

# 情報科学演習／データベース入門 A 資料 8

## XSLT スタイルシートの基本構造と反復処理

平成 29 年 5 月 29 日

### 目次

<b>1</b>	<b>XSLT スタイルシートの基本構造</b>	<b>1</b>
<b>2</b>	<b>XSLT の反復構造</b>	<b>2</b>
2.1	XSLT における相対ロケーションパスの基点 . . . . .	3
2.2	xsl:for-each 要素による反復 . . . . .	3
<b>3</b>	<b>データの並べ換えと空白文字の出力</b>	<b>4</b>
3.1	xsl:sort 要素による並べ換え . . . . .	5
3.2	xsl:text 要素を使った空白文字の出力 . . . . .	5
<b>4</b>	<b>練習問題</b>	<b>5</b>

## 1 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。

次の例は、前回の資料と同様の、変換対象となる XML 文書から、XML 宣言やタグ等を除いた文字データ部分をすべて出力するスタイルシートである。

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

XSLT スタイルシート自体も整形形式の 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。

- `<?xml version="1.0"?>` : XML 宣言
- `<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 の用語でトップレベル要素 (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、その中でも `xsl:output` と `xsl:template` 要素のみを取り上げる。

- `<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<sup>1</sup>とし、出力の文字コードを UTF-8 としている。

- `<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。

<sup>1</sup>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。text の他には xml と html がある。

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。テンプレートには、

1. 変換対象として与えられた XML 文書に基づいて、何かを出力するための要素や、
2. 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

を記述する。

テンプレート内に記述した文字データは、基本的にはそのまま出力される。ただし、空白文字（ブランクや改行）のみからなる文字データは出力されない。出力の細かな振る舞いは `xsl:output` 要素の `method` 属性の指定によって変わる。

なお、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、扱いが煩雑となるため、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こととする。

- `<xsl:value-of select="/" />`

前回扱ったとおり、`xsl:value-of` 要素を用いると、変換対象となる XML 文書から、`select` 属性で指定した値を取り出すことができる。

## 2 XSLT の反復構造

前回資料の XML 文書例（以下、単に「XML 文書例」と呼ぶ）と、その構造を木で表現した図を以下に示す。

```

<?xml version="1.0" ?>
<ref>
  <book isbn="1111">
    <title>Title1</title>
    <author>Author1</author>
  </book>
  <book>
    <title>Title2</title>
    <author>Author2</author>
  </book>
  <web>
    <title>Title3</title>
  </web>
</ref>

```

```

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

```

本章では、この XML 文書例から

```
Title1 Title2
```

のような出力を得るための XSLT テンプレートの書き方を紹介する。ここで `Title1` と `Title2` はロケーションパス `/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" />` としても、ノードセット内の最初のノードの文字列値 (`Title1`) しか得られない。ノードセット内の全ノードの文字列値はノードセットに”反復処理”を施すことで得ることができる。

## 2.1 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り、これを . (ドット) で表す。

XSLT スタイルシートでは、ロケーションパスをテンプレート内で使うが、その際の (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="." />` の出力は同じである。また、ロケーションパス `/ref/book` と `./ref/book` および `ref/book` は等価である。

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<sup>2</sup>。

## 2.2 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。

```
<?xml version="1.0" ?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text" encoding="UTF-8" />
  <xsl:template match="/">
    Books:
    <xsl:for-each select="/ref/book">
      <xsl:value-of select="title" />
      <xsl:value-of select="author" />
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

これを XML 文書例に適用すると<sup>3</sup>、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。

- `<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。

<sup>2</sup>`xsl:for-each` 要素がこれに該当する。

<sup>3</sup>カレントディレクトリに XSLT テンプレート `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。

- `<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、終了タグ `</xsl:for-each>` の前まで（要素の内容）に記述された処理を繰り返して行うものである。

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 回反復処理される。

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 属性値を相対ロケーションパスの ref/book や ./ref/book としても動作は変わらない。

- `<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。

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。

- `</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 データの並べ換えと空白文字の出力

第 2.2 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。

### 3.1 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、処理順序の並べ換えが行われる。

### 3.2 xsl:text 要素を使った空白文字の出力

第 2.2 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。

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 練習問題

- 第 2.2 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 テンプレートを作成し、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 5. XML の基礎」の 4. 練習問題の 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を作るための XSLT テンプレ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。