Intraweb versus Morfik

Michaël Van Canneyt

August 2, 2009

Abstract

Intraweb - Currently at version 10 - has been around for quite some time. It is a mature technology, and as such can be expected to have a well-established code base and a rich feature set. By contrast, Morfik is a relative young technology, and still has some way to go. In this article, the basics of Intraweb are explained, and at the same time a comparison with Morfik is made.

1 Introduction

Since quite some years, Delphi is shipped with a basic version of Intraweb from AtoZed software: it allows to develop web applications in a RAD way: the HTML pages are designed as one would design forms in Delphi, using a point-and-click manner. No knowledge of HTML or Javascript is required to make a fully functional application: the details of that are hidden by the intraweb framework: An Object Pascal developer can create Web Applications just as he would create a desktop application. With the exception of interaction with the client PC (which is, for all purposes, not accessible) all can be programmed as one would program a desktop application. Knowledge of Javascript may be required only for some very specialized tasks such as creating new components for Intraweb. Although a version is shipped with Delphi, a commercial license must be purchased from AToZed if one wished to have the sources of the complete Intraweb package: Some key units are not distributed with Delphi in Binary form. Purchase is possible from the AToZed website:

`http://www.atozed.com/intraweb/`

The above featurelist can equally well be applied to Morfik - although obviously it is not shipped by default with Delphi: it has it's own IDE. Morfik was presented in some articles some time ago, and will not be explained here again. It can be evaluated and purchased from

`http://www.morfik.com/`

If the above featurelist is nearly equal, where then is the difference between Morfik and Intraweb ? The purpose of this article is to introduce Intraweb and compare where possible.

2 Starting an Intraweb application

Create a new Intraweb application is like creating any other Delphi application: under the 'File|New' menu, 'Other' must be chosen: In the dialog that pops up, under the 'VCL for the Web' section, the 'VCL For the Web Application Wizard' item will start a small wizard, which will ask for some options. The most important ones are:

Application type This is one of 'Standalone application', 'Service application' or 'ISAPI extension'. For testing and debugging purposes, the 'Standalone application' is most suitable. The ISAPI extension should be used if the web application must run under IIS.

Create User Session if this is checked, a datamodule will be created which can be used to hold user session data: this is data which is shared between the various forms in the application.

Main Form here the type of main form can be set: this is one of HTML 4.0, 3.2 or WAP, and determines which HTML constructs can be used in the application: Nowadays, HTML 4.0 can be safely assumed.

Project Name the name of the new application

Project directory

After pressing the OK button, a new application is created, and a main form for the application is set. Depending on whether 'Create user session' was checked, a session datamodule is also created.

3 Adding controls

An Intraweb application does not use the regular Delphi VCL controls. Instead, it uses it's own set of controls which are suitable for use on the web: they are located on the 'VCLWeb Standard' tab of the component palette. On this tab an Intraweb version of most of the controls on the 'Standard' tab of the component palette can be found: Buttons, Checkboxes, labels, edit or memo controls, comboboxes and all controls found in a regular application.

As such, anyone familiar with Delphi development can get started with Intraweb right away, dropping controls on the forms just as one would drop normal controls. The properties of the controls will be slightly different, but sufficiently close enough to normal controls to be understandable: top, left, width, height, Anchors: all these are present. Likewise, the events are different from normal events, but again they resemble enough the normal events: OnClick becomes OnAsyncClick etc. (to make it clear that they are handled asynchronuous: The browser continues to work while the handlers are executed on the webserver.)

4 Executing event handlers

So far, this is nothing spectacular. The real magic happens when an event handler is executed. To explain this, a simple example is used, where 2 labels (called LEName and LLeave), a button (BOK) and an edit box (EName) are put on an intraweb main form (TMainForm). The 'OnClick' event handler of the button gets the following code:

```
procedure TMainForm.BOKClick(Sender: TObject);

Var
  S : String;

begin
  If BOK.Tag=0 then
    begin
```

```
    FirstName:=EName.Text;
    EName.Text:='';
    LEName.Caption:='Enter your last name:';
    BOK.Tag:=1;
    BOK.Caption:='Go !';
    end
  else
    begin
    LastName:=EName.Text;
    LEName.Caption:='Done entering data';
    EName.Visible:=False;
    S:=Format('Hello %s %s !',[FirstName,LastName]);
    WebApplication.ShowMessage(S);
    end;
end;
```

The `FirstName` and `LastName` string variables are fields of the `TMainForm` class. At first sight, there is nothing unusual about this code. Apart from the `WebApplication` prefix to the `ShowMessage` call, this code could be code from a normal desktop application.

What happens now when the application is run in the browser, is roughly the following:

- The browser sends a request to the webserver. From the request session information, the Intraweb code in the server determines what form is needed - the main form initially.
- If the form was not instantiated, the server Intraweb code instantiates the form.
- The form generates the necessary HTML (and javascript) to render the form in the browser: Intraweb also generates the Javascript code to translate the click of the button to a call to the server. This Javascript and HTML is sent to the browser. The form remains instantiated in memory.
- The browser receives the response, and renders the form.
- When the user clicks the button, the Javascript generated by Intraweb sends a request to the server, which is translated - this is part of the magic - to the above event handler.
- the Event handler code executes (note that this is on the server).
- The HTML for the form is again generated and compared to the original HTML: the differences are sent to the browser, which re-renders the form.

In the case the WebApplication.ShowMessage() is executed, additional Javascript instructions are sent to the browser which make it show the message in a message dialog.

So, to perform the actions of this form, 2 server requests are needed: one for each click on the button. The whole chain of actions described here is handled transparantly by Intraweb. The fact that the mainform remains in memory means that the FirstName and LastName variables retain their values between the execution of the various event handlers. It also means that the web-application can be programmed as a desktop application: the instance of the form on the server contains a mirror-image of the state of the form in the browser. All written code is executed on the server , and the Intraweb code makes sure that any changes in the state of the form are communicated back to the browser.

It is instructive to compare this with what happens if the same application is made with Morfik appsbuilder. There, a new project may be created, and on the main page (called

Index), the same controls can be dropped as in the Intraweb page. The code of the `BOK` button's `Onclick` handler looks quite similar to the one for Intraweb:

```
Procedure Index.BOKClick(Event: TDOMEvent);
Begin
If BOK.Caption='OK' then
    begin
    FirstName:=EName.Text;
    EName.Text:='';
    LEName.Caption:='Enter your last name:';
    BOK.Caption:='Go !';
    end
  else
    begin
    LastName:=EName.Text;
    LEName.Caption:='Done entering data';
    EName.Visible:=False;
    ShowMessage('Hello '+FirstName+' '+LastName+' !');
    end;
End;
```

It looks quite similar. The visible differences are:

- Morfik components do not have a 'Tag' property, so the caption must be checked to see how many times the user clicked.
- ShowMessage is a global procedure.
- There is no Format() procedure. The reason will become obvious in the below paragraph.

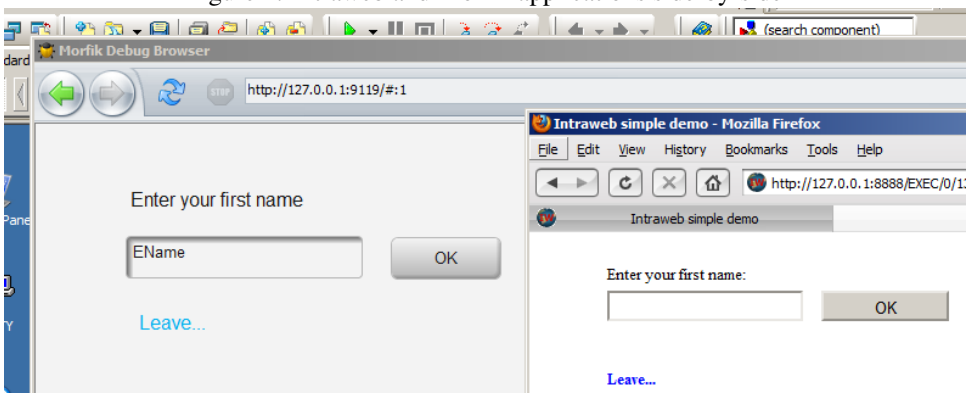
The biggest difference, however, is not immediatly visible: Where the Intraweb event handler code was executed on the webserver, the Morfik code is executed on the client: no round-trip to the server is needed. The Morfik IDE translates the pascal code of the event handler (and any other code it needs) to Javascript, and sends the code to the browser together with the layout of the form. This means that the Intraweb server application will need to do a lot more work than the Morfik application, and that any user action will necessarily introduce a small delay while the server processes the event. By contrast, the Morfik event handler will be executed at once on the client.

The fact that the Format call is not available is also a consequence of the translation to Javascript: Javascript doesn't offer a variable-number-of arguments call as used in the Format function.

5 Application Looks

A visible difference between Morfik and Intraweb is shown in figure 1 on page 5, where both applications are shown side-by-side. The Morfik application - using defaults - has a more modern look than the Intraweb application: Morfik makes extensive use of styles, offering the option to create the styles in the IDE. To do the same in Intraweb, a lot of manual CSS editing is needed. While this is not a fundamental problem, eye-candy is an important factor in Web-development, and a factor to take into consideration when considering a development environment.

Figure 1: Intraweb and Morfik applications side-by-side



Intraweb does offer Template support for it's generated web-pages: It is possible to provide a HTML page, which it will use to generate the forms: the controls on the form will placed inside the HTML page by looking for a tage of the form {%Controlname%} : the tag will be replaced with the HTML code for the control named `ControlName`. To use this feature 3 steps must be done:

- In the subdirectory Templates of the project directory, the HTML file must be placed.
- a `TIWTemplateProcessorHTML` component must be placed on the form. If the HTML file has the same name as the form (with extension .html), then no extra properties must be set. Otherwise, the fields of the `Templates` property must be set to the filename of the HTML template: there is a field per browser, so different templates can be used for different browsers.
- The form's `LayoutMgr` property must be set to the `TIWTemplateProcessorHTML` component.

If the HTML template refers to other files (images, stylesheets) then these must be placed in the Files subdirectory of the project. In figure 2 on page 6 the result of such an operation can be seen. Note that only the surrounding page is changed: the controls themselves still retain their default look, and must be treated separately if one wants to give them a different look.

6 Databases

Intraweb is able to use Delphi's database components: The standard DB-Aware controls also exist in a Intraweb version on the 'VCLWeb Data' tab of the component palette. These controls can be used to create data-aware applications. As a small demo, an application is made to reserve a seat in a film theater. For this 2 clientdatasets are used: `CDSMovies` and `CDSProjections`. The first dataset contains the title, genre, duration, description and some other fields describing the movie. The `IMDBID` field is used as a unique key: it refers to the ID of the film in the IMDB (Internet Movie Database) website. The `CDSProjections` dataset contains the projection dates, times and in which theater it is being projected, it also contains the max number of seats, and the number of seats already reserved.

The datasets will be used in various forms of the application, and therefor they are placed on the usersession datamodule: If the 'usersession' option is checked in the project wiz-

Figure 2: Intraweb application using Free Pascal webpage layout

ard, Delphi will automatically have created a usersession Datamodule. In the `OnCreate` handler of the usersession, the data is loaded from file:

```
procedure TIWUserSession.IWUserSessionBaseCreate(Sender: TObject);

begin
  FDir:=ExtractFilePath(Paramstr(0));
  LoadMovies;
end;

procedure TIWUserSession.LoadMovies;

begin
  CDSMovies.LoadFromFile(FDir+'Movies.xml');
end;
```

The movie data is in a separate XML file, located in the same directory as the application.

Showing this data is done in the same manner as in a normal desktop application: dropping an IWDBGrid on the main form, and setting the 'Datasource' property. After that, the columns property can be configured: for each column, a lot of properties can be set, but at least the 'DataField' property must be set. The Title, Genre and Duration properties will be shown. To let the user ask details about a movie, we'll make the title clickable. In the OnCLick handler, a new form will be shown which shows the details of the movie:

```
procedure TSelectMovieForm.GMoviesColumns0Click(
  ASender: TObject;
  const AValue: string);
begin
  With TMovieDetailsForm.Create(WebApplication) do
    Show;
end;
```

As can be seen, this code looks no different from code that one would write in a desktop application if one wanted to dynamically create and show a new form. Note the extra 'AValue' parameter. The only difference is that in a desktop application, the main window would still be visible. In the webapplication, only 1 window is shown at a time, so the main window will be hidden by the newly created window.

The last column in the grid shows a calculated field with a fixed text: 'Projections'. When the user clicks this column, a form should be shown that allows the user to select a projection for which he wants to book a seat. To do this, the 'LinkField' property of this column will be set to the IMDBID field. The contents of this field will be passed in the `AValue` parameter of the `OnClick` event handler of the column:

```
procedure TSelectMovieForm.GMoviesColumns3Click(
  ASender: TObject;
  const AValue: string);
begin
  With TSelectProjectionForm.Create(WebApplication) do
    begin
    IMDBID:=AValue;
    Show;
    end;
end;
```

The `TSelectProjectionForm` shows the projection dates and times for the selected movie. Which movie should be used is determined by the IMDBID property of the form. When the IMDBID property is set, the form calls the usersession datamodule and asks it to filter the `CDSProjections` dataset:

```
procedure TSelectProjectionForm.SetIMDBID(const Value: String);
begin
  FIMDBID := Value;
  UserSession.FilterProjections(FIMDBID);
end;
```

The exact details of the method are not so imported, but the interested reader can check the source code accompagnying this article. On this form, a grid is set up which shows the projection dates, times and other data.

When the user clicks one of the rows, a seat must be booked. For this purpose, all columns have their 'OnClick' handler set to the following

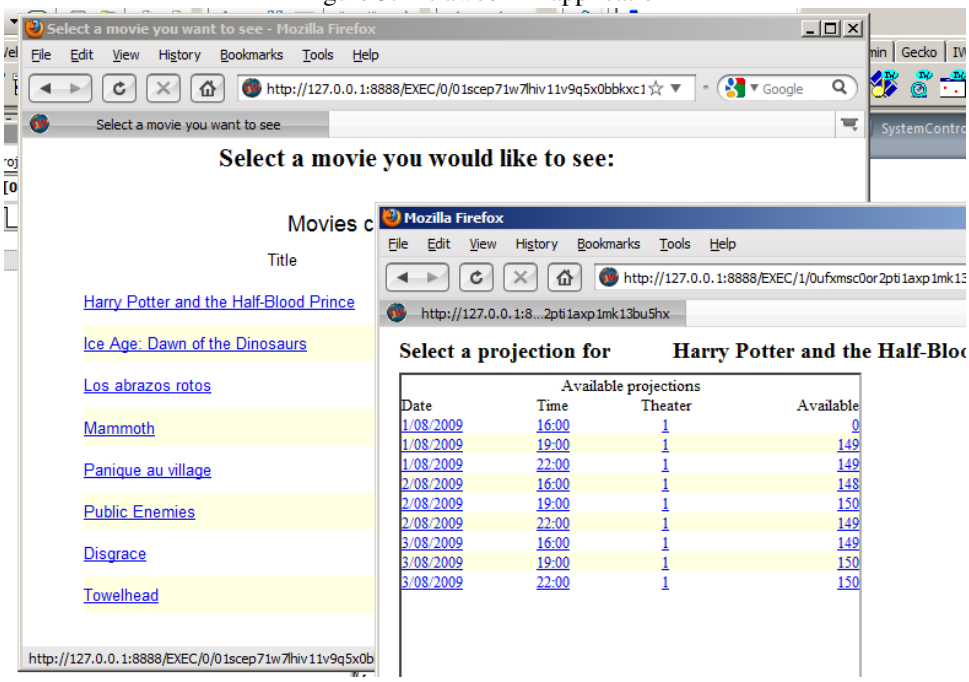
```
procedure TSelectProjectionForm.GProjectionsColumns0Click(
  ASender: TObject;
  const AValue: string);
begin
  UserSession.Book(Avalue);
  WebApplication.ShowMessage('Succesfully booked a seat!');
end;
```

The details of the 'Book' method can again be found in the sources: it simply adds 1 to the value of the 'UsedSeats' field and posts the data.

Finally, a button is placed on the form, which closes the form:

```
procedure TSelectProjectionForm.IWButton1Click(Sender: TObject);
begin
```

Figure 3: Intraweb DB application




```
  Self.Release;
end;
```

The 'Release' method closes the form, and releases it from memory. As a result, the main form (which was simply hidden by this form) is shown again.

With this very small amount of code, a complete database-aware web-application was created which has multiple forms, updates data, and is even able to show external websites. The result is shown in figure 3 on page 8. A similar application can be made with Morfik: This application can be created with equally little code, but with a slightly different approach, specific to Morfik: The sources are on the disk accompagnying this article. An explanation of the principles involved can be found in Toolboxes 2007/5 and 2007/6.

7 Conclusion

Hopefully, the above has shown that Intraweb development is easy for anyone having created Delphi desktop applications: apart from some small details, it is no different from coding a regular desktop application, once one gets used to the different properties. Therein lies it's main value: for Delphi programmers, the migration from Desktop to web could not be more simple: Most - if not all - of the existing business logic can be reused, only the GUI must be recoded, which is ideal for upgrading desktop applications.

If a Delphi programmer must start a new web-based, however, Intraweb gets serious competition from Morfik. Morfik creates web-applications which are coded just as easily, which have a more modern look out of the box, and which make for a slightly smoother user experience: much of the code does not need any server calls at all, indeed: Morfik allows to create browser-only applications which don't need a server at all.

Despite it's obvious advantages, Intraweb also suffers from some deficiencies:

- There are hardly any third-party components: Torry's pages list 2 companies that

provide a set of components, one of which has open-sourced and abandoned the components.

- Despite being included in Delphi, the Intraweb documentation is virtually non-existent; one has to guess at the purpose of many properties.
- If Intraweb 10 had not been released recently - it was released this year or at the end of last year (a fact which is not even mentioned on the project page of the AtoZed Website), one could get the impression it was abandoned: searching the web for Intraweb related sites is a disappointment.

These points leads the author to conclude that Intraweb is definitely a safe bet if existing Delphi applications must be carried to the web, but that using Morfik for new applications may be a more satisfying experience both for the developer and the end user.