Gittok Lecture Note

04 XML入門

太田守重

2014

地物や関連のインスタンスは、XML文書として外部に提供される。

ではXMLとは何か
どのように記述するのか、
その妥当性はどのように検証するか
gittok で使用するXML文書の記法はどうか

XML (eXtended Markup Language)

XML は，マークアップ (markup) またはタグ (tag) と呼ばれるシンボルでデータに意味付けし，構造化することによって，プログラムが，必要な情報を検索したり，切り出したりすることを可能にする，情報記述の規則である.

XML を使って表現したデータの集りはXML文書と呼ばれる.
マークアップによって，文書は要素に分割される.
要素はさらに下位の要素をもつことができ，これによって，要素の順序や階層構造を示す.
XML文書は，下位の要素群をもつ最上位の要素と，文書の先頭に記述される省略可能なメタ情報からなる.

XML文書の例 (book.xml)

```
<?xml version=“1.0” encoding="UTF-8"?>
<!DOCTYPE book SYSTEM “/dtds/book.dtd”>
<book classification=“Geospatial Technology” >
        <title>Introduction to gittok</title>
        <authorName>
                <first>Morishige</first>
                <last>Ota</last>
        </authorName>
        <authorName>
                <first>Taro</first>
                <last>Yamada</last>
        </authorName>
        <publication>2013-10-1</publication>
        <cover imgref=“gittokCover.jpg”/>
</book>
```

XML宣言<? … ?>
XMLのバージョンや文字符号
の種類等が分かる.

文書型宣言<! … >
文法を記述する文書 (この場合は
DTD: Document Type Declaration)
への参照
<! … >は宣言用のマークアップ

ルート要素
<book …>は全ての要素を束ねる要
素で，ルート要素と呼ばれる.

要素の識別や性質を示す属性は，
属性名＝”…”と表現する.

<cover …/>は空要素と呼ばれ，要
素のデータはない.

XML文書の形式を備えている文書は
整形式のXML文書と呼ばれる.

XML文書の妥当性検証

DTD (Document Type Declaration)

要素やその他のマークアップに関する規則の宣言

要素の宣言は，要素の中に何がどの順序で含まれるかを示す，内容モデル(content model)の定義で行う.

XMLスキーマ (XML Schema)

DTDより詳細な，データのパターンを指定したり，より柔軟な記述法を規定したりできる.

文書型宣言

<!DOCTYPE book SYSTEM “/xml-resource/dtds/book.dtd”>

<!DOCTYPE	文書型宣言の開始
book	対象となる要素名
SYSTEM	DTDの識別子
	SYSTEMは，ローカルのファイルとしてDTDが保存されることを示す．
	PUBLICの場合は，公開されるURLにDTDが置かれることを示す．
“….”	DTDが保存される場所とDTDのファイル名を示すURL

ところで，

<!-- …. -->

で囲まれた中はコメントになる．

DTD

DTDの役割

記述できる要素の一覧を示す.

それぞれの要素の内容モデルを示す.

それぞれの要素の下位にくる要素の一覧を示す.

要素の宣言には名前，データ型を示し，デフォルト値，その要素が必須か任意かを示すことができる.

DTDで使われる要素

<!ELEMENT … >　　要素の内容モデル

<!ATTLIST …>　　属性の宣言

DTDの例

```
<!ELEMENT book (title, authorName+, publication, cover)>
<!ATTLIST book classification CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT authorName (first, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT publication (#PCDATA)>
<!ELEMENT cover EMPTY>
<!ATTLIST cover imgref CDATA #IMPLIED>
```

DTDの問題

データ型の種類が少ないので，要素や属性のデータの厳密な定義ができない．また，複数の名前空間（次頁参照）を使用することができない．

bookは4種類の下位要素をもつ．その内authorNameは１つ以上ある (+).

book要素はclassificationという属性をもち，それは文字列で必須の属性である．

title要素は文字列である．

aouthorName要素は２種類の下位要素をもつ．

first要素もlast要素も文字列．

publication要素は文字列．DTDでは詳細なフォーマットまでは定義できない．

cover要素はデータをもたない空要素

ただし，imgref という属性をもち，それは文字列で，省略可能 (#IMPLIED).
必須にする場合は#REQIREDを使う．

名前空間（name space）

要素名と属性名をグループに割り当てるための仕組み．
グループは接頭辞によって示される．例えば以下のタグの属性は名前空間宣言といわれ，

<catalogue xmlns:kw="http://www.knitware.com/" ...>

xmlnsが名前空間の宣言であることを示し．kwが名前空間の識別子，名前空間を示すURIは、全世界でユニークな名前を確保するために使用され、URIの示す先に何かの情報が存在する必要は全くないし，情報があることを期待すべきでもない．文書中，この名前空間で定義されている名前には kw: という接頭辞が付く．

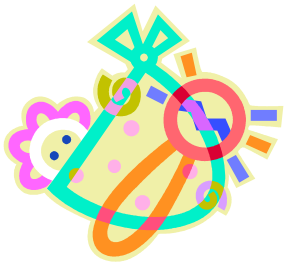
複数の名前空間を使って複数のXMLの語彙（要素名や属性名の集り）を一つの文書中で組み合わせて用いることができる．例えば，

<catalogue xmlns:kw="http://www.knitware.com/"
xmlns="http://www.kidware.com/">

というタグで，最初の名前空間はkwという接頭辞を用いるが，2つめの名前空間の名前は接頭辞を使わないタグの名前空間を示す．

名前空間を使用したXML文書の例

```
<catalogue xmlns:kw="http://www.knitware.com/"
                    xmlns="http://www.kidware.com/">
    <kw:entry kw:number="1235">
        <kw:description>幼児用のセータ</kw:description>
    </kw:entry>
    <entry number="A5647">
        <name>ポンポン</name>
    </entry>
</catalogue>
```



XMLスキーマ

DTDの弱点を改善し，型の定義が厳密にできる．

XSLスキーマの文書自体がXML文書であり，整形式チェックや妥当性の検証ができる．

DTDよりも記述は長くなる．

単純なXMLスキーマの例

xsの名前空間

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="title" type="xs:string"/>
</xs:schema>
```

文字列をデータ型とする要素titleの宣言

XMLスキーマの単純データ型（例）

型名	使用法
xs:string	文字列
xs:token	空間で区切られたトークン（単語や記号）の列
xs:QName	名前空間で修飾された名前
xs:decimal	任意の10進数
xs:integer	整数
xs:float	32ビットの浮動小数点数
xs:ID, xs:IDREF, xs:IDREFS	IDはユニークな識別子．IDREFはIDによる別の要素への参照．IDREFSは空白で区切られたIDREFのリストを使った別の複数の要素への参照
xs:boolean	ブール値trueまたはfalse
xs:time	ISO8601 に示された書式(HH:MM:SS-時差)による時刻．日本の場合例えば，10:23:32-09:00
xs:date	ISO8601に示された書式(CCYY-MM-DD) による日付.日本の場合例えば，2013-06-12
xs:dateTime	ISO8601に示された書式(CCYY-MM-DDTHH:MM:SS-時差)による日時．日本の場合例えば，2013-0-12T10:23-32-09:00

複合データ型を含むスキーマの宣言（例）

book.xml のXMLスキーマ文書

```
<?xml version="1.0" encoding="utf-16"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="title" type="xs:string" />
        <xs:element name="authorName" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="first" type="xs:string" />
              <xs:element name="last" type="xs:string" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="publication" type="xs:date" />
        <xs:element name="cover">
          <xs:complexType>
            <xs:sequence/>
            <xs:attribute name="imgref" type="xs:anyURL" use="optional" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="classification" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

bookは複数の要素からなるので，そのデータ型は複合型(xs:complexType)である．また，子要素の順番が決まっているので，
<xs:sequence>が設定される．
bookに含まれる子要素は，
1. 文字列型をとるtitle
2. 重複を許す(unbounded)要素autherName
3.日付型をとるpublication
4. 画像の保存場所を属性にするcover
である．その内，
autherNameは文字列型をとるfirst,lastを子要素にとり，
coverは子要素をもたず，URLを示すxs:anyURLを型とするimgrefを属性とする．この属性は任意(optional)である．
bookは文字列型を型とするclassificationを属性とする．この属性は必須(required)である．

XMLスキーマの問題と対応

多くの機能をつめこんだため、仕様が複雑で巨大

XML Schema の仕様を完全に実装した処理系の開発が困難

初学者にとって XML スキーマ文書の読み書きが難しい

そこで，gittokでは

DTDやXML Schemaは使用せず，処理系の中で，型ごとにXML文書の入出力操作(setXML，getXML)を用意することによって，妥当性検証を省略している.

gittokにおけるXML表現の基本方針

UMLクラス図で示されたスキーマのインスタンスをXMLで記述するとき，**gittok** では以下の基本的な方針のもとで行う.

１．クラス（型）はXML要素になる.
２．クラス（型）の属性は，XML要素の属性になる.
３．２つのクラス間の関連は，役割名または関連クラスを介して行う.
４．応用スキーマの中で，継承関係の親は inheritance 要素の子要素とする.

属性のデータ型が単純データ型の場合

クラス図で単純データ型をもつインスタンスの属性は，XMLの属性として表現する．XML属性は，開始タグの中で．属性名＝“属性値”という形式で示す。

例：〈Employee id=“e01” name=“木下なおこ” position=“総務課長”/〉

Employee
name: String position: String

idってなに？

gitto では，XML要素には自動的に，他と区別する識別コードが自動でふられることにしている．

単純データ型をとる属性の配列を表現する場合は，カンマで区切られた属性値の列で表現する．

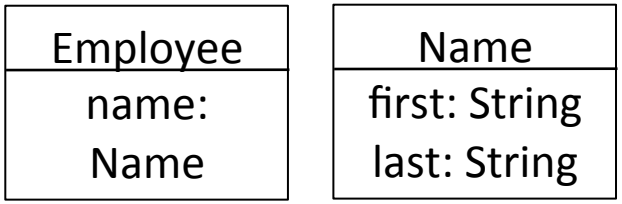
例：〈GroupMembers id=“g10” name=“斉藤洋,富山翼,春日美代子”/〉

GroupMembers
name[0..*]:String

属性の型が複合データ型になる場合

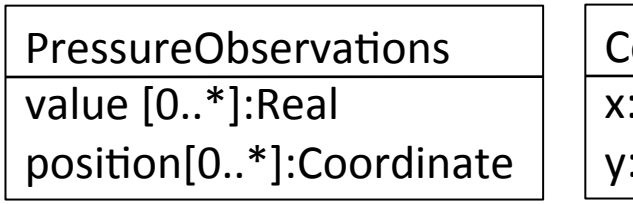
属性が複合データ型を取る場合は，その属性の名前を役割名とする関連として表現する.

例： <Employee>
　　<name idref="n01">
</Employee>
・・・・・・・・・
<Name id="n01" first="健治" last="相田">



複合データ型をとる属性が配列なるときは、参照はidの列で表現する。

例： <PressureObservations id="p23" value="1000, 998, 983">
　　<position idref="c01, c03, c10" />
</PressureObservations>
・・・・・・・・・
<Coordinate id="c01" x="10.2" y="987.3"/>
<Coordinate id="c03" x="23.5" y="855.9"/>
<Coordinate id="c10" x="50.0" y="723.6"/>

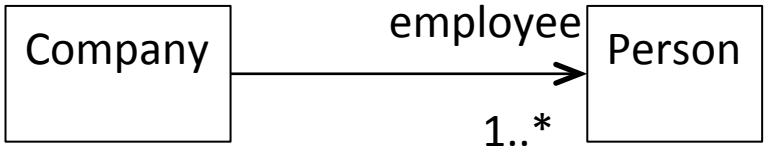


複数ならidrefs
じゃないの？

gittokでは単数と複数は区別しない.

役割名を使った関連の表現

例えば,




```
<Company id="c02", …>
    <employee>
        <Person id="e01", …/>
        <Person id="e02", …/>
        ・・・・
    </employee>
</Company>
```

または

```
<Company …>
    <employee idref="e01, e02, …"/>
</Company>
・・・・・
<Person id="e01", …/>
<Person id="e02", …/>
・・・・・
```

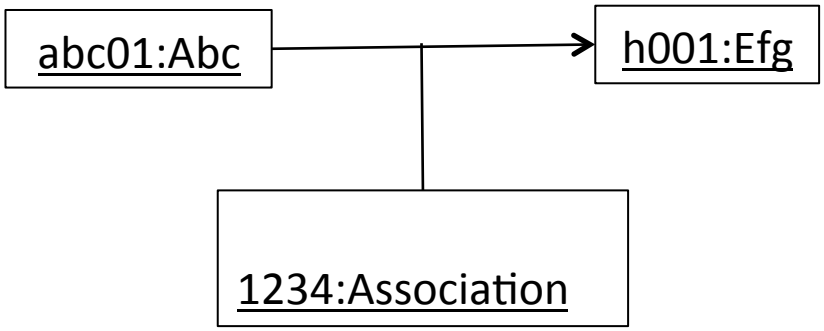
どう使い分けるの？

全体と部品の関係のような場合は左,
対等な関連の場合は右.
上の例の場合は, 右かも.

関連クラスを使った関連の表現

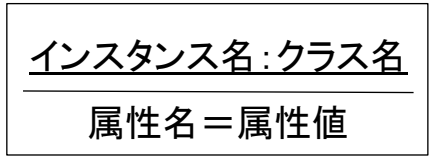
他のインスタンスへの関連は，関連型のインスタンスを使って，以下の様に記述する。

例：
```
<Abc id="abc01" ……>
……
    <connects idref="1234"/>
……
</Abc>
……
<Efg id="h001" ……>
……
    <connectedBy idref="1234"/>
……
</Efg>
……
<Association id="1234" …>
    <relateFrom idref="abc01"/>
    <relateTo idref="h001"/>
</Association>
```



関連を含むオブジェクト図の例

＊オブジェクト（インスタンス）図とは，インスタンスのある時点での状態を示す図で，以下の様に示す．



関連クラスを使った多重の関連

クラスが複数のクラスと関連することがある.

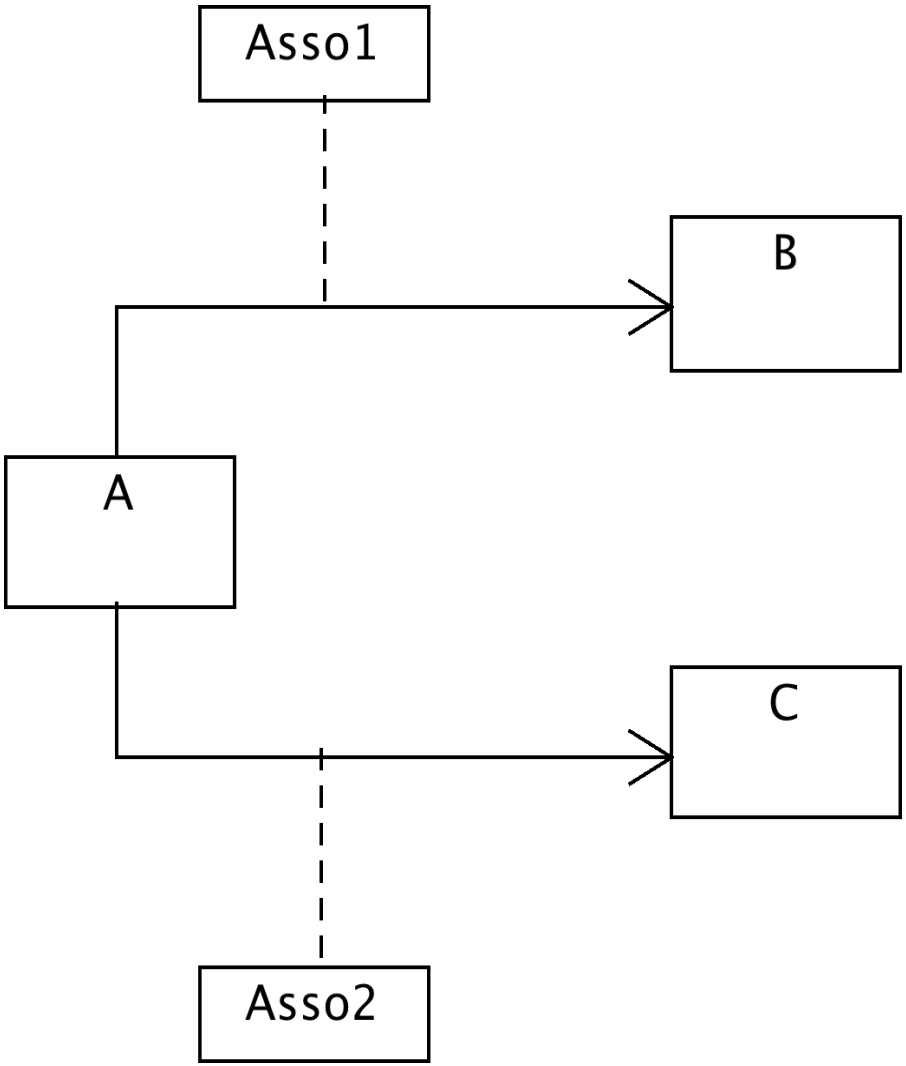
例：
```
<A id="a" ……>
……
    <connects idref="as1, as2"/>
……
</A>
……
<B id="b" ……>
……
    <connectedBy idref="as1"/>
……
</B>
……
<C id="c" ……>
……
    <connectedBy idref="as1"/>
……
</C>
```

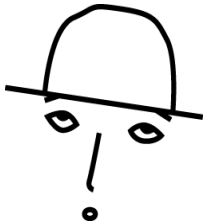
```
<Asso1 id="as1">
    <relateFrom idref="a"/>
    <relateTo idref="b"/>
</Asso1>

<Asso2 id="as2">
    <relateFrom idef="a"/>
    <relateTo idef="c"/>
</Asso2>
```

これってどんな関連？

次頁を見てみよう.


Asso1
B
A
C
Asso2



何で関連には2種類の表現法があるの？

単純な関連の場合は，役割名による関連でいい．

しかし，関連自体に属性や操作を持たせたい場合は，関連クラスを使用する．

例えば，子供の頃通った小学校があなたの家から近かったとする．

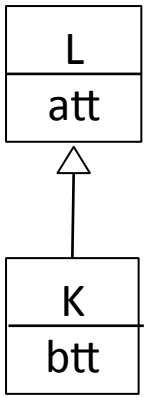
でも，家から小学校が近かった，とも言える．

何故なら，「近い」は両者の関係がもつ性質だから．

継承

上位のクラスのプロパティを継承する場合，継承関係の親は
inheritance 要素の子要素として表現する.

例：KがLの子クラスなるとき




```
<K id="k01" btt="…">
        <inheritance>
                <L id="l41" att="…"/>
        </inheritance>
</K>
```

参考文献

Erik T. Ray, 入門 XML 第2版，株式会社オライリージャパン (2004)

Erik van der Vlist，XML Schema，株式会社オライリージャパン (2003)