Paper# 4009

Developing for the Intranet

How to Use WWW Technology for Just About Everything

by Dr. David Johnson
Johnson Computer Software Team Limited
5072 - 3080 Yonge Street, Toronto, Ontario M4N 3N1
(416) 487-3631

"Client/server as we know it is dead. It is being replaced by this framework"[1]. This is probably on a par with the continued predictions of the death of COBOL but it does point us in a new and interesting direction.

Since IBM "invented" the personal computer, the world of IS has been heavily focused on the desktop. So much so that Microsoft was able to put the world on hold again and again while it struggled to deliver Windows95. Because the world has been so focused on the desktop, we all waited. The crack that Win95 equals Mac89 is funny precisely because it is largely true. There is nothing terribly innovative or new about this "new" operating system.

Now Where Was I ...

Where were we when Microsoft put the world on hold? Focused almost exclusively on the desktop. More and more desktop applications with more and more arcane features described aptly by one scribe as Jaba the Huts squatting on your disc.

What about client/server - isn't that looking beyond the desktop? Yes but the *focus* with most development tools is almost exclusively on the desktop. "Fat" clients do all the work and the server is often not considered more than a large shared disc. Moreover all too often the client development has to be cognizant of the organization of the server data: if you have two different databases on the server then you have two ways of interacting with your server.

Is it any wonder then at the excitement being generated by the World Wide Web? Originally released by CERN for use in 1991 to distribute scientific papers and results, it became a revolution with the release of a graphical browser, Mosaic, two years later.

We have the ultimate Open System - nobody owns this stuff - built on a universal client/server architecture. The need for proprietary e-mail, proprietary work group

[1]Marc Andreesen VP Technology, Netscape

software or proprietary data warehouse systems, etc., begins to fade in the light of the things that can be and are being done with the WWW technology.

The WWW technology provides a wonderful way to organize and retrieve information; there is a beautiful simplicity to it that gives it elegance, that allows you to think of all of the information out there available to you as one vast hyperlinked document. Or think of the Web as the planetary disc drive. This can be applied to the "intranet" as well: use the WWW technology to handle corporate information for internal use. Being open systems technology it runs on the HP3000 just as well as UNIX servers

The more we explored the concepts and the technology, the more we came to realize, as expressed by Chris Cobb, "There are times when the 'cosmic tumblers' fall into place and the resulting harmonic convergence changes the world overnight"[2].

We realized the real power of the Web concept comes not from having a Web server with information stored in html pages but from having a server with application specific databases and a way of allowing the user to interact with these databases by having html pages constructed 'on the fly'. After all, a basic interactive application allows the user to:

> get information into the system either by entering it or automatically from another system
> store and manipulate the information
> retrieve and process the information.

That is, get it in, store it and get it out. It appeared that these could all be done quite simply with the Web technology so we set out to see if we could actually build real applications using it.

The First Application: Replacing Reporting with Publishing

We chose the path of least resistance: we tried the simplest thing first. We have a major advertising package that runs on the HP3000 and dates back to the early 1980s and still runs mostly on character based screens. The reporting requirements are significant, a large advertising agency can produce dozens of reports a night, each resulting in hundreds of pages. The first prototype was to produce on of these reports as a set of linked html pages. All we did was to adapt an existing report to create files that are html pages instead of printing the output and inserted what we thought were

[2]Weaving a World Wide Web: An Overview of Hewlett-Packard's Web Strategy - hp-ux/usr January 1996

appropriate hyperlinks to aid user navigation. This has a number of immediate advantages over producing the paper:

> Speed: it takes a long time to print and deliver a 700 page report, particularly if there are a large number to produce. Eliminating the paper consumption means the report is available to the user the instant it is done, usually by the time the user comes into work in the morning.
> Accessibility: by inserting hyperlinks into the report, the user can navigate to the desired sections instantly. Flipping through a 700 page report is tedious, pointing and clicking is much faster and more accurate. And most browsers have a Find command to let you hunt for specific text strings in the loaded page.
> Flexibility: users do not have to be in the office to access their information. Remote users simply need an internet connection.
> Client accessiblity: with a little work in designing security, this approach solves a problem we have faced since we first started developing on-line systems 20 years ago: clients wanting on-line access to their advertising campaigns. For both technical and political reasons, allowing direct access to the information in the application database has never been a viable option. Now users can "publish" their client's information onto their Web server and allow the client to access this snapshot with a Web browser over the internet.

One of the most interesting findings was that there is *not* a lot of work involved to get your first Intranet application up and running.

The Poor Mans Data Warehouse

The next project dealt with involved a complex corporate hierarchy. An attempt to report this hierarchy on paper, in complete detail, foundered on volume issues. When we tried to create a report that showed the hierarchy at every stage there was so much repetition of certain pieces at different levels we were attempting to create a report with over half a million records. What we did instead was to present the information to the users as a set of linked Web pages eliminating all 'repetition'. This is the beginnings of a data warehousing: fetch existing data, organizing it in 'layers' to create information and allowing the user to drill down from the top to increasing layers of detail. The final stage is to allow users to do their own manipulation of the data. How do you do this with "published" pages? Very simply, at the bottom of your data hierarchy, create csv (comma separated value) files instead of (or as well as) html pages. The users' browser can be configured to load Excel or Lotus in response to receiving the file.

Since the results of the publishing are a set of html pages, links can be added or modified by hand as required.

The result? A very fast, very flexible and quite functional data analysis tool for management at a very low cost.

Data Analysis

One facility in our advertising system is to provide access to audience survey data. The rating information is provided by the organizations that perform the actual measurements and make the resulting information available on tape. We load the data into a database on the server to be used for post-analysis or after-the-fact reporting. But the real use of the information is in the buying process where the users generally resort to looking up the numbers in published books. There are ways to access the data electronically but this generally involves either getting it on a CD-ROM and acquiring special purpose custom software or running on terminals or terminal emulation software accessing the database on the server. Developing a custom client/server special purpose application was simply too expensive to justify.

Using the Web technology, we developed the application in only a few days. Up and running involved creating 5 html templates (3 of which are shown in figures 5, 6 and 7) and three scripts or programs to be run by the Web server. The scripts are written in C and vary from 400 to 1100 lines of C code.

Here's how it works: the user activates a link from another, permanent or static page that launches the first of the three scripts. This first script opens the audience information database, finds out what demographics, surveys and markets are available. With this information it reads the appropriate html templates and *constructs* the following page for the user:

figure 1: **selecting the search parameters**

The user selects the desired demographic group, e.g. Women 18-49, the survey and the market they are after and press the button "Get Programs". This launches the second script which receives the chosen parameters as input. This script goes to the database and retrieves all the programs for the selected market, and it too reads the appropriate html templates and *constructs* the following page for the user:***figure 2:*** **choose program or day and time**

The user requests either a program name or a time period and clicks the button "Get Audience" that launches the third script. This script dutifully returns to the database and gets the requested audience numbers for the requested program or time period:

figure 3: **requested audience information**

This page is presented as a separate browser window leaving the previous page up in the background to be used for subsequent searches.

There is actually a little more happening 'under the hood' in terms of automatically developing numbers for non-standard demographics and averaging multiple weeks (things a user simply would never have time to do manually), some security features, etc. but the essence of the application is very simple and easy to create.

Some interesting notes are:

> The system is very "modular": scripts are quite independent of each other making it easy to distribute development. In fact there is nothing to say the scripts have to be written in the same language.
>
> It is important to separate the scripts that generate the information and the presentation of the results to the users. We set it up so that the scripts use template pages to create the pages for the user which means the layout of the page and even whether the final screen of audience information is presented as an html page or as a csv file to launch the user's favourite spreadsheet, is a matter of a few keystrokes editing a template file rather than going back to the programmers to make changes.
>
> The Web technology provides the developer with the universal client/server application. This audience lookup will run in-house if the agency has their server set up as a Web server. It will run over the internet with the user in one city and the server in another. It doesn't matter if the user is in the office or at home so long as internet access is available. Last but not least it doesn't matter if the user has a PC or a MAC.

A Complete Application

So far we have only discussed applications where the data is already in the data base and all we are doing is extracting it. But what about using this universal client/server to enter data? The browsers allow user input, of course, to get parameters and fill in forms to capture information. Using this feature, it is easy to create dynamic input forms to allow the user to upload files from another system and to capture input data.

In this application, most of the input is done via an electronic transfer, the main data entry function is to upload a file. The html page that does this uses form input:

```
<FORM ENCTYPE="multipart/form-data" ACTION="/cgi-bin/update.nmp"
METHOD=POST>
Send this file: <INPUT NAME="userfile" TYPE="file"><BR><BR>
 <INPUT TYPE="submit" VALUE="Send File">
To clear the form, press this button: <INPUT TYPE="reset" VALUE="Clear
Form">
<hr>

</FORM>
```

But the users do have to add some additional data. For example, the users need to enter a foreign exchange rate on a daily basis. They do this by accessing a page to choose the month and the script executed in response pulls the existing entries, displays them, as shown in figure 4. (they can be changed at this point) and allows the user to add additional rates.

***figure 4:* input/edit exchange rates**

With all the information in place, requesting the 'reports' is a simple process of accessing the "request" page, choosing the type of information required and receiving an html page in the browser or a .CSV file into a spreadsheet. Once again, a simple, flexible and functional application at a very low cost.

How to

To arrive at the functionality demonstrated in figure 1, requires a script (approximately 400 lines of C code) and 3 html templates (and a "blank" html page). The templates are shown in figures 5, 6 and 7.

```
<HTML><HEAD><TITLE>BBM Lookup Request</TITLE></HEAD>
<FRAMESET ROWS="37%,63%">
  <FRAME SRC="/templates/bbmhead.html"  TARGET="access" SCROLLING="no">
  <FRAMESET COLS="50%,50%">
  <FRAME SRC="/options.html" SCROLLING="yes" NAME="client" TARGET="search">
    <FRAME  SRC="/blank.html" SCROLLING="yes" NAME="search"
TARGET="lookup">
  </FRAMESET>
  <!FRAME SRC="/blank.html" SCROLLING="yes" NAME="lookup">
</FRAMESET>
</HTML>
```

***figure 5:* bbmmast.html**

```
<HTML><HEAD><TITLE>BBM Television Access</TITLE></HEAD>
<BODY><H1><IMG SRC="/images/jcst.jpg"></IMG>
BBM Television Access
</H1>
```

***figure 6:* bbmhead.html**

```
<HTML><HEAD><TITLE>Select Client</TITLE></HEAD>
<BASE TARGET="search">
<BODY>
<IMG SRC="/images/jcst.jpg"></IMG>
<FORM ACTION="/cgi-bin/search.nmprg" METHOD="GET">
<TABLE>
<TD ALIGN="LEFT"><STRONG>Demo:</STRONG>
<TD><SELECT NAME="demo#">
<![+DEMOS]>
<OPTION><![DEMO]></OPTION>
<![-DEMOS]>
</SELECT>
<TD><TR>
<TD ALIGN="LEFT"><STRONG>B</STRONG>ook:
<TD><SELECT NAME="start#">
<![+SURVEYS]>
<OPTION><![SURVEY]></OPTION>
<![-SURVEYS]>
</SELECT>
<TD> for: <INPUT TYPE="text" NAME="weeks" SIZE=2> weeks
```

```
<TR><TD><STRONG>M</STRONG>arket:
<TD><SELECT NAME="market">
<![+MKTS]>
<OPTION><![MARKET]></OPTION>
<![-MKTS]>
</SELECT>
<TR><TD ALIGN="RIGHT"><I>and press</I>
<TD><INPUT TYPE="SUBMIT" VALUE="Get Programs" ALIGN="MIDDLE">
</TABLE></FORM>
</BODY></HTML>
```

figure 7: **options.html**

Users may access the system from one of several pages. The link they follow launches the first script. This program is launched by the http daemon and gets its input in the QUERY_STRING environment variable which contains the information passed from the Web browser to the http daemon.

The basic logic is quite simple:

```
get the input parameters
open the audience information database
open the appropriate template file and html file
get the requested information and 'merge' the information from the database with
the template and return it (to standard list) to the http daemon.
```

Some actual code segments are shown below:

```
****************************INPUT PARAMETERS***************************
  /* see if a directory specified in QUERY_STRING environment variable */
  /* set the fully qualified name of the options.html file */

  strcpy(hfname,"/WWW/WWW/ARPA/httpd_1.3/htdocs"); /* document root */

  pc=getenv("QUERY_STRING");
  if (pc != NULL) {
    /* get the directory name DIR=dirname */
    pl=strstr(pc,"DIR=");
    if (pl != NULL) {
      pr=strchr(pl,'&');
      if (pr != NULL)
        *pr=0;
      if (pl[4] != '/')
        strcat(hfname,"/");
      strcat(hfname,pl+4);
      }
    }

  if (hfname[strlen(hfname)-1] != '/')
    strcat(hfname,"/");
  strcat(hfname,"options.html");  /* fully qualified name of options.html
*/
**********************************************************************
```

```
*****************OPEN THE AUDIENCE INFORMATION DATA BASE**************
  /* open the BBM data base */
  strcpy((char *)base,"  BBM.BBMTV.BBM;");
  mode=5;
  DBOPEN(base,"READONLY;",&mode,&stat);
  if (stat.cc)return 0; /* data base error */
***********************************************************************


***********************OPEN THE APPROPRIATE TEMPLATE FILE**************
/************************AND THE HTML FILE****************************
  /* open the OPTIONS template file */
  strcpy(tfname,"/WWW/PUB/OPTIONS");
  tfile=open(tfname,O_RDONLY);
  if (tfile == -1)return 0;  /* error opening file */

  /* create and open the options.html file */
  umask(0);
  hfile=open(hfname,O_WRONLY|O_CREAT|O_TRUNC,
            S_IRWXU|S_IRWXG|S_IRWXO);
  if (hfile == -1)return 0;  /* error opening file */
***********************************************************************


*****************GET THE REQUESTED INFORMATION FROM DATABASE***********
***********************AND MERGE IT WITH TEMPLATE**********************
  /* read template file, find keywords */
  /* get data for keywords from data base */
  /* write recs to html file */

  while ((tlen=read(tfile,trec,MAXREC)) != -1) {
    /* read template file */
    if (!tlen)break; /* end of file */
    trec[tlen]=0;

    /* find the keyword enclosed in square brakets */
    pl=strchr(trec,'[');
    if (pl != NULL)
      pr=strchr(pl,']');
    else pr=NULL;

    if (pl != NULL && pr != NULL) {
      /* keyword found */
      /* get data for the keyword */

      if (strstr(pl,"[DEMO]") != NULL) {
        /* get the demo */
        DBGET(base,demo_set,&mode,&stat,demo_list,dvalue,dvalue);
        if (stat.cc)return 0; /*data base error */
        }

      else
      if (strstr(pl,"[SURVEY]") != NULL) {
        /* get the survey */
        DBGET(base,survey_set,&mode,&stat,survey_list,dvalue,dvalue);
        if (stat.cc)return 0; /*data base error */
        }

      else
      if (strstr(pl,"[MARKET]") != NULL) {
        /* get the market */
        DBGET(base,market_set,&mode,&stat,market_list,dvalue,dvalue);
        if (stat.cc)return 0; /*data base error */
        }

      else
        pr=NULL; /* unrecognized keyword */
      }

    if (pl == NULL || pr == NULL) {
      /* just a rec with no keyword or unrecognized keyword, */
      /* nothing to replace, just write rec to html file */
      if (write(hfile,trec,strlen(trec)) == -1)
        return 0;
      }

    else {
      /* got a keyword, write rec to html file, insert keyword value */
      strcpy(szkey,(char *)dvalue);
      if (write(hfile,trec,pl-1-trec) == -1 ||
          write(hfile,szkey,strlen(szkey)) == -1 ||
          write(hfile,pr+1,strlen(pr)-1) == -1)
        return 0;
      }
    }
***********************************************************************


******************RETURN INFORMATION TO THE HTTP DAEMON****************
  /* open the BBMMAST template file */
  /* read it and write it to STDLIST */
  strcpy(tfname,"/WWW/PUB/BBMMAST");
  tfile=open(tfname,O_RDONLY);
  if (tfile == -1)return 0; /* error opening file */

  while ((tlen=read(tfile,trec,MAXREC)) != -1) {
    if (!tlen)break;
    trec[tlen]=0;
    printf("%s",trec);
    }
***********************************************************************
```

Where Do We Go From Here

We have found a number of advantages to this approach: the flexibility, the modularity, platform independence and the enormous popularity of the Web in general make this a very enticing path to explore for application development.

Underlying this is the assumption that users are or will quickly become used to using their browser as the normal means for finding and viewing information. By using the browser as the standard interface, user training requirements are minimized as are client (PC) software costs. Another bonus is that the client is platform independent: we don't know or care if the user has a PC or a Mac on their desktop.

But the applications discussed here are all quite simple - what about complex applications requiring more user interaction? This approach eschews the "fat client" architecture that is representative of current client/server development and replaces it with a decidedly anorexic one. It has been referred to as a "3270 terminal on steroids" because the mode of operation is to fill in the blanks and press the button. The Web is stateless and, for many business applications, we need to establish more than a transitory link between the client and the server. The solution we are exploring is to add a development tool such as Sun's Java to fatten up the client to a healthier looking state. With this in place, we expect to see a new generation of applications that are more fun, more functional, more robust and are easier, cheaper and faster to develop.