsl:text 要素が使える。例えば，内容が空白文字である xsl:text 要素

```
<xsl:text> </xsl:text>
```

や

```
<xsl:text>
</xsl:text>
```

をテンプレート内に書けば，空白や改行を出力できる。

## 4 問題

1. 第 3.2 節の XSLT スタイルシートにおける xsl:value-of 要素の select 属性値 title と author を，各々/ref/book/title と /ref/book/author に変更して，XML 文書例 (/pub/eis/xml/data_model.xml) を xsltproc コマンドで処理してみなさい。その出力が得られる理由を考えなさい。

2. 参考文献の XML データ (/pub/eis/xml/ref.xml) から，title 要素の内容のみを全て出力するための XSLT スタイルシートを作成し，xsltproc コマンドで処理しなさい。xsl:for-each 要素を使うこと。xsl:text 要素を使って，適切な改行を出力すること。xsl:sort 要素の使い方も確認してみなさい。

3. 資料「XML の基礎」の 4. 問題の設問 4. で作成した郵便番号の XML 文書から，元の設問にあるような郵便番号の一覧表を作るための XSLT スタイルシートを作成しなさい。ただし，各列の標題である「市名 町名 郵便番号」の出力は省いてよい。罫線も無くてよい。

[8] 空白文字のみのテキストノードは出力されない。

# 5　発展: XML 文書から Web ページ用文書を作る

これまでの XSLT の利用目的は，主に XML 文書から必要なデータを抽出することであった。しかしながら，XSLT は XML 文書を適切な表現形式に変換する目的一般に使用されるものであるので，その一例として，この章では XML 文書を利用した Web ページ用文書作成の基本的な方法を紹介する。

## 5.1　データと表現形式

Web ページはデータを人に提示するのに適した表現形式の一つであり，XML 文書から Web ページ用の文書を作ることができれば，人がデータを閲覧するには便利である。

一方で，例えば，文書を用紙に印刷して閲覧したいこともある。もちろん Web ページをそのまま印刷して使うことも可能であるものの，Web ページはコンピュータの画面等で読むことを前提として作られているため，そのまま印刷すると用紙上の行数や文字数等が不適切になったり，ハイパーリンクが使えなくなるために読みにくい文書となることがある。かといって，Web と印刷用に，同じデータを別々に用意するのは適切ではない。複数の文書用にデータをそれぞれ用意しておくと，データを更新する際などに，各文書用のデータをそれぞれ更新する手間が生じる上に，文書間でデータの不整合が起きる可能性があるからである。データは一つの XML 文書として作成しておき，データの表現形式を定める役割を XSLT テンプレートに負わせれば，元の XML 文書だけを更新して，予め作成しておいた Web ページ作成用 XSLT テンプレートと印刷文書作成用のテンプレートを適用するだけで文書の更新が済む。

このように，XSLT 等 による XML の変換を使えば，データを格納したひとつの XML 文書を様々な形で表現することが可能になる。なお，データと表示形式を切り離して扱う考え方は，データ処理において重要である。このことは，人が読む通常の文書の作成においても同様である

## 5.2　HTML

HTML (HyperText Markup Language) は Web ページ記述用のマークアップ言語である。XML と HTML は共に SGML (Standard Generalized Markup Language) というマークアップ言語を基に開発されたものなので，両者の記法や用語は似ている。XML との違いは，使えるタグの名前が予め決められている点や，HTML のバージョンによっては開始タグのみの要素があることやタグ名に大文字・小文字の区別がないこと等である。なお，HTML はあくまで Web ページ (人間が読むもの) の記述用言語であり，XML のような汎用性のあるデータ記述言語ではない。

以下に簡単な HTML の例[9] を示す。ここでは，body 要素の中のタグ (h1, h2, a, p, table) の使い方が分かれば十分である。

```
1  <!DOCTYPE html>
2  <html lang="ja">
3    <head>
4      <meta charset="utf-8" />
5      <title>Sample HTML</title>
6    </head>
```

[9]実習用コンピュータの /pub/eis/xml/sample.html

```
 7      <body>
 8        <h1>HTML 文書のサンプル</h1>
 9        <h2>段落</h2>
10        <p>p タグは段落を作ります。</p>
11        <h2>ハイパーリンク</h2>
12        <p>外部文書へのリンクには href 属性を持つ a タグを使います
13          (例:<a href="http://echoes.hak.hokkyodai.ac.jp/db/30/">授業のページ
    へのリンク</a>)。
14          同じコンピュータ内のファイルにリンクを張るときには,
15          href 属性値にファイルのパス名が使えます。
16          同じディレクトリ内のファイルであれば, href="ファイル名" でいいです。
17        </p>
18        <h2>表</h2>
19        <p>表を作るには table タグを使います。
20          表の中に行を作るのは tr タグです。
21          行の中に何か書くには td タグを使います。行や列や表題は th タグです。
22        </p>
23        <table>
24          <tr>
25            <th></th><th>列 A</th><th>列 B</th>
26          </tr>
27          <tr>
28            <th>行 1</th><td>1A</td><td>1B</td>
29          </tr>
30          <tr>
31            <th>行 2</th><td>1A</td><td>2B</td>
32          </tr>
33        </table>
34      </body>
35    </html>
```

実習用コンピュータで HTML 文書を Web ページとして閲覧するには w3m というページャーが利用できる。ただし文字表示のみである。w3m は，コマンドラインの引数に .html で終わるファイル名を指定して実行すると (例: `w3m /pub/eis/xml/sample.html`)，自動的にそのファイルを HTML 文書として解釈して表示する。w3m を終了するには q あるいは Q をタイプする。

## 5.3 XML から HTML への変換

XSLT を使って XML 文書を HTML 文書に変換すれば，XML 文書内のデータを Web ページとして見ることができる。

以下に，そのための XSLT スタイルシートの例[10] を示す。このスタイルシートは参考文献の入った XML 文書[11] を元に HTML 文書を出力するためのスタイルシートであり，これまでに示した

[10] 実習用コンピュータの /pub/db/xml/ref2html.xsl
[11] 実習用コンピュータの /pub/db/xml/ref.xml

XSLT スタイルシートと異なるのは次の点である。

1. xsl:output 要素の method 属性値を html にすることで，xsl:template 要素の内容（以下では単にテンプレートと呼ぶ）に HTML のタグを出力させる[12]（4 行目）。
2. テンプレートに HTML の要素を記し（6 行目以降），その内容として XSLT の要素を記すことで，出力される HTML 文書に XML 文書内のデータを埋め込む（14–20 行目等）。
3. HTML の href 属性値を XML 文書から得るために，属性値テンプレート”{*location-path*}”を用いている（26 行目）。

   > XSLT スタイルシートのテンプレート記述において，タグの属性値部分に XML 文書のデータを埋め込みたければ，{ } 内にロケーションパスを記述すればよい。これを属性値テンプレート (attribute value template) という。

   下記のスタイルシートでは，`<a href="{uri}">` において，XML 文書内の uri 要素の文字列値を取得し，それを Web ページのリンク先 URI として使う。

```
1  <?xml version="1.0" ?>
2  <xsl:stylesheet version="1.0"
3    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
4  <xsl:output method="html" encoding="utf-8" />
5  <xsl:template match="/">
6  <html>
7   <head>
8    <title>Reference List</title>
9   </head>
10  <body>
11   <h1>Reference List</h1>
12   <h2>book</h2>
13   <table>
14 <xsl:for-each select="ref/book">
15    <tr>
16      <td><xsl:value-of select="@isbn" /></td>
17      <td><xsl:value-of select="title" /></td>
18      <td><xsl:value-of select="author|editor" /></td>
19    </tr>
20 </xsl:for-each>
21   </table>
22   <h2>web</h2>
23   <table>
24 <xsl:for-each select="ref/web">
25    <tr>
26     <td><a href="{uri}"><xsl:value-of select="title" /></a></td>
```

[12] これまで利用してきた `<xsl:output method="text">` は，xslt:template の内容のうち，タグ等は出力せずに文字データのみを出力する指定である。

```
27       </tr>
28    </xsl:for-each>
29      </table>
30     </body>
31    </html>
32    </xsl:template>
33    </xsl:stylesheet>
```

## 5.4 練習問題

1. 参考文献の入った XML 文書 (/pub/db/xml/ref.xml) に，第 5.3 節の XSLT スタイルシート例 (/pub/db/xml/ref2html.xsl) を適用し，結果を出力しなさい。xsltproc コマンドを使うこと。

   その出力をパイプと less を使って確認してから，リダイレクトを使ってファイルに保存しなさい。ファイル名の終り（拡張子）は .html とする。保存したファイルを w3m で表示しなさい。

2. 第 4 節の練習問題 3 で作成した XSLT スタイルシートを，HTML を出力するようにしなさい。

   xsltproc コマンドを使って，郵便番号表の入った HTML ファイルを作成し，w3m で表示しなさい。