Under Construction:
Delphi 4 Web Broker Technology

by Bob Swart

In Issues 24 and 25 we started to play with the Delphi 3 Web Modules. Now, a full year later, Delphi 4 is available, and so are the new Web Module enhancements. In this article we will check out these updates and how we can extend them even further.

First of all, Web Modules used to ship only with the Delphi 3 Client/Server Suite. They are still included by default only with the Delphi 4 Client/Server Suite. However, the Web Broker technology is now also available as a separate add-on for Delphi 4 Professional users. That means a whole lot more people who may want to know how to work with those Web Modules (if you don't have Issues 24 and 25, they are included on *The Delphi Magazine Collection '98 CD-ROM*, available now!). The add-on is not free: see the box at end for contact and pricing details.

Delphi 4 Web Broker

Let's start now with the new additions to the Web Broker technology in Delphi 4. The enhancements are twofold: we have a new Wizard (the Database Web Application Wizard) and a new component (the `TDataSetPageProducer`). The former can be found in the Object Repository after a `File | New` operation to start a new project, but we'll start with the latter here: the new `TDataSetPageProducer`.

TDataSetPageProducer

This is derived from `TPageProducer` and adds some special data-aware functionality to the `PageProducer`. It comes in a separate source file DSPROD.PAS (to be found in the SOURCE/INTERNET directory of your Delphi 4 C/S or Web Broker installation). `TDataSetPageProducer` adds a property called `DataSet` to the normal `TPageProducer`. As we all know (or can read in the previous articles), we can supply a `PageProducer` with a predefined HTML text, in which we can embed special 'tags' that start with a `#`. With the normal `TPageProducer`, we have to write special code in the `OnHTMLTag` event handler to replace these tags with another text. However, using a `TDataSetPageProducer`, the tags are replaced automatically by the current value of the field whose name is the same as the name of the current tag. A very handy feature, and one that can save us a lot of typing, as everything is now handled automatically behind the scenes.

As an example, we can have the HTML text in Listing 1 assigned to the `HTMLDoc` property (of type `TStrings`).

To use this example HTML text, start a new Web Module Application (with `File | New Web Module Application`), drop a `TTable` and a `TDataSetPageProducer` on the Web Module. Assign the `TTable` to the `BIOLIFE` table (alias `DBDEMOS`) and connected it to the `DataSet` property of the `TDataSetPageProducer`. In the above HTML text, we use four special tags: one that has nothing to do with the `BIOLIFE` table, and three that refer to fields inside the table.

We need to create a default action (right click on the Web Module) and write only one line of code for the `OnAction` event handler `WebModule1WebActionItem1Action`:

```
Response.Content :=
  DataSetPageProducer1.Content
```

Now it's time to execute the Web Module application. Personally, I always generate ISAPI DLLs and then use my IntraBob CGI and ISAPI debugger (included on the disk with Issue 34, also avaialble from www.drbob42.com) to execute them from the Delphi IDE (just set IntraBob as the host application in the `Run | Parameters` dialog). The results of executing this ISAPI DLL are shown in Figure 1.

Note that the `#TIME` tag was not replaced by a new value (we didn't write any code for the `OnHTMLTag` event handler). What's worse, however, is that the length in centimeters isn't replaced either (the fourth tag). The worst thing is that this field, which has a space and bracket characters in it, *cannot* be used in combination with the `TDataSetPageProducer`. The space in the fieldname `Length (cm)` denotes the end in the tagname and, without the space, the tagname can never match the fieldname. An attempt to surround the tagname with quotes has no effect. And adding double quotes to it produces an access violation and *Internal Server Error 500* in the ISAPI DLL (I could reproduce this for every tag, so it seems we should never use double quotes in tags).

Fortunately, there is a way to at least try to fix this, since the `TDataSetPageProducer` will still call the `OnHTMLTag` event for each tag encountered. Including the tags for which a `ReplaceText` is already found, by the way. We can hook into this event and make sure that we replace the tag with the name `Length` with the actual value of the field `Length (cm)`. This will actually perform the same actions as the `TDataSetPageProducer`, by the way.

➤ *Listing 1*

```
It's now <#TIME>
<P>
The current Fish is called <#Common_Name>
and has a length of <#Length_in> inches
and <#Length (cm)> centimeters.
```

Executing the code in Listing 2 will ensure that the tag with the name Length is indeed replaced by the value of the field with the name Length (cm) of the current record in the DataSet assigned to the DataSetPageProducer. It's a bit of an ugly workaround, but I hope not many people use fieldnames with spaces (it's a good reason to avoid them).

The results of executing this version of the Web Module DLL are shown in Figure 2 (note that this time only the #TIME tag is not replaced by another text).

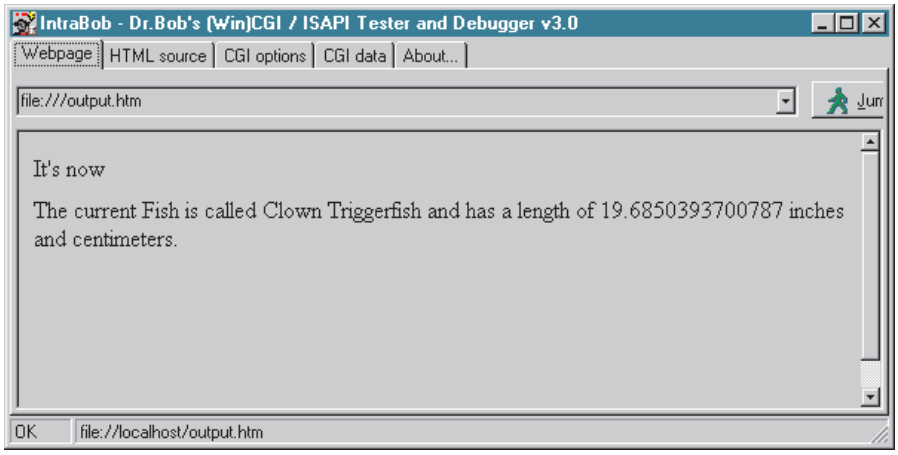
The final piece of code (Listing 3) involves replacing the #TIME tag with a meaningful value. This can be done by a few simple lines added to the OnHTMLTag event where we convert the value of NOW to a String using DateTimeToStr. The final result looks like Figure 3 (this time all the tags have been replaced).

As an optional enhancement, we could hook into the OnHTMLTag event again, and modify the ReplaceText for the Length_in tag to show only 2 digits after the numeric dot, so we don't get a stupid accurate looking number such as 19.6850393700787 inches, but a mere 19.68 inches *(and no, I don't use banker's rounding, I'm simply truncating the string)*. I leave it as an exercise for you to implement this. Remember that the OnHTMLTag event is still called for the Length_in tag, even if a ReplaceText is already given (and that's actually the right moment to go in and change the ReplaceText itself).

I'm sure it's clear that the TDataSetPageProducer can be especially helpful in cases where we want to present a single record from a given table, as opposed to the TDataSetTableProducer which is good for cases where we need to present an overview of multiple records at once.

DB Web Application Wizard

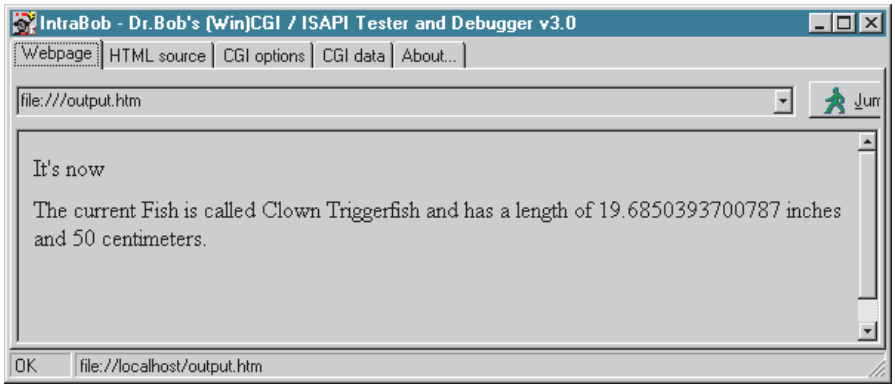
The Database Web Application Wizard can generate CGI or WinCGI applications or ISAPI/NSAPI DLLs, just like the regular Web Application Wizard. It is similar to the Database Form Wizard, with the exception that it doesn't offer us



➤ *Figure 1*

```
procedure TWebModule1.DataSetPageProducer1HTMLTag(Sender: TObject; Tag: TTag;
  const TagString: String; TagParams: TStrings; var ReplaceText: String);
var Field: TField;
begin
  if TagString = 'Length' then begin
    if Assigned(DataSetPageProducer1.DataSet) then begin
      Field := DataSetPageProducer1.DataSet.FindField('Length (cm)');
      if Assigned(Field) then
        ReplaceText := Field.DisplayText
    end
  end
end;
```

➤ *Listing 2*



➤ *Figure 2*

```
procedure TWebModule1.DataSetPageProducer1HTMLTag(Sender: TObject; Tag: TTag;
  const TagString: String; TagParams: TStrings; var ReplaceText: String);
var Field: TField;
begin
  if TagString = 'Length' then begin
    if Assigned(DataSetPageProducer1.DataSet) then begin
      Field := DataSetPageProducer1.DataSet.FindField('Length (cm)');
      if Assigned(Field) then
        ReplaceText := Field.DisplayText
    end
  end else begin
    if TagString = 'TIME' then
      ReplaceText := DateTimeToStr(Now)
  end
end;
```

➤ *Listing 3*

```
if IsMultiThread then
  Response.Content := 'This is a multi-threaded ISAPI DLL'
else
  Response.Content := 'Warning: this ISAPI DLL is not thread-safe...'
```

➤ *Listing 4*

the ability to define a master-detail relationship. Instead, we can pick an alias, select a table from that alias, select the fields we want and input some HTML webpage specific information.

If you look closely at Figure 4, it will already spoil the fact that a `TDataSetTableProducer` will be used to display the information. Maybe the Wizard was written before the `TDataSetPageProducer` was available, or maybe Inprise just didn't feel like using the latter in a Wizard. Especially when combined with the wish to show a master-detail relationship, the `TDataSetPageProducer` could be used for the 'master' where the `TDataSetTableProducer` would be perfect for the 'detail'. Alas, there's no such Wizard yet, but keep an eye on my website at www.drbob42.com, as I'm working on one as we speak (I was hoping to have it available in time to show it in this article, but you'll have to come by my website yourself, I'm afraid).

IsMultiThread?

There's one last thing I want to check before I call it a day, and that's the value of the `IsMultiThread` variable of the ISAPI DLL right after it's loaded by IIS. As we recall from the ISAPI DLL back in issue 34, there was a bug in Delphi 3.0x (or IIS, depends on which way you want to look at it) that prevented the `IsMultiThread` variable from the ISAPI DLL to be correctly initialised to True, meaning it could cause problems, especially when it comes to memory management operations (which are not thread-safe).

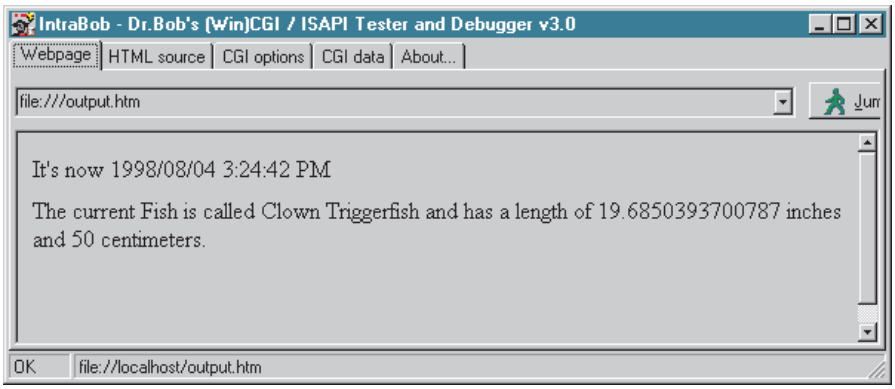
Using the new Web Module Wizards of Delphi 4, I generated a number of ISAPI DLLs, and tested them in our copy of Microsoft Internet Information Server (version 3.0) on our Intranet. The `OnAction` event handler of the default action had the line of code shown in Listing 4.

Sure enough, with Delphi 3.0x we got the message that the ISAPI DLL would not be thread-safe. Inprise did pay attention, however, because with Delphi 4 we now get the correct message *This is a multi-threaded ISAPI DLL*, indicating that `IsMultiThread` is now correctly initialised. So that's score 1 for Inprise, (but still 2 for me!).

Next Time

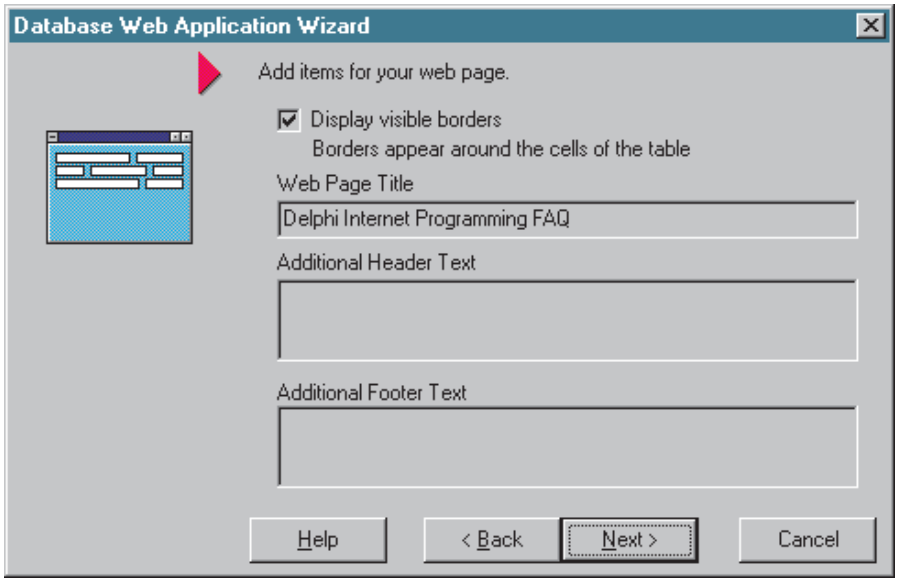
This is not the end of the story: I'm sure we'll get back to Web Broker topics in future issues. And be sure to check out my website for a special Master-Detail Database Web Broker Application Wizard. Next time, we'll see how we can integrate some middleware technology (such as MIDAS or CORBA) with our client components and applications to turn them into multi-tier applications.

Bob Swart (aka Dr.Bob, visit www.drbob42.com) is a professional knowledge engineer technical consultant using Delphi, JBuilder and C++Builder for Bolesian and a freelance technical author. In his spare time, Bob likes to watch video tapes of *Star Trek Voyager* and *Deep Space Nine* with his 4.5-year old son Erik Mark Pascal and his almost 2-year old daughter Natasha Louise Delphine.



➤ *Above: Figure 3*

➤ *Below: Figure 4*



Delphi 4 Web Broker Add-On: Contact Information And Pricing

When ordered directly the Web Broker add-on costs $199.95 from Inprise USA and £139 from Inprise UK. Quote the SKU of **HDW1310WWFS180.**

Inprise USA
100 Enterprise Way
Scotts Valley
CA 95066, USA
Tel: +1 408 431 1000

Inprise UK
8 Pavilions, Ruscombe Business Park
TWYFORD, Berkshire
RG10 9NN, United Kingdom
Tel: 0800 454065 (this is Customer Services) or from outside the UK call +44 (0)171 814 5047

Web: www.inprise.com/delphi/ordering/index.html