A Web Server Text Difference Engine

by Paul Warren

Last month we extended Steve Troxell's TCGI component to handle file uploads. File uploads add an extra dimension to CGI programming with Delphi. This time, as promised, we will explore that extra dimension by creating a useful web server utility that depends on the ability to accept files from a client browser.

The most common use of file uploads on the web is to allow a user to submit an image file, along with a message, to a newsgroup or guestbook. There are many other potential uses, though, especially when you consider corporate intranets and personal infranets as well as the web.

Think how useful it would be to be able to submit a text file to a corporate server and have it converted to a web page complete with the company logo and formatting. Other ideas include spelling and grammar checking, an image catalog and an html source code archive, complete with syntax highlighting.

There are so many possibilities that deciding what utility would best illustrate the value of file uploads wasn't easy. I finally settled on a text file difference engine. Besides being illustrative it should also prove useful.

Extending TCGI Further

While the TCGI component can accept a single file from an upload capable browser, a text file difference engine requires two files: a base revision and a comparison revision. We'll need to do a little more work on TCGI before continuing.

I mentioned last month that the provision for multiple file uploads given in RFC 1867 had not been implemented. While it's true we can't submit multiple files from one <INPUT> element there is nothing to stop us from putting multiple <INPUT> elements on a form. Although quite different from the multiple uploads suggested in the RFC, this suits our needs here perfectly.

In a way TCGI already handles multiple files uploaded this way. Both files get uploaded correctly, but unfortunately there will only be FILENAME and TEMPFILE form variables for the last file uploaded. This is a common problem with some html form elements (for example the <SELECT MULTI=YES> element) that submit multiple values: they are all submitted with the same name. We will need to find a way to distinguish between these variables.

The easiest way to accomplish this is to increment a counter and tack the number on the end of the variable name. Listing 1 shows the code for the modified LoadMultiCGIUserData method of TCGI. Note also that I have modified the code to accommodate Netscape 3's rogue behavior that I described under the heading *Postscript* in the last article.

The Difference Engine

With these minor changes out of the way let's start building our difference engine. A difference engine needs to compare a base revision with a comparison revision. Lines common to both revisions must be reproduced without change. Any line in the base revision that is not in the comparison revision needs to be identified and marked as deleted. Similarly, any line found in the comparison revision that was not in the base revision needs to be marked as added.

To implement the difference engine we will declare a class TDifference with three class fields of type TStrings. These fields hold the base and comparison revisions and a destination for the marked up result.

To best envision how the engine will work you should open two small text files on your desktop and place them side by side. With the base revision on the left highlight the first line. Next, delete any blank lines from the top of the comparison revision. Now look through the comparison revision one line at a time. If the line doesn't exist it has clearly been deleted from the base revision. If the line is at the top of the comparison revision then the line exists and is unchanged. Finally, if the line exists anywhere but the top then the intervening lines have been added to the base revision. Finally, repeat this procedure for all the lines in the base revision.

Creating a method to perform this search is straightforward. The function Found in Listing 2 first sets Result to -1, the state where no matching line is found. Then we search all the lines in the comparison revision. If we find a match in the top position we set Result to 0 and Break the loop. If we find a

A Word About Infranets

So what is an infranet? When you use a personal web server to serve content on a small, personal, network (one or two PCs) I call that an infranet (from infra, smaller than or below). Other than size there is nothing to distinguish an infranet from a corporate intranet or, for that matter, the internet.

Installing a personal web server is all about bringing the power of internet technologies to your own desktop. I use O'Reilly's Website as my personal web server because of its modest 4.7Mb compressed size. Very modest when compared to Microsoft's 27Mb behemoth, Personal Web Server.

If you don't already have a personal web server on your PC, I have included a description of how to set up O'Reilly's Website on this month's companion disk. For a description of how to set up MS Personal Web Server see Eyal Hirsch's article *Implementing Active Server Pages* in Issue 42.

match elsewhere we set `Result` to `i` (the line number) and `Break` the loop.

Next, we need a public routine to generate the html result page. This routine, which I have called `Execute`, iterates through the base revision text file and calls `Found` for every non-blank line. As you can see from Listing 3, we examine the result of `Found` inside a case statement.

If `Found` returns -1 we mark up the line as being deleted. If `Found` returns `0` we copy the line unchanged and remove any blank lines from the top of the comparison revision. In the case `Found` returns any other value we copy all the lines in the comparison revision up to the match and mark them as added.

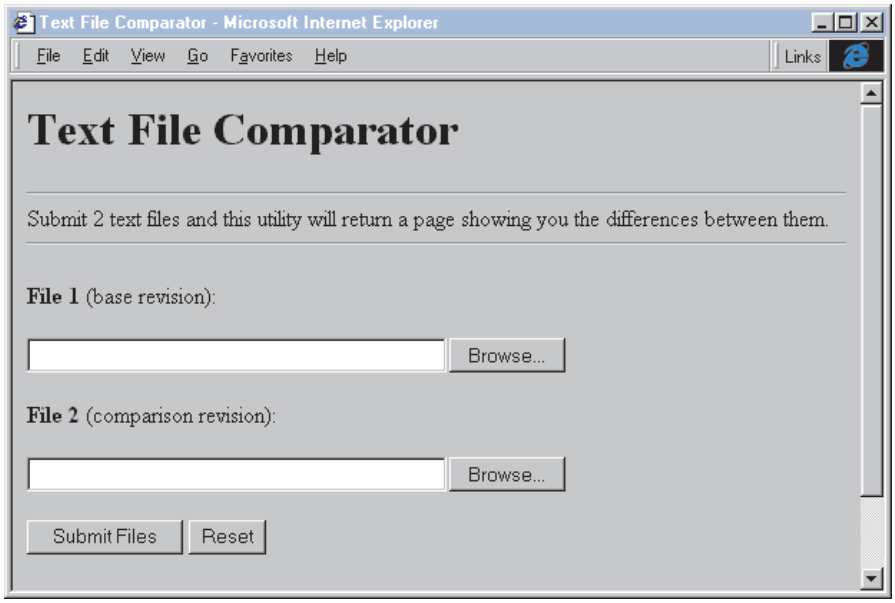
After iterating through the base revision, any lines left in the comparison revision must have been added. To add these lines to the result page we call `Dest.AddStrings`, after marking them as added of course.

That completes our simple but effective difference engine (you can add to it if you wish). All we need to do now is incorporate it into a CGI web server application.

➤ *Listing 1*

```
procedure TCGI.LoadMultiCGIUserData;
{ Reads, parses, and decodes values for the standard CGI
  form variables in a multipart form. }
const
  Eom: boolean = false;
  FileCounter: integer = 0;
var
  ContentLength: LongInt;
  InputFCB: File;
  RequestMethod: string;
  S: string;
  LabelStr: String;
  LastLabelStr: String;
  ValueStr: String;
  Buffer: array of char;
  AttachStream: TMemoryStream;
  UniqueFileName: string;
  function read1ln(var Value: string): integer;
  begin
    Result := SearchBuf(#13#10,Buffer[0],ContentLength)+2;
    SetLength(Value, Result);
    Move(Buffer[0], Value[1], Result);
    Move(Buffer[Result], Buffer[0], Length(Buffer)-Result);
  end;
  function readAttachment: integer;
  begin
    Result := SearchBuf(#13#10'--'+
      CGIItems.Values['CONTENT BOUNDARY'],
      Buffer[0], ContentLength);
    AttachStream.Write(Buffer[0], Result);
    Move(Buffer[Result], Buffer[0], Length(Buffer)-Result);
  end;
begin
  RequestMethod :=
    Uppercase(FCGIItems.Values['REQUEST METHOD']);
  if RequestMethod = 'POST' then begin
    if FCGIItems.Values['CONTENT TYPE'] <> '' then begin
      ContentLength :=
        StrToInt(FCGIItems.Values['CONTENT LENGTH']);
      AssignFile(InputFCB, '');  { standard input }
      Reset(InputFCB, 1);
      try
        SetLength(Buffer, ContentLength);
        BlockRead(InputFCB, Buffer[0], ContentLength);
        while not Eom do begin
          read1ln(S); // read a line
          if S <> #13#10 then begin
            while true do begin
              if Pos('Content-Disposition', S) <> 0
                then begin
                // delete to first "
                System.Delete(S, 1, Pos('"', S));
                LabelStr := System.Copy(S, 1,
                  Pos('"', S)-1); // copy name
                System.Delete(S, 1,
                  Pos('"', S)); // delete name
                if Pos('FILENAME',
                  uppercase(S)) <> 0 then begin
                  LabelStr := UniqueLabelStr('FILENAME');
                  LastLabelStr := LabelStr;
                  // delete to filename
                  System.Delete(S, 1, Pos('"', S));
                  ValueStr := System.Copy(S, 1,
                    Pos('"', S)-1); // copy value
                  if ValueStr <> '' then begin
                    FFormItems.Values[LabelStr] := ValueStr;
                    LabelStr := '';
                    ValueStr := '';
                  end;
                  read1ln(S); // read another line
                  if Pos('Content-Type', S) <> 0 then begin
                    LabelStr :=
                      UniqueLabelStr('CONTENT-TYPE');
                    System.Delete(S, 1,
                      Pos(':', S)+1); // delete to :
                    ValueStr := System.Copy(S, 1,
                      Length(S)); // copy name
                    if ValueStr <> '' then begin
                      FFormItems.Values[LabelStr] :=
                        ValueStr;
                      LabelStr := '';
                      ValueStr := '';
                    end;
                    read1ln(S); // read another line
                  end;
                  if S = #13#10 then begin
                    // if there is content...
                    AttachStream := TMemoryStream.Create;
                    try
                      // copy to memory stream
                      readAttachment;
                      // create new file name
                      UniqueFileName := UpldrDir+
                        ChangeFileExt(ExtractFileName(
                        FFormItems.Values[LastLabelStr]),'')
                        +FloatToStr(TimeStampToMSecs(
                        DateTimeToTimeStamp(Time)))+
                        IntToStr(FileCounter)+
                        ExtractFileExt(
                        FFormItems.Values[LastLabelStr]);
                      // write file to disk
                      AttachStream.SaveToFile(
                        UniqueFileName);
                      // save temp file name as
                      // form variable
                      FFormItems.Values[UniqueLabelStr(
                        'TEMPFILE')] := UniqueFileName;
                      Inc(FileCounter);
                    finally
                      AttachStream.Free;
                    end;
                  end;
                end;
                Break;
              end;
              if Pos(CGIItems.Values['CONTENT BOUNDARY'], S)
                <> 0 then begin
                // remove first 2 chars
                System.Delete(S, 1, 2);
                // check for Eom
                System.Delete(S, 1,
                  Length(CGIItems.Values[
                  'CONTENT BOUNDARY']));
                if S = '--'#13#10 then
                  Eom := true;
                Break;
              end;
              // append to valuestr
              ValueStr := ValueStr + S;
              // read another line
              read1ln(S);
            end;
          end;
          if ValueStr <> '' then begin
            // remove CRLFs from the end
            FFormItems.Values[LabelStr] :=
              System.Copy(ValueStr, 1,
                Length(ValueStr)-2);
            LabelStr := '';
            ValueStr := '';
          end;
        end;
      finally
        CloseFile(InputFCB);
      end;
    end;
  end;
end;

function TCGI.UniqueLabelStr(Value: string): string;
var
  Counter: integer;
begin
  Result := Value;
  Counter := 0;
  while FFormItems.IndexOfName(Result) <> -1 do begin
    Inc(Counter);
    Result := Result+IntToStr(Counter);
  end;
end;
```



➤ *Figure 1*

The Web Server Application

Figure 1 shows the form we'll use to submit our files for comparison. There are two `<INPUT TYPE=FILE>` elements and `Submit` and `Reset` buttons. There is also a hidden field containing the url of this page so we can return to it from the difference results. When you submit the form the first file input will be called `FILEINPUT` and the second will be called `FILEINPUT1`. The uploaded files will be `TEMPFILE` and `TEMPFILE1`.

The first thing the application must do is copy the required form variables into local variables for speed and convenience. Next, we must validate the user entry. Here we check to see if files of the right type have been submitted correctly. If this is not the case, an exception is raised.

It's important to note here that the entire application is effectively inside a `try...finally` block. In case of any exception, whether raised in the user validation or elsewhere, the uploaded files will be deleted. Remember that it is the responsibility of the calling application to clean up.

After validating entries we run the difference engine. First we create three `TStringLists`, one each for the base and comparison revisions and one for the marked up result page. Then we load the submitted files from disk.

We then create an instance of `TDifference` with the three string lists as parameters and call `Execute`. Finally, we construct the return web page using `writeln` calls. Listing 4 shows the complete code for the difference utility application.

If you compile the utility and copy it into the cgi-bin directory of your web server you should be able to submit two text files and see the difference page returned to you. Figure 2 shows the result page of a pair of similar text files submitted to the application. Note that files must have a MIME type of `text/plain` or `text/html`. You can add or change MIME types from your `File types` tab in Internet Explorer's `Options` property page.

➤ *Listing 2*

```
function TDifference.Found(Value: string): integer;
var
  i: integer;
  S: string;
begin
  Result := -1; // assume not found
  for i := 0 to Comp.Count-1 do begin
    S := Comp[i];
    if S <> '' then
      if CompareStr(Value, S) = 0 then begin
        if i = 0 then
          Result := 0
        else
          Result := i;
        Break;
      end;
  end;
end;
```

➤ *Listing 3*

```
procedure TDifference.Execute;
var
  i, j: integer;
  Line: integer;
begin
  Dest.Add('<PRE>');
  for i := 0 to Source.Count-1 do begin
    if source[i] <> '' then
      case Found(Source[i]) of
        // doesn't exist so mark deleted
        -1 : Dest.Add('<FONT COLOR=GREEN><B><U>'+ Source[i]+'</U></B></FONT>');
        0  : begin
               // exists at top so copy unmarked, remove found()
               Dest.Add(Source[i]);
               Comp.Delete(0);
               RemoveLeadingBlanks;
             end;
        else
          begin
            // mark all lines up to found() as added: remove found()
            Line := Found(Source[i]);
            for j := 0 to Line-1 do
              Dest.Add('<FONT COLOR=BROWN><U><I>'+Comp[j]+'</I></U></FONT>');
            Dest.Add(Comp[Line]);
            for j := 0 to Line do
              Comp.Delete(0);
            RemoveLeadingBlanks;
          end;
        end;
      else
        Dest.Add(Source[i]);
  end;
  // add any remaining lines from Comp - mark as added
  if Dest.Count <> 0 then begin
    Dest.Add('<FONT COLOR=BROWN><U><I>');
    Dest.AddStrings(Comp);
    Dest.Add('</I></U></FONT>');
  end;
  Dest.Add('</PRE>');
end;
```

```
program differ;
{$APPTYPE CONSOLE}
uses
  SysUtils,
  CGIAPI,
  Classes,
  FileCtrl,
  DiffEng in '..\..\Tools\DiffEng.pas';
var
  FileName: string;
  TempFile: string;
  FileType: string;
  FileName1: string;
  TempFile1: string;
  FileType1: string;
  ReturnUrl: string;
  A: array[0..100] of char;
  DifEng: TDifference;
  SSource, SComp,
  SDest: TStringList;
  i: integer;
procedure Error(ErrorStr: string);
begin
  writeln('content-type: text/html'#13#10#13#10);
  writeln(ErrorStr);
end;
begin
  try
    with CGI do begin
      // get required variables for first file
      FileName :=
        ExtractFileName(FormItems.Values['FILENAME']);
      TempFile := FormItems.Values['TEMPFILE'];
      FileType := FormItems.Values['CONTENT-TYPE'];
      // get required variables for second file
      FileName1 :=
        ExtractFileName(FormItems.Values['FILENAME1']);
      TempFile1 := FormItems.Values['TEMPFILE1'];
      FileType1 := FormItems.Values['CONTENT-TYPE1'];
      ReturnUrl := FormItems.Values['RETURN_URL'];
    end;
    // validate user entries
    if (FileName = '') or (TempFile = '') then begin
      Error('');
      raise EAbort.Create('File 1 not submitted correctly');
    end;
    if (FileName1 = '') or (TempFile1 = '') then begin
      Error('');
      raise EAbort.Create('File 2 not submitted correctly');
    end;
    if (Pos('text', FileType) = 0) or
      (Pos('text', FileType1) = 0) then begin
      Error('');
      raise EAbort.Create('Files must be of type text');
    end;
    // run difference engine on the 2 temp files
    try
      SSource := TStringList.Create;
      SComp := TStringList.Create;
      SDest := TStringList.Create;
      try
        SSource.LoadFromFile(TempFile);
        SComp.LoadFromFile(TempFile1);
        DifEng := TDifference.Create(SSource, SComp, SDest);
        try
          DifEng.Execute;
          // create and display response page
          writeln('content-type: text/html'#13#10#13#10);
          writeln('<HTML>');
          writeln('<HEAD><TITLE>>Text File Difference'+
            ' Results</TITLE>');
          writeln('<META NAME="GENERATOR"'+
            ' CONTENT="Generated by HomeGrown Text File'+
            ' Difference Engine">');
          writeln('</HEAD>');
          writeln('<BODY BGCOLOR="#FFFFFF">');
          writeln('<BODY>');
          writeln('<H1>Text File Difference Results</H1>');
          writeln('<HR SIZE=2>');
          writeln('Difference results for <B>'+
            ExtractFileName(Filename)+
            '</B> compared to <B>'+
            ExtractFileName(FileName1)+'</B>.');
          writeln('<P><B>Legend:</B> Same in both'+
            ' revisions --- <FONT COLOR=GREEN><B>'+
            '<U>Deleted from base revision</U></B></FONT>'+
            ' --- <FONTCOLOR=BROWN><U><I>'+
            'Added to base revision</I></U></FONT>');
          writeln('<HR SIZE=2>');
          for i := 0 to SDest.Count-1 do
            writeln(SDest[i]);
          writeln('<HR SIZE=2>');
          writeln('<A HREF="'+ReturnUrl+
            '">Return to referring page</A>');
          writeln('<HR SIZE=2>');
          writeln('<B>Generated by:</B> HomeGrown''s Text'+
            'File Difference Engine v1.0');
          writeln('</BODY></HTML>');
        finally
          DifEng.Free;
        end;
      finally
        SSource.Free;
        SComp.Free;
        SDest.Free;
      end;
    except
      Error('');
    end;
  finally
    // always delete TempFiles
    DeleteFile(StrPCopy(A, TempFile));
    DeleteFile(StrPCopy(A, TempFile1));
  end;
end.
```

➤ *Listing 4*

Conclusions

We have extended the capabilities of the TCGI component to handle files submitted from an upload capable browser. As a side effect we have a mechanism to handle multiple form variables with the same name, a useful addition in itself.

Using these new capabilities we have created a simple but useful text file difference utility. More importantly, though, we can now explore the myriad possibilities that file uploading creates, on the web, on our intranets and on standalone PCs too.

Paul Warren runs HomeGrown Software Development in Langley, British Columbia, Canada and can be contacted at hg_soft@uniserve.com

➤ *Figure 2*