ーXML入門とXSLTー

岐阜経済大学　経営学部 経営情報学科　井戸　伸彦

来歴：

0．0版　2004年8月8日

スライドの構成

はじめに

(1) XMLの概要

(2) 最初のXML文書

(3) DTDによるXML文書の構文チェック

(4) 一覧表を表示する

(5) リンク・分岐条件により表示を充実させる

(6) 新しいXSLスタイルシート

(7) 表示を充実させる

はじめに

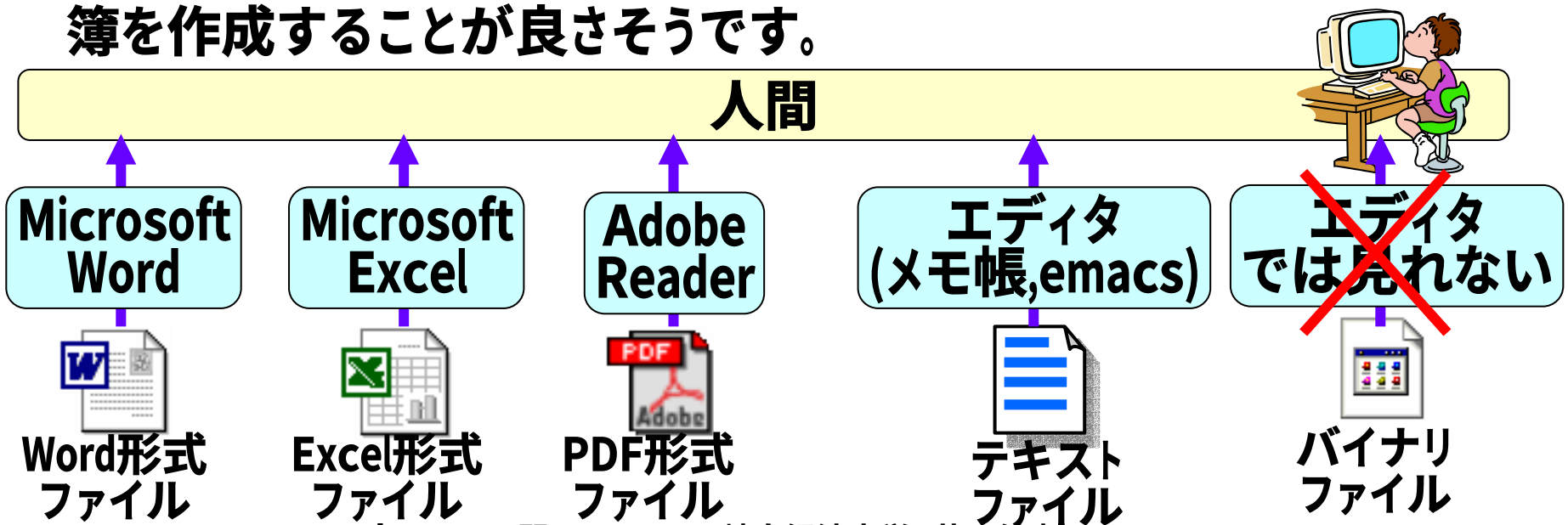
- 本スライドでは、ＸＭＬ、ＸＳＬＴスタイルシート、ＤＴＤの初歩的な説明を行います。
- 直感的な説明を行い、若干不正確な言い回しを含んでいます。
- 実習は次の環境で行うことを想定しています。
  - Linux PC(実際に授業で使ったのは、Fedora Core2)

(1．1) 名簿のファイルを作成する

■名簿を作成して電子的に保存する(ファイルを作成する)ことを考えます。どのように作っておくと便利なのでしょうか？

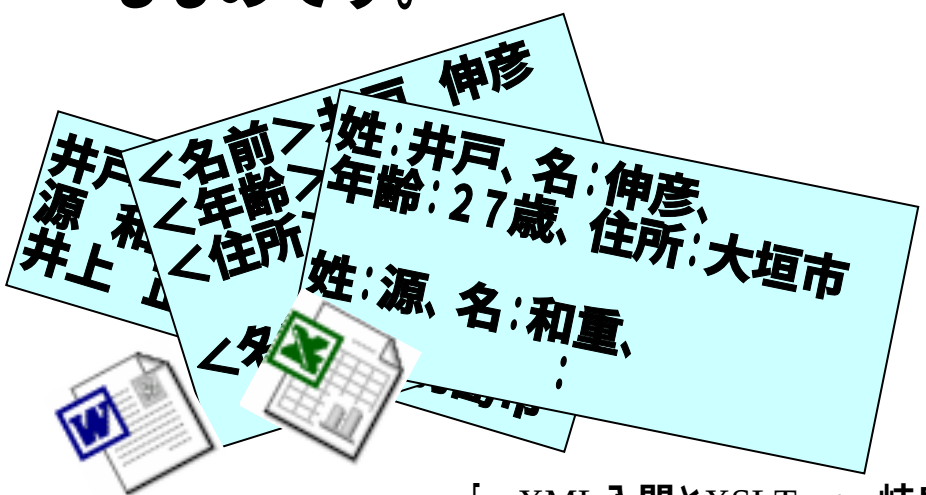
■WordやExcelのファイルで名簿を作成するとすれば、それは特定のアプリケーションソフトウェアで扱うことを前提とすることになります。

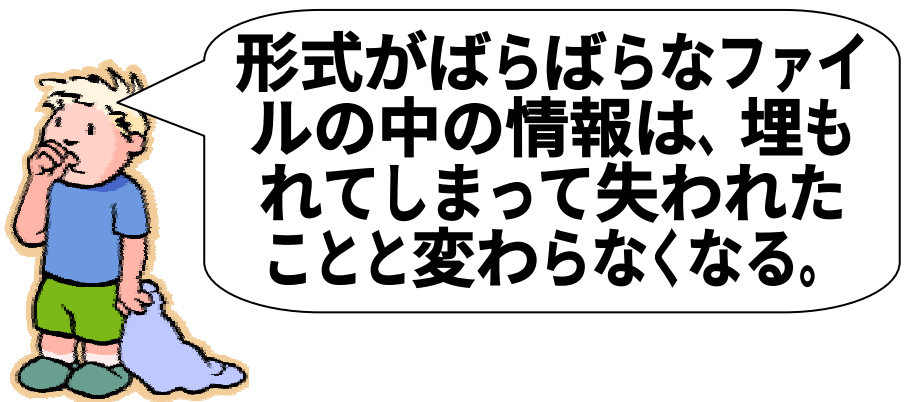
■特定のアプリケーションに依存せず、人間が扱いようにするには、テキストファイル(emacsやメモ帳で編集するファイル)で名簿を作成することが良さそうです。



(1．2)形式を統一すること

- コンピュータが一般化してから何十年かが過ぎ、膨大なファイルが世の中には存在しています。
- これらのファイルのかなりのものは、「形式がばらばらなので読み出すことに手間が掛かる」という理由から、永遠に埋もれてしまう恐れが大です。それはWordやExcelで作成されたファイルについても同じことです。
- 将来的にも利用な形でファイルに情報を残すには、統一された形式であることが必須です。XMLは、統一された形式を提供するものです。





(1．3) どのようなテキストファイルが良いのか？

■テキストファイルで名簿を作成するとき、その方法（すなわち形式）は千差万別です。どのような形式で統一すれば良いのでしょうか？

```
井戸　伸彦　27歳　大垣市
源　和重　37歳　岐阜市
井上　正美　33歳　羽島市
```

```
<名前>井戸　伸彦
<年齢>27歳
<住所>大垣市
<名前>源　和重
<年齢>37歳
<住所>岐阜市
<名前>井上　正美
<年齢>33歳
<住所>羽島市
```

```
姓:井戸、名:伸彦、
年齢:27歳、住所:大垣市

姓:源、名:和重、
年齢:37歳、住所:岐阜市

姓:井上、名:正美、
年齢:33歳、住所:羽島市
```

■そもそも、形式の良い･悪いは、どのような観点から決めれば良いのでしょうか？

■名簿を作成していく中で、どうしてＸＭＬを用いることが良いのかをみていきます。

(1．4) XMLで作成した名簿

■XML文書として作成した名簿の例を示します。

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<listOfNames category="卓球同好会">
  <lastupdate>2004.7.7</lastupdate>
  <member id="01">
    <name>井戸 伸彦</name>
    <age>27</age>
    <address>大垣市</address>
    <misc>がんばります。</misc>
  </member>
  <member id="02">
    <name>源 和重</name>
    <age>37</age>
    <address>岐阜市</address>
    <misc>オリンピックめざします。</misc>
  </member>
</listofnames>
```

(1．4．1)XML文書であることの宣言

■すべてのXML文書は、XML文書であることを文頭で宣言します。

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
```

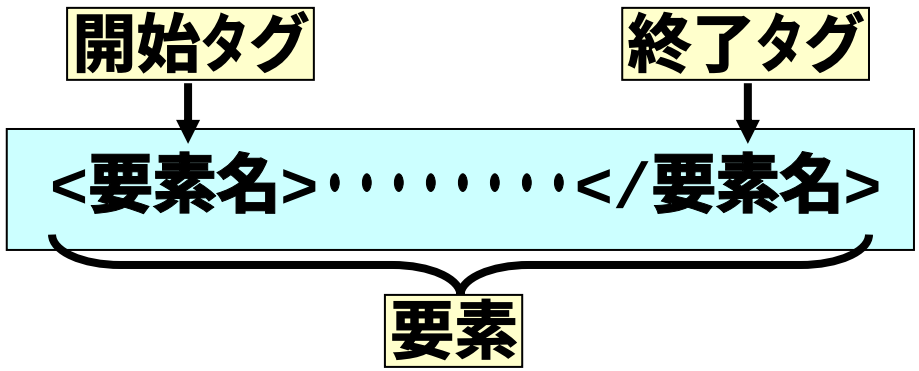
- version：XMLのバージョンを示します。現在は、1．0です。
- encoding：
  - ◆使用している文字コードを指定します。
  - ◆デフォルトは、Unicode(UTF-8,UTF-16)となっており、これを用いる時は省略出来ます。
  - ◆実際に使っている文字コードと、encodingで指定する文字コードとが違っていると、文字化けの原因となります。

（1．4．2）要素(Element)と属性(Attribute)

■XML文書は、要素と属性とによって構成されます。

■要素

- 開始タグ(<xxx>)と終了タグ(</xxx>)とに囲まれています。



■属性

- タグの中に、「属性名＝“属性値”」という形式で記します。

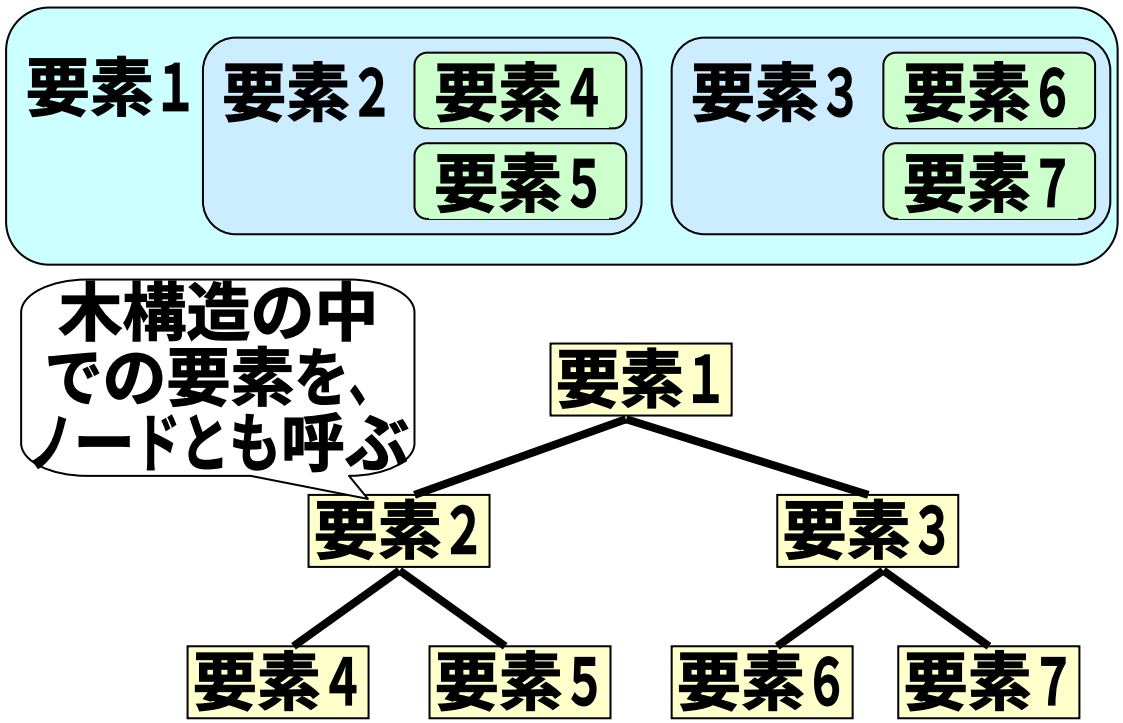
```
<要素名　属性名=“属性値”>・・・・・・・・</要素名>
```

■ＨＴＭＬでは決められたタグ･属性名を用いるのに対して、ＸＭＬでは自由にタグ･属性名を決めることが出来ます。

(1．4．3) 要素の木構造

■ＸＭＬ中の要素は、木構造をなしています。すなわち、要素1の中に含まれる要素2を要素1の子供と考え、血統図（樹形図）を作っていくと、木の形になります。

```
<要素名1>
    <要素2>
        <要素4>･･･</要素4>
        <要素5>･･･</要素5>
    </要素2>
    <要素3>
        <要素6>･･･</要素6>
        <要素7>･･･</要素7>
    </要素3>
</要素名1>
```



■ＸＭＬでは、必ず1つだけの要素（ルート要素）が、他のすべての要素を含む形をしています。

■問い：スライド(1.4)のＸＭＬ文書の木構造を図示せよ。

(1．4．4) 正しいネスト

■要素が木構造になっているということは、要素が正しくネスト(入れ子)されていることを意味します。

■HTMLではブラウザが適当に解釈してくれた次のような記述は、XMLでは受け入れられません。

✗
```
<h1><font color="red">赤い字</h1></font>
```

■どちらかが他方を含む次のような形にする必要があります。

○
```
<h1><font color="red">赤い字</font></h1>
```

要素<h1></h1>　要素

(1．4．5) 必ず終了タグ

■要素が正しくネストされているために、XMLでは終了タグの省略は許されません。

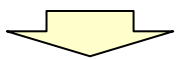
■HTMLではブラウザが適当に解釈してくれた左のような記述は、XMLでは受け入れられません。右ように必ず終了タグを記さなければなりません。

×
```
<dt>名前一覧
<dd>井戸　伸彦
```

○
```
<dt>名前一覧</dt>
<dd>井戸　伸彦</dd>
```

■開始タグと終了タグで囲まれる中身のデータが無い場合は、次のような省略形が許されます。

```
<img src="picIdo.jpg"></img>
```



省略形：
```
<img src="picIdo.jpg" />
```

(1．4．6) 属性値、大文字・小文字

■XMLでは属性値を必ずダブルクォーテーション(“”)で囲みます(HTMLでは省略も許されていました)。

✗ `<img src=picIdo.jpg />`

○ `<img src=“picIdo.jpg” />`

■XMLでは、大文字と小文字とを区別します(HTMLでは区別がありませんでした)。

✗ `<h1>私の主張</H1>`

○ `<h1>私の主張</h1>`

(1．4．7) コメント

■次のように“<!--”と“-->”とで囲うと、XML文書内ではコメントとして扱われます。

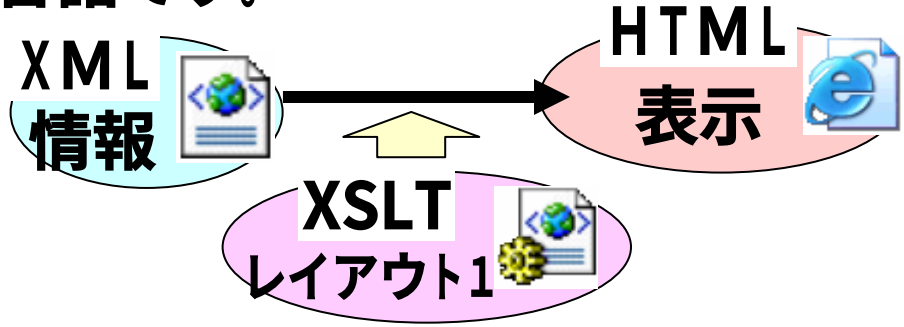
```
<!--  この部分はコメント  -->
```

(1．5) 様々な関連技術

■ＸＭＬは(1．4)に記したような文書である訳ですが、そのシンプルで拡張性のある性質により、様々な関連技術と結びついて、有用な技術体系を形成しています。

■ＸＳＬＴスタイル･シート((1.6)で説明,(2),(4)～(6)で実習)

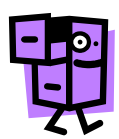
- ＸＳＬＴは、ＸＭＬ文書変換用言語です。
- ブラウザでＸＭＬ文書を表示する際に、ＨＴＭＬ文書に変換するような使い方があります。



■ＤＴＤ((1.7)で説明、(3)で実習)

- あるＸＭＬ文書に記述することが出来る要素、属性の種類とそれらの関係を定義するためのドキュメントです。

：整形式で、妥当なＸＭＬ文書



：ＤＴＤ

(1．6．1) HTMLに比べて何が良い？

■たとえば、スライド(1.4)に示したＸＭＬ文書と似たような右のHTML文書を見てみます。

■“２７”が何を意味しているのか、右のＨＴＭＬ文書では解りません。大垣市は住所(address)なのか、出身地(home town)なのかも解りません。

```
<html>
<head><title>卓球同好会
</title></head>
<body>
   <p>2004.7.7</p>
   <dt>
      <dd>井戸　伸彦<dd>
      <dd>27</dd>
      <dd>大垣市</dd>
      <dd>がんばります。</dd>
   </dt>
            :
</body>
</html>
```

■ 例えば、“<h2>宮崎</h2>”とあった場合、宮崎県のことなのか、宮崎さんのことなのかも解りません。

■ 何を書いているのかを、要素と属性で明示するやりかたが、ＸＭＬ流なのです。

(1．6．2) 守備範囲の違い

■前スライドから、ＸＭＬはHTMLに比べて、「情報を記す」という観点から優れていることは、解ると思います。

■しかしながら、ＨＴＭＬには「レイアウトを記す」という機能もあります (こちらが主機能でしょう)。

■ＸＭＬには、「レイアウトを記す」機能はありません。ＸＭＬで記された情報をブラウザ (Webサイト) で表示するときは、レイアウトを指定する文書を別途用意し、これによりブラウザでの表示を行います。

■レイアウトを指定する文書が、ＸＳＬＴスタイルシートです。

HTML
情報＋レイアウト

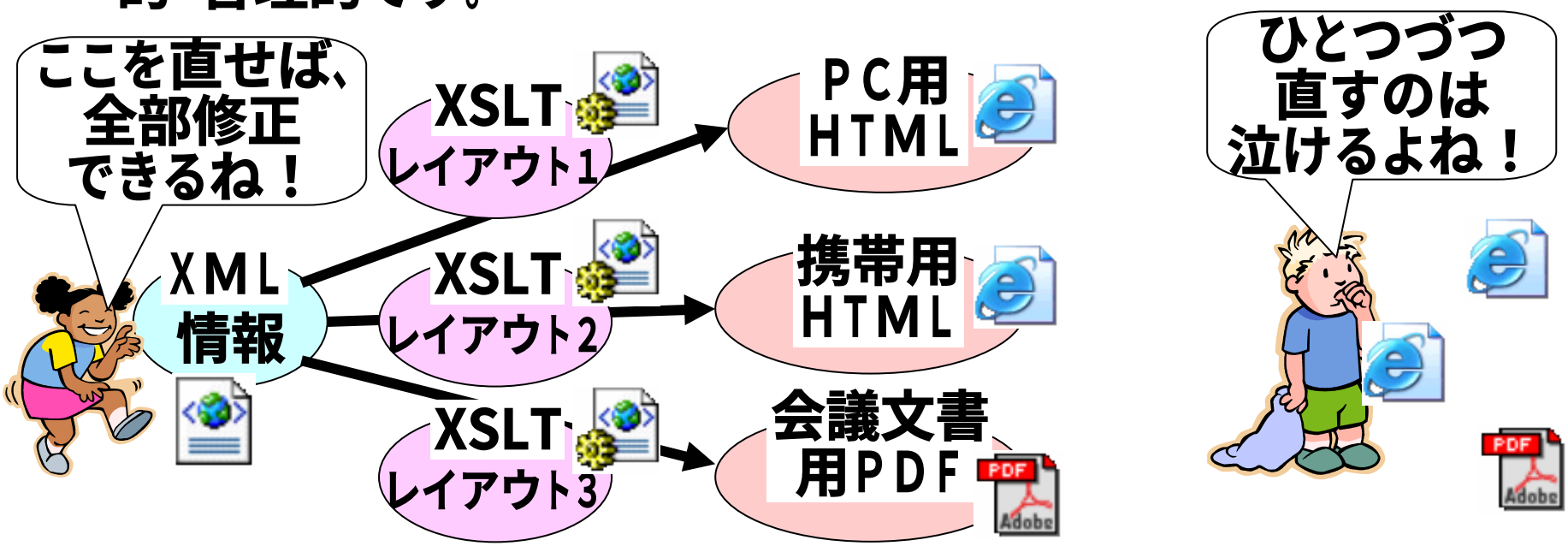
XML
情報

＋

XSLT
レイアウト

(1．6．3) XSLT

- XSLTは、eXtensible Stylesheet Language Transformationの略です。名前のとおり、ＸＳＬＴは変換を行うものであり、実際には、ＸＳＬＴによりＸＭＬ文書はＨＴＭＬに変換されます。HTMLだけでなく、他の文書形式（XML自身やＰＤＦ）にも変換できます。
- このように情報はＸＭＬ一本にして、ＸＳＬＴで様々な形式に変換することは、それぞれの形式ごとに文書を作成するよりも効率的・合理的です。



(1．7．1) さまざまな書き方、整形式

■あるXML文書が、スライド(1．4)に示した要件を満たしているとき、「文書は整形式(Well-Formed)である」と言います。

■XMLによる次の2つの同じ内容を持つ名簿の一部はいずれも整形式です。

＜文書A＞

```
          :
<member id="01">
  <name>井戸 伸彦</name>
  <age>27</age>
  <address>大垣市</address>
  <misc>がんばります。</misc>
</member>
          :
```

＜文書B＞

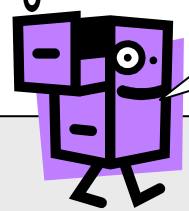
```
          :
<member id="01"
    firstname="伸彦"
    lastname="井戸" >
  <age>27</age>
  <address>大垣市</address>
  <misc>がんばります。</misc>
</member>
          :
```

■どちらも認められるのであれば、「形式を統一する」というXMLの目的は果たせなくなります。

(1．7．2) DTD (Document Type Definition)

■整形式なXML文書のなかでも、どのような形式とするかを規定しておくのが、DTDです。

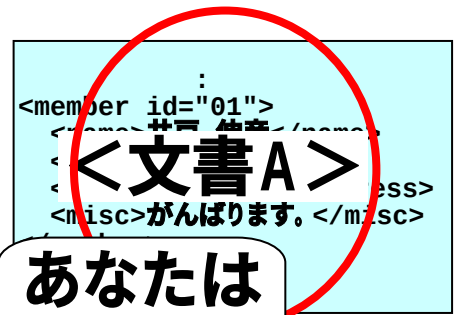
■次のリストは、前スライドの＜文書A＞のDTDを示します。＜文書B＞はこのDTDで規定された形式には合っていないわけです。



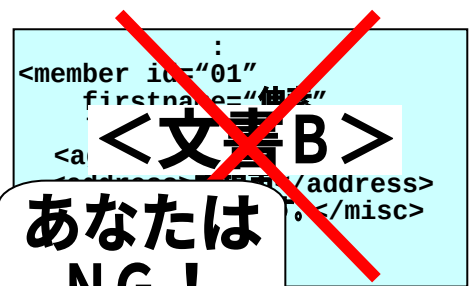
私の規定から言えば、

＜DTD＞

```
<!ELEMENT listOfNames (lastUpdate,member*)>
<!ELEMENT lastUpdate (#PCDATA)>
<!ELEMENT member (name,age,address,misc)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT misc (#PCDATA)>
<!ATTLIST listOfNames category CDATA #REQUIRED>
<!ATTLIST member id CDATA #REQUIRED>
```



あなたはOK！

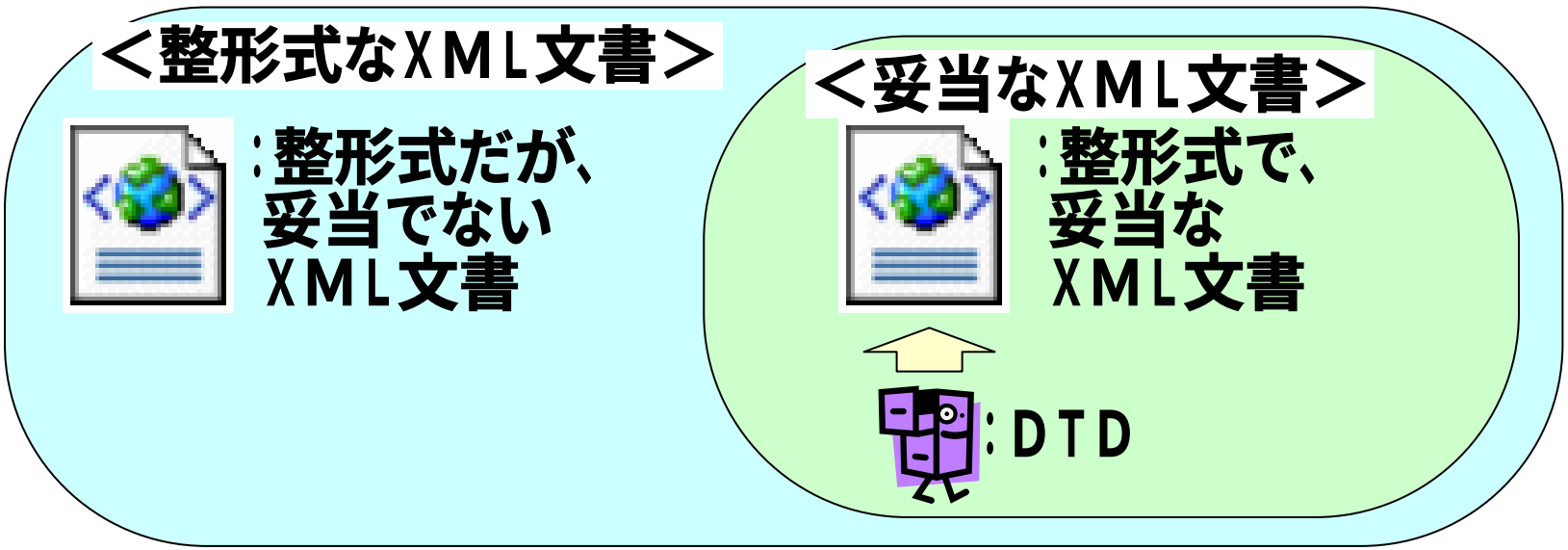


あなたはNG！

(1．7．3) 妥当 (Valid) なＸＭＬ

■文書Ａのようにｄｔｄに従っているＸＭＬ文書を、妥当なＸＭＬ文書と言います。

■妥当なＸＭＬ文書は必ず整形式ですが、整形式なＸＭＬ文書であっても妥当であるとは限りません。



■ＤＴＤにより形式をきちんと規定することが望ましいのですが、“整形式だが妥当でないＸＭＬ文書”が使われる場合もあります。

(1．7．4) スキーマ言語

- ■ＤＴＤのように、“どのようなＸＭＬ文書か”を規定する言語を、スキーマ言語と言います。
- ■ＤＴＤはスキーマ言語として、次のような欠点があると言われています (内容について説明は略します)。
  - ＸＭＬとは別の形式で記述されている。
  - データ型を持たない。 ・名前空間をサポートしない。
- ■上記のような欠点を改善したスキーマ言語が“ＸＭＬ Schema”です。
- ■しかしながら、ＸＭＬ Schemaの評判も全面的に良いという訳ではありません (複雑すぎて覚えにくいなどの批判があります)。対抗馬としては、Relax NG”という日本発のスキーマ言語がありますが、この言語の先行きは不透明とも思われます。
- ■本スライドでは、ＤＴＤのみを扱うことにします。

(1．7．5) どのように利用されているか？

■ＣＭＬ(Chemical Markup Langugae)
- 化学領域での文書向けタグセット（＝ＤＴＤで規定された一連のタグと要素の定義）。
- 化学領域では、スペクトラムデータ、結晶構造、化学結合式など多様な形式の情報が扱われる。これらの情報の記述に、ＸＭＬが適用されている。

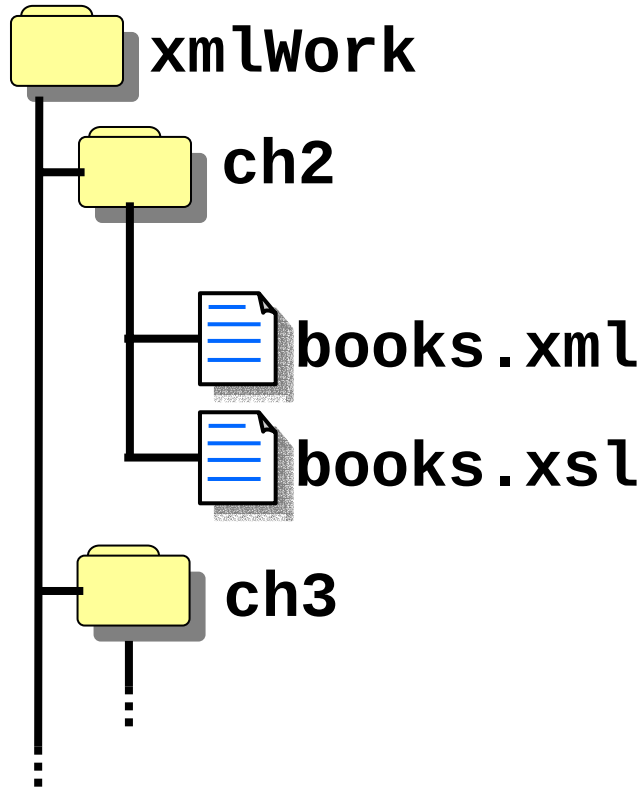
■MathＭＬ(Mathematics Markup Language)
- 数学領域（数式表現）向けタグセット

■ＰＧＭＬ(Precision Graphics Markup Language)
- グラフィックデータを構造的に表 現するタグセット

■ＯＦＸ(Open Financial Exchange)
- 財務情報の記述用タグセット

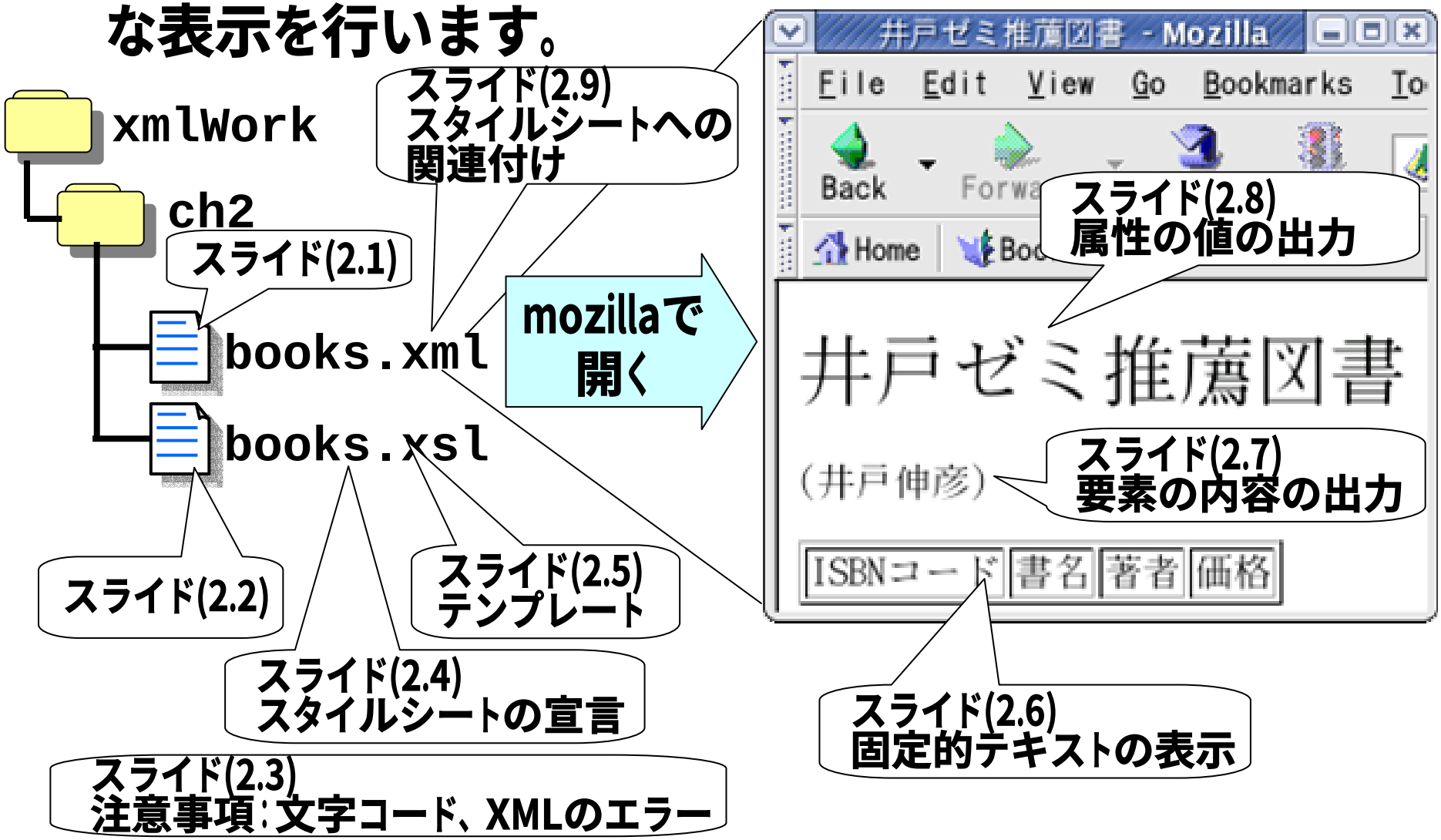
(1．8) 実習の進め方

- 各章で説明に用いるファイルは、右図のようなディレクトリに収めて、実習時に供給します。
- スライドではファイルの中身が見づらい点があると思いますので、供給されたファイルで補ってください。
- 供給されたファイルをコピー＆ペーストすることで実習自体が済んでしまうことになりますが、それでは何も頭に残らない恐れがあります。なるべく自分で打ち込むようにしてください。



(2) 最初のXML文書作成

■XML文書とXSLTスタイルシートとを作成し、次のような表示を行います。



(2．1．1) 最初のXMLファイル

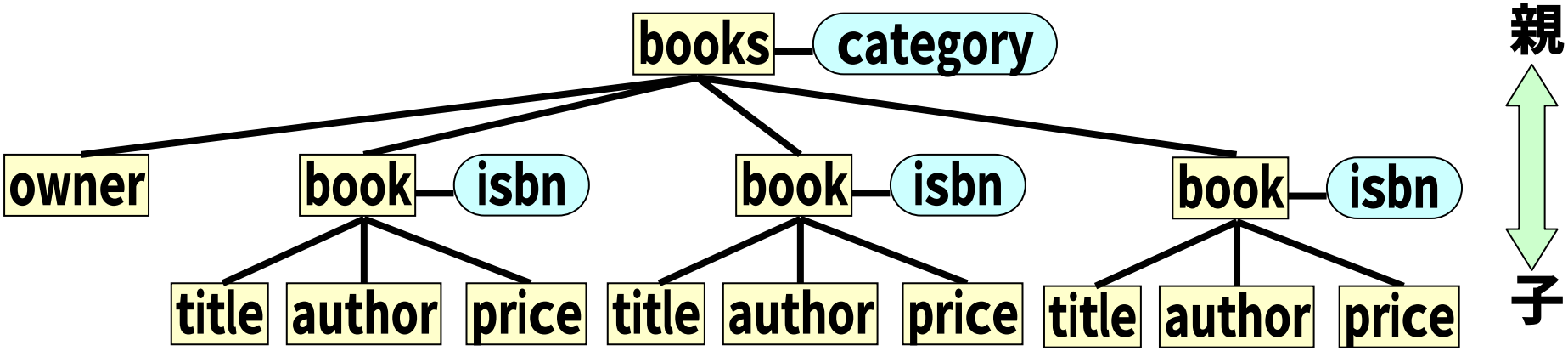
■emacsを使って、次のファイルを編集します。

＜books.xml＞

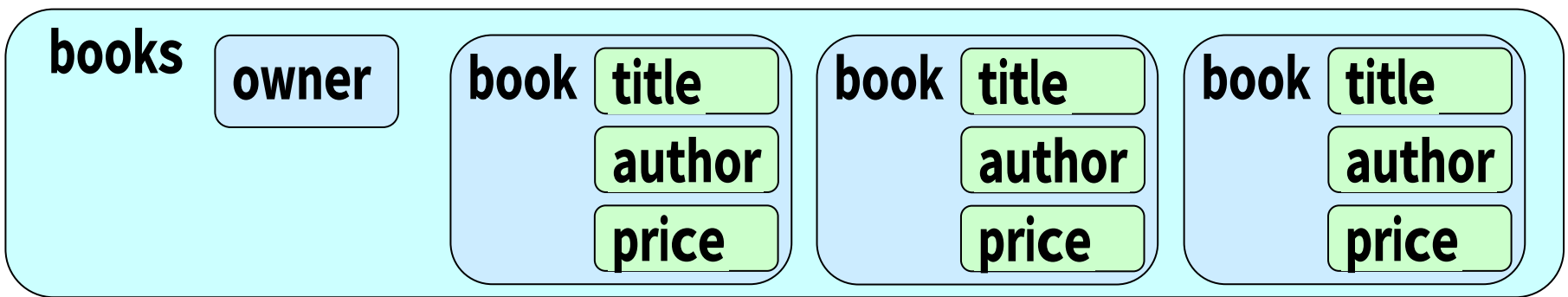
```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<?xml-stylesheet type="text/xsl" href="books.xsl" ?>
<books category="井戸ゼミ推薦図書">
  <owner>井戸伸彦</owner>
  <book isbn="ISBN4-7981-0439-6">
    <title>10日でおぼえるJakarta入門教室</title>
    <author>山田祥寛</author>
    <price>2800</price>
  </book>
  <book isbn="ISBN4-7973-1318-8">
  <title>新Linux/UNIX入門</title>
  <author>林 晴比古</author>
  <price>4100</price>
  </book>
  <book isbn="ISBN4-7973-1857-0">
    <title>新Java言語入門</title>
    <author>林 晴比古</author>
    <price>2400</price>
  </book>
</books>
```

(2．1．2) XML文書の木構造

■前スライドに記したXML文書は、次のような木構造を持っていることが解ると思います。



■<xxx>(開始タグ)～</xxx>(終了タグ)で囲んでいるほうを親要素、囲まれているほうを子要素とする木ですね。



(2．2) XSL

同様に、XSLTスタイルシートを編集します。

(行番号は入力しないでください。)

改行なし

＜books.xsl＞

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL
/Transform“ version="1.0">
  <xsl:output method="html" encoding="iso-2022-jp" />
  <xsl:template match="/">
    <html>
    <head>
    <title><xsl:value-of select="books/@category" /></title>
    </head>
    <body>
    <h1><xsl:value-of select="books/@category" /></h1>
    <p>(<xsl:value-of select="books/owner" />)</p>
    <table border="1">
    <tr><th><xsl:text>ISBNコード</xsl:text></th>
        <th><xsl:text>書名</xsl:text></th>
        <th><xsl:text>著者</xsl:text></th>
        <th><xsl:text>価格</xsl:text></th></tr>
    </table>
    </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

(2．3．1) 注意1：文字コード

■本スライドで扱うファイルの文字コードは、すべて“iso-2022-jp”(JISコード)とします。

■emacsでの文字コード変換

- 文書を開いた状態で、次の操作を行う。
  - “M-x”([Esc]キーを押下、[x]キーを押下)
  - 次のように入力して[Enter]キー押下。

    set-buffer-file-coding-system
  - “Coding system for visited file(default, nill):”のプロンプトに、“iso-2022-jp”と入力し、[Enter]キー押下。

■ターミナル上での文字コード変換 (books.xmlを変換する)

```
% nkf -j books.xml>temp
% mv temp books.xml
```

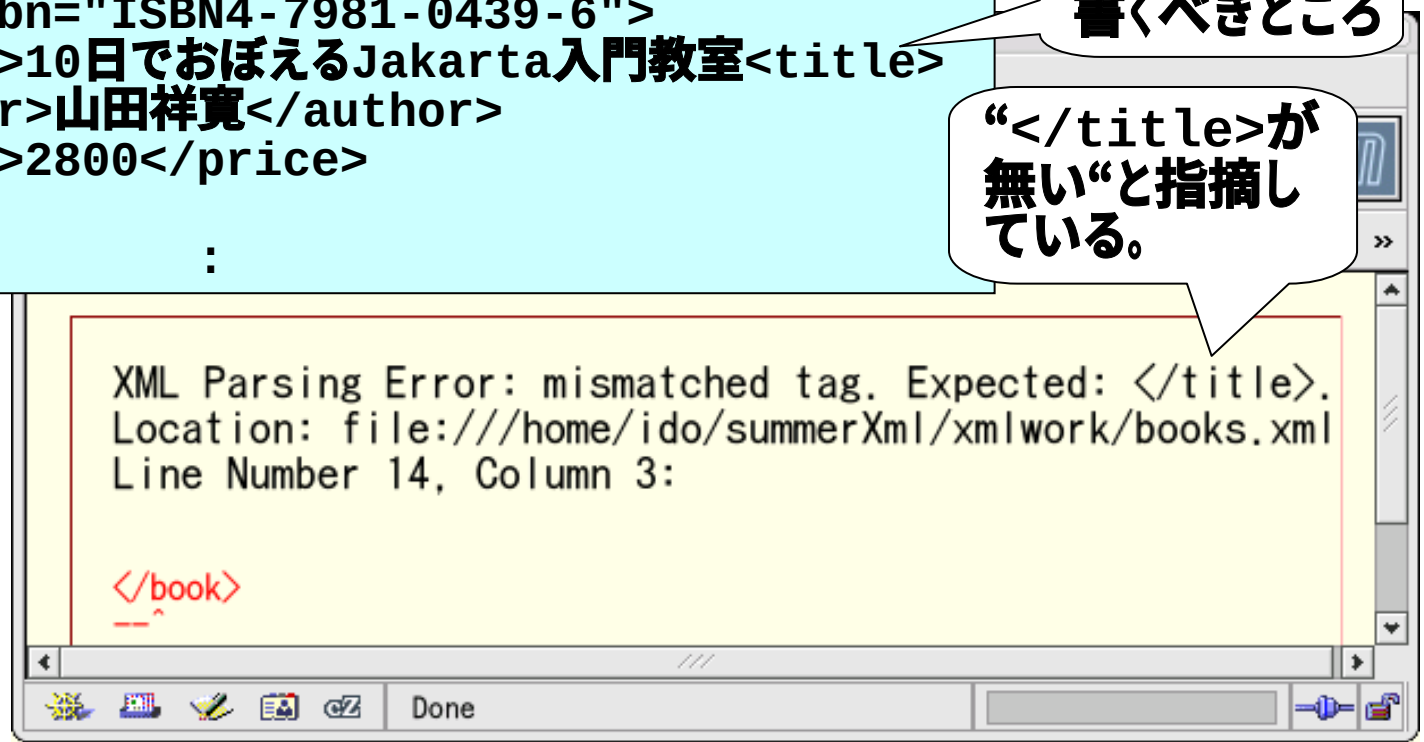
(2．3．2) 注意2：xmlのエラー

■ＸＭＬの構文解析エラー（Parsing Error）がある場合、ブラウザで表示されるエラー箇所よりも、前に実際の間違いがあると思ってください。</xxx>でエラーであれば、これに対応する<xxx>との間にエラーがあります。

```
                    :
10:   <book isbn="ISBN4-7981-0439-6">
11:     <title>10日でおぼえるJakarta入門教室<title>
12:     <author>山田祥寛</author>
13:     <price>2800</price>
14:   </book>
                    :
```

</title>と書くべきところ

“</title>が無い“と指摘している。



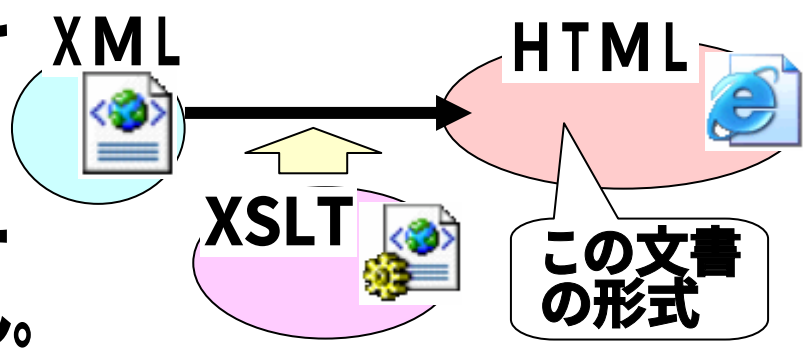
(2.4)ＸＳＬＴスタイルシートの宣言,出力方法の宣言

■ＸＳＬＴスタイルシートは、ＸＭＬの文書の形式にのっとって記述されています。

- ルート要素は、<xsl:stylesheet>要素になります。
- <xsl:stylesheet>要素の属性については、ここではそのように記すものであると理解しておいてください。

```
 1:<?xml version="1.0" encoding="iso-2022-jp" ?>
 2:<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999
/XSL/Transform“ version="1.0">
                  :
21:</xsl:stylesheet>
```

■ <xsl:output>要素では、変換により出力される文書の形式を指定します。文字コードは、ＸＭＬ文書やＸＳＬＴスタイルシートの文字コードではありません。




```
3: <xsl:output method="html" encoding="iso-2022-jp" />
```

(2．5．1) テンプレート －1－

■XSLTスタイルシートは、テンプレートと呼ばれる要素の塊で出来ています (テンプレートが入れ子になることはありません)。

```
<xsl:template match="・・・">
        :
</xsl:template>
        :
<xsl:template match="・・・">
        :
</xsl:template>
```

■各々のテンプレートは、適用されるXML文書の要素ごとに、何を出力するかが示されています。

```
<xsl:template match="xxx">
        ここの内容が、
        適用されたXML文書中の
        要素に対して　出力される
</xsl:template>
```

適用される XMLの要素

(2．5．2) テンプレート －2－

- 4行目から始まるテンプレートは、XML文書の“/”に適用されます(“match=“/””)。
- “/”は文書全体を現しています。よって、XML文書文書全体に対して、スタイルシートの5行目から19行目にしるされた内容が出力されます(この場合、ほとんどはHTMLそのものすね)。

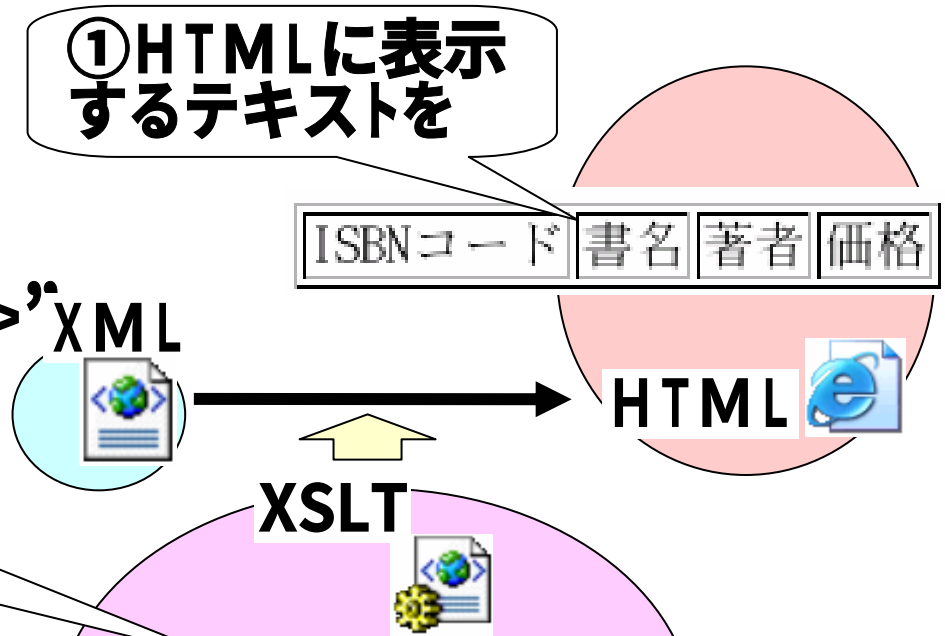
```
 4:   <xsl:template match="/">
            適用される文書全体(“/”)に対して
            この部分が出力される
            (この場合、ほとんどはHTMLそのもの)
20:   </xsl:template>
```

“/”(文書全体)

- 文書全体に適用されるテンプレート(<xsl:template match=“/”>)は、必ず存在しなければなりません。

(2．6) 固定的なテキストの表示

■変換されたHTML上に表示するテキストを直接XSLTに記す際は、次のように“<xsl:text>”要素を用います




```
13:   <tr><th><xsl:text>ISBNコード</xsl:text></th>
14:       <th><xsl:text>書名</xsl:text></th>
15:       <th><xsl:text>著者</xsl:text></th>
16:       <th><xsl:text>価格</xsl:text></th></tr>
```

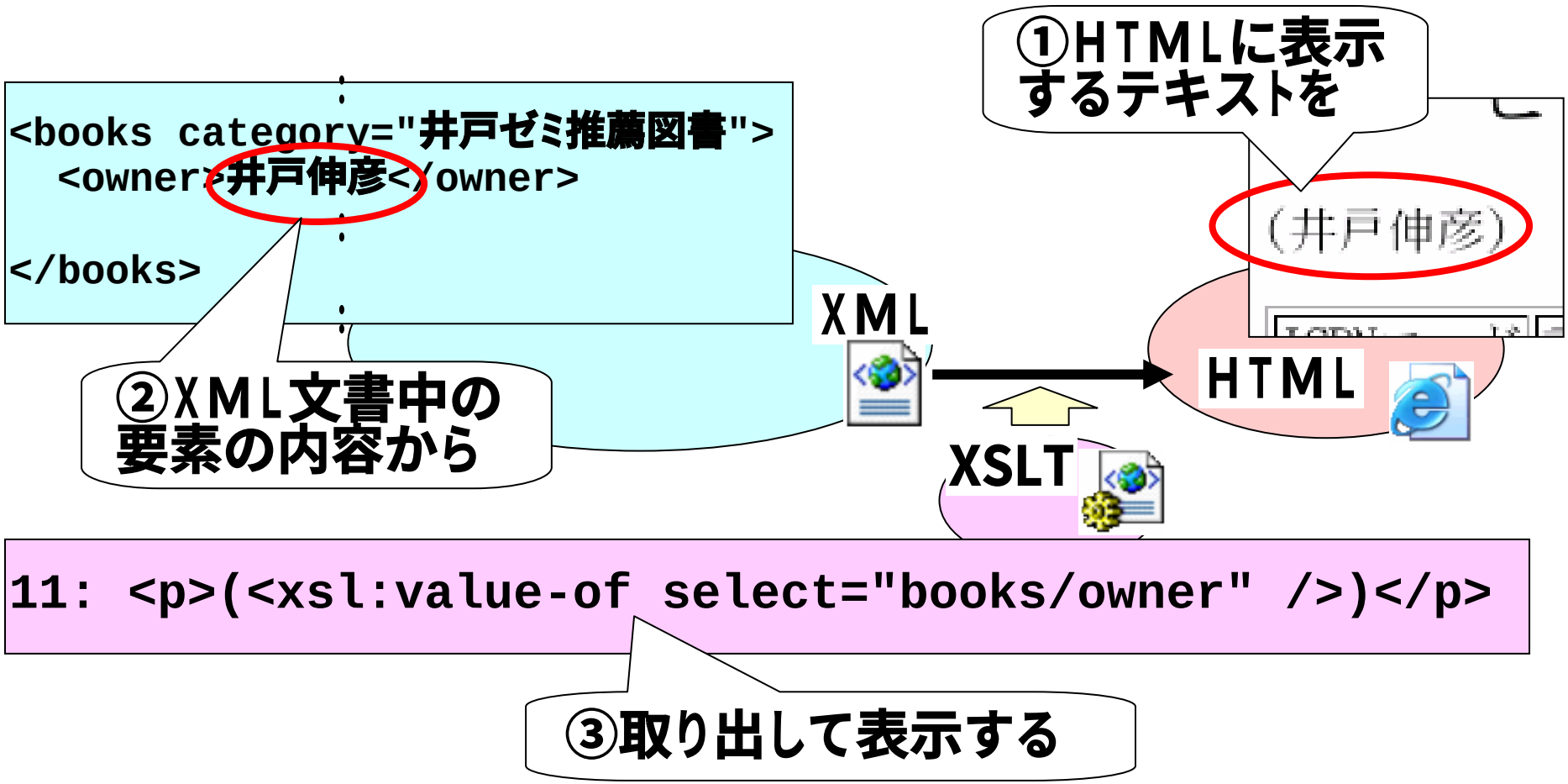
■<xsl:text>は特殊な場合を除いて、省略が可能です。

```
13:   <tr>ISBNコード</th>
```

これでもOK

(2．7．1) XML上の要素の内容の出力ー1ー

■HTML上に表示するテキストを、XML文書中の要素<owner>の内容から取り出して表示するには、“<xsl:value-of>”要素を用います。



(2．7．2) XML上の要素の内容の出力ー2ー

■“<xsl:value-of>”要素では、select属性に取り出す要素を指定します。

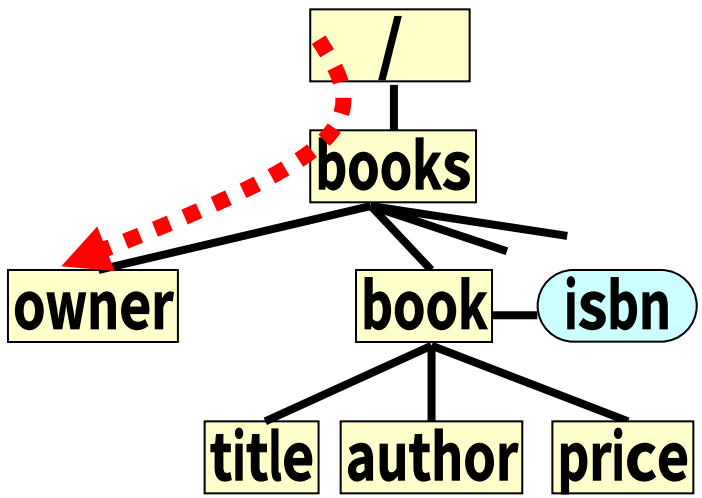
```
11: <p>(<xsl:value-of select="books/owner" />)</p>
```

■11行目の“<xsl:value-of>”要素は、文書全体(“／”)に適用したテンプレートの中にあります。

■よって、“owner”要素を指定するとき、右図のように考えて、“books/owner”と指定します。

■ここの指定の仕方は、“xPath”と呼ぶ規約に基づいています。

■xPathには様々な規約があります。

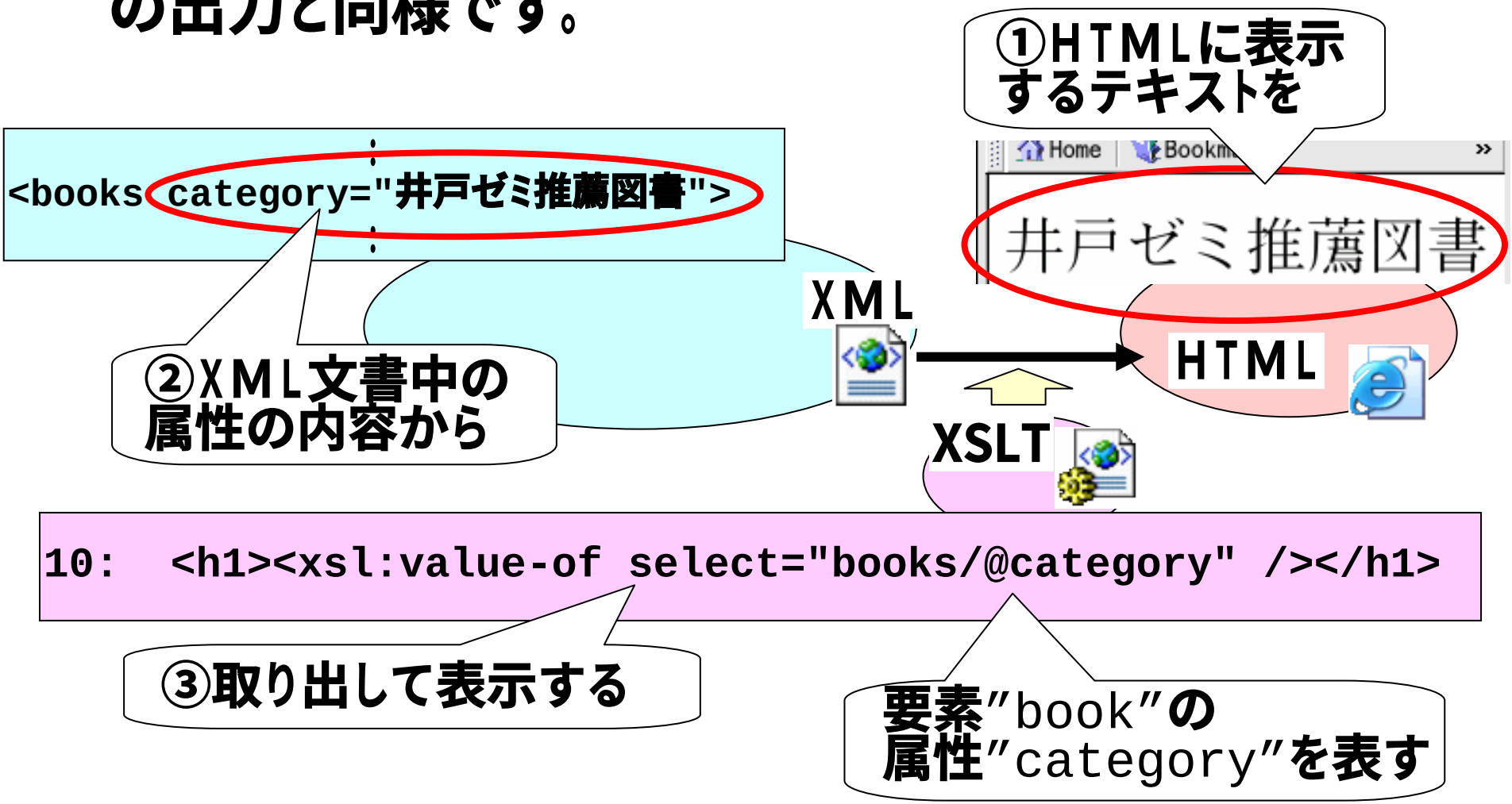


■詳しい解説は、Webサイト等を参照してください。

(http://www.asahi-net.or.jp/~ps8a-okzk/xml/xpath10/new.html)

(2．8) XML上の属性の値の出力

■select属性での指定にて“@”を用いる以外は、要素の出力と同様です。



(2．9) XML文書のスタイルシートへの関連付け

■ＸＭＬ文書“books.xml”では、ＸＳＬＴスタイルシート“books.xsl”に結びつけるために、次のような記述を行います。

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<?xml-stylesheet type="text/xsl" href="books.xsl" ?>
                          :
```

- type属性：スタイルシートの形式を指定します。
- href属性：スタイルシートの格納場所を指定します。

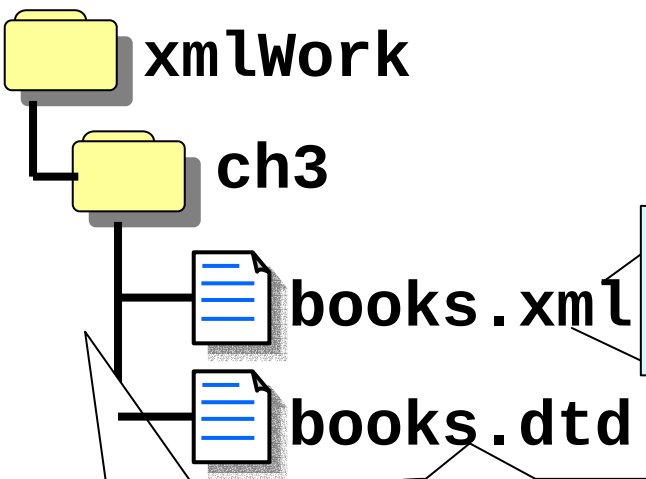
(3) DTDによるXML文書の構文チェック

■スライド(2)で作成したXML文書のDTDファイルを作成し、XML文書の構文チェックをemacs上で行います。

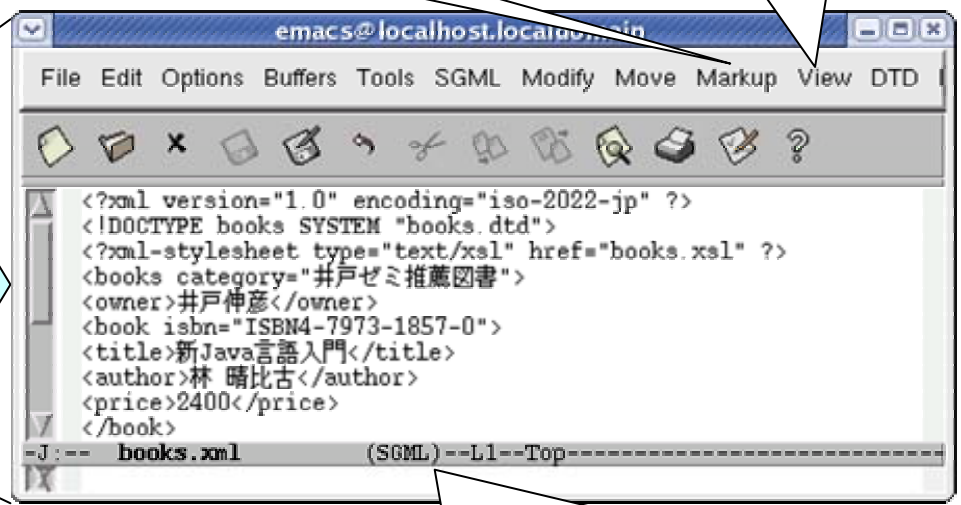
スライド(3.1)
構文チェックとは？

スライド(3.3)
構文チェックの実行

スライド(3.5)
入力補助



emacsで開く



スライド(3.4)
DTDの書き方

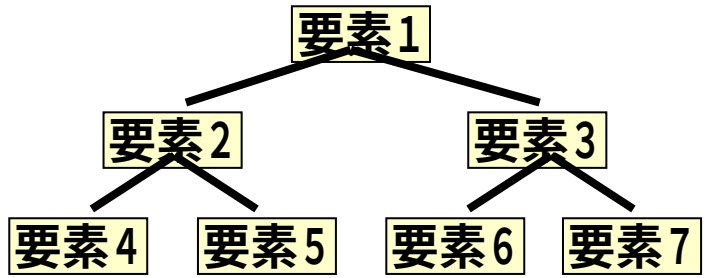
スライド(3.2)
DTDの作成、
XML文書の関連付け

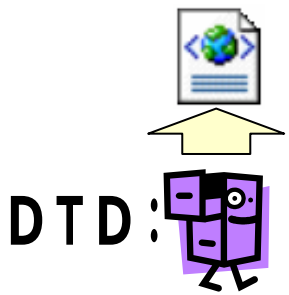
emacsでXMLファイルを開くと、SGMLモードとなり、様々な機能が使用できるようになります。

(3．1) 構文チェック

■構文チェックは、2つの内容に分かれます。

- 構文解析 (Parseing)
  - ◆おおまかに言えば、ＸＭＬ文書が整形式であること (スライド(1.4)) のチェックです。
  - ◆ちゃんと木構造になっているかをチェックします。
- 妥当性判定 (Validation)
  - ◆妥当なＸＭＬ文書であるかを、ＤＴＤに基づきチェックします。





■mozillaの表示においても、構文解析は行っています (間違えるとエラーが表示されましたね) 。

■emacsでは、上記の両方のチェックが出来、さらにＤＴＤに基づく入力補助機能があります。

(3.2) ＤＴＤの作成、ＸＭＬ文書の関連付け

■スライド(2.1.1)に示したＸＭＬ文書のＤＴＤを、次のように編集します。

＜books.dtd＞

```
<!ELEMENT books (owner,book*)>
<!ELEMENT book (title,author,price)>
<!ELEMENT owner (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ATTLIST books category CDATA #IMPLIED>
<!ATTLIST book isbn CDATA #REQUIRED>
```

■上記ＤＴＤへの対応付けを、ＸＭＬ文書に記述します。

＜books.xml＞

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<!DOCTYPE books SYSTEM "books.dtd">
<?xml-stylesheet type="text/xsl“...
                   :
```

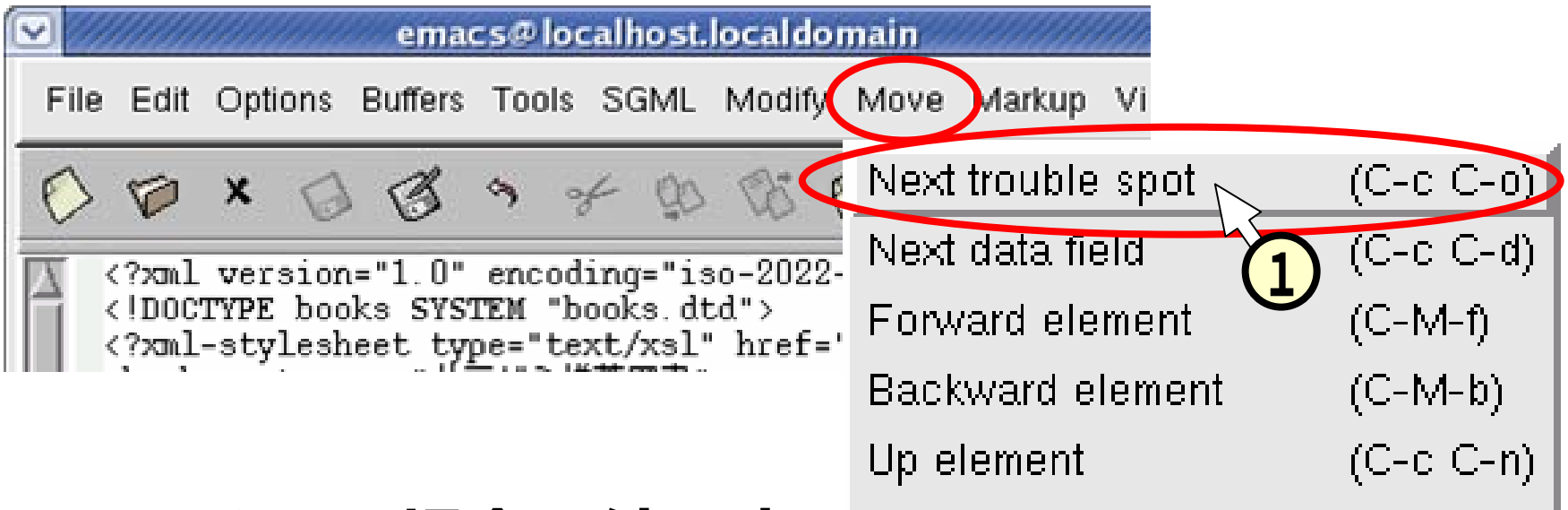
(3．3．1) 構文チェックの実行

■チェックするＸＭＬファイル (books.xml) を開いて、メニューより、[DTD]-[Parse DTD]をクリック (①) します。

■ショートカットの“C-c C-p”([Ctrl]キーを押下しながら[c]のキーを押下、続いて、[Ctrl]キーを押下しながら[p]のキーを押下)でも、もちろんＯＫです。

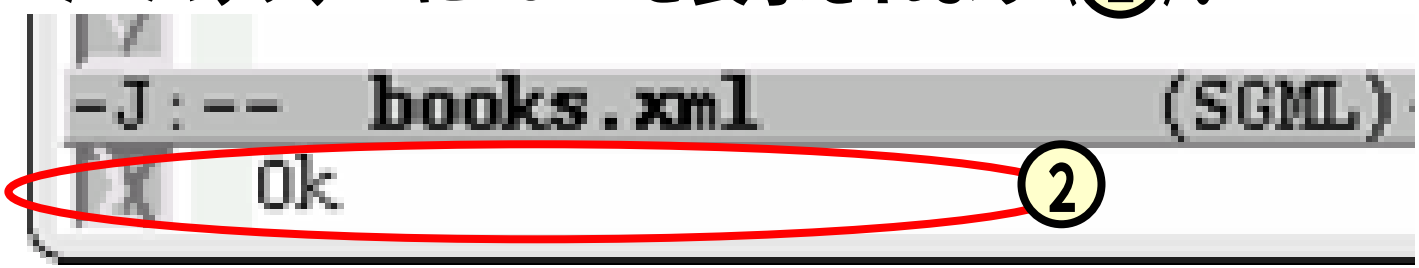
(3．3．2) 結果の表示

■先頭にカーソルを移動させて、メニューから[Move]-[Next trouble spot (C-c C-o)]をクリック (①) します。



■エラーなしの場合の結果表示

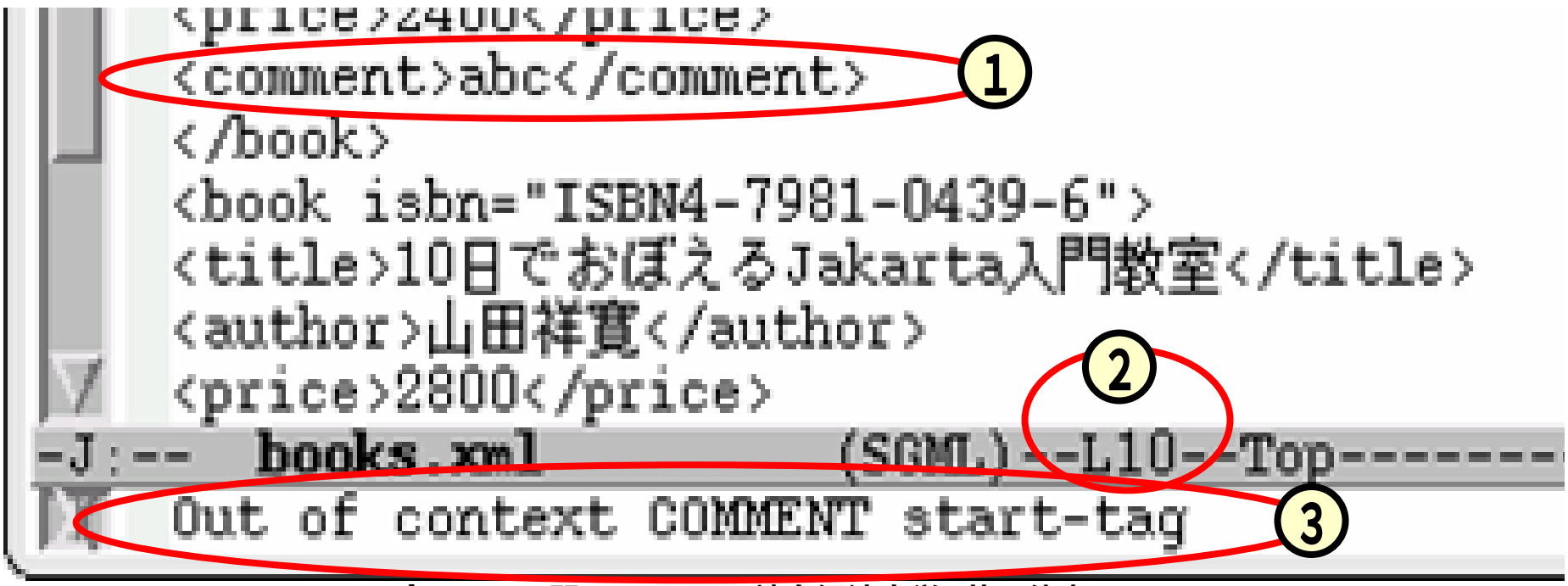
- ミニバッファーに“OK”と表示されます (②)。



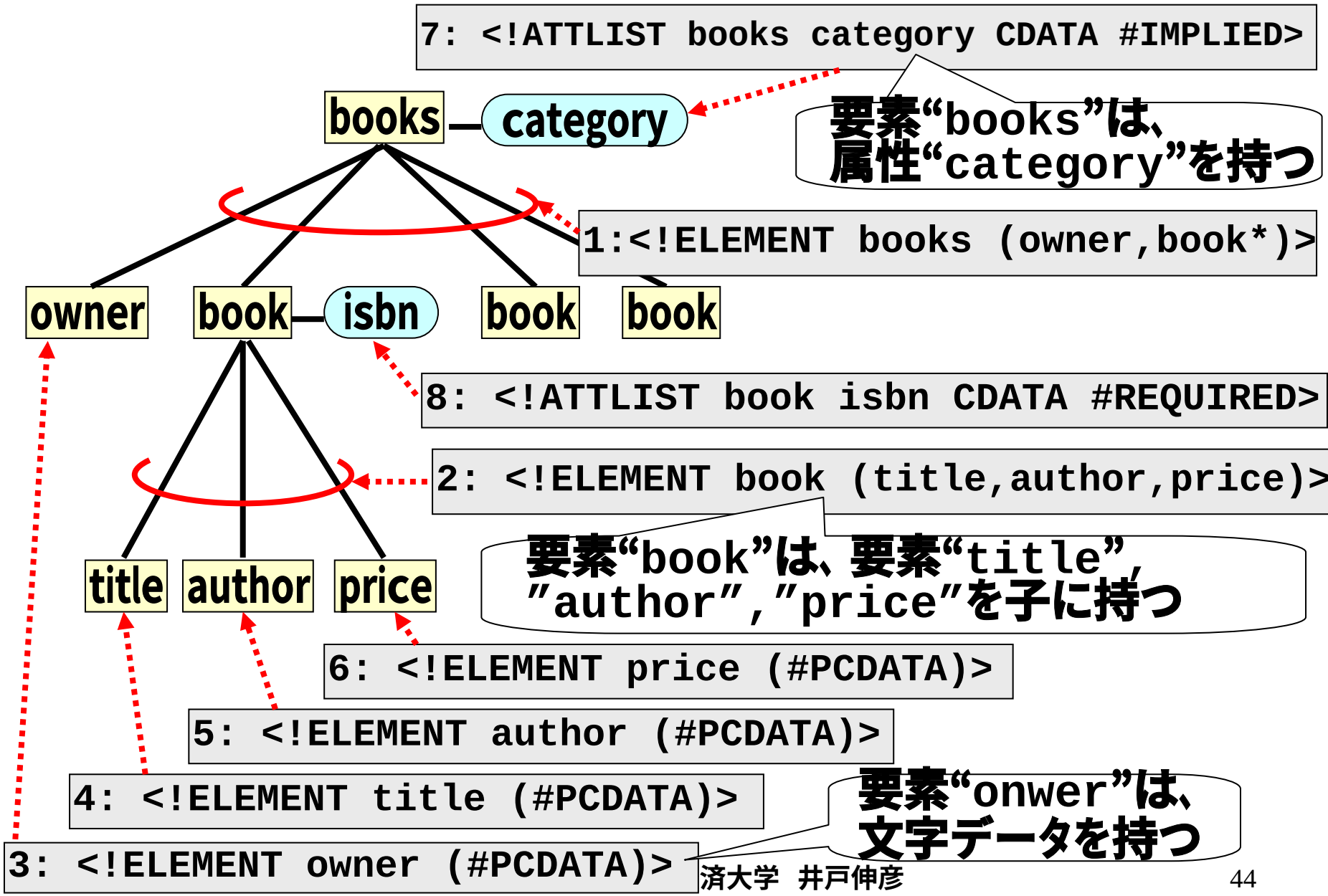
(3．3．3) エラーがある場合

■DTDに定義されていない要素“<comment>”を入れて、わざとエラーを混入しました(10行目、①)。

■[Move]-[Next trouble spot (C-c C-o)]をクリックすると、エラーのある行でカーソルが止まり(②)、ミニバッファーには、次のようにエラーが表示されます(③)。



(3．4．1) DTDの概要


7: <!ATTLIST books category CDATA #IMPLIED>
books
category
要素“books”は、
属性“category”を持つ
1:<!ELEMENT books (owner,book*)>
owner
book
isbn
book
book
8: <!ATTLIST book isbn CDATA #REQUIRED>
2: <!ELEMENT book (title,author,price)>
要素“book”は、要素“title”,
”author”,”price”を子に持つ
title
author
price
6: <!ELEMENT price (#PCDATA)>
5: <!ELEMENT author (#PCDATA)>
4: <!ELEMENT title (#PCDATA)>
3: <!ELEMENT owner (#PCDATA)>
要素“onwer”は、
文字データを持つ

(3．4．2) 要素の宣言

■要素の宣言は、次の形式で記述します。

```
1:<!ELEMENT books (owner,book*)>
```

定義する要素

定義する要素の子の要素

出現回数を示す記号

- 出現回数を記す記号は、次の意味になります。
  - なし ：かならず1回出現
  - ？　：0回、もしくは、1回出現
  - ＊　：0回以上出現
  - ＋　：1回以上出現
- “、”(カンマ) で区切られた子要素は、その順番に出現することを意味します。
- 次の区切り“｜”を用いた指定では、子要素”bbb”と子要素“bbb”のいずれか一方が出現することを意味します。

```
<!ELEMENT aaa (bbb|ccc)>
```

(3．4．3) 要素に含まれるデータ

■要素の宣言は、次の形式で記述します。

```
3: <!ELEMENT owner (#PCDATA)>
```

定義する要素

文字データを意味するキーワード

■子要素”bbb”の後に、文字データが現れる場合、次のように記します。

```
<!ELEMENT aaa (bbb,#PCDATA)>
```

■子要素”bbb”と文字データとが混じって0回以上現れる場合、次のように記します。

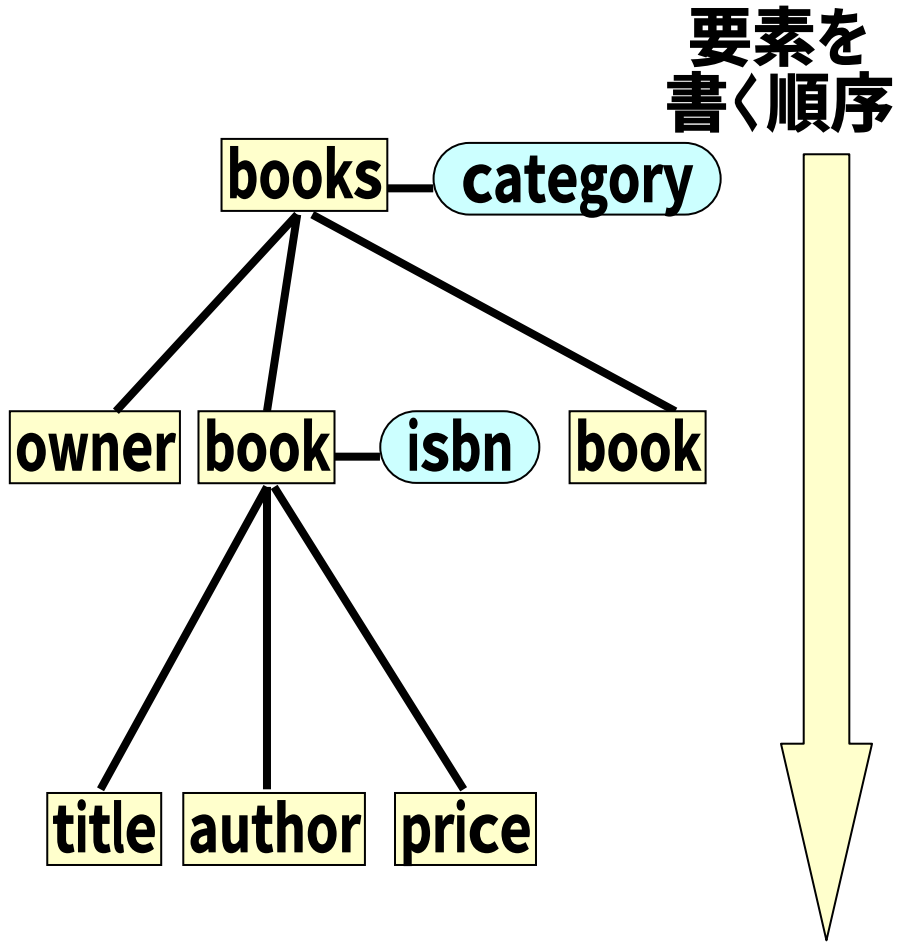
```
<!ELEMENT aaa (bbb|#PCDATA)*>
```

```
<aaa>
あいうえお<bbb></bbb><bbb></bbb>かきくけこ
</aaa>
```

(3．4．4) 要素を書く順序

■ＤＴＤ中、要素の定義は、親から子への順序で記述していきます。

■すなわち、子孫にあたる要素が、祖先に当たる要素よりも先に現れないように記述していきます。



(3．4．5) 属性の定義

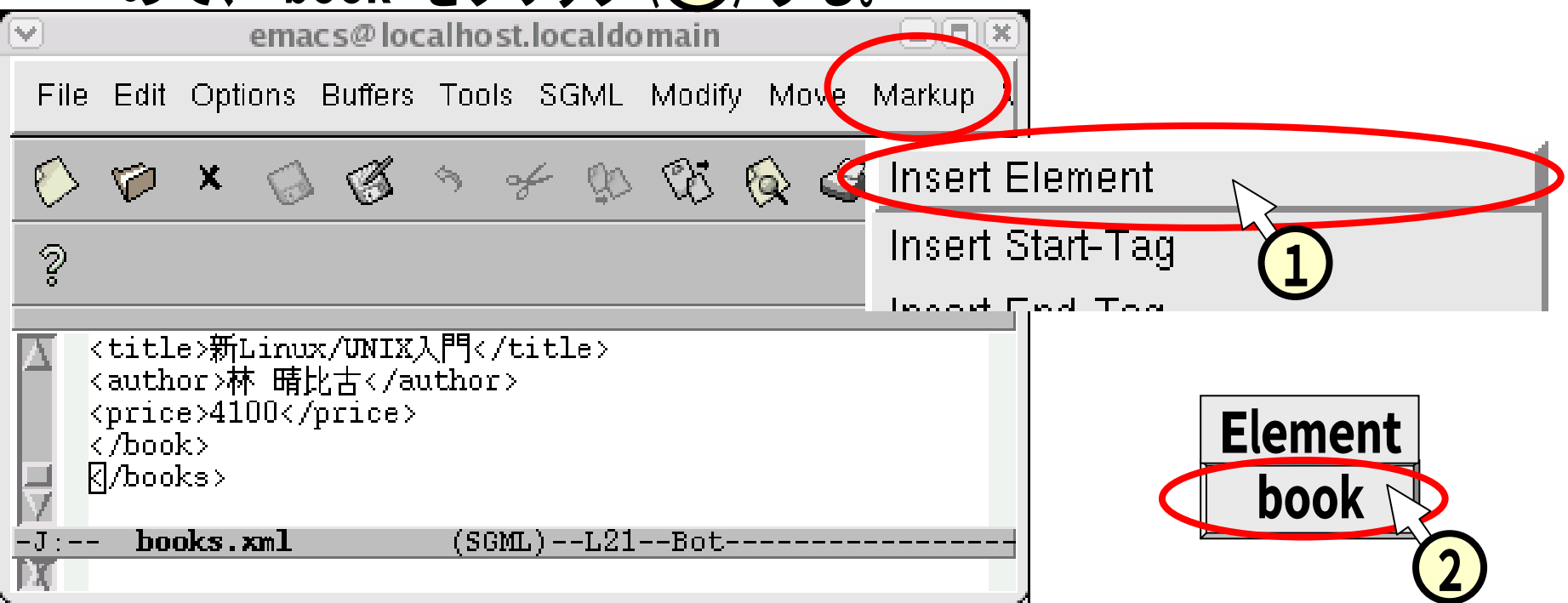
■属性の定義は、次のように記述します。

```
8: <!ATTLIST book isbn CDATA #REQUIRED>
```

- 属性のデータ型には色々ありますが、本スライドでは、ＣＤＡＴＡ（＝文字列）のみを扱うこととします。
- デフォルト値には、具体的な値をそのまま書く以外に、次のような記述が可能です。
  - ◆#REQUIRED　：属性値は必須
  - ◆#IMPLIED　：属性値は任意（省略可能）
  - ◆#FIXED “aaa”　：属性値は固定（この場合、aaaに固定）

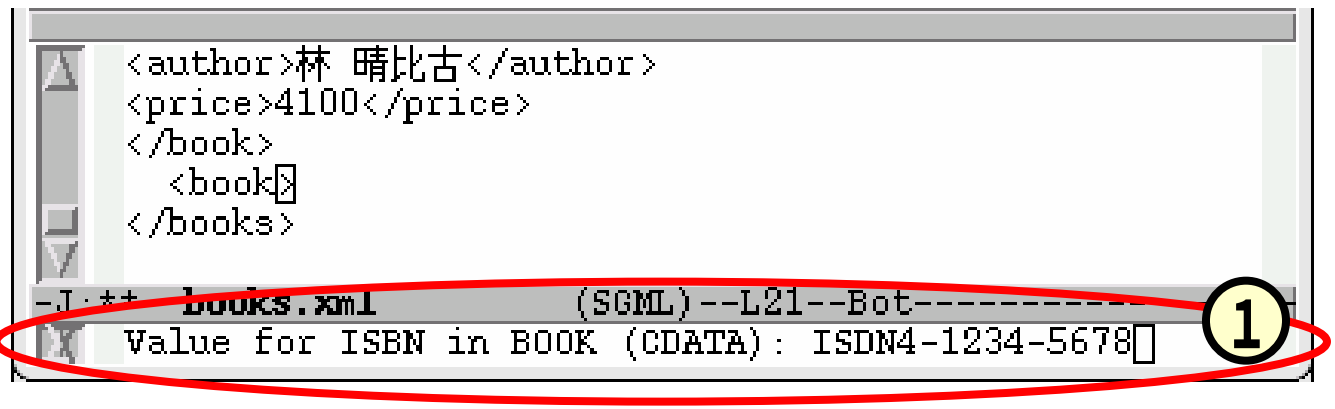
(3．5．1) 入力補助　－1－

- XML文書の構文チェックを行った状態 (スライド(3.3)の状態) のemacsは、ＤＴＤに記された構文に基づく入力補助を行います。
- 終了タグ“</book>”の後にカーソルをおいて、メニューから[Markup]-[Insert Element]をクリック (①) します。
- 候補となる要素 (この場合は“book”のみ) のメニューが現れるので、“book”をクリック (②) する。

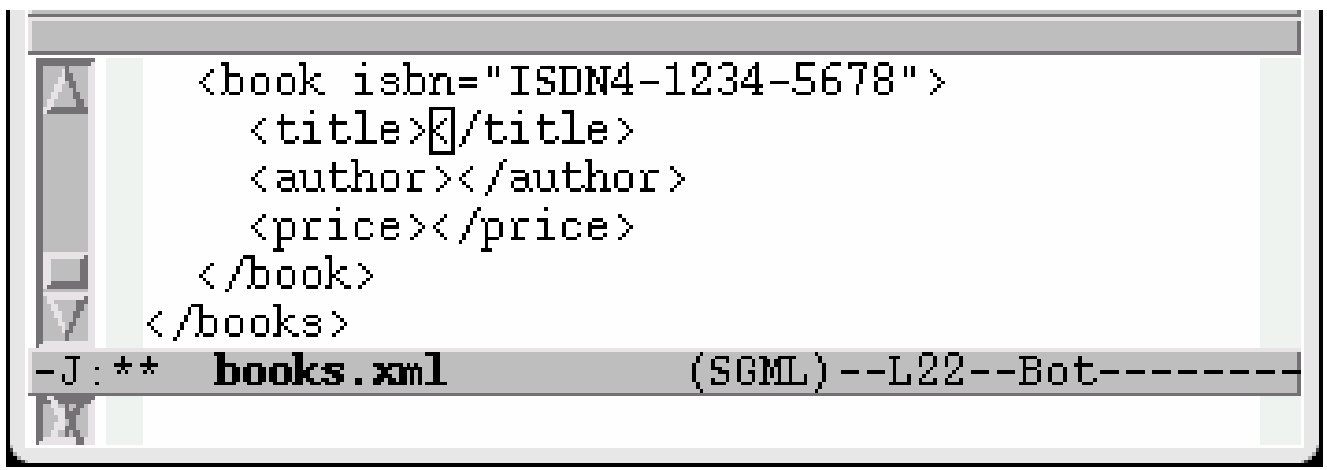


(3．5．2) 入力補助　－2－

■ミニバッファーにて、“book”要素の必須属性である“isdn”の属性値を聞かれます。入力して(①)、[Enter]を押下します。



■属性ISDNが入力され、必須の要素のタグが入力されます。



(3．6) 課題

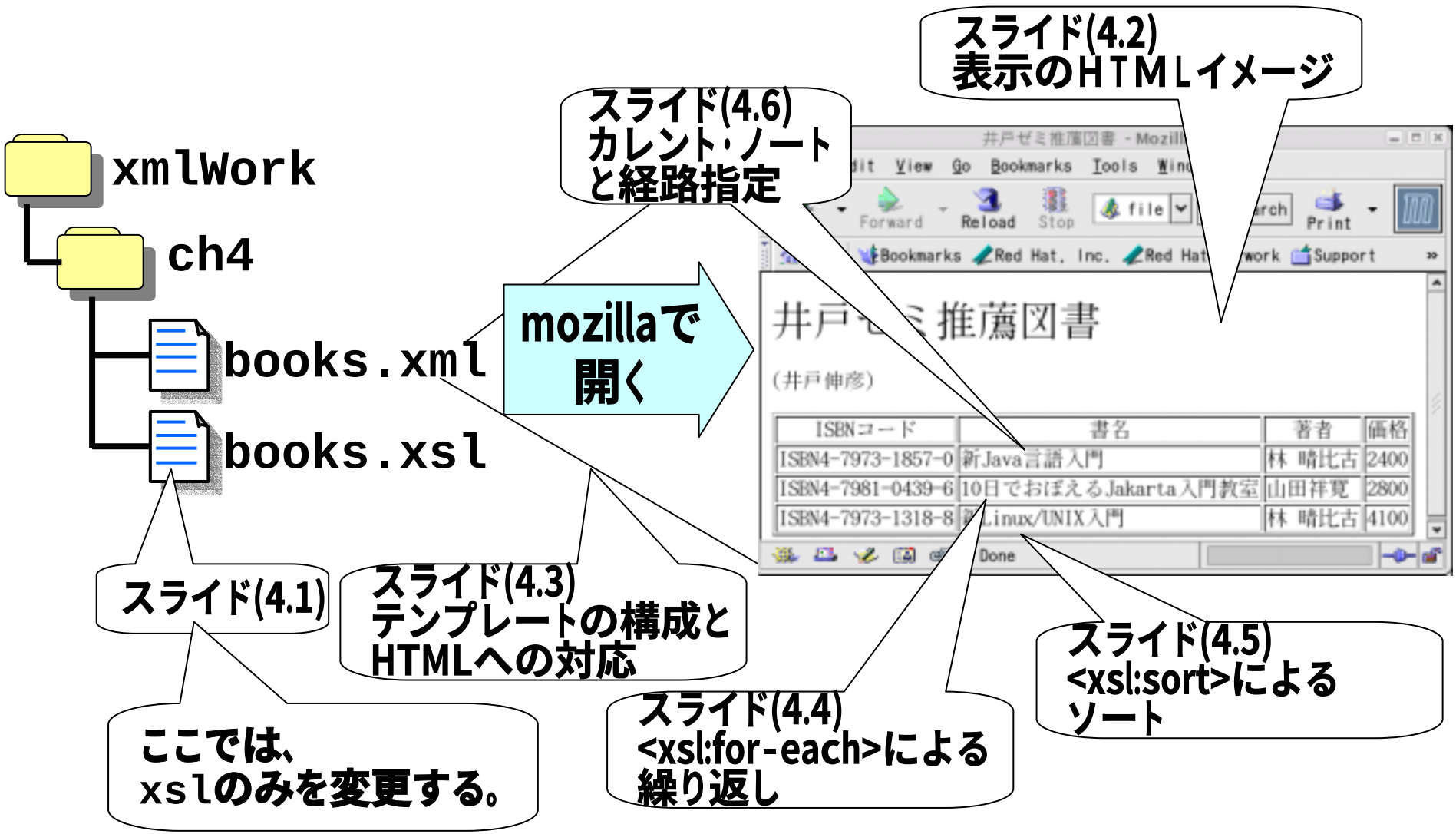
■Webサイトから次のような表を探し、そのXML文書、DTDを作成してください。

- 複数の野球選手のプロフィール
- 複数のPCの製品仕様
- 日本の各県の統計データ
- プロスポーツチームの戦績
- 星座に関する情報一覧
- 列車に関する情報一覧
- その他なんでもOKです。

■XML文書には、すべてのデータを入力する必要はありません。動作が確かめられるだけの数があれば十分です。

(4) 一覧表を表示する

■XML文書中の図書を一覧表で表示します。



(4．1) books.xslの変更

■スライド(2.2)のbooks.xslに、次のリストの行番号が入った部分を追加します。

＜books.xsl＞

```
                    :(ここまでは、スライド(2.1)と同じ)
      <th><xsl:text>価格</xsl:text></th></tr>
19:    <xsl:apply-templates select="books" />
      </table>
   </body>
   </html>
   </xsl:template>
24:<xsl:template match="books">
25: <xsl:for-each select="book">
26:  <xsl:sort select="price" data-type="text"
order="ascending" />
27:      <tr><td><xsl:value-of select="@isbn" /></td>
28:          <td><xsl:value-of select="title" /></td>
29:          <td><xsl:value-of select="author" /></td>
30:          <td><xsl:value-of select="price" /></td></tr>
31: </xsl:for-each>
32:</xsl:template>
   </xsl:stylesheet>
```

改行なし

(4．2) 表示されるHTML

■表示されるHTMLのイメージは、次のとおりです。

```
<html>
<head>
<META http-equiv="Content-Type" content="text/html;
 charset=iso-2022-jp">
<title>井戸ゼミ推薦図書</title>
</head>
<body>
<h1>井戸ゼミ推薦図書</h1>
<p>井戸伸彦</p>
<table border="1">
<tr><th>ISBNコード</th><th>書名</th><th>著者</th><th>価格</th></tr>
<tr><td>ISBN4-7973-1857-0</td><td>新Java言語入門</td>
    <td>林 晴比古</td><td><font color="red">2400</font></td></tr>
<tr><td>ISBN4-7981-0439-6</td><td>10日でおぼえるJakarta入門教室</td>
    <td>山田祥寛</td><td><font color="red">2800</font></td></tr>
<tr><td>ISBN4-7973-1318-8</td><td>新Linux/UNIX入門</td>
    <td>林 晴比古</td><td>4100</td></tr>
</table>
</body>
</html>
```

(4．3．1) 2つめのテンプレート

■スタイルシートは、2つのテンプレートを含みます。

■前のテンプレートの中で、後のテンプレートを呼び出しています。

＜books.xsl＞

```
 4:<xsl:template match="/">          (テンプレート1)
19:<xsl:apply-templates select="books" />
23:</xsl:template>
```

後のテンプレート2を呼び出す

```
24:<xsl:template match="books"> (テンプレート2)
32:</xsl:template>
```

(4．3．2) 出力されるHTMLとの対応

■変換されるHTMLは、それぞれのテンプレートにより、次のように出力されます。

＜books.xsl＞

(テンプレート1)

<xsl:apply-templates ～/>

(テンプレート2)

＜HTMLのイメージ＞

```
<html>
<head>
<META http-equiv="Content-Ty
 charset=iso-2022-jp">
<title>井戸ゼミ推薦図書</title>
</head>
<body>
<h1>井戸ゼミ推薦図書</h1>
<p>井戸伸彦</p>
<table border="1">
<tr><th>ISBNコード</th><th>書名
<tr><td>ISBN4-7973-1857-0</t
    <td>林 晴比古</td><td><fo
<tr><td>ISBN4-7981-0439-6</t
    <td>山田祥寛</td><td><fon
<tr><td>ISBN4-7973-1318-8</
    <td>林 晴比古</td><td>41
</table>
</body>
</html>
```

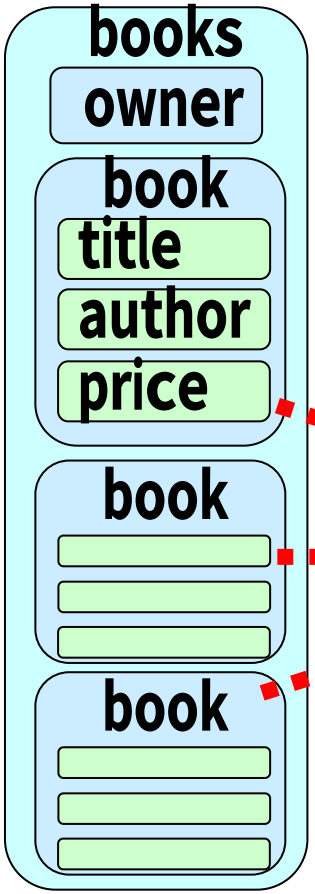
テンプレート1の部分は、スライド(2)と同じ

(4．4)<xsl:for-each>要素

■“books”に適用される2つめのテンプレートでは、<xsl:for-each>要素により、ＸＭＬ文書内の<book>要素が繰り返し処理され、ＨＴＭＬに出力されます。

```
24:<xsl:template match="books">
25: <xsl:for-each select="book">
          ここの部分が、
          要素“book”に繰り返し
          適用される
31: </xsl:for-each>
32:</xsl:template>
```

＜books.xml＞



＜books.xsl＞

```
<xsl:for-each ...>

</xsl:for-each>
```

＜ＨＴＭＬのイメージ＞

```
<tr><td>ISBN4-7973-1857-0</t
    <td>林 晴比古</td><td><fo

<tr><td>ISBN4-7981-0439-6</t
    <td>山田祥寛</td><td><fon

<tr><td>ISBN4-7973-1318-8</
    <td>林 晴比古</td><td>41
```

(4.5)<xsl:sort>要素

■ＸＭＬ文書中の<book>要素は、<xsl:for-each>により繰り返し処理される際、<xsl:sort>要素により並べ替え(ソート)されます。

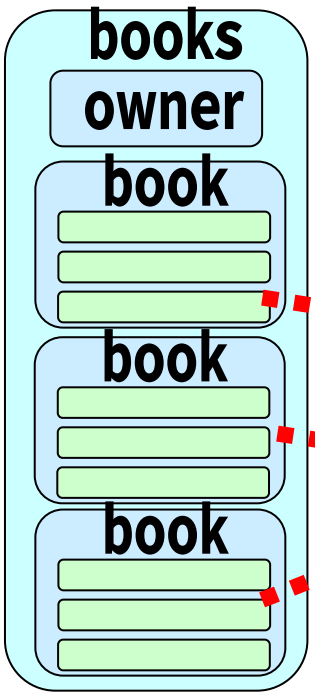
price要素の内容でソート

```
26:  <xsl:sort select="price" data-type="text"
                order="ascending" />
```

昇順でソート

文字としてソート
(数としてのソートでは、
“number”とする)

<books.xml>



<books.xsl>

```
<xsl:for-each ...>
<xsl:sort>
</xsl:for-each>
```

price要素を
文字として
昇順でソート

<HTMLのイメージ>

```
:
<tr><td>ISBN4-7973-1857-0</t
    <td>林 晴比古</td><td><fo

<tr><td>ISBN4-7981-0439-6</t
    <td>山田祥寛</td><td><fon

<tr><td>ISBN4-7973-1318-8</
    <td>林 晴比古</td><td>41
:
```

(4．6．1) 要素の経路 (ロケーションパス) 指定

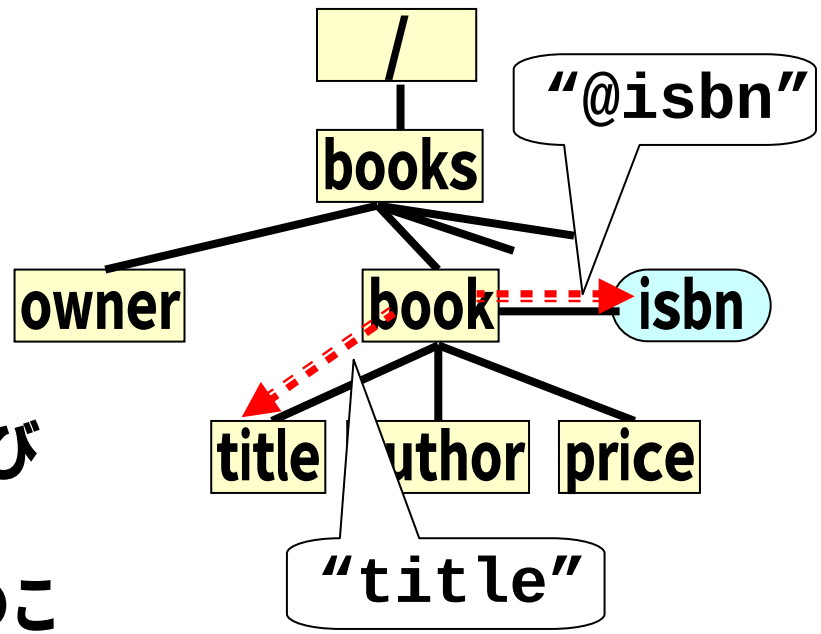
■<xsl:for-each>要素内での<vsl:value-of>要素では、要素の指定を次のように行っています。

```
25: <xsl:for-each select="book">
27:  <tr><td><xsl:value-of select="@isbn" /></td>
28:       <td><xsl:value-of select="title" /></td>
                    :
31: </xsl:for-each>
```

■ これは、要素“book”から見た経路による指定です。

■ ノードを呼ぶ元となる位置を、カレント・ノードと言います。

- 木構造での要素を、ノードと呼びます。
- 木構造にもとづく要素の指定のことを、ロケーションパスと言います。

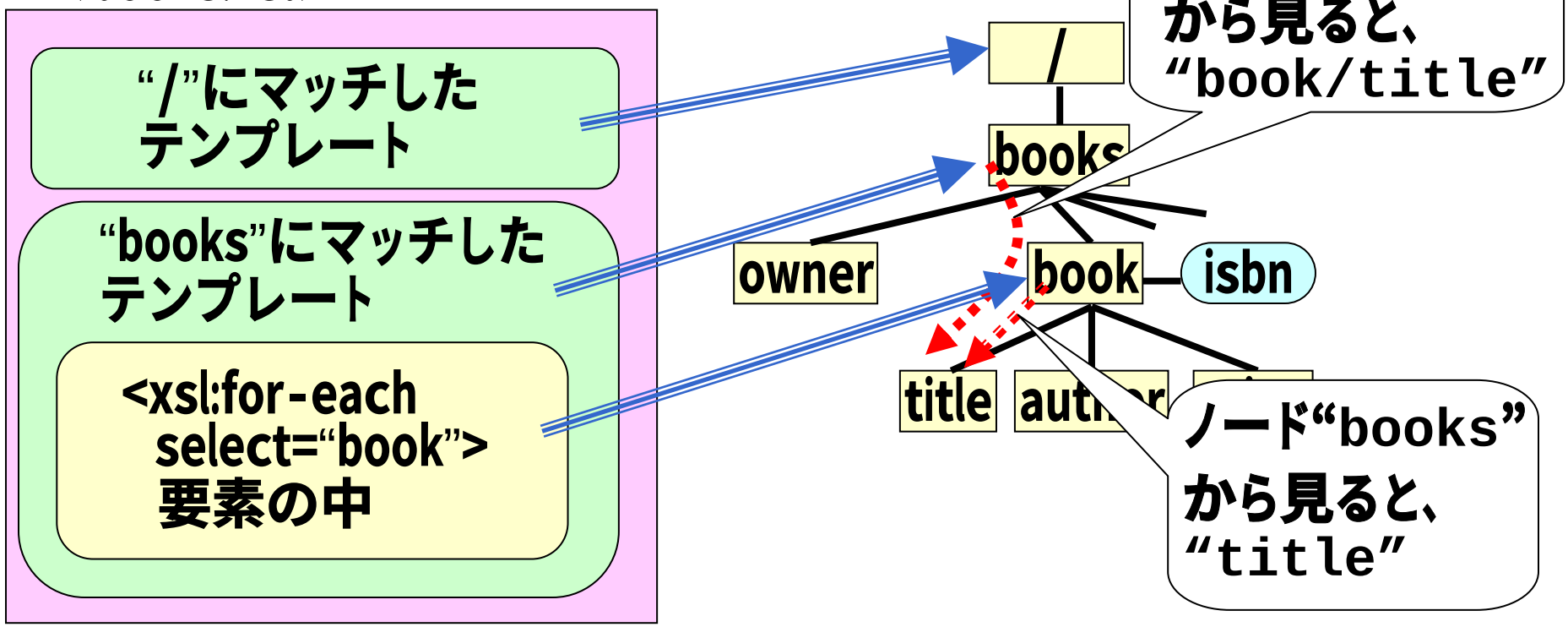


(4．6．2) カレント・ノードの移動

■カレントノードは、スタイルシートのどの位置かにより、変わってきます。

■(4)でのスタイルシートでは、カレントノードは次のようになります。

＜books.xsl＞



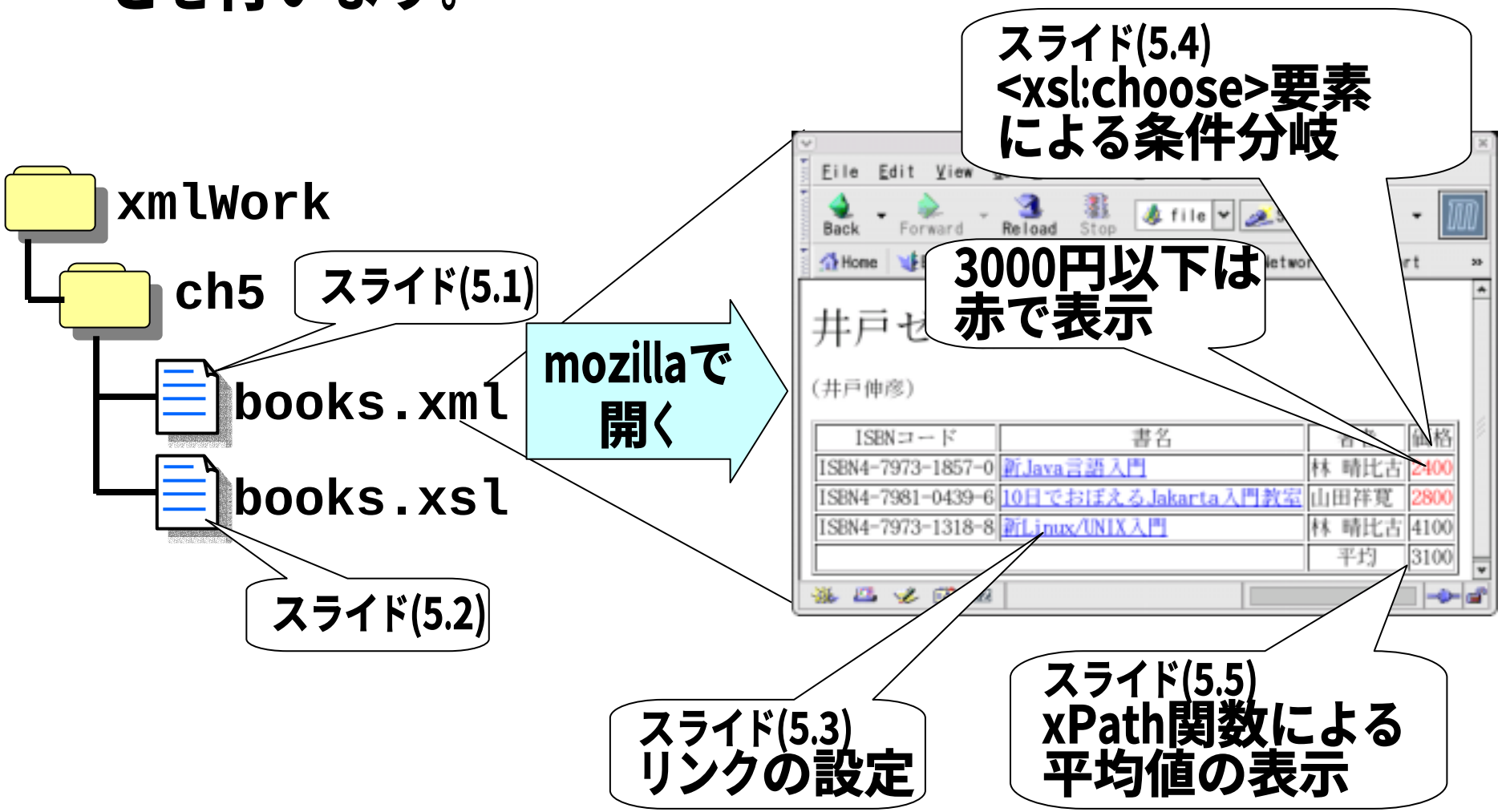
(4．7) 課題

■スライド(3．6)で作成したＸＭＬ文書を表示させるＸＳＬＴスタイルシートを作成してください。

- スライド(4)で出てきた手法をすべて盛り込んでください。

(5) 表示を充実させる

■リンク、条件分岐による表示の変更、平均値の計算などを行います。



(5．1) books.xmlの変更

■要素<url>を追加します。

＜books.xml＞

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<!DOCTYPE books SYSTEM "books.dtd">
<?xml-stylesheet type="text/xsl" href="books.xsl" ?>
<books category="井戸ゼミ推薦図書">
<owner>井戸伸彦</owner>
<book isbn="ISBN4-7981-0439-6">
        :
<price>2800</price>
<url>http://www.amazon.co.jp/exec/obidos/ASIN/xxx</url>
</book>
<book isbn="ISBN4-7973-1318-8">
        :
<price>4100</price>
<url>http://www.amazon.co.jp/exec/obidos/ASIN/xxx</url>
</book>
<book isbn="ISBN4-7973-1857-0">
        :
<price>2400</price>
<url>http://www.amazon.co.jp/exec/obidos/ASIN/xxx</url>
</book>
</books>
```

(5．2．1) books.xslの変更 (1／2)

■スライド (4.1) のbooks.xslに、次のリストの行番号が入った部分を追加します。

＜books.xsl＞

```
   <?xml version="1.0" encoding="iso-2022-jp" ?>
            :(中略)
   <table border="1">
       <tr>
         <th><xsl:text>ISBNコード</xsl:text></th>
         <th><xsl:text>書名</xsl:text></th>
         <th><xsl:text>著者</xsl:text></th>
         <th><xsl:text>価格</xsl:text></th>
       </tr>
       <xsl:apply-templates select="books" />
20:    <tr><td colspan="2" /><th>平均</th>
21:        <td><xsl:value-of select="sum(books//price)
div count(books//price)" /></td></tr>
       </table>
       </body>
       </html>
     </xsl:template>
```

改行なし

(5．2．2) books.xslの変更 (2／2)

```
   <xsl:template match="books">
      <xsl:for-each select="book">
        <xsl:sort select="price" data-type="text"
order="ascending" />
           <tr><td><xsl:value-of select="@isbn" /></td>
30:            <td><xsl:element name="a">
31:                  <xsl:attribute name="href">
32:                    <xsl:value-of select="url" />
33:                  </xsl:attribute>
34:                  <xsl:value-of select="title" />
35:                </xsl:element></td>
36:            <td><xsl:value-of select="author" /></td>
37:            <td><xsl:choose>
38:                  <xsl:when test="price[number(.) &lt;=3000]">
39:                    <font color="red">
<xsl:value-of select="price" /></font>
40:                  </xsl:when>
41:                  <xsl:otherwise>
42:                    <xsl:value-of select="price" />
43:                  </xsl:otherwise>
44:                </xsl:choose></td></tr>
      </xsl:for-each>
    </xsl:template>
  </xsl:stylesheet>
```

改行なし

改行なし

(5．3．1) リンク：予想される方法は駄目

■ご存知のとおり、“こちらへ”という表示に、“http://yyy”へのリンクを設定したい場合、HTMLでは次のように記述します。

```
<a href=“http://yyy”>おはよう</a>
```

■このhttp://yyyのところへ、XML文書から取り出した値を入れたい時、次のようにしたくなります。

＜駄目な例：XSLTスタイルシート＞

```
<a href=“<xsl:value-of select=“...”>”>おはよう</a>
```

■ところがこれはうまく行きません。HTMLのタグ内の属性値に、<xsl:～>のタグは代入出来ません。

(“” (ダブル・クォーテーション) の内部はXSLTの変換処理がされないということです。)

(5．3．2) リンク：記述方法

■スタイルシート内では、先ほどのリンクを次のように記述します。

＜XSLTスタイルシート＞

```
<xsl:element name="a">
  <xsl:attribute name="href">
    <xsl:value-of select="..." />
  </xsl:attribute>
  おはよう
</xsl:element>
```

＜駄目な例＞

```
<a
href="
<xsl:value-of ...>
">
おはよう
</a>
```

■上記に示した“駄目な例”との対応が示すように、次のように記述します。

- 属性を持つ要素 (<a>) は、<xsl:element>要素に置き換える。
- 属性 (href=“～”) は、<xsl:attribute>要素に置き換え、<xsl:element>要素内に記す。
- 属性値は、上記<xsl:attribute>要素内に記す。ここでは、<xsl:value-of>要素が使える。

（5．3．3）books.xsl内でのリンク

■books.xsl内では、次のようにリンクを設定しています。

＜books.xsl＞

```
<td><xsl:element name="a">
          <xsl:attribute name="href">
            <xsl:value-of select="url" />
          </xsl:attribute>
          <xsl:value-of select="title" />
        </xsl:element></td>
```

■たとえば、XML文書中より、各要素へ次の値が読み出されたとします。

- <xsl:value-of select=“url” /> ← “http://www...”
- <xsl:value-of select=“title” /> ← “新Java言語入門”

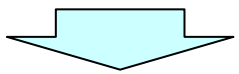
■このとき、次のようなリンクがＨＴＭＬ上に作成されます。

```
<a href=“http://www...”>新Java言語入門</a>
```

(5．4．1) 分岐処理

■books.xsl中の次の分岐処理を、やや乱暴ですが、プログラム風に書いて見ます。

```
37:  <td><xsl:choose>
38:        <xsl:when test="price[number(.) &lt;=3000]">
39:          <font color="red">
             <xsl:value-of select="price" /></font>
40:        </xsl:when>
41:        <xsl:otherwise>
42:          <xsl:value-of select="price" />
43:        </xsl:otherwise>
44:      </xsl:choose></td></tr>
```



```
37:38: if("price[number(.) &lt;=3000]“){
39:          <font color="red">
             <xsl:value-of select="price" /></font>
40:    }
41:     else{
42:          <xsl:value-of select="price" />
43:44:     </xsl:otherwise>
44:    }
```

(5.4.2) <xsl:choose>, <xsl:when>, <xsl:otherwise>

■<xsl:choose>, <xsl:when>, <xsl:otherwise>により、次のように条件分岐を記述します。

```
<xsl:choose>
  <xsl:when test="条件1">
        条件1が成立したときに出力される記述
  </xsl:when>
  <xsl:when test="条件2">
        条件2が成立したときに出力される記述
  </xsl:when>
  <xsl:otherwise>
        条件1、条件2とも成立しなかったときに出力される記述
  </xsl:otherwise>
</xsl:choose></td></tr>
```

■<xsl:when>要素はいくつあってもかまいません。すなわち、上記は多岐分岐に使われる構文です。

(5．4．3) 条件、フィルタパターン

■<xsl:when>要素の中で、条件にあたる“test”属性は、次のように記述されています。

```
38: <xsl:when test="price[number(.) &lt;=3000]">
```

■test属性値は、次のようになっています。

- 属性値全体は、xPathの記述規則に従っている。
- 典型的には、指定した要素（上記例では“price”）をカレント・ノードとして、条件を[...]の中に記す。
  （[...]をフィルタパターンと呼ぶ。）
- xPathでは“.”はカレントノード自身を表す。
- xPathでは、四則演算や比較演算子、論理演算子などの演算子が使える。
- xPathでは、さまざまな関数（上記では、numberがxPath関数）が使える。

(5．4．4) xPath：演算子

■前述のとおり、xPathは非常に多岐に亘る規約のため、本スライドでは限定されたものだけを紹介します。

■演算子

- 次の演算子が使えます。

比較演算子	= != < > <= >=
算術演算子	+ - * div mod
論理演算子	and or

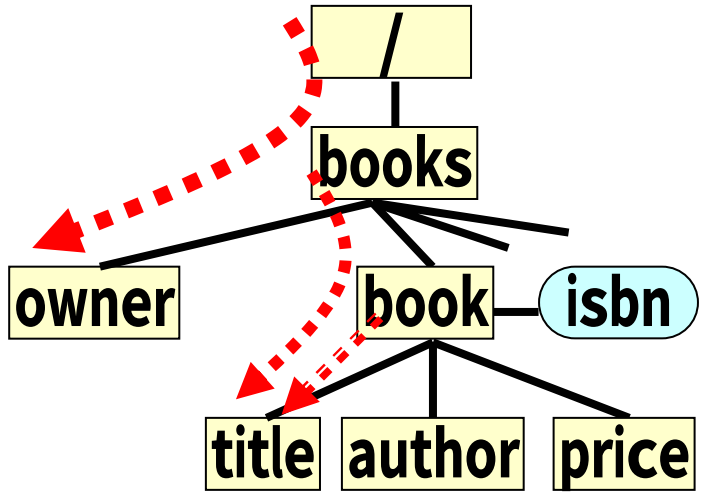
- XSLT内では、タグと混同する“＜”、“＞”は使えないので、表のように記します。

<	<
<=	<=
>	>
>=	>=

(5．4．5) xPath：ロケーションパスと軸

■ロケーションパス

- 基本は、Unixのパス表記に類似しています。



■軸とノードテスト

- よく使われる略記を記します。

略記	意味	表すもの
*	子要素すべて	ノードの集合
.	カレントノード	ノード
..	親要素	ノード
//要素	子孫の要素	ノードの集合

（5．4．6）xPath：関数

■ノードセット関数、文字列関数、ブーリアン関数、数値関数、ＸＳＬＴ関数の４週類があります。

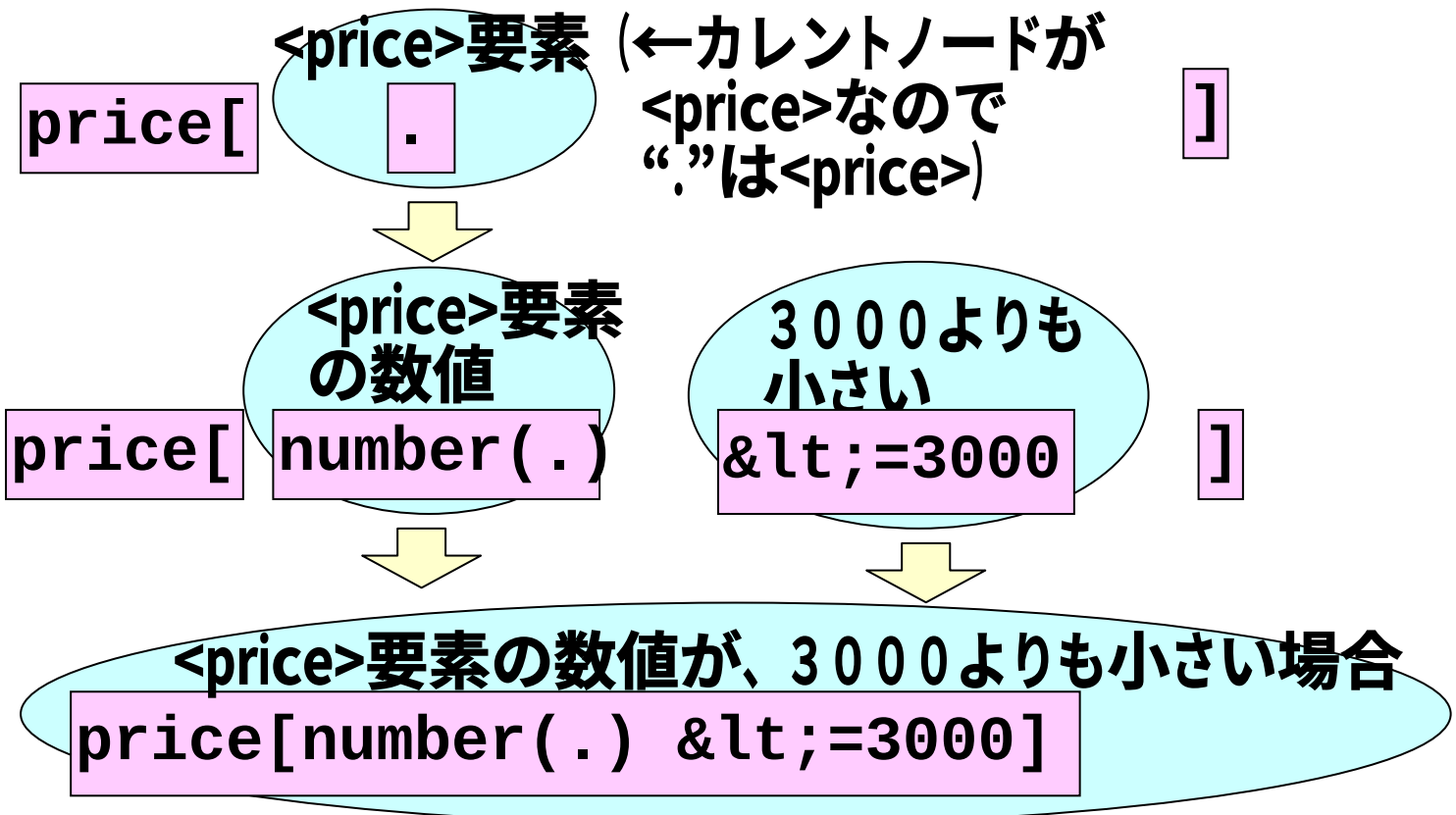
■次の２つの数値関数のみを扱います。

関数	戻り値	説明
number(ノード名)	数値	指定されたノードの値を数値に変換 number(price)：<price>要素の数値
sum(数値)	数値	与えられた数値の合計 sum(price)：<price>要素の合計
count(ノードの集合)	数値	ノードの集合の数 sum(items/*)：<items>要素の子要素の数

(5．4．7) books.xslでの条件記述

```
38: <xsl:when test="price[number(.) &lt;=3000]">
```

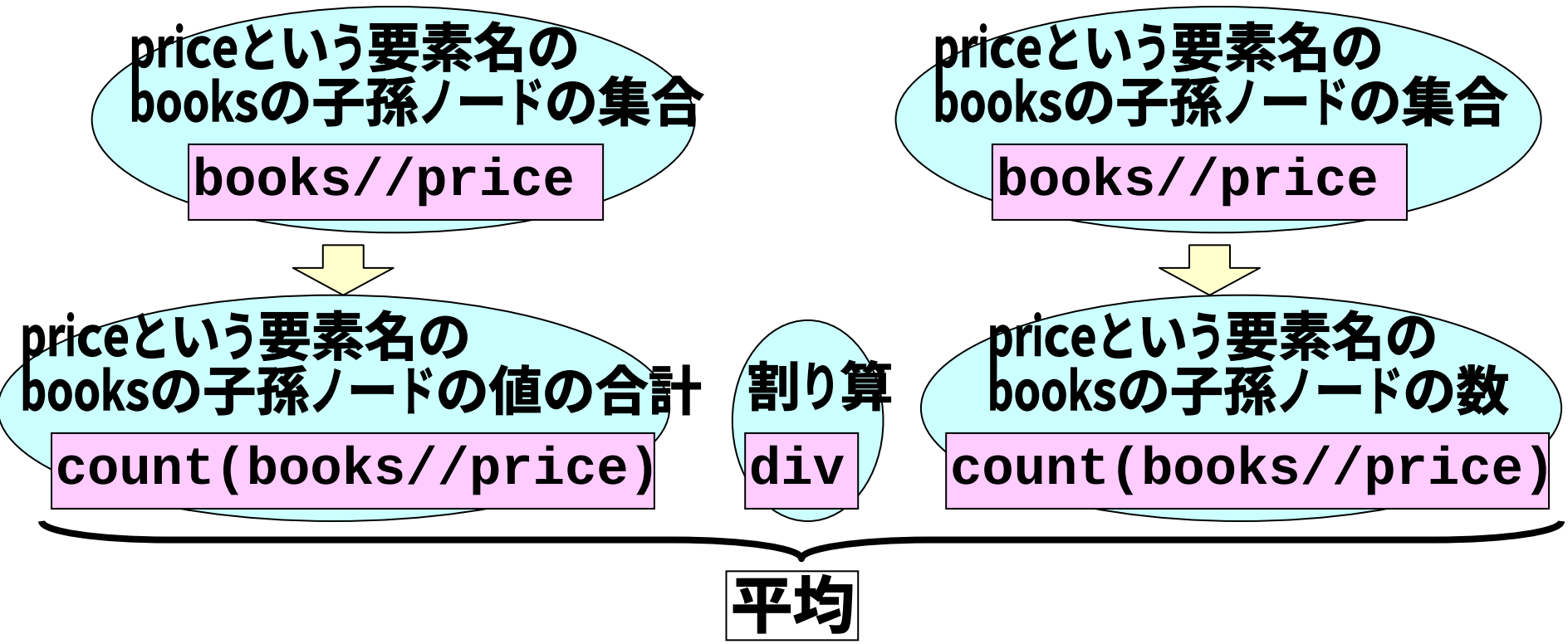
■次の条件は、次のように解釈できます。



(5．5) 平均の表示

■xPathの演算子により、次のように記述されています。

```
20:     <tr><td colspan="2" /><th>平均</th>
21:         <td><xsl:value-of select="sum(books//price)
                  div count(books//price)" /></td></tr>
```



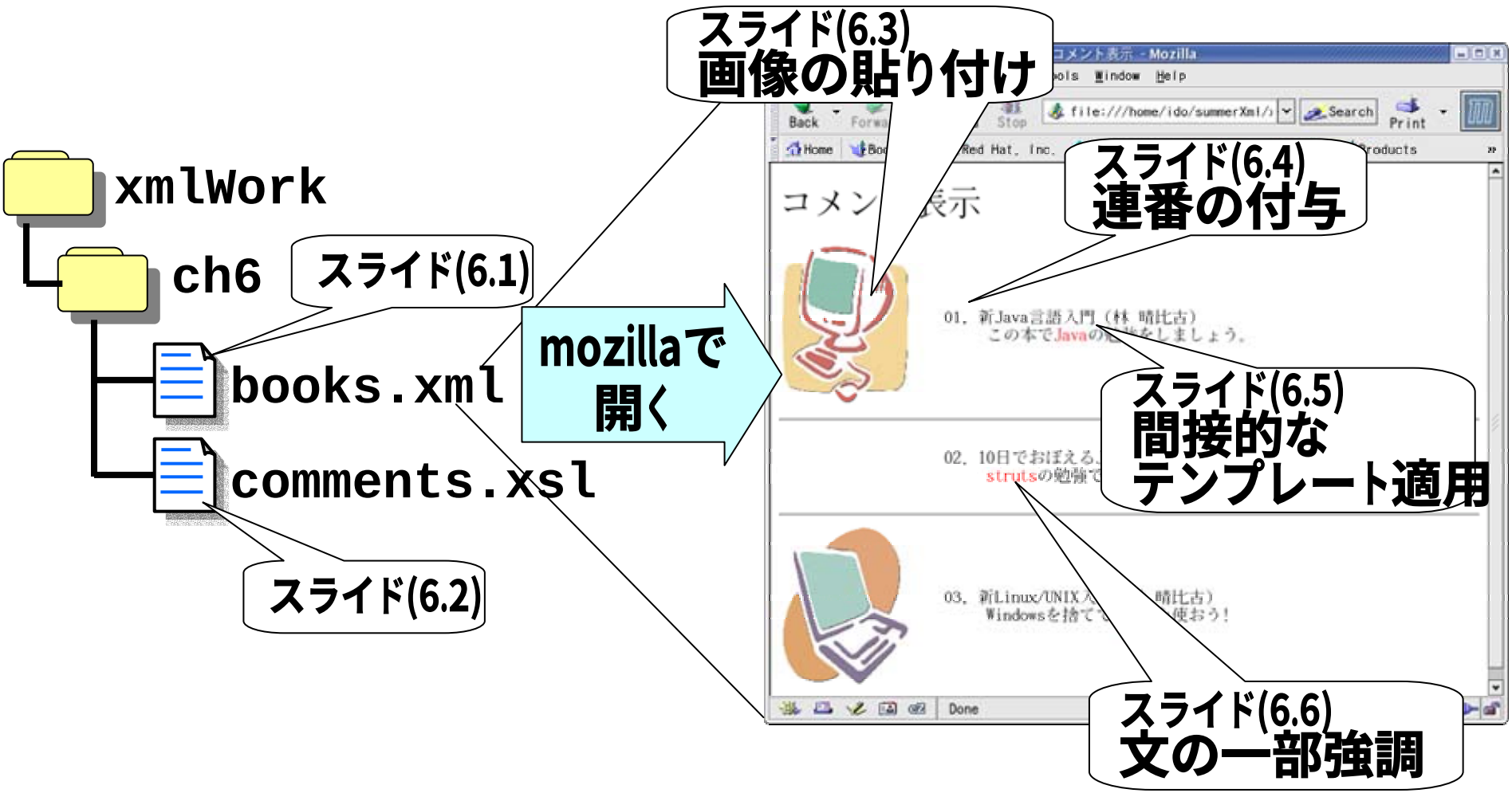
(5．6) 課題

■スライド(4．7)で作成したXML文書を表示させるXSLTスタイルシートをアップデートしてください。

- スライド(5)で出てきた手法をすべて盛り込んでください。

(6) 新しいスタイルシート

■comment.xslという新しいスタイルシートを作成します。



（6．1）books.xmlの変更

■要素<cut>,<comment>,<keywork>,<ref>を追加します。

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<!DOCTYPE books SYSTEM "books.dtd">
<?xml-stylesheet type="text/xsl" href="comments.xsl" ?>
<books category="井戸ゼミ推薦図書">
<owner>井戸伸彦</owner>
<book isbn="ISBN4-7981-0439-6">
        :
<cut>comp01.jpg</cut>
<comment><keyword>struts</keyword>の勉強で用います。
<ref addr="http://www.gifu-keizai.ac.jp/~ido">井戸のサイト</ref>も
参照してください。</comment>
</book>
<book isbn="ISBN4-7973-1318-8">
        :
<comment>Windowsを捨てて<keyword>Linux</keyword>を使おう!</comment>
</book>
<book isbn="ISBN4-7973-1857-0">
        :
<cut>comp02.jpg</cut>
<comment>この本で<keyword>Java</keyword>の勉強をしましょう。</comment>
</book>
</books>
```

(6．2．1) comments.xsl (1／3)

■新しいスタイルシートです。

■ひとつめのスタイルシート("/"にマッチ) の部分です。

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" encoding="iso-2022-jp" />
  <xsl:template match="/">
    <html>
    <head>
    <title>コメント表示</title>
    </head>
   <body>
    <h1>コメント表示</h1>
    <xsl:apply-templates select="books" />
    </body>
    </html>
  </xsl:template>
```

(6．2．2) comments.xsl (2／3)

```
15: <xsl:template match="books">
16:    <xsl:for-each select="book">
17:      <table border="0">
18:      <tr><td width="150">
19:            <xsl:element name="img">
20:              <xsl:attribute name="src">
21:                <xsl:value-of select="cut" />
22:              </xsl:attribute>
23:              <xsl:attribute name="width">120</xsl:attribute>
24:              <xsl:attribute name="height">150</xsl:attribute>
25:            </xsl:element></td>
26:          <td><dl><dt><xsl:number format="01" />.
27:                      <xsl:value-of select="title" />
28:                      (<xsl:value-of select="author" />)</dt>
29:                  <dd><xsl:apply-templates select="comment" />
</dd></dl></td></tr>
30:      </table>
31:      <hr />
32:    </xsl:for-each>
33:  </xsl:template>
```

（6．2．3）comments.xsl（3／3）

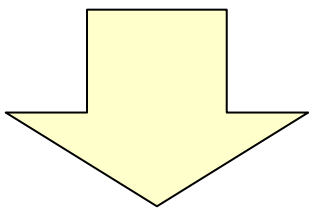
■2つのテンプレートがこの部分にあります。

```
34:  <xsl:template match="keyword">
35:    <font color="red"><xsl:value-of select="." /></font>
36:  </xsl:template>
37:  <xsl:template match="text()">
38:    <xsl:value-of select="." />
39:  </xsl:template>
40:</xsl:stylesheet>
```

(6．3) 画像の貼り付け

■画像の貼り付けは、“(5.3)リンクの設定”の時と同じ理屈で理解出来ます。

```
<img src=“<xsl:value-of select=“cut”>”
     width=“120” height=“150” />
```



上のように書きたいけど、
属性値のなかに<xsl>タグは入れられないから、
下のようにする。

```
19:  <xsl:element name="img">
20:     <xsl:attribute name="src">
21:        <xsl:value-of select="cut" />
22:     </xsl:attribute>
23:     <xsl:attribute name="width">120</xsl:attribute>
24:     <xsl:attribute name="height">150</xsl:attribute>
25:  </xsl:element></td>
```

(6．4) 連番の付与

■ＸＳＬＴでは、繰返しの中で連番を振ることが出来ます。

■ このスタイルシートの<xsl:for-each>要素による繰り返しの中に、<xsl:number>要素があり、この要素が処理される度に連番が振られます。

```
16:  <xsl:for-each select="book">
                          :
26:      <td><dl><dt><xsl:number format="01" />.
                          :
32:  </xsl:for-each>
```

番号の出力形式

■<xsl:number>要素にはさらに複雑な連番を振るために指定する属性がありますが、ここでは立ち入りません。

(6．5．1) 対応がつかないテンプレート

■ここまで見てきたテンプレートは、次のように呼び元の<xsl:apply-template>要素と呼び先のテンプレートとが対応していました。

```
<xsl:apply-templates select="books" />
```

呼び元の要素

```
<xsl:template match="books">
</xsl:template>
```

(テンプレート)

呼び先のテンプレート

■スタイルシートでは、上記の対応が無い呼び元・呼び先があります。

```
29:   <dd><xsl:apply-templates select="comment" />

34:  <xsl:template match="keyword">
36:  </xsl:template>
37:  <xsl:template match="text()">
39:  </xsl:template>
```

呼び先がない

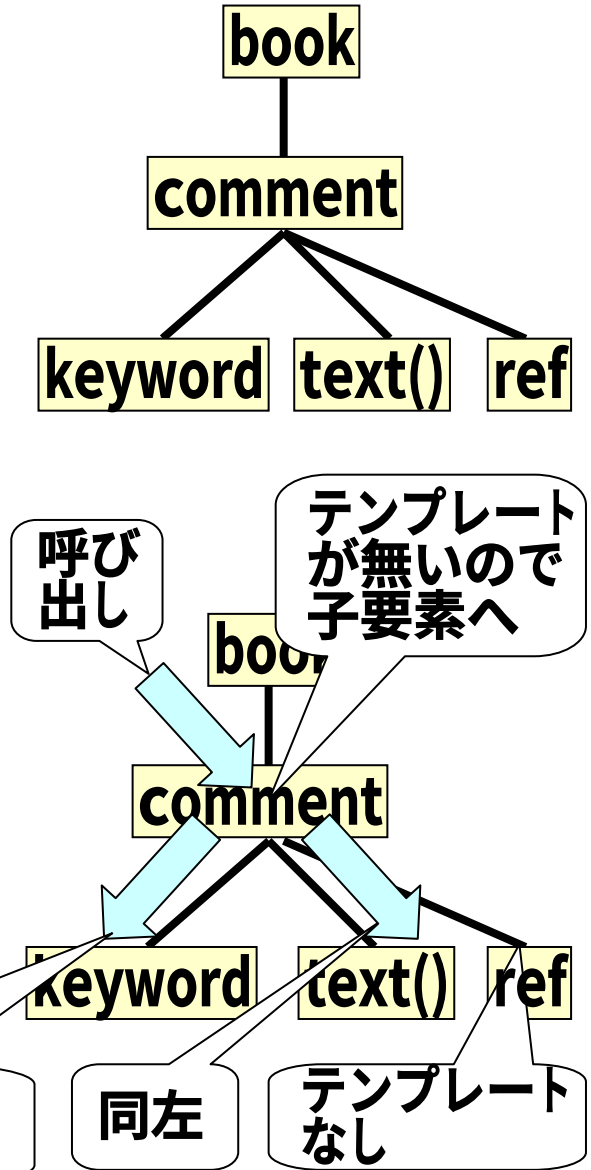
?

?

呼び元がない

(6．5．2) 間接的なテンプレートの適用

- スライド(6.1)のＸＭＬ文書の木構造は、次のようになっています (部分)。
- text() という要素は、コメント要素中のテキスト (文字列) のことを指します ("の勉強で使います"など)。ＸＭＬ文書中のテキストは、このように要素として理解します。
- このような木構造の中で、要素"comment"へのテンプレートが呼び出されると、"comment"にマッチするテンプレートが無いので、木構造中の"comment"の子供のテンプレートにお鉢が回ってきます。
- すなわち、"comment"要素を呼ぶ呼び元の<xsl:apply-template>要素が、間接的にテンプレート"keyword"/"text()"を呼ぶことになるのです。



(6．6) 文の一部強調

- 前スライド(6.5)のような事情で、コメント要素の中身である要素“keyword”と“text()”には、別々のテンプレートが適用されます。
- “text()”要素、すなわちベタのテキストの部分には、自分自身(“.”)を出力するテンプレートが適用されます。

```
34:  <xsl:template match="keyword">
35:    <font color="red"><xsl:value-of select="." />
       </font>
36:  </xsl:template>
```

- “keyword”要素には、赤いフォントで自分自身(“.”)を出力するテンプレートが適用されます。すなわち、キーワードの部分だけ、赤で強調されます。

```
37:  <xsl:template match="text()">
38:    <xsl:value-of select="." />
39:  </xsl:template>
```

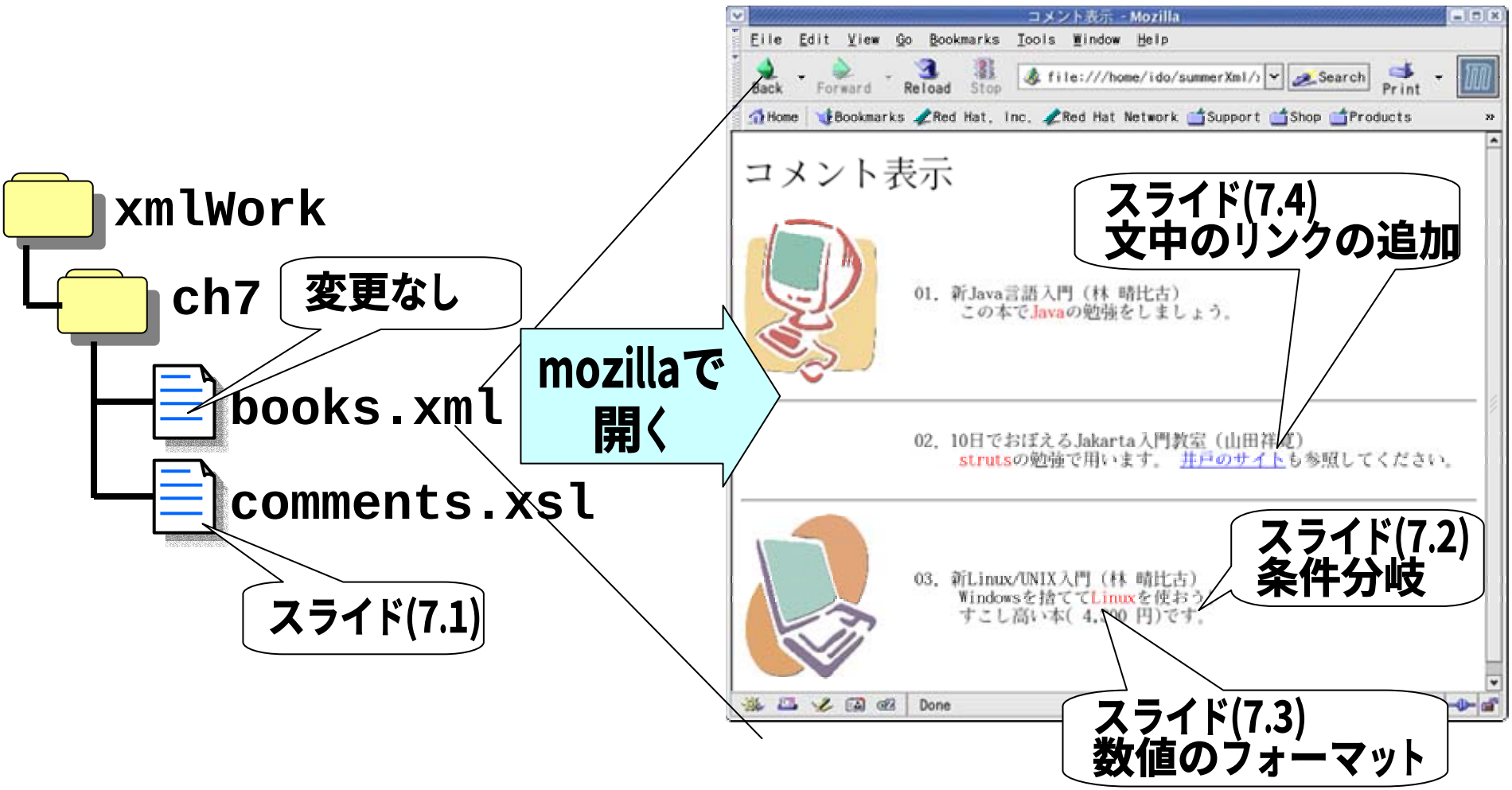
```
<comment>この本で
<keyword>Java</keyword>
の勉強をしましょう。</comment>
```

変換

この本で**Java**の勉強をしましょう。

(7) 条件分岐と文中のリンクの追加

■最後のスタイルシートです。



(7．1．1) comments.xsl (1／3)

■ひとつめのスタイルシート(“/”にマッチ)の部分に、変更はありません。

```
<?xml version="1.0" encoding="iso-2022-jp" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/
Transform" version="1.0">
  <xsl:output method="html" encoding="iso-2022-jp" />
  <xsl:template match="/">
    <html>
    <head>
    <title>コメント表示</title>
    </head>
   <body>
    <h1>コメント表示</h1>
    <xsl:apply-templates select="books" />
    </body>
    </html>
  </xsl:template>
```

(7．1．2) comments.xsl (2／3)

```
    <xsl:template match="books">
       <xsl:for-each select="book">
         <table border="0">
         <tr><td width="150">
               <xsl:element name="img">
                 <xsl:attribute name="src">
                   <xsl:value-of select="cut" />
                 </xsl:attribute>
                 <xsl:attribute name="width">120</xsl:attribute>
                 <xsl:attribute name="height">150</xsl:attribute>
               </xsl:element></td>
             <td><dl><dt><xsl:number format="01" />.
                         <xsl:value-of select="title" />
                         (<xsl:value-of select="author" />)</dt>
               <dd><xsl:apply-templates select="comment" /></dd>
30:            <xsl:if test="price[number(.) &gt;=3000]">
31:                <dd>すこし高い本(
32:      <xsl:value-of select="format-number(price,'#,###')" />
33:                  円)です。</dd>
34:            </xsl:if>
               </dl></td></tr>
         </table>
         <hr />
       </xsl:for-each>
     </xsl:template>
```

（7．1．3）comments.xsl（3／3）

■新たに要素<ref>にマッチするテンプレートが追加されました。

```
    <xsl:template match="keyword">
      <font color="red"><xsl:value-of select="." /></font>
    </xsl:template>
43:  <xsl:template match="ref">
44:    <xsl:element name="a">
45:      <xsl:attribute name="href">
46:        <xsl:value-of select="@addr" />
47:      </xsl:attribute>
48:      <xsl:value-of select="." />
49:    </xsl:element>
50:  </xsl:template>
    <xsl:template match="text()">
      <xsl:value-of select="." />
    </xsl:template>
  </xsl:stylesheet>
```

(7．2) 条件分岐

■スライド(5.4)で見た多岐分岐に似ています。多岐分岐よりも簡単ですね。

```
<xsl:if test="条件">
        条件が成立したときに出力される記述
</xsl:if>
```

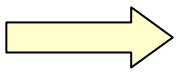
■このテンプレートでは、スライド(5.4.7)で見たものと同じ条件が設定されています。

```
30:  <xsl:if test="price[number(.) &gt;=3000]">
31:     <dd>すこし高い本(
32:     <xsl:value-of select="format-number(price,'#,###')" />
33:     円)です。</dd>
34:  </xsl:if>
```

(7．3) 数値のフォーマット

■32行めでは、数値のフォーマットを行っています。

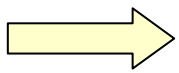
```
32:   <xsl:value-of select="format-number(price,'#,###')" />
```



4,100 と表示される。

■次のように書いた場合との出力の差を見てください。

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



4100 と表示される。

■すなわち、”number-format(～,’#,###’)”で3桁区切りの整形をおこなっています。

(7．4) 文中のリンクの追加

■<ref>要素にもテンプレートが追加されて、リンクを追加する変換が加えられます。

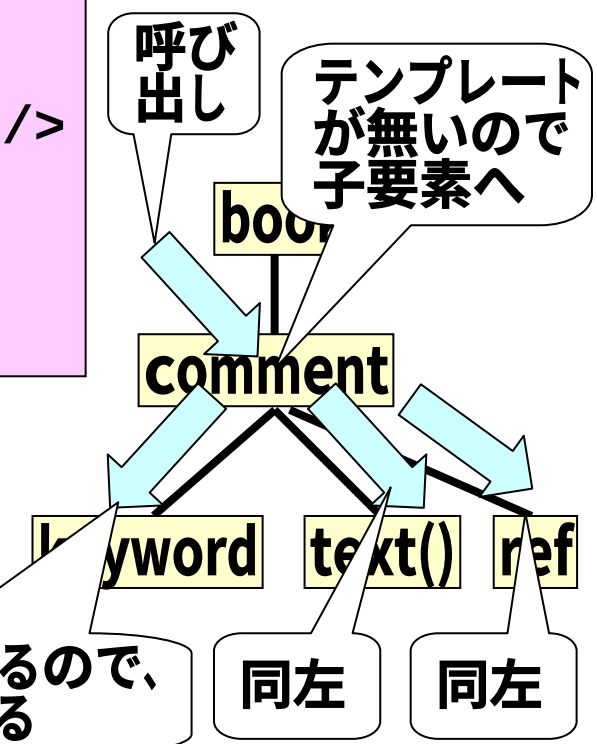
- テンプレートが適用される理屈は、スライド(6.5.2)と同じです。
- リンクの設定のやり方は、スライド(5.3)と同じです。

```
43:  <xsl:template match="ref">
44:    <xsl:element name="a">
45:      <xsl:attribute name="href">
46:        <xsl:value-of select="@addr" />
47:      </xsl:attribute>
48:      <xsl:value-of select="." />
49:    </xsl:element>
50:  </xsl:template>
```

```
<comment>...
<ref addr="http://www.gifu-
keizai.ac.jp/~ido">井戸のサイト</ref>
も参照してください。</comment>
```

変換

```
...
井戸のサイトも参照してください。
```



(7．5) 課題

■スライド (5) までに作成したものとは異なるスタイルシートを作成してください。その際、次のようなものとしてください。

- スライド (6) (7) で見たコメントのような要素を持つようにしてください。
- 上記の要素に、強調やリンクを行う対象となる要素を入れてください。

■ＤＴＤを更新してください。