

# 目次

はじめに .....	17
<b>第 1 章 XSLT の基本</b> .....	<b>23</b>
1.1 XSL = XSLT + XSL FO .....	23
1.2 背景 .....	24
1.2.1 XSLT .....	25
1.2.2 XSL FO .....	26
1.2.3 W3C の仕様 .....	26
1.2.4 XSLT のバージョン .....	27
1.3 XML 文書 .....	29
1.3.1 体裁が整った XML 文書 .....	30
1.3.2 正しい XML 文書 .....	31
1.4 XML はブラウザでどのように表示されるか .....	34
1.5 XSLT 変換 .....	35
1.6 XSLT 変換の実行 .....	38
1.7 スタンドアロンの XSLT プロセッサ .....	39
1.7.1 Java ベースの XSLT プロセッサ .....	41
1.7.2 James Clark の XT .....	43
1.7.3 Saxon .....	44
1.7.4 Oracle の XSLT プロセッサ .....	45
1.7.5 Xalan .....	45
1.8 ブラウザによる XML 文書の変換 .....	47
1.9 Internet Explorer における XSLT と JavaScript の使用 .....	51
1.10 Web サーバーにおける XSLT 変換 .....	52
1.11 XML から XML への変換 .....	55
1.12 XML から XHTML への変換 .....	58
1.13 XSLT の参考資料 .....	61
1.13.1 XSLT の仕様、チュートリアル、例 .....	61

1.13.2	XSLT エディタ	63
1.13.3	XSLT 関連のユーティリティ	66
1.14	XSL FO	67
1.15	XSL FO の参考資料	67
1.16	XML 文書の書式化	68
1.17	XSLT スタイルシート	70
1.18	XSL FO 文書への変換	71
1.19	書式化された文書の作成	74

## 第 2 章 スタイルシートの作成 77

2.1	ツリーとノード	77
2.1.1	空白	80
2.2	XML Infoset モデルと XSLT ツリーモデル	82
2.3	XSLT 要素の操作	83
2.4	<?xml-stylesheet?> 処理命令	84
2.5	<xsl:stylesheet> 要素	86
2.5.1	XSLT 名前空間	87
2.6	最上位要素	88
2.7	<xsl:template> 要素	90
2.8	テンプレート本体	91
2.8.1	XSLT 命令	91
2.8.2	拡張要素	92
2.8.3	リテラル結果要素	92
2.8.4	要素の照合	93
2.9	<xsl:apply-templates> 要素	95
2.10	ノード値へのアクセス	98
2.11	XML Base のサポート	103
2.12	出力方式の選択	104
2.12.1	出力方式 : HTML	106
2.12.2	出力方式 : XML	108
2.12.3	出力方式 : テキスト	109
2.13	簡易スタイルシート	111

2.14	埋め込みスタイルシート .....	113
2.15	<xsl:include> 要素 .....	116
2.16	<xsl:import> 要素 .....	118
2.17	<xsl:apply-imports> 要素 .....	119
2.18	Internet Explorer による XML 文書の変換 .....	121

### 第3章 テンプレートの作成 125

3.1	テンプレート .....	125
3.2	子ノードの処理 .....	127
3.3	ノード値へのアクセス .....	128
3.4	照合パターンの作成 .....	130
3.5	適用するテンプレートの選択 .....	131
3.6	属性値の読み取り .....	134
3.7	<xsl:text> 要素 .....	136
3.8	出力エスケープの無効化 .....	138
3.9	属性値の書き込み .....	141
3.10	属性値テンプレート .....	142
3.11	空白の処理 .....	144
3.12	<xsl:strip-space> 要素と <xsl:preserve-space> 要素 .....	146
3.13	自動インデント .....	148
3.14	テンプレートのデフォルトの規則 .....	151
3.15	内容の削除 .....	154
3.16	テンプレートの競合の解決 .....	156
3.17	<xsl:copy> 要素 .....	159
3.18	<xsl:copy-of> 要素 .....	162
3.19	<xsl:message> 要素 .....	163

### 第4章 パターンの作成 167

4.1	ルートノードの照合 .....	168
4.2	要素の照合 .....	169

4.3	子の照合 .....	169
4.4	要素の子孫の照合 .....	170
4.5	属性の照合 .....	170
4.6	パターンの正式な定義 .....	172
4.7	パターンの軸 .....	175
4.7.1	省略形 .....	178
4.8	ノードテスト .....	178
4.8.1	コメントの照合 .....	179
4.8.2	node() によるノードの照合 .....	180
4.8.3	text() によるテキストノードの照合 .....	182
4.8.4	処理命令の照合 .....	183
4.9	述語 .....	184
4.10	述語の作成 .....	187
4.10.1	ノードセット .....	187
4.10.2	ブール値 .....	189
4.10.3	数値 .....	193
4.10.4	文字列 .....	194
4.10.5	結果ツリーフラグメント .....	197
4.10.6	述語の省略形 .....	197
4.11	ID による照合 .....	198
4.12	キーによる照合 .....	199
4.13	論理和演算子の使用 .....	201
4.14	パターンの例 .....	203

## 第5章 データの選択とソート 213

5.1	<xsl:if> 要素 .....	213
5.2	<xsl:choose> 要素、<xsl:when> 要素、<xsl:otherwise> 要素 .....	218
5.3	<xsl:for-each> 要素 .....	225
5.4	要素のソート .....	230
5.4.1	複数のソート条件 .....	235
5.5	<xsl:number> 要素 .....	235
5.5.1	単一レベルの番号付け .....	236

5.5.2	任意レベルの番号付け	239
5.5.3	複数レベルの番号付け	243
5.6	XSLT の拡張性	244
5.7	拡張関数	246
5.7.1	<xsl:script> 要素	249
5.7.2	function-available() 関数	254
5.7.3	外部オブジェクト	255
5.8	拡張要素	255
5.8.1	element-available() 関数	258
5.9	<xsl:fallback> 要素	258

## 第 6 章 XML 文書の変換と内容の変更 261

6.1	<xsl:output> 要素	262
6.1.1	HTML	264
6.1.2	XML	268
6.1.3	テキスト	272
6.1.4	XHTML	274
6.2	入力に基づいた文書構造の変更	278
6.3	<xsl:element> 要素	279
6.4	<xsl:attribute> 要素	282
6.5	<xsl:comment> 要素	283
6.6	<xsl:processing-instruction> 要素	284
6.7	<xsl:document> 要素 / <xsl:result-document> 要素	286
6.8	<xsl:namespace> 要素	288
6.9	<xsl:attribute-set> 要素	289
6.10	XML 宣言の省略と XML フラグメントの生成	292
6.11	generate-id() による一意な識別子の作成	293
6.12	CDATA セクションの作成	297
6.13	文字エンコーディングの設定	298
6.14	モード	299

## 第7章 XPath 307

---

7.1 XPath の理解 .....	308
7.2 XPath のデータ型 .....	311
7.2.1 ノードセット .....	312
7.2.2 数値 .....	315
7.2.3 文字列 .....	316
7.2.4 ブール値 .....	319
7.3 ロケーションパスの作成 .....	320
7.4 ステップの軸 .....	321
7.5 ステップのノードテスト .....	322
7.6 ステップの述語 .....	323
7.7 XPath の軸の使用 .....	323
7.8 ancestor 軸 .....	324
7.9 ancestor-or-self 軸 .....	324
7.10 descendant 軸 .....	326
7.11 descendant-or-self 軸 .....	327
7.12 following 軸 .....	328
7.13 following-sibling 軸 .....	330
7.14 namespace 軸 .....	331
7.15 parent 軸 .....	333
7.16 preceding 軸 .....	334
7.17 preceding-sibling 軸 .....	335
7.18 self 軸 .....	336
7.19 ロケーションパスの例 .....	337
7.20 XPath 式の省略形 .....	338
7.21 XPath 式のテスト .....	340
7.22 XPath 2.0 .....	341

## 第8章 XSLT と XPath の関数 343

---

8.1 XSLT の関数 .....	345
--------------------	-----

8.2	XPath のノードセット関数	362
8.3	XPath の文字列関数	368
8.4	XPath の数値関数	378
8.5	XPath のブール関数	380
8.6	<xsl:decimal-format> 要素	383
8.7	XSLT 2.0 と XPath 2.0 の新しい関数	387
<b>第 9 章 名前付きテンプレート、パラメータ、変数</b>		<b>389</b>
9.1	<xsl:variable> 要素	390
9.1.1	変数のスコープ	391
9.1.2	変数の使用例	394
9.2	<xsl:call-template> 要素	398
9.3	<xsl:param> 要素と <xsl:with-param> 要素	400
9.3.1	テンプレートの再帰呼び出し	406
9.3.2	テンプレートのデフォルト値	408
9.3.3	コマンドラインによるパラメータ値の指定	409
9.4	<xsl:key> 要素	409
9.5	<xsl:document> 要素 / <xsl:result-document> 要素	415
9.6	<xsl:namespace-alias> 要素	418
<b>第 10 章 XSLT プロセッサの API</b>		<b>421</b>
10.1	Internet Explorer における XSLT と JavaScript の使用	422
10.2	解析エラーの処理	423
10.3	Internet Explorer と動的なスタイル	426
10.4	Internet Explorer と XML データアイランド	432
10.5	XSLT プロセッサと Java	435
10.6	XSLT の API	437
10.7	Xalan と Java	437
10.8	Saxon と Java	440
10.9	Oracle の XSLT プロセッサと Java	443
10.10	XT と Java	447

10.11	XML から SQL データベースへの変換 .....	449
10.12	XSLT と ASP .....	453
10.13	XSLT と JSP .....	456
10.14	XSLT と Java サブレット .....	458
<b>第 11 章 XSL FO 文書の作成 : テキストと表</b>		<b>461</b>
11.1	XSL の書式化 .....	461
11.2	書式化オブジェクト .....	462
11.3	書式化プロパティ .....	467
11.4	XML 文書の書式化 .....	478
11.5	XSLT スタイルシートによる XSL FO 文書への変換 .....	479
11.6	文書のルートの作成 : <fo:root> .....	487
11.7	マスターセットレイアウトの作成 : <fo:layout-master-set> .....	488
11.8	ページマスターの作成 : <fo:simple-page-master> .....	489
11.9	リージョンの作成 .....	490
11.10	ページシーケンスの作成 : <fo:page-sequence> .....	492
11.11	フローの作成 : <fo:flow> .....	493
11.12	ブロックレベルの内容の作成 : <fo:block> .....	494
11.13	表の作成 .....	498
11.14	表の作成 : <fo:table> .....	506
11.15	表の列の作成 : <fo:table-column> .....	507
11.16	表の本体の作成 : <fo:table-body> .....	509
11.17	表の行の作成 : <fo:table-row> .....	510
11.18	表のセルの作成 : <fo:table-cell> .....	512
<b>第 12 章 XSL FO 文書の作成 : リスト、画像、列、配置</b>		<b>515</b>
12.1	リストの作成 .....	515
12.1.1	リストの作成 : <fo:list-block> .....	519
12.1.2	項目の作成 : <fo:list-item> .....	521
12.1.3	項目のラベルの作成 : <fo:list-item-label> .....	522
12.1.4	項目の本体の作成 : <fo:list-item-body> .....	523

12.2	ブロックコンテナによるテキストの配置 : <code>&lt;fo:block-container&gt;</code> .....	524
12.3	インラインレベルの書式化オブジェクト .....	530
12.3.1	インラインリージョンの作成 : <code>&lt;fo:inline&gt;</code> .....	531
12.3.2	文字の処理 : <code>&lt;fo:character&gt;</code> .....	533
12.3.3	ページ番号の作成 : <code>&lt;fo:page-number&gt;</code> .....	535
12.3.4	画像の挿入 : <code>&lt;fo:external-graphic&gt;</code> .....	537
12.3.5	最初の行の書式化 : <code>&lt;fo:initial-property-set&gt;</code> .....	540
12.4	脚注の作成 : <code>&lt;fo:footnote&gt;</code> と <code>&lt;fo:footnote-body&gt;</code> .....	542
12.5	リンクの作成 : <code>&lt;fo:basic-link&gt;</code> .....	545
12.6	列の作成 .....	548
12.7	ページシーケンスとページ番号 .....	550
 <b>付録 A XSLT の DTD</b> .....		<b>561</b>
A.1	XSLT スタイルシートの DTD フラグメント(参考) .....	563
 <b>付録 B XSL FO の書式化プロパティ</b> .....		<b>571</b>
 <b>索引</b> .....		<b>597</b>

# 第 1 章 XSLT の基本

XSLT( Extensible Stylesheet Language Transformations )の世界へようこそ。本書は、一瞬ごとに思いがけない展開を見せる XSLT の大きな世界へ読者を案内する。本書では、XSLT の世界を自分のものにしていく。最近、XSLT は最も興味深く用途の広い技術となりつつあるため、本書では XSLT に関するさまざまな話題を取り上げる。また、XSLT の実際の用途も詳しく見ていく。

XSLT は XML 文書进行处理し、書式化するために使われる( XML については拙著 *Inside XML* を参照)。XML は非常にホットな話題となっているが、今度は XSLT の番である。XML は文書内のデータの構造化を可能にするが、XSLT は XML 文書の内容の操作を可能にする。たとえば、XML で書かれた従業員データのデータベースをソートして、そのデータを HTML 文書に格納したり、細かい書式を設定するなどの処理を行い、XML 文書の内容を操作して他の文書を作成することができる。

また、XML パーサーの API を使うプログラムを作成して XML 文書の内容を操作することもできるが、その場合は自分でコードを書かなければならない。一方、XSLT を使用すれば、プログラミングを行わずに同じ処理を実行できる。XML 文書の内容を操作するために Java、Visual Basic、C++ のコードを独自に書くのではなく、XSLT を使用して必要な処理を指定するだけでよい。残りの作業は XSLT プロセッサが実行してくれる。それが XSLT であり、XML の世界において次の立て役者となるのである。

## 1.1 XSL = XSLT + XSL FO

---

XSLT 自体は、実際には XSL( Extensible Stylesheet Language )という大きな仕様の一部である。XSL は、文書の正確な書式をミリ単位で指定するために使われる。XSL の書式化に関する部分は、XSLT よりはるかに大きな仕様であり、特殊な書式化オブジェクトをベースとする。この仕様は、XSL FO( または XSL:FO、XSLFO )と呼ばれる。書式化オブジェクト( FO )による文書のスタイル化は複雑であるため、本書では XSL FO についても取り上げる。XSLT は本来、XSL の書式化オブジェクトに基づく XML 文書の変換を容易にするために XSL に追加されたものである。

本書は XSLT の解説書であるが、XSLT を使用して XML 文書を XSL FO 文書に変換する方法など、XSL の書式化オブジェクトについても紹介する。XSLT は XSL の書式化オブジェクトの操作を容易にするために導入された仕様であるからだ。まず、本章では、XSLT と XSL の両方について簡単に説明しよう。

## 1.2 背景

XSL 自体は、Tim Berners-Lee が創設した W3C( World Wide Web Consortium、<http://www.w3.org/>)によって開発された。W3C は、XSL などの各種の仕様をリリースしている組織である。XML と XSL を現在の姿にまとめたのが W3C である。

### W3C とスタイル言語

W3C におけるスタイル言語の開発の歴史については、<http://www.w3.org/Style/History/> を参照してほしい。W3C における開発の進行状況や長年にわたるスタイル言語の変遷が興味深い。

W3C は 1980 年代に XML の祖父にあたる SGML( Standard Generalized Markup Language )を開発したが、非常に複雑であることからあまり使用されていない。事実、XML は HTML と同様に SGML の簡易版である。W3C は SGML で使用する DSSSL( Document Style Semantics and Specification Language )と呼ばれるスタイル言語も作成した。XML が SGML から派生したように、XSL は DSSSL に基づいている。W3C は、「画面に文書を描画するために XSL が使用するモデルは、DSSSL と呼ばれる複雑な ISO 標準のスタイル言語に対する長年の蓄積に基づいている」と述べている。

しかし、XSL のオリジナル部分(つまり XSL FO のオリジナル部分)も、広く使用されるほどに容易であるかどうかはまだ実証されていないために、XML 文書から XSL FO 文書への変換の簡易化を目的に XSLT が導入された。結果として、実用化されたのは XSLT である。コードを書くことなく、XML 文書の内容を操作したり、XML 文書を別の XML 文書、HTML、その他のテキストベースの形式に変換するための完全な言語を XSLT が提供しているためだ。XSLT には、W3C さえ驚くほどのサクセスストーリーが隠されているのである。

## 1.2.1 XSLT

---

XSLT を使用すると、XML 文書の内容を直接操作できる。たとえば、野球の最近のシーズンに関する統計情報を含む巨大な XML 文書があるが、投手の情報だけに興味がある場合、Java、Visual Basic、C++ で XML パーサーと連動するプログラムを書いて必要な情報を抽出することが可能である。パーサーは、XML 文書を読み取り、文書内のすべてのデータを 1 つずつコードに渡す特殊なソフトウェアパッケージである。この方法により、投手に関するデータだけを含む `pitchers.xml` という新しい XML 文書を作成することができる。

この方法で目的を達成することはできるが、かなりのプログラミング作業、大量の時間、テストを要する。XSLT は、このような問題を解決するために開発された。XSLT は、実際に XML 文書の操作を行う XSLT プロセッサによって読み取られる。ユーザーは、ある文書を別の文書に変換するために適用する規則を含む XSLT スタイルシートを作成するだけでよい。プログラミングは不要である。そのために、多くのユーザーや経験を積んだプログラマにとって XSLT が魅力的なものになる。野球の例で言うと、必要な操作を指定する XSLT スタイルシートを書くだけで、後は XSLT プロセッサに任せればよいのだ。

XSLT を使うと、XML 文書を、別の XML 文書だけでなく、HTML 文書、RTF 文書、XSL FO 文書などの別の種類の文書に変換することもできる。また、XML ベースの他の言語( MathML、MusicML、VML、XHTML など)に変換することも可能だ。いずれの場合もプログラミングは不要である。

多くの点で、XSLT の機能は SQL( Structured Query Language、有名なデータベースアクセス言語)などのデータベース言語に似ている。XML 文書から目的のデータを抽出するのは、SQL ステートメントをデータベースに適用するのに似ている。XSLT は Web 版の SQL と考えている人さえいる。SQL を知っていれば、XSLT の無限の可能性を理解できるはずだ。たとえば、XSLT スタイルシートを使用し、XML 文書からデータのサブセットを抽出したり、長い文書の目次を作成したり、特定の郵便番号に該当する顧客などの特定の条件に一致するすべての要素を検索するなど、さまざまな処理が可能である。しかも、すべてを 1 ステップで実行できるのだ。

## 1.2.2 XSL FO

---

XSL のもう一方は XSL FO という書式化言語である。本書では XSL FO、つまり XSL の書式化オブジェクトについても取り上げる。XSL FO を使用すると、XML 文書のデータの表現方法を、余白のサイズ、フォント、配置、ヘッダーとフッターのサイズ、ページの幅などにより詳しく指定できる。XML 文書を書式化するときには、考慮すべき項目が非常に多い。したがって、XSL FO の仕様は XSLT より大幅に大きい。

一方、XSL FO は非常に複雑であるため、まだ、XSLT ほどの人気はない。現時点では、XSL FO をサポートしているソフトウェアは少なく、完全な標準に近い形で実装しているものは皆無である。XSLT の最も一般的な用途が XML から HTML への変換であるのと同じように、XSL FO の最も一般的な用途は XML から PDF ( Portable Data Format ) への変換である。PDF は、Adobe Acrobat で使用される形式である。変換の例は、主に第 11 章で紹介する。

## 1.2.3 W3Cの仕様

---

W3C は XML と XSL の仕様をリリースしており、本書はその仕様に従っている。W3C の仕様は標準とは呼ばれない。これは、標準は政府が承認した機関によってのみ作成されるという国際的な合意があるからだ。W3C は、新しい仕様の要件をリリースすることから始める。要件とは目標で、仕様がどのようなものになるかという一種のプレビューを列挙したものであり、その時点では仕様は作成されない。その後、W3C は、誰でもコメントを付けられるワーキングドラフトとして仕様をリリースし、続いてやはり検討の対象である勧告候補をリリースする。最後に勧告として仕様をリリースするのだ。

次に、本書で使用する XSLT 関連の W3C の仕様と入手先を示す。

**XSL 1.0 の勧告** XSL 1.0 を規定した巨大な文書で、2001 年 10 月に発表された。

<http://www.w3.org/TR/xsl/>

**XSLT 1.0 の勧告** XSLT の機能は XML 文書の内容を他の文書に変換することである。この機能が XSLT の人気の理由である。

<http://www.w3.org/TR/xslt>

**XSLT 1.1 のワーキングドラフト** XSLT 1.1 のワーキングドラフトで、勧告にはならない。W3C は XSLT 1.1 の機能を XSLT 2.0 に追加する予定である。

<http://www.w3.org/TR/xslt11/>

**XSLT 2.0 の要件** W3C は、XML Schema のサポートの強化などを目標に、XSLT 2.0 の仕様を作成する。

<http://www.w3.org/TR/xslt20req>

**XSLT 2.0 のワーキングドラフト** 2001 年 12 月に発表された。XSLT 1.1 のワーキングドラフトに代わるもので、XPath 2.0 のサポートなどを盛り込んでいる。

<http://www.w3.org/TR/xslt20/>

**XPath 1.0 の勧告** XPath は、XML 文書内の特定のセクションと要素を操作するために、位置の指定と参照に使用される。

<http://www.w3.org/TR/xpath>

**XPath 2.0 の要件** XPath 2.0 では XSLT 2.0 に対するサポートが強化される。

<http://www.w3.org/TR/xpath20req>

**XPath 2.0 のワーキングドラフト** 2001 年 12 月に発表された。

<http://www.w3.org/TR/xpath20/>

## 1.2.4 XSLTのバージョン

---

XSLT の仕様の策定作業は、全体として XSL の仕様よりかなり活発である。XSLT 1.0 の勧告は 1999 年 11 月 16 日に策定され、今日の XSLT のバックボーンを形成している。

次に登場したのが、XSLT 1.1 のワーキングドラフトである。本来はこれが新しい勧告になる予定であったが、W3C では XSLT 2.0 の作業が開始された。その後、W3C は XSLT 1.1 の**キャンセル**を決定した。これは、XSLT 1.1 のワーキングドラフトはそれ以上先へ進まず、勧告にはならないことを意味する。つまり、XSLT には正式なバージョン 1.1 は存在しないことになる。

ただし、W3C は XSLT 1.1 のワーキングドラフトで行われた作業の多くを XSLT 2.0 に統合し、2001 年 12 月には XSLT 2.0 のワーキングドラフトを発表した。そのため、本書では XSLT 1.1 または 2.0 のワーキングドラフトについても取り上げる。XSLT 1.1 または 2.0 のワーキングドラフトで導入された新機能については、その旨を明記する。

XSLT 1.1 における XSLT 1.0 からの変更点を次に示す。これはあくまで参考として示すもので、現時点では実際にはあまり意味を持たない。

XSLT 1.0 でサポートされていた結果ツリーフラグメントのデータ型を使わない。

名前空間の修正が自動的に適用されるため、出力方式の指定により自由に名前空間のノードを追加することはできない。

XML Base をサポートする。

変換により複数の結果ツリーを生成できる。

`<xsl:apply-imports>`要素にパラメータを指定できる。

Java または JavaScript による拡張関数の定義が可能である。

拡張関数は XPath のデータ型に対応しない**外部オブジェクト**を返すことができる。

さらに、XSLT 2.0 では、XPath 2.0 のサポート、ノードのグループ化、XHTML の出力、ソート、複数の結果ツリーの出力などの機能が追加、改訂されている。

本書では、XSLT 1.0 の勧告だけでなく、XSLT 1.1 のワーキングドラフトも取り上げる。また、W3C は先を急ぎ、XSLT 2.0 の要件およびワーキングドラフトをリリースしているため、XSLT 2.0 に含まれる予定の機能についても取り上げる。次に、XSLT 2.0 の目標の概要を示す。

XML Schema のデータ型の操作を容易にする。

文字列の操作を容易にする。

関連する XML の標準をサポートする。

XSLT の使いやすさを向上させる。

XSLT の相互運用性を向上させる。

国際化のサポートを向上させる。

XSLT 1.0 および 1.1 との下位互換性を維持する。

プロセッサの効率化をサポートする。

XSLT 2.0 の勧告がリリースされるのはまだ先のことであるが、関連するトピックの説明では現時点でわかっている限り XSLT 2.0 の内容をカバーする。たとえば、W3C では、XML ベースの XHTML を HTML の後継としている。XSLT 1.0 の勧告と XSLT 1.1 のワーキングドラフトでは、XML から XHTML への変換をサポートしていないので、この変換は独自に行う必要がある。ただし、XSLT 2.0 では XHTML への変換がサポートされる予定であり、XHTML について説明するときにはその事実に言及する。

以上で概要の説明を終わる。では、実際の内容に移ろう。XSL は XML 文書の操作を目的としているので、まず、XML 文書の構造を復習しよう。操作対象は XML 文書だが、XSLT スタイルシート自体も実際には XML 文書である。XSLT スタイルシートを書くときには、この事実を念頭におく必要がある。本書では、XML と HTML についてある程度の知識を持っていることを前提としている(以前にも述べたように、*Inside XML* を参考にしてほしい)。

## 1.3 XML 文書

今後は XML 文書の動作を理解していることが重要になるため、ここでは XML 文書について復習しよう。例として、次の XML 文書を使用する。

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
  <GREETING>
    Hello From XML
  </GREETING>
  <MESSAGE>
    Welcome to the wild and woolly world of XML.
  </MESSAGE>
</DOCUMENT>
```

この文書の機能を説明しよう。まず、最初に**処理命令**がある。XML 処理命令は`<?>`で始まり、`?>`で終わる。これは、現時点で定義されている唯一のバージョンである XML 1.0 と UTF-8 の文字エンコーディングを使用することを指示している。つまり、Unicode の 8 ビットのエンコーディング方式を使用する。

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
  <GREETING>
    Hello From XML
  </GREETING>
  <MESSAGE>
    Welcome to the wild and woolly world of XML.
  </MESSAGE>
</DOCUMENT>
```

次に、`<DOCUMENT>`という名前の新しい**タグ**を作成する。タグには、`DOCUMENT`だけでなく、好きな名前を付けることができる。ただし、名前は英字またはアンダースコア(`_`)で始まり、英字、数字、アンダースコア、ピリオド(`.`)、ハイフン(`-`)で構成され、空白を含んではならない。XML では、タグは常に`<`で始まり、`>`で終わる。

XML 文書は XML **要素**で構成される。XML 要素は、`<DOCUMENT>`などの開始タグに始まり、テキストやその他の要素などの内容が続き、`</DOCUMENT>`のように`</>`で示される終了タグで終わる。処理命令を除き、文書全体を 1 つの要素で囲むが、それを**ルート要素**と呼ぶ。ここでは、ルート要素は`<DOCUMENT>`要素になる。

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

ここで、この XML 文書にテキスト(“Hello From XML”)を含む新しい<GREETING>要素を追加しよう。

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
  <GREETING>
    Hello From XML
  </GREETING>
  :
</DOCUMENT>
```

また、テキストを含む新しい要素として<MESSAGE>を追加することもできる。

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
  <GREETING>
    Hello From XML
  </GREETING>
  <MESSAGE>
    Welcome to the wild and woolly world of XML.
  </MESSAGE>
</DOCUMENT>
```

これで、<DOCUMENT>のルート要素には<GREETING>と<MESSAGE>という2つの要素が含まれることになる。また、<GREETING>と<MESSAGE>のそれぞれがテキストを保持する。このようにして新しい XML 文書を作成する。

しかし、これで終わりではない。XML 文書には、**体裁が整った XML 文書**と**正当な XML 文書**の2種類がある。

### 1.3.1 体裁が整ったXML文書

体裁が整った XML 文書は、XML 1.0 の勧告(<http://www.w3.org/TR/REC-xml>)で W3C によって規定された構文規則に従う必要がある。わかりやすく言うと、「体裁が整った」とは、文書が1つ以上の要素を含み、1つの要素(ルート要素)が他のすべての要素を含まなければならないことを意味する。また、それぞれの要素に含まれる要素は正しいネスト構造をとらなければ

ばならない。たとえば、次の文書は、終了タグである</GREETING>が次の要素の開始タグである<MESSAGE>の後にあるため、体裁が整った文書ではない。

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
  <GREETING>
    Hello From XML
  <MESSAGE>
</GREETING>
    Welcome to the wild and woolly world of XML.
  </MESSAGE>
</DOCUMENT>
```

### 1.3.2 正当なXML文書

ほとんどの XML ブラウザは体裁が整った文書であるかどうかをチェックする。また、一部の XML ブラウザは正当な文書であるかどうかにもチェックする。正当な XML 文書は、関連付けられている DTD (Document Type Declaration : 文書型宣言) または XML Schema に準拠していなければならない。つまり、DTD またはスキーマによって文書自体の内部整合性の規則が規定され、文書がその規則に従っていることがブラウザによって確認された場合、正当な XML 文書であるとみなされる。

XML Schema は広く受け入れられており、XSLT 2.0 では、XML Schema のサポートが強化される(事実、XML Schema のサポートは XSLT 2.0 の策定の大きな動機となっている)。しかし、正当な XML 文書であることを示すために最も一般的に使用されているのは、やはり DTD である。DTD は、個別のファイルに格納したり、XML 文書そのものに<!DOCTYPE>要素として格納することができる。次に、XML 文書に<!DOCTYPE>要素を追加する例を示す。

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="first.css"?>
<!DOCTYPE DOCUMENT [
  <!ELEMENT DOCUMENT (GREETING, MESSAGE)>
  <!ELEMENT GREETING (#PCDATA)>
  <!ELEMENT MESSAGE (#PCDATA)>
]>
<DOCUMENT>
  <GREETING>
    Hello From XML
  </GREETING>
```

```
<MESSAGE>
  Welcome to the wild and woolly world of XML.
</MESSAGE>
</DOCUMENT>
```

本書では DTD については説明しない(DTD の詳細については *Inside XML* を参照)が、ここで指定した DTD は、<DOCUMENT>要素内で<GREETING>および<MESSAGE>の各要素を指定できること、<DOCUMENT>要素がルート要素であること、<GREETING>および<MESSAGE>の各要素がテキストを保持できることを示している。

XML 文書では、あらゆる種類の階層を設定することができる。つまり、ある要素の中に別の要素を指定し、任意の深さまでネストさせることが可能である。また、要素に属性を持たせることができる。たとえば、<CIRCLE COLOR="blue">は、COLOR 属性が値として blue を持つことを意味する。このような属性を使用し、要素に追加データを持たせることができる。さらに、特定の要素を詳しく説明するコメントを XML 文書に含めることも可能だ。コメントは、<!--と-->で指定する。

次に、これらの機能の実例となる XML 文書を示す。planets.xml という文書は、水星 Mercury)、金星(Venus)、地球(Earth)に関して質量、1日の長さ、密度、太陽からの距離などのデータを格納する。この文書は本書を通じて使用する XML のサンプルで、XML の機能の大半が凝縮されている。

リスト 1-1 planets.xml

```
<?xml version="1.0"?>
<PLANETS>

  <PLANET>
    <NAME>Mercury</NAME>
    <MASS UNITS="(Earth = 1)">.0553</MASS>
    <DAY UNITS="days">58.65</DAY>
    <RADIUS UNITS="miles">1516</RADIUS>
    <DENSITY UNITS="(Earth = 1)">.983</DENSITY>
    <DISTANCE UNITS="million miles">43.4</DISTANCE><!--At perihelion-->
  </PLANET>

  <PLANET>
    <NAME>Venus</NAME>
    <MASS UNITS="(Earth = 1)">.815</MASS>
    <DAY UNITS="days">116.75</DAY>
    <RADIUS UNITS="miles">3716</RADIUS>
```

```

    <DENSITY UNITS="(Earth = 1)">.943</DENSITY>
    <DISTANCE UNITS="million miles">66.8</DISTANCE><!--At perihelion-->
</PLANET>

<PLANET>
  <NAME>Earth</NAME>
  <MASS UNITS="(Earth = 1)">1</MASS>
  <DAY UNITS="days">1</DAY>
  <RADIUS UNITS="miles">2107</RADIUS>
  <DENSITY UNITS="(Earth = 1)">1</DENSITY>
  <DISTANCE UNITS="million miles">128.4</DISTANCE><!--At perihelion-->
</PLANET>

</PLANETS>

```

また、次に示す XML の定義も理解してほしい。

**CDATA** 単純な文字データ(マークアップを含まないテキスト)を示す。

**ID** 適切な XML 名を示す。この名前は一意でなければならない(つまり ID 型の他の属性と共有されない)。

**IDREF** 何らかの要素(通常、現在の要素に関連する別の要素)の ID 属性の値を保持する。

**IDREFS** 複数の要素の ID(空白によって区切られる)を示す。

**名前前文字** 英字、数字、ピリオド、ハイフン、アンダースコア、コロンを使用できる。

**NAME** XML 名を示す。先頭は必ず英字、アンダースコア、コロンのいずれかになる。

**NAMES** 空白で区切られた名前前のリストを示す。

**NMTOKEN** 1 個以上の英字、数字、ハイフン、アンダースコア、コロン、ピリオドから構成されるトークンを示す。

**NMTOKENS** 空白で区切られた複数の適切な XML 名を示す。

**NOTATION** 記法の名前を示す(DTD で宣言されていなければならない)。

**PCDATA** 解析済みの文字データを示す。PCDATA はマークアップを含まず、実体の参照は PCDATA 内で展開済みである。

体裁が整った文書と正当な文書の違いを含め、XML 文書の概要を述べた。XML 文書の知識が不十分であれば、*Inside XML* などの書籍を参照してほしい。また、次のように、Web にも XML 関連のリソースが数多く存在する。

<http://www.w3.org/XML/> W3CのXMLのWebサイトで、すべての出発点となる。

<http://www.w3.org/XML/1999/XML-in-10-points.html> *XML In 10 Points*(実際には7点のみ)というドキュメントで、XMLの概要を説明している。

<http://www.w3.org/TR/REC-xml> W3Cの正式なXML 1.0の勧告(2000年10月発表)で、現時点での唯一のバージョンである。あまり読みやすいとは言えない。

<http://www.w3.org/TR/xml-styleSheet/> XML文書にスタイルシートを関連付ける方法を定めている。

<http://www.w3.org/TR/REC-xml-names/> XMLにおける名前空間を規定している。

<http://www.w3.org/XML/Activity> XMLに関する現在の活動状況を説明している。

<http://www.w3.org/TR/xmlschema-0/>、<http://www.w3.org/TR/xmlschema-1/>、

<http://www.w3.org/TR/xmlschema-2/> DTDに代わるXML Schemaを規定する。

<http://www.w3.org/TR/xlink/> XLinkの仕様である。

<http://www.w3.org/TR/xptr> XPointerの仕様である。

<http://www.w3.org/TR/xhtml1/> XHTML 1.0の仕様である。

<http://www.w3.org/TR/xhtml11/> XHTML 1.1の仕様である。

<http://www.w3.org/DOM/> W3CのDOM(Document Object Model)のWebサイトである。

XML文書の作成方法は理解した。では、XML文書はどのように参照するのだろうか。

## 1.4 XMLはブラウザでどのように表示されるか

---

Microsoft Internet Explorer 5.0以降などのブラウザを使用すれば、未加工のXML文書を直接表示できる。たとえば、作成したXML文書を `planets.xml` という名前で保存し、その文書を Internet Explorer で開くと図 1-1 のように表示される。

図 1-1 には完全なXML文書が表示されている。特別な書式化は何も行われていない。Internet Explorer では、XML文書はプリンタで印刷したとおりのイメージで表示される(事実、図 1-1 では、Internet Explorer のXML文書用のデフォルトのスタイルシートが使用されている。このスタイルシートは、XMLをInternet Explorerで認識できるDynamic HTMLに変換する)。しかし、データの表示方法を変えたい場合はどうだろう。たとえば、HTML文書内の `planets.xml` をHTMLの表として表示したいときにはどうすればよいのだろうか。

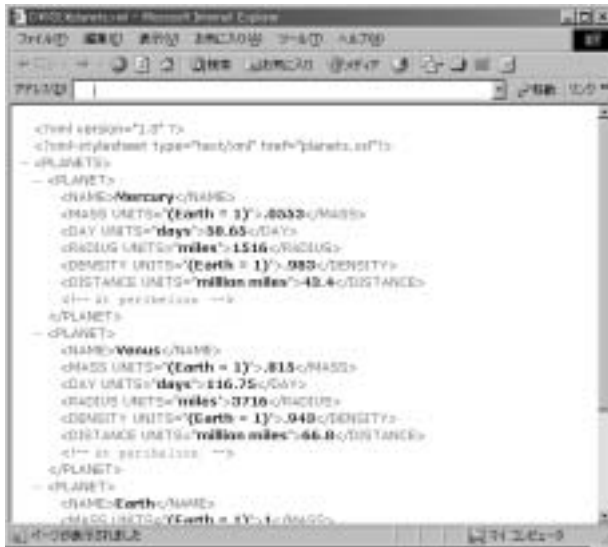


図 1-1 Internet Explorer で表示した XML 文書

ここで XSLT 変換処理が登場する。本章では、まず XSLT 変換の概要を見ていく。後半では、XSL のもう一方の側面である XSL FO の概要を説明する。

## 1.5 XSLT変換

XSLT は、XML 文書内のデータを操作するための強力な言語である。たとえば、XSLT スタイルシートを使用して planets.xml 内のデータを HTML の表に変換することができる。スタイルシートは、XML 文書を変換するためにユーザーが設定した規則を含む。本書の大部分では、スタイルシートの書き方とスタイルシートの動作を理解することに焦点を当てる。planets.xml 内のデータを HTML の表に変換する XSLT スタイルシート(planets.xsl)を次に示す(詳細については第 2 章を参照)。

リスト 1-2 planets.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

```
<xsl:template match="/PLANETS">
  <HTML>
    <HEAD>
      <TITLE>
        The Planets Table
      </TITLE>
    </HEAD>
    <BODY>
      <H1>
        The Planets Table
      </H1>
      <TABLE BORDER="2">
        <TR>
          <TD>Name</TD>
          <TD>Mass</TD>
          <TD>Radius</TD>
          <TD>Day</TD>
        </TR>
        <xsl:apply-templates/>
      </TABLE>
    </BODY>
  </HTML>
</xsl:template>

<xsl:template match="PLANET">
  <TR>
    <TD><xsl:value-of select="NAME"/></TD>
    <TD><xsl:apply-templates select="MASS"/></TD>
    <TD><xsl:apply-templates select="RADIUS"/></TD>
    <TD><xsl:apply-templates select="DAY"/></TD>
  </TR>
</xsl:template>

<xsl:template match="MASS">
  <xsl:value-of select="."/>
  <xsl:text> </xsl:text>
  <xsl:value-of select="@UNITS"/>
</xsl:template>

<xsl:template match="RADIUS">
  <xsl:value-of select="."/>
  <xsl:text> </xsl:text>
  <xsl:value-of select="@UNITS"/>
</xsl:template>
```

```

    <xsl:template match="DAY">
      <xsl:value-of select="."/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="@UNITS"/>
    </xsl:template>
  </xsl:stylesheet>

```

明らかに、この XSLT スタイルシートは XML 文書のように見える。それもそのはず、これはまさしく XML 文書である。XSLT スタイルシートは XML 文書であり、体裁が整った XML 文書でなければならない。本書を通じて、さまざまな方法で XSLT 変換を行う際は、XML 文書の `planets.xml` (リスト 1-1) とスタイルシートの `planets.xsl` (リスト 1-2) を使用する。

このスタイルシートを XML 文書の `planets.xml` にどのように関連付けるのだろうか。第 2 章で説明するが、1 つの方法として `<?xml-stylesheet?>` という XML 処理命令を使用できる。この処理命令は 2 つの属性を使用する。最初の属性は `type` である。この属性を `text/xml` に設定することによって、XSLT スタイルシートを使用することを指定する(別の種類のスタイルシートとして HTML で通常使用される CSS がある。CSS を使用する場合、この属性を `text/css` に設定する)。2 番目の属性は、`href` である。この属性にはスタイルシートの URI (Uniform Resource Identifier) を設定する (XML では URL ではなく URI を使用する)。

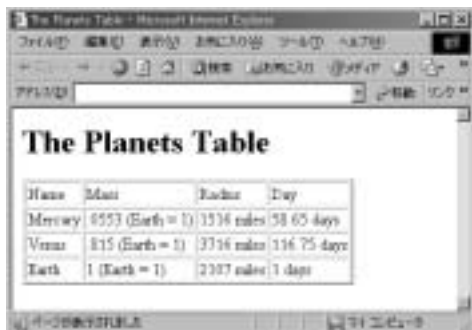
```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="planets.xsl"?>
<PLANETS>

  <PLANET>
    <NAME>Mercury</NAME>
    <MASS UNITS="(Earth = 1)">.0553</MASS>
    <DAY UNITS="days">58.65</DAY>
    <RADIUS UNITS="miles">1516</RADIUS>
    <DENSITY UNITS="(Earth = 1)">.983</DENSITY>
    <DISTANCE UNITS="million miles">43.4</DISTANCE><!--At perihelion-->
  </PLANET>
  :
  :

```

続いて、XSLT **プロセッサ**を使用して `planets.xsl` を `planets.xml` に適用し、新しい文書として `planets.html` を生成できる。XSLT プロセッサによって生成された新しい HTML 文書の `planets.html` を図 1-2 に示す。

A screenshot of a Microsoft Internet Explorer browser window. The title bar reads 'The Planets Table - Microsoft Internet Explorer'. The address bar is empty. The main content area displays a table titled 'The Planets Table'. The table has four columns: 'Name', 'Mass', 'Radius', and 'Day'. The rows are for Mercury, Venus, and Earth. The data for Mercury is: Mass 0.0553 (Earth = 1), Radius 3334 miles (58.65 days). The data for Venus is: Mass 0.815 (Earth = 1), Radius 3736 miles (116.75 days). The data for Earth is: Mass 1 (Earth = 1), Radius 2307 miles (1 day).

Name	Mass	Radius	Day
Mercury	0.0553 (Earth = 1)	3334 miles	58.65 days
Venus	0.815 (Earth = 1)	3736 miles	116.75 days
Earth	1 (Earth = 1)	2307 miles	1 day

図 1-2 XSLT プロセッサによって作成された HTML 文書

図 1-2 を見るとわかるように、XSLT プロセッサは planets.xml のデータを読み取って planets.xsl の規則を適用し、planets.html に HTML の表を作成している。

実際には何が行われたのだろうか。XML 文書の planets.xml と XSLT スタイルシートの planets.xsl はすでに見たが、この 2 つのファイルがどのように結び付けられて planets.html が生成されたのだろうか。

## 1.6 XSLT変換の実行

planets.xml から planets.html へのように、XSLT 変換を行うには XSLT プロセッサを使用する。XSLT を使用して XML 文書を変換するには 3 つの方法がある。

**XSLT プロセッサというスタンドアロンプログラムによる変換** XSLT 変換を実行するプログラムはいくつかあり、通常は Java をベースとする。本章では、さまざまな XSLT プロセッサを紹介する。

**クライアントによる変換** ブラウザなどのクライアントプログラムは、`<?xml-stylesheet?>` という処理命令で指定したスタイルシートを読み込んで変換を行うことができる。Internet Explorer は、この方法による変換をある程度処理できる。

**サーバーによる変換** Java サーブレットなどのサーバープログラムは、スタイルシートを使用して文書を自動的に変換し、それをクライアントに送信することができる。

本書では、この 3 つの方法をすべて説明する。本章では、各方法の概要を説明する。