Developing Dynamic Web Pages: Part 2

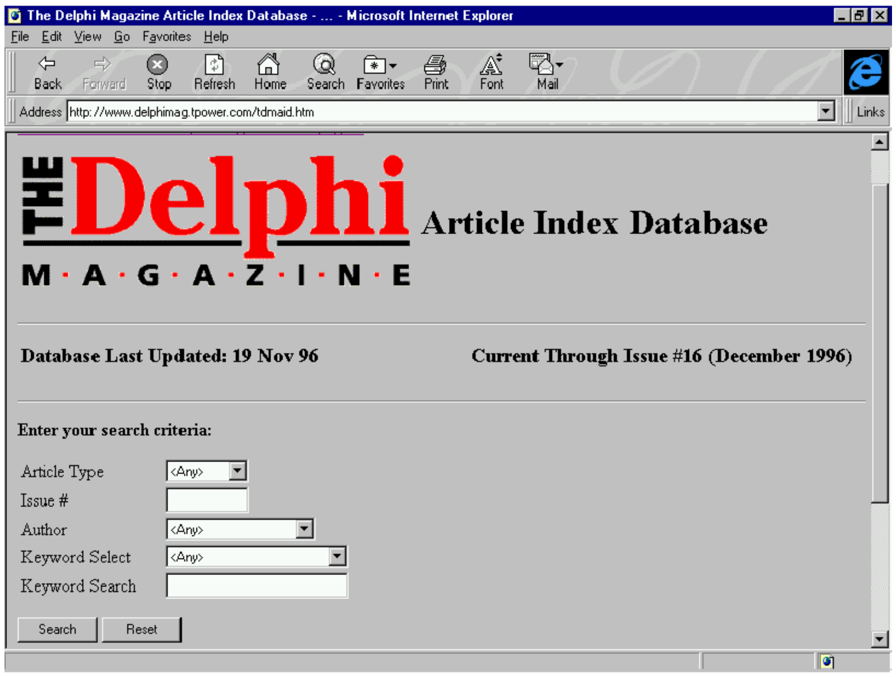
by Steve Troxell

Back in the November issue, Bob Swart got us rolling with building Internet CGI applications using Delphi 2. Last month, we developed a rudimentary `TCGI` component to take care of the nitty-gritty details of acquiring user-supplied data from the web browser through either standard CGI or WinCGI. With this foundation in place, this month I'd like to take you through the development of the web-version of the BDE-aware Article Index Database (TDMAid) for The Delphi Magazine, which Bob first revealed back in July (Issue #11). Along the way we'll get a chance to see `TCGI` in action. I'll try to minimize covering the same ground that Bob did back in November. Our focus here is on the specifics of TDMAid, in particular accessing BDE databases on a web server. The TDMAid Online application can be seen at

http://members.aol.com/delphimag

TDMAid Online Architecture

In designing TDMAid Online we must keep two things in mind. First, the user's experience with the behavior and features of the site should be substantially similar to the offline version, while still exploiting the conventions and capabilities of the web arena. Second, since the TDMAid database is updated regularly each month, maintenance of the site should be as painless as possible. With these two objectives in mind, TDMAid Online's query page looks and acts virtually the same as the offline version's query dialog (see Figure 1). Also, the two versions use exactly the same database (a set of Paradox tables) so that when updates are made, there is no conversion of data necessary – the new database files are simply copied to the appropriate location on the web server.


The Delphi Magazine Article Index Database - ... - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print Font Mail
Address http://www.delphimag.tpower.com/tdmaid.htm
Links
THE Delphi MAGAZINE
Article Index Database

Database Last Updated: 19 Nov 96

Current Through Issue #16 (December 1996)

Enter your search criteria:

Article Type	<Any>
Issue #	
Author	<Any>
Keyword Select	<Any>
Keyword Search	

➤ *Figure 1*

The Query Page

TDMAid Online, like most CGI applications, is a two-stage process: a query page where the user defines their criteria, and a response page where the matching results are presented. On the query page, the user may select articles by type, author, issue or keyword. The article type, author and keyword select controls are listboxes populated from the database.

My first inclination was for this page to build itself through a CGI program, filling its listboxes from the database each time. This would have reduced updating the site each month to a single step: copy new database files. However, the data is really not volatile enough to warrant a CGI application, which would have slowed down the page's load time somewhat. Since the database remains static for about a month at a stretch, we'll just regenerate a static web page once when we update the database. An abridged version of the HTML code for this page is shown in Listing 1.

From Listing 1, you can see that the `FORM` statement defines a program called TDMAID2.EXE in the same directory as this HTML page to be the CGI program to process this request when the user clicks the `Search` button. It is important to remember that execute permissions must be granted to the directory containing the CGI program.

To produce the query page, we can write a simple non-CGI Delphi application (called TDMAID1) that outputs a regular text file containing the HTML code for our query page. Since no CGI access is required for this stage, and Bob already covered the details of constructing HTML query pages, I won't dwell on the TDMAID1 program (but you can examine it on this issue's disk). The important thing to note is that when the user clicks `Search`, the contents of the page's data controls are posted as

CGI variables to the TDMAID2 program we're about to get into.

The CGI App

TDMAID2 is a simple console app divided into three main units. MAIN2.PAS processes the input through the `TCGI` component and produces the output response page, AIDDATA.PAS is a data module containing the TQuery components we use to access the database, and BLDSQL.PAS contains a procedure to construct the SQL query which finds the desired articles. With the exceptions that we are using `TCGI` to read our input values and that we are creating an HTML page as output, there is nothing out of the ordinary about this code. We simply access the data components as we would with any other Delphi application.

MAIN2 contains a procedure called `Process` which is called from the .DPR file as soon as the program begins executing (see Listing 2). Like the offline version, TDMAid Online uses the values entered by the user to build an SQL statement on the fly which finds the matching articles in the database. The first thing the `Process` procedure does is extract values from the form variables passed in from the browser and passes them off to the query building routine. Remember that our `TCGI` component automatically reads in the CGI environment variables as soon as the program starts up, so they are available to us in the `CGI.FormItems` string list. That's all that's needed to process the CGI input. Then we pass the form values into the `BuildSQL` method in the BLDSQL unit, which will populate the SQL property of the `ArticleQuery` component.

I won't show the `BuildSQL` procedure because it is substantially identical to the code used in the offline version Bob wrote about originally in last July's issue. Given the data values passed into it, and assuming that any values that are not to be used in the search are empty strings, `BuildSQL` simply constructs the appropriate SQL statement to fetch the matching articles out of the database. Since the query building code is fairly extensive and is identical in both the offline and online versions, we can compile the `BldSQL` unit into both versions and share the code between them.

Once the SQL statement is built, we can run the query and use its output to build our HTML response page (Listing 2). As you can see, we simply read the values from the query component and use them in constructing our HTML output for the response page (see Figure 2).

```
<HTML><HEAD> <TITLE>The Delphi Magazine Article Index Database - Search
Form</TITLE> </HEAD><BODY> <BR>
  <A HREF="http://members.aol.com/DelphiMag/index.htm">
  Click to return to The Delphi Magazine main page...</A>
<P><IMG SRC="http://members.aol.com/DelphiMag/dmlogo2a.gif"
  ALT="The Delphi Magazine" ALIGN=CENTER>
<FONT SIZE=+3><STRONG>Article Index Database</STRONG></FONT></P>
<HR>
<TABLE BORDER=0 WIDTH="100%" HEIGHT="10">
<TR><TD>
<H3>Database Last Updated: 19 Nov 96</H3>
</TD>
<TD ALIGN=RIGHT>
<H3>Current Through Issue #16  (December 1996)</H3>
</TD></TR>
</TABLE>
<HR>
<STRONG>Enter your search criteria:</STRONG>
<BR>
<FORM METHOD="POST" ACTION="tdmaid2.exe">
<BR>
<TABLE BORDER=0 WIDTH="300" HEIGHT="60">
<TR><TD WIDTH="35%">Article Type</TD>
<TD><SELECT NAME="ArticleType" SIZE=1>
  <OPTION>&ltAny&gt</OPTION>
  <OPTION>Article</OPTION>
  <OPTION>Review</OPTION>
  <OPTION>Tip</OPTION>
  <OPTION>Clinic</OPTION>
  <OPTION>Disk</OPTION>
  <OPTION>Misc</OPTION>
</SELECT>
</TD></TR>
<TR><TD WIDTH="35%">Issue #</TD>
<TD><INPUT TYPE="TEXT" NAME="IssueNo" SIZE=11></TD></TR>
<TR><TD WIDTH="35%">Author</TD>
<TD><SELECT NAME="Author" SIZE=1>
  <OPTION>&ltAny&gt</OPTION>
  <OPTION VALUE="58">Allen Bauer</OPTION>
  <OPTION VALUE="62">Andrew McLellan</OPTION>
  (. . . excess items omitted . . .)
</SELECT>
</TD></TR>
<TR><TD WIDTH="35%">Keyword Select</TD>
<TD><SELECT NAME="KeywordSelect" SIZE=1>
  <OPTION>&ltAny&gt</OPTION>
  <OPTION>$APPTYPE CONSOLE</OPTION>
  <OPTION>$C compiler directive</OPTION>
  <OPTION>$I compiler directive</OPTION>
  <OPTION>$M+ compiler directive</OPTION>
  (. . . excess items omitted . . .)
</SELECT><BR>
</TD></TR>
<TR><TD WIDTH="35%">Keyword Search</TD>
<TD><INPUT TYPE="TEXT" NAME="KeywordSearch" SIZE=29></TD></TR>
</TABLE>
<BR>
<INPUT TYPE="SUBMIT" NAME="SubmitBtn" VALUE="Search">
<INPUT TYPE="RESET" NAME="ResetBtn" VALUE="Reset">
</FORM>
</BODY></HTML>
```

➤ *Listing 1*

New Keyword Search Feature

Since you first heard about TDMAid in July, Bob has been busy making improvements. The most significant feature is the addition of a keyword search function. Previously, you could only search for articles by keyword by selecting a phrase from a list of all possible keywords. The editorial staff have been very busy revamping the keywords assigned in the database. The result of this is that the keywords are more extensive, more detailed and more useful for searching. The downside is that there are a lot of them! There are nearly 2000 unique keywords at the time of this writing.

To make keywords a little easier to work with, the `Keyword Search` function was added (the original keyword mechanism is still available and is called `Keyword Select`). Here you simply type in freeform text, which is then used as a substring search against the keyword table. Therefore, if you enter `DLL`, you'll hit articles with the keywords `Writing DLLs` and `Creating DLLs`. To accomplish this, we use the `LIKE` operator in SQL, as shown in Listing 3.

A little quirk in `LIKE` you should be aware of: `LIKE` for Local SQL is case-insensitive in the 16-bit BDE, but is case-sensitive in the 32-bit BDE. To be assured of case-insensitivity, we always compare uppercase data to uppercase text.

Console Apps And Exceptions

CGI applications should be compiled as console apps (that's why we have `$APPTYPE CONSOLE` at the beginning of our program). This is because there should be no interaction between the program and the server machine which would result in a window or dialog that requires user intervention. When this happens to an Internet user running your CGI app, their browser appears to hang up since it has sent a request to run the CGI program and is waiting for a response HTML page to come back from it. The CGI program keeps running until someone clears the dialog on the server machine.

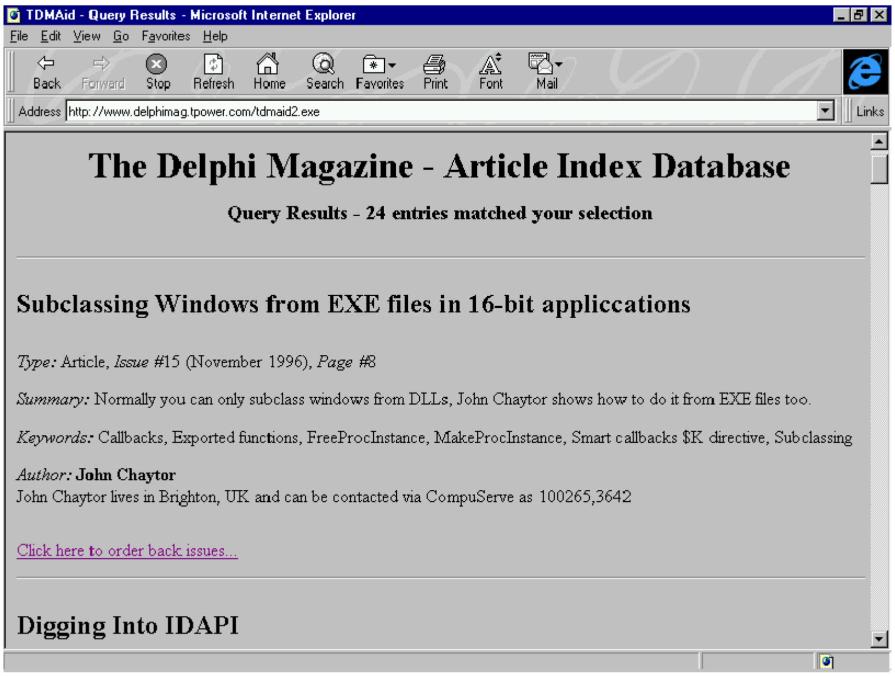
When a console program generates a runtime error or raises an exception, Delphi's error-handling mechanism displays the error messages as regular text output on the standard output device. If you look at the exception handling source code for `SysUtils`, you'll see that it examines the system variable `IsConsole` to decide how to display the exception message. Unfortunately, there is a bug in Delphi in that it does not set the `IsConsole` variable until after all unit initialization code has been executed. Therefore, if an exception occurs in this area in your console app, Delphi displays the error message

➤ *Listing 3*

```
SELECT DISTINCT Article.*
  FROM Article, Keyword
  WHERE Article."Article ID" = Keyword."Article ID")
    AND (UPPER(Keyword."Keyword") LIKE “%DLL%”)
  ORDER BY Article."Issue #", ARTICLE."Page Number"
```

➤ *Listing 2*

```
procedure Process;
var A, I: Integer;
    FieldName: string;
    ArticleType: string;
    AuthorName: string;
    IssueNo: string;
    KeywordSelect: string;
    KeywordSearch: string;
    NoneFOund: Boolean;
begin
  { Extract form variables and hand off to query builder }
  with CGI.FormItems do begin
    ArticleType := Values['ArticleType'];
    if ArticleType = '<Any>' then ArticleType := '';
    AuthorName := Values['Author'];
    if AuthorName = '<Any>' then AuthorName := '';
    IssueNo := Values['IssueNo'];
    KeywordSelect := Values['KeywordSelect'];
    if KeywordSelect = '<Any>' then KeywordSelect := '';
    KeywordSearch := Values['KeywordSearch'];
  end;
  BuildSQL(ArticleType, AuthorName, IssueNo, KeywordSelect,
    KeywordSearch, AIDDataModule.ArticleQuery.SQL);
  with CGI do begin
    WriteLn('<HTML><HEAD>');
    WriteLn('<TITLE>TDMAid - Query Results</TITLE>');
    WriteLn('</HEAD><BODY>');
    WriteLn('<BR><A HREF="' + DelphiMagURL+'index.htm">'+
      'Click to return to The Delphi Magazine main '+
      'page...</A><BR>');
    WriteLn('<A HREF="' + PagesURL +
      'tdmaid.htm">Return to Search Form...</A><BR>');
    try
      with AIDDataModule.ArticleQuery do begin
        NoneFound := True;
        Open;
        try
          NoneFound := Eof;
          WriteLn('<CENTER>');
          WriteLn('<H1>The Delphi Magazine - '+
            'Article Index Database</H1>');
          if NoneFound then
            WriteLn('<H3>Query Results - '+
            'No entries matched your selection</H3>')
          else
            WriteLn('<H3>Query Results -' +
              IntToStr(RecordCount)+
              ' entries matched your selection</H3>');
          WriteLn('</CENTER>');
          if not NoneFound then WriteLn('<HR><BR>');
          while not Eof do begin
            { Show Article Information... }
            WriteLn('<H2>' + FieldByName('Title').AsString +
              '</H2><BR>');
            Write('<EM>Type:</EM> ' + FieldByName('
              Article Type').AsString + ', ');
            Write('<EM>Issue #</EM>' +
              IntToStr(FieldByName('Issue #').AsInteger));
            Write(' (' + FieldByName('MonthYear').AsString +
              '), ');
            Write('<EM>Page #</EM>' + IntToStr(FieldByName('
              Page Number').AsInteger) + '<BR>');
            WriteLn('<BR>');
            Write('<EM>Summary:</EM>  ');
            WriteBLOB(TBlobField(FieldByName('Summary')));
            WriteLn('<BR><BR>');
            { Show Keywords... }
            with AIDDataModule.KeywordQuery do begin
              ParamByName('ArticleID').Value :=
                AIDDataModule.ArticleQuery.FieldByName('
                Article ID').Value;
              Open;
              try
                Write('<EM>Keywords:</EM>  ');
                Write(
                  AIDDataModule.KeywordQueryKeyword.Value);
                Next;
                while not EOF do begin
                  Write(', ' +
                    AIDDataModule.KeywordQueryKeyword.Value);
                  Next;
                end;
                WriteLn('<BR><BR>');
              finally
                Close;
              end;
            end;
            { Show Author Information... }
            for A := 1 to 3 do begin
              FieldName := 'Author ' + IntToStr(A);
              if not FieldByName(FieldName).IsNull then begin
                with AIDDataModule.AuthorQuery do begin
                  ParamByName('AuthorID').AsInteger :=
                    AIDDataModule.ArticleQuery.FieldByName(
                    FieldName).AsInteger;
                  Open;
                  WriteLn('<EM>Author:</EM> <STRONG>' +
                    FieldByName('Name').AsString +
                    '</STRONG><BR>');
                  WriteBlob(TBlobField(FieldByName(
                    'Bio & E-mail address')));
                  WriteLn('<BR><BR>');
                  Close;
                end;
              end;
            end;
            WriteLn('<BR>');
            WriteLn('<A HREF="' + BackIssuesURL +
              '">Click here to order back issues...</A>');
            WriteLn('<HR><BR>');
            Next;
          end;
        finally
          Close;
        end;
      end;
    finally
      if not NoneFound then begin
        WriteLn('<H3>End of list.</H3><BR><BR>');
        WriteLn('<A HREF="' + PagesURL +
          'tdmaid.htm">Return to Search Form...</A>');
      end;
      WriteLn('</BODY></HTML>');
    end;
  end;
end;
```


TDMAid - Query Results - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print Font Mail
Address http://www.delphimag.tpower.com/tdmaid2.exe
Links

The Delphi Magazine - Article Index Database

Query Results - 24 entries matched your selection

Subclassing Windows from EXE files in 16-bit appliccations

Type: Article, *Issue* #15 (November 1996), *Page* #8

Summary: Normally you can only subclass windows from DLLs, John Chaytor shows how to do it from EXE files too.

Keywords: Callbacks, Exported functions, FreeProcInstance, MakeProcInstance, Smart callbacks $K directive, Subclassing

Author: **John Chaytor**
John Chaytor lives in Brighton, UK and can be contacted via CompuServe as 100265,3642

Click here to order back issues...

Digging Into IDAPI

➤ *Figure 2*

in a dialog on the web server machine and waits for a user to come along and click the `OK` button.

You might think that this isn't much of an issue. It's really quite easy to avoid writing CGI programs with unit initialization blocks that might raise exceptions. However, this same problem exists with Delphi's own units and their initialization blocks as well. Specifically, when the CGI application uses BDE units (`DB` and `DBTables`), the BDE is initialized within a unit initialization block. If a problem occurs, like the BDE was not installed on the server machine, you'll get an error dialog on the server.

Conclusion

There you have it, a database-aware CGI application. Really not much to it. Just be very careful to guard against BDE initialization errors and if you're using NTFS partitioning on your NT web server, you'll have to be very careful about setting up the access permissions for the anonymous account which Internet users will be connecting through. Local SQL with BDE likes to create temporary files in the BDE directories, the application directory and the data directory. Internet users will need access to all these places.

In March, we'll return to look at server APIs. Specifically, we'll learn how to use Microsoft's Internet Server API (ISAPI).

Steve Troxell is a Senior Software Engineer with TurboPower Software. He can be reached by email at stevet@tpower.com or on CompuServe at 74071,2207