XML言語の基礎

奥井康弘
(株) 日本ユニテック
(Tel) 03-3595-8241
(Fax) 03-3595-8248
(E-mail) yokui@utj.co.jp

第1章 XMLとは

Extensible Markup Language
拡張可能マークアップ言語

- Extensible　タグセットの自由な設定が可能
- Markup　定義されたタグでデータを記述
- Language　タグ定義のメタ言語

タグを定義する機能を持った汎用マークアップ言語
SGML（Standardized Generalized Markup Language）
をWeb上で利用するために規格内容をスリムにしたもの

XMLの目的

互換性

- 独自形式によるデータ交換の弊害への反省
- だれにでも理解可能なテキストでの記述

資産性

- 一時点での技術に依存しないデータ表現
- 階層構造／リンクによるデータの再利用

「データはもっと構造を持ち、
かつスマートでなければならない」
— Tim Bray

「情報そのものと、情報のプレゼンテーションの分離を助け、データベース情報をブラウザに提供できるXMLは画期的だと思う」
— Bill Gates

XMLの特徴

機能

- 用途に合わせたタグ定義が可能
- 階層型の任意のデータ構造が表現可能

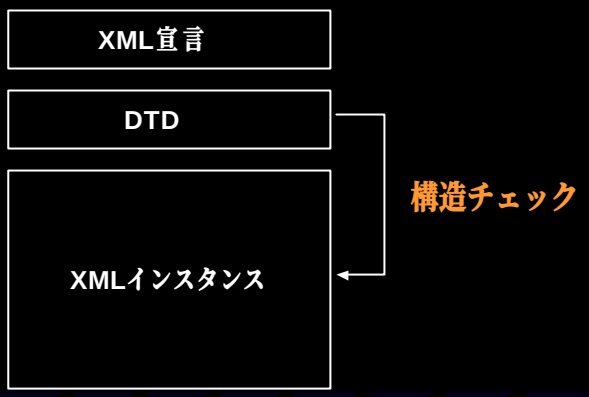
DTD（Document Type Definition）の開発が必要

データ形式

- 開始タグと終了タグが必須
- データ処理時にはDTDはオプション

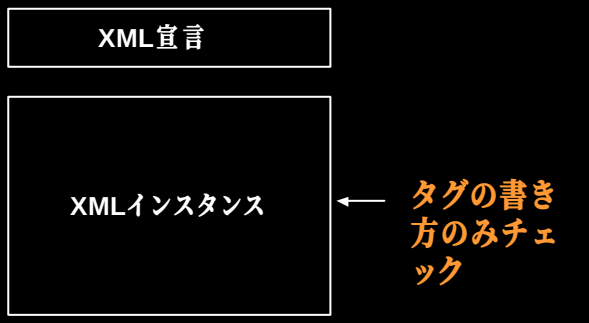
XML文書の構成

検証済みXML文書



XML文書の構成（続き）

ウェルフォームドXML文書



XMLの役割

アイデア

データ表現法

XML

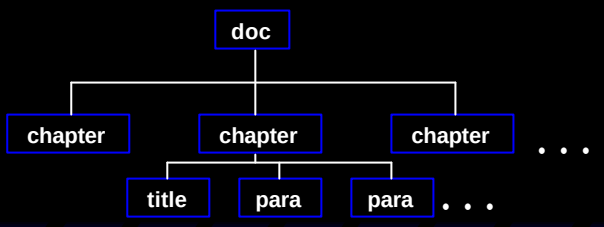
CDFの記述例

```
<?xml version="1.0" ?>
<!DOCTYPE Channel SYSTEM "http://www.w3c.org/Channel.dtd" >
<CHANNEL HREF="http://www.utj.co.jp" >
  <SELF HREF="http://www.utj.co.jp/ut.cdf"  />
  <TITLE>NIHON UNITEC Home Page</TITLE>
  <LOGO HREF="http://www.utj.co.jp/ut.ico" STYLE="ICON"/>
  <LOGO HREF="http://www.utj.co.jp/ut.gif" STYLE="IMAGE"/>
  <SCHEDULE>
    <INTERVALTIME  HOUR="2" />
  </SCHEDULE>
   <CHANNEL>
     <TITLE>SGML Salonr News</TITLE>
     <ITEM HREF="http://www.utj.co.jp/SGML/sgmlnews.htm" />
        <TITLE>Latest SGML Salon News Letter</TITLE>
     </ITEM>
     <ITEM HREF="http://www.utj.co.jp/XML/xml.htm" />
        <TITLE>Articles on XML</TITLE>
     </ITEM>
   </CHANNEL>
</CHANNEL>
```


XML DTDの記述例

簡単なマニュアル用DTD

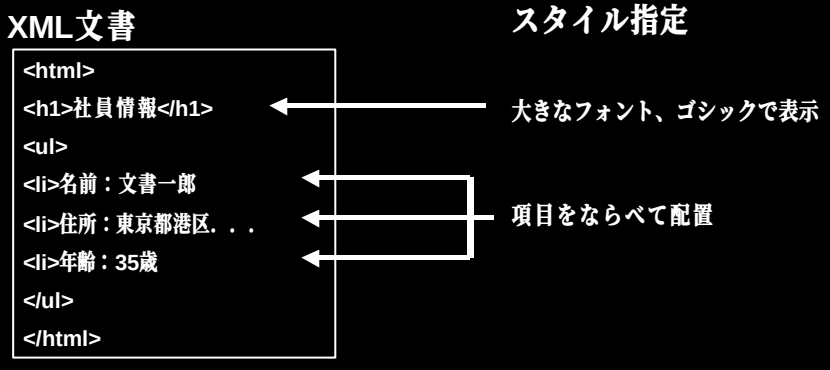
```
<!ELEMENT doc                (chapter*)      >
<!ELEMENT chapter            (title,para*)   >
<!ELEMENT title              (#PCDATA)       >
<!ELEMENT para               (#PCDATA)       >
<!ATTLIST chapter id  ID  #REQUIRED      >
```



第２章 HTML－その限界とXMLの出現

HTMLの限界

スタイル指定のためのタグ



タグに意味が付与されたXML

要素名に意味を込める

全文検索の非効率

平均気温が60度の国を探したい．．．．．

「60度」で検索 ．．．．

。。。。。。
<h2>ＸＸ国の気象</h2>
<p>平均気温：60度
<p>平均降雨量：100mm
。。。。。。

。。。。。。
<h2>ＸＸ市のプロフィール</h2>
<p>位置：東経135度、北緯60度
<p>人口：40万人
。。。。。。

。。。。。。
<h2>問題その１</h2>
<p>3つの角度が60度である多角形を何とよぶでしょう？
<p>回答：正三角形
。。。。。。

XMLを使った絞り込み検索

平均気温が60度の国を探したい．．．．．

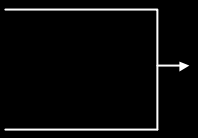
「temparature要素の内容が“60度”であるcountry要素」として検索 ．．．．

<country>
<name>ＸＸ国</name>
<temparature>60度</temparature>
<rainfall>100mm</rainfall>
。。。。。。
</country>

名前空間を利用したHTMLとXMLの共存

→ XHTML

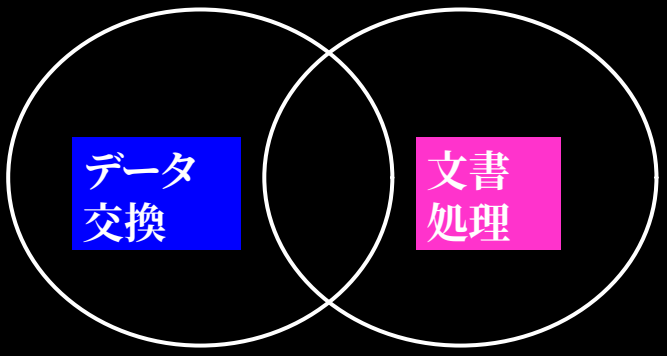
```
<HTML>
<H1>HTMLへのMathMLの埋め込み</H1>
<P>数式はMathMLを使って表現できます</P>
<M:EXPR>
  <M:MI>x</M:MI>
    <M: PLUS/>
  <M:MN>5</M:MN>
</M:EXPR>
</HTML>
```



数式を表すXMLを解釈するアプリケーションへ渡される

第3章 XMLの利用

XMLの２つの利用分野



XMLの利用（Eコマース）

RosettaNet

サプライチェーン取引の標準のための非営利組織
ＤＴＤを含むパートナー間のインターフェースプロセス
を定義

BizTalk

米マイクロソフトが提唱する電子商取引技術の枠組み

XMLの利用（Eコマース）

OTP（Internet Open Trading Protocol）

請求書・領収書発行等のインターネット取引全体のプロセスをサポートする標準取引プロトコル。記述言語としてXMLを使用。

XMLの利用（マスコミ）

NITF（News Industry Text Format）

新聞を始めとするニュース発信業でのXMLをベースとしたデータ形式

Wall Street Journal Interactive Edition

Dow Jones系の情報発信にXMLを活用

XMLの利用（マスコミ）

PC World Online

Oracleに貯えられたXMLをHTML文書に変換してデータ配信を行っている

Wall Street Journal Interactive Edition

Dow Jones系の情報発信にXMLを活用

XMLの利用（マスコミ）

ワシントンポスト

150の雇用主の25000の職種に対応するWeb求人欄をXMLを利用して運用

XMLの利用（書籍）

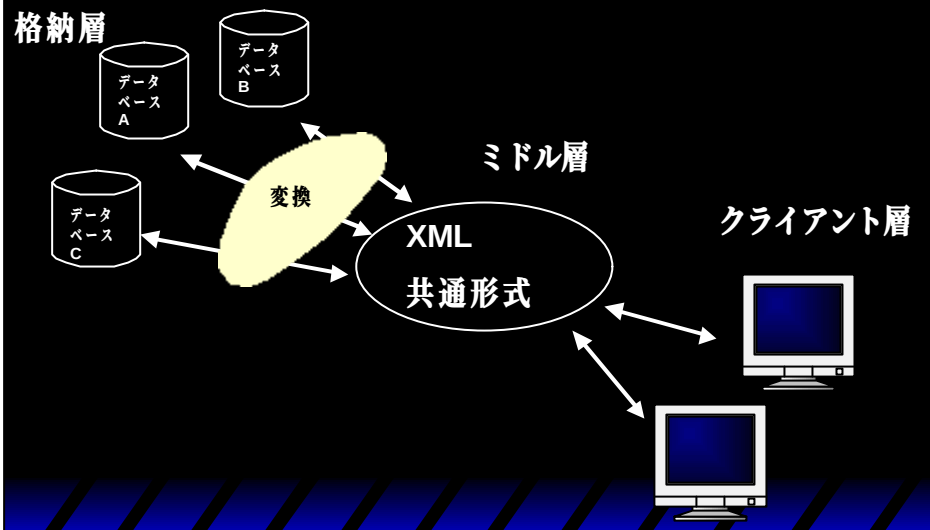
Open eBook

書籍、論文、Webページの出版社や著者など、コンテンツ作成者が単一の形式で原稿を配布するためこXMLを採用

XML利用のために必要な作業

- 表現したいデータ構造の分析
- DTDあるいはスキーマの開発
- XMLアプリケーションの開発

Microsoftの戦略： 3-Tierモデル



第4章 ウェルフォームドXML文書

ウェルフォームドXML文書とは
(Well-formed XML Document)

- XMLインスタンスにおいて、タグの書き方が正しいかどうかのチェックが行われたデータ
- DTDに表現された要素構造のチェックは行わない

→ パフォーマンスを上げるための処理モード

ウェルフォームドXML文書の条件

- 要素を１つ以上含む
- きちんとした入れ子構造でなっていなければならない
- ルート要素はただ１つでなければならない

タグ付けの基本

- 要素を開始タグと終了タグで囲む

<要素名>..要素内容..</要素名>

タグ付けの基本（続き）

- きちんと要素の階層を表すように書く

<上位要素>
<下位要素>. .要素内容.. </下位要素>
<下位要素>. .要素内容.. </下位要素>
</上位要素>

タグ付けの基本（続き）

間違ったタグ付け

トマトは夏野菜<fn>夏が旬の野菜のこと </fn>

正しいタグ付け

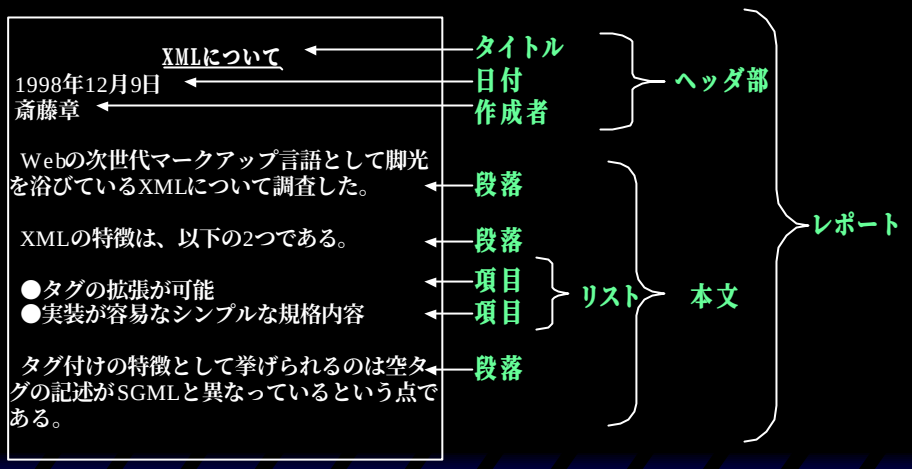
トマトは夏野菜<fn>夏が旬の野菜のこと</fn>

タグ付けの基本（続き）

間違ったタグ付け - ルート要素が１つに定まらない

```
<sec>
<title>ＳＧＭＬ</title>
<p>SGMLは1986年.....</p>
</sec>
<sec>
<title>ＸＭＬ</title>
<p>XMLは1998年.....</p>
</sec>
```


文書／データの内容の要素分け



要素に従ったタグ付け

```
<report>
<header>
<title>XMLについて</title>
<date>1998年12月9日</date>
<author>斎藤章</author>
</header>
<body>
<p>Webの次世代マークアップ言語として脚光を浴びているXMLについて調査した。</p>
<p>XMLの特徴は、以下の2つである。</p>
<list>
<item>タグの拡張が可能</item>
<item>実装が容易なシンプルな規格内容</item>
</list>
<p>タグ付けの特徴として挙げられるのは空タグの記述がSGMLと異なっているという点である。</p>
</body>
</report>
```


属性を指定する

- 属性は開始タグの中に書く

<要素名 属性名＝”属性値 ” >

属性は複数指定が可

空要素の特別な書き方

- テキストの内容や下位要素を持たない空要素は特別な書き方をする

<要素名/>

SGMLにはない新たな書き方

コメント文の書き方

<!-- コメント文 -->

XML宣言の書き方

<?xml version="1.0" encoding="文字コード指定" ?>

encoding指定はオプション

第５章 検証済みXML文書のタグ付け

検証済みXML文書とは
(Valid XML Document)

- XMLインスタンスにおいて、タグの書き方が正しいかどうかのチェックが行われたデータ（ウェルフォームドXML文書）
- DTDに表現された要素構造／属性型のチェックを行う

→ SGMLと同じ

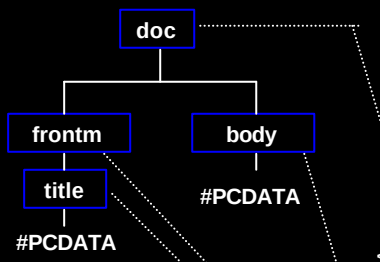
検証済みXML文書の例
(Valid XML Document)

```
<?xml version="1.0"?>
<!DOCTYPE document [
<!ELEMENTdocument (frontm,body)>
<!ELEMENTfrontm   (title)>
<!ELEMENTtitle    (#PCDATA)>
<!ELEMENTbody     (#PCDATA)>
]>
<document>
<frontm>
<title> Valid XML 文書の実際</title>
</frontm>
<body>
これはValid XML文書です。
</body>
```

DTD

XMLインスタンス

検証とは？



DTDによって表現された要素出現の順番にしたがってXMLインスタンスをチェックすること。

属性の書き方などもDTDに照らしてチェックする。

```
<doc>
<frontm>
<title> Valid XML 文書の実際</title>
</frontm>
<body>
これはValid XML文書です。
</body>
</doc>
```


DTDで設定できること

- 要素の定義と出現順序／回数
- 属性の定義
- エンティティの定義
- 記法の定義

要素型宣言

<!ELEMENT 要素名 要素内容>

要素内容で、下位に来る要素の種類、
出現順序、出現回数を表現

出現回数記号

記号なし	：	要素は必ず１回出現する。
＋	：	要素は１回以上出現する。
＊	：	要素は０回以上出現する。
？	：	要素は０回あるいは1回出現する。

出現順序記号

,	：	要素は並べられた順に出現する。
|	：	並べられた要素のうちどれか１つが出現する。

属性リスト宣言

```
<!ATTLIST 要素名 属性名 属性値の候補 ”デフォルト値”>
```


DTDの指定法ーDOCTYPE宣言

DOCTYPE宣言内に直接書く

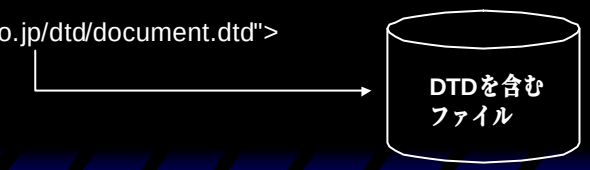
ー内部サブセット

```
<!DOCTYPE ルート要素名 [
  要素宣言
  属性宣言
  ・・
]>
```

外部のファイルを参照する
―外部サブセット

<!DOCTYPE ルート要素名 SYSTEM “ファイルのURL">

<!DOCTYPE doc SYSTEM
“http://www.utj.co.jp/dtd/document.dtd">



DTDとXMLプロセッサ

非検証XMLプロセッサ

- タグの書き方のチェックを行う
- 属性値のデフォルトについてはDTDを使用
  （その他、置換文字列、外部ファイル指定も）
- DTDの外部サブセットは読んでも読まなくてもよい（どちらがサポートされているか注意！）

DTDとXMLプロセッサ（その２）

検証XMLプロセッサ

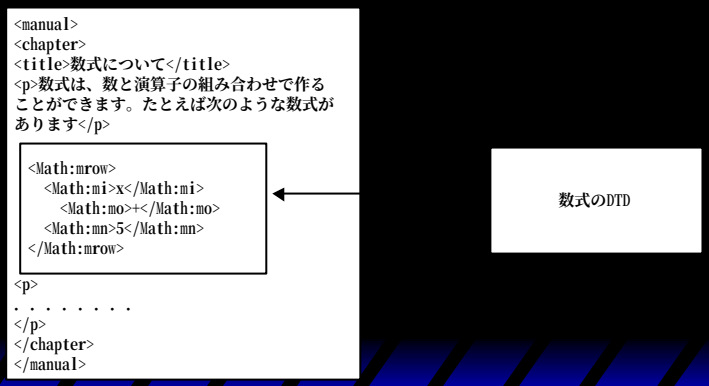
- タグの書き方のチェックを行う
- DTDに照らしながらXMLインスタンスの構造をチェックする
- DTDの外部サブセットを含め、すべての外部ファイルを読む

第6章 名前空間

名前空間（namespace）とは

制定の動機：

既存のDTDをそのまま自分の文書で使用したい。



名前空間の宣言

<要素名 xmlns:名前空間接頭辞 = “名前空間を表すURI”>

- 名前空間を使う最も外側の要素（あるいはさらに外側でも可）で定義
- その要素自身から使用可

複数のDTD（スキーマ）に従う要素を１つのXML文書に混在させる

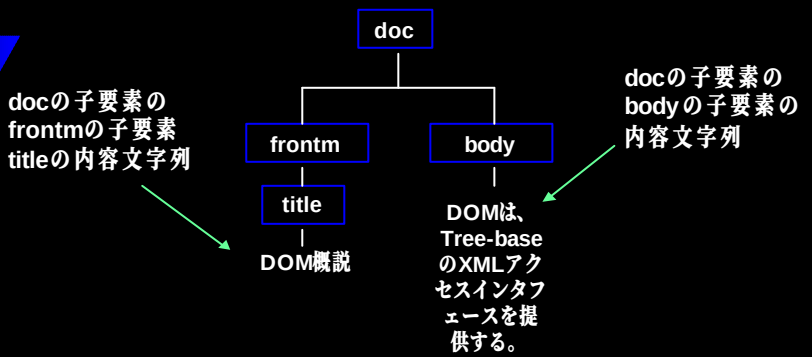
```
<?xml version=”1.0”  >
<m:memo xmlns:m = ”http://www.utj.co.jp/dtd/memo” >
<m:title>書籍受注 </m:title>
<m:p>今日、以下の書籍の注文がありました。 </m:p>
<od:order xmlns:od=”http://www.utj.co.jp/dtd/order” >
<od:soldto>奥井康弘</od:soldto>
<od:otem>標準XML完全解説</od:item>
<od:price>2280円</od:price>
</od:order>
</m:memo>
```


第７章 XMLハンドリング

DOM
（Document Object Model）

- XMLのオブジェクトモデル
- XMLの要素間の関係を定義するインタフェースセット

DOMによるデータへのアクセス



IE5.0でのDOMコーディング例

```
var xmldoc = new  ActiveXObject ("Microsoft.XMLDOM");
var xmlurl = xmlsource.value;
xmldoc.load(xmlurl);

var xsldoc = new ActiveXObject  ("Microsoft.XMLDOM");
xsldoc.load("genreselect.xsl");

var templatenode = xsldoc.selectSingleNode("//xsl:template[2]");
var genreField = templatenode.selectSingleNode("@match");
genreField.nodeValue = "book[@genre='" + genre.value + "']";

DISP.innerHTML = xmldoc.transformNode(xsldoc);
```


DOMの利点／欠点

利点：

- 木構造がメモリに作成される
- XML文書の作成・変更ができる

欠点：

- メモリを消費する

DOM木は汎用の木構造であり、オブジェクト指向言語のプログラマにとって自然な木構造ではない。

（要素は常にクラスElementのインスタンスになり、アプリケーションプログラムが使いたいクラスであるYear, Personなどとは異なる）

注意すべき点:
要素内容における空白

```
<ol>
 <li>test</li>
</ol>
```

この要素(<ol>...)に関数firstChildを適用して先頭要素を求める。

答えは、
test
ではない！

答えは、<ol>と<li>の間の空白（改行とスペース）

注意すべき点:
要素内容における空白 (2)

要素olに要素
test
を関数appendChildで追加した結果のファイル出力

```
<ol><li>test</li></ol>
```

関数createTextNodeを使って空白テキストノードを生成し、関数appendChild で繋ぎ込む必要あり

DOM実装リスト

Javaベース:

- IBM XML Parser for Java 付属のDOM
- Sun Java Project X 付属のDOM

C++ベース:

- MS IE 5.0付属のDOM
- IBM XML4C付属のDOM

DOM実装リスト（続き）

Perlベース：

- XML::DOM

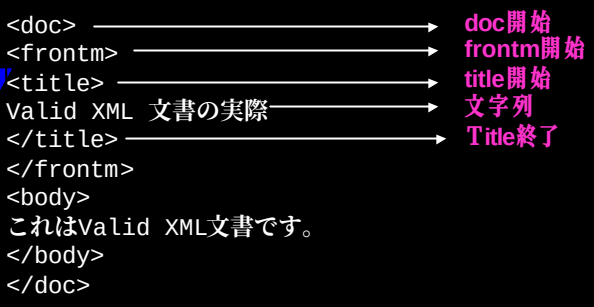
JavaScript、VBScriptベース：

- MS IE 5.0付属のDOM

SAX
（Simple API for XML）

- イベントベースのXMLパージング・インタフェース
- XML-DEV（XML開発者のメーリングリスト）のメンバーの議論の成果

SAXによるイベント通知



SAXの利点／欠点

利点：

- メモリ効率が良い。

欠点：

- 木構造が必要な場合は、プログラマが自分で作成しなければならない

プログラマが作成する イベントハンドラ

開始タグ（および空要素タグ）：

```
public abstract void startElement (String name,
                                   AttributeList atts)
    throws SAXException;
```


終了タグ（および空要素タグ）：

```
public abstract void endElement (String name)
    throws SAXException;
```

文字データ：

```
public abstract void characters (char ch[], int start,
                                 int length)
    throws SAXException;
```


無視できる文字データ：

```
public abstract void ignorableWhitespace (char ch[],
                                          int start, int length)
    throws SAXException;
```


注意すべき点: 文字内容

```
<para>
開始タグは&lt;と&gt;で囲まれ
ます。
</para>
```

関数startElementが<para>に対して呼ばれたあと、関数endElementが<li>に対して呼ばれるまで、関数characterは何回呼ばれるか？

1回？（まとめて）
3回？（各行ごと）
5回？（<以前、<、"と"、>、>以降）

注意すべき点:
要素内容における空白

```
<ol>
  <li>test</li>
</ol>
```

関数startElementが<ol>に対して呼ばれたあと、ただちに
関数startElementが<li>に対して呼ばれるのではない!

DTDが存在し、検証を行うパーサを用いた場合

```
public abstract void ignorableWhitespace (char ch[],
                                          int start, int length)
    throws SAXException;
```

が何回か呼び出される。

改行とスペースに対してまとめて一回？
改行で一回、スペースで一回？

DTDが存在しないか、検証を行わないパーサを用いた場合

```
public abstract void characters (char ch[], int start, int
                                                    length)
    throws SAXException;
```

が何回か呼び出される。

改行とスペースに対してまとめて一回？
改行で一回、スペースで一回？

SAX実装リスト

Javaベース：

- IBM XML Parser for Java 付属のSAX
- Sun Java Project X 付属のSAX

C++ベース：

- IBM XML4C付属のDOM
- expat

SAX実装リスト（続き）

Perlベース：

- XML::Parser

標準APIによる開発の効率アップ