情報科学演習 資料 8
XML の基礎

令和 8 年 6 月 15 日

目 次

1 XML

XML とは拡張可能なマーク付け言語 (eXtensible Markup Language) のことであり，XML で記述されたデータを **XML 文書 (XML document)** という。すなわち XML はデータの記述形式を定める規約であり，XML 文書が個々のデータのことである。

XML は広く普及しており，多くのアプリケーションが XML 文書としてデータをファイルに保存する。また，XML は Web 技術との親和性が高く，Web でデータを交換する際の標準的なデータ記述言語となっている[1]。

この資料では，簡単な XML 文書を作成するために必要な XML 文書の構造や記述の規則に関する基礎的事項を紹介する。

1.1 XML 文書例

XML 文書はテキスト形式の文書（データ）である。XML 文書では，`<title>...</title>` 等の**タグ**でデータを囲むことにより，データに意味情報を付与することができる。また，タグで囲まれた部分に，さらにタグを含めることで，データの階層的な構造記述も可能である。

XML 文書の例として，この資料の参考文献データを格納した XML 文書を示す[2]。

```
1  <?xml version="1.0" encoding="UTF-8" ?>
2  <!-- 参考文献データ -->
3  <ref>
4   <web>
5    <title>Extensible Markup Language (XML) 1.0 (Fifth Edition)</title>
6    <uri>http://www.w3.org/TR/xml/</uri>
7   </web>
8   <web>
9    <title>Translations of Current W3C Technical Reports</title>
10   <uri>http://www.w3.org/Translations/</uri>
11  </web>
12  <book isbn="4-7561-4584-1">
13   <title>はじめて読む XML</title>
14   <author>株式会社日本ユニテック Web 技術研究グループ</author>
15   <publisher>アスキー</publisher>
16   <year>2005</year>
17  </book>
18  <book isbn="4-534-03620-5">
19   <title>すっきりわかる XML</title>
20   <author>西村めぐみ</author>
21   <publisher>日本実業出版社</publisher>
22   <year>2003</year>
23  </book>
```

[1] 類似の用途に使われる，より簡易なデータ記述形式として JASON (JavaScript Object Notation) や YAML (YAML Ain't Markup Language; Yet Another Markup Language) がある。

[2] 行頭の数字は行番号であり，XML 文書の一部ではない。

```
24    <!-- 空要素の例 -->
25    <book isbn="1-55860-622-X" />
26   </ref>
```

1.2 XML パーサ

XML 文書はテキストデータなので，ファイルに格納された XML 文書を grep 等の UNIX コマンドで処理することも可能ではあるものの，通常，XML 文書の構造解析や処理を行うには XML パーサまたは XML プロセッサと呼ばれるプログラム（の部品）を使う。

例えば，多くの Web ブラウザは XML 文書を読み込んで，その構造を階層的に表示することができる。これは，Web ブラウザに XML パーサが組み込まれているために行えることである。

2 XML 文書の構造

2.1 XML 文書の構成要素

この節では，XML 文書の主要な構成要素を，第 1.1 節の XML 文書例を基に説明する。

2.1.1 XML 宣言

第 1.1 節の XML 文書例における，

```
<?xml version="1.0" encoding="UTF-8" ?>
```

を **XML 宣言 (XML declaration)** という。

XML 宣言は，その文書が XML 文書であることを明示するものである。XML の仕様では,「そのバージョンを含んだ XML 宣言で XML 文書を始めるべき (should)」とされている。

XML 宣言では，その文書で用いられている文字の符号化方式 (文字コード) 等も指定でき，上記の `encoding="UTF-8"` は符号化方式が UTF-8 であることを示している。

なお，XML 宣言や，宣言内の encoding=”...” を省略しても，文字エンコーディングは UTF-8 または UTF-16 とみなされる。この指定が実際に用いられている文字コードと異なると，XML 文書を処理する際にいわゆる文字化け等の不具合が起きる[3]。

XML 宣言の記述位置は XML 文書の冒頭でなければならない。そのため，次に紹介するコメントも，XML 宣言の前には書くことはできないので注意すること。

2.1.2 コメント

XML 文書では，データとして扱わない部分を**コメント (comment)** にできる。コメントは `<!--` と `-->` で囲んで記述する。

第 1.1 節の XML 文書例では

```
<!-- 参考文献データ -->
```

[3] Windows では日本語文字コードとして Shift_JIS が標準的に使われてきた。そのため，Windows 上で，日本語を含む XML 文書に不具合があれば `encoding="Shift_JIS"` とすることで解消する可能性がある。

はコメントである。

コメント内に -- を書くことはできない。そのため，コメントを入れ子にすることは許されない。また，XML 宣言の前にはコメントも書けない。

2.1.3 要素

XML 文書内の，タグに囲まれた

```
<title>Extensible Markup Language (XML) 1.0 (Fifth Edition)</title>
```

などを**要素 (element)** という。ここで，title は**要素の名前 (name)** すなわち**要素名**であり，Extensible Markup Language (XML) 1.0 (Fifth Edition) は**要素の内容 (contents)** である。なお，タグ自体も要素の一部である。

要素

<*name*>*contents*</*name*>

において，<*name*> を**開始タグ (start-tag)** といい，</*name*> を**終了タグ (end-tag)** という。*contents* には，さらに要素を含めることができる。その場合，要素は正しく入れ子になっていなければならない。

入れ子になった要素同士の関係を，親や子などの用語で表現する。第 1.1 節の XML 文書例には web 要素が複数存在するが，何れの web 要素も ref 要素の子であり，逆に ref 要素は web 要素の親である。また，title 要素は ref 要素の子孫であり，ref 要素は title 要素の祖先である。

XML 文書には，**ルート要素 (root element)** と呼ばれる唯一の要素が必要である[4]。ルート要素は他の全要素の祖先となる要素であり，ルート要素以外のすべての要素はルート要素の開始タグと終了タグの間に入る。先の XML 文書例では<ref> から </ref> までがルート要素である。

内容 *contents* が空である要素

<*name*></*name*>

を**空要素 (empty element)** という。通常，空要素 <*name*></*name*> を

<*name* />

と書く。

2.1.4 属性

開始タグには一つ以上の**属性 (attribute)** を含めることができる。

例えば，<book isbn="4-7561-4584-1"> における isbn="4-7561-4584-1" は book 要素の属性であり，isbn を**属性名 (attribute name)**，4-274-13285-4 を**属性値 (attribute value)** と呼ぶ。属性値は " (二重引用符) または ’ (単一引用符) で囲む。

開始タグに複数の属性を記述するには，それらを空白文字で区切る。ただし，一つの開始タグ内に，同じ名前の属性を複数記述することはできない。

[4] ルート要素を文書要素 (document element) と呼ぶこともある。

2.2 整形式 XML 文書

XML の規則に従って正しく記述された XML 文書を**整形式 (well-formed) 文書**と呼ぶ。整形式でなければ，XML 文書でない。

xmllint というコマンドが使える環境では，

```
xmllint file
```

によって，*file* が整形式であることを検証できる。整形式であれば XML 文書が出力され，整形式でなければエラーが出力される。

3 文書型宣言と XML 文書の妥当性

XML 文書を作成する際には，要素名や属性名等を基本的に自由に決めてよい。ただし，複数の人やアプリケーションで共有される XML 文書の場合，用いる要素の名前等に一定の規則を設けるのが望ましい場合がある。

そのような記述規則を XML 文書に与えるには，**文書型宣言 (document type declaration)** を使う。文書型宣言に則った文書を**妥当 (valid)** な XML 文書という。

次に示すのは DTD (document type definition) という文法で記述した文書型宣言の例である。

```
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
]>
```

この文書型宣言は次のことを規定している。

- `!DOCTYPE greeting` : ルート要素名が greeting である
- `[ <!ELEMENT greeting (#PCDATA)> ]` : 文書内の要素は greeting のみで，その内容は文字データのみ (`#PCDATA`) である

第 1.1 節の参考文献 XML 文書のための DTD は例えば次のように書ける。

```
<!DOCTYPE ref [
  <!ELEMENT ref (book | web)*>
  <!ELEMENT web (title, uri)>
  <!ELEMENT book (title, author, publisher, year)?>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT uri (#PCDATA)>
  <!ELEMENT author (#PCDATA)>
  <!ELEMENT publisher (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
  <!ATTLIST book isbn CDATA #IMPLIED>
]>
```

この DTD について，ここで詳細は扱わないものの，次のような意味を持つ[5]。

[5] , 区切りは順序の拘束あり，| 区切りは順序の拘束なし，*: 0 回以上の出現，?: 0 回または 1 回の出現，+: 1 回以上の出現（資料では使っていない）

- `<!ELEMENT ref (book | web)*>` : ref 要素の内容は book 要素あるいは web 要素の 0 回以上の繰り返しである。
- `<!ELEMENT web (title, uri)>` : web 要素の内容は title 要素一つに続き uri 要素一つである。
- `<!ATTLIST book isbn CDATA #IMPLIED>` : book 要素には isbn 属性があり，属性値は文字データ (`CDATA`) である。ただし book 要素に isbn 属性は無くてもよい (`#IMPLIED`) 。

文書型宣言を XML 文書に含める場合，記述場所はルート要素よりも前としなければならない。次に示す XML 文書は，文書型宣言の規定に適合するため，妥当である。

```
1  <?xml version="1.0" ?>
2  <!DOCTYPE greeting [
3    <!ELEMENT greeting (#PCDATA)>
4  ]>
5  <greeting>Hello, world!</greeting>
```

XML 文書の妥当性は，

`xmllint --valid` *file*

で検証できる。ここで *file* は XML 文書の入ったファイルの名前である。さらに，

`xmllint --valid --noout` *file*

とすれば，整形した XML 文書の出力無しに，エラーメッセージのみを出力できる。

4 問題

以下の問題では，エディタを用いて XML 文書を編集あるいは作成すること。XML 文書のファイルを作成する際には，ファイル名を `.xml` で終る名前（拡張子を .xml）にすることを勧める。なお，設問の括弧内に示した XML ファイルは，実習用コンピュータのディレクトリ/pub/eis/xml にあるので，自分のホームディレクトリ（の配下の例えば eis26）にコピーして利用するとよい。

1. 次の文書 (corrupt.xml) が整形式でないことを xmllint コマンドで確認し，エラーメッセージに従って整形式にしなさい。

```
<?xml version="1.0" ?>
<page>w3c<uri>http://www.w3.org/TR/xml/</uri>
<uri>http://www.w3.org/MarkUp/</page></uri>
```

2. 次の文書 (not_well_formed.xml) が整形式の XML 文書であるかどうかを xmllint コマンドで検証してみなさい。文書に適当なものを追加して，整形式にしなさい。

```
<?xml version="1.0" ?>
<greeting>Hello, world!</greeting>
<greeting>Good afternoon, sir.</greeting>
```

3. 次の XML 文書 (invalid.xml) の妥当性を xmllint コマンドで検証し，妥当な文書に直しなさい。

```
<?xml version="1.0" ?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
]>
<GREETING>Hello, world!</GREETING>
```

4. 次の表を基に XML 文書を作成しなさい。

市名	町名	郵便番号
hakodate	aoyagi-cho	0400044
hakodate	akagawa	0410804
hakodate	(自宅の町名)	(自宅の郵便番号)
otaru	aioi-cho	0470028
otaru	akaiwa	0470046

文書内の要素は以下の指定に従うこと。

要素名 postcode, city, town の三つのみとする。各要素の用途は次のとおり。

- postcode: ルート要素
- city: 市名用 (ただし市名は属性として記述)
- town: 町名・郵便番号用 (ただし町名は属性として記述)

要素の入れ子関係

- city 要素は postcode 要素の子要素
- town 要素は city 要素の子要素

各要素に与える属性

- postcode 要素: 属性無し
- city 要素: 属性名 name, 属性値 市名
- town 要素: 属性名 name, 属性値 町名

各要素の内容

- postcode と city 要素は，要素以外の内容を持たない
- town 要素は郵便番号を内容とする

上記の指定に基づく文書型宣言 (DTD) を以下に示す。この DTD を，作成した XML 文書に含めれば，xmllint コマンドで XML 文書の妥当性を確認できる。

```
<!DOCTYPE postcode [
 <!ELEMENT postcode (city)*>
 <!ELEMENT city (town)*>
 <!ELEMENT town (#PCDATA)>
 <!ATTLIST city name CDATA #REQUIRED>
 <!ATTLIST town name CDATA #REQUIRED>
]>
```