

## コンピュータ教室内サーバにおける ASP 環境を利用した XML ファイルを使用したホームページの作成

末 木 俊 之

Creation of the Homepage Which Used XML Files Using ASP Environment  
in the Computer Classroom Server

Toshiyuki SUEKI

### 1. ASP 環境を利用した XML ファイルを使 用するホームページ

昨年度の研究紀要では、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（画像）を切り換えて表示するホームページを作成した。画像データが格納された1本のXMLファイルがサーバ側からクライアント側に送信され、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（画像）を表示する方式になっている。クライアントパソコンのメモリ上でXMLデータファイルにXSLTスタイルシートを適用してHTML形式に変換して表示させているものであった。HTML形式に変換されたXMLデータ（画像）を表示するデータ表示ブロックとは別に、表示情報を切り替えるための独立した3つの操作ブロックを用意し、それぞれの操作ブロックは、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）②XML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、以上3種の画像表示切替機能を持たせた。しかし、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を、データ表示ブロックに表示中の情報に合わせて自動的に変化させるのは簡単にはいかないようだ。DOM(Document Object Model)の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、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。単純なHTMLファイルが表示されているのではない点に、クライアント上での操作の難点があるようであった。

今回は、データ表示ブロックとデータ操作ブロックの連携を簡単にするために、サーバ側のメモリ上にてXMLファイルにXSLTスタイルシートを適用してHTML形式に変換した後、単純なHTML形式のホームページとしてクライアント側に送信する方法で同様の機能を有するホームページを作成することを目標にした。

クライアント側では、単純な HTML 形式のページを扱えるので、DOM (Document Object Model) の関数が問題なく使用でき、データ表示ブロックとデータ操作ブロックの情報表示の整合性を取ることが簡単に実現できることが想像される。

また、自分が学生へのパソコン指導で使用しているコンピュータ実習室が、今年 9 月に新パソコン・サーバ環境 (OS は、Windows Server2003 Standard Edition) に更新された。簡単な操作で教室のサーバに WEB サーバの一種である IIS (Internet Information Server) サーバと、ASP (Active Server Pages) をセットアップすることができた。また通常のセットアップさえ完了すれば、サーバ上で XML ファイルに XSLT スタイルシートを適用して HTML 形式に変換するための関数を使える環境が整った。今回作成したホームページは、教室のサーバに格納して使用することができる。さらに、これまでは単なる HTML 形式のホームページとして作成してあったコンピュータ教室ホームページの一部を、サーバ側で XML ファイルに XSLT スタイルシートを適用して HTML 形式に変換して学生の使用するクライアントパソコンに送信する形態のページに変更してみようと考えた。具体的には、コンピュータ教室にて開講する演習科目のメニューデータを XML データファイル化してサーバ内に格納しておいて、指定された科目コードに応じたデータのみ抽出し、XSLT スタイルシートを適用して、HTML 形式の開講科目メニューに変換して送信するホームページである。このページの場合には、複数の XML ファイルから必要なデータのみ抽出する機能が必要とされたが、『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) に紹介されている方法を使って簡単に実現できた。

単純な XML ファイルの利用であれば、表計算ソフトを使ってデータの管理をすることも可能であり、それを基にホームページを運用することもできるだろう。表計算ソフトから生み出されるデータを XML データファイル化して、サーバ側に格納しておき、必要なデータのみ抽出して XSLT スタイルシートによって HTML 形式のホームページに変換して送信する、あくまで難解なコーディングと、複雑な XML ファイルの管理を必要としない簡単なホームページを作成するのが目的である。

## 2. ASP 環境を利用した多画像表示ホームページ作成

### 2-1. ホームページに求められる機能について

今回作成するホームページは、クライアントへの送信前にサーバ側にて ASP プロセスによりスクリプトが実行され、クライアントページから送信される画像切り替え要求を受信し、要求に応じて XML ファイルから必要なデータのみ抽出し、HTML 形式に変換し、単純なホームページを作成してクライアント側に送信するものである。(図 1.) が作成したページ (bookx-ml4-2. asp) の様子である。昨年度作成したページと見た目はほとんど同じである。画面左に表示情報を切り替えるための独立した 3 つの操作ブロック (①②③) がある。右側には、グラフ画像が表示されるデータ表示ブロックがある。このページはテーブル形式で作成した。1 行 2 列のテーブルから成っており、1 列目が操作ブロック (①②③)、2 列目がデータ表示ブロックとなっている。

3 つの操作ブロック (①②③) は、データ表示ブロックの表示画像を切り替えるものであり、機能は前年度作成したページと同様なので、説明は省略する。

3 つの操作ブロック (①②③) に対応して

XML ファイルから目的のデータを抽出して HTML 形式に変換するための 3 つの XSLT スタイルシート (SType1. xsl、SType2. xsl、SType3. xsl) および画像データを格納する XML ファイル (graph0. xml) がサーバ側に存在する点も昨年度のページとは違っている。3 つの XSLT スタイルシート (SType1. xsl、SType2. xsl、SType3. xsl) および XML ファイル (graph0. xml) については、昨年度と全く同じであるので説明を省略する。(図 2.) に作成した bookxml4-2. asp ページの機能概説を記述した。

データ表示の切り替え指令をクライアント側からサーバ側に送信し、受信したサーバ側でデータ抽出している。今回のページの場合、データ操作ブロックは①、②、③の 3 種類あり、それぞれの操作ブロックには、さらにデータ表示を切り替える方法が 3 通りあり (実行、戻る、進む)、クライアントからサーバへ送信される指令が割合複雑なケースとなった。このクライアント側からサーバ側への情報の受け渡しの実現

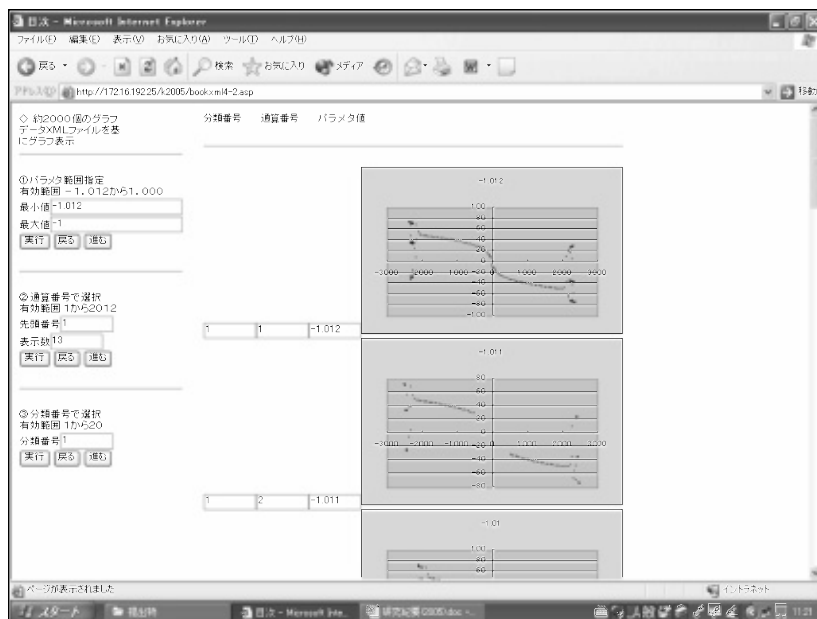
には少々悩み試行錯誤があった。しかし、完成してみるとシンプルなものができあがった。今回のケースのようにクライアント側で行っていた処理をサーバ側に移行させると、クライアント側の処理はシンプルになるが、クライアント／サーバ間の情報受け渡しに悩むケースがありそうである。

## 2-2. ホームページのソースファイル説明

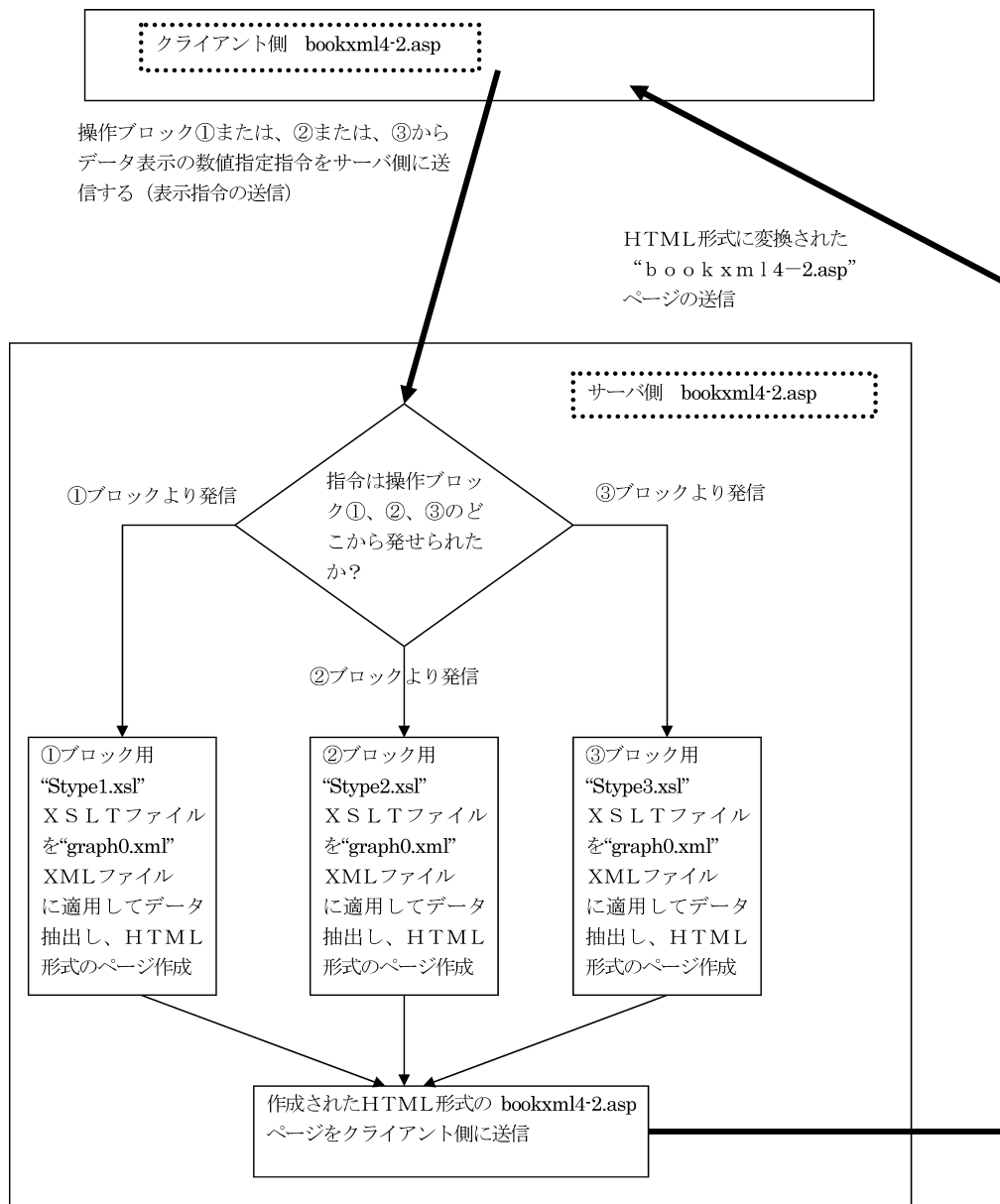
(表 2.) は、“bookxml4-2. asp” ページのソーステキストである。44 行目から 76 行目が表示を切り替えるための指令をサーバ側 “bookxml4-2. asp” に送信するための <form> ブロックである。データを抽出するための指令のタイプ、数値データおよび、動作の区別 (実行・戻る・進む) をサーバ側に送信する。数値のアップ、ダウンはサーバ側で実行する。

80 行目から 163 行目は、サーバ側で実行される VBScript 部分である。(表 1.) にて説明を記述する。

10 行目～30 行目の “setParam ()” ファンクションは、クライアント側でホームページロード



(図 1. “bookxml4-2. asp” ホームページ)



(図 2. “bookxml4-2.asp” ホームページ機能概説図)

時に実行される JScript スクリプトである。DOM (Document Object Model) の関数を使用することによって、ホームページ上のテーブル右列に表示されるデータ表示ブロックの先頭データと最後尾データについての情報を取得して、テーブル左列の操作ブロックのインプットボックスの表示に反映している箇所である。デ

ータ表示ブロックは HTML 形式のテキストとになっているので、DOM の関数が問題なく機能している。

クライアント側にて XML ファイルに XSLT スタイルシートを適用して、HTML 形式に変換して表示させている方法と比べて、ASP の動作するサーバ側にて、HTML 形式に

(表 1. サーバ側実行部分スクリプトソーステキスト説明)

90行目～132行目	サーバ側から送信される指令のタイプ、パラメータ、数値データを受信して、数値のアップ／ダウンを実行する。
134行目～137行目	“graph0. xml” XML ファイルを読み込んでいる。
139行目～156行目	サーバ側から送信されるパラメータに従って、“Stype1. xml”、“Stype2. xml”、“Stype3. xml” の 3 種類のスタイルシート of のいずれかを読み込んでいる。
158行目～160行目	“graph0. xml”XML ファイルデータに、139行目～156行目で読み込んだスタイルシートを適用して HTML ファイル形式に変換している。
162行目	HTML 形式に変換されたテキストを書き込んで、クライアント側に送信するホームページを完成する。

(表 2. ホームページのソーステキスト)

1	<%@Language="VBScript"%>
2	<% option explicit %>
3	<HTML>
4	<HEAD>
5	<link rel="stylesheet" href="graph. css" type="text/css">
6	
7	<TITLE>目次</TITLE>
8	<script type="text/JScript">
9	<!--
10	function setParam() {
11	var type1;
12	var paramLength;
13	
14	paramLength=graph. paramSave. length;
15	if (paramLength==1) {
16	fm1. paramMin. value=graph. paramSave. value;
17	fm1. paramMax. value=graph. paramSave. value;
18	fm2. paramTopNo. value=graph. positionSave. value;
19	fm2. paramCNo. value=1;
20	fm3. paramGrNo. value=graph. kindSave. value;
21	}
22	else if(paramLength>1) {
23	fm1. paramMin. value=graph. paramSave [0] . value;
24	fm1. paramMax. value=graph. paramSave [paramLength-1] . value;
25	fm2. paramTopNo. value=graph. positionSave [0] . value;
26	fm2. paramCNo. value=graph. positionSave[paramLength-1]. value-
27	fm2. paramTopNo. value+1;
28	fm3. paramGrNo. value=graph. kindSave [0] . value;
29	}
30	}
31	//-->
32	</script>
33	
34	</HEAD>
35	<BODY onLoad="setParam()">
36	<TABLE cellSpacing=0 cellPadding=0 border=0>
37	<TR>
38	<TD valign="top">
39	◇約2000個のグラフ 
40	データ XML ファイルを基 
41	にグラフ表示 
42	<HR>

```

43
44 <FORM name="fm1" METHOD="post" ACTION="bookxml4-2. asp">
45 ①パラメタ範囲指定</BR>
46 有効範囲 -1.012から1.000<BR>
47 最小値<input type="text" size="30" name="paramMin" value="-1.012"></BR>
48 最大値<input type="text" size="30" name="paramMax" value="-1.000"></BR>
49 <input type="text" name="type" value="1" style="display : none ; ">
50 <input type="submit" name="go" value="実行">
51 <input type="submit" name="down" value="戻る">
52 <input type="submit" name="up" value="進む"></BR>
53 </FORM>
54 <HR>
55
56 <FORM name="fm2" METHOD="post" ACTION="bookxml4-2. asp">
57 ②通算番号で選択</BR>
58 有効範囲 1 から2012<BR>
59 先頭番号<input type="text" size="10" name="paramTopNo" value="1"></BR>
60 表示数<input type="text" size="10" name="paramCNo" value="10"></BR>
61 <input type="text" name="type" value="2" style="display : none ; ">
62 <input type="submit" name="go" value="実行">
63 <input type="submit" name="down" value="戻る">
64 <input type="submit" name="up" value="進む"></BR>
65 </FORM>
66 <HR>
67
68 <FORM name="fm3" METHOD="post" ACTION="bookxml4-2. asp">
69 ③分類番号で選択</BR>
70 有効範囲 1 から20<BR>
71 分類番号<input type="text" size="10" name="paramGrNo" value="1"></BR>
72 <input type="text" name="type" value="3" style="display : none ; ">
73 <input type="submit" name="go" value="実行">
74 <input type="submit" name="down" value="戻る">
75 <input type="submit" name="up" value="進む"></BR>
76 </FORM>
77 </TD>
78 <TD>
79 <div id="graphArea">
80 <%
81 dim type0, upDown
82 dim paramMin, paramMax
83 dim paramTopNo, paramCNo
84 dim paramGrNo, interval
85 dim filePath
86 dim xmldoc, stylesheet, xmlData, chikan
87 dim variableNode
88 dim htmlData
89
90 type0=Request. Form("type")
91 upDown=0
92 if Request. Form("go")="実行" then
93 upDown=0
94 elseif Request. Form("up")="進む" then
95 upDown=1
96 elseif Request. Form("down")="戻る" then
97 upDown=2

```

```

98     end if
99
100    if type0="" then
101        type0=1
102        paramMin=-1.012
103        paramMax=-1.000
104    elseif type0=1 then
105        paramMin=Request. Form("paramMin")
106        paramMax=Request. Form("paramMax")
107        if upDown=1 then
108            interval=paramMax-paramMin
109            paramMin=paramMax
110            paramMax=paramMin+interval
111        elseif upDown=2 then
112            interval=paramMax-paramMin
113            paramMax=paramMin
114            paramMin=paramMin-interval
115        end if
116    elseif type0=2 then
117        paramTopNo=Request. Form("paramTopNo")
118        paramCNo=Request. Form("paramCNo")
119        if upDown=1 then
120            paramTopNo=int(paramTopNo)+int(paramCNo)
121        elseif upDown=2 then
122            paramTopNo=int(paramTopNo)-int(paramCNo)
123        end if
124    else
125        type0=3
126        paramGrNo=Request. Form("paramGrNo")
127        if upDown=1 then
128            paramGrNo=int(paramGrNo)+1
129        elseif upDown=2 then
130            paramGrNo=int(paramGrNo)-1
131        end if
132    end if
133
134    filePath=Server. MapPath("./")
135    set xmldoc=Server. createObject("MSXML2. DOMdocument")
136    xmldoc. async=false
137    xmldoc. load(filePath & "/graph0. xml")
138
139    set stylesheet=Server. createObject("MSXML2. DOMdocument")
140    stylesheet. async=false
141
142    if type0=1 then
143        stylesheet. load(filePath & "/SType1. xsl")
144        set variableNode=stylesheet. getElementsByTagName("xsl : variable")
145        variableNode(0). text=paramMin
146        variableNode(1). text=paramMax
147    elseif type0=2 then
148        stylesheet. load(filePath & "/SType2. xsl")
149        set variableNode=stylesheet. getElementsByTagName("xsl : variable")
150        variableNode(0). text=paramTopNo
151        variableNode(1). text=int(paramTopNo)+int(paramCNo)-1
152    else

```

```

153     stylesheet. load(filePath & "/SType3. xsl")
154     set variableNode=stylesheet. getElementsByTagName("xsl : variable")
155     variableNode(0). text=paramGrNo
156 end if
157
158     htmlData=xmldoc. transformNode(stylesheet)
159     Chikan=replace(htmlData, "UTF-16", "shift_JIS")
160     htmlData=Chikan
161
162     Response. Write(htmlData)
163 %>
164     </div>
165 </TD>
166 </TR>
167 </TABLE>
168
169 </BODY>
170 </HTML>

```

変換してクライアント側に送信する方法では、サーバ側での負荷は大きくなるが、サーバ／クライアント間のトラフィック量の低下、クライアント側での DOM 関数を利用するスクリプトでの操作を容易にすることができる。

### 3. ASP 環境を利用した複数 XML ファイルを使用するホームページ作成

#### 3-1. ホームページ概説

2 章と同様に、サーバ側で XML ファイルに XSLT スタイルシートを適用して HTML 形式に変換して学生の使用するクライアントパソコンに送信する形態のページを作成することを試みた。すでに旧教室ではコンピュータ教室にて開講する演習科目のメニューデータを表示するホームページ（図 4. 参照）を作成して運用している。（図 4.）ページは、「コンピュータ初級 B」科目のメニュー画面であり、コンピュータ教室におけるメインホームページ（図 3. 参照）より、「コンピュータ初期 B メニュー」リンクをクリックすることにより展開するようになっている。同様に、「コンピュータ初期 C メニュー」リンクをクリックすれば、（図 4.）ページと類似の「コンピュータ初級 C」科目のメニュー

画面が表示される。「コンピュータ初級 B」科目のメニュー画面と「コンピュータ初級 C」科目のメニュー画面は、全く別物の単純な HTML 形式のホームページとして作成してあった。

今回は、見た目は（図 4.）と同様のページであるが、開講する演習科目のメニューデータを XML データファイル化してサーバ内に格納しておいて、指定された科目コードに応じたデータのみ抽出し、XSLT スタイルシートを適用して、HTML 形式の開講科目メニューに変換して送信する形式の 1 本のホームページを作成してみた。コンピュータ教室におけるメインホームページ（図 3. 参照）より、科目コードを指定して呼び出し、その科目コードに一致するデータを自動的に XML データファイルから抽出して表示する形態のページである。

2 章で作成したホームページは、1 つの XML データファイル内から必要なデータのみを抽出して表示させるタイプのページであるが、このページの場合には、複数の XML データファイルから必要なデータのみを抽出して表示する形式のページとして作成できれば運用が容易であると思われた。複数の XML データファイルとは、科目毎に別ファイルとしてメニューデ





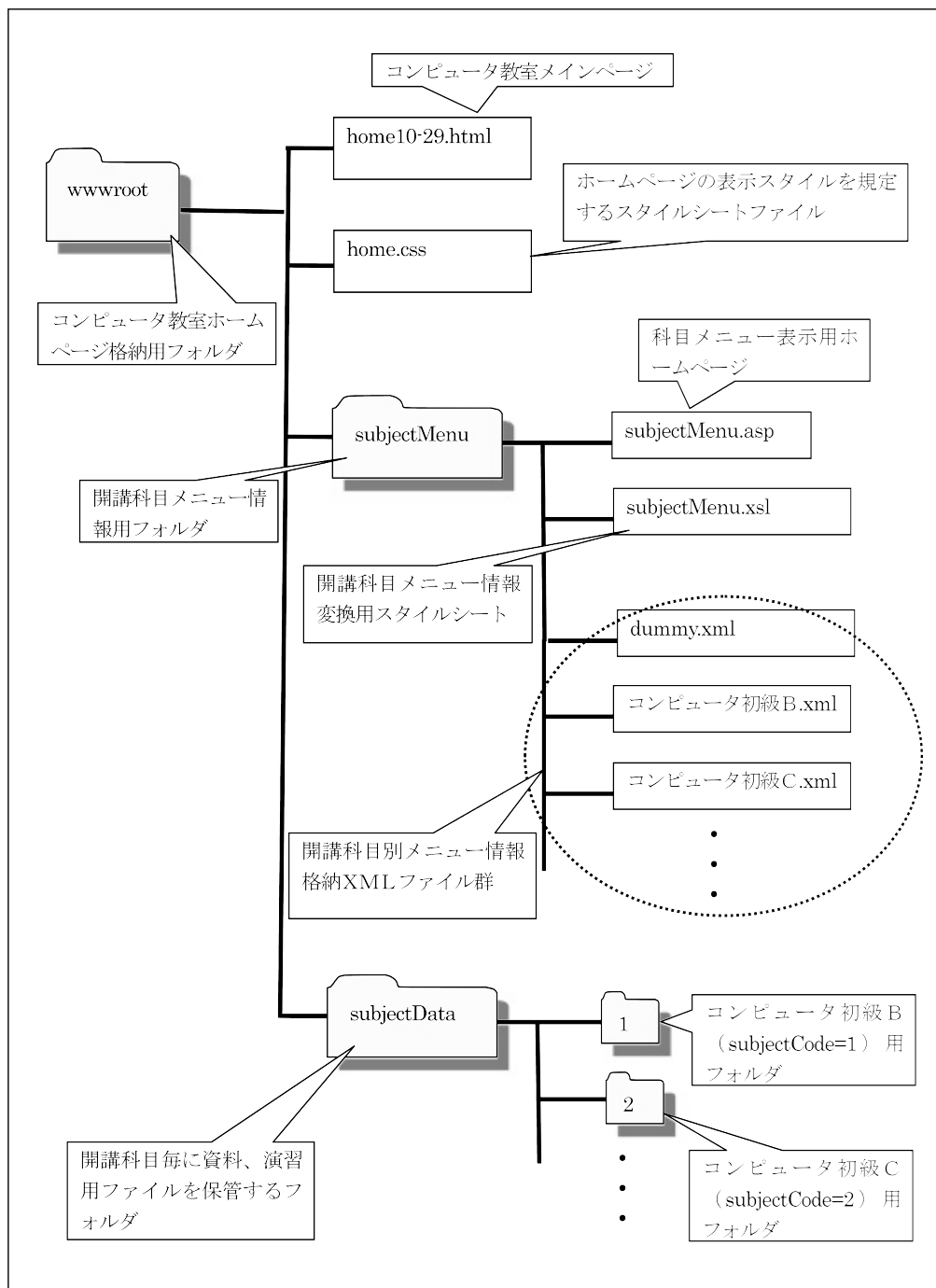
(図3. コンピュータ教室メインホームページ)

コンピュータ初級Bメニュー				
回数	内容	資料	演習用ファイル	注意
第1章	Excelの特長・データ入力	第1章ワークブック資料 b1.doc	なし。自給のExcelシートから演習する	Excelのファイルは、“第1章”という名前で、各自のフォルダに保存して下さい。
第2章	コピー・移動・削除・行ノ列ノシートの操作	第2章ワークブック資料 b2.doc	演習用ファイルb2.xlsを開き、それを使って演習する	“b2.xls”を開き演習し、終了後は各自のフォルダに“第2章.xls”という名前で保存して下さい。
第3章	計算式を含む表作成	第3章ワークブック資料 b3.doc	なし。自給のExcelシートから演習する	Excelのファイルは、“第3章”という名前で、各自のフォルダに保存して下さい。
第4章	関数を含む表作成(合計、平均、最大、最小、標準偏差関数および偏差値の計算)	第4章ワークブック資料 b4.doc	演習用ファイルb4.xlsを開き、それを使って演習する	“b4.xls”を開き演習し、終了後は各自のフォルダに“第4章.xls”という名前で保存して下さい。
第5章	関数を含む表作成(斜り挿入、西暦五入)	第5章ワークブック資料 b5.doc	なし。自給のExcelシートから演習する	Excelのファイルは、“第5章”という名前で、各自のフォルダに保存して下さい。
第6章	関数を含む表作成(IF、OR、AND関数)	第6章ワークブック資料 b6.doc	なし。自給のExcelシートから演習する	Excelのファイルは、“第6章”という名前で、各自のフォルダに保存して下さい。
第7章	グラフ作成(棒グラフの作成とグラフの書式設定)	第7章ワークブック資料 b7.doc	白紙のExcelシートから演習する。第3章のシートを、第7章のファイルにコピーして演習する。	Excelのファイルは、“第7章”という名前で、各自のフォルダに保存して下さい。
第8章	グラフ作成(円グラフ)	第8章ワークブック資料 b8.doc	白紙のExcelシートから演習する。第3章のシートを、第8章のファイルにコピーして演習する。	Excelのファイルは、“第8章”という名前で、各自のフォルダに保存して下さい。
第9章	グラフ作成(レーダーチャート)	第9章ワークブック資料 b9.doc	白紙のExcelシートから演習する。一部第3章のシートを、第9章のファイルにコピーして演習する。	Excelのファイルは、“第9章”という名前で、各自のフォルダに保存して下さい。
第10章	複合グラフ作成	第10章ワークブック資料 b10.doc	白紙のExcelシートから演習する。一部第3章のシートを、第10章のファイルにコピーして演習する。	Excelのファイルは、“第10章”という名前で、各自のフォルダに保存して下さい。

(図4. コンピュータ教室メインホームページから呼び出される開講科目メニューページ)

(表3. ホームページのソーステキスト)

1	<td bgcolor="#ffc0ff">
2	・<a href="subjectMenu/subjectMenu.asp?subjectCode=1">コンピュータ初級Bメニュー</a></BR>
3	・<a href="subjectMenu/subjectMenu.asp?subjectCode=2">コンピュータ初級Cメニュー</a></BR>
4	・<a href="literacy/color/redColorBook.asp">色見本</a></BR>
5	・<a href="literacy/image/L5_1_3_2.asp">背景など</a></BR>
6	・<a href="literacy/guest/L5_1_3_2.asp">年賀画像</a></BR>
7	・<a href="literacy/guest 1/L5_1_3_2.asp">ゲスト</a></BR>
8	</td>



(図5. 開講科目メニュー画面に関するディレクトリ・ファイル構成図)

ータを格納する XML ファイルの集まりのことである。例として「コンピュータ初級 B」科目のメニューデータを格納する XML ファイルを（表 4.）に掲載した。この XML データファイルは、サーバ内にファイル名“コンピュータ初級 B. xml”として保存した。科目情報を格納する XML ファイルのタグについては、（表 5.）に説明を記述した。

（図 5.）に開講科目メニュー画面に関するディレクトリ・ファイル構成図を記載した。教室のサーバ内に subjectMenu という名称のフォルダを作成し、その中に必要なすべての科目のメニューデータを格納する XML データファイルを保存した。

1 本の XML データファイルに必要な全ての科目データを格納するのは、運用が難しいと思われる。各科目につき 1 本の XML データファイルが対応する形態ならば、メンテナンスも容易である。また XSLT スタイルシートにより、科目コードからデータ抽出が可能であり、XML データファイルのファイル名はどんな名前であってもかまわない。この点も運用が簡単になるポイントである。

技術的には、およそ 2 つのポイントがある。1 つは、複数 XML ファイルのデータを一旦メモリ上で融合することと、もう 1 つは、メモリ上で融合された XML データから必要なデータのみ抽出することである。この 2 つがサーバ側

（表 4．科目別メニュー情報格納用 XML ファイルの 1 例）

1	<?xml version="1.0" encoding="Shift_JIS"?>
2	<menu>
3	<subject>
4	<subjectCode>1</subjectCode>
5	<subjectName>コンピュータ初級 B</subjectName>
6	<chapter>
7	<no>1</no>
8	<title>第 1 章</title>
9	<contents>Excel の特徴・データ入力</contents>
10	<data>第 1 章ワープロ資料 <a href="../subjectData/1/b1.doc">b1.doc</a></data>
11	<files>なし。白紙の Excel シートから演習する</files>
12	<cautions>Excel のファイルは、"第 1 章" という名前で、各自のフォルダに保存して下さい。
13	</cautions>
14	</chapter>
15	<chapter>
16	<no>2</no>
17	<title>第 2 章</title>
18	<contents>コピー・移動・削除・行／列／シートの操作</contents>
19	<data>第 2 章ワープロ資料 <a href="../subjectData/1/b2.doc">b2.doc</a></data>
20	<files>演習用ファイル<a href="../subjectData/1/b2.xls">b2.xls</a>を開き、それを使って演習する
21	</files>
22	<cautions>"b2.xls"を開き演習し、終了後は各自のフォルダに"第 2 章.xls"という名前で保存して下さい。
23	</cautions>
24	</chapter>
25	.
26	.
27	.（途中省略）
28	.
29	.
30	</subject>
31	</menu>

(表 5. 科目情報格納 XML ファイルタグの説明)

<menu>	XML ファイル全体のルートとなるタグ
<subject>	1 科目の科目情報を格納する。子供のタグとして <subjectCode>、<subjectName>、<chapter> を含む
<subjectCode>	科目コードを格納する。
<subjectName>	科目名を格納する。
<chapter>	各章の、または 1 回の授業分の情報を格納する。子供のタグとして <no>、<title>、<contents>、<data>、<files>、<cautions> を含む。
<no>	章番号を格納する。
<title>	章のタイトル (名称) を格納する。
<contents>	章の内容についての情報を格納する。
<data>	章において使用する資料の情報を格納する。
<files>	演習で直接使用する (成果を保持する) ファイルについての情報を格納する。
<cautions>	注意事項テキストを格納する。

のスキプトで実行できればよい。この 2 つの技術については、『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) に記載がある。その記述を参考に作成させて頂いた。

開講科目メニュー画面を表示するホームページは、“subjectMenu.asp” (図 5. 参照) である。

コンピュータ教室メインホームページ (“home10-29.html”) における開講科目メニュー画面 (“subjectMenu.asp”) へのリンク記述部分のソーステキストを (表 3.) に掲載した。(表 3.) 2 行目が、「コンピュータ初級 B」科目のメニュー画面を表示させるためのリンク箇所である。ページ呼び出し時に、科目コードを指定するパラメータ “1” がサーバ側に一緒に送信される。同様に 3 行目が、「コンピュータ初級 C」科目のメニュー画面を表示させるためのリンク箇所である。ページ呼び出し時に、科目コードを指定するパラメータ “2” がサーバ側に一緒に送信される。

(表 4.) は、科目別メニュー情報格納用 XML ファイルの 1 つの例として、コンピュータ初級 B メニューページ (図 4.) を表示するための

XML ファイル (“コンピュータ初級 B.xml”) の一部を示している。各章において使用する資料・演習用ファイルへのリンク情報を格納する <data> タグ内のテキストには、HTML タグ <a> を <data> タグの子要素として記述する必要があった。これに関しては、(表 4.) における例が示す通り、子要素のタグとして HTML タグを記述しても問題はなかった。

(表 4.) における 10 行目の

```
「<data>第 1 章ワープロ資料<a href=”../
subjectData/1/b1.doc”>b1.doc</
a></data>」
```

と記述されている箇所である。

各開講科目で使われる資料・演習用ファイルは、subjectData フォルダ内にまとめて格納した (図 5. 参照)。subjectData フォルダ内にさらに科目毎に独立したフォルダを設けて、科目ごとに必要な資料・演習用ファイルを保管した。例えば「コンピュータ初級 B」科目は、科目コード (subjectCode) は 1 であり、フォルダ名 “1” のフォルダを用意して資料・演習用ファイルを保管した。

### 3-2. 開講科目メニューページソースファイル 概説

開講科目メニュー画面を表示するホームページ “subjectMenu. asp” のソーステキストについては、(表 7.) に掲載した。(表 6.) はソーステキストの要点の解説である。

### 3-3. XSLT スタイルシートソースファイル 概説

クライアントから送信された科目コードに合致するデータを抽出し、開講科目メニュー画面 (図 4.) の体裁の HTML 形式のホームページに変換するための XSLT スタイルシートファイル “subjectMenu. xsl” のソーステキストについては、(表 9.) に掲載した。(表 8.) はソーステキストの要点の解説である。

コンピュータ教室メイン画面 (図 3.) と同一の “home. css” スタイルシート (カスケーディングスタイルシート) を参照させて、表示スタイルを付与している (表10. 参照)。

## 4. 結び

昨年度の研究紀要では、サーバ側から 1 本の

XML ファイルがクライアント側に送信され、クライアントパソコンのメモリ上で、XSLT スタイルシートが適用され HTML 形式のホームページに変換されて表示されるタイプのページを作成した。クライアントパソコンのメモリ上でホームページが構成される状況は、後の DOM (Document Object Model) 関数による操作を難しくした。その難点を解決するために、今回はサーバ側のメモリ上で、XSLT スタイルシートが適用され HTML 形式のホームページに変換されてクライアント側に送信されるタイプのページを作成してみた。クライアント側からサーバ側へデータ抽出情報を送信し、その情報に基づいて XML ファイルから必要なデータを抽出しなければならないので、クライアント側からサーバ側への情報の受け渡し方法に少々悩んだが、完成したものはシンプルなページとなった。ホームページソースファイルは、テキスト量的には昨年度作成したクライアント側で

(表 6. “subjectMenu. asp” ソーステキスト説明)

15行目～ 16行目	クライアント側から送信される科目コードを受信する。科目コードが送信されない場合には、科目コードを 0 (ダミーの科目コード) とみなす。
18行目～ 34行目	カレントディレクトリ (“subjectMenu. asp” ホームページと同一ディレクトリに存在する全ての XML ファイルのファイル名を取得し、XML ファイル内の <menu> タグ内の、<subject> タグ以下の全てのデータを対象として選択するための select 文テキストを作成し、xmlFileName 変数に代入している。 例えば、カレントディレクトリに、“コンピュータ初級 B. xml” と “コンピュータ初級 C. xml” の 2 つの XML ファイルが存在する場合には、xmlFileName 変数には、 “document (‘コンピュータ初級 B. xml’)/menu/subject”   document (‘コンピュータ初級 C. xml’)/menu/subject” というテキストが格納されることになる。
36行目～ 49行目	メインとなる XML ファイル “dummy. xml” XML ファイルを読み込んでいる。このファイルは科目コード “0” のダミー科目データである。 後に、読み込み対象となる XML ファイルデータを指定する select 文は、xmlFileName 変数の内容に置き換えられることになる。
41行目～ 43行目	“subjectMenu. xsl” スタイルシートを読み込んでいる。
45行目～ 46行目	“subjectMenu. xsl” スタイルシート内の動的変数 \$subjectCode (表 9. 参照) を、クライアント側から送信された科目コードに置換する。
48行目～ 49行目	“subjectMenu. xsl” スタイルシート内の select 文動的変数 \$xmlFiles (表 9. 参照) を、上記18行目～34行目に生成した xmlFileName 変数のテキストに置換する。
51行目～ 53行目	対象となる XML ファイルデータを、41行目～43行目で読み込んだスタイルシートを適用して HTML ファイル形式に変換している。
55行目	HTML 形式に変換されたテキストを書き込んで、クライアント側に送信するホームページを完成する。

(表 7.1 "subjectMenu.asp" ホームページのソーステキスト)

```

1  <%@Language="JScript"%>
2  <%
3  var check ;
4  var xmlPath=Server.MapPath("./") ;
5  var fso=Server.createObject("Scripting.FileSystemObject") ;
6  var xmlFolder=fso.GetFolder(xmlPath) ;
7  var xmlGet=new Enumerator(xmlFolder.files) ;
8      var subjectCode ;
9      var xmlCount ;
10     var xmlFileName ;
11
12     xmlCount=0 ;
13     xmlFileName="" ;
14
15     subjectCode= Request.QueryString("subjectCode") ;
16     if(subjectCode=="") {subjectCode=0 ; }
17
18 while(!xmlGet.atEnd())
19 {
20     check=xmlGet.item().Name.indexOf(".xml") ;
21     if(check>0)
22     {
23         if(xmlCount==0) {
24             xmlFileName="document('"+xmlGet.item().Name+"')/menu/subject" ;
25         }
26         else {
27             xmlFileName+xmlFileName+" | document('"+xmlGet.item().Name+"')/menu/subject" ;
28         }
29         xmlCount++ ;
30     }
31     xmlGet.moveNext() ;
32 }
33
34     if(xmlFileName=="") {xmlFileName="dummy.xml" ; }
35
36 var filePath=Server.MapPath("./") ;
37     var xmldoc=Server.createObject("MSXML2.DOMDocument") ;
38     xmldoc.async=false ;
39     xmldoc.load(filePath+"/dummy.xml") ;
40
41     var stylesheet=Server.createObject("MSXML2.DOMDocument") ;
42     stylesheet.async=false ;
43     stylesheet.load(filePath+"/subjectMenu.xml") ;
44
45     var variableNode=stylesheet.getElementsByTagName("xsl : variable") ;
46     variableNode(0).text=subjectCode ;
47
48     var variableSelect=stylesheet.getElementsByTagName("xsl : variable/@select") ;
49     variableSelect(0).text+xmlFileName ;
50
51     var htmlData=xmldoc.transformNode(stylesheet) ;
52     var chikan=htmlData.replace("UTF-16","shift_JIS") ;
53     htmlData=chikan ;
54

```

55	Response.write(htmlData) ;
56	
57	%>

(表 8 . “subjectMenu. asp” ソーステキスト説明)

3行目	抽出すべき科目コードを示す動的変数\$subjectCode を定義する。
4行目	扱う対象となるべきノードを示す select 文テキストを格納する動的変数\$xmlFiles を定義する。
9行目～58行目	扱う対象となるべきノードに対して、ノードの数だけ繰り返し処理する xsl : for-each 要素部分。繰り返し処理対象のノードは、4 行目に定義した動的変数\$xmlFiles を使って、select 文で指定する。
10行目～57行目	条件分岐による処理を行う xsl : choose 要素部分。
11行目～56行目	test="subjectCode=\$subjectCode"の条件<subjectCode>タグのテキストが、動的変数\$subjectCode に格納されたテキストと一致するもの) に合致するノードのみ処理する。
30行目～51行目	<chapter>ノードに対して、ノードの数だけ繰り返し処理する xsl : for-each 要素部分。
32行目～40行目	HTML 形式のテーブル行を作成する箇所であるが、奇数行要素の背景色を指定して一行を作成している。ホームページにテーブル構造を作成する場合、一般的に奇数行と偶数行の背景色を変えて、見やすい表にするためのものである。
41行目～49行目	HTML 形式のテーブル行を作成する箇所であるが、偶数行要素の背景色を指定して一行を作成している。

(表 9 . “subjectMenu. xsl” XSLT ファイルのソーステキスト)

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="subjectCode">0</xsl : variable>
4	<xsl : variable name="xmlFiles" select="document('dummy.xml')/menu/subject"/>
5	
6	<xsl : template match="/">
7	
8	<HTML>
9	<xsl : for-each select="\$xmlFiles">
10	<xsl : choose>
11	<xsl : when test="subjectCode = \$subjectCode">
12	
13	<HEAD>
14	<TITLE><xsl : value-of select="subjectName" />メニュー</TITLE>
15	<meta http-equiv="Content-Type" content="text/css"/>
16	<link rel="stylesheet" href="../../home.css" type="text/css" />
17	</HEAD>
18	<BODY>
19	
20	<h1><xsl : value-of select="subjectName" />メニュー</h1>
21	<div class="mainText">
22	<table border="1">
23	<tr align="center" bgcolor="# c0ffff" height="30">
24	<td width="5%">回数</td>
25	<td width="20%">内容</td>
26	<td width="20%">資料</td>
27	<td width="30%">演習用ファイル</td>
28	<td width="25%">注意</td>
29	</tr>
30	<xsl : for-each select="chapter">

```

31      <xsl:choose>
32      <xsl:when test="position() mod 2=1">
33          <tr bgcolor="#ffc0ff">
34              <td><xsl:value-of select="title" /></td>
35              <td><xsl:copy-of select="contents [* | text()]" /></td>
36              <td><xsl:copy-of select="data [* | text()]" /></td>
37              <td><xsl:copy-of select="files [* | text()]" /></td>
38              <td><xsl:copy-of select="cautions [* | text()]" /></td>
39          </tr>
40      </xsl:when>
41      <xsl:otherwise>
42          <tr bgcolor="#ffffa0">
43              <td><xsl:value-of select="title" /></td>
44              <td><xsl:copy-of select="contents [* | text()]" /></td>
45              <td><xsl:copy-of select="data [* | text()]" /></td>
46              <td><xsl:copy-of select="files [* | text()]" /></td>
47              <td><xsl:copy-of select="cautions [* | text()]" /></td>
48          </tr>
49      </xsl:otherwise>
50  </xsl:choose>
51  </xsl:for-each>
52 </table>
53 </div>
54 </BODY>
55 </xsl:when>
56 </xsl:choose>
57 </xsl:for-each>
58 </HTML>
59
60 </xsl:template>
61 </xsl:stylesheet>
62
63
64

```

(表10."home. css" スタイルシートのソーステキスト)

```

body{background-color : # c0ff00 ; }

h1{text-align : center ;
font-size : 350% ;
letter-spacing : 0.2em ;
padding : 0.3em ;
color : white ;
background-color : orange ; }

div. maintext{text-align : center ;
background-color : # c0FF00 ; }

```

XML ファイルから HTML ファイル形式に変換するタイプのページよりも少量となっている。コンピュータ教室のサーバおよび、個人のパソコンに IIS (Internet Information Server) サ

ーバと、ASP (Active Server Pages) をセットアップして、どちらでも簡単に作成したホームページを動作させることができた。さらにによりむずかしい応用として、サーバ側



の複数の XML ファイルを対象として、XSLT スタイルシートを適用して、HTML 形式のホームページに変換してクライアント側に送信するタイプのページも作成してみた。コンピュータ教室で開講する科目のメニュー画面を表示するページである。以前は単純な HTML 形式のページであったが、開講科目データを保持する XML ファイルから自動的に HTML 形式のページが作成されて送信されるタイプのページに作り変えてみた。技術的には『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)に記述されている事例を応用すれば容易に作成することができた。慣れるまでは試行錯誤があり、すぐに動くページを作成できなかったが、完成してみるとコーディングの量的には少量であった。XSLT スタイルシートを複数の XML ファイルを対象に適用するのは簡単に実現できた。

以上ここ 2、3 年は、以下の 3 つのタイプのページ作成について検討してきたことになる。

- ①クライアント側のメモリ上で XML ファイルに XSLT ファイルを適用し、HTML 形式のページを生成するタイプのページ。
- ②サーバ側のメモリ上で XML ファイルに XSLT ファイルを適用し、HTML 形式のページを生成してクライアント側に送信するタイプのページ。(クライアント側からサーバ側へのデータ抽出範囲等のパラメータ送信方法に悩む場合あり)
- ③サーバ側のメモリ上で、複数の XML ファイルに XSLT ファイルを適用し、HTML 形式のページを生成してクライアント側に送信するタイプのページ。

②、③タイプのページは、最初は試行錯誤し時間が掛かったが、完成したページのコーディング量は少量であった。1 回動作するものが作成できれば、ポイントがつかめ、次回からは同様なページを作るのは難しいことではないようだ。結局は複数 XML ファイルを扱うことは、コーディング量的には簡単であった。特別なデータベース (Access 等) にデータを格納して、必要なデータをそのデータベースから取得して HTML 形式のページを作成してクライアントに送信するような面倒なシステムにしなくても、XSLT スタイルシートファイルと、ASP で動作する割合少量のスキプットのコーディングで済ませることができた。

さらに、今回特にコンピュータ教室で使うような事例がなかったが、メモリ上で XSLT スタイルシートを多段階に適用させるようなことも簡単に実現できるようだ。つまり、XML データファイルに XSLT スタイルシートを適用し別の XML データファイルに変換し、さらに続いて別の XSLT スタイルシートを適用して 2 次的な変換を施して最終的な HTML 形式のページを作成するような事例である。これについては、『XML+XSLT 実用スーパーサンプル集』(2002年 CQ 出版株式会社)に事例の説明がある。これなどはサーバ側で実行する特別なスキプット無しで、XSLT スタイルシートの記述のみで実現されていた。

上記①、②、③の事例に加えて、④ XSLT スタイルシートの多段階適用の事例、以上 4 パターンの事例については、特に複雑な状況がなければ、XSLT スタイルシートと ASP で動作する少量のスキプットを記述すれば割合簡単に実現できそうである。

やはり、XML・XSLT 技術は取り付きにくいことは確かであるが、いくつかの基本的なパタ

ーン（事例）に慣れればコーディング量的には少量で動的なホームページを作成することができた。表計算ソフトからは、画像データも含め多系列のデータが生み出されるが、それを最終的に使い勝手良くクライアントに提示するのはそう容易なことでは無い。XML・XSLT 技術を利用して、表計算ソフトが生成した多系列情報をホームページとしてクライアントに提示するというのが1つの割合簡単な方法だと思われる。今回まとめた①、②、③の3パターンのXML・XSLT 技術の利用で、とりあえずはそれが実現可能であろうと思われる。今後、この3つのパターンについては、表計算ソフトとXML・XSLT 技術付近のコンピュータ利用中級程度の技術として、学生に提示することができるかもしれない。

#### 【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) 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
- 3) 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
- 4) PROJECT KySS 著  
『XML+XSLT 実用スーパーサンプル集』2002年 CQ 出版株式会社