

XSLT スタイルシートを利用した 多画像ホームページについて

末 木 俊 之

Multi-Picture Homepage Using XSLT Style Sheet

Toshiyuki SUEKI

1. 多画像表示ホームページについて

1-1. 静的ホームページから動的ホームページ への展開

昨年度の研究紀要では、表計算ソフトにおける多系列表および、パラメータ値を変更することにより変化する画像からマクロにより多数の画像が生み出されるケースを扱った。生成された画像の画像名称、パラメータ値などの情報を記述したXMLテキストファイルを作成し、XSLT スタイルシートを参照させてブラウザソフト Internet Explorer に組み込まれた XSLT プロセッサ (MSXML3) により自動的に HTML 形式に変換して表示させ、多画像表示のホームページとして表示させることを試みた。昨年度のケースでは、約2000個の画像を扱っている。それら全ての画像を1つのホームページで表示させるのは、データ容量的に巨大過ぎ、使い勝手も悪い。そこで20個のXMLファイルを用意し、それぞれ表示すべき画像を絞り込む条件部分のみが異なっているXSLTファイルを参照させてホームページとして表示させた。そして、その20個のXMLファイルと呼び出せる目次を用意した。20個のXMLファイルと20個のXSLTファイルを作成する必要があった。基になる1つのXMLファイルと1つの

XSLT ファイルをコピーして一部分のみ変更すれば他の19個のXMLファイルおよびXSLTファイルが作成できる。昨年度の段階ではここまでで留めておいた。しかし、1つのホームページに100個の画像が表示されるのでは、まだ画像が多過ぎる。しかし、このXMLファイルと静的なXSLTファイルの組み合わせによりホームページとして画像を表示させる方法では、これ以上表示させる画像数を減らしたホームページを作成するのは困難であるし、使い勝手も悪くなる。

今年度はこの続きとして昨年度のような静的な方法ではなく、1つのホームページに表示する画像数、表示する画像の指定などが柔軟に変更できるような動的なページ作成を目指してみることにした。その場合、JavaScript スクリプトなどの複雑過ぎるスクリプト、ASP (Active Server Pages) などのWWWサーバ側での複雑な処理をなるべく必要としないシンプルな方法で作成したい。また、XMLテキストファイルと、XSLT スタイルシートの組み合わせによる実現方法に極力こだわりたい。

XML、XSLT などの技術も世間一般に普及し、学生に対しても今後教材として取り上げる機会も出てくることも考えられるが、なかなか

取り付き難い技術である。今回の取り組みが、日々の一般的なパソコン利用の場面で、そんなに難解でも無く、XSLT などの技術が有効活用できる 1 つのケースを生み出せば良いと思われる。

いずれにせよ、表計算ソフトの日常的なビジネス利用から多系列、多画像を有する情報が生み出されるケースはよくある。その情報を高度な情報処理技術を持った人間に依頼せずに割合簡単に最終処理できる手段について考察する意味でも、今回の取り組みの意義を感じる。XML と XSLT の技術には可能性が感じられるのではあるが、このケースでどれほど簡単な利用が可能であるのか実際に使ってみなくては中々理解し難いものがある。可能性は感じるが、取り付き難い技術のようである。

1-2. 多画像表示ホームページに求められる機能について

まず多画像を表示するホームページに求められる機能を列挙するために XML ファイルデータをホームページ上で扱う最も簡単な方法と思われる Data Island を考察してみる。これは Internet Explorer だけの機能であり、ホームページ内に、XML ファイルデータを表示することが簡単に実現できる機能である。昨年度は、『VB と ASP でつくる XML』(2001年株式会社ピアソン・エデュケーション発行)を参照して、グラフ画像データ XML ファイルを格納したホームページを作成してみた。昨年度の作成時点では、Data Island ではグラフ画像の表示が不可能であると認識していたが、考え違いであった。『XML の徹底解説から ASP 連携 Web サイト構築まで XML 超入門』(2001年有限会社セレンディップ発行)でも Data Island を使用して画像を表示しているケースが紹介されてい

た。昨年度の研究紀要で掲載したものに、画像表示をさせる部分を修正したものを(図 1.)に掲載する。ホームページのソーステキストについては修正が含まれた部分のみ(表 1.)に掲載した。(表 1.)の 11 行目が変更した部分である。ホームページ内にテーブル構造を構築し、2 列目に IMG タグを入れ、SRC 属性に XML ファイル内の<グラフ名>タグデータをバインドさせて画像表示が実現できている。

この Data Island の有する機能について考察すると、(表 2.)に記述した 5 点が指摘できる。

100 件程度までの少数のデータ(画像)なら、(表 2.)の 5 つの機能があれば対応できそうであるが、より多量のデータを操作するケースおよび表計算ソフトの多系列表から生み出された情報を扱うケースを想定すると、さらに(表 3.)に記述した 3 つほどの機能が欲しい。

また、Data Island では、表示を制御するボタンなどの機能がデータ表示部分の下に表示されるため、(図 1.)のような画像が表示されるケースでは、少数のデータ表示でも画面下に消え、画面をスクロールアップしないと使えない。この点も工夫が必要である。

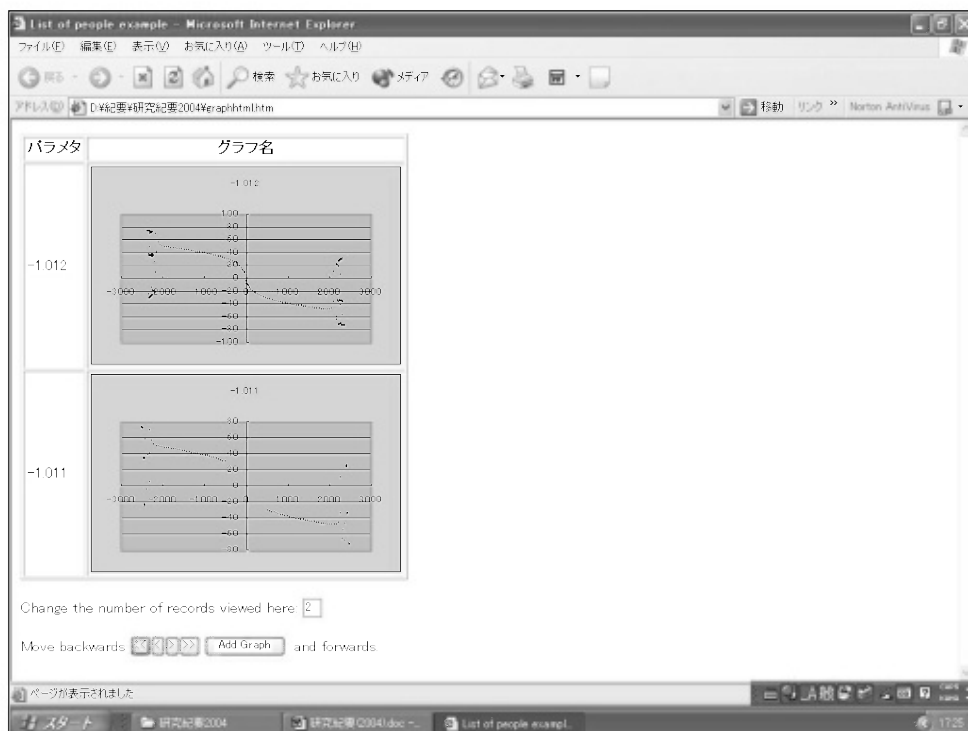
以上、Data Island が有する機能とそれに追加する機能を実現するホームページを作成することが目標となった。

2. 動的多画像表示ホームページ作成

2-1. フレーム方式動的ホームページ作成

まずフレーム機能を使用して、1 つの XML ファイルと XSLT スタイルシートから画面上から指定された条件に合致するデータのみ抽出して動的に画面表示が変化するホームページの基本的なパターンを作成してみた。『XSLT+

図 1. Data Island を使用したホームページ



(表 1. 図 1 ホームページのソーステキスト)

1	<TABLE id=tblGraph dataSrc=# xmlGraph cellPadding=3 dataPageSize=10 border=1>
2	<THEAD>
3	<TR>
4	<TH>パラメタ</TH>
5	<TH>グラフ名</TH>
6	</TR>
7	</THEAD>
8	<TBODY>
9	<TR>
10	<TD></TD>
11	<TD></TD>
12	</TR>
13	</TBODY>
14	</TABLE>

(表 2. Data Island の有する機能一覧表)

- ①画面に表示する画像数(データ数)を柔軟に変更できること。
- ②画面表示中のデータに連続する次のデータに表示を切り替えることができること。
- ③画面表示中のデータに連続する前のデータに表示を切り替えることができること。
- ④XML ファイル中の先頭データに戻って表示させることができること。
- ⑤XML ファイル中の最後尾データに跳んで表示させることができること。

(表 3. 追加機能一覧表)

①あるデータ項目 (タグ) に着目し、指定した範囲に該当するデータのみ抽出して表示する機能	(範囲指定抽出機能)
② XML ファイル内のデータを絶対位置で指定して、指定されたデータを先頭として、それに連続するデータを数個表示する機能	(絶対位置指定抽出機能)
③あるデータ項目 (タグ) が、その項目データが同一なら同一グループに属すると判断できる項目である場合、同一グループに属するデータのみ抽出して表示する機能	(同一グループ抽出機能)

XPath 実践マスター』(2002年ソフトバンク パブリッシング株式会社発行) を参考に作成した。

2-1-1. フレーム方式動的ホームページ全体機能

(図 2.) が作成した “bookxml0. html” ホームページの画面である。左フレームに表示させたい XML データの条件を指定して操作する 3 つの機能のブロックを作成した。

・左フレームの①ブロックは、(表 3.) ①の範囲指定抽出機能を実現するための部分である。XML ファイル内の<パラメタ>タグデータが、

「最小値」インプットボックスに入力された数値～「最大値」インプットボックスに入力された数値範囲に該当するもののみ抽出して表示させるための機能ブロックである。

・左フレームの②ブロックは、(表 3.) ②の絶対位置指定抽出機能を実現するための部分である。XML ファイル内のデータの絶対位置が入力された「先頭番号」データと一致するデータを先頭に、「表示数」インプットボックスに入力された数分の連続データを抽出して表示させるための機能ブロックである。

・左フレームの③ブロックは、(表 3.) ③の

図 2. フレーム方式動的多画像表示ホームページの様子



同一グループ抽出機能を実現するための部分である。XML ファイル内の<分類>タグデータが、「分類番号」インプットボックスに入力されたデータと一致するデータのみ抽出して表示させるための機能ブロックである。

左フレームの①、②、③ブロックの「実行」ボタンをクリックすると、入力された条件に合致するデータを XML ファイルから抽出し、右フレームに画像等の表示がされる仕組みになっている。

また、左フレームの①、②、③ブロックには、それぞれ「戻る」、「進む」ボタンがあるが、「実行」ボタンを押すことによって右フレームに表示されたデータを起点に、XML ファイル内の前・後のデータに表示を切り替える機能である。①ブロックの「戻る」、「進む」ボタンの機能は解りにくいですが、「最小値」インプットボックスに入力された数値と「最大値」インプットボックスに入力された数値の差分だけ、抽出範囲指定を前後に変化させる機能になっている。

(図 2.) “bookxml0. html” ホームページのソーステキストは、(表 4.) に掲載した。フレーム分割を規定している。

左フレームに表示するホームページは、“mokujixml0. html” である。

右フレームに表示するホームページの指定は無い。左フレームの “mokujixml0. html” にある JavaScript スクリプトが動作することによ

り、XML ファイルに XSLT スタイルシートを適用してホームページ形式に変換した結果を右フレームに表示させている。

使用している XML ファイルは “graph0. xml” だけであるが、XSLT ファイルは、左フレームの①、②、③ブロックにそれぞれ対応する 3 種類を使用している。

(図 3.) に “bookxml0. html” ホームページの機能概説図を記載した。

(表 5.) は、“graph0. xml” XML ファイルである。14行～15行は途中省略している箇所である。実際には、2012個の画像データが格納されている。<分類>タグが追加されているのが昨年度の研究紀要との違いである。これは、昨年度の研究紀要にて 1 つのホームページに表示した約100画像を同一グループとして扱うことを意図して追加したものである。(表 3.) にて記述した、あるグループに属するデータのみ抽出して表示する機能を入れるために使用するデータ項目として追加した。

2-1-2. 3 種の XSLT スタイルシート

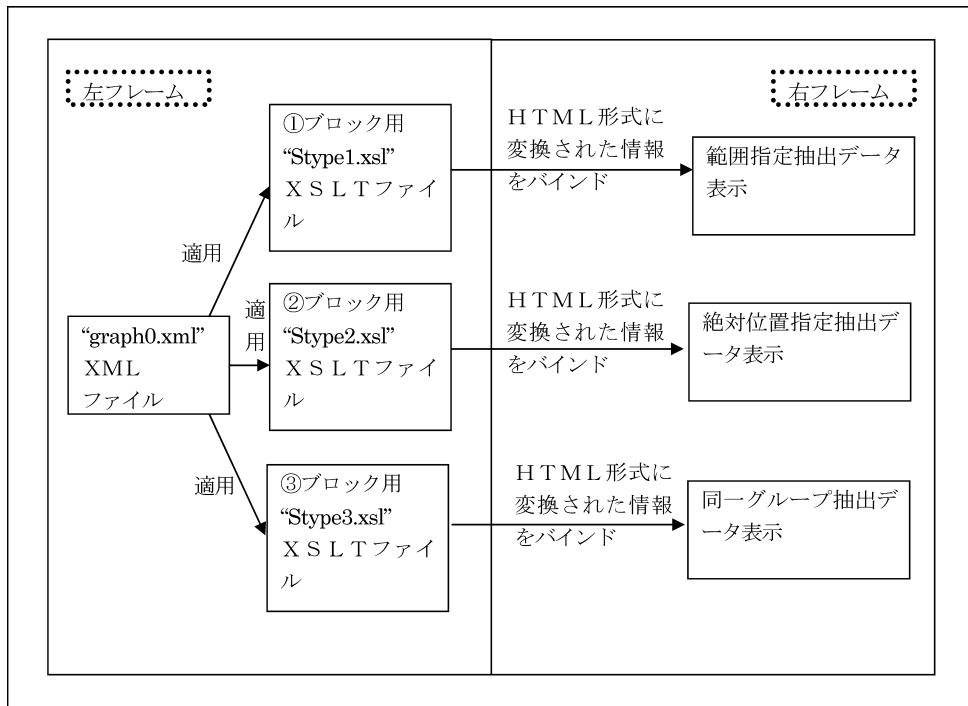
2-1-2-1. “Stype1. xsl” スタイルシート

(表 6.) は、“Stype1. xsl” XSLT ファイルのテキストである。XML ファイル内の<パラメタ>タグデータに着目し、指定された範囲に該当するデータのみ抽出して HTML 形式に変換

(表 4 . “bookxml0. html” ソーステキスト)

1	<HTML>
2	<HEAD>
3	<TITLE>グラフ表示パターン 0 </TITLE>
4	</HEAD>
5	<FRAMESET COLS="30%,80%">
6	<FRAME NAME="mokuji" TARGET="child" SRC="mokujixml0.html">
7	<FRAME NAME="child" SRC="">
8	</FRAMESET>
9	</HTML>

図 3. “bookxml0.html” ホームページ機能概説図



(表 5. “graph0. xml” テキスト)

1	<?xml version="1.0" encoding="Shift_JIS"?>
2	
3	<graph>
4	<record>
5	<分類>1</分類>
6	<パラメタ>-1.012</パラメタ>
7	<グラフ名>F0.files/L1_image001.gif</グラフ名>
8	</record>
9	<record>
10	<分類>1</分類>
11	<パラメタ>-1.011</パラメタ>
12	<グラフ名>F0.files/L2_image001.gif</グラフ名>
13	</record>
14	・
15	・
16	<record>
17	<分類>20</分類>
18	<パラメタ>0.999</パラメタ>
19	<グラフ名>F0.files/L2012_image001.gif</グラフ名>
20	</record>
21	</graph>

して表示する XSLT ファイルである。

3 行目～4 行目が、minvalue, maxvalue と

いう 2 つの変数を用意している箇所である。そ

れぞれ初期値として、-1.012、-1.000を指定し

(表 6. “Stype1. xsl” XSLT ファイルのテキスト)

```

1 <?xml version="1.0" encoding="Shift JIS" ?>
2 <xsl : stylesheet xmlns : xsl="http : //WWW.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl : variable name="minvalue">-1.012</xsl : variable>
4 <xsl : variable name="maxvalue">-1.000</xsl : variable>
5
6 <xsl : template match="/">
7
8 <HTML>
9 <HEAD>
10 </HEAD>
11 <BODY>
12 <PRE>分類番号 通算番号 パラメタ値</PRE>
13 <HR />
14 <FORM>
15 <xsl : for-each select="graph/record">
16 <xsl : choose>
17 <xsl : when test="パラメタ >= $minvalue and パラメタ <= $maxvalue">
18 <xsl : element name="input">
19 <xsl : attribute ame="type">text</xsl : attribute>
20 <xsl : attribute name="size">10</xsl : attribute>
21 <xsl : attribute name="name">kindSave</xsl : attribute>
22 <xsl : attribute name="value">
23 <xsl : value-of select="分類"/>
24 </xsl : attribute>
25 </xsl : element>
26 <xsl : element name="input">
27 <xsl : attribute name="type">text</xsl : attribute>
28 <xsl : attribute name="size">10</xsl : attribute>
29 <xsl : attribute name="name">positionSave</xsl : attribute>
30 <xsl : attribute name="value">
31 <xsl : number value="position()"/>
32 </xsl : attribute>
33 </xsl : element>
34 <xsl : element name="input">
35 <xsl : attribute name="type">text</xsl : attribute>
36 <xsl : attribute name="size">10</xsl : attribute>
37 <xsl : attribute name="name">paramSave</xsl : attribute>
38 <xsl : attribute name="value">
39 <xsl : value-of select="パラメタ"/>
40 </xsl : attribute>
41 </xsl : element>
42
43 <xsl : element name="IMG">
44 <xsl : attribute name="SRC">
45 <xsl : value-of select="グラフ名" />
46 </xsl : attribute>
47 </xsl : element>
48 <BR/>
49 </xsl : when>
50 </xsl : choose>
51 </xsl : for-each>
52 </FORM>
53 </BODY>
54 </HTML>
55 </xsl : Template>
56 </xsl : stylesheet>

```

である。17行目が、データを抽出するための条件判定部分であり、3行目～4行目で定義された minvalue, maxvalue を条件判定部分に入れて、変数内の数値を変更することにより抽出されるデータを切り替えることができるようになっている。これらの3行目～4行目および17行目が昨年度の研究紀要での静的な XSLT ファイルとの大きな違いである。

また18～41行目に、3つのインプットボックスを作成し、表示された各画像データの<分類>データ、XML ファイル内での先頭からの絶対位置と、<パラメタ>データを表示させた。これらのデータがグラフ画像といっしょに表示されることにより、使いやすくなるとともに、後にいろいろと活用できる可能性を考えて追加したものである。

2-1-2-2. “Style2. xsl” スタイルシート

(表7.) は、“Style2. xsl” XSLT ファイルのテキストである。XML ファイル内の絶対位置で指定して抽出し、HTML 形式に変換して表示する XSLT ファイルである。

3行目～4行目が、minvalue, maxvalue という2つの変数を用意している箇所である。それぞれ初期値として、1、10を指定してある。17行目が、データを抽出するための条件判定部分であり、3行目～4行目で定義された minvalue, maxvalue を条件判定部分に入れて、変数内の数値を変更することにより抽出されるデータを切り替えることができるようになっている。position () は、データの XML ファイル内での絶対位置を取得できる関数である。18～41行目は、(表6.) “Style1. xsl” XSLT ファイルと同じである。

2-1-2-3. “Style3. xsl” スタイルシート

(表8.) は、“Style3. xsl” XSLT ファイルの

テキストである。XML ファイル内の<分類>タグデータに着目し、指定された値に合致するデータのみ抽出して、HTML 形式に変換して表示する XSLT ファイルである。

3行目が、grNo という変数を用意している箇所である。初期値として1を指定してある。16行目が、データを抽出するための条件判定部分であり、<分類>タグデータが、grNo という変数内に格納されたデータと合致するもののみ抽出している部分である。

17～40行目は、(表6.) “Style1. xsl” XSLT ファイルの18～41行目の内容と同じである。

2-1-3. 左フレームページ

(表9.) は、左フレームに表示するページ “mokuji.xml0. html” のソーステキストである。左フレームには①、②、③の3つの異なった機能ブロックが存在する。JavaScript スクリプト部分を全て掲載するには長くなってしまったので、スクリプト部分は、①機能ブロックに対応する部分のみを掲載し、②、③機能ブロックに対応する部分は省略した。①機能ブロック内の 実行、戻る、進む ボタンがクリックされると、それぞれ “graphGo1 ()”、“downGo1 ()”、“upGo1 ()” 関数が動作する。

(表9.) のソーステキスト重要ポイントの説明は、(表10.) に記載した。DOM (Document Object Model) の関数を使用して、XML ファイルに XSLT スタイルシートを適用して HTML 形式に変換している。詳細な解説については、『XSLT+XPath 実践マスター』(2002年ソフトバンク パブリッシング株式会社発行)などに記述されているので省略した。

XML ファイルに、XSLT ファイルを適用して HTML 形式に変換して動的に表示内容を変

(表 7. “Stype2. xsl” XSLT ファイルのテキスト)

```

1 <?xml version="1.0" encoding="Shift_JIS" ?>
2 <xsl : stylesheet xmlns : xsl="http : //WWW.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl : variable name="minvalue">1</xsl : variable>
4 <xsl : variable name="maxvalue">10</xsl : variable>
5
6 <xsl : template match="/">
7
8 <HTML>
9 <HEAD>
10 </HEAD>
11 <BODY>
12 <PRE>分類番号 通算番号 パラメタ値</PRE>
13 <HR />
14 <FORM>
15 <xsl : for-each select="graph/record">
16 <xsl : choose>
17 <xsl : when test="position() &gt;= $minvalue and position() &lt;= $maxvalue">
18 <xsl : element name="input">
19 <xsl : attribute name="type">text</xsl : attribute>
20 <xsl : attribute name="size">10</xsl : attribute>
21 <xsl : attribute name="name">kindSave</xsl : attribute>
22 <xsl : attribute name="value">
23 <xsl : value-of select="分類" />
24 </xsl : attribute>
25 </xsl : element>
26 <xsl : element name="input">
27 <xsl : attribute name="type">text</xsl : attribute>
28 <xsl : attribute name="size">10</xsl : attribute>
29 <xsl : attribute name="name">positionSave</xsl : attribute>
30 <xsl : attribute name="value">
31 <xsl : number value="position()" />
32 </xsl : attribute>
33 </xsl : element>
34 <xsl : element name="input">
35 <xsl : attribute name="type">text</xsl : attribute>
36 <xsl : attribute name="size">10</xsl : attribute>
37 <xsl : attribute name="name">paramSave</xsl : attribute>
38 <xsl : attribute name="value">
39 <xsl : value-of select="パラメタ" />
40 </xsl : attribute>
41 </xsl : element>
42
43 <xsl : element name="IMG">
44 <xsl : attribute name="SRC">
45 <xsl : value-of select="グラフ名" />
46 </xsl : attribute>
47 </xsl : element>
48 <BR />
49 </xsl : when>
50 </xsl : choose>
51 </xsl : for-each>
52 </FORM>
53 </BODY>
54 </HTML>
55 </xsl : template>
56 </xsl : stylesheet>

```

(表 8 . “Stype3. xsl” XSLT ファイルのテキスト)

```

1 <?xml version="1.0" encoding="Shift_JIS" ?>
2 <xsl : stylesheet xmlns : xsl="http : //WWW.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl : variable name="grNo">1</xsl : variable>
4
5 <xsl : template match="/">
6
7 <HTML>
8 <HEAD>
9 </HEAD>
10 <BODY>
11 <PRE>分類番号 通算番号 パラメタ値</PRE>
12 <HR />
13 <FORM>
14 <xsl : for-each select="graph/record">
15 <xsl : choose>
16 <xsl : when test="分類 = $grNo">
17 <xsl : element name="input">
18 <xsl : attribute name="type">text</xsl : attribute>
19 <xsl : attribute name="size">10</xsl : attribute>
20 <xsl : attribute name="name">kindSave</xsl : attribute>
21 <xsl : attribute name="value">
22 <xsl : value-of select="分類" />
23 </xsl : attribute>
24 </xsl : element>
25 <xsl : element name="input">
26 <xsl : attribute name="type">text</xsl : attribute>
27 <xsl : attribute name="size">10</xsl : attribute>
28 <xsl : attribute name="name">positionSave</xsl : attribute>
29 <xsl : attribute name="value">
30 <xsl : number value="position()" />
31 </xsl : attribute>
32 </xsl : element>
33 <xsl : element name="input">
34 <xsl : attribute name="type">text</xsl : attribute>
35 <xsl : attribute name="size">10</xsl : attribute>
36 <xsl : attribute name="name">paramSave</xsl : attribute>
37 <xsl : attribute name="value">
38 <xsl : value-of select="パラメタ" />
39 </xsl : attribute>
40 </xsl : element>
41
42 <xsl : element name="IMG">
43 <xsl : attribute name="SRC">
44 <xsl : value-of select="グラフ名" />
45 </xsl : attribute>
46 </xsl : element>
47 <BR />
48 </xsl : when>
49 </xsl : choose>
50 </xsl : for-each> </FORM>
51 </BODY>
52 </HTML>
53 </XSLTemplate>
54 </xsl : stylesheet>

```

(表 9. 左フレーム “mokujixml0. html” ソーステキスト)

```

1 <HTML>
2 <HEAD>
3 <TITLE> 目次</TITLE>
4   <script type="text/JScript">
5     <!--
6     function graphGo1() {
7       var minData=paramMin. value ;
8       var maxData=paramMax. value ;
9
10      if(isNaN(minData)==true) {
11        alert('数値を入力してください！') ;
12        paramMin. value="" ;
13        return(0) ;
14      }
15      if(minData. length<=0) {
16        paramMin. value=0 ;
17        minData=paramMin. value ;
18      }
19
20      if(isNaN(maxData)==true) {
21        alert('数値を入力してください！') ;
22        paramMax. value="" ;
23        return(0) ;
24      }
25      if(maxData. length<=0) {
26        paramMax. value=0 ;
27        maxData=paramMax. value ;
28      }
29
30      var XMLdoc=new ActiveXObject("MSXML2. DOMdocument") ;
31      xmldoc. async=false ;
32      xmldoc. load('graph0. xml') ;
33
34      var stylesheet=new ActiveXObject("MSXML2. DOMdocument") ;
35      stylesheet. async=false ;
36      stylesheet. load('Style1. xsl') ;
37
38      var paramValue=stylesheet. getElementsByTagName('xsl : variable') ;
39      paramValue(0). text=minData ;
40      paramValue(1). text=maxData ;
41
42      var htmlData=xmlDoc. transformNode(stylesheet) ;
43      var Chikan=htmlData. replace("UTF-16","shift_JIS") ;
44      htmlData=Chikan ;
45      top. frames [1] . document. close() ;
46      top. frames [1] . document. write(htmlData) ;
47      top. frames [1] . document. close() ;
48    }
49
50    function upGo1() {
51      var interval ;
52
53      interval=parseFloat(paramMax. value)- parseFloat(paramMin. value) ;
54

```

```

55     paramMin. value= paramMax. value ;
56     paramMax. value= parseFloat(paramMin. value) + interval ;
57     graphGo1() ;
58
59 }
60
61 function downGo1() {
62     var interval ;
63
64     interval=parseFloat(paramMax. value)- parseFloat(paramMin. value) ;
65
66     paramMax. value= paramMin. value ;
67     paramMin. value= parseFloat(paramMin. value)-interval ;
68     graphGo1() ;
69 }
70
71     (②、③機能ブロック対応部分——省略箇所)
72
73     //-->
74     </script>
75
76 </HEAD>
77 <BODY onLoad="graphGo1()">
78 ◇約2000個のグラフ<BR>
79 データ XML ファイルを基<BR>
80 にグラフ表示<BR>
81 <HR>
82 ①パラメタ範囲指定</BR>
83 有効範囲-1.012から1.000
84     最小値<input type="text" size="30" name="paramMin" value="-1.012"></BR>
85     最大値<input type="text" size="30" name="paramMax" value="-1.000"></BR>
86     <button onClick="graphGo1()">実行</button>
87     <button onClick="downGo1()">戻る</button>
88     <button onClick="upGo1()">進む</button></BR>
89 <HR>
90 ②通算番号で選択</BR>
91 有効範囲 1 から2012<BR>
92 先頭番号<input type="text" size="10" name="paramTopNo" value="1"></BR>
93 表示数<input type="text" size="10" name="paramCNo" value="10"></BR>
94     <button onClick="graphGo2()">実行</button>
95     <button onClick="downGo2()">戻る</button>
96     <button onClick="upGo2()">進む</button></BR>
97 <HR>
98 ③分類番号で選択</BR>
99 有効範囲 1 から20<BR>
100 分類番号<input type="text" size="10" name="paramGrNo" value="1"></BR>
101     <button onClick="graphGo3()">実行</button>
102     <button onClick="downGo3()">戻る</button>
103     <button onClick="upGo3()">進む</button></BR>
104 </BODY>
105 </HTML>

```

(表10. “mokujixml0. html” ソーステキスト説明表)

10～28行目	インプットボックスに入力された数値をチェックしている部分で、省略しても特に重大な問題は無いと思われる。
30行目～47行目	XML ファイルに XSLT スタイルシートを適用して HTML 形式に変換したものを、右フレームに表示させている重要な箇所である。
32行目	“graph0. xml” XML ファイルを読み込んでいる箇所である。
36行目	“Stype1. xsl” XSLT ファイルを読み込んでいる箇所である。
39行目～40行目	“Stype1. xsl” XSLT スタイルシート内の 2 つの変数 \$minvalue、\$maxvalue それぞれに、「最小値」インプットボックスに入力された数値と「最大値」インプットボックスに入力された数値を設定している箇所である。
42行目	“graph0. xml” XML ファイルに、“Stype1. xsl” XSLT スタイルシートを適用し、HTML 形式に変換している箇所である。
45行目～47行目	HTML 形式に変換されたものを右フレームに書き込んで表示している箇所である。

更している核心部分は、20行足らずの JavaScript スクリプトコードで実現できた。XML ファイル名称、XSLT ファイル名称を記述する部分および、XSLT スクリプト内変数に抽出条件となる値をセットする部分などを変更するくらいで、汎用的に使えるスクリプトである。②、③機能ブロック用のスクリプトもほぼ同様であった。

2-1-4. フレーム方式動的ホームページについての考察

このホームページは、表計算ソフトの多系列表から生み出された 1 つの XML ファイル形式情報から 3 つのパターンで情報を抽出して HTML 形式に変換するための 3 種の XSLT スタイルシートを使って抽出結果を右フレーム側に表示させる 2 分割フレームのホームページの基本型を構築しようと意図して作成したものである。

“Stype1. xsl”スタイルシート、“Stype3. xsl”スタイルシート内に現れるタグの名称、変数の初期値、JavaScript スクリプトコード内のタグの名称などの修正を施す程度の手間で、今回以外のケースでも画像を表示するホームページと

して使用することができると思われる。また画像以外のタグのデータを表示させたい場合でも、スタイルシートにテーブル構造を作成し、適当なテーブル枠内にタグデータを表示するコードを追加するくらいで対応できると思われる。

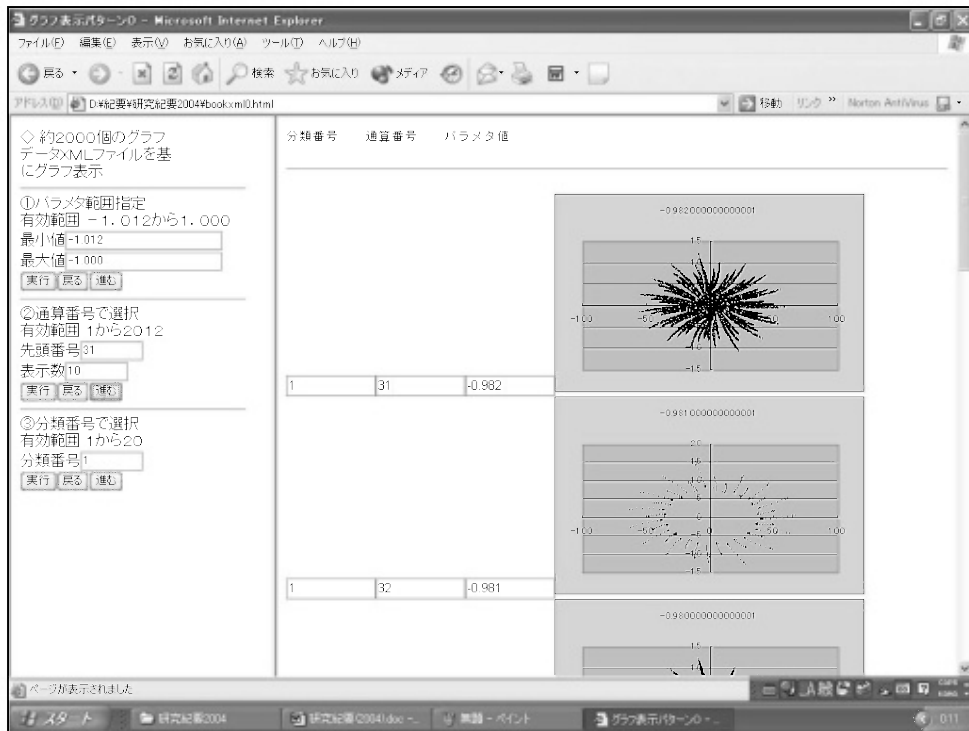
JavaScript スクリプトは使用したが、それほど難解なコーディングにはなっていないと思われる。一応 1 つの多画像表示ホームページの基本的なパターンとして利用できそうである。

しかし、2 つほどの問題点がある。左フレームの①機能ブロックでは、入力された最大値と最小値の差分を使い、、 ボタンが押された時には、その差分だけ前方および後方に進んで表示するデータを切り替えているが、差分が実数の場合には、小数点以下の小さな誤差が蓄積し、小数点以下に多桁数値が表示されるケースが発生し美しくない。これは、ケースバイケースで、差分の小数点下のある桁以下を切り捨てるなどの処置を追加する必要がある。

もう 1 つの問題点は、左フレームの①、②、③機能のブロックを連動させるのが難しい点である。

例えば (図 4.) は、初期状態から左フレーム②機能ブロックの ボタンを 3 回クリッ

図 4. フレーム方式動的多画像表示ホームページの様子

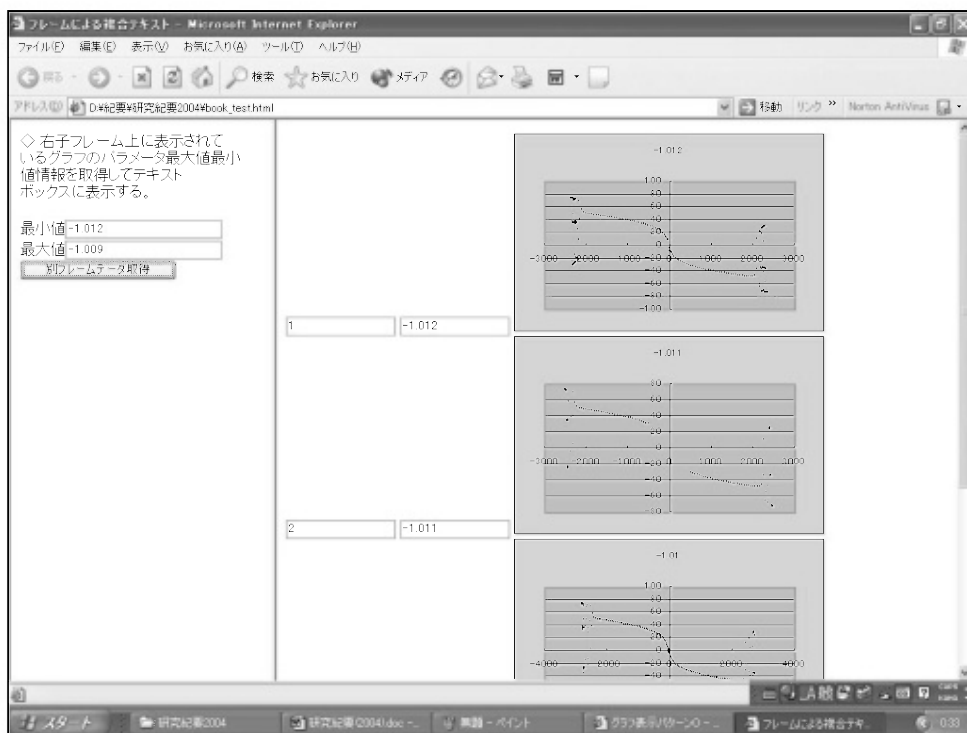


クすることにより、右フレームに XML ファイル内絶対位置31番のデータを先頭に10個分のデータが表示されている様子である。この場合、先頭に表示されている絶対位置31番のデータは、パラメタ値は、 -0.982 のデータであるので、①機能ブロックの「最小値」インプットボックス内の表示は、連動して -0.982 と自動的に変化するように動作をさせるのが好ましい。「最大値」インプットボックスの表示も連動してそれに自動変化するべきである。右フレームには、パラメタ値、XML ファイル内での絶対位置、分類値などが表示されているインプットボックス要素を作成してあるので、その内容を参照できれば、右フレームの表示状態の実情に合致した表示に、左フレームの①、②、③機能ブロックのインプットボックス表示を修正することが可能である。

しかし、それはなかなか難しいことのようにある。右フレームに単純な HTML 形式のテキストが表示されている場合では、左フレームから右フレームのインプットボックスの内容にアクセスすることは容易である。例えばテスト用に作成した(図5.) ホームページの右フレームは、今回作成したホームページと同様に表示されているが、実際は、単純な HTML 形式のページであり、左フレームから右フレーム内のインプットボックスに格納されている情報に簡単にアクセスできる。(表11.) に右フレームページのソーステキストを掲載した。

(表12.) に左フレームページのソーステキストを掲載した。9 行目から13行目のスクリプト部分が右フレームに表示されているパラメタデータの最大値と最小値を取得して、その結果を「最大値」、「最小値」インプットボックスに書

図 5. テスト用ホームページの様子



(表11. テスト用ホームページ右フレームページソーステキスト)

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM>
  <input type="text" name="countValue" value="1"> </input>
  <input type="text" name="paramValue" value="-1.012"> </input>
  <IMG SRC="F0.files/L1_image001.gif"> <BR>

  <input type="text" name="countValue" value="2"> </input>
  <input type="text" name="paramValue" value="-1.011"> </input>
  <IMG SRC="F0.files/L2_image001.gif"> <BR>

  <input type="text" name="countValue" value="3"> </input>
  <input type="text" name="paramValue" value="-1.01"> </input>
  <IMG SRC="F0.files/L3_image001.gif"> <BR>

  <input type="text" name="countValue" value="4"> </input>
  <input type="text" name="paramValue" value="-1.009"> </input>
  <IMG SRC="F0.files/L4_image001.gif"> <BR>
</FORM>
</BODY>
</HTML>
```

(表12. テスト用ホームページ左フレームページソーステキスト)

1	<HTML>
2	<HEAD>
3	<TITLE>目次</TITLE>
4	<script type="text/JScript">
5	<!--
6	function getChildInfo() {
7	var paramLength;
8	
9	paramLength= top.frames [1] .document.forms [0] .paramValue.length
10	if (paramLength>0) {
11	paramMin.value=top.frames [1] .document.forms [0] .paramValue [0] .value;
12	paramMax.value=top.frames [1] .document.forms [0] .paramValue [paramLength-1] .value;
13	}
14	}
15	//-->
16	</script>
17	
18	</HEAD>
19	<BODY>
20	◇右子フレーム上に表示されて
21	いるグラフのパラメータ最大値
22	最小値情報を取得してテキスト
23	ボックスに表示する。
24	最小値<input type="text" size="30" name="paramMin"></BR>
25	最大値<input type="text" size="30" name="paramMax"></BR>
26	<button onClick="getChildInfo()">別フレームデータ取得</button>
27	</BR></BR>
28	</BODY>
29	</HTML>

き込む動作を実行する。(図5.)は、左フレームの「別フレームデータ取得ボタン」をクリックして“getChildInfo ()”関数を動作させ、「最大値」、「最小値」インプットボックス内に右フレームから取得したデータが格納されている様子を示している。

しかし、同様なスクリプトを、(図2.)の左フレームページ“mokujixml0.html”のスクリプトに入れても動作しないようである。(図2.)の右フレームの表示が単純なHTMLテキストファイルでは無いこと、XMLファイルにXSLTファイルを適用して変換した結果が、見た目はHTML形式で表示されているが、ブラウザ内部的な状況は単純なHTMLファイル表示の状況とは違いがあるのかもしれない。単純

なHTMLテキスト形式では動作するDOMの関数が動作しない状況になっているのかもしれない。

あるいは、左フレームから右フレーム内の情報にアクセスするという複雑な状況の相乗効果で関数が動作しない状況になっているということも言えるのかもしれない。

2-2. テーブル方式動的ホームページ作成

2-1-4.節で指摘した問題点が解消される可能性および、フレームの使用が今後の発展において諸々の操作を困難にする可能性を考慮して、もう1つフレームを使用しない方法にて類似の機能を有するホームページ作成を試みた。将来的にCSS(カスケーディングスタイルシート)

の技術が進歩すれば、2-1. 節ホームページの左フレームに対応する部分を画面左上に fix（固定）させ画面のスクロールアップ・ダウンに影響を受けないブロックとして表示させることも可能になるようである。しかし、現状では不可能なので 2 つの列を持つテーブル構造を持ち、左の列に 2-1. 節ホームページの左フレームに対応する部分を表示させ、右の列に XML データを基に HTML 形式に変換された結果を表示するホームページを作成してみた。(図 6.) が作成した “bookxml4. html” ホームページの様子である。テーブル 1 列目枠内の部分は、2-1. 節で作成したホームページの左フレームと全く同じ機能である。また 2-1. 節で作成したホームページと全く同一の 1 つの XML ファイルと、3 種類の XSLT ファイルを使用している。2-1. 節の (図 3.) と同様に作成したホームページの機能概説図を (図 7.) に記載した。XML ファイ

ルに XSLT ファイルを適用して変換された HTML 形式データを表示する場所が、右フレームでは無く、テーブルの 2 列目枠内であることが異なるのみでその他に違いは無い。

したがって、(表 13.) に “bookxml4. html” のソーステキストを掲載するが、ほとんど 2-1. 節の左フレームホームページ “mokujixml0. html” のコードと同一である。データ抽出する ①、②、③の 3 つの異なった機能ブロックがあるが、スクリプト部分は、①機能ブロックに対応する部分のみを掲載し、②、③機能ブロックに対応する部分は省略した。また①機能ブロック用のスクリプトでも [実行] ボタンに対応する重要なスクリプトのみ掲載し、[戻る]、[進む] ボタンから呼び出されるスクリプトの記載は省略した。

(表 14) に “bookxml4. html” ソーステキスト

図 6. テーブル方式動的多画像表示ホームページの様子

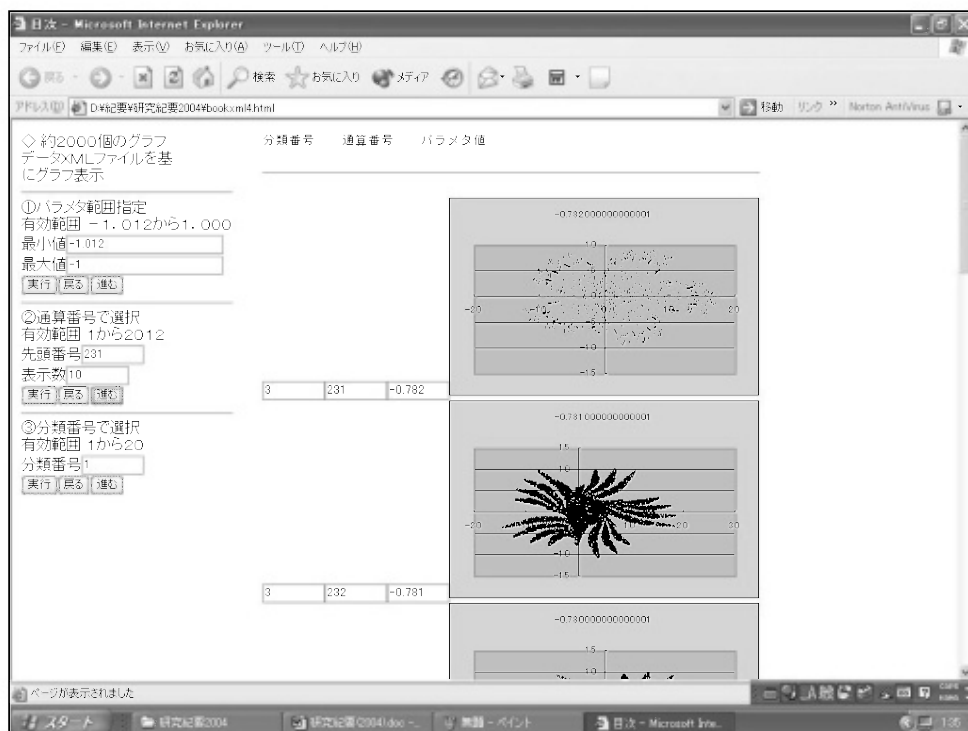
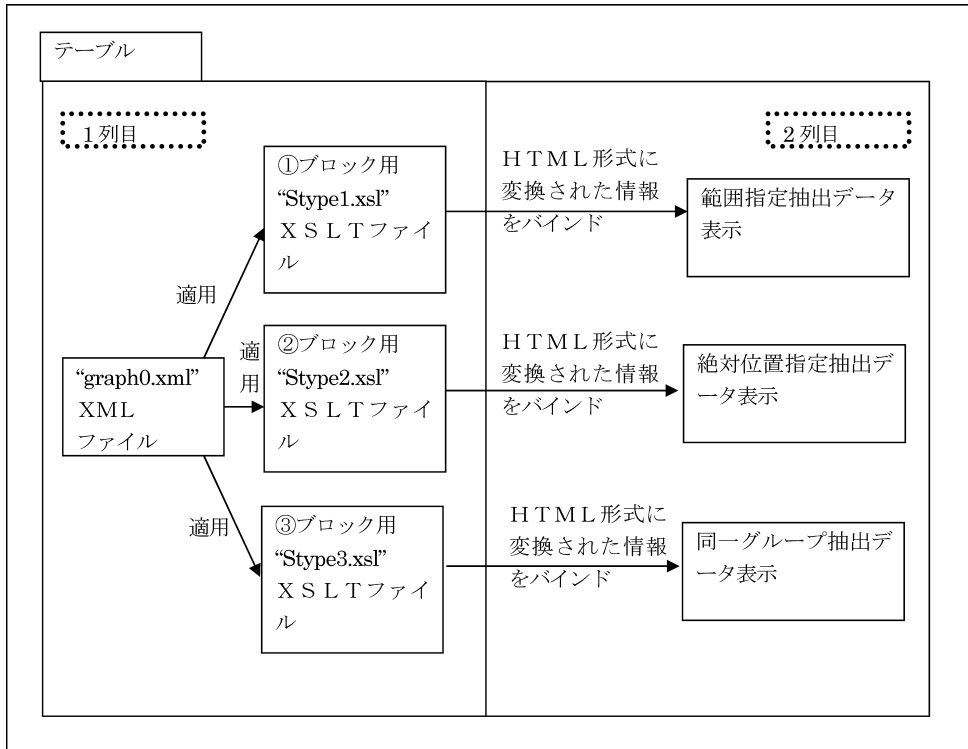


図 7. “bookxml4.html” ホームページ機能概説図



(表13. “bookxml4. html” ソーステキスト)

1	<HTML>
2	<HEAD>
3	<link rel="stylesheet" href="graph.css" type="text/css">
4	
5	<TITLE>テーブルを使用したデータ表示ページ</TITLE>
6	<script type="text/JScript">
7	<!--
8	function graphGo1() {
9	var minData=paramMin.value ;
10	var maxData=paramMax.value ;
11	
12	if (isNaN(minData) == true) {
13	alert("数値を入力してください!") ;
14	paramMin.value="" ;
15	return (0) ;
16	}
17	if (minData.length<=0) {
18	paramMin.value=0 ;
19	minData=paramMin.value ;
20	}
21	
22	if (isNaN(maxData) == true) {
23	alert("数値を入力してください!") ;

```

24     paramMax.value="";
25     return (0) ;
26 }
27 if (maxData.length <= 0) {
28     paramMax.value=0 ;
29     maxData=paramMax.value ;
30 }
31
32 var xmlDoc=new ActiveXObject("MSXML2.DOMdocument") ;
33 xmlDoc.async=false ;
34 xmlDoc.load ("graph0.xml") ;
35
36 var stylesheet=new ActiveXObject("MSXML2.DOMdocument") ;
37 stylesheet.async=false ;
38 stylesheet.load ("SType1.xml") ;
39
40 var paramValue=stylesheet.getElementsByTagName("xsl : variable") ;
41 paramValue(0).text=minData ;
42 paramValue(1).text=maxData ;
43
44 var htmlData=xmlDoc.transformNode(stylesheet) ;
45 var Chikan=htmlData.replace("UTF-16","shift_JIS") ;
46 htmlData=Chikan ;
47
48 graphArea.innerHTML=htmlData ;
49 }
50
51 (②、③機能ブロック対応部分——省略箇所)
52
53 //-->
54 </script>
55
56 </HEAD>
57 <BODY onLoad="graphGo1()">
58     <TABLE cellSpacing=0 cellPadding=0 border=0>
59         <TR>
60             <TD valign="top">
61 ◇約2000個のグラフ<BR>
62 データ XML ファイルを基<BR>
63 にグラフ表示<BR>
64 <HR>
65 ①パラメタ範囲指定</BR>
66 有効範囲-1.012から1.000<BR>
67     最小値<input type="text" size="30" name="paramMin" value="-1.012"></BR>
68     最大値<input type="text" size="30" name="paramMax" value="-1.000"></BR>
69     <button onClick="graphGo1()">実行</button>
70     <button onClick="downGo1()">戻る</button>
71     <button onClick="upGo1()">進む</button></BR>
72 <HR>
73 ②通算番号で選択</BR>
74 有効範囲 1 から2012<BR>
75 先頭番号<input type="text" size="10" name="paramTopNo" value="1"></BR>
76 表示数<input type="text" size="10" name="paramCNo" value="10"></BR>
77     <button onClick="graphGo2()">実行</button>
78     <button onClick="downGo2()">戻る</button>

```

79	<button onClick="upGo2()">進む</button></BR>
80	<HR>
81	③分類番号で選択</BR>
82	有効範囲 1 から20
83	分類番号<input type="text" size="10" name="paramGrNo" value="1"></BR>
84	<button onClick="graphGo3()">実行</button>
85	<button onClick="downGo3()">戻る</button>
86	<button onClick="upGo3()">進む</button></BR>
87	</TD>
88	<TD>
89	<div id="graphArea">
90	</div>
91	</TD>
92	</TR>
93	</TABLE>
94	
95	</BODY>
96	</HTML>

(表14. “bookxml4. html” ソーステキスト説明表)

3 行目	“graph. css” CSS スタールシートを参照する命令を記述した。
48行目	“graphArea” と名前をつけたブロックに HTML 形式に変換されたテキストを書き込んで表示している箇所である。
89行目～90行目	テーブル 2 列目に “graphArea” という名前のブロックを作成している箇所である。このブロックに抽出された XML データが HTML 形式に変換されて表示される。

(表15. “graph. css” スタイルシートテキスト)

div#graphArea { padding : 0em 0em 0em 2em ; }

説明表を掲載したが、2-1. 節の(表10. “moku-jixml0. html” ソーステキスト説明表) と殆ど同一の内容なので違いのある部分のみ記述した。

“graph. css” スタイルシートを参照している。これは(図 6.)の表示で、単に“graphArea”ブロックの表示が、左列に近接し過ぎていると格好がよくないので、2 文字分程度右に寄せて表示するように指定しているだけのものである。

2-3. テーブル方式動的ホームページについての考察

2-1-4. 節で指摘した問題点は、テーブルを使用する方法で試みても解消されなかった。1 列目の①、②、③機能ブロック内のインプットボックスデータを 2 列目の表示内容に連動させることはできなかった。自分の記述方法に問題があるのかもしれないが、現状上手くいっていない。

やはり、単純な HTML テキスト表示とは内部的に違いがあるのかもしれない。DOM の関数が動作しない状況になっているのかもしれない。

い。

テーブル方式のホームページの場合の問題点は、1列目枠内の①、②、③機能ブロック部分は画面の上方に表示させてあるが、画面を下方にスクロールすることにより上方に消えて隠れてしまうことである。この点が不便である。フレーム方式表示、テーブル方式表示のどちらの方法でも結局はXMLデータを抽出してHTML形式に変換して表示させたブロックのデータにDOMの関数を使用してアクセスすることが、現段階では上手くいっていない。現時点での状況では2-1.節のフレーム方式表示のほうが良い。

3. 結び

フレーム方式と、テーブル方式の2方法で、XMLファイルとXSLTスタイルシートの組み合わせで動的にグラフ画像を表示させるホームページを作成することができた。1-2.節で目標として掲げた機能は実現できたと思われる。昨年度の静的なホームページでは、20種のXMLファイルと20種のXSLTスタイルシートを使用した。今回の動的な2方法のホームページの場合では、1つのXMLファイルと3種のXSLTスタイルシートを使用したのみで、昨年度の静的なホームページより使い勝手の良いホームページができたと思われる。

結局JavaScriptスクリプトを使用する必要があったが、核心部分は20行弱のほぼ定型パターンのものであり、それ以外の表示情報を進ませたり、戻らせたりする「進む」、「戻る」ボタンに対応する部分は、足し算、引き算のちょっとした計算処理の単純なスクリプトで済んだ。その程度のスクリプトは最低限動的なホームページを作成するには不可欠でありしかたがないと思われる。

表示情報を切り替える操作ブロックには、①、

②、③の3つの操作ブロックを用意したことによりJavaScriptスクリプトの量も増えたが、この3種の操作があれば、表計算多系列表から生み出されたXMLファイル情報の抽出表示に汎用的に使えるのではないと思われる。タグ名称などをケースに合わせて修正するくらいで、それほどの手間をかけずに目的のホームページを作成できる。

表示情報を切り替える操作ブロックにあるインプットボックス内に表示される情報は、グラフ表示ブロックに表示中の情報に合わせて自動的に変化するようになっていたほうが好ましいのだろうが、現状は上手くいっていない。ブラウザソフトに組み込まれているDOM(Document Object Model)準拠パーサのサポートする関数を使用することによって、ブラウザに表示中のHTML文書、XML文書、XSLTスタイルシートテキスト内情報の取得、変更ができる筈であるが、今回のようなXMLファイルにXSLTスタイルシートを適用してHTML形式に変換したものをフレームまたはホームページ内のブロックにバインドして表示させているようなケースでは、DOMの関数が働かないのかもしれない。あるいは、自分のDOM関数の使い方に問題があるのかもしれない。現状は上手くいっていないが、将来的には状況が変わるかもしれない。この点については、今後も継続して注目していきたい。

ともかく今回、XMLファイルとXSLTスタイルシートの組み合わせで動的に情報を表示するホームページの作成は一応達成された。次の段階は、WWWサーバとの連携を考える段階である。今回のホームページの場合では、サーバ側からクライアント側に1本のXMLファイルが送信され、後はクライアント側のコンピュータ内でその1本のXMLファイルから部分抽出した情報を表示する方式になっている。今回の

ケースではXML ファイル内のデータ数（レコード数）は2000件程度であり、それほど巨大でも無いので WWW サーバとクライアント間ネットワークのトラフィック量が多過ぎるという問題は無いと思われる。しかし、WWW サーバ側に巨大な XML ファイルのデータベースが存在するケースおよび、複数に分割された XML ファイルが存在するようなケースでは、WWW サーバ側でクライアントに送信する情報を必要な分だけに絞り込んで送信する必要があるケースもある。

そのような WWW サーバが関係し、しかも XML ファイルと XSLT スタイルシートの組み合わせで動的に情報を表示させようとするようなケースはかなり難解な状況があると想像される。今回フレーム方式と、テーブル方式の2つの方法で、情報を抽出表示するホームページを作成したが、WWW サーバが関係する場合では特にフレーム方式ではかなり難しいことが想像される。通常のビジネス上表計算ソフトを利用するケースでは、それほど巨大な XML データが生み出されるようなケースはまずあり得ないと想像されるが、XML/XSLT 技術の利用について理解を深める意味では、そういう複雑なケースについて考察するという点にも意義があるかもしれない。

[参考文献]

- 1) Mark Wilson、Tracy Wilson 著、浜田真理訳、浜田光之監訳
『VB と ASP でつくる XML』2001年株式会社ピアソン・エデュケーション
- 2) Michael Kay 著、IDEA・C 訳、佐藤直生監修
『XSLT バイブル』2002年 株式会社インプレス
- 3) 井上孝司著
『Web コンテンツ作成のための XSLT 入門』2002年 株式会社毎日コミュニケーションズ
- 4) 大田一郎・柴田史久著
『XML ツールキット』2001年 株式会社秀和システム
- 5) PROJECT KySS / 宮坂雅輝著
『XML+XSL による Web サイトの構築と活用』2000年 ソフトバンク パブリッシング株式会社
- 6) PROJECT KySS / ビスケット株式会社著
『XSLT+XPath 実践マスター』2002年 ソフトバンク パブリッシング株式会社
- 7) 藤田泰徳著
『XML の徹底解説から ASP 連携 Web サイト構築まで XML 超入門』
2001年 有限会社セレンディップ
- 8) 古籾一浩著
『最新実用 HTML タグ辞典』2000年 株式会社技術評論社
- 9) 半場方人著
『詳解 Java Script 辞典』2002年 株式会社秀和システム