BLAISE PASCAL MAGAZINE 93

Multi Platform / Object Pascal / Internet / JavaScript / WebAssembly / Pas2Js / Databases
CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux

Delphi 10.4.2 / new features By Detlef Overbeek

Max Box RSS Feeds of BBC News By Max Kleiner

New Number Component inside Delphi 10.4.2 By Detlef Overbeek & Marco Cantu (Embarcadero)

New ControlList Component inside Delphi 10.4.2 By Detlef Overbeek & Marco Cantu (Embarcadero)

How differs the new FastReport VCL 2021.0 from the previous versions?

Fastreport VCL 2021 Working with the ReportDesigner By Detlef Overbeek

InstallAware By Detlef Overbeek

What's coming in TMS WEB Core v1.7 Ancona By Bruno Fierens

Lazarus: Webform inside Lazarus By Michael van Canneyt

Code Snippets 11By Detlef Overbeek / project by Michael van Canneyt

Web Service Part 2 Storage By Danny Wind

ORM in kbmMW #3 Copying a table from one database to another By Kim Bo Madsen

BLAISE PASCAL MAGAZINE 93

Multi platform /Object Pascal / Internet / JavaScript / WebAssembly / Pas2Js / Databases
CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux

Blaise Pascal

ARTICLES

From Your Editor page 4
Cartoon
By Jerry king page 5
Delphi 10.4.2 / new features
By Detlef Overbeek page 7
Max Box RSS Feeds of BBC News
By Max Kleiner page 26
New Number Component inside Delphi 10.4.2
By Detlef Overbeek & Marco Cantu (Embarcadero) page 11
New ControlList Component inside Delphi 10.4.2
By Detlef Overbeek & Marco Cantu (Embarcadero) page 15
How differs the new FastReport VCL 2021.0
from the previous versions? page 32
Fastreport VCL 2021 Working with the ReportDesigner
By Detlef Overbeek page 34
InstallAware
By Detlef Overbeek page 52
What's coming in TMS WEB Core v1.7 Ancona
By Bruno Fierens page 60
Lazarus: Webform inside Lazarus
By Michael van Canneyt page 65
Code Snippets 11
By Detlef Overbeek / project by Michael van Canneyt page 78
Web Service Part 2 Storage
By Danny Wind page 84
ORM in kbmMW #3
Copying a table from one database to another
By Kim Bo Madsen page 97
The Bumble bee page 101

Delphinium

ADVERTISERS

The new LibStick page 6
Super Offer Bundle page 24
Subsription +Lazarus Handbook - hardcover page 25
Fastreport page 31
Delphi 10.4.2 available page 30
Barnsten page 59
Lazarus Handbook - Pocket (softcover) page 76
Lazarus Handbook - Hardcover page 77
Delphi Company page 83
Subscription + Library USB Stick page 96
Componnets4Developers page 102

Niklaus Wirth

Pascal is an imperative and procedural programming language, which Niklaus Wirth designed (left below) in 1968–69 and published in 1970, as a small, efficient language intended to encourage good programming practices using structured programming and data structuring. A derivative known as Object Pascal designed for object-oriented programming was developed in 1985. The language name was chosen to honour the Mathematician, Inventor of the first calculator: Blaise Pascal (see top right).

Publisher: PRO PASCAL FOUNDATION in collaboration © Stichting Ondersteuning Programmeertaal Pascal - Netherlands

Contributors

Stephen Ball http://delphiaball.co.uk @DelphiABall		**Peter Bijlsma -Editor** peter @ blaisepascal.eu
Dmitry Boyarintsev dmitry.living @ gmail.com	**Michaël Van Canneyt,** michael @ freepascal.org	**Marco Cantù** www.marcocantu.com marco.cantu @ gmail.com
David Dirkse www.davdata.nl E-mail: David @ davdata.nl	**Benno Evers** b.evers @ everscustomtechnology.nl	**Bruno Fierens** www.tmssoftware.com bruno.fierens @ tmssoftware.com
Holger Flick holger @ flixments.com		
Primož Gabrijelčič primoz @ gabrijelcic.org	**Mattias Gärtner** nc-gaertnma@netcologne.de	**Peter Johnson** http://delphidabbler.com delphidabbler @ gmail.com
Max Kleiner www.softwareschule.ch max @ kleiner.com	**John Kuiper** john_kuiper @ kpnmail.nl	**Wagner R. Landgraf** wagner @ tmssoftware.com
Vsevolod Leonov vsevolod.leonov@mail.ru		**Andrea Magni** www.andreamagni.eu andrea.magni @ gmail.com www.andreamagni.eu/wp
	Paul Nauta PLM Solution Architect CyberNautics paul.nauta @ cybernautics.nl	**Kim Madsen** www.component4developers.com
Boian Mitov mitov @ mitov.com		**Jeremy North** jeremy.north @ gmail.com
Detlef Overbeek - Editor in Chief www.blaisepascal.eu editor @ blaisepascal.eu	**Howard Page Clark** hdpc @ talktalk.net	**Heiko Rompel** info @ rompelsoft.de
Wim Van Ingen Schenau -Editor wisone @ xs4all.nl	**Peter van der Sman** sman @ prisman.nl	**Rik Smit** rik @ blaisepascal.eu
Bob Swart www.eBob42.com Bob @ eBob42.com	**B.J. Rao** contact @ intricad.com	**Daniele Teti** www.danieleteti.it d.teti @ bittime.it
Anton Vogelaar ajv @ vogelaar-electronics.com	**Danny Wind** dwind @ delphicompany.nl	**Siegfried Zuhr** siegfried @ zuhr.nl

Editor - in - chief
Detlef D. Overbeek, Netherlands Tel.: Mobile: +31 (0)6 21.23.62.68
News and Press Releases email only to editor@blaisepascal.eu

Editors
Peter Bijlsma, W. (Wim) van Ingen Schenau, Rik Smit

Correctors
Howard Page-Clark, Peter Bijlsma

Trademarks All trademarks used are acknowledged as the property of their respective owners.
Caveat Whilst we endeavour to ensure that what is published in the magazine is correct, we cannot accept responsibility for any errors or omissions. If you notice something which may be incorrect, please contact the Editor and we will publish a correction where relevant.
Subscriptions (2019 prices)

	Internat. excl. VAT	Internat. incl. 9% VAT	Shipment
Printed Issue **±60 pages**	€ 155,96	€ 250	€ 80,00
Electronic Download Issue **60 pages**	€ 64,20	€ 70	———
Printed Issue inside Holland (Netherlands) **60 pages**	———	€ 250,00	€ 70,00

Member and donator of WIKIPEDIA
Member of the **Royal Dutch Library**

Subscriptions can be taken out online at www.blaisepascal.eu or by written order, or by sending an email to office@blaisepascal.eu
Subscriptions can start at any date. All issues published in the calendar year of the subscription will be sent as well.
Subscriptions run 365 days. Subscriptions will not be prolonged without notice. Receipt of payment will be sent by email.
Subscriptions can be paid by sending the payment to:
ABN AMRO Bank Account no. 44 19 60 863 or by credit card or Paypal
Name: Pro Pascal Foundation-Foundation for Supporting the Pascal Programming Language (Stichting Ondersteuning Programeertaal Pascal)
IBAN: NL82 ABNA 0441960863 BIC ABNANL2A VAT no.: 81 42 54 147 (Stichting Programmeertaal Pascal)
Subscription department
Edelstenenbaan 21 / 3402 XA IJsselstein, The Netherlands
Mobile: + 31 (0) 6 21.23.62.68 office@blaisepascal.eu

Copyright notice

From your editor

Spring is in the air and I found something very nice: Delphinium, a flower that is beautiful as you can see on the cover and I think might be an ode to Delphi and to spring.
I thought it to be so nice that everybody whom subscribes or re-subscribes will get a little bag of seed which should be sown at once in this time of the year directly into a spot in your garden. Splendid flower for a summer bouquet.

And now for something completely different - as you know the phrase:
during last month the news accumulated enormously. This issue has now 102 pages!
A new version of Delphi was announced. This time with only small news: two new components and of course a lot of bug fixes.
Happily Bruno Fierens and his TMS team created again and again new items for Delphi. He displays his new plans in this issue.
He is the only one that really keeps Delphi connected to the Internet, I know there are a few others but they are not quite so sophisticated as WEBCore.
I will however investigate those suppliers as soon as possible, and write about their products. It is necessary to overlook the entire range…I think the Internet is the main future platform.

Lazarus has some big news: The Lazarus and Free Pascal foundation had asked Martin Friebe to create a form that will be able to store a form inside the IDE which is capable of using a browser (Chromium) where almost all the browsers are built on (even Microsoft's Edge).
So we hope to achieve that the form will be connected to the ObjectInspector and all other technologies that Pas2JS uses, to make happen what we want: a form that is actually the browser it self and has one special ability: WYSIWYG (What You See Is What You Get), respond to the debugging necessities,
has an underlying help and code completion and CSS styles. In this issue Michael van Canneyt has written an explanation that will show you for the first time how this can work and even wrote some code that will show the result of this. Try it!
Yet there is till a lot to be done!

I had asked Danny Wind of the Delphi Company to write a series of articles to explain very low level how to make use of a web Client and Server with the help of a REST service.
He made it very easy and understandable: break the problem into pieces, solve each part separately and then start with the total structure. He did it. It's that simple. It will also show you for once how to setup a server under Delphi.

FastReport had also some great news:
they created their newest version, to have a look at that I wrote a starters article.
Blaise Pascal Magazine is going to create - with the help of FastReport of course – a book about the use of reporting. We will try to publish that by the end of the year.

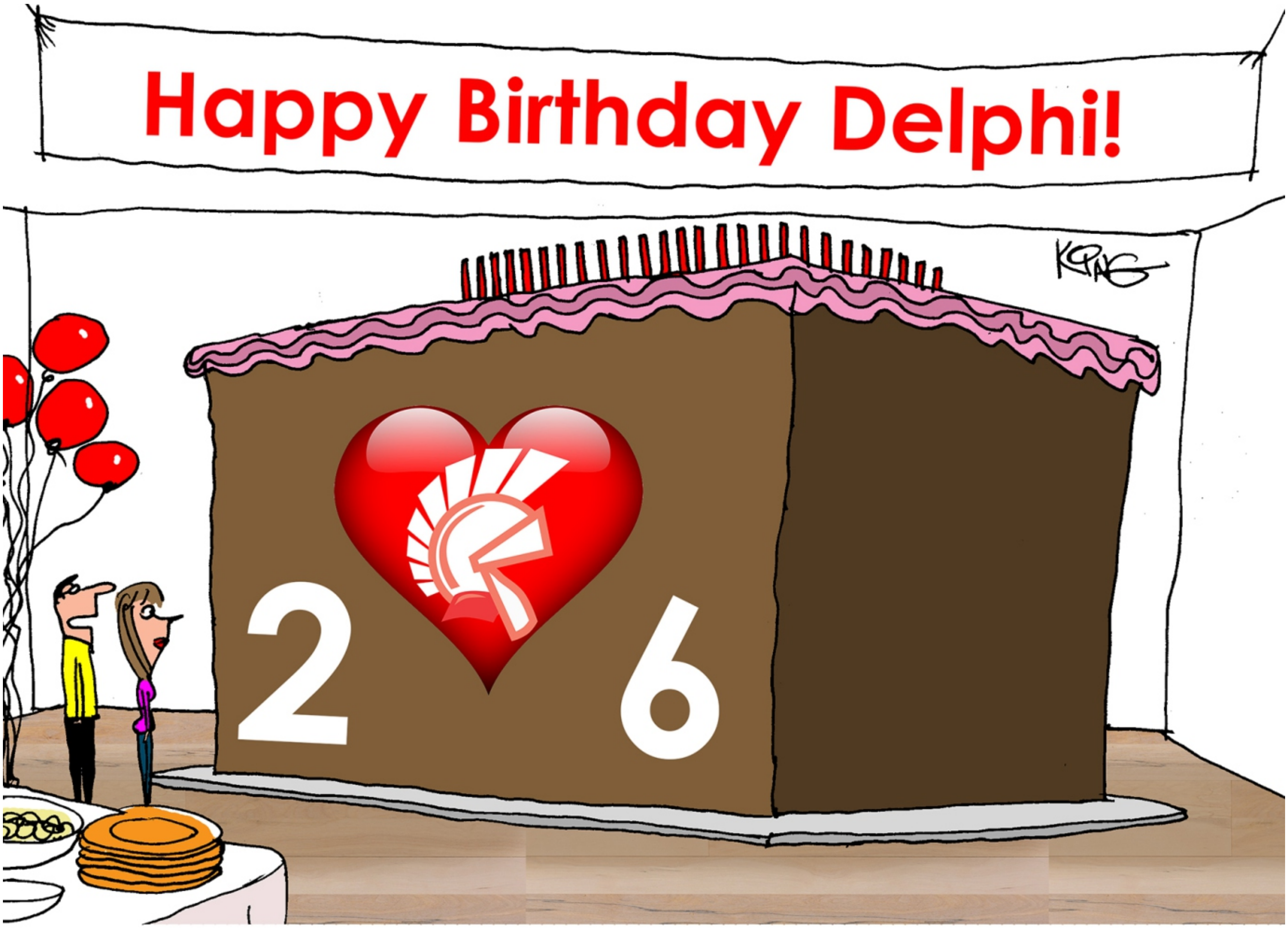
For some reason I never took a look at various install programs: only InnoSetup.
InstallAware surprised me very much: you need to evaluate and consider that installer program absolutely. Value for money and for the Delphi users there is a free version. Jason Strathmore of InstallAware told me they are working on a version for Lazarus, and because of that plan: they are working on a version that is capable of handling instalments on Linux as well as macOS - and of course Windows.

So all together that is why this issue is over a hundred pages.

Hope you'll enjoy it. Let's have fun programming!

Detlef

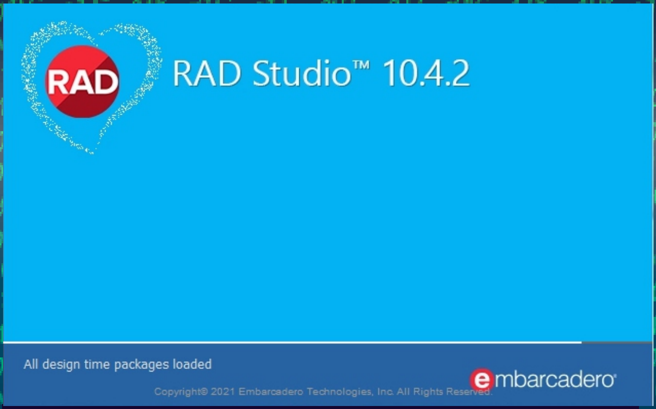
From our Technical advisor: Cartoons from Jerry King



"Our Delphi turns 26. I bought a big enough cake so everyone it's helped gets a piece."

Delphi 10.4.2 / UPDATE: New features / New Components

By Detlef Overbeek



All design time packages loaded

Copyright© 2021 Embarcadero Technologies, Inc. All Rights Reserved

INTRODUCTION

In this very long article I'll talk about the enhancements of Delphi and two new components that really are interesting. I have tried to give as much info as possible, but they're not (yet) very well documented. To understand them I tried to use them and especially the Binding mechanisms are interesting.

PRODUCTIVITY ENHANCEMENTS

- RAD Studio 10.4.2 includes new tooling to help developers build apps quickly,
- enhanced migration tooling for fast upgrades, and a new automatic silent installer.
- A new progress dialog shows what the IDE is doing during lengthy operations.
- RAD Studio's Code Insight support (using LSP) has been significantly expanded to offer better, faster.
- improved Migration Tool helps you upgrade easily and copy your IDE configurations.

PERFORMANCE AND QUALITY IMPROVEMENTS

- 10.4.2 includes improved integration with **SOAP** web services,
- improving the **WSDL** import tool and the SOAP client invocations.
- The updated **Parallel Programming Library** offers a modern style for writing multi-threaded applications that can take advantage of modern multi-core CPUs.
- You can now easily **see the data stored in generic collections in the debugger** with the introduction of a new specific debug visualizer.
- Database and remote data access technologies have also been improved for `FireDAC InterBase`, `SQLite`, `PostgresQL`, and `Oracle` integrated drivers, the `REST Client Library`, `AWS` and Azure support.
- RAD Studio's 10.4.2 release resolves a huge number of reported issues.

***1 SOAP (acronym for Simple Object Access Protocol)** *is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks. Its purpose is to provide extensibility, neutrality, verbosity and independence.*
It uses XML Information Set for its message format, and relies on application layer protocols, most often Hypertext Transfer Protocol (HTTP), although some legacy systems communicate over Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.

SOAP allows developers to invoke processes running on disparate operating systems (such as Windows, macOS, and Linux) to authenticate, authorize, and communicate using Extensible Markup Language (XML). Since Web protocols like HTTP are installed and running on all operating systems, SOAP allows clients to invoke web services and receive responses independent of language and platforms.

***2 WSDL**
Web Services Description Language
is an XML-based interface description language that is used for describing the functionality offered by a web service. The acronym is also used for any specific WSDL description of a web service (also referred to as a WSDL file), which provides a machine-readable description of how the service can be called, what parameters it expects, and what data structures it returns. Therefore, its purpose is roughly similar to that of a type signature in a programming language.

The current version of WSDL is WSDL 2.0. The meaning of the acronym has changed from version 1.1 where the "D" stood for "Definition".

***3 PPL Parallel Programming Library**
The RTL provides the Parallel Programming Library (PPL), giving your applications the ability to have tasks running in parallel taking advantage of working across multiple CPU devices and computers. The PPL includes a number of advanced features for running tasks, joining tasks, waiting on groups of tasks, etc. to process. For all this, there is a thread pool that self tunes itself automatically (based on the load on the CPU's) so you do not have to care about creating or managing threads for this purpose.

You can use this library by including System.Threading in your apps. This unit is made up of several features that can be included into new and existing projects. The unit also includes a number of overloaded arguments to make it suitable for C++ as well as Delphi.

Using the PPL, your applications can easily:
Make looping faster with TParallel.For.
Run multiple tasks in parallel using `TTask` *and* `Itask`. *Leave a process running focusing on other tasks and then get the result of that process at the point you want. IFuture allows you to establish a priority for the running code blocks and still return the results when needed.*
Platform Support
The PPL works on Windows, MacOSX, Android, and iOS devices.

EXPANDED DELPHI SUPPORT
Delphi sees over 20 significant compiler performance improvements that substantially reduce compilation time.
Delphi also has enhanced Code Insight support, highlighting warnings and hints in the code editor, new improvements to rendering to enhance visibility, and better support for packages, ctrl-click navigation, and much more.

EXPANDED WINDOWS SUPPORT
RAD Studio 10.4.2 delivers best-in-class Windows app support, library for native Windows app development, the Visual Component Library.
These include the new `TControlList` VCL control, a virtual and high-performance control for very long lists, and a new `VCL TNumberBox` control that supports integer, floating point and currency values. See issue page 9 and 3/17

Windows Store packaging is made easier through support for **MSIX,** Microsoft's newly recommended Windows application packaging format.

There is improved styles support for the Konopka Components suite of over 200 Windows UI controls and designers for your Delphi and C++ Builder VCL applications.

The `TEdgeBrowser VCL` component has been updated with support for the GA version of Microsoft's WebView2 control and its SDK.

The TNumberBox
The TNumberBox is a special kind of box that mainly does three things:
It can receive Integers
It can receive floats
and currency (can’t we all).

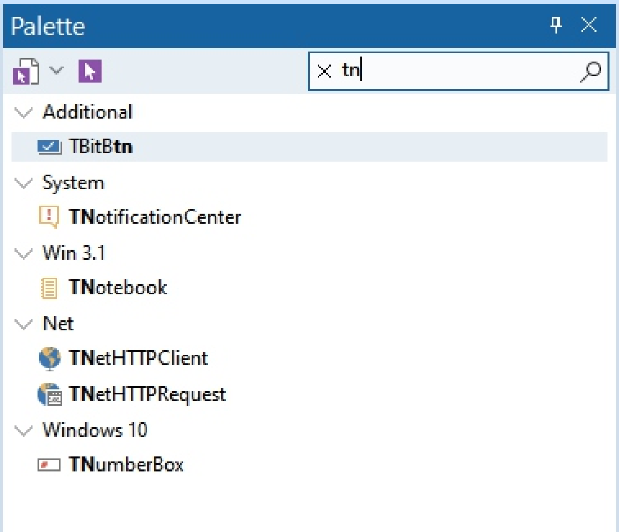
What makes this component very attractive is its flexibility: Just enter a simple expression into it and enter you'll have the answer instantly actually you could use it as calculator.

By the time I saw it there was no description or help yet or any explanation.
The new VCL TNumberBox control is a normal looking numeric input control and is designed after the Windows platform WinUI NumberBox control.

The component also allows simple expression evaluation;
through the dropdownbox you can add an expression like 12+2 +20*(1234) or anything else you would like to enter.
After using the enter key with the keyboard the control will replace it with the result.

On these pages we show several aspects of theproject Marco build:
On the next pages you see the way of letting the customer choose from Styles that are available.

Figure 1: the numberbox in the palette



Palette

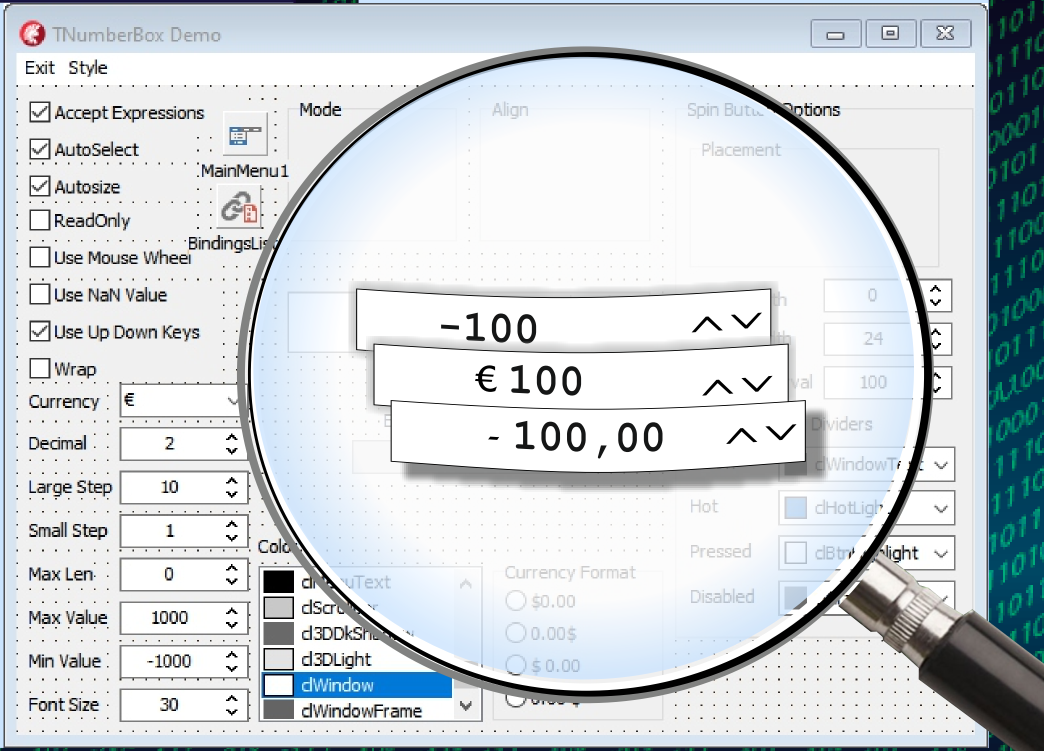
tn

- Additional
  - TBitBtn
- System
  - TNotificationCenter
- Win 3.1
  - TNotebook
- Net
  - TNetHTTPClient
  - TNetHTTPRequest
- Windows 10
  - TNumberBox

Figure 2: integers, currency and floats

The control supports computing inline calculations of basic equations such as multiplication, division, addition, and subtraction (allowing the use of brackets).

Notice that you can use the symbols + and - both as **binary** and **unary** operations, so you can type -73 or + 73, you can write 57+13 and 59-23, and even combine them as in 53++23 or 53--23, which is evaluated as 53 - (-23). thus adds the two values.

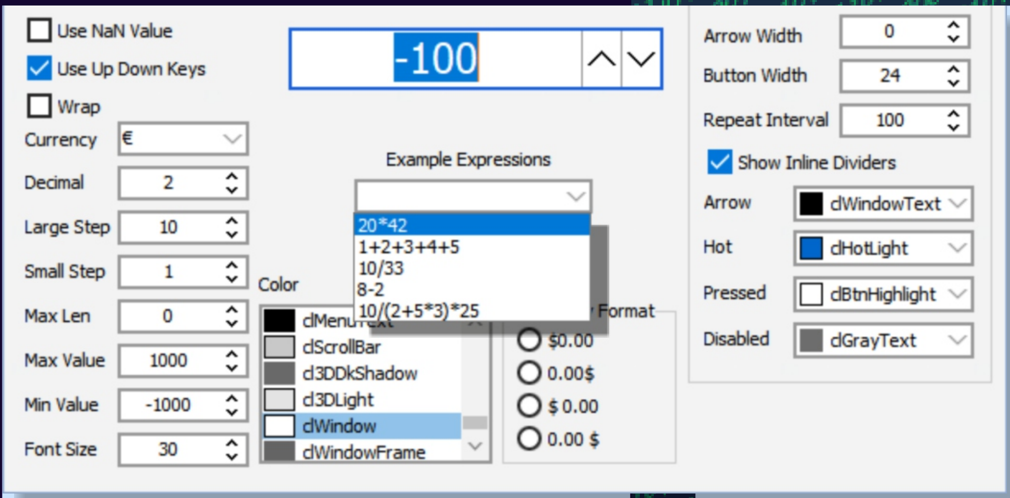


TNumberBox Demo

Exit Style

☑ Accept Expressions
☑ AutoSelect
☑ Autosize
☐ ReadOnly
☐ Use Mouse Wheel
☐ Use NaN Value
☑ Use Up Down Keys
☐ Wrap

Currency	€
Decimal	2
Large Step	10
Small Step	1
Max Len	0
Max Value	1000
Min Value	-1000
Font Size	30



☐ Use NaN Value
☑ Use Up Down Keys
☐ Wrap
Currency: €
Decimal: 2
Large Step: 10
Small Step: 1
Max Len: 0
Max Value: 1000
Min Value: -1000
Font Size: 30

-100

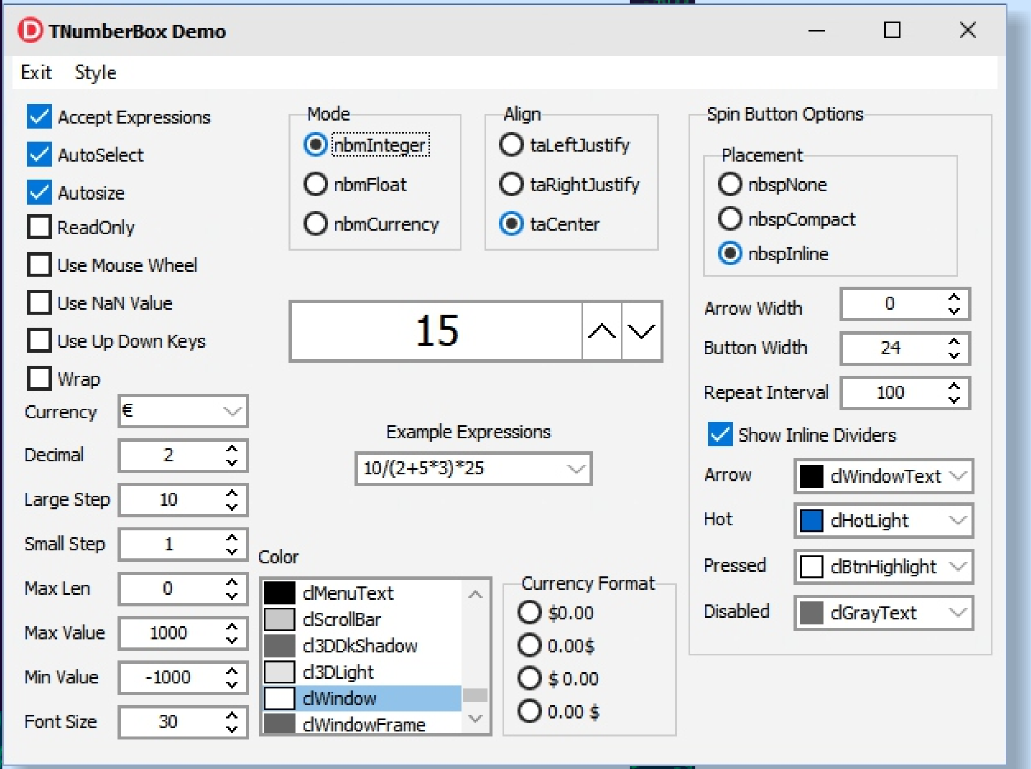
Example Expressions:
20*42
1+2+3+4+5
10/33
8-2
10/(2+5*3)*25

Color: clMenuText, clScrollBar, cl3DDkShadow, cl3DLight, clWindow, clWindowFrame

Format: ○ $0.00 ○ 0.00$ ○ $ 0.00 ○ 0.00 $

Arrow Width: 0
Button Width: 24
Repeat Interval: 100
☑ Show Inline Dividers
Arrow: clWindowText
Hot: clHotLight
Pressed: clBtnHighlight
Disabled: clGrayText

Figure 3: the example expressions



TNumberBox Demo

☑ Accept Expressions
☑ AutoSelect
☑ Autosize
☐ ReadOnly
☐ Use Mouse Wheel
☐ Use NaN Value
☐ Use Up Down Keys
☐ Wrap
Currency: €
Decimal: 2
Large Step: 10
Small Step: 1
Max Len: 0
Max Value: 1000
Min Value: -1000
Font Size: 30

Mode: ● nbmInteger ○ nbmFloat ○ nbmCurrency

Align: ○ taLeftJustify ○ taRightJustify ● taCenter

15

Example Expressions: 10/(2+5*3)*25

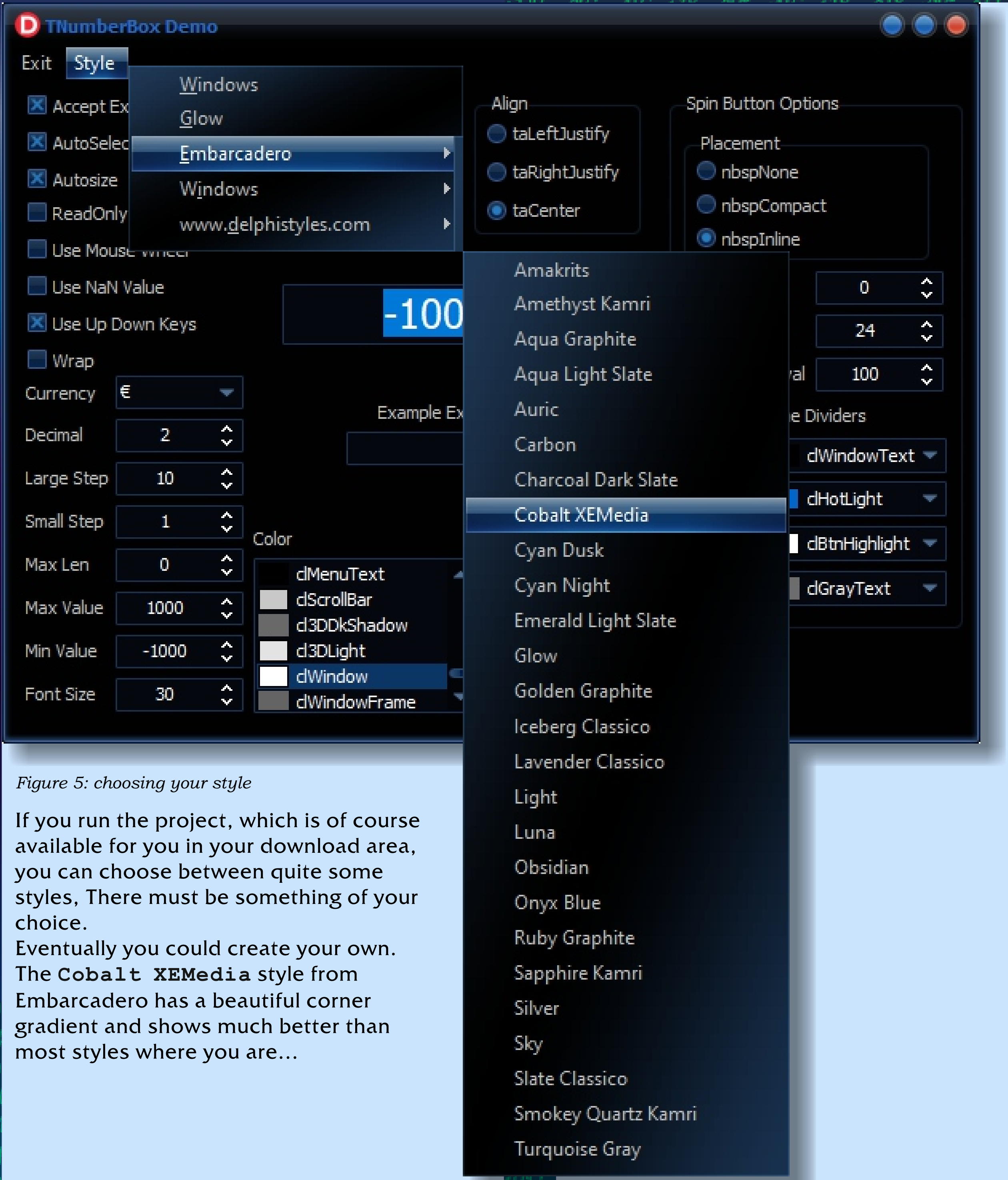
Color: clMenuText, clScrollBar, cl3DDkShadow, cl3DLight, clWindow, clWindowFrame

Currency Format: ○ $0.00 ○ 0.00$ ○ $ 0.00 ○ 0.00 $

Spin Button Options
Placement: ○ nbspNone ○ nbspCompact ● nbspInline
Arrow Width: 0
Button Width: 24
Repeat Interval: 100
☑ Show Inline Dividers
Arrow: clWindowText
Hot: clHotLight
Pressed: clBtnHighlight
Disabled: clGrayText

Figure 4: the executed expression

In figure 3 you see the listbox with example expressions you can use (make your own as well and on the fly). Below you can see what the outcome is of the chosen expression. Almost all of the properties are shown over here and might be changed if necessary".



☒ Accept Ex
☒ AutoSelec
☒ Autosize
□ ReadOnly
□ Use Mouse Wheel
□ Use NaN Value
☒ Use Up Down Keys
□ Wrap

Currency: €
Decimal: 2
Large Step: 10
Small Step: 1
Max Len: 0
Max Value: 1000
Min Value: -1000
Font Size: 30

-100

Align
○ taLeftJustify
○ taRightJustify
● taCenter

Spin Button Options
Placement
○ nbspNone
○ nbspCompact
● nbspInline

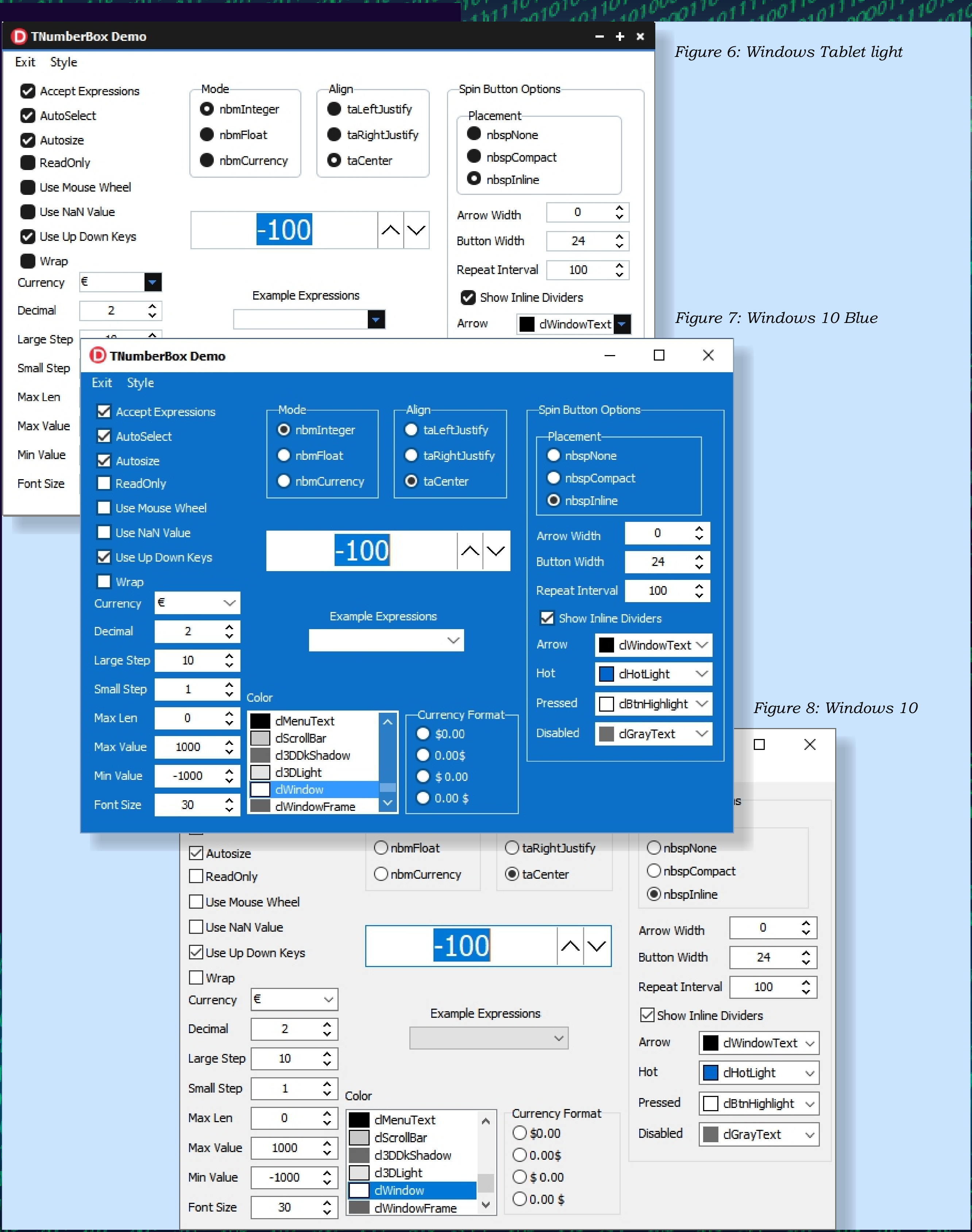
0
24
100

clWindowText
clHotLight
clBtnHighlight
clGrayText

clMenuText
clScrollBar
cl3DDkShadow
cl3DLight
clWindow
clWindowFrame

Figure 5: choosing your style

If you run the project, which is of course available for you in your download area, you can choose between quite some styles, There must be something of your choice.
Eventually you could create your own.
The **Cobalt XEMedia** style from Embarcadero has a beautiful corner gradient and shows much better than most styles where you are...



TNumberBox Demo

Exit Style

☑ Accept Expressions
☑ AutoSelect
☑ Autosize
□ ReadOnly
□ Use Mouse Wheel
□ Use NaN Value
☑ Use Up Down Keys
□ Wrap

Currency	€
Decimal	2
Large Step	10
Small Step	1
Max Len	0
Max Value	1000
Min Value	-1000
Font Size	30

Mode
● nbmInteger
○ nbmFloat
○ nbmCurrency

Align
○ taLeftJustify
○ taRightJustify
● taCenter

-100

Example Expressions

Color
clMenuText
clScrollBar
cl3DDkShadow
cl3DLight
clWindow
clWindowFrame

Currency Format
○ $0.00
○ 0.00$
○ $ 0.00
○ 0.00 $

Spin Button Options

Placement
○ nbspNone
○ nbspCompact
● nbspInline

Arrow Width	0
Button Width	24
Repeat Interval	100

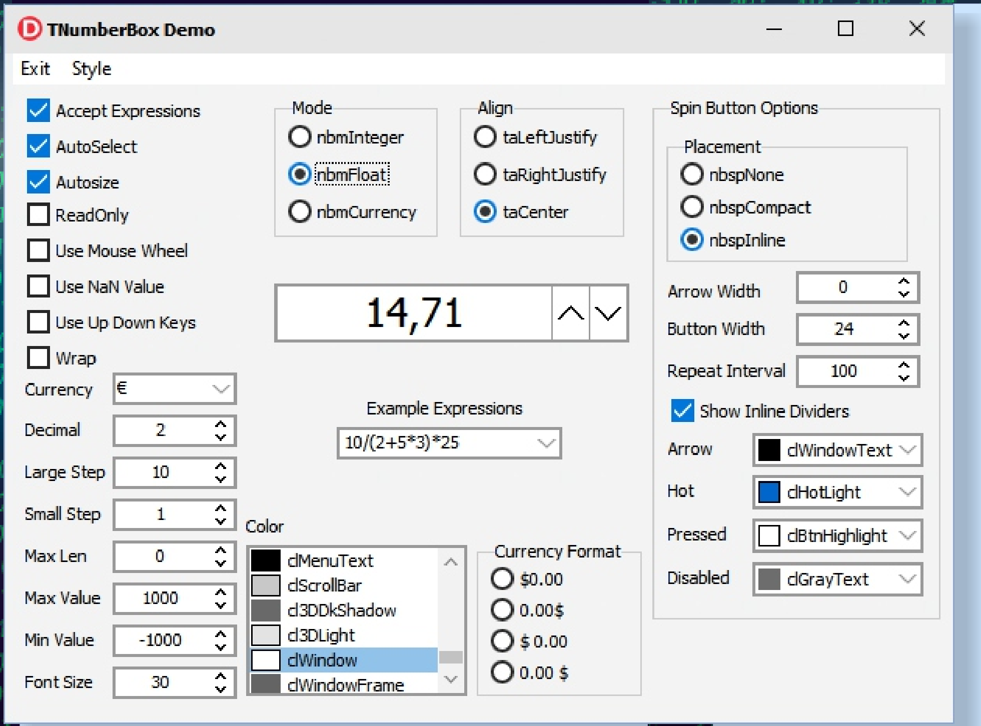
☑ Show Inline Dividers

Arrow	clWindowText
Hot	clHotLight
Pressed	clBtnHighlight
Disabled	clGrayText

Figure 6: Windows Tablet light

Figure 7: Windows 10 Blue

Figure 8: Windows 10



☑ Accept Expressions
☑ AutoSelect
☑ Autosize
☐ ReadOnly
☐ Use Mouse Wheel
☐ Use NaN Value
☐ Use Up Down Keys
☐ Wrap

Currency: €
Decimal: 2
Large Step: 10
Small Step: 1
Max Len: 0
Max Value: 1000
Min Value: -1000
Font Size: 30

Mode: ○ nbmInteger ● nbmFloat ○ nbmCurrency

Align: ○ taLeftJustify ○ taRightJustify ● taCenter

14,71

Example Expressions: 10/(2+5*3)*25

Color: clMenuText, clScrollBar, cl3DDkShadow, cl3DLight, clWindow, clWindowFrame

Currency Format: ○ $0.00 ○ 0.00$ ○ $ 0.00 ○ 0.00 $

Spin Button Options

Placement: ○ nbspNone ○ nbspCompact ● nbspInline

Arrow Width: 0
Button Width: 24
Repeat Interval: 100
☑ Show Inline Dividers
Arrow: clWindowText
Hot: clHotLight
Pressed: clBtnHighlight
Disabled: clGrayText

Figure 9: The executed program

```
procedure TNumBoxDemo.ComboBox2Change(Sender: TObject);
begin
 NumberBox1.AcceptExpressions := True;
 NumberBox1.SetFocus;
 NumberBox1.Text := ComboBox2.Text;
end;

procedure TNumBoxDemo.Exit1Click(Sender: TObject);
begin
 Application.Terminate;
end;
```

On the next page the complete code of the `onCreate`. The inline variables are non of my taste. This is Pascal. Not C# or whatever. This form of frivolous novelties should not be used. It confuses and in this way it loses the the strength of Pascal: clear view and clear structure. Know what is where. The benefit is only a few lines of typing, and actually that makes you aware of the construction of **Pascal.**

```
procedure TNumBoxDemo.rdoAlignClick(Sender: TObject);
begin
  NumberBox1.Alignment := TAlignment(rdoAlign.ItemIndex);
end;

procedure TNumBoxDemo.rdoCurrencyFormatClick(Sender: TObject);
begin
  NumberBox1.CurrencyFormat := rdoCurrencyFormat.ItemIndex;
end;

procedure TNumBoxDemo.rdoModeClick(Sender: TObject);
begin
  NumberBox1.Mode := TNumberBoxMode(rdoMode.ItemIndex);
end;

procedure TNumBoxDemo.rdoPlacementClick(Sender: TObject);
begin
  NumberBox1.SpinButtonOptions.Placement :=
   TNumberBoxSpinButtonPlacement(rdoPlacement.ItemIndex);
end;

procedure TNumBoxDemo.StyleMenuItemClick(Sender: TObject);
begin
  TStyleManager.TrySetStyle(TMenuItem(Sender).Caption.Replace('&','',[rfReplaceAll]));
end;
```

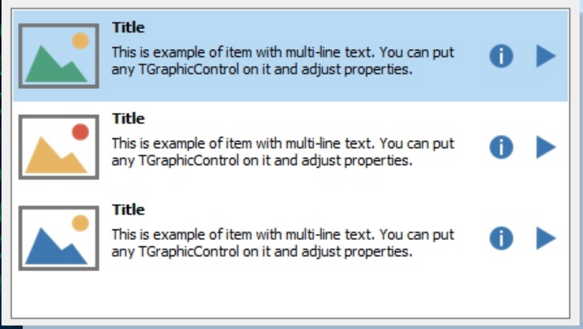
```
procedure TNumBoxDemo.FormCreate(Sender: TObject);
begin
   for var nbsp := Low(TNumberBoxSpinButtonPlacement)
      to  High(TNumberBoxSpinButtonPlacement)
   do  rdoPlacement.Items.Add( TRttiEnumerationType.GetName(nbsp) );

   rdoPlacement.ItemIndex := Ord(NumberBox1.SpinButtonOptions.Placement);

   for var nbm := Low(TNumberBoxMode) to High(TNumberBoxMode)
   do rdoMode.Items.Add( TRttiEnumerationType.GetName(nbm) );

   rdoMode.ItemIndex := Ord(NumberBox1.Mode);

   for var algn := Low(TAlignment) to High(TAlignment)
   do  rdoAlign.Items.Add( TRttiEnumerationType.GetName(algn) );

   rdoAlign.ItemIndex := Ord(NumberBox1.Alignment);

   for var sname in TStyleManager.StyleNames do
   begin
      var styleItem := TMenuItem.Create(mnuStyle);
      styleItem.Caption := sName;
      styleItem.OnClick := StyleMenuItemClick;
       mnuStyle.Add(styleItem);
   end;

   var Authors := TStringList.Create;
   for var fname in TDirectory.GetFiles('C:\Users\Public\Documents\Embarcadero\Studio\21.0\Styles','*.vsf')
   do begin
      var info: TStyleInfo;
      if TStyleManager.IsValidStyle(fname, info)
      then
         begin
            var Author: string;
            if info.Author.StartsWith('Embarcadero')
            then  Author := 'Embarcadero'
            else
              Author := info.Author;

            if info.Name.StartsWith('Metro')
            or info.Name.StartsWith('Tablet')
            or info.Name.StartsWith('Windows')
            then
              Author := 'Windows';
              authors.AddPair(Author, info.Name);
            try
              TStyleManager.LoadFromFile(fname);
              except on EDuplicateStyleException do
            end;
          end;
     end;
      authors.Sort;
   var authorItem: TMenuItem := nil;
   for var i := 0 to pred(authors.Count) do
   begin
   if not Assigned(authorItem) or (authorItem.Caption <> authors.Names[i]) then
      begin
         authorItem := TMenuItem.Create(mnuStyle);
         authorItem.AutoHotkeys := TMenuItemAutoFlag.maManual;
         authorItem.Caption := authors.Names[i];
         mnuStyle.Add(authorItem);
      end;
         var styleItem := TMenuItem.Create(mnuStyle);
         styleItem.AutoHotkeys := TMenuItemAutoFlag.maManual;
         styleItem.Caption := Authors.ValueFromIndex[i];
         styleItem.OnClick := StyleMenuItemClick;
         authorItem.Add(styleItem);
   end;
end;
```

starter expert

NEW VCL TCONTROLLIST CONTROL

Embarcadero is introducing in the VCL library a new flexible and virtualized list control.
The idea behind this control is to offer a new modern looking VCL control offering custom UI configuration and a high performance control, which can be used with very long lists. This list represents a single selection list and all items are visually of the same height and width.

The new control allows the developer to define the content by designing one of the elements of the list using graphical controls (that is, `TGraphicControl` descendants) and provide data to the control to display individual elements, without creating all of the controls for all of the elements in the list, but only those needed to display the data.
Being totally virtual, the list can handle thousands and even millions of items, offering extremely fast scrolling. Beside calculating and displaying only the items that fit on screen, **the list caches the content of the items using in-memory bitmaps.**



Title
This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

Title
This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

Title
This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

Figure 10: The example of the layout

The new control resembles the classic TDBCtrlGrid control -- there is a panel for controls, you put controls on it and have virtual items created at run time. **Unlike** **`DBCtrlGrid`** we can put only **`TGraphicControl`** on it and all items are virtual. Below you can see the control at design time (*with the surface of a single item available for editing*) and at runtime (*with the same content multiplied many times*).

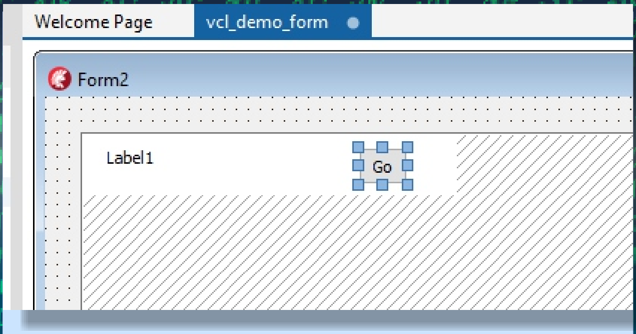
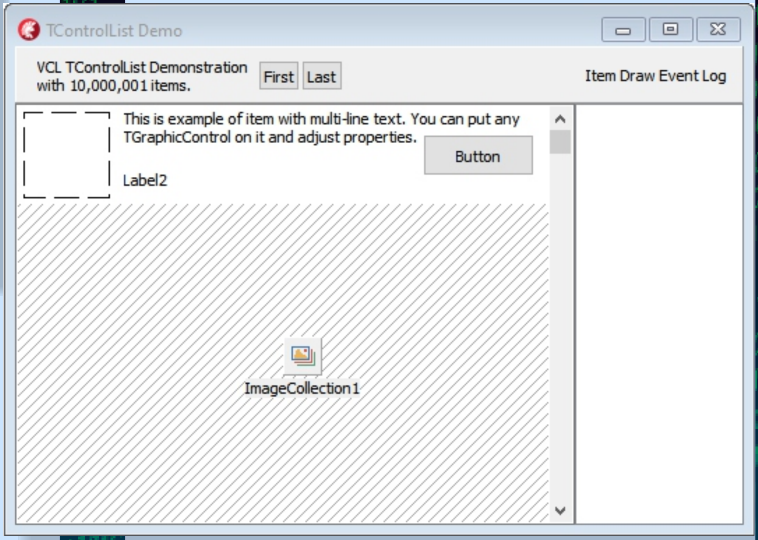


Figure 11: The projectof the vcl demo

This list doesn’t include a collection of items with specific information. The data might be provided either via live bindings (*including binding to a dataset or a collection of objects*) or via an event to query for the data of an individual item (*so that the direct storage and mapping would be completely up to the developer*). For each item to display the control calls an event handler you can use to customize each item appearance, in this case just modifying the Caption of the label:

```
procedure TForm2.ControlList1BeforeDrawItem(AIndex:
Integer; ACanvas: TCanvas; ARect: TRect;
Astate: TOwnerDrawState);
begin
  Label1.Caption := 'Label' + AIndex.ToString;
end;
```



VCL TControlList Demonstration with 10,000,001 items.

This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

Figure 12: Overview of all compoents

With the previous design, 10,000 items and multiple columns, this trivial code produces an output like below:

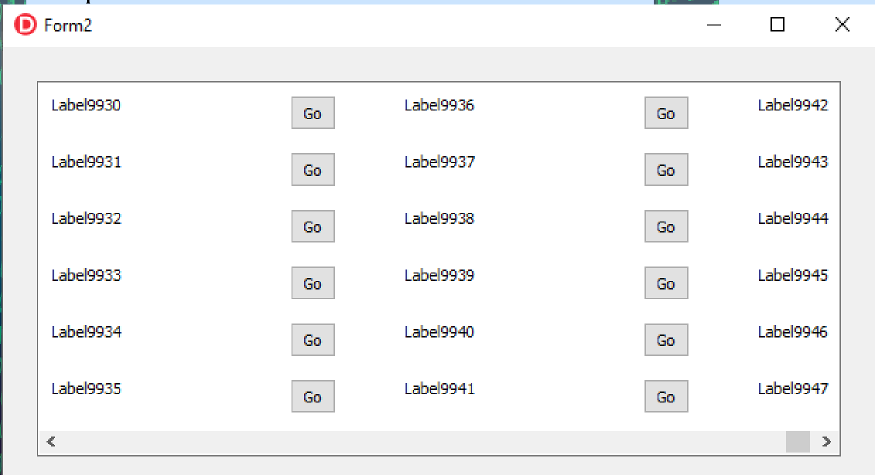


Figure 13: Overview of the form

At design time, there is special dialog with a collection of preset configurations, which include adjustment for **TControlList** properties and control collections with specific properties. You use the arrows at the top to pick the core configuration and you can fine tune it with some of the other checkbox options at the bottom. The wizard overrides the control list setting.

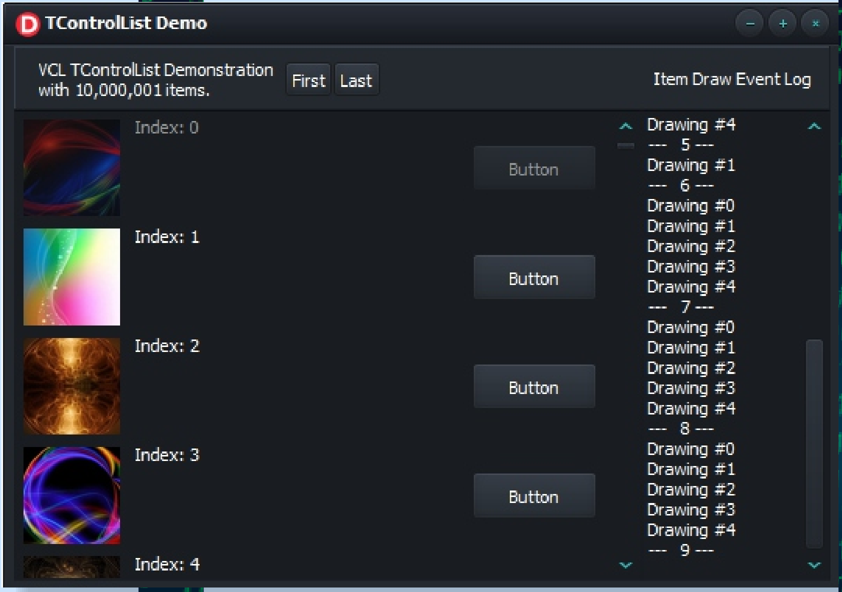
The item you design is replicated (virtually) for each of the items requested with the ItemCount property. The visible surface of the control generally allows for a number of items, all with the same width and height. The control has 3 different layouts:

In general terms, you can use the **OnClick** event for any control on the control list. The control supports **High-DPI options and VCL Styles** and it is fully Live Bindings enabled.

THE NEW TCONTROLLISTBUTTON COMPONENT

We can't use **TSpeedButton** directly on the panel, because the control doesn't handle special interactions like button changed state.

For controls, which can have different states, we added a special **TControlListControl** class (*inherited from* **TGraphicControl)** . You can create new controls that inherit from **TControlListControl** class and can use mouse events for their items. This is the approach used by **TControlListButton** - the analogue of a **TSpeedButton** that can be used with **TControlList**. This button has 3 styles - push button, tool button and link.

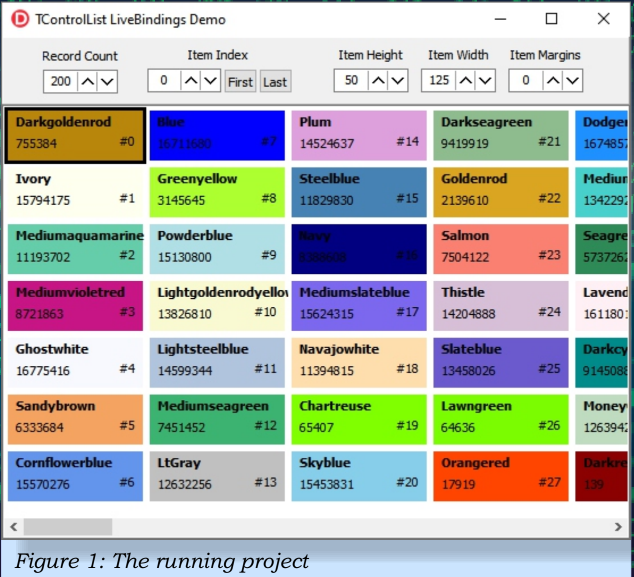


```
Drawing #4
--- 5 ---
Drawing #1
--- 6 ---
Drawing #0
Drawing #1
Drawing #2
Drawing #3
Drawing #4
--- 7 ---
Drawing #0
Drawing #1
Drawing #2
Drawing #3
Drawing #4
--- 8 ---
Drawing #0
Drawing #1
Drawing #2
Drawing #3
Drawing #4
--- 9 ---
```

Figure 13: The running application

starter expert

This new VCL TcontrolList is nothing but a very fats itself repeating control (with different images or labels) which has the ability to repeat itself and graphically render up to 10.000 times in milliseconds.
The refresh time is fantastic and this opens a possibility to create grids about subjects with various items which has never before been possible.
Yet there is no project description in the Helpfile. I suppose that soon will be arranged for. On their website is no real help-text available. So I suggest you simply have a look at our explanation and make use of the trial project.

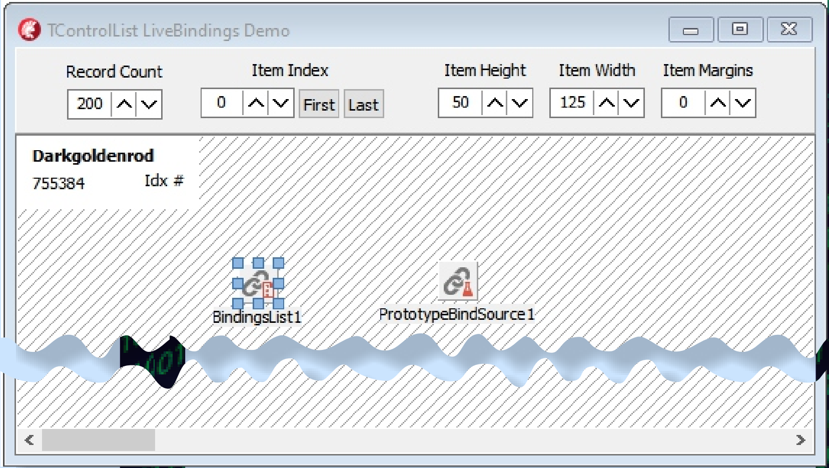

TControlList LiveBindings Demo

Record Count 200 · Item Index 0 · First · Last · Item Height 50 · Item Width 125 · Item Margins 0

Darkgoldenrod 755384 #0	Blue 16711680 #7	Plum 14524637 #14	Darkseagreen 9419919 #21	Dodge 1674857
Ivory 15794175 #1	Greenyellow 3145645 #8	Steelblue 11829830 #15	Goldenrod 2139610 #22	Mediun 1342292
Mediumaquamarine 11193702 #2	Powderblue 15130800 #9	Navy 8388608 #16	Salmon 7504122 #23	Seagre 5737262
Mediumvioletred 8721863 #3	Lightgoldenrodyellow 13826810 #10	Mediumslateblue 15624315 #17	Thistle 14204888 #24	Lavend 161180
Ghostwhite 16775416 #4	Lightsteelblue 14599344 #11	Navajowhite 11394815 #18	Slateblue 13458026 #25	Darkcy 9145088
Sandybrown 6333684 #5	Mediumseagreen 7451452 #12	Chartreuse 65407 #19	Lawngreen 64636 #26	Money 1263942
Cornflowerblue 15570276 #6	LtGray 12632256 #13	Skyblue 15453831 #20	Orangered 17919 #27	Darkre 139

Figure 1: The running project

First we start the project normally as seen in Figure 2, that is the way the project starts with. A number of labels are put on a panel, but that is not what it is all about. The shaded square containing in the upper left corner a white area with some text label on it and a number and something like a label with an idx#. We need to find out what's going on here. Since there is no help we simply do several things: take a look at the code, that might help finding out what does what?The first two procedures are obvious.


TControlList LiveBindings Demo

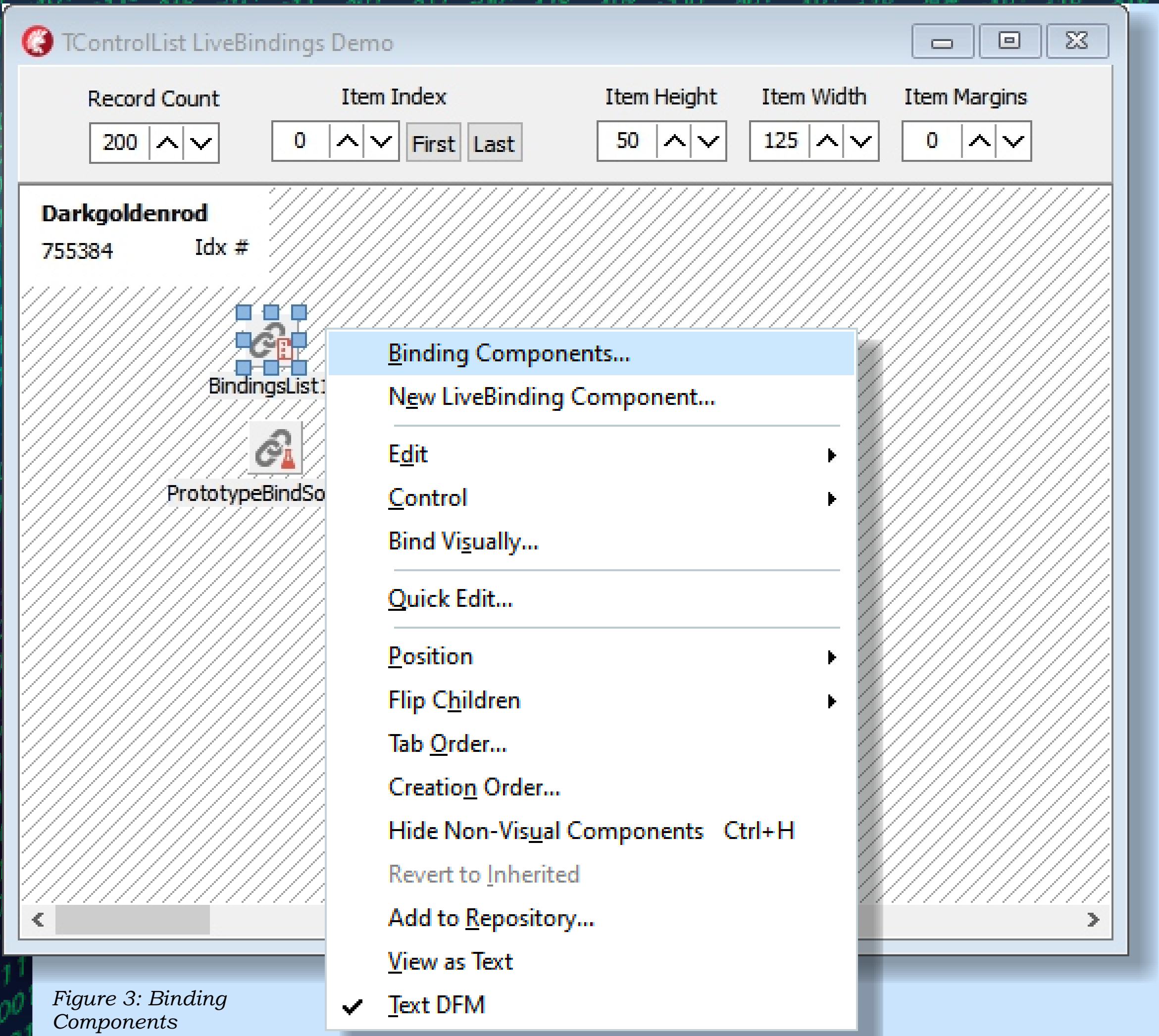
Record Count 200 · Item Index 0 · First · Last · Item Height 50 · Item Width 125 · Item Margins 0

Darkgoldenrod
755384 Idx #

Figure 2: The project

```
procedure TfrmCtrlListDemo3.btnFirstClick(Sender: TObject);
begin
  ControlList1.ItemIndex := 0;
end;

procedure TfrmCtrlListDemo3.btnLastClick(Sender: TObject);
begin
  ControlList1.ItemIndex := ControlList1.ItemCount;
end;

procedure TfrmCtrlListDemo3.ControlList1BeforeDrawItem(AIndex: Integer;
  ACanvas: TCanvas; ARect: TRect; AState: TOwnerDrawState);
begin
  ACanvas.Brush.Color   := StrToInt(lblColorValue.Caption);
  ACanvas.Pen.Width      := 5;
  if odSelected in AState
  then ACanvas.Pen.Color := clBlack
  else
     ACanvas.Pen.Color   := clWhite;
     ACanvas.Rectangle(ARect);
     lblIndex.Caption := Format('#%d',[AIndex]);
end;

procedure TfrmCtrlListDemo3.ControlList1ItemClick(Sender: TObject);
begin
  numIndex.Value := ControlList1.ItemIndex;
end;
```

The third procedure **TfrmCtrlListDemo3.ControlList1BeforeDrawItem** reacts to the **Acanvas.Brush.Color := StrToInt(lblColorValue.Caption);**
So here is a label involved.
Maybe it would be best to have alook at the objects and see what their properties are:
On the next page you will find an overview of using the right clicking method.
If there is any hint of use by that way we will find out.

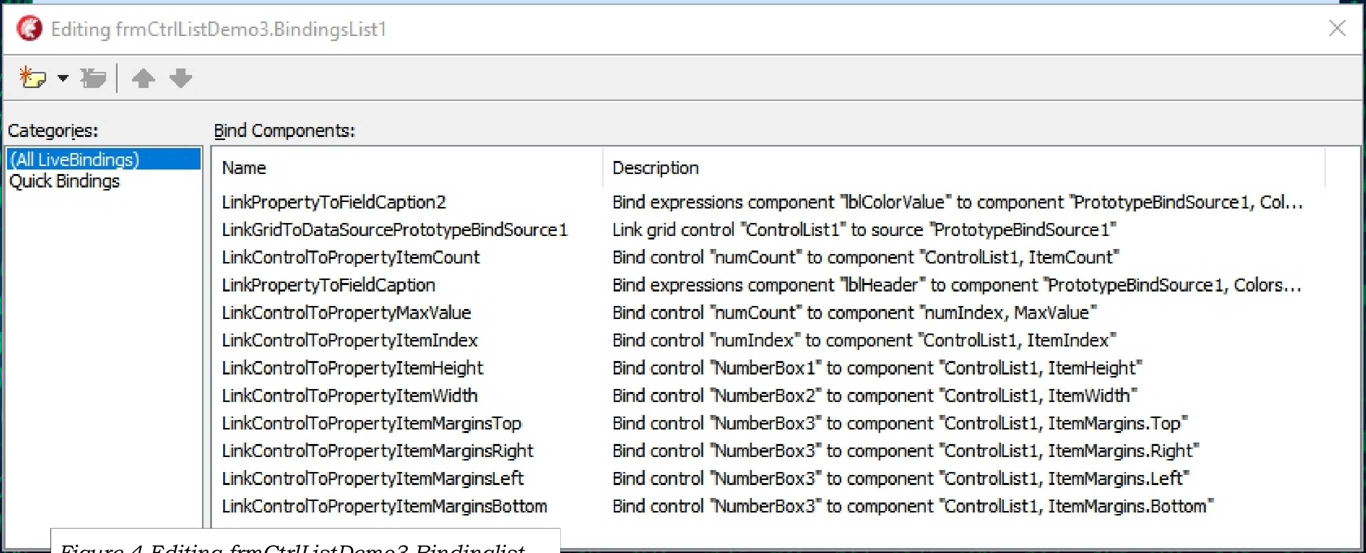


Record Count 200 | Item Index 0 First Last | Item Height 50 | Item Width 125 | Item Margins 0

Darkgoldenrod
755384 Idx #

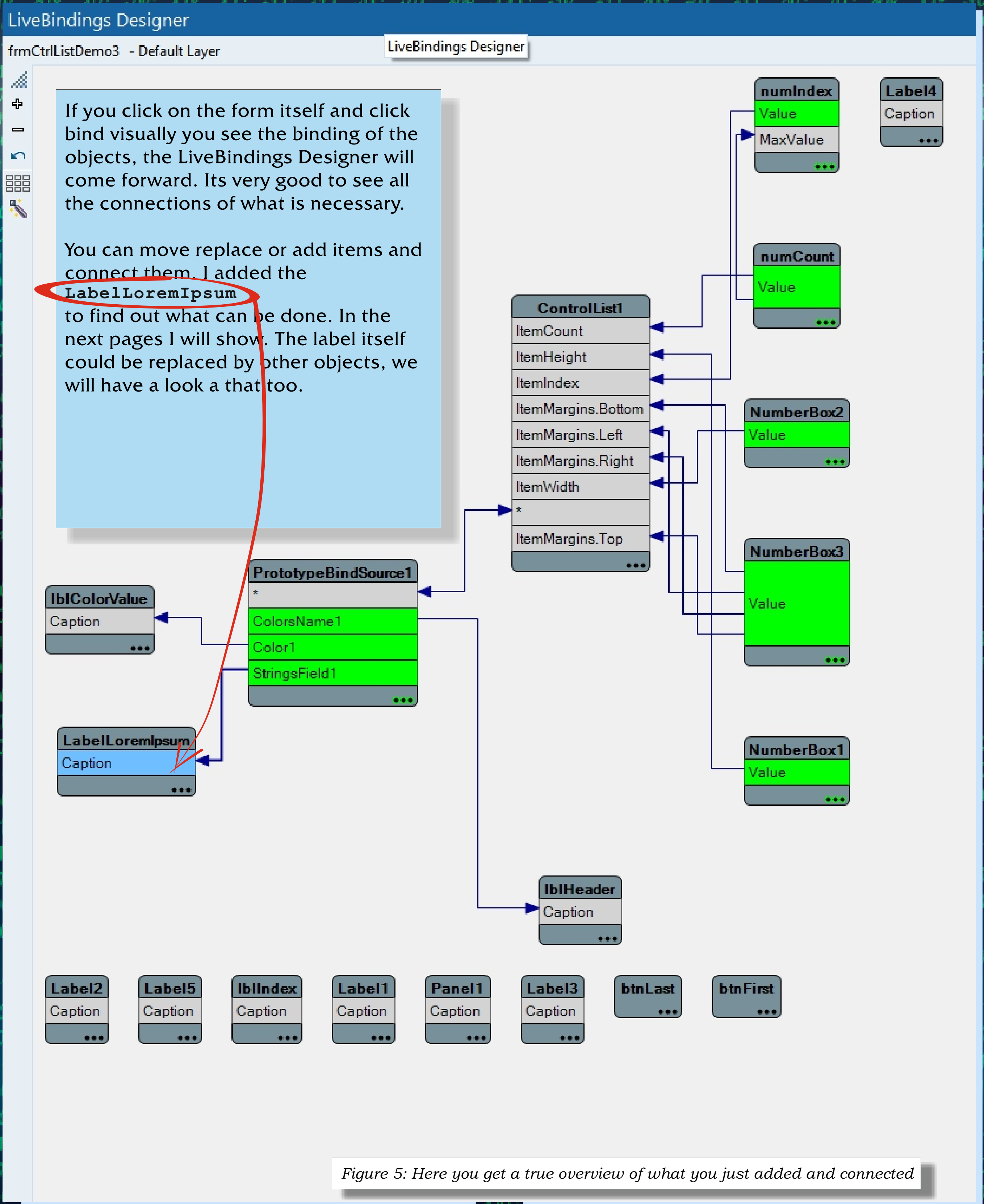
Figure 3: Binding Components

There is a Binding Components item. If we choose that A new Window pops up: The Editing `frmCtrlListDemo3.Bindinglist`. It shows all the links created by binding. See figure 4.



Name	Description
LinkPropertyToFieldCaption2	Bind expressions component "lblColorValue" to component "PrototypeBindSource1, Col...
LinkGridToDataSourcePrototypeBindSource1	Link grid control "ControlList1" to source "PrototypeBindSource1"
LinkControlToPropertyItemCount	Bind control "numCount" to component "ControlList1, ItemCount"
LinkPropertyToFieldCaption	Bind expressions component "lblHeader" to component "PrototypeBindSource1, Colors...
LinkControlToPropertyMaxValue	Bind control "numCount" to component "numIndex, MaxValue"
LinkControlToPropertyItemIndex	Bind control "numIndex" to component "ControlList1, ItemIndex"
LinkControlToPropertyItemHeight	Bind control "NumberBox1" to component "ControlList1, ItemHeight"
LinkControlToPropertyItemWidth	Bind control "NumberBox2" to component "ControlList1, ItemWidth"
LinkControlToPropertyItemMarginsTop	Bind control "NumberBox3" to component "ControlList1, ItemMargins.Top"
LinkControlToPropertyItemMarginsRight	Bind control "NumberBox3" to component "ControlList1, ItemMargins.Right"
LinkControlToPropertyItemMarginsLeft	Bind control "NumberBox3" to component "ControlList1, ItemMargins.Left"
LinkControlToPropertyItemMarginsBottom	Bind control "NumberBox3" to component "ControlList1, ItemMargins.Bottom"

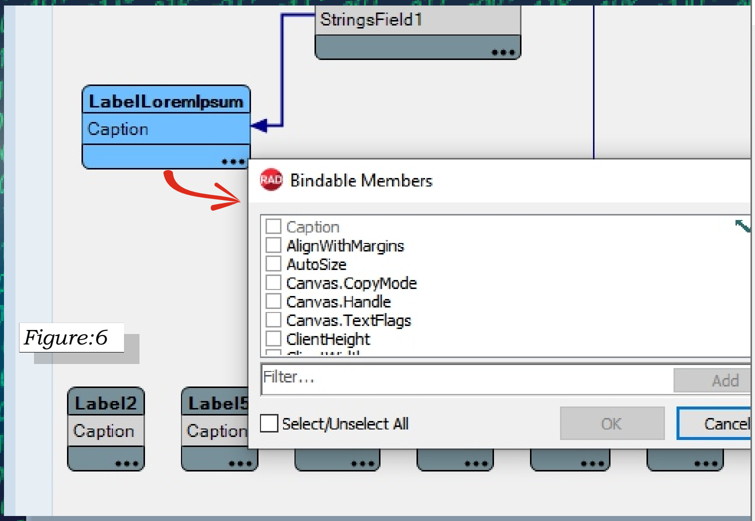
Figure 4 Editing frmCtrlListDemo3.Bindinglist



If you click on the form itself and click bind visually you see the binding of the objects, the LiveBindings Designer will come forward. Its very good to see all the connections of what is necessary.

You can move replace or add items and connect them. I added the `LabelLoremIpsum` to find out what can be done. In the next pages I will show. The label itself could be replaced by other objects, we will have a look a that too.

Figure 5: Here you get a true overview of what you just added and connected



□ Caption
□ AlignWithMargins
□ AutoSize
□ Canvas.CopyMode
□ Canvas.Handle
□ Canvas.TextFlags
□ ClientHeight

□ Select/Unselect All

Figure:6

If you select the **LabelLoremIpsum** (I named it after its field value) and again right-click this too, you have the next form available: Bindable Members. *Figure 6*).

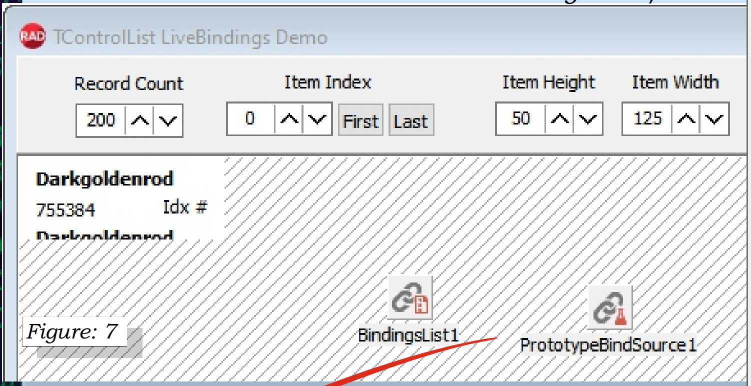


Figure: 7

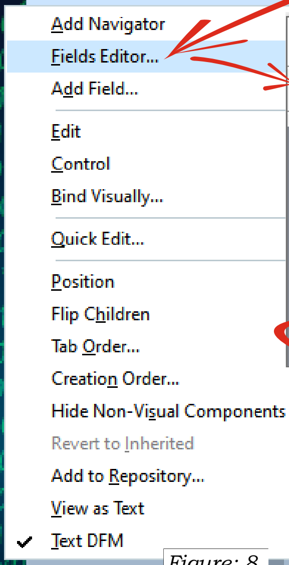
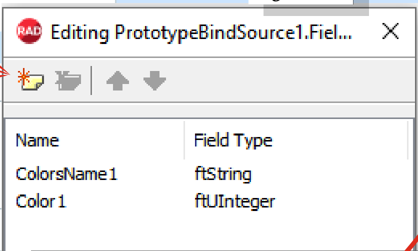
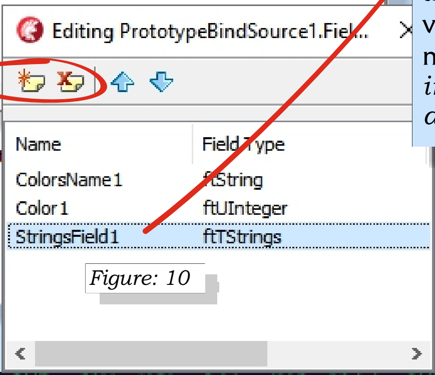


Figure: 8



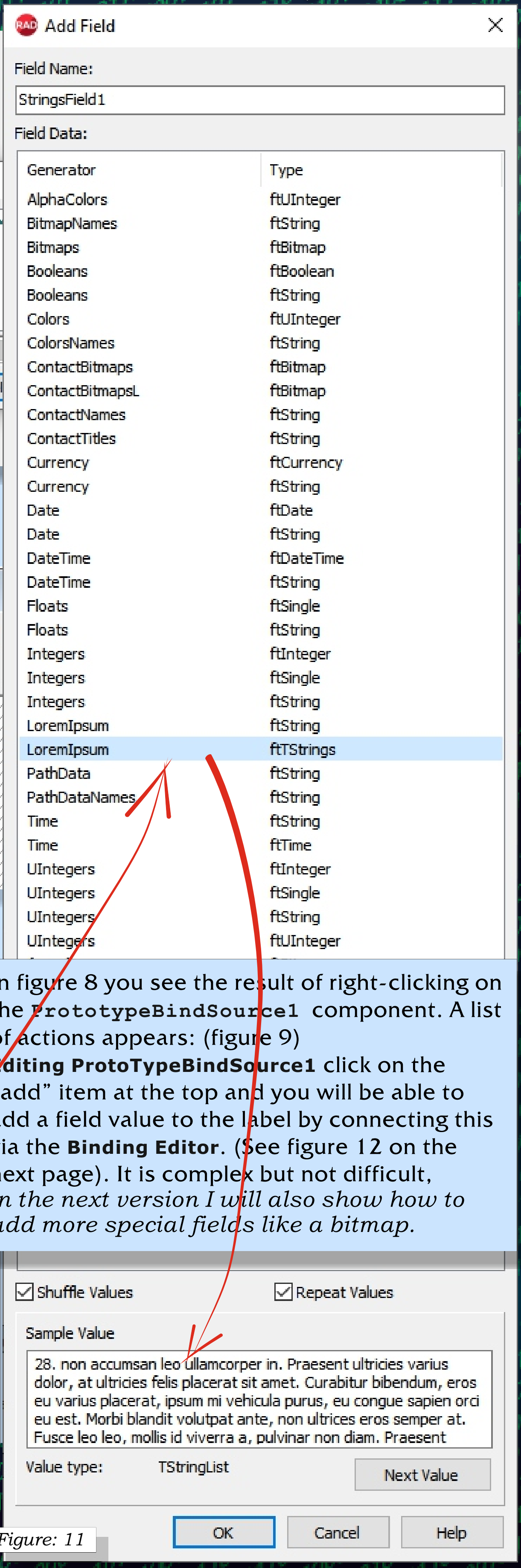
Name	Field Type
ColorsName1	ftString
Color1	ftUInteger

Figure: 9



Name	Field Type
ColorsName1	ftString
Color1	ftUInteger
StringsField1	ftTStrings

Figure: 10



Field Name: StringsField1

Field Data:

Generator	Type
AlphaColors	ftUInteger
BitmapNames	ftString
Bitmaps	ftBitmap
Booleans	ftBoolean
Booleans	ftString
Colors	ftUInteger
ColorsNames	ftString
ContactBitmaps	ftBitmap
ContactBitmapsL	ftBitmap
ContactNames	ftString
ContactTitles	ftString
Currency	ftCurrency
Currency	ftString
Date	ftDate
Date	ftString
DateTime	ftDateTime
DateTime	ftString
Floats	ftSingle
Floats	ftString
Integers	ftInteger
Integers	ftSingle
Integers	ftString
LoremIpsum	ftString
LoremIpsum	ftTStrings
PathData	ftString
PathDataNames	ftString
Time	ftString
Time	ftTime
UIntegers	ftInteger
UIntegers	ftSingle
UIntegers	ftString
UIntegers	ftUInteger

☑ Shuffle Values ☑ Repeat Values

Sample Value

28. non accumsan leo ullamcorper in. Praesent ultricies varius dolor, at ultricies felis placerat sit amet. Curabitur bibendum, eros eu varius placerat, ipsum mi vehicula purus, eu congue sapien orci eu est. Morbi blandit volutpat ante, non ultrices eros semper at. Fusce leo leo, mollis id viverra a, pulvinar non diam. Praesent

Value type: TStringList

Figure: 11

In figure 8 you see the result of right-clicking on the **PrototypeBindSource1** component. A list of actions appears: (figure 9) **Editing ProtoTypeBindSource1** click on the "add" item at the top and you will be able to add a field value to the label by connecting this via the **Binding Editor**. (See figure 12 on the next page). It is complex but not difficult, *in the next version I will also show how to add more special fields like a bitmap.*

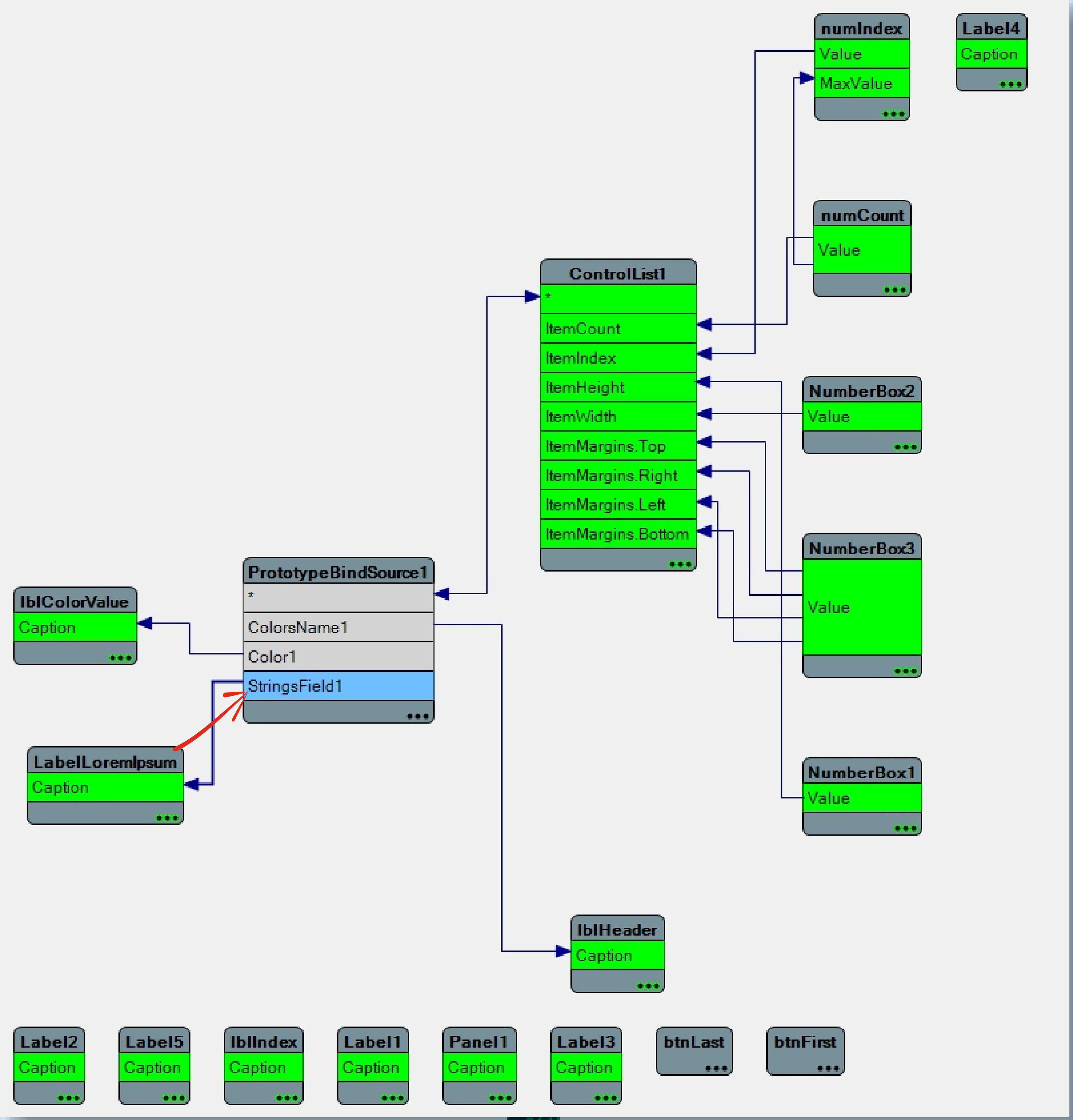
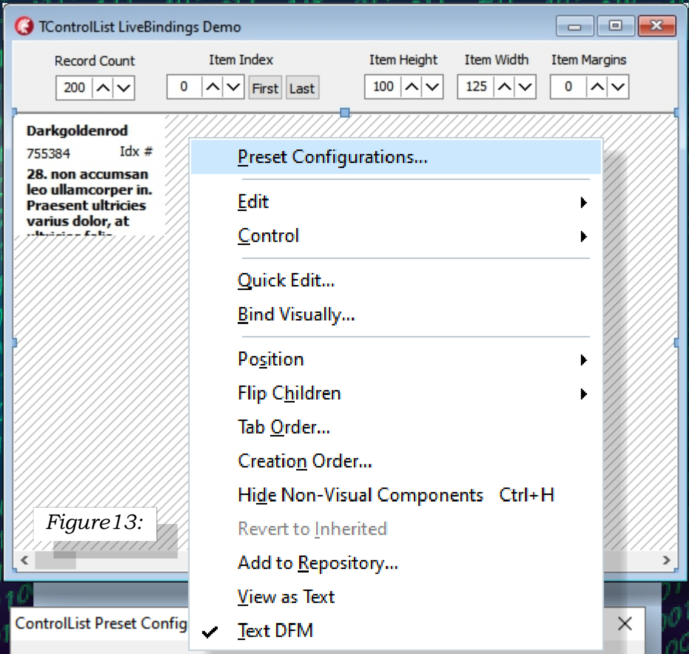


Figure12: See the structure of the fields

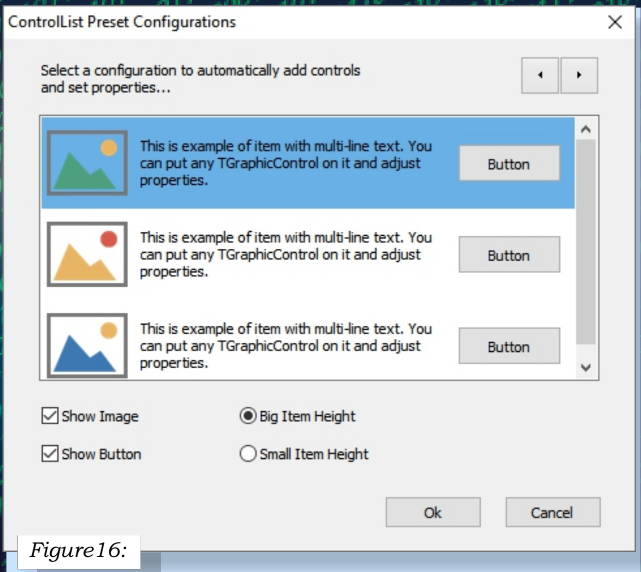


TControlList LiveBindings Demo

Record Count: 200
Item Index: 0
Item Height: 100
Item Width: 125
Item Margins: 0

Darkgoldenrod
755384 Idx #
28. non accumsan leo ullamcorper in. Praesent ultricies varius dolor, at

Figure13:



ControlList Preset Configurations

Select a configuration to automatically add controls and set properties...

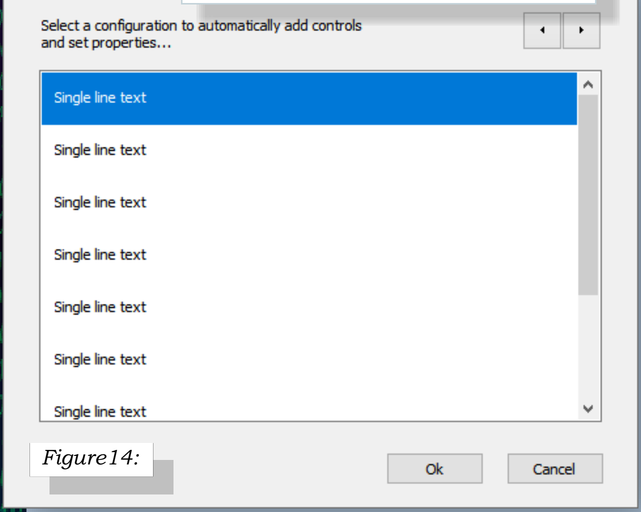
This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

☑ Show Image
☑ Show Button
● Big Item Height
○ Small Item Height

Figure16:

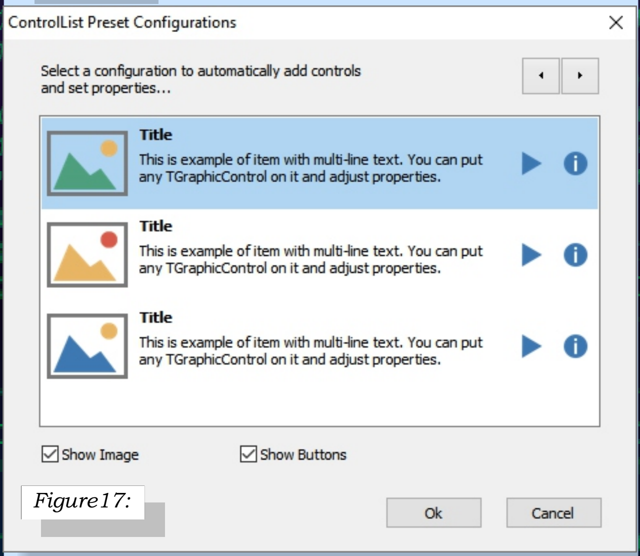


ControlList Preset Config

Select a configuration to automatically add controls and set properties...

Single line text
Single line text
Single line text
Single line text
Single line text
Single line text
Single line text

Figure14:



ControlList Preset Configurations

Select a configuration to automatically add controls and set properties...

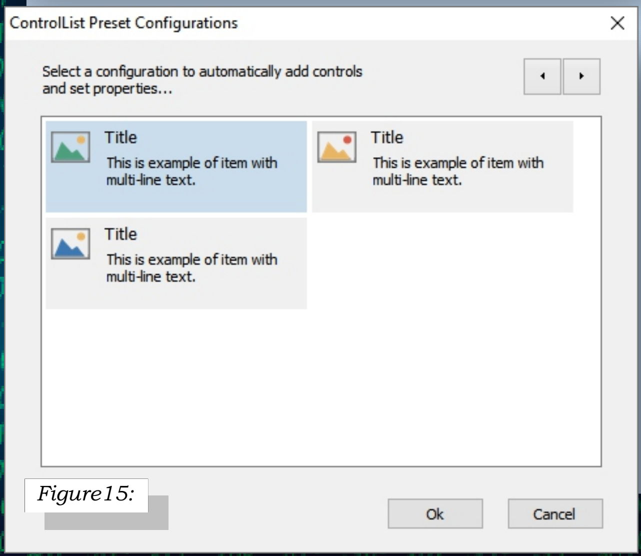
Title
This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

Title
This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

Title
This is example of item with multi-line text. You can put any TGraphicControl on it and adjust properties.

☑ Show Image
☑ Show Buttons

Figure17:



ControlList Preset Configurations

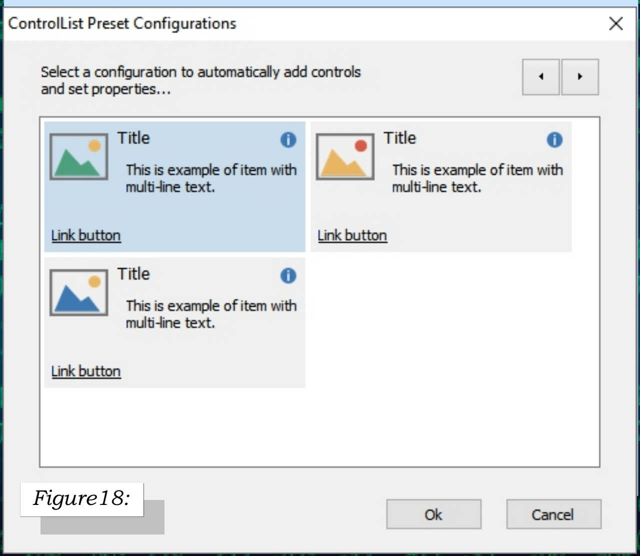
Select a configuration to automatically add controls and set properties...

Title
This is example of item with multi-line text.

Title
This is example of item with multi-line text.

Title
This is example of item with multi-line text.

Figure15:



ControlList Preset Configurations

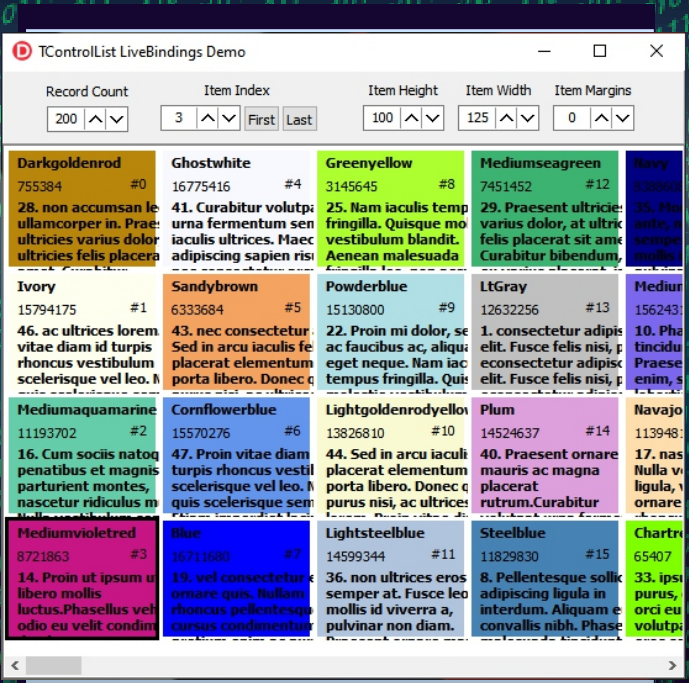
Select a configuration to automatically add controls and set properties...

Title
This is example of item with multi-line text.
Link button

Title
This is example of item with multi-line text.
Link button

Title
This is example of item with multi-line text.
Link button

Figure18:


TControlList LiveBindings Demo

Record Count 200
Item Index 3

Item Height 100
Item Width 125
Item Margins 0

Darkgoldenrod
755384 #0
28. non accumsan le
ullamcorper in. Prae:
ultricies varius dolor
ultricies felis placera

Ivory
15794175 #1
46. ac ultrices lorem.
vitae diam id turpis
rhoncus vestibulum
scelerisque vel leo. N

Mediumaquamarine
11193702 #2
16. Cum sociis natoq
penatibus et magnis
parturient montes,
nascetur ridiculus m

Mediumvioletred
8721863 #3
14. Proin ut ipsum u
libero mollis
luctus.Phasellus vel
odio eu velit condim

Ghostwhite
16775416 #4
41. Curabitur volutpa
urna fermentum sen
iaculis ultrices. Maec
adipiscing sapien risı

Sandybrown
6333684 #5
43. nec consectetur
Sed in arcu iaculis fe
placerat elementum
porta libero. Donec q

Cornflowerblue
15570276 #6
47. Proin vitae diam
turpis rhoncus vestil
scelerisque vel leo. N
quis scelerisque sem

Blue
16711680 #7
19. vel consectetur
ornare quis. Nullam
rhoncus pellentesqu
cursus condimentur

Greenyellow
3145645 #8
25. Nam iaculis temp
fringilla. Quisque mo
vestibulum blandit.
Aenean malesuada

Powderblue
15130800 #9
22. Proin mi dolor, se
ac faucibus ac, aliqu
eget neque. Nam iac
tempus fringilla. Quis

Lightgoldenrodyellow
13826810 #10
44. Sed in arcu iaculi
placerat elementum
porta libero. Donec q
purus nisi, ac ultrices

Lightsteelblue
14599344 #11
36. non ultrices eros
semper at. Fusce leo
mollis id viverra a,
pulvinar non diam.

Mediumseagreen
7451452 #12
29. Praesent ultricie:
varius dolor, at ultric
felis placerat sit ame
Curabitur bibendum,

LtGray
12632256 #13
1. consectetur adipis
elit. Fusce felis nisi, p
econsectetur adipisc
elit. Fusce felis nisi, p

Plum
14524637 #14
40. Praesent ornare
mauris ac magna
placerat
rutrum.Curabitur

Steelblue
11829830 #15
8. Pellentesque sollic
adipiscing ligula in
interdum. Aliquam e
convallis nibh. Phase

Navy
838860
35. Ma
ante, a
semper
mollis

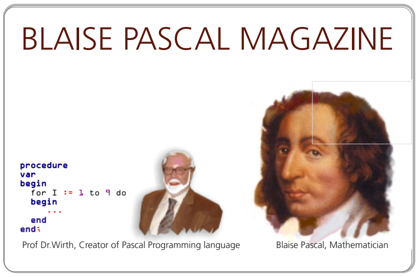
Mediur
156243
10. Pha
tincidu
Praese
enim, s

Navajo
113948
17. nas
Nulla v
ligula, v
ornare

Chartr
65407
33. ipsu
purus,
orci eu
volutpa

The **Preset Configurations** are very interesting Because they have the ability to let you have a single line configuration. (*Figure14*).
The next one is to automatically add controls of an item with multi- line text (*Figure 15*) and an other one is to create an item with a `TGraphicControl` and arrange settings for that case. (*Figure 16*).
Figure 17 is similar and Figure 18 with a Url-Link Button. All figures are available on page: article 17/17.

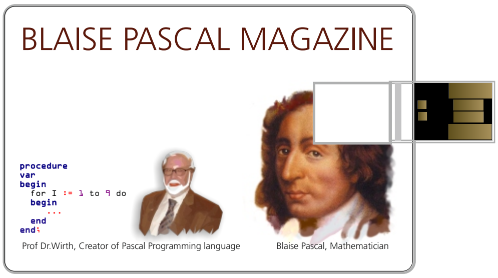
Advertisement


BLAISE PASCAL MAGAZINE
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician

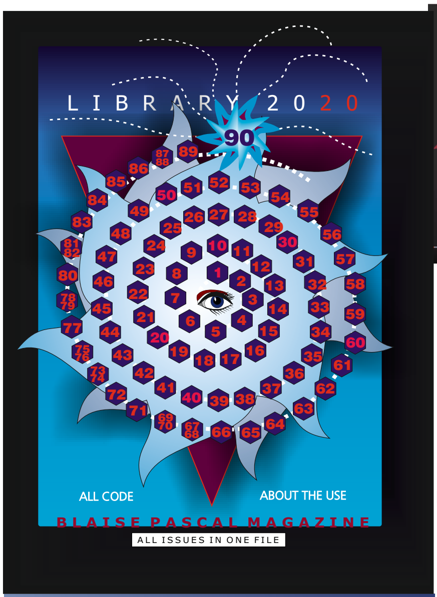
```
procedure
var
begin
  for I := 1 to 9 do
  begin
    ...
  end
end;
```

Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands

editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu

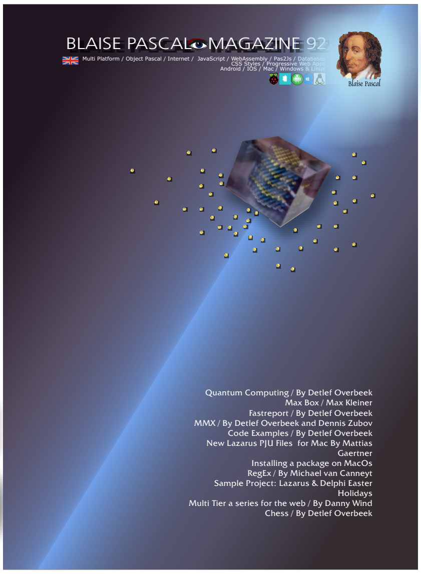

BLAISE PASCAL MAGAZINE
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician

```
procedure
var
begin
  for I := 1 to 9 do
  begin
    ...
  end
end;
```

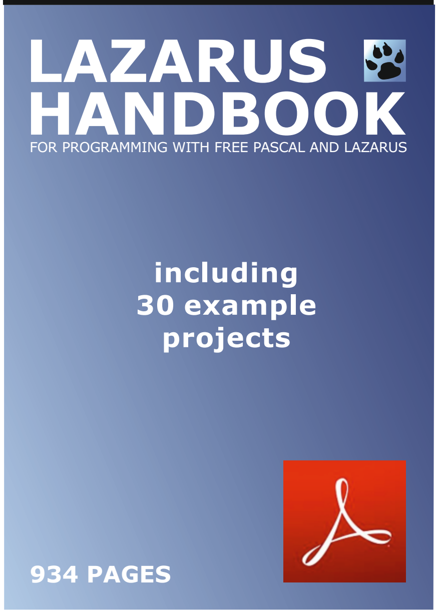


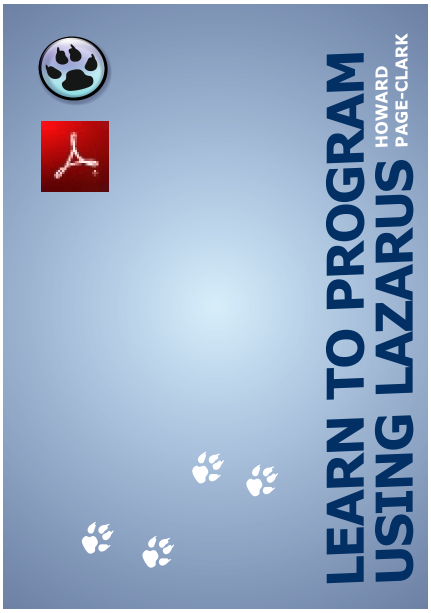
LIBRARY 2020
Programming Language
PASCAL
for Delphi and Lazarus
VIDEO
Object Pascal / Internet / JavaScript / WebAssembly
Pas2Js / Databases / CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux
BLAISE PASCAL MAGAZINE
0:01 / 7:04

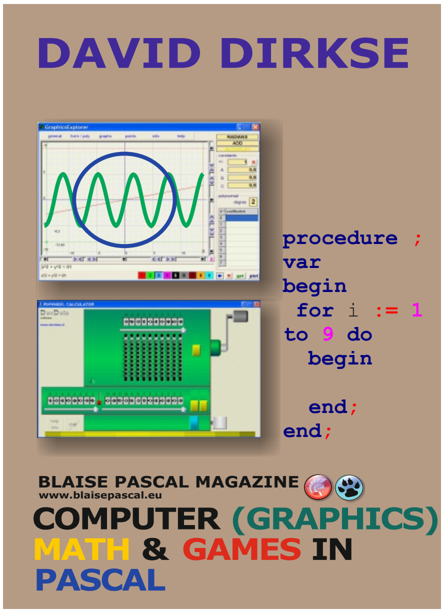
SUPER OFFER (5)
€ 150 ex Vat
free shipping


BLAISE PASCAL MAGAZINE 92
Multi Platform / Object Pascal / Internet / JavaScript / WebAssembly / Pas2Js / Databases
CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux
Blaise Pascal

Quantum Computing / By Detlef Overbeek
Max Box / Max Kleiner
Fastreport / By Detlef Overbeek
MMX / By Detlef Overbeek and Dennis Zubov
Code Examples / By Detlef Overbeek
New Lazarus PJU Files for Mac By Mattias Gaertner
Installing a package on MacOs
RegEx / By Michael van Canneyt
Sample Project: Lazarus & Delphi Easter Holidays
Multi Tier a series for the web / By Danny Wind
Chess / By Detlef Overbeek

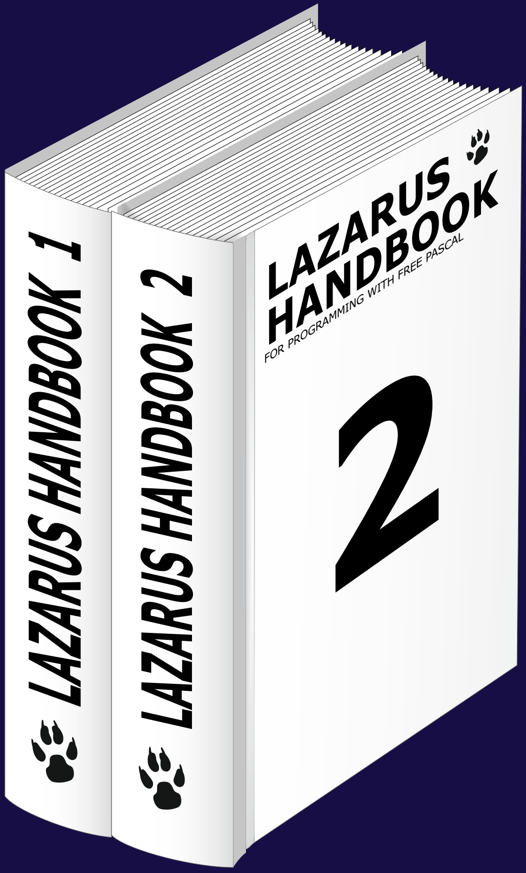

LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL AND LAZARUS
including 30 example projects
934 PAGES





```
procedure ;
var
begin
 for i := 1
to 9 do
   begin

   end;
end;
```

1. One year **Subscription**
2. **The newest LIB Stick**
   - including Credit Card USB stick
3. **Lazarus Handbook** - Personalized
   -PDF including Code
4. Book **Learn To Program** using Lazarus PDF including 19 lessons and projects
5. **Book Computer Graphics Math & Games** book + PDF including ±50 projects


LAZARUS HANDBOOK 1
LAZARUS HANDBOOK 2
LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL
2

BBC News Sentiment Analysis

maXbox

Author: Max Kleiner

„Yesterday I was clever,
so I wanted to change the world.

Today I am wise,
so I am changing myself." - Rumi

As you way know, we went through the last magazine report on the BBC News feed, line by line; now we want to equip and enlarge these texts with a sentiment analysis like the following:

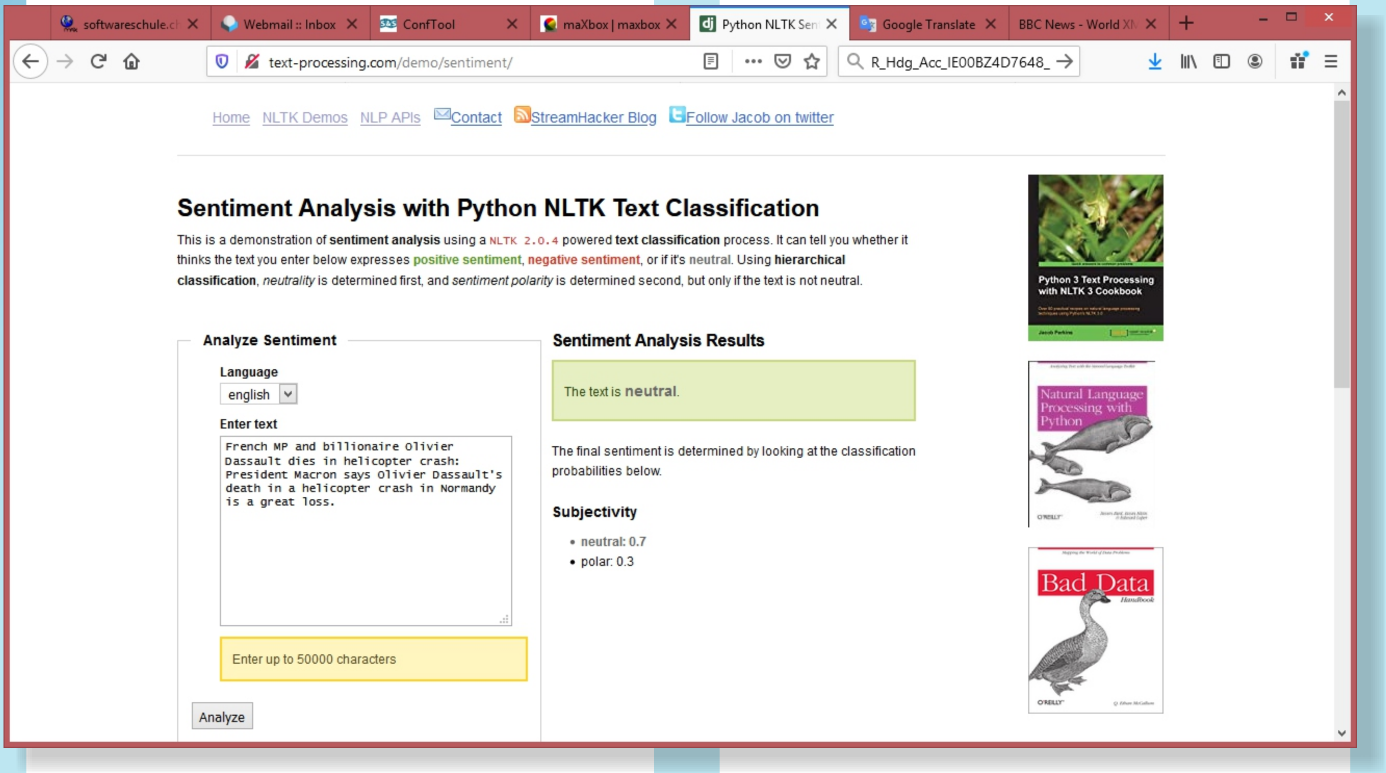
```
11: French MP and billionaire Olivier
Dassault dies in helicopter crash:
Sun, 07 Mar 2021 20:04:17 GMT
{"probability": {"neg":
0.46705201547833042, "neutral":
0.81510771060379195, "pos":
0.53294798452166958}, "label":
"neutral"}
```

So the label shows neutral in the middle of possibilities. The english sentiment uses classifiers trained on both twitter sentiment as well as movie reviews from the data sets. The dutch and french sentiment is based on book reviews.

Because of the nature of the text and its categories, the classification we will be doing is a form of sentiment analysis or opinion mining. If the classifier returns pos, then the text expresses a positive sentiment, whereas if we get neg, then the text expresses a negative sentiment.

This procedure of discovering and classifying opinions expressed in a piece of text *(like comments/feedback text/news feed in our case*) is called the sentiment analysis. The intended output of this analysis would be to determine whether the producers mindset toward a topic, product, headline or service etc., is in most cases neutral, positive, or negative.
Let's get started with the use of an API:

```
Const
  URLSentimentAPI2=
  'http://text-processing.com/
                   api/sentiment/';
```



Sentiment Analysis with Python NLTK Text Classification

This is a demonstration of **sentiment analysis** using a `NLTK 2.0.4` powered **text classification** process. It can tell you whether it thinks the text you enter below expresses **positive sentiment**, **negative sentiment**, or if it's neutral. Using **hierarchical classification**, *neutrality* is determined first, and *sentiment polarity* is determined second, but only if the text is not neutral.

Analyze Sentiment

Language: english

Enter text:

```
French MP and billionaire Olivier
Dassault dies in helicopter crash:
President Macron says Olivier Dassault's
death in a helicopter crash in Normandy
is a great loss.
```

Enter up to 50000 characters

Analyze

Sentiment Analysis Results

The text is **neutral**.

The final sentiment is determined by looking at the classification probabilities below.

Subjectivity

- neutral: 0.7
- polar: 0.3

Python 3 Text Processing with NLTK 3 Cookbook
Jacob Perkins
Natural Language Processing with Python
O'REILLY
Bad Data Handbook
O'REILLY

Our text lines are based on the BBC-News Feed:

```
BCFeed =
 'http://feeds.bbci.co.uk/
news/world/rss.xml';
```

To call the API we use a late binding OLE Automation from `<msxml2.xmlhttp>`.
The text-processing.com API is a simple JSON over HTTP web service for text mining and natural language processing. It is currently free and open for public use without authentication or login, but could be changed in the future. As of JSON we use the delphi4json library to parse the return.

The script you can find at: http://www.softwareschule.ch/examples/newssentiment2.txt

```
XMLhttp:= CreateOleObject('msxml2.xmlhttp')
XMLhttp.Open('POST', URLSentimentAPI2, False)  //False: async
XMLhttp.setRequestHeader('Content-Type','application/json');
XMLhttp.Send('text='+'"'+textin+'"'+CRLF+'language=english');
response:= XMLhttp.responseText; //assign data
writeln(response)
writeln('statuscode: '+itoa(XMLhttp.status;))
XMLhttp:= unassigned;
```

On success, a 200 OK response will be returned containing a JSON object that looks like:

```
{"probability":
  {"neg": 0.37517484595971884,
   "neutral": 0.091034274541377691,
   "pos": 0.62482515404028116},
   "label": "pos"}
```

A 503 Throttled response will be returned if you exceed the daily request limit. Using async = false in Open() is not always recommended, but for a few small requests this can be ok. Remember that the script will NOT continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop.
Anyway we open our XMLhttpObject (which is late binding) as not asynchronous, that is to say, synchronous because we just post a single block of data with send().

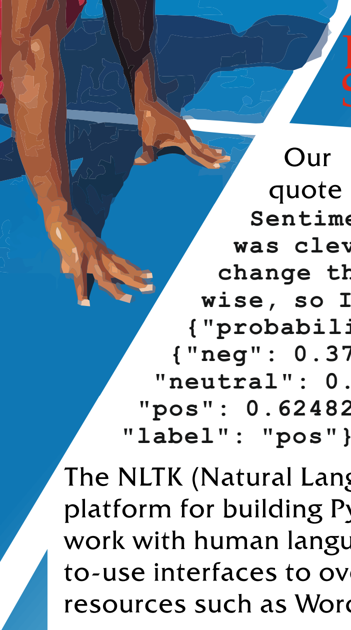
The send() method (XMLhttp.Send();) needs more explanation: send() accepts an optional parameter which lets you specify the requests body; this is primarily used for requests such as PUT or POST. If the request method is GET or HEAD, the body parameter is ignored and the request body is set to null. Im not sure if the content-type is the right (text or application), the MIME media type for JSON text is application/json and the default encoding is UTF-8. (Source: RFC 4627).
{content-type: application/json; charset=utf-8}

To analyze some text, we do an HTTP POST to our API with form encoded data containg the text we want to analyze. We get back a JSON object response with 2 attributes label and probability which we parse with a JSON object (more of that in the next number):

```
with TJson.create() do begin
 clear;
 parse(response);
 cnode:= JsonObject.items[0].name;  //'probability'
 writeln(itoa(JsonObject.count));
 writeln('prob: '+values[cnode].asObject.values['neutral'].asstring);
 writeln('result: '+(values['label'].asstring));
 free;
end;
```

```
>>> 2
>>> prob: 0.854074272795421
>>> result: neutral
```

As you may see many of the most commonly used words or phrases are insignificant when it comes to discerning the meaning of a phrase. For example, in the phrase the movie was terrible, the most significant words are movie and terrible, while the and was are almost useless. You could get the same meaning if you took them out, that is, movie terrible or terrible movie. Either way, the sentiment is the same. Another approach is to measure the sentiment of face feelings with 3 flavours: Joy, Sadness and Anger or Disbelief, but that's kind of research.



Our quote from above results in:

```
Sentiment of: "Yesterday I
was clever, so I wanted to
change the world. Today I am
wise, so I am changing myself".
{"probability":
{"neg": 0.37517484595971884,
"neutral": 0.09103427454137769l,
"pos": 0.6248251540402811 6},
"label": "pos"} statuscode: 200
```

The NLTK (Natural Language Toolkit) is a leading platform for building Python or API programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet and many others.

CONCLUSION:
This is a demonstration of sentiment analysis using a NLTK 2.0.4 powered text classification process with msxml2.xmlhttp and TJson objects. It can tell you whether it thinks the text you enter below expresses positive sentiment, negative sentiment, or if it is neutral.

The feedback results will be more accurate on text that is similar to original training data. If you get an odd result, it could be the words you have used are unrecognized. Try entering more words or blocks to improve accuracy. Note that the public API is for non-commercial purposes, and each method is throttled to 1000 calls per day per IP.
You may also test the same context with different languages, the default language is english, but this API also supports dutch and french, but don't forget to change the language in the API call:

```
XMLhttp.Send('text='+'"'+textin+'"'+C
RLF+'language=dutch');
```

Trump's false election fraud claims face a dead end

```
{"probability":
{"neg": 0.52,
 "neutral": 0.64,
 "pos": 0.47},
 „label": "neutral"}
```

Trumps falsche Wahlbetrugsansprüche stehen vor einer Sackgasse

Trump's valse verkiezingsfraude claims lopen dood

Les fausses allégations de fraude électorale de Trump font face à une impasse

```
{"probability":
{"neg": 0.33,
 "neutral": 0.46,
 "pos": 0.66},
 "label": "pos"}
```

Trump's valse verkiezingsfraudeclaims lopen dood

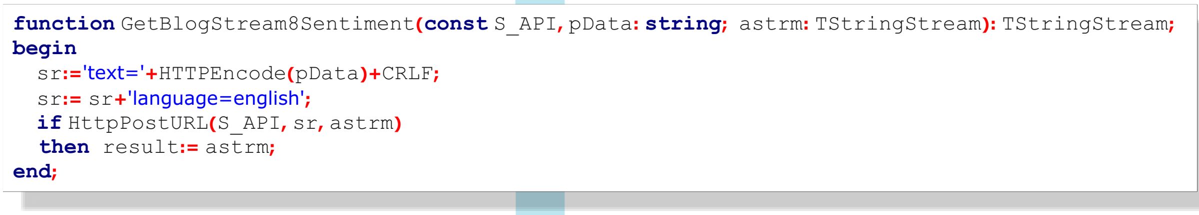
```
{"probability":
{"neg": 0.48,
 "neutral": 0.72,
 "pos": 0.51},
 „label": "neutral"}
```

Ref:
http://www.softwareschule.ch/examples/newssentiment2.txt
http://text-processing.com/docs/sentiment.html
https://www.nltk.org/

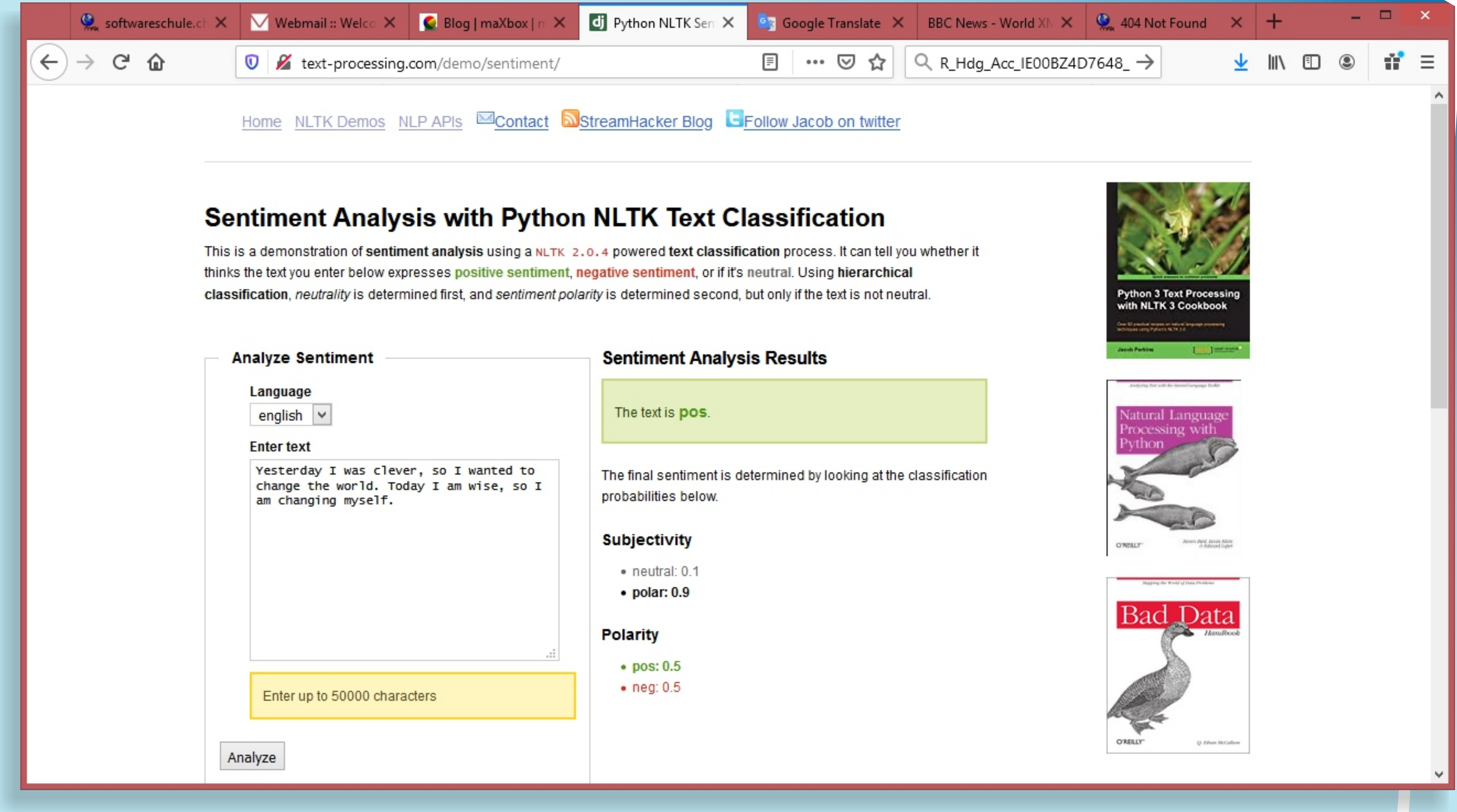
Script:
1017_RSSDatacampSSLStreamSentiment2.pas

Doc:
https://maxbox4.wordpress.com

Appendix: **Alternate HTTPPost-Routine:**



```
function GetBlogStream8Sentiment(const S_API, pData: string; astrm: TStringStream): TStringStream;
begin
  sr:='text='+HTTPEncode(pData)+CRLF;
  sr:= sr+'language=english';
  if HttpPostURL(S_API, sr, astrm)
   then  result:= astrm;
end;
```



Sentiment Analysis with Python NLTK Text Classification

This is a demonstration of **sentiment analysis** using a `NLTK 2.0.4` powered **text classification** process. It can tell you whether it thinks the text you enter below expresses **positive sentiment**, **negative sentiment**, or if it's neutral. Using **hierarchical classification**, *neutrality* is determined first, and *sentiment polarity* is determined second, but only if the text is not neutral.

Analyze Sentiment

Language

english

Enter text

```
Yesterday I was clever, so I wanted to
change the world. Today I am wise, so I
am changing myself.
```

Enter up to 50000 characters

Analyze

Sentiment Analysis Results

The text is **pos**.

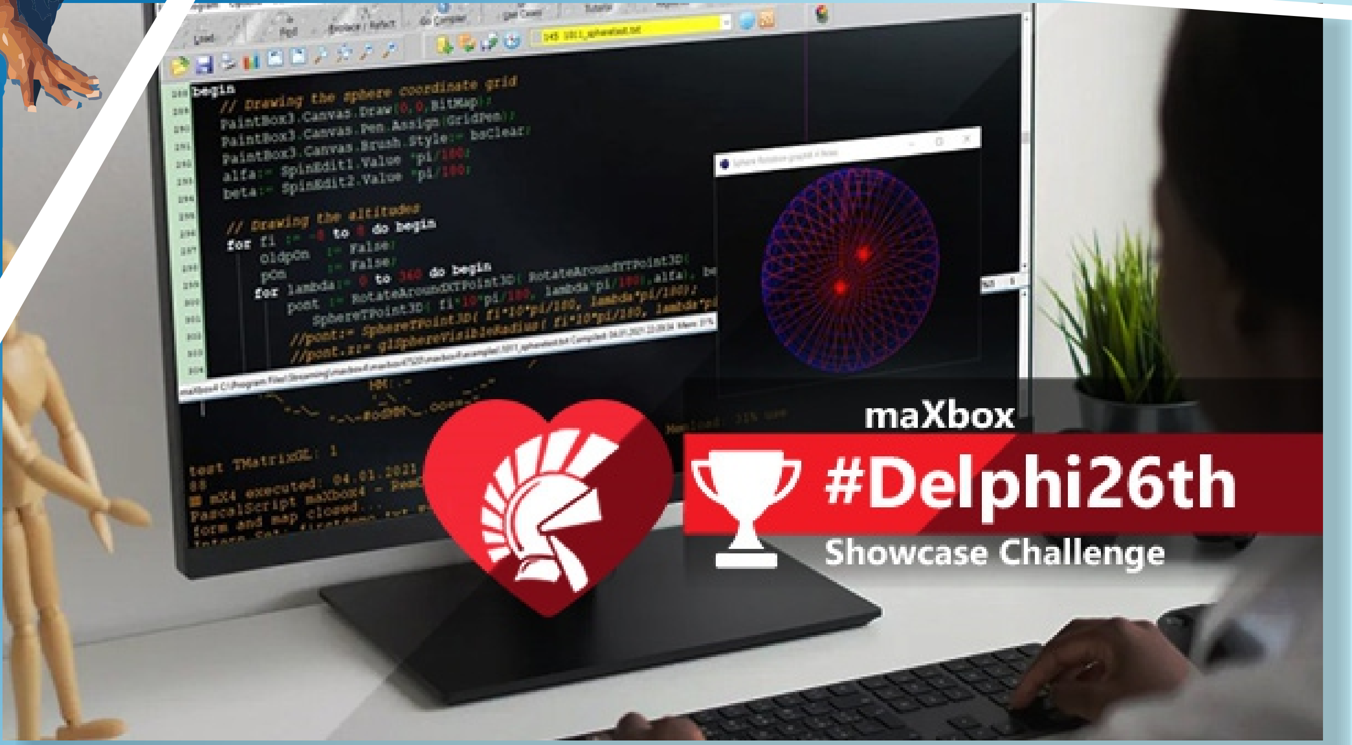
The final sentiment is determined by looking at the classification probabilities below.

Subjectivity

- neutral: 0.1
- **polar: 0.9**

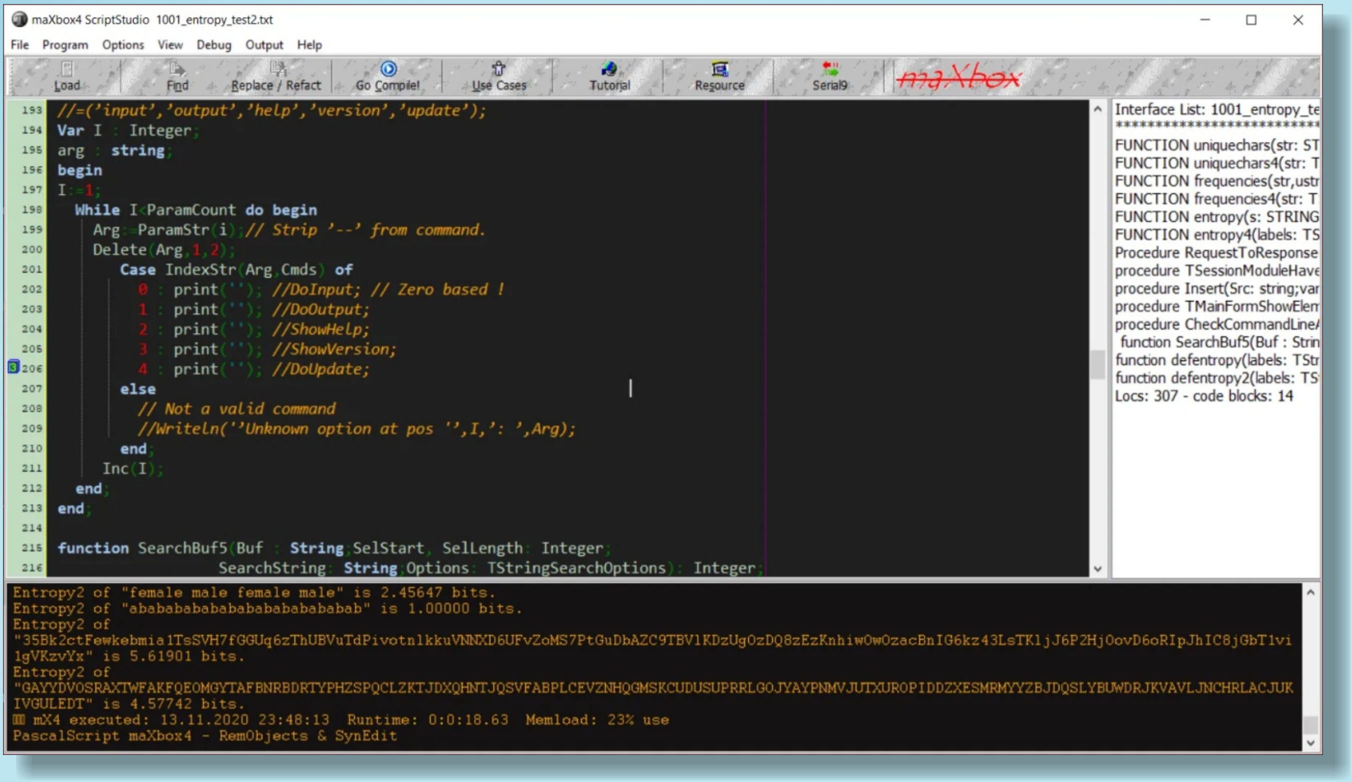
Polarity

- pos: 0.5
- neg: 0.5



Fun maXbox Is An All-In-One Script Engine Application: Powered By Delphi. maXbox is an open source script studio with an Object Pascal scripting language. According to the developer, “it is an open-source script tool engine, compiler, and source library, all in one. Use it to design and code your scripts! Pure Code for Object Pascal Scripting. The guiding principles are simplicity and the reduction of complexity to the max. The app is self-containment “out of the box” with no need for installation or registration.”

https://blogs.embarcadero.com/fun-maxbox-is-an-all-in-one-script-engine-application-powered-by-delphi/



```
//=('input','output','help','version','update');
Var I : Integer;
arg : string;
begin
I:=1;
  While I<ParamCount do begin
    Arg:=ParamStr(i);// Strip '--' from command.
    Delete(Arg,1,2);
       Case IndexStr(Arg,Cmds) of
         0 : print(''); //DoInput; // Zero based !
         1 : print(''); //DoOutput;
         2 : print(''); //ShowHelp;
         3 : print(''); //ShowVersion;
         4 : print(''); //DoUpdate;
       else
         // Not a valid command
         //Writeln(''Unknown option at pos '',I,': ',Arg);
       end;
     Inc(I);
  end;
end;

function SearchBuf5(Buf : String;SelStart, SelLength: Integer;
                  SearchString: String;Options: TStringSearchOptions): Integer;
```

```
Interface List: 1001_entropy_te
*****************************************
FUNCTION uniquechars(str: ST
FUNCTION uniquechars4(str: T
FUNCTION frequencies(str,ustr
FUNCTION frequencies4(str: T
FUNCTION entropy(s: STRING
FUNCTION entropy4(labels: TS
Procedure RequestToResponse
procedure TSessionModuleHave
procedure Insert(Src: string;var
procedure TMainFormShowElem
procedure CheckCommandLine/
 function SearchBuf5(Buf : Strin
function defentropy(labels: TStr
function defentropy2(labels: TS
Locs: 307 - code blocks: 14
```

```
Entropy2 of "female male female male" is 2.45647 bits.
Entropy2 of "ababababababababababababab" is 1.00000 bits.
Entropy2 of
"35Bk2ctFewkebmia1TsSVH7fGGUq6zThUBVuTdPivotnlkkuVNNXD6UFvZoMS7PtGuDbAZC9TBV1KDzUg0zDQ8zEzKnhiw0w0zacBnIG6kz43LsTKljJ6P2Hj0ovD6oRIpJhIC8jGbT1vi
1gVKzvYx" is 5.61901 bits.
Entropy2 of
"GAYYDVOSRAXTWFAKFQEOMGYTAFBNRBDRTYPHZSPQCLZKTJDXQHNTJQSVFABPLCEVZNHQGMSKCUDUSUPRRLGOJYAYPNMVJUTXUROPIDDZXESMRMYYZBJDQSLYBUWDRJKVAVLJNCHRLACJUK
IVGULEDT" is 4.57742 bits.
▥ mX4 executed: 13.11.2020 23:48:13  Runtime: 0:0:18.63  Memload: 23% use
PascalScript maXbox4 - RemObjects & SynEdit
```

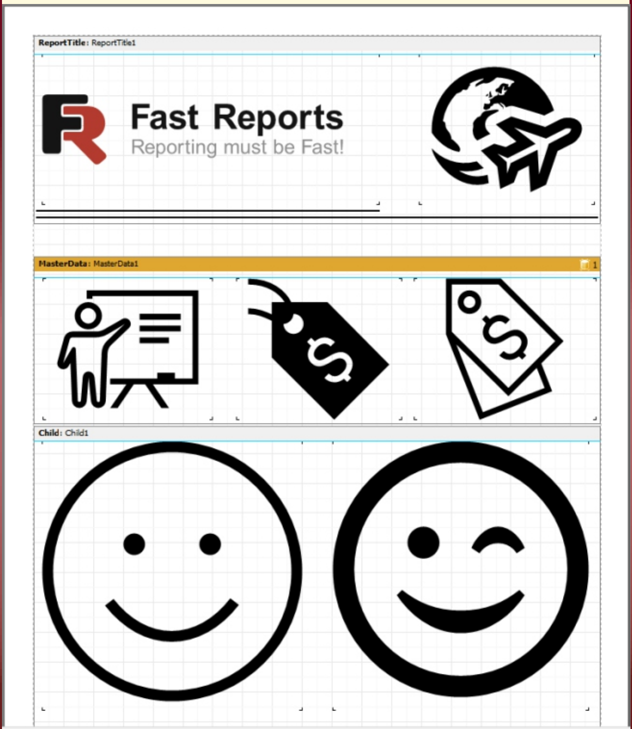
How differs the new FastReport VCL 2021.0 from the previous versions?

Significantly improved work with images - as in image editors:
- High-quality vector SVG images in reports
- Improved image transparency in different formats

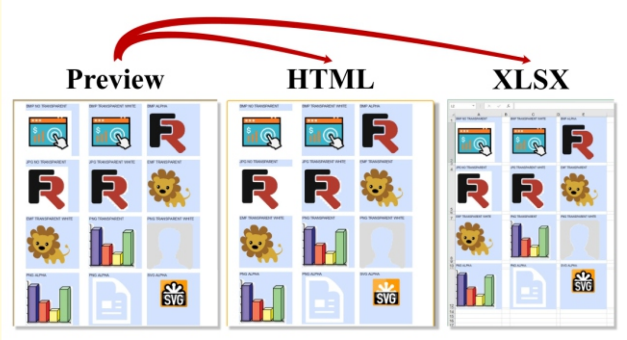
New objects widen the concept of a "report":
Two-Track Pharmacode for designing and printing medication and vaccine packages
Report safety and security:
Now reports in PDF are protected with a digital signature. It guarantees its uniqueness, allows to clearly establish the authorship, and protects it from editing. Your reports now correspond with the docflow standards.
Resource optimization:
- Page miniatures are formed faster
- Less memory required for work

New licensing model:
Starting March 2021 all FastReport VCL editions are subscription-based. It means that you will always have an up-to-date version as long as your subscription is valid.

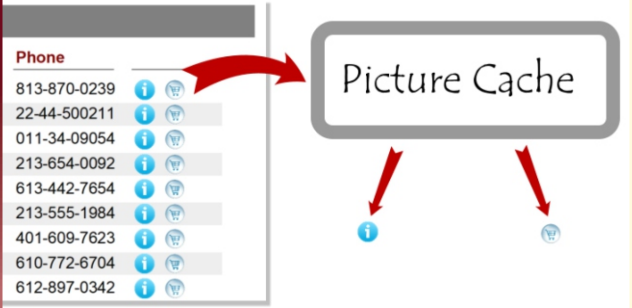
In detail:
Loading and output images in vector SVG format through standard “Picture” object (only for Delphi). Enhance the look of your reports!



Improved transparency support for images inside a report. Now FastReport VCL supports not only color mask but also an alpha channel in the report preview, on the printout, and exports which supports transparent images.

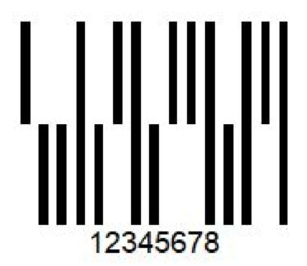


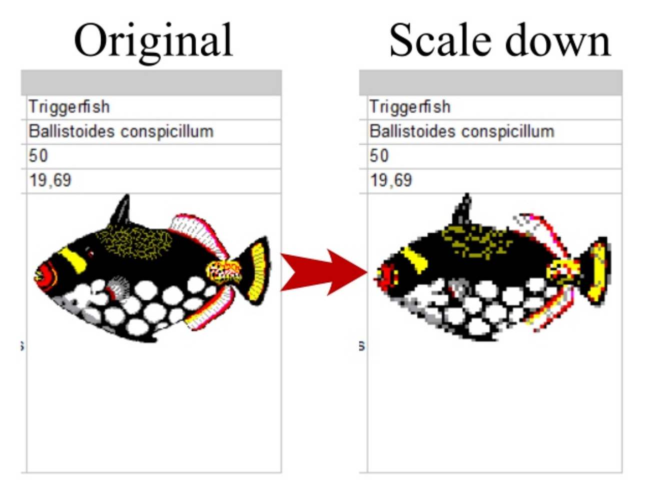
Added experimental picture cache with the ability to generate thumbnails and control overall image quality. New picture cache saves memory usage and GDI descriptors. It loads only one instance of duplicated image (can be turned on with Report.PictureCacheOptions.CachedImagesBuildType=tbtOriginal property).



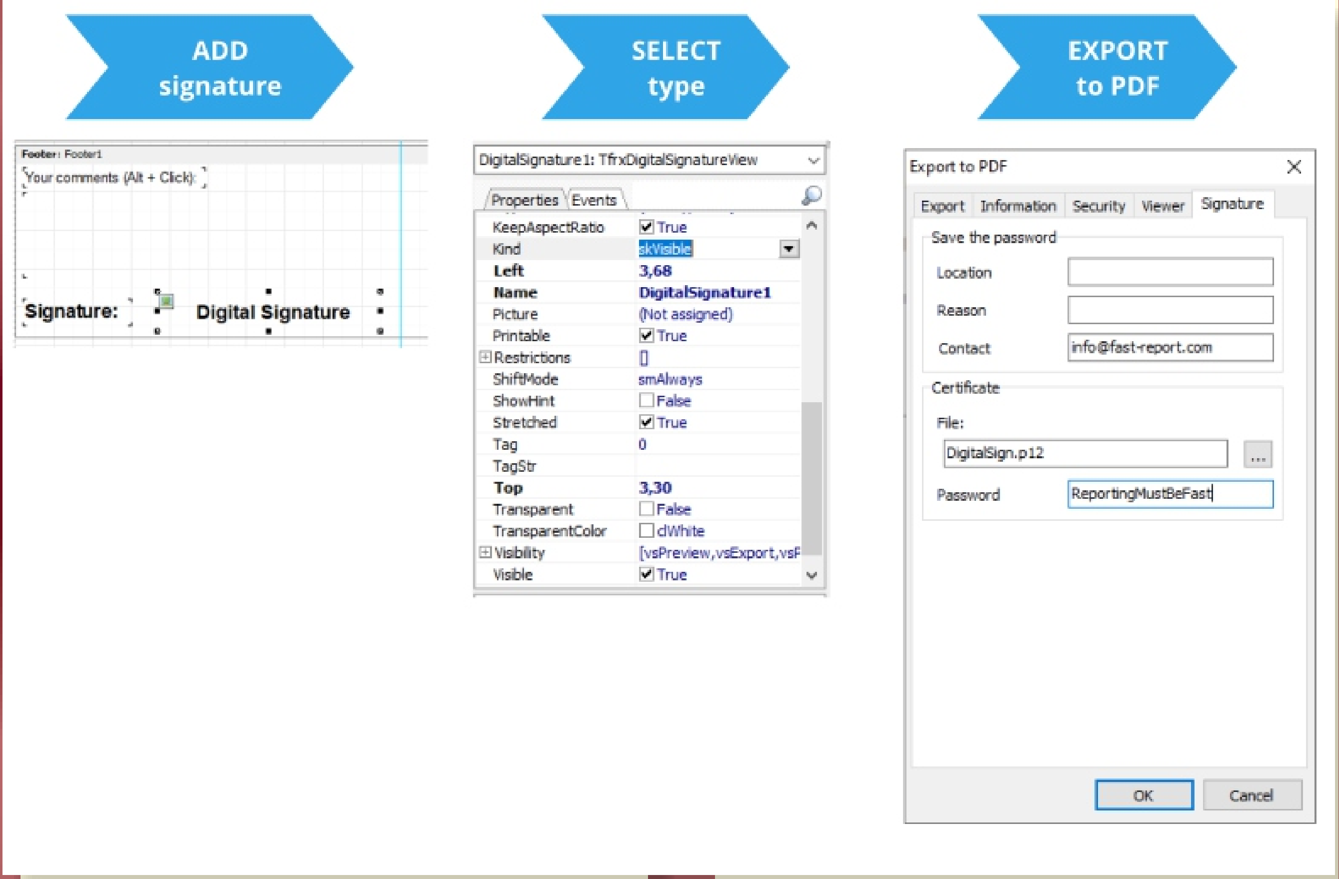
Phone
813-870-0239
22-44-500211
011-34-09054
213-654-0092
613-442-7654
213-555-1984
401-609-7623
610-772-6704
612-897-0342

Added new barcode type Two-Track Pharmacode.





The picture cache can be set up for thumbnail generation which's using for a fast load of images in the preview window (*can be turned on with* `Report.PictureCacheOptions.CachedImagesBuildType=tbtAtPrepare`). The thumbnail quality controls by `Report.PictureCacheOptions.ThumbnailQualityReducer` properties and allows setting the percentage of compression and conditions. In addition, it is possible to control the overall quality of compression for all pictures through Report.PictureCacheOptions.OriginalQualityReducer property. Those images using for preview, printout, and export of a report.



DigitalSignature1: TfrxDigitalSignatureView

KeepAspectRatio	☑ True
Kind	skVisible
Left	3,68
Name	DigitalSignature1
Picture	(Not assigned)
Printable	☑ True
⊞Restrictions	[]
ShiftMode	smAlways
ShowHint	□ False
Stretched	☑ True
Tag	0
TagStr	
Top	3,30
Transparent	□ False
TransparentColor	□ clWhite
⊞Visibility	[vsPreview,vsExport,vsP
Visible	☑ True

Export to PDF

Save the password

Location:

Reason:

Contact: info@fast-report.com

Certificate

File: DigitalSign.p12

Password: ReportingMustBeFast

Added support of Digital signature in PDF export with pfx and p12 certificates support. Sign up your PDF documents just in 3 simple steps: Add the “Digital signature” object (TfrxDigitalSignatureView), select type of signature (hidden, visible, image), and sign up documents with your certificate.

WORKING WITH THE REPORT DESIGNER

INTRODUCTION

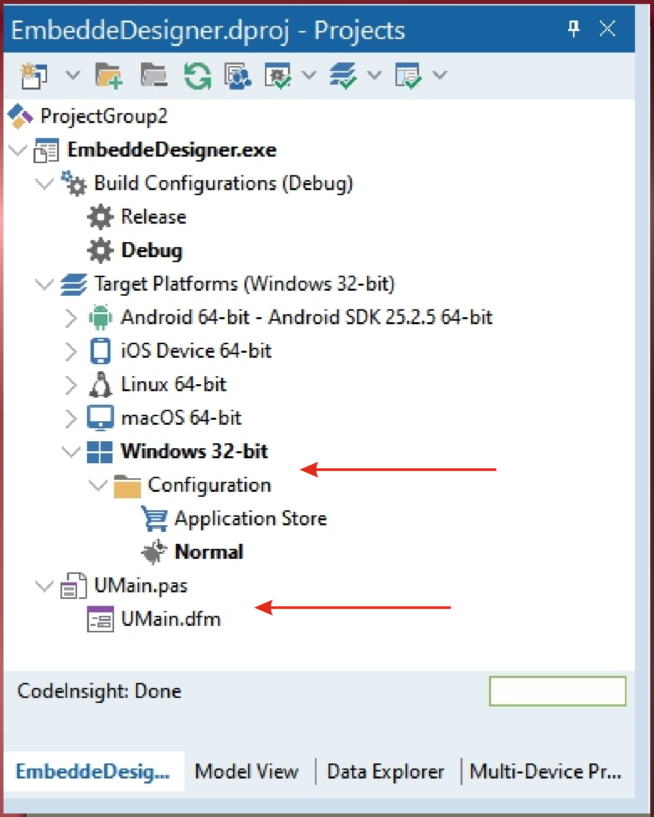
FastReport is a reporting tool, that has one big advantage: It is so easy to understand that even with very little knowledge you can find out how to create a report. To be precise: Put some components on the form (or possibly a DataModule) and simply start.
To make it even more easy FastReport has prepared a complete running project that you could start with. Its called the Embedded designer: it is located over here:
`c:\Program Files (x86)\FastReport VCL Enterprise\Demos\EmbedDesigner.`

START

But to make it even more practical – everyone wants to use a database- I updated the project to collaborate with a database: One of the example databases that come with Delphi: Employee.

So first let's find out where that is hiding:
`c:\ProgramData\Embarcadero\InterBase\gds_db\examples\database\employee.gdb`
So now we found out where the necessary files are residing we can start Delphi. I have done this with Delphi 10.4.2 but an older version will not be any problem. (In the next issue I will do the same Project In Lazarus). So there is no reason for you not to try and I must say the price of the program is very affordable.
If you want to know details please go to:
`https://www.fast-report.com/en/buy/#!/VCL%20(Embarcadero%20RAD%20Studio|Delphi|C++%20Builder)/FastReport%20VCL/`

But for now we need to start Delphi.
Open the project EmbeddedDesigner. This project will be made available for you with your sources on the website downloads address:
`https://www.blaisepascalmagazine.eu/your-downloads/`



EmbeddeDesigner.dproj - Projects

ProjectGroup2
EmbeddeDesigner.exe
Build Configurations (Debug)
Release
Debug
Target Platforms (Windows 32-bit)
Android 64-bit - Android SDK 25.2.5 64-bit
iOS Device 64-bit
Linux 64-bit
macOS 64-bit
Windows 32-bit
Configuration
Application Store
Normal
UMain.pas
UMain.dfm

CodeInsight: Done

EmbeddeDesig... | Model View | Data Explorer | Multi-Device Pr...

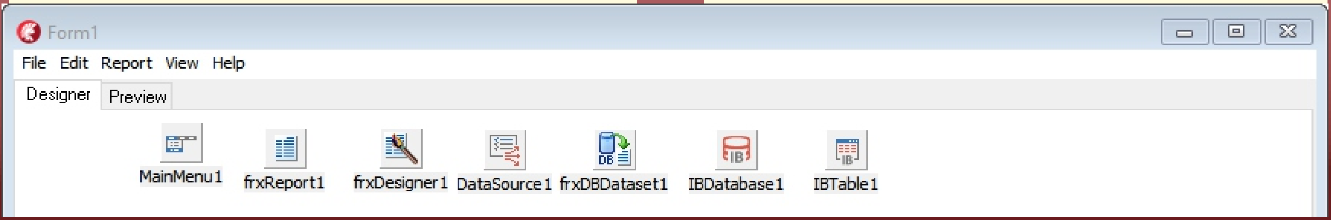
Figure 2: The project

Let's start with the more problematic part of the project: getting the database connected.
We have to do that in Delphi itself – you should be sure InterBase is running on your computer. Let's find out…

1. DELPHI CONNECTING TO DATABASE

(*See the next page figure 3 of the Data Explorer Tab*) Go to the Data Explorer Tab at the right side of your Delphi IDE. Choose "Firedac." Freedom of choice! Not because you need to, but this is the one I used for this simple trial. You might also choose for dbExpress where there is also a database available. All this explanation is done to make it understandable for you if you have to engage your own database. (*See figure 4 of dbExpress on the next page).* There are small x-crosses and that means it is not connected.

Figure 1: The project form with components needed



You can click on the InterBase Item and the EMPLOYEE table will show up: (*See figure 3*)

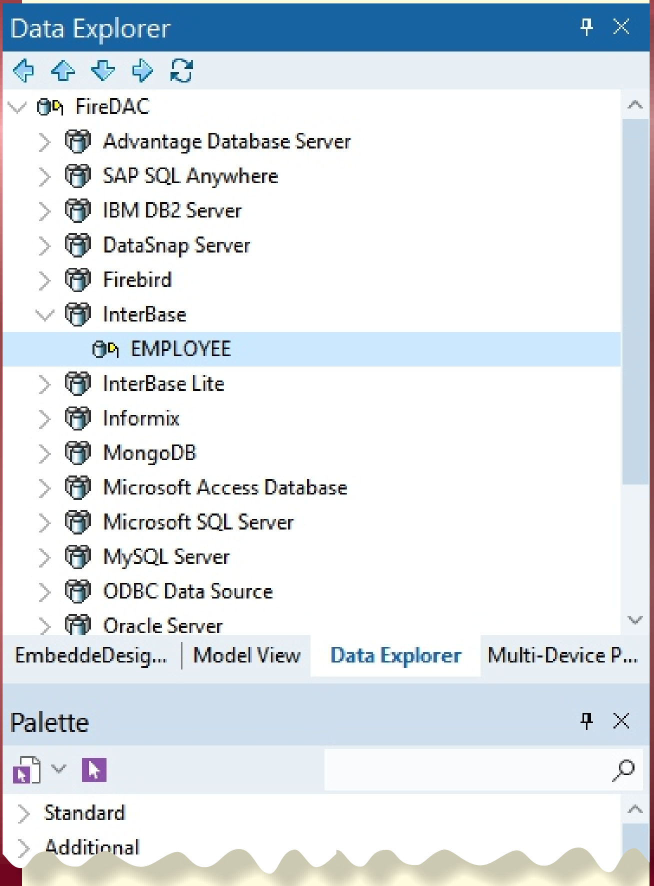


Figure 3: Interbase with EMPLOYEE

You need to fill out the form that pops up if you want make changes after having chosen Modify; right clicking allows you to either Close /Refresh /Modify /Rename or Delete.

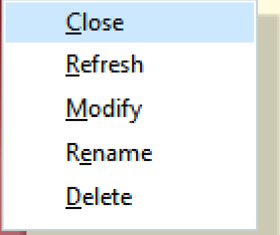


Figure 4: Refreshing?

If you have given the right path and settled for you username "**Sysdba**" and Password "**masterkey**" the connection will become available.
No more red x-cross there, if still, refresh.
See figure 5

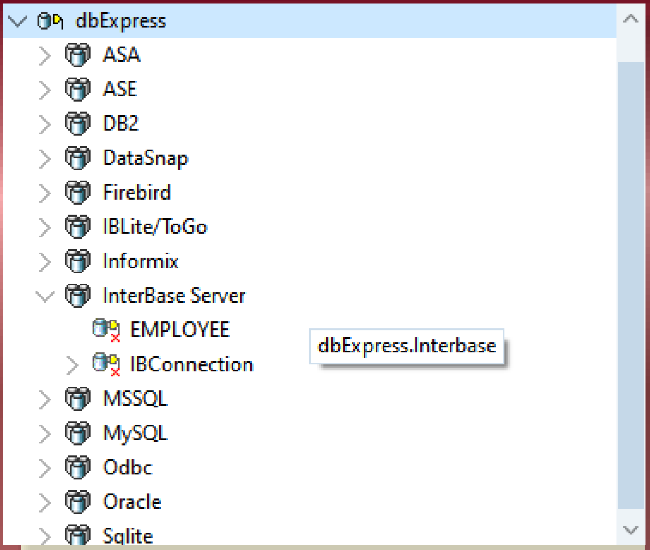


Figure 5: The dbExpress list of databases

2. EXTENDING THE PROJECT

Now we have the database, we need to extend the project a little:
Normally this project starts without the database connection so we have only three components on the form:

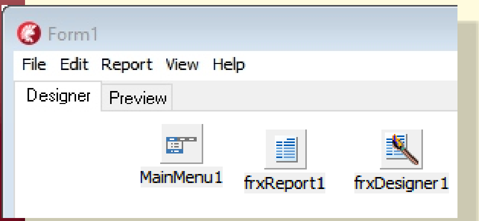


Figure 6: The Form

❶ the main menu
❷ the frxReport
❸ the frxDesigner
these are to be found on the Palette.

Now as an extra we need:

❹ a DataSource from the Data Acces Tab

DataSource1

❺ an frxDBDataset from the Fastreport VCL Tab

frxDBDaset1

❻ an IBDatbase component from the IB Tab

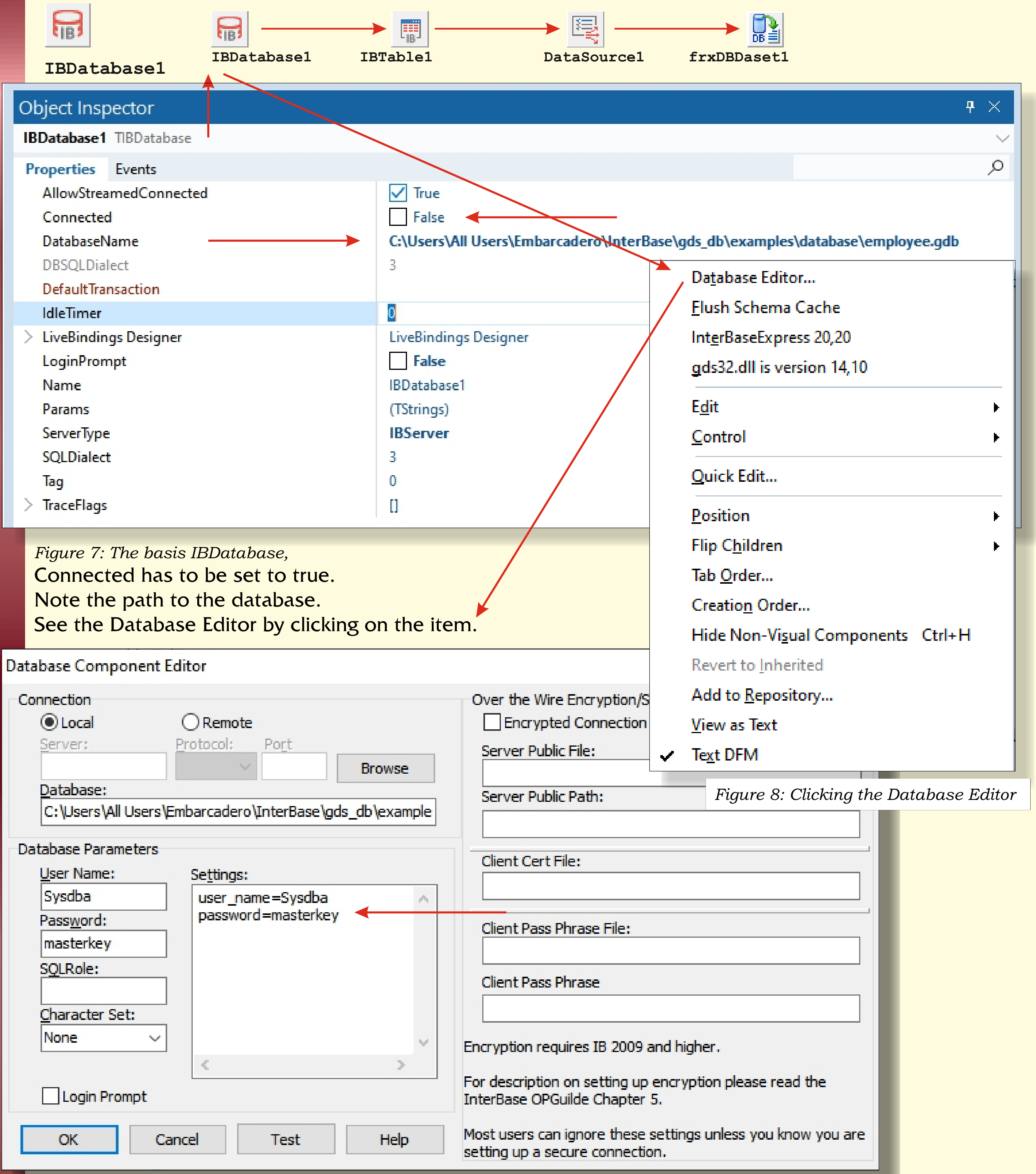
IBDatabase1

❼ and finally an IBTable from the IB Tab

IBTable1

3. CREATING THE RIGHT SETTINGS

Let’s do it per component



IBDatabase1 TIBDatabase

AllowStreamedConnected	☑ True
Connected	□ False
DatabaseName	C:\Users\All Users\Embarcadero\InterBase\gds_db\examples\database\employee.gdb
DBSQLDialect	3
DefaultTransaction	
IdleTimer	0
LiveBindings Designer	LiveBindings Designer
LoginPrompt	□ False
Name	IBDatabase1
Params	(TStrings)
ServerType	IBServer
SQLDialect	3
Tag	0
TraceFlags	[]

Figure 7: The basis IBDatabase,

Connected has to be set to true.
Note the path to the database.
See the Database Editor by clicking on the item.

Figure 8: Clicking the Database Editor

Connection

● Local ○ Remote

Server:

Protocol:

Port

Database: C:\Users\All Users\Embarcadero\InterBase\gds_db\example

Database Parameters

User Name: Sysdba

Password: masterkey

SQLRole:

Character Set: None

Settings:

```
user_name=Sysdba
password=masterkey
```

□ Login Prompt

Over the Wire Encryption/S

□ Encrypted Connection

Server Public File:

Server Public Path:

Client Cert File:

Client Pass Phrase File:

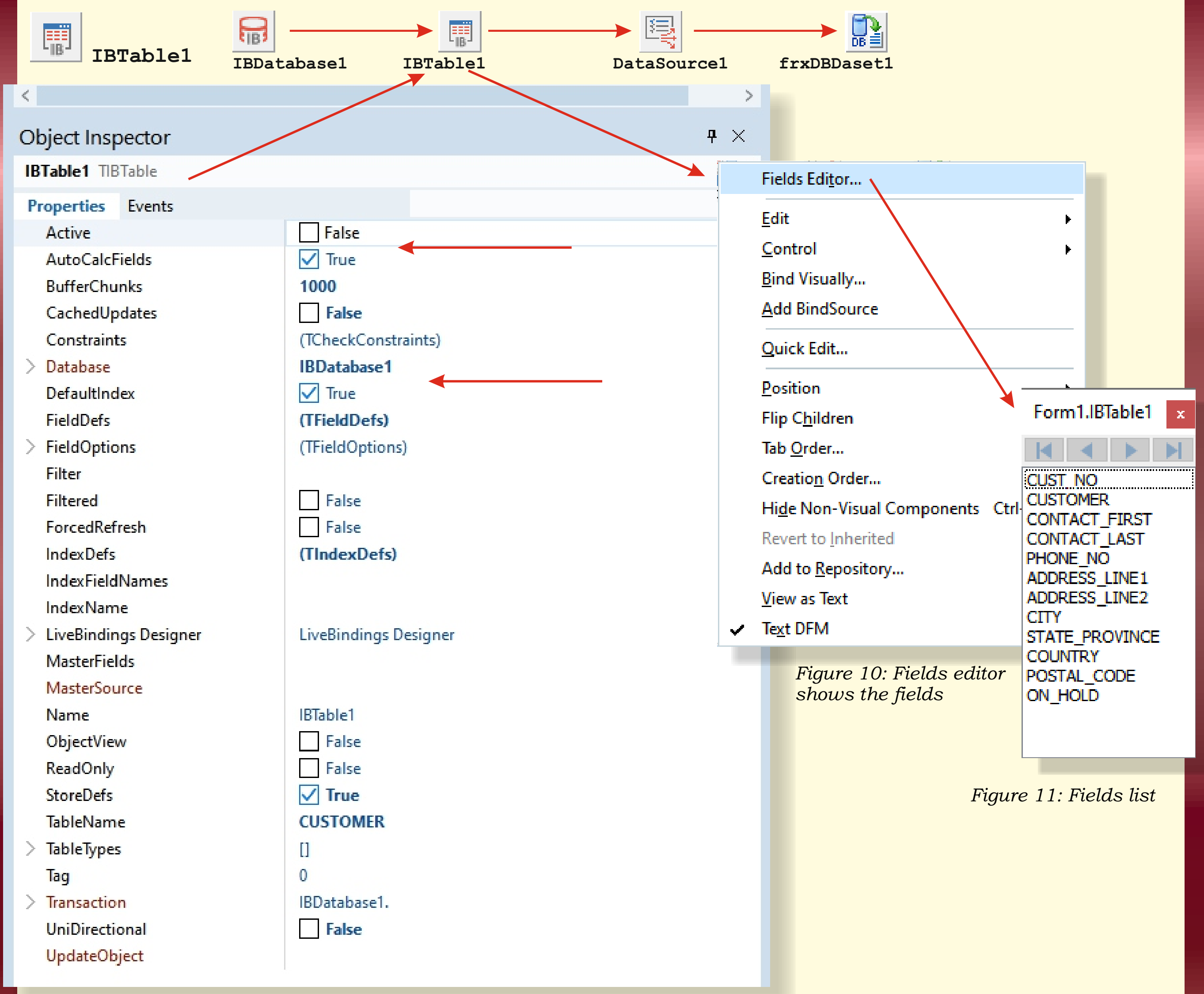
Client Pass Phrase

Encryption requires IB 2009 and higher.

For description on setting up encryption please read the InterBase OPGuilde Chapter 5.

Most users can ignore these settings unless you know you are setting up a secure connection.

Figure 9: Check the settings



IBTable1 TIBTable

Properties | Events

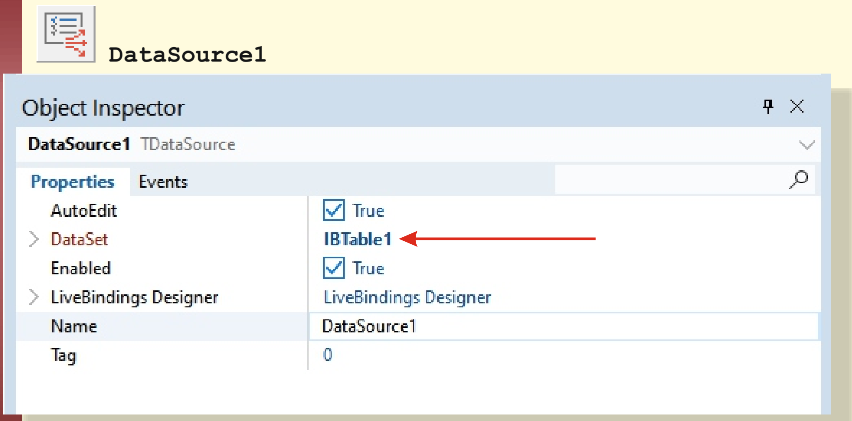
Active	□ False
AutoCalcFields	☑ True
BufferChunks	1000
CachedUpdates	□ False
Constraints	(TCheckConstraints)
Database	IBDatabase1
DefaultIndex	☑ True
FieldDefs	(TFieldDefs)
FieldOptions	(TFieldOptions)
Filter	
Filtered	□ False
ForcedRefresh	□ False
IndexDefs	(TIndexDefs)
IndexFieldNames	
IndexName	
LiveBindings Designer	LiveBindings Designer
MasterFields	
MasterSource	
Name	IBTable1
ObjectView	□ False
ReadOnly	□ False
StoreDefs	☑ True
TableName	CUSTOMER
TableTypes	[]
Tag	0
Transaction	IBDatabase1.
UniDirectional	□ False
UpdateObject	

CUST_NO
CUSTOMER
CONTACT_FIRST
CONTACT_LAST
PHONE_NO
ADDRESS_LINE1
ADDRESS_LINE2
CITY
STATE_PROVINCE
COUNTRY
POSTAL_CODE
ON_HOLD

Figure 10: Fields editor shows the fields

Figure 11: Fields list

Figure 9: the table needs to be set to active

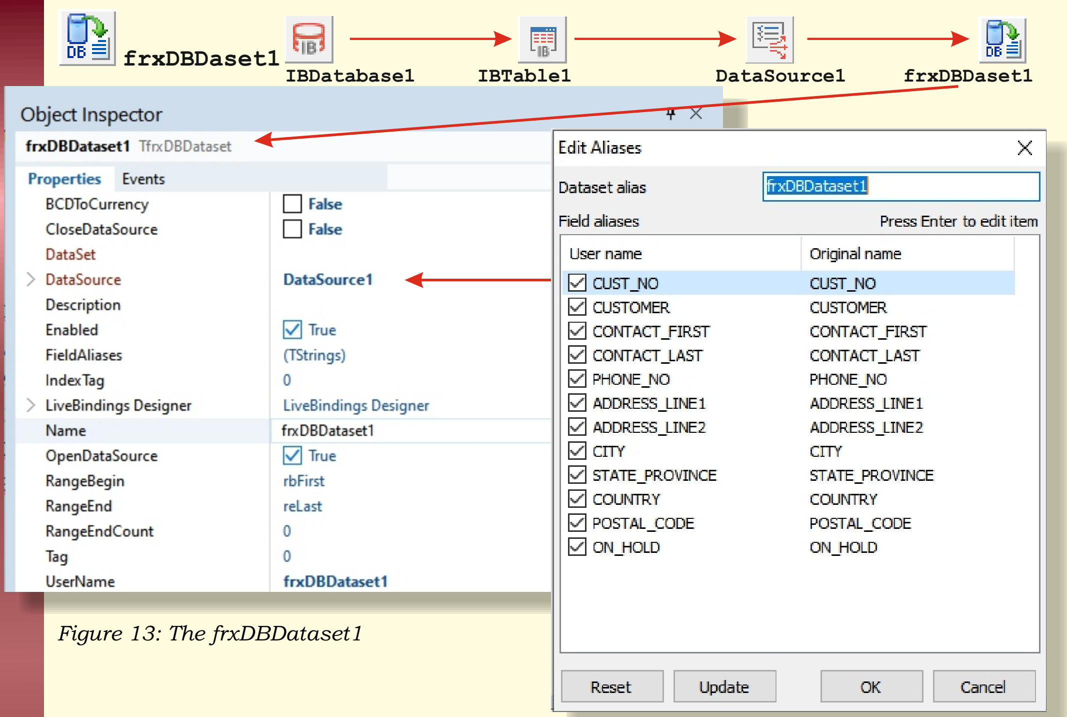


DataSource1 TDataSource

Properties | Events

AutoEdit	☑ True
DataSet	IBTable1
Enabled	☑ True
LiveBindings Designer	LiveBindings Designer
Name	DataSource1
Tag	0

Figure 12: Connection with IBTable



Object Inspector

frxDBDataset1 TfrxDBDataset

Properties | Events

BCDToCurrency	☐ False
CloseDataSource	☐ False
DataSet	
DataSource	DataSource1
Description	
Enabled	☑ True
FieldAliases	(TStrings)
IndexTag	0
LiveBindings Designer	LiveBindings Designer
Name	frxDBDataset1
OpenDataSource	☑ True
RangeBegin	rbFirst
RangeEnd	reLast
RangeEndCount	0
Tag	0
UserName	frxDBDataset1

Edit Aliases

Dataset alias: frxDBDataset1

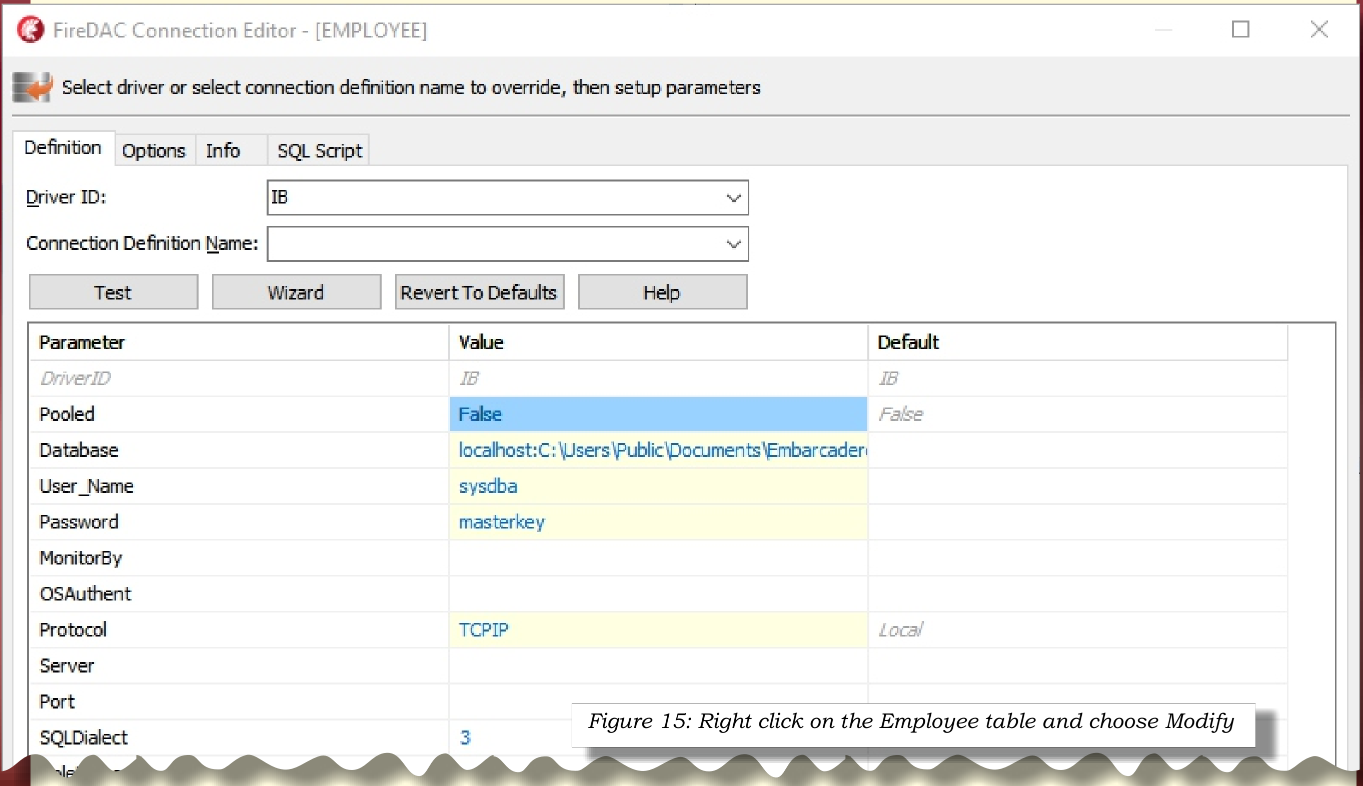
Field aliases — Press Enter to edit item

User name	Original name
☑ CUST_NO	CUST_NO
☑ CUSTOMER	CUSTOMER
☑ CONTACT_FIRST	CONTACT_FIRST
☑ CONTACT_LAST	CONTACT_LAST
☑ PHONE_NO	PHONE_NO
☑ ADDRESS_LINE1	ADDRESS_LINE1
☑ ADDRESS_LINE2	ADDRESS_LINE2
☑ CITY	CITY
☑ STATE_PROVINCE	STATE_PROVINCE
☑ COUNTRY	COUNTRY
☑ POSTAL_CODE	POSTAL_CODE
☑ ON_HOLD	ON_HOLD

Figure 13: The frxDBDataset1

Figure 14: If so desired you can rename the fields

After all these settings there is one more item to be named:
If your connection fails: check the Data Explorer tab and double click on the Employee table item.
If all is settled you can see the table views procedures and generators (*see figure 16 next page*)


FireDAC Connection Editor - [EMPLOYEE]

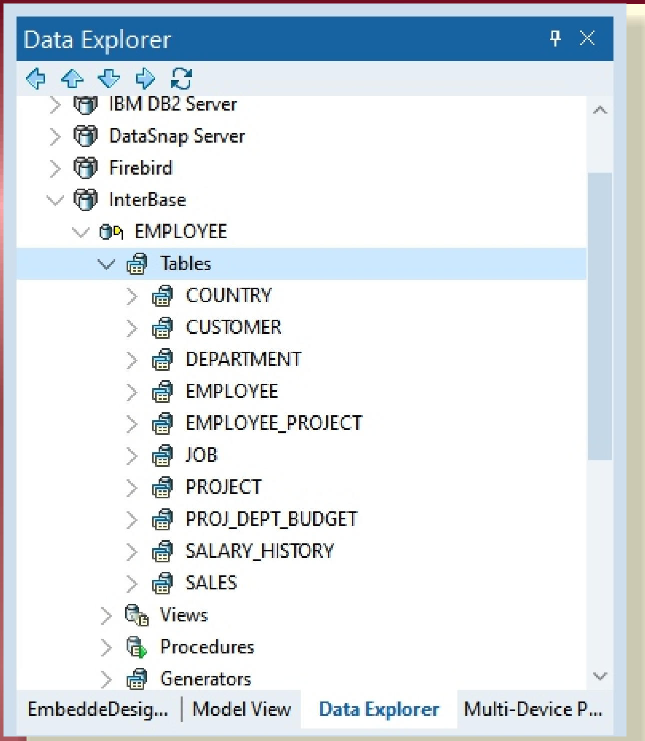
Select driver or select connection definition name to override, then setup parameters

Driver ID: IB

Connection Definition Name:

Parameter	Value	Default
DriverID	IB	IB
Pooled	False	False
Database	localhost:C:\Users\Public\Documents\Embarcader	
User_Name	sysdba	
Password	masterkey	
MonitorBy		
OSAuthent		
Protocol	TCPIP	Local
Server		
Port		
SQLDialect	3	

Figure 15: Right click on the Employee table and choose Modify



Data Explorer

- IBM DB2 Server
- DataSnap Server
- Firebird
- InterBase
  - EMPLOYEE
    - Tables
      - COUNTRY
      - CUSTOMER
      - DEPARTMENT
      - EMPLOYEE
      - EMPLOYEE_PROJECT
      - JOB
      - PROJECT
      - PROJ_DEPT_BUDGET
      - SALARY_HISTORY
      - SALES
    - Views
    - Procedures
    - Generators

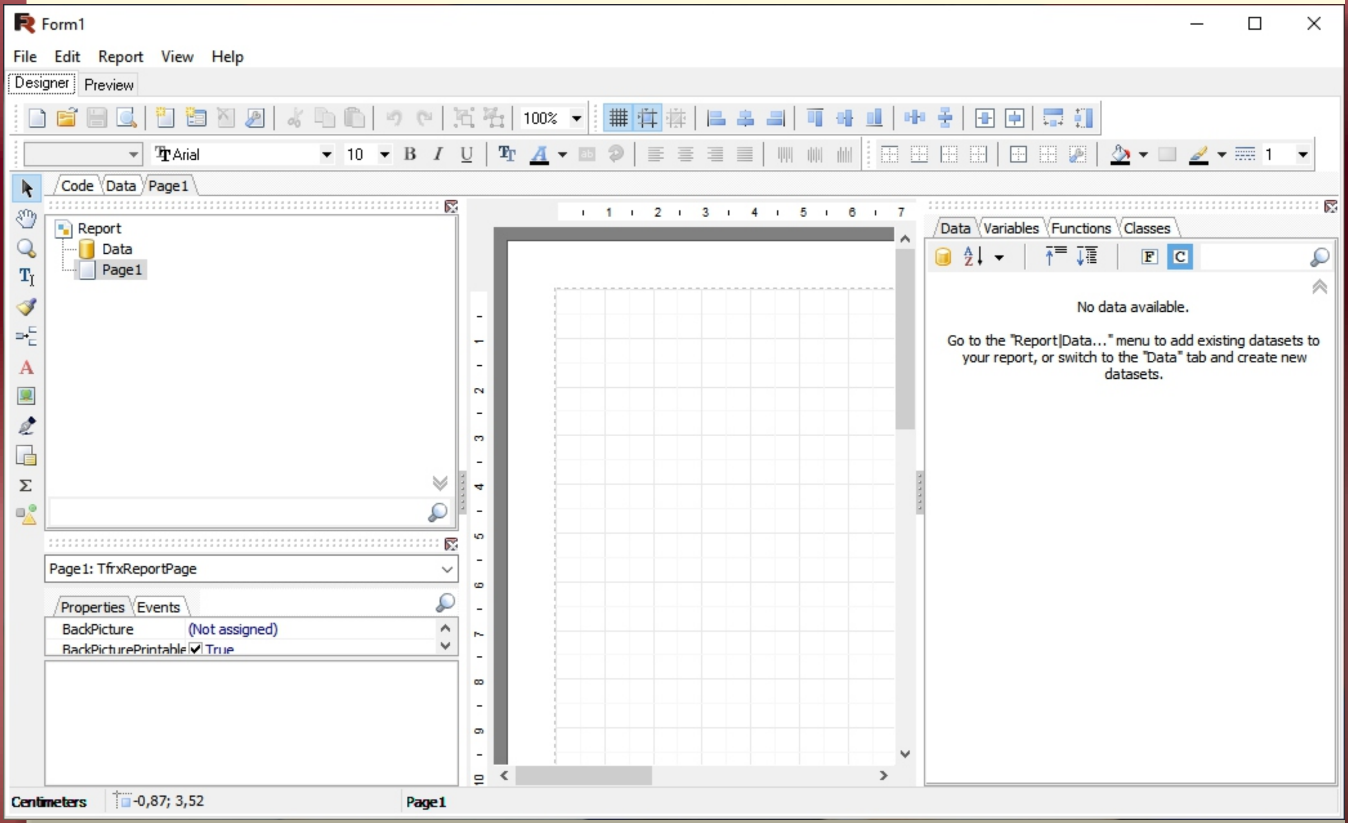
EmbeddeDesig... | Model View | Data Explorer | Multi-Device P...

Figure 16: The content of the table

So now on the side of Delphi everything is set and should be working. Let's prepare the Designer.
Run the program and you will see the Designer for the first time. (See figure 17).
Now you can see a separated program, with this you can create your new forms etc., even coupled to a database of your choice. For those of you that want to create forms like in Adobe Acrobat, here is an other option.

It's saved in the directory of the project, so you can easily find it: (it's in your download area)

```
...\DelphiProjects\
       _history\
       _recovery\
       EmbeddeDesigner.identcache
       EmbeddeDesigner.exe
       EmbeddeDesigner.dproj.local
       EmbeddeDesigner.res
       EmbeddeDesigner.dproj
       UMain.dfm
       UMain.pas
       UMain.dcu
       Detlef_FastreportTest.fr3
       EmbeddeDesigner.dpr
       Project1_Icon.ico
       EmbeddeDesigner_Icon.ico
```



Page1: TfrxReportPage

Properties Events

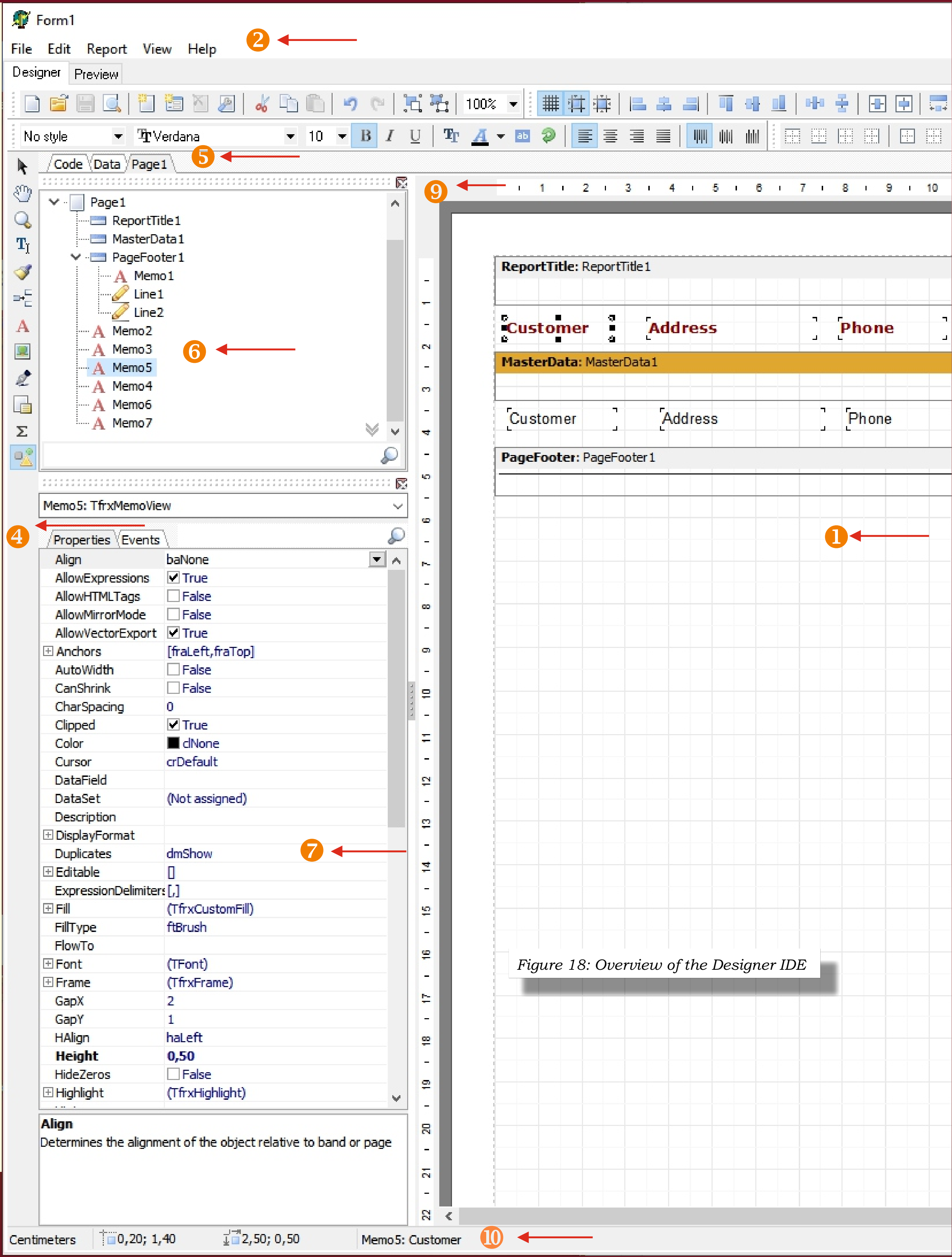
BackPicture	(Not assigned)
BackPicturePrintable	☑ True

No data available.

Go to the "Report|Data..." menu to add existing datasets to your report, or switch to the "Data" tab and create new datasets.

Centimeters -0,87; 3,52 Page1

Figure 17: The Designer at the beginning



- Page1
  - ReportTitle1
  - MasterData1
  - PageFooter1
    - Memo1
    - Line1
    - Line2
  - Memo2
  - Memo3
  - Memo5
  - Memo4
  - Memo6
  - Memo7

ReportTitle: ReportTitle1

Customer Address Phone

MasterData: MasterData1

Customer Address Phone

PageFooter: PageFooter1

Memo5: TfrxMemoView

Align	baNone
AllowExpressions	☑ True
AllowHTMLTags	□ False
AllowMirrorMode	□ False
AllowVectorExport	☑ True
⊞ Anchors	[fraLeft,fraTop]
AutoWidth	□ False
CanShrink	□ False
CharSpacing	0
Clipped	☑ True
Color	■ clNone
Cursor	crDefault
DataField	
DataSet	(Not assigned)
Description	
⊞ DisplayFormat	
Duplicates	dmShow
⊞ Editable	[]
ExpressionDelimiters	[,]
⊞ Fill	(TfrxCustomFill)
FillType	ftBrush
FlowTo	
⊞ Font	(TFont)
⊞ Frame	(TfrxFrame)
GapX	2
GapY	1
HAlign	haLeft
Height	**0,50**
HideZeros	□ False
⊞ Highlight	(TfrxHighlight)

Align
Determines the alignment of the object relative to band or page

Figure 18: Overview of the Designer IDE

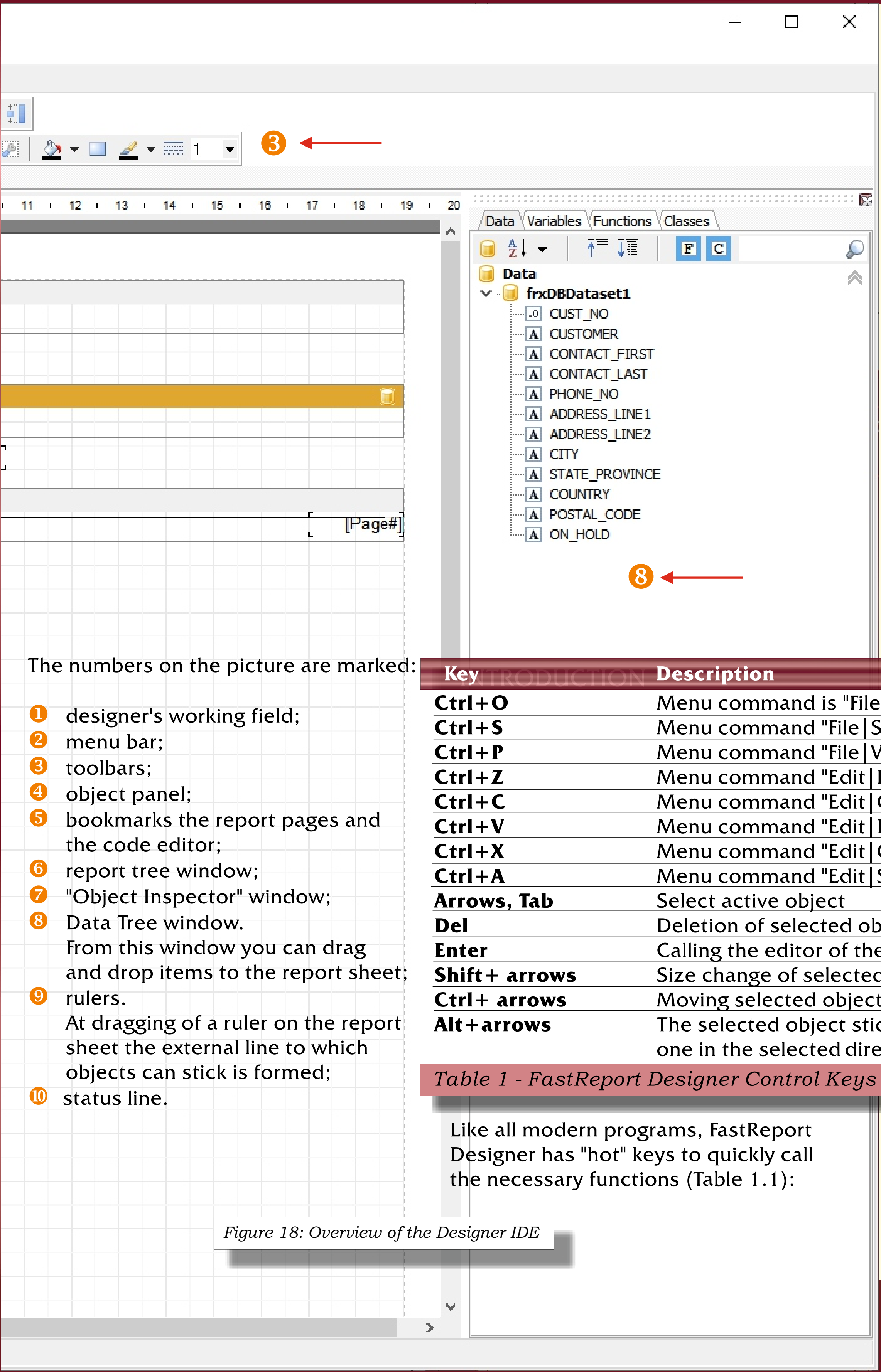


Figure 18: Overview of the Designer IDE

The numbers on the picture are marked:

1. designer's working field;
2. menu bar;
3. toolbars;
4. object panel;
5. bookmarks the report pages and the code editor;
6. report tree window;
7. "Object Inspector" window;
8. Data Tree window.
   From this window you can drag and drop items to the report sheet;
9. rulers.
   At dragging of a ruler on the report sheet the external line to which objects can stick is formed;
10. status line.

Like all modern programs, FastReport Designer has "hot" keys to quickly call the necessary functions (Table 1.1):

Key	Description
Ctrl+O	Menu command is "File|Open..."
Ctrl+S	Menu command "File|Save"
Ctrl+P	Menu command "File|View"
Ctrl+Z	Menu command "Edit|Discard"
Ctrl+C	Menu command "Edit|Copy"
Ctrl+V	Menu command "Edit|Paste"
Ctrl+X	Menu command "Edit|Cut"
Ctrl+A	Menu command "Edit|Select all"
Arrows, Tab	Select active object
Del	Deletion of selected objects
Enter	Calling the editor of the selected object
Shift+ arrows	Size change of selected objects
Ctrl+ arrows	Moving selected objects
Alt+arrows	The selected object sticks to the nearest one in the selected direction.

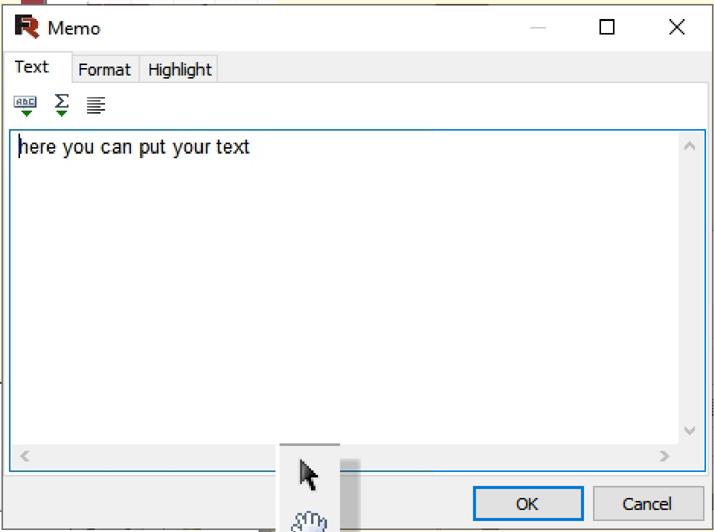
Table 1 - FastReport Designer Control Keys

Once you start a new project:
File → New etc.
You can start designing it.

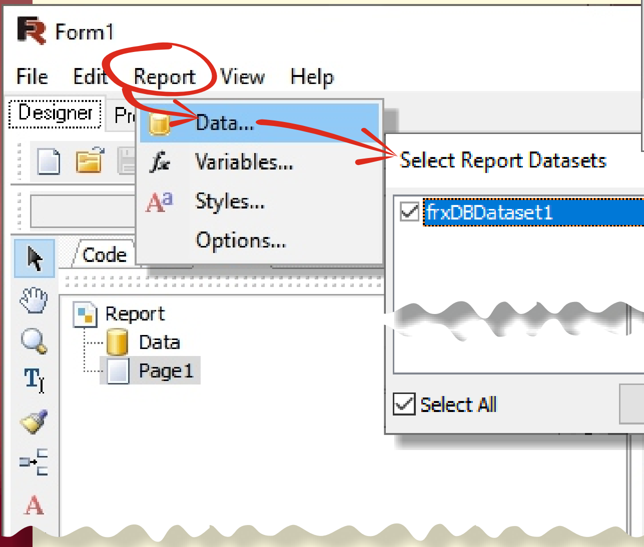
(see the large Figure 18 at the top number 2
For this I have prepared a form where you can see the final coupling to a database.
Let's describe the necessary steps to make contact with the database:

Select →**Report** and a small window pops up and your database is mentioned. So check it and immediately you will see at the right side Data and fields of your database.

Figure 21: The outline of the memo text to be drawn



here you can put your text



☑ frxDBDataset1

☑ Select All

Figure 19: Open Report Data and then select the Dataset of your choice

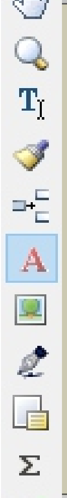
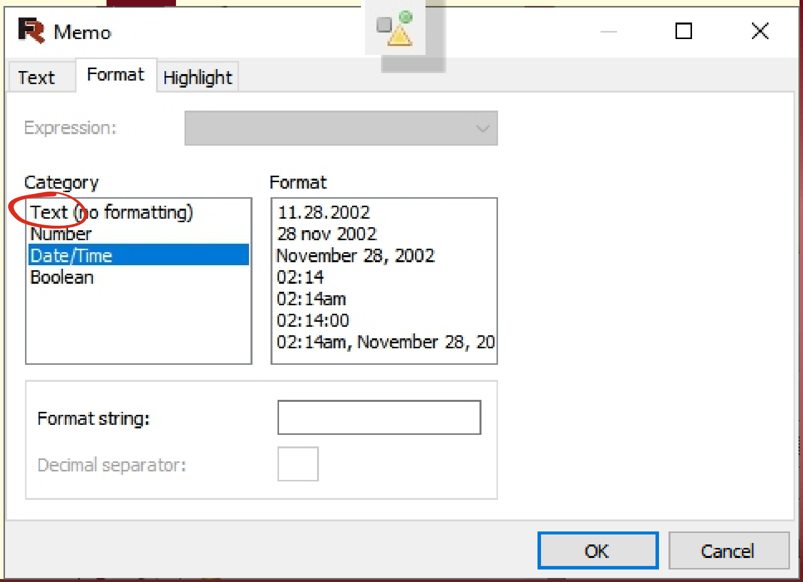


Figure 22: The Memo text selector

You now can under **File→ New Report** create your first form or if you want to, do it with just a blank form.

To add data and show them you can use the TMemo (figure 18)
if you click the icon you'll see a tool which enables you to set the place and size of this memo.
There is something special to this memo:
It holds the ability to simply set text, or couple a field from your database and on top of that create a formula.



Expression:

Category	Format
Text (no formatting)	11.28.2002
Number	28 nov 2002
Date/Time	November 28, 2002
Boolean	02:14
	02:14am
	02:14:00
	02:14am, November 28, 20

Format string:

Decimal separator:

Figure 20: Here you can even alter the properties

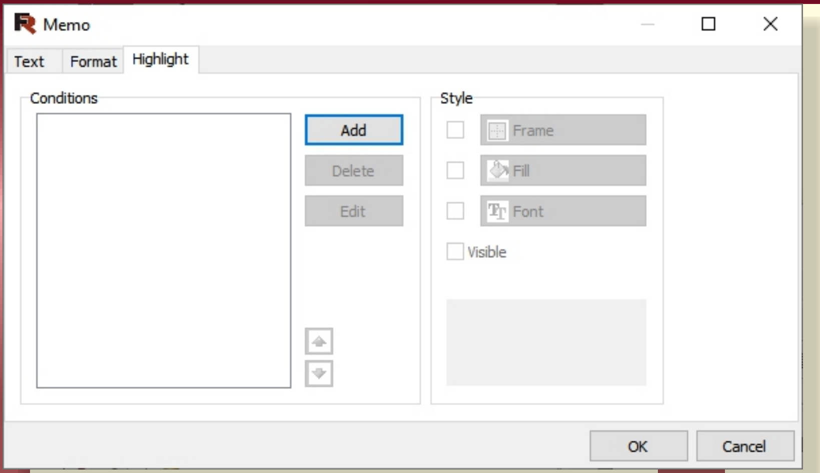
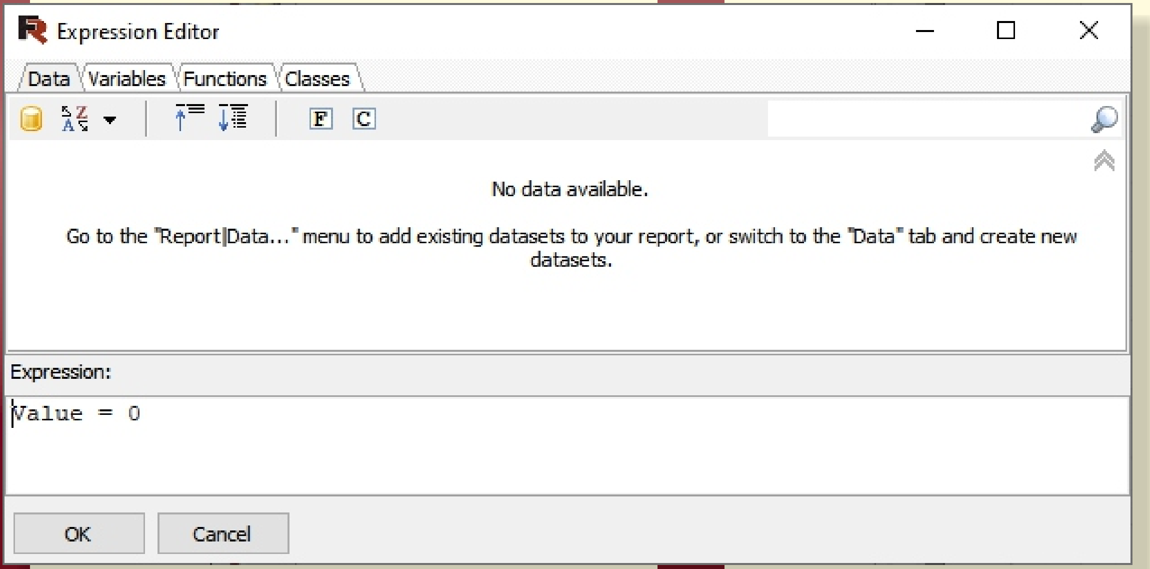


Figure 23: Create Conditions



No data available.

Go to the "Report|Data..." menu to add existing datasets to your report, or switch to the "Data" tab and create new datasets.

```
Value = 0
```

Figure 24: Click add to show this window and create Expressions

These are the first steps to create a form and connect to data.
Have Fun!

In the rest of this article you will find some detailed information about FastReports and the functionality.

The control of the mouse manipulator is described.

Action	Description
Left Mouse Button	Selecting an object; inserting a new object; moving and resizing the object(s). You can scale the selected objects by pulling the red quare at the bottom right corner of the selected group of objects.
Right Mouse Button	Context menu of the object(s) above which the mouse pointer is placed.
Double-click the left Mouse button	Call the object editor. If you double-click on an empty page, the "Page Options" dialog box will be called.
Mouse Wheel	Scrolling through the report page.
Ctrl + Mouse Wheel	Scope
Shift + left Mouse button	If an object is already selected, it removes the selection from it, otherwise the object is selected. The selection of other objects does not change.
Ctrl + left Mouse button	A frame is drawn as soon as you press and move the mouse; after releasing the mouse button, all the objects included in the frame are highlighted. The same effect can be achieved by clicking with the left mouse button on an empty area of the page and dragging the mouse to the desired position without releasing the button.
Alt + left Mouse button	If the object "Text" is selected, it edits its content inside.

Table 1.2 - Mouse control modes

By learning the "hotkeys" and mouse techniques, you can significantly accelerate the work in the designer in the future.

Let's look at the toolbars. In the designer it is possible to allocate five panels:

❶ Designer mode panel (object panel);
❷ The "Standard" toolbar;
❸ The Text toolbar;
❹ Rectangle toolbar;
❺ Alignment toolbar.

The Designer Mode Panel is combined with the Object Panel.

Icon	Title	Description
↖	`Object selection`	Normal operation mode, where the mouse pointer allows selecting objects, changing their size, etc.
✋	`Hand pointer`	If you click with the left mouse button, you can drag the report sheet.
🔍	`Magnifying Glass`	A single left mouse click zooms in 100%, the right mouse button zooms out 100%. If you press the left mouse button and drag the mouse without releasing it, it zooms the selected area.
TI	`Text Editor`	By clicking on the Text object, you can edit its content right on the report sheet. If you press the left button and drag the mouse without releasing it, the "Text" object will be created on the selected place and its editor will start.
🖌	`Format copying`	The button becomes active if the "Text” object is selected. If you press with the left mouse button on the "Text" object, it copies to the object the formatting that has the previously selected object "Text".

Table 1.3 - Designer Mode Panel

The "Standard" toolbar is placed at the top of the program window (Figure 1.2).

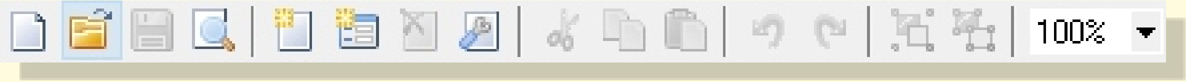


Figure 25 - "Standard" toolbar left under item on the overview at page 40 2

The "Standard" toolbar has the following icons:

Icon	Title	Description
	New report	Creates a new blank report.
	Open report	Opens an existing report from a file. The keyboard analogue is Ctrl+O.
	Save the report	Saves the report as a file. The keyboard analogue is Ctrl+S.
	Preview	Performs report generation and preview. The keyboard analogue is Ctrl+P.
	New Page	Adds a new page to the report.
	New dialog form	Adds a new dialog form to the report.
	Delete Page	Deletes the current page.
	Page Properties	Calls up a dialog with page properties.
	Cut out	Cuts the selected objects into the clipboard. The keyboard analogue is Ctrl+X.
	Copy	Copies the selected objects to the clipboard. The keyboard analogue is Ctrl+C.
	Insert	Inserts objects from the clipboard. The keyboard analogue is Ctrl+V.
	Cancel	Cancels the last operation. The keyboard analogue is Ctrl+Z.
	Repeat	Repeats the last operation that was cancelled. The keyboard analogue is Ctrl+Y.
	Group	Groups the selected objects.
	Ungroup	Ungroups the selected objects.
75% ▾	Zoom level	Defines the report page scale.

Table 1.4 - Contents of the "Standard" toolbar

The Text toolbar is also placed by default at the top of the program window, slightly below the Standard toolbar (Figure 26).

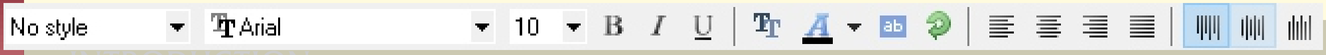


Figure 26 - Toolbar "Text" see item 3

The "Text" toolbar has the following icons:

Icon	Title	Description
No Style	Style	Allows you to choose a style. To define the liststyle. of styles, call the menu item "Report|Styles...".
Arial	Font	Allows you to select a font name from the drop-down list. Remembers the last five fonts used.
10	Font Size	Allows you to select the font size from the drop-down list. The size can also be entered manually.
B	Bold	Sets/removes font thickening.
I	Italic	Sets/removes the cursive of the font.
U	Underlining	Sets/releases font underscore.
Tт	Font Properties	Allows you to specify font properties using the standard font selection dialog box.
A	Font Colour	Selects the font color from the drop-down list.
ab	Conditional allocation	Shows a dialog with selection attributes for the selected Text object.
⟳	Rotate text	Allows you to select text rotation.
≡	Alignment to the left	Sets text alignment to the left.
≡	Center alignment	Sets the text to centerline.
≡	Alignment to the right	Sets the text alignment to the right.
≡	Width Alignment	Sets the alignment of text evenly across the width.
‖‖	Upperedge alignment	Sets the alignment of text at the top edge.
‖‖	Height alignment	Sets the text height alignment.
‖‖	Lower edge alignment	Sets the alignment of text at the bottom edge.

Table 1. 5 - Contents of the "Text" toolbar

By default, the Rectangle toolbar is placed on the right side of the Text toolbar.

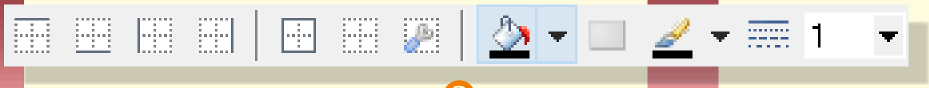
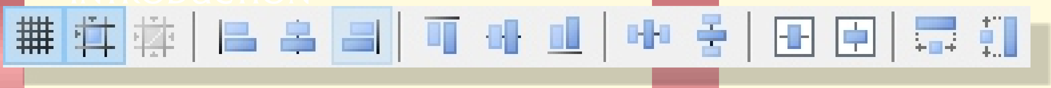


Figure 27 - Rectangle toolbar see item 3 on page 9/20

Icon	Title	Description
	Upper line	Turns on/off the top line of the frame.
	Lower Line	Enables/disables the frame bottom line.
	Left Line	Switches on/off the left frame line.
	Right Line	Switches on/off the right frame line.
	All Lines	Includes all frame lines.
	No Lines	Turns off all frame lines.
	Frame Editor	Calling the frame editor.
	Back Ground Color	Selects the font color from the drop-down list.
	Line Color	Selects the line color from the drop-down list.
	Fill Editor	Calling the fill editor.
	Line style	Selects a line style from a drop-down list.
1	Line thickness	Selects the thickness of the line from the drop-down list.

Table 1. 6 - Contents of the "Rectangle" toolbar

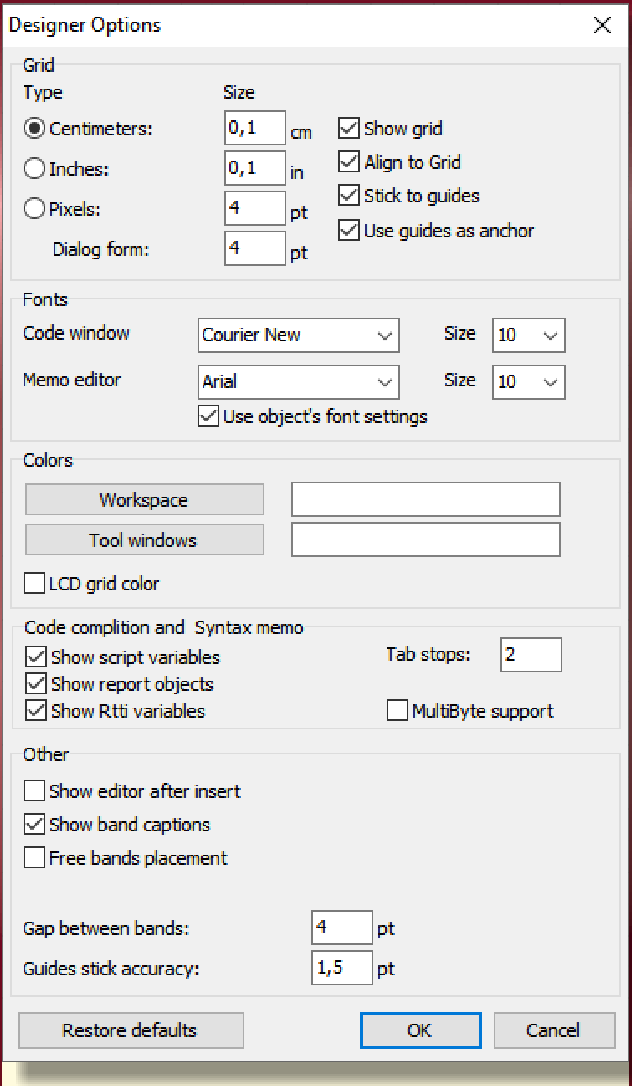
The "Alignment Palette" toolbar is not displayed by default, but can be easily added from the "View"→"Toolbars"→"Alignment Palette" menu (Figure 28 see item 3).



Icon	Description
	Show the grid.
	Align objects with the grid.
	Arrange in the nodes of the grid.
	Align the left edges.
	Horizontal center.
	Align right edges
	Align the upper edges.
	Centralize vertically.
	Align the lower edges.
	Arrange evenly across the width.
	Arrange evenly in height.
	Center horizontally in the window.
	Center vertically in the window.
	Set the same width as the first selected object.
	Set the same height as the first selected object.

Table 1.7 - Contents of the "Alignment" toolbar

Now let us move on to the settings of the designer (Figure 29).
To set the options of the designer, use the menu command "View"→ "Settings...".



Designer Options

Grid

Type / Size

● Centimeters: 0,1 cm ☑ Show grid

○ Inches: 0,1 in ☑ Align to Grid

○ Pixels: 4 pt ☑ Stick to guides

Dialog form: 4 pt ☑ Use guides as anchor

Fonts

Code window: Courier New, Size 10

Memo editor: Arial, Size 10

☑ Use object's font settings

Colors

Workspace

Tool windows

□ LCD grid color

Code complition and Syntax memo

☑ Show script variables

☑ Show report objects

☑ Show Rtti variables

Tab stops: 2

□ MultiByte support

Other

□ Show editor after insert

☑ Show band captions

□ Free bands placement

Gap between bands: 4 pt

Guides stick accuracy: 1,5 pt

Restore defaults | OK | Cancel

Figure 30 - Designer settings form

In this form you can specify the used units of measurement (centimeters, inches, pixels), specify the grid spacing for each unit. You can also switch the units of measure in the designer by double-clicking on the left side of the status bar where the current units of measure are displayed.

You can immediately set up grid display and alignment of objects to the grid.
You can also do it using the buttons on the "Standard" toolbar in the designer.

In addition, there is an option to configure the font for the code editor window and for the editor of the "Text" object.

If the "Use object font" option is enabled, the font in the text editor window will correspond to the font of the edited object.

If you are not satisfied with the white background of the designer's work field and service windows, you can change it using the "Work field" and "Windows" buttons. The "Mesh Color for LCD Monitor" option slightly increases the contrast of the mesh lines, which allows them to be better seen on LCD displays.

The "Call editor after inserting" option controls the process of inserting new objects. If this option is enabled, its editor will be shown each time an object is inserted. When inserting a large number of empty objects, it is better to disable this option.

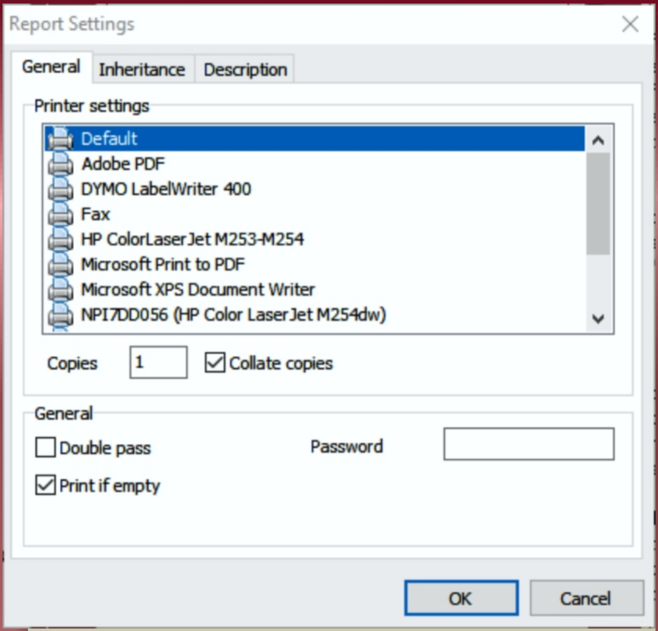
By disabling the "Show band headers" option, you can disable band headers to save space on the page. In this case, the name of the band is written inside it.

If you disable the "Show drop-down list of fields" option, you can disable showing the drop-down list when pointing the mouse cursor over the object "Text" connected to the data. It can be necessary if there are many small objects in the report.

The "Free band placement" option disables band binding to the sheet. By default, this option is disabled, and bands are automatically grouped on the page according to their purpose. The gap between bands is set in the "Band gap" field.

1.2 REPORT OPTIONS

The report also has its own settings (Figure 31). The window with report parameters is available from the menu "Report" -> "Settings...". The form of settings has three tabs: Basic, Inheritance, Description.



Report Settings

General | Inheritance | Description

Printer settings

- Default
- Adobe PDF
- DYMO LabelWriter 400
- Fax
- HP ColorLaserJet M253-M254
- Microsoft Print to PDF
- Microsoft XPS Document Writer
- NPI7DD056 (HP Color LaserJet M254dw)

Copies 1 ☑ Collate copies

General

☐ Double pass Password

☑ Print if empty

Figure 31 - Report settings window. Tab "General"

On the "Basic" tab, you can attach the report to one of the printers installed in the system. This means that the report will be printed to the selected printer by default. If there are several printers in the system, this function allows, for example, to attach text documents to monochrome printers, and documents with graphics - to color printers. As you can see on Figure 31 there is a "Default" printer in the list of printers. When it is selected, the report will be displayed on the current active printer.

Here you can specify the number of copies of the report to be printed and the parse function. The values specified in this window will be displayed in the "Print" window.

If the "Two passes" checkbox is checked, the report will be generated in two stages. On the first pass, the report is generated, it is divided into pages, but the result is not saved anywhere. On the second pass, the normal report generation is performed with saving the result in the flow.

Why do you have to do two passes for? This option is most often used when there is a mention of the total number of pages in a report, i.e. information like "Page 1 of 15".

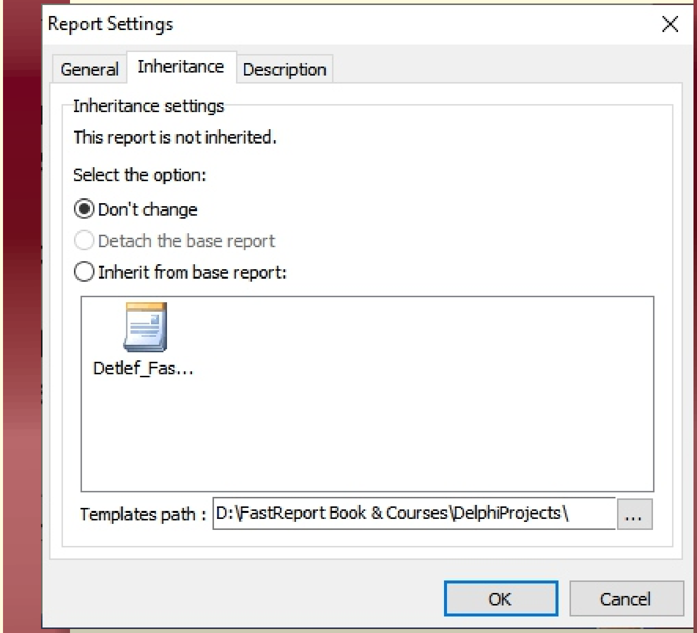
The total number of pages is counted on the first pass and can be accessed via the system variable TOTALPAGES. The most common error is an attempt to apply this variable in a single-pass report, in which case it returns 0.

Another area of application is to perform any calculations on the first pass and display the results on the second pass. For example, when you want to display an amount in the group header, which is usually calculated and displayed in the group basement. Such calculations are connected with the use of the built-in FR language.

The "Print if empty" checkbox allows you to build a report that does not contain any data lines. If this option is disabled, no empty reports will be generated.

The "Password" field allows you to specify a password that will be requested from the user when opening the report.

The controls on the "Inheritance" tab (Figure 32) allow you to set the report inheritance options.



Report Settings

General | Inheritance | Description

Inheritance settings

This report is not inherited.

Select the option:

● Don't change

○ Detach the base report

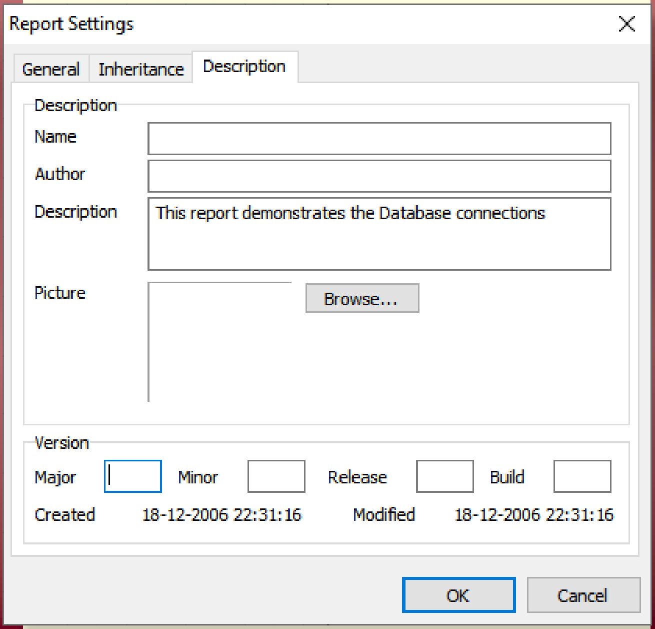
○ Inherit from base report:

Detlef_Fas...

Templates path : D:\FastReport Book & Courses\DelphiProjects\

Figure 32 - "Inheritance" tab

Here you can find out from which report the current report is inherited, disconnect the basic report (*in this case the report ceases to be inherited and becomes independent*), and also inherit the report from one of the selected ones. For more information about inheritance, see chapter "Report Inheritance".
The controls on the "Description" tab (*Figure 33*) allow you to set the description of the report.



Description
Name:
Author:
Description: This report demonstrates the Database connections
Picture:

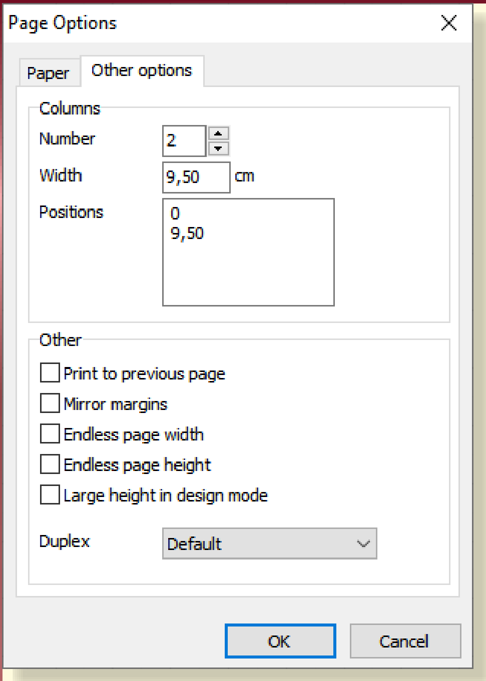
Version
Major: | Minor: | Release: | Build:
Created: 18-12-2006 22:31:16 | Modified: 18-12-2006 22:31:16

Figure 33 - "Description" tab

In this tab, you can specify the name of the report (it is displayed in the preview window and is assigned to the print job), author, description, picture and version of the report. All these parameters are optional and serve for information purposes.

1.3 PAGE PARAMETERS

Like in text editors, in FR Studio you can set page parameters, namely size, margins, paper orientation and so on. Page parameters are available through "File"->"Page Parameters..." menu or by double-clicking on an empty page. The window has two tabs:



Columns
Number: 2
Width: 9,50 cm
Positions: 0 9,50

Other
□ Print to previous page
□ Mirror margins
□ Endless page width
□ Endless page height
□ Large height in design mode

Duplex: Default

Figure 34 - Page settings. Tab "Other Options"

The "Print on front page" checkbox allows you to start printing the page starting from the free space on the previous sheet. This option can be used if the report template consists of several sheets or when printing batch (composite) reports.
The "Mirror Fields" option allows you to swap the right and left margins of a page for even pages when viewing or printing a report.

Options "Infinite Width", "Infinite Height" allow you to enable the mode in which the page size increases depending on the output data. In this case, in the preview window you see one dimensionless page containing all data (unlike the normal mode, when a new page is formed when reaching the end of the page).

The option "Greater height in designer" allows you to increase the height of the page by several times. This can be useful if there are many bands on the page. The actual page height does not change when building a report.
In the next issue we will extend the learning session,.

Install Aware Installer soon not only for Windows

by Detlef Overbeek

ABSTRACT:
InstallAware is a program that started as a specific Windows installer for Visual Studio. It has grown to something very professional and is now even going to gain ground on other OS's. There is a free version that I will present to you here.

INTRODUCTION

In several occurrences I have used Install programs like Inno Setup. Problem with this program was that you need to find out by trial and error how it works. And that consumes a lot of time.
For testing reasons I took the simplest version of Install Aware and found that I could install a program almost instantly within a few minutes. AND working.

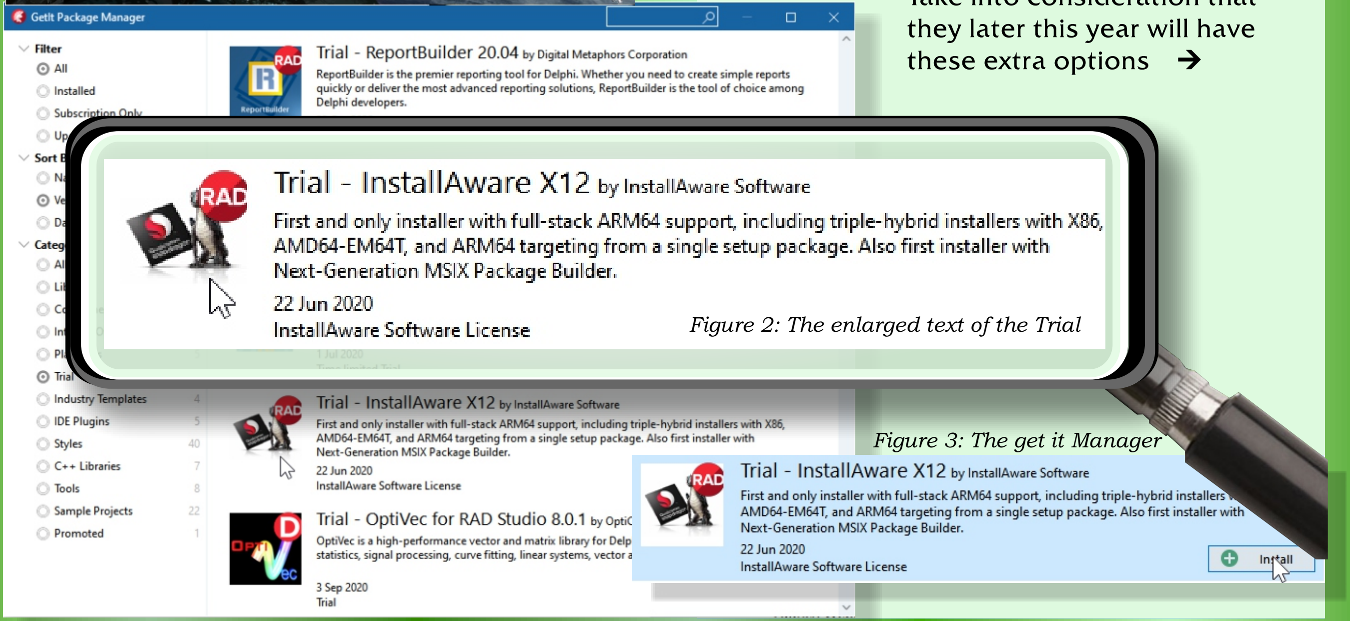
InstallAwareX12
MSI(X), EXE, APP-V, VMWARE, HYPER-V, FACEBOOK, TWITTER, ETH
https://www.installaware.com
© 1996-2020 InstallAware Software. All rights reserved.

Figure 1: The starting screen

There is of course a financial factor in this: Inno Setup is free, so costs nothing at all.

INSTALLAWARE also has a free version: InstallAware can be found in Delphi in the **GetIt Package Manager**, located at the **Welcome - Page** at the subject **Promoted**, I found it by setting the filter to All / Name / Trial.
Simply typing the name in the search box didn't help. Maybe because the **Embarcadero GetIt** server could not be reached, which happens quite a lot.
There is of course an other way: go to their website, but I found that they explicitly say the free installer is a VisualStudio Extension.
So I simply installed the version from Get it.

Now for testing reasons I had a look at the other options for downloading a free or trial version: I ran into `https://www.installaware.com/landing/code-gear.html` which confused me a little.
I thought Code Gear didn't exist any more?
What I found wasn't exactly a free trial.
Again I tried to find the free version download at their site:
`https://www.installaware.com/downloads-product-downloads.htm`
Here I get a warning that you will need Visual Studio. That's not what I want.
So I'll go back to the version I installed by choosing install through the GetIt Package Manager.
Take into consideration that they later this year will have these extra options →



Trial - ReportBuilder 20.04 by Digital Metaphors Corporation
ReportBuilder is the premier reporting tool for Delphi. Whether you need to create simple reports quickly or deliver the most advanced reporting solutions, ReportBuilder is the tool of choice among Delphi developers.

Trial - InstallAware X12 by InstallAware Software
First and only installer with full-stack ARM64 support, including triple-hybrid installers with X86, AMD64-EM64T, and ARM64 targeting from a single setup package. Also first installer with Next-Generation MSIX Package Builder.
22 Jun 2020
InstallAware Software License

Trial - OptiVec for RAD Studio 8.0.1 by Opti
OptiVec is a high-performance vector and matrix library for Delp statistics, signal processing, curve fitting, linear systems, vector a
3 Sep 2020
Trial

Figure 2: The enlarged text of the Trial

Figure 3: The get it Manager

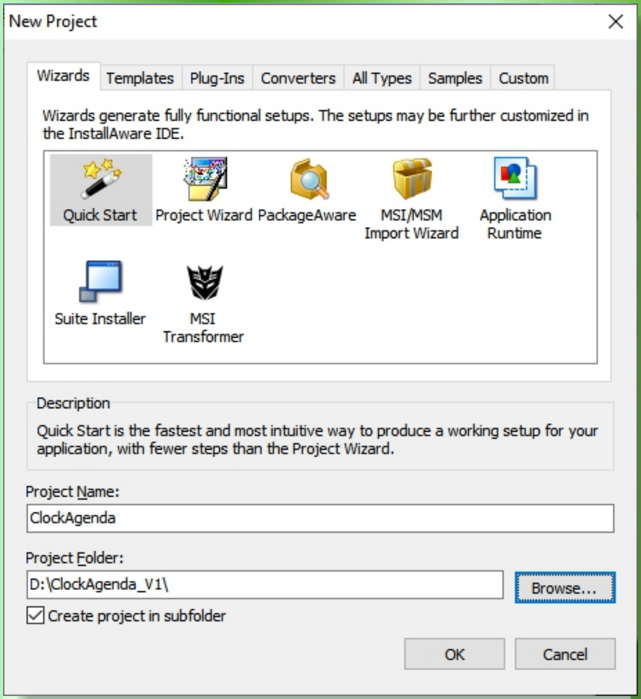
- **Three Major Platforms:**
  Code and compile natively on Linux, Mac, and Windows. Build for any platform from any environment – ex: you will be able to build Mac installers on a PC, and vice versa!
- Advanced Dialogue Editing: Enjoy a dialogue editor virtually identical to its Windows Installer brethren, with the same ease of use, drag&drop design experience, and visual configuration.
- Conditionally Flowing Setup Script: Code, commands, and syntax to remain as similar as possible to InstallAware's existing scripting language, and a zero learning curve.
- NO JAVA: InstallAware Multi Platform offers 100% native code setup engines for Linux, Mac, and Windows –
  ZERO DEPENDENCIES and runtime free delivery!

Since I downloaded the program a few weeks ago I had to restart my memory and started the program again. Now I need to (see figure) state that I would like to try it again.
As I hoped the program started. To begin with: let's do a quick start.

QUICK START

Running through the Install setup is easy. Because I wanted to do something useful, I tried a program example I made earlier and which could use an update. The **ClockAgenda**. The clock for the desktop you can put as a small app somewhere on your desktop which I want to give an update in the next issue: create a colouring menu, add a startup for the app when windows starts and make it also in Lazarus, so you can choose what you want: Windows/Linux or Mac.
The code is of course as always free...
Hopefully by that time the new **InstallAware** abilities will be available...

The Install-Setup is decorated with all the windows that pop up during the process.
You'll find the extra info under each picture.



Wizards generate fully functional setups. The setups may be further customized in the InstallAware IDE.

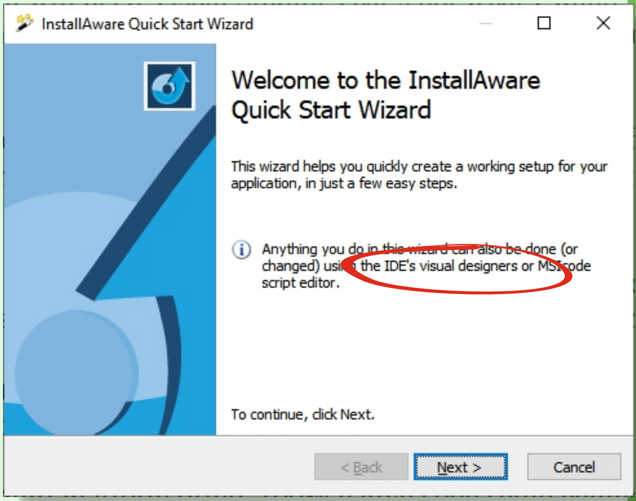
Description
Quick Start is the fastest and most intuitive way to produce a working setup for your application, with fewer steps than the Project Wizard.

Project Name: ClockAgenda

Project Folder: D:\ClockAgenda_V1\

☑ Create project in subfolder

Figure 4: The Quick Start Option which we test here.



Welcome to the InstallAware Quick Start Wizard

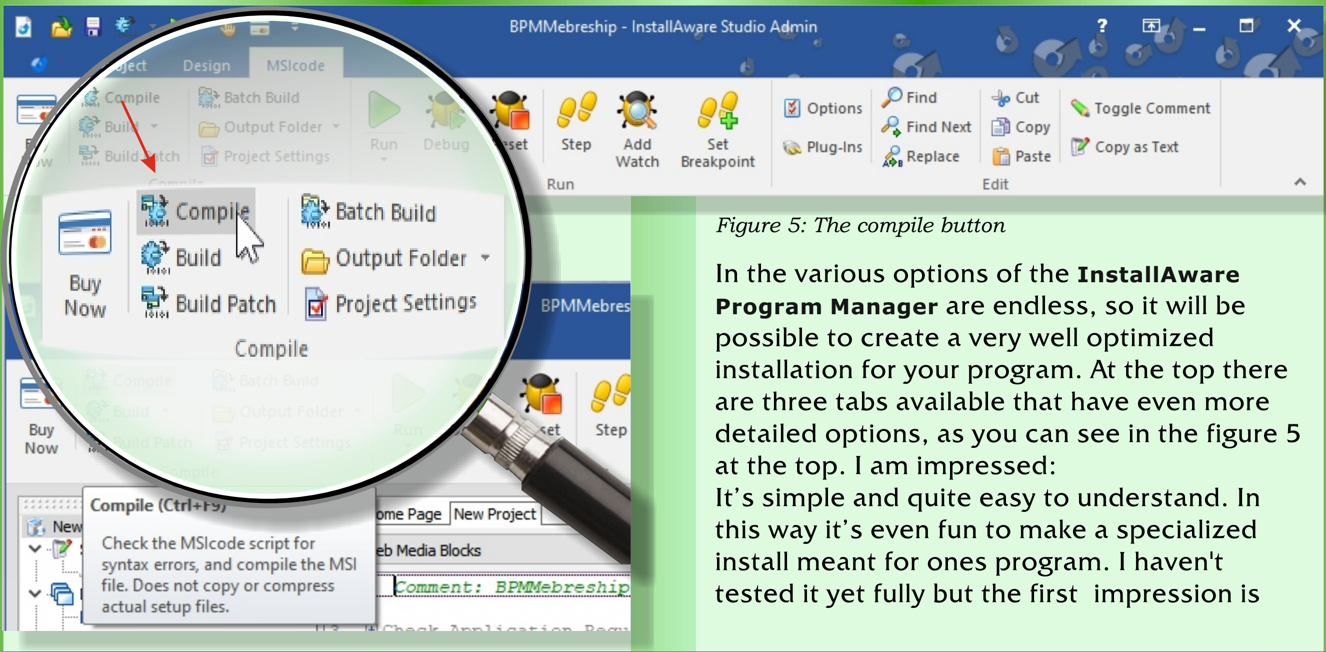
This wizard helps you quickly create a working setup for your application, in just a few easy steps.

Anything you do in this wizard can also be done (or changed) using the IDE's visual designers or MSIcode script editor.

To continue, click Next.

Figure 5: The subtext on the wizard shows the importantant messages

Anything you do in this wizzard can also be done (or changed) using the **IDE visual desginer** (see figure 7, page 4/7) or **MSIcodesript editor** (see figure 8, page 5/7).



Compile (Ctrl+F9)

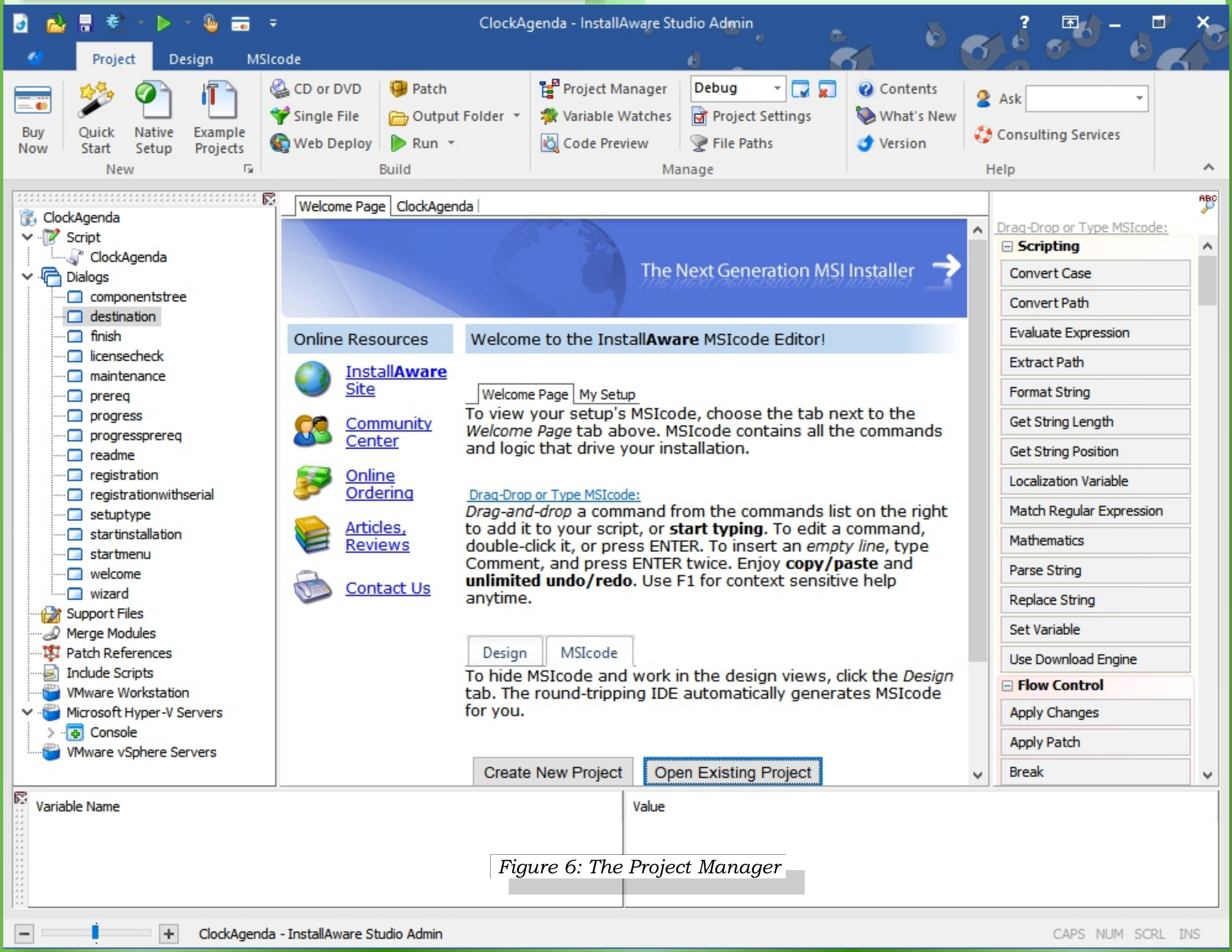
Check the MSIcode script for syntax errors, and compile the MSI file. Does not copy or compress actual setup files.

```
Comment: BPMMebreship
```

Figure 5: The compile button

In the various options of the **InstallAware Program Manager** are endless, so it will be possible to create a very well optimized installation for your program. At the top there are three tabs available that have even more detailed options, as you can see in the figure 5 at the top. I am impressed:

It’s simple and quite easy to understand. In this way it’s even fun to make a specialized install meant for ones program. I haven't tested it yet fully but the first impression is



Welcome to the Install**Aware** MSIcode Editor!

Welcome Page | My Setup

To view your setup's MSIcode, choose the tab next to the *Welcome Page* tab above. MSIcode contains all the commands and logic that drive your installation.

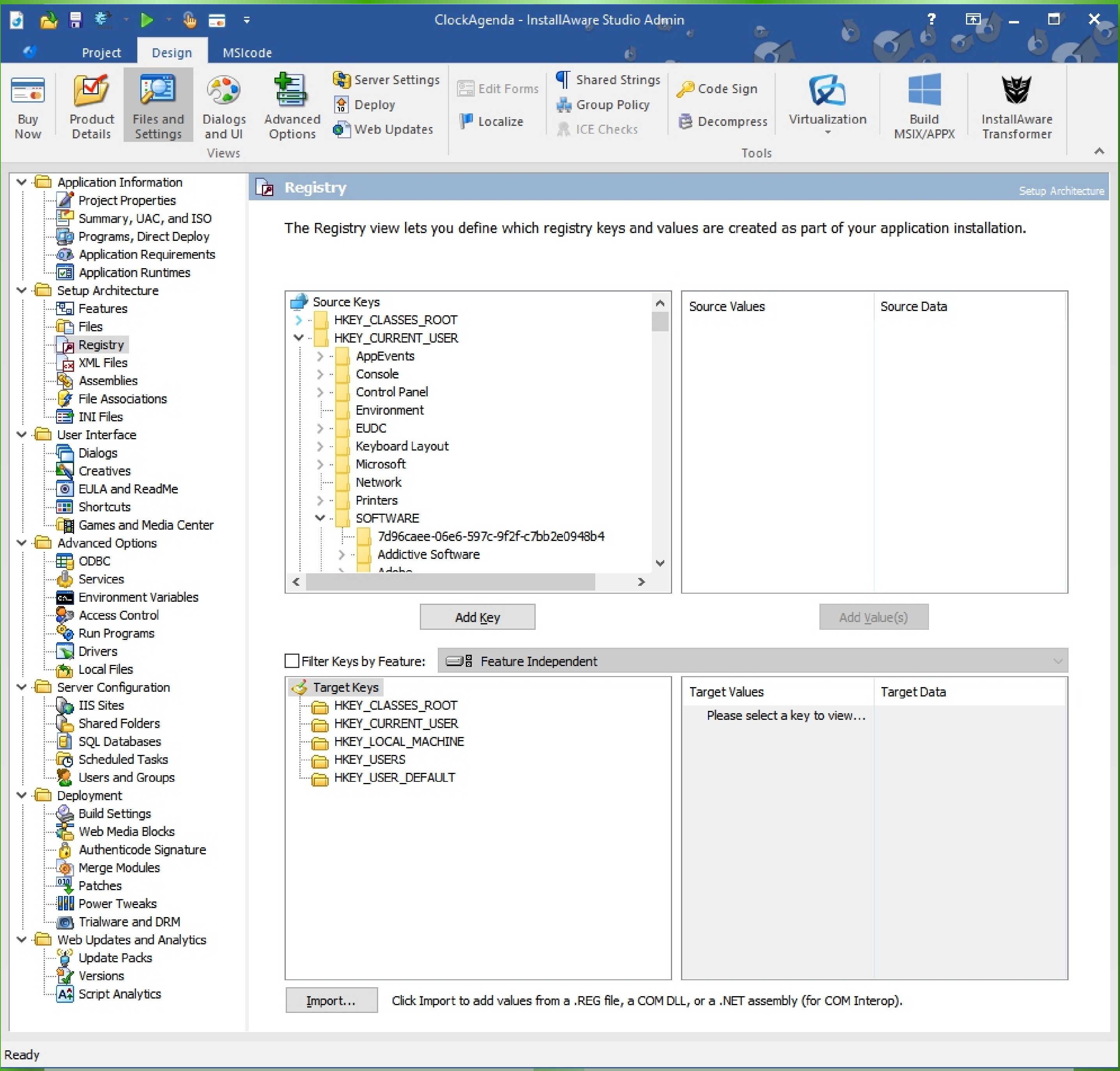
Drag-Drop or Type MSIcode:

Drag-and-drop a command from the commands list on the right to add it to your script, or **start typing**. To edit a command, double-click it, or press ENTER. To insert an *empty line*, type Comment, and press ENTER twice. Enjoy **copy/paste** and **unlimited undo/redo**. Use F1 for context sensitive help anytime.

Design | MSIcode

To hide MSIcode and work in the design views, click the *Design* tab. The round-tripping IDE automatically generates MSIcode for you.

Figure 6: The Project Manager



The Registry view lets you define which registry keys and values are created as part of your application installation.

Please select a key to view...

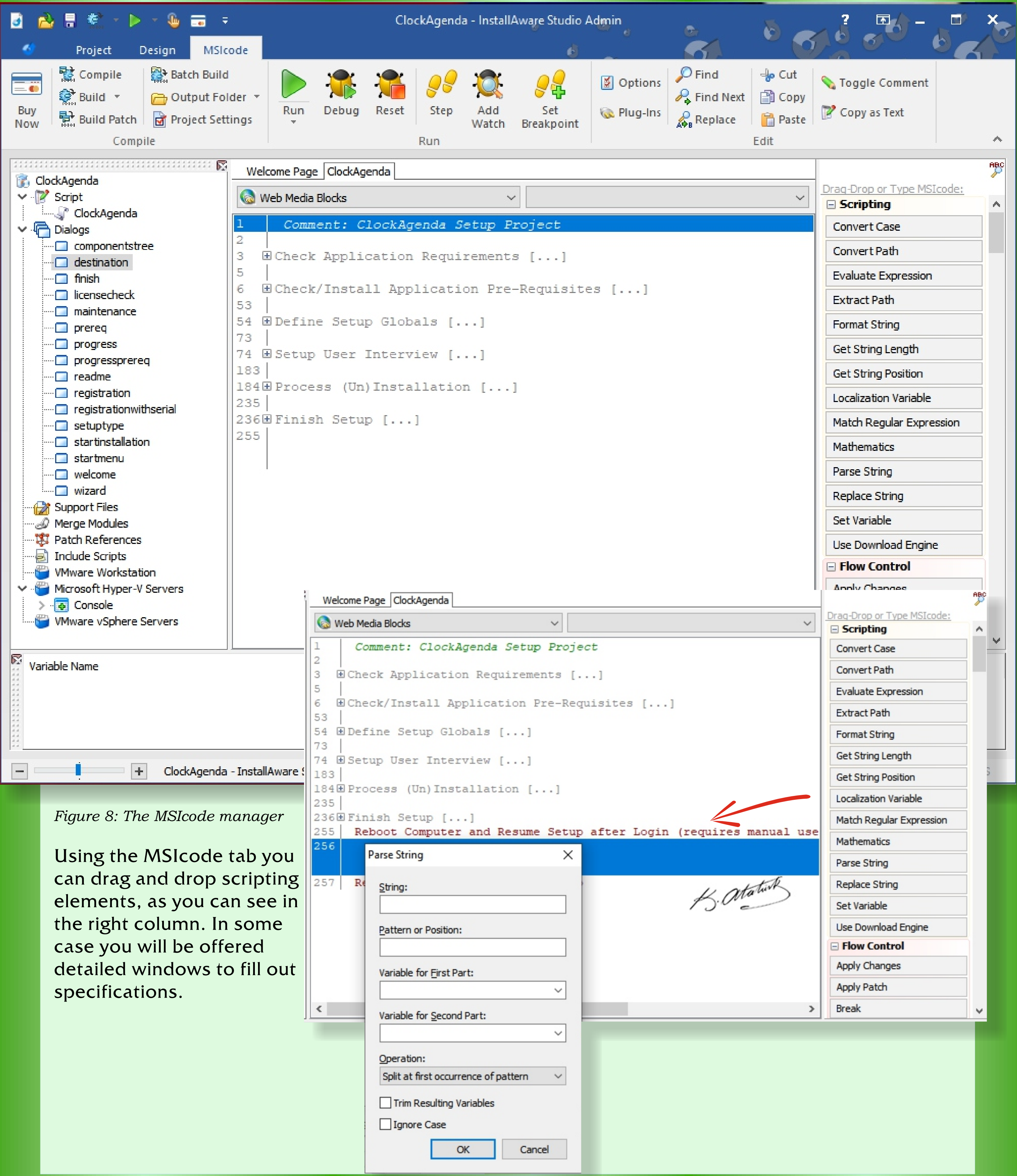
Click Import to add values from a .REG file, a COM DLL, or a .NET assembly (for COM Interop).

Ready

Figure 7: The Visual Designer

The Visual Designer shows at the left a list of subjects which show a more detailed plan at the The main window in the middle/right section. Its worth to browse through them. It will maybe even give you new ideas. I think if they are capable of doing this for various OS's this Installer is a must have.

Here you see what you could do with the Windows registry, if you need a controlled installation.



```
Comment: ClockAgenda Setup Project

Check Application Requirements [...]

Check/Install Application Pre-Requisites [...]

Define Setup Globals [...]

Setup User Interview [...]

Process (Un)Installation [...]

Finish Setup [...]
Reboot Computer and Resume Setup after Login (requires manual use

Re
```

Parse String

String:

Pattern or Position:

Variable for First Part:

Variable for Second Part:

Operation: Split at first occurrence of pattern

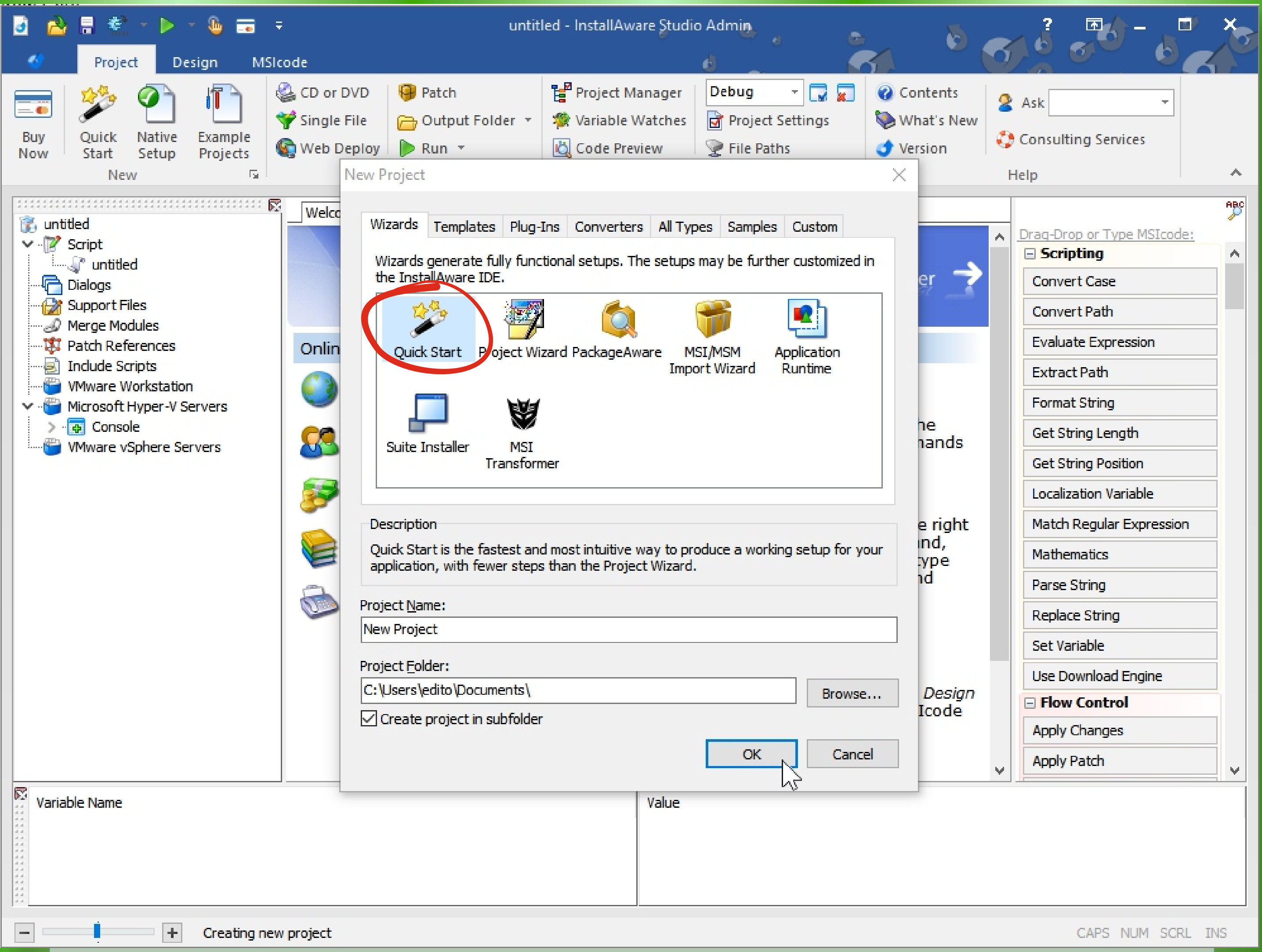
□ Trim Resulting Variables

□ Ignore Case

OK Cancel

Figure 8: The MSIcode manager

Using the MSIcode tab you can drag and drop scripting elements, as you can see in the right column. In some case you will be offered detailed windows to fill out specifications.



New Project

Wizards | Templates | Plug-Ins | Converters | All Types | Samples | Custom

Wizards generate fully functional setups. The setups may be further customized in the InstallAware IDE.

Quick Start | Project Wizard | PackageAware | MSI/MSM Import Wizard | Application Runtime | Suite Installer | MSI Transformer

Description

Quick Start is the fastest and most intuitive way to produce a working setup for your application, with fewer steps than the Project Wizard.

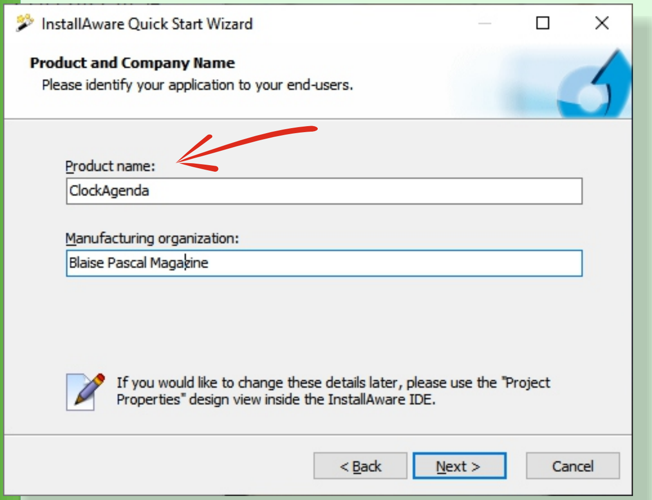
Project Name: New Project

Project Folder: C:\Users\edito\Documents\ Browse...

☑ Create project in subfolder

OK Cancel

Figure 9: The Quick Start menu



InstallAware Quick Start Wizard

Product and Company Name

Please identify your application to your end-users.

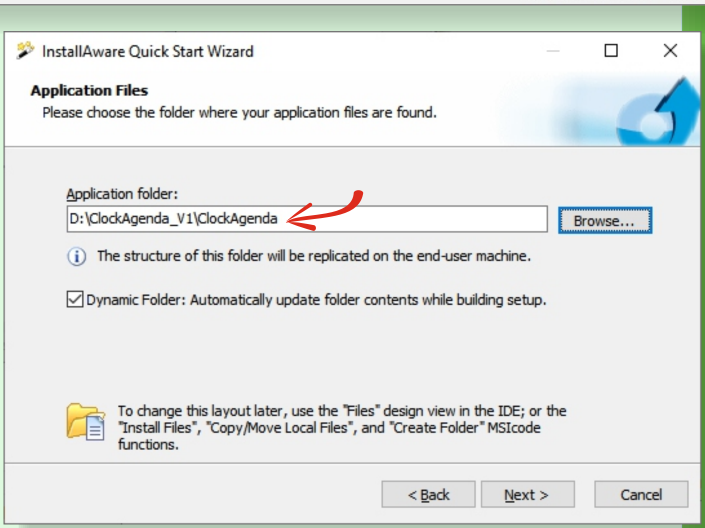
Product name: ClockAgenda

Manufacturing organization: Blaise Pascal Magazine

If you would like to change these details later, please use the "Project Properties" design view inside the InstallAware IDE.

< Back Next > Cancel

Figure 10: Filling in the product name



InstallAware Quick Start Wizard

Application Files

Please choose the folder where your application files are found.

Application folder: D:\ClockAgenda_V1\ClockAgenda Browse...

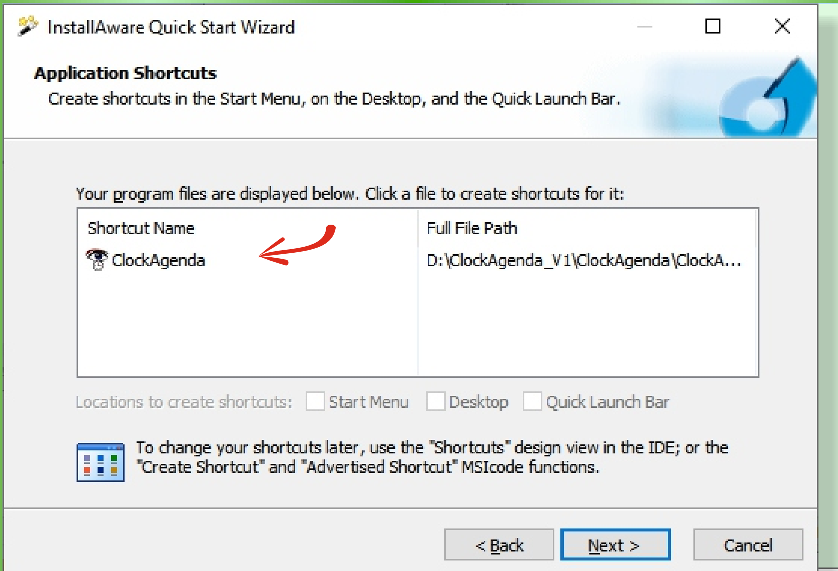
The structure of this folder will be replicated on the end-user machine.

☑ Dynamic Folder: Automatically update folder contents while building setup.

To change this layout later, use the "Files" design view in the IDE; or the "Install Files", "Copy/Move Local Files", and "Create Folder" MSIcode functions.

< Back Next > Cancel

Figure 11: Filling in the Application folder



InstallAware Quick Start Wizard

Application Shortcuts

Create shortcuts in the Start Menu, on the Desktop, and the Quick Launch Bar.

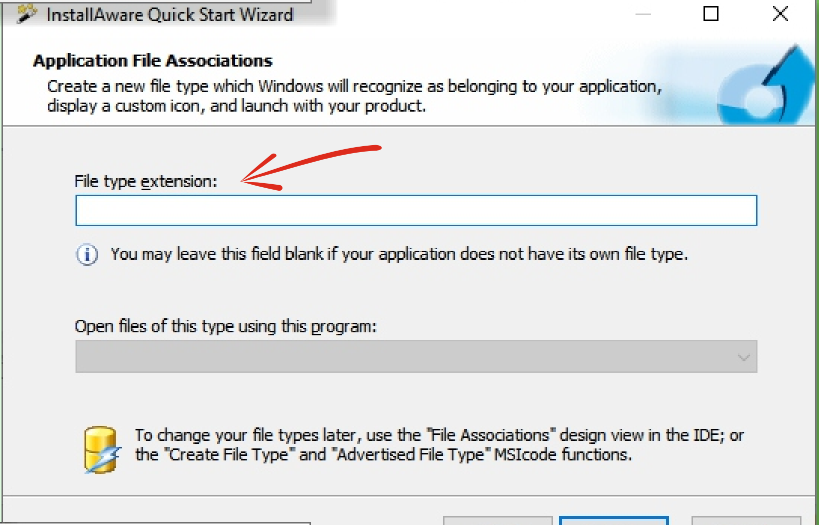
Your program files are displayed below. Click a file to create shortcuts for it:

Shortcut Name	Full File Path
ClockAgenda	D:\ClockAgenda_V1\ClockAgenda\ClockA...

Locations to create shortcuts: □ Start Menu □ Desktop □ Quick Launch Bar

To change your shortcuts later, use the "Shortcuts" design view in the IDE; or the "Create Shortcut" and "Advertised Shortcut" MSIcode functions.

Figure 12: adding the executable



InstallAware Quick Start Wizard

Application File Associations

Create a new file type which Windows will recognize as belonging to your application, display a custom icon, and launch with your product.

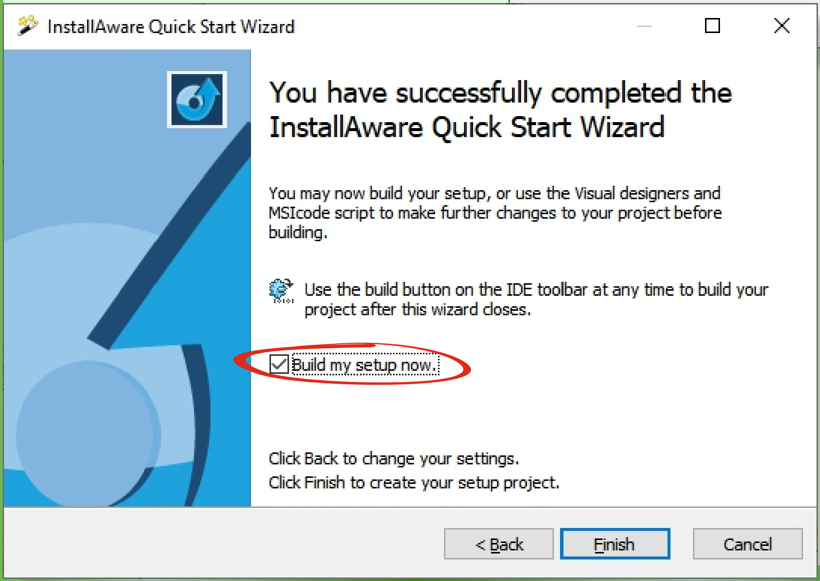
File type extension:

You may leave this field blank if your application does not have its own file type.

Open files of this type using this program:

To change your file types later, use the "File Associations" design view in the IDE; or the "Create File Type" and "Advertised File Type" MSIcode functions.

Figure 13: If you need you can add a file type extension



InstallAware Quick Start Wizard

You have successfully completed the InstallAware Quick Start Wizard

You may now build your setup, or use the Visual designers and MSIcode script to make further changes to your project before building.

Use the build button on the IDE toolbar at any time to build your project after this wizard closes.

☑ Build my setup now.

Click Back to change your settings.
Click Finish to create your setup project.

Figure 14: Build the stup

What's coming in TMS WEB Core v1.7 Ancona

The new version v1.7 of TMS WEB Core has been in development for about 6 months by now. Many of its features were already in development in parallel to v1.6. And yes, our team already is working on v1.8! It will not come as a surprise to you that in v1.7 there are new game changing features in TMS WEB Core also.

Those who have been following the TMS WEB Core development since the first version TMS WEB Core v1.0 Brescia will know that we name the releases after cities along the famous historic race “MilleMiglia”. To be more precise, the legendary race of 1955. And as such, after we visited the city Pesaro with v1.6, for v1.7 we land in Ancona. The historical meaning of the word “Ancona” is elbow which is commonly associated with the shape of the coastline. After the 'elbow' Ancona, there are new sights towards the more southern coasts of Italy.

Enough history! Let's bring an overview of what our team has been working on for TMS WEB Core v1.7 Ancona.

❶ Components wrapping browser API for local file access

The W3C consortium proposed an API forlocal file access from Web browser applications and Google Chrome implements it already. Given that Microsoft uses the Google Chromium engine, it is also available in Microsoft Edge Chromium. With TMS WEB Core, you can take a head start, explore, and start using this with its three components today:

- TWebLocalTextFile
- TWebLocalBinaryFile

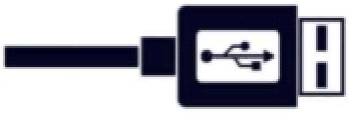
These components allow you to directly open text and binary files from the local file system as well as traverse the local file system's folder structure from a Web application. Of course, for security reasons, the users have to give their consent first.

❷ Popup menu component and popup menu support in components

In TMS WEB Core, we offered the regular TWebMainMenu since its inception and now we completed this trend with the new TWebPopupMenu. It works very similar to a Windows VCL TPopupMenu. Simply drop a TWebPopupMenu on your Web forms and assign it to control.PopupMenu to obtain an automatic context menu for controls.

TMS WEB Core rocks!

❸ USB device access

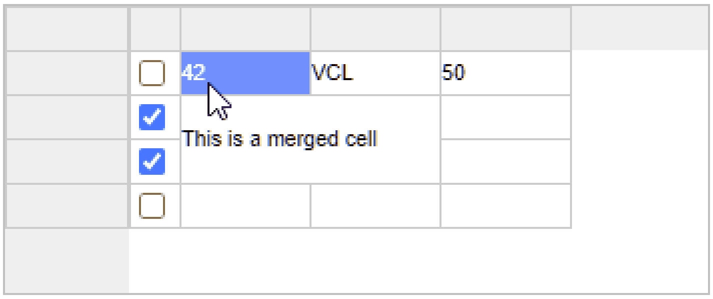


The times that software developers thought you could never use a Web application for controlling your machine's hardware are officially over. We already introduced support for Bluetooth with our TWebBluetooth component, and with this release we add two new components TWebUSBSerial and TWebUSBHID that allow you to write applications communicating with locally connected USB devices using a serial protocol or the HID protocol. This opens up a whole new field of applications that can be implemented using Web technologies.

❹ TWebStringGrid & TWebDBGrid extensions

We added a whole range of new grid features in TWebStringGrid and TWebDBGrid. Thereare now:

- Methods to insert & remove rows
- A range of different inplace editor types
- Cell merging and cell splitting
- Loading and saving to streams
- Loading from a stringlist with CSV data
- Direct access to grid data as TJSArray
- Add checkboxes to the grid

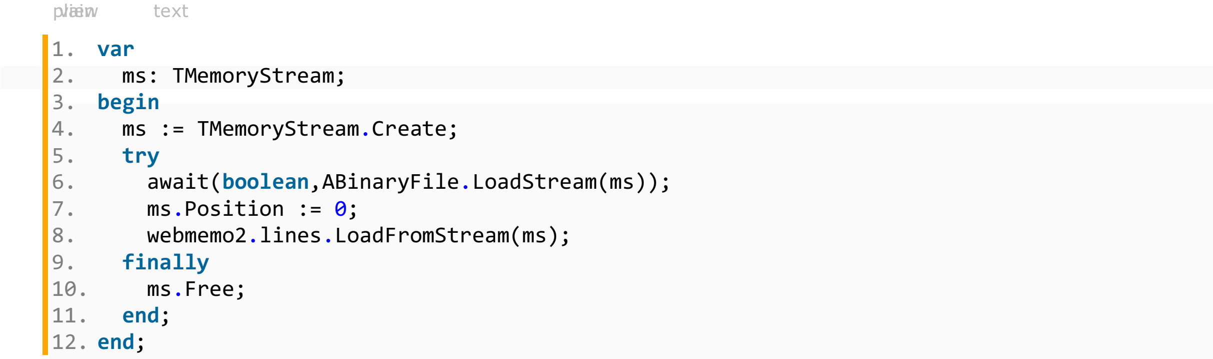


	☐	42	VCL	50
	☑	This is a merged cell		
	☑			
	☐			

❺ Async methods

There is no way around it in the Web browser, there are different APIs that can only be used in an asynchronous way. The reason for this is very clear: Always guarantee a responsive UI even when lengthy operations are taking place in the application.

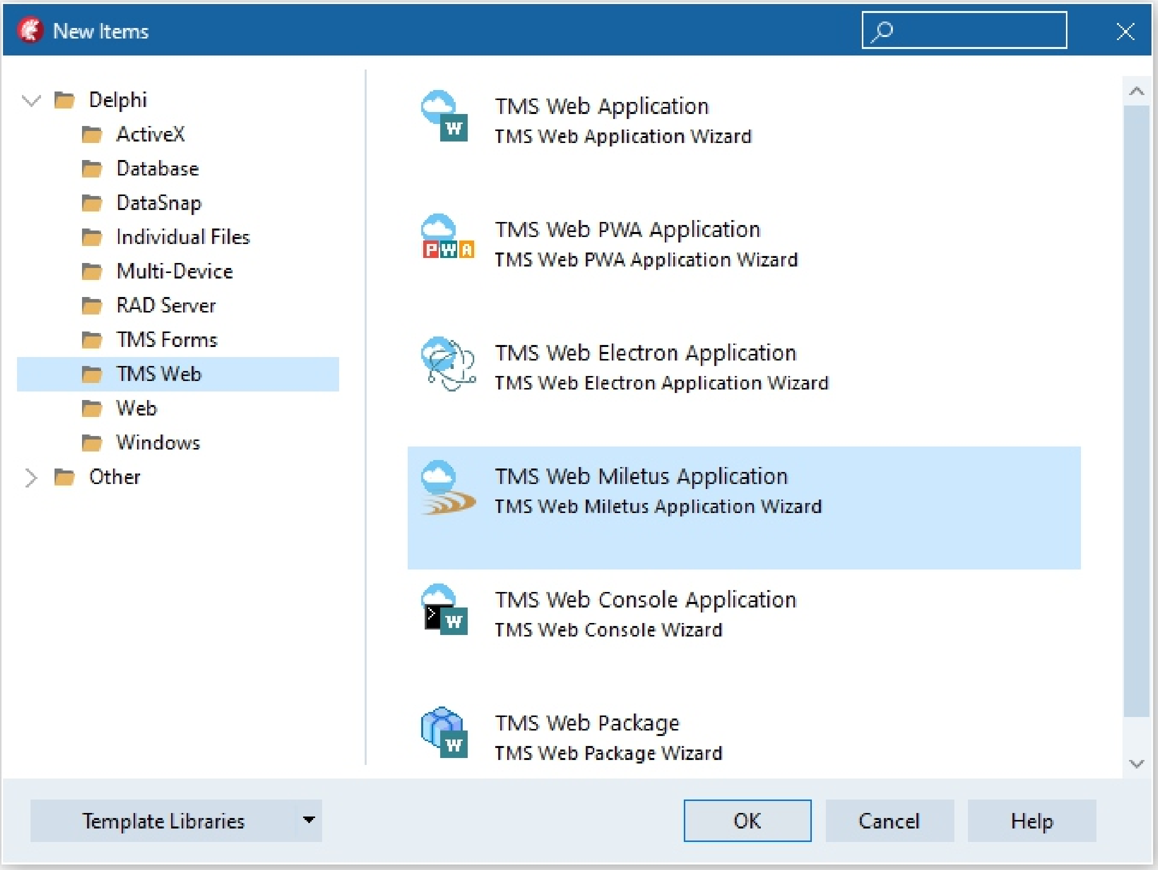
Many developers struggle with implementing sequential logic for processes that occur asynchronously. We already offered anonymous methods that can be used to handle results of methods asynchronously, but, in v1.7, we added a whole range of async methods to classes. With a simple await() construct around an async method, you can write your code as if it were synchronous and sequential, but, behind the scenes, the browser still handles it asynchronously and keeps your UI responsive. The components to perform HTTP(S) requests for example, load data from a URL, load images from URL, etc... now all have async equivalents making you write code that is cleaner and consequently easier to maintain.



```
var
  ms: TMemoryStream;
begin
  ms := TMemoryStream.Create;
  try
    await(boolean,ABinaryFile.LoadStream(ms));
    ms.Position := 0;
    webmemo2.lines.LoadFromStream(ms);
  finally
    ms.Free;
  end;
end;
```

❻ Miletus framework

This is probably the biggest feature of TMS WEB Core v1.7! Miletus is a framework for creating native desktop applications with Web technologies. It permits to create wonderful and modern user-interfaces using (and in many cases reusing) HTML/CSS templates.



Still, it allows to run as a desktop application and to access desktop features normally reserved for a native desktop app. This includes:

- local file access
- operating system menu
- drag & drop interaction with the operating system
- access to the taskbar
- access to local databases (SQLite, MSSQL, PostgreSQL, mySQL, MS Access)
- so much more...

This first introduction of Miletus in TMS WEB Core offers the capability to create Win32 and Win64 native single EXE file applications with a size below 10MB. And in the near future, we will add macOS and Linux to that.

Stay tuned for another blog with a more detailed overview of the Miletus features. An additional benefit of Miletus is that we developed this framework from the ground up as opposed to a similar framework called Electron that is from a 3rd party. This means that for the future, we have full control to design any feature set we want for Miletus.

❼ TWebSocketClient method added to send & receive binary data

When communicating with a web socket server, TWebSocketClient can now directly send & receive binary data whereas in previous versions this had to be sent as Base64 encoded string data.

❽ TWebForm extended

We added more flexibility to the TWebForm. This includes:

- Option to have a close button in the caption
- Property to choose whether a shadow border is used or not
- ElementCaptionClassName property added to allow the use of CSS for the caption
- OnDOMContentLoaded event added, that signals when the browser loaded the entire DOM content
- OnHashChange event added to handle browser back & forward buttons

❾ New server-side filtering + multi-tenant support in TWebFireStoreClientDataSet

We have also significantly improved TWebFireStoreClientDataSet. Now,TWebFireStoreClientDataSet supports to create more flexible and more performant Web client applications using Google's Firestore as the backend for data storage. In the new version, it is possible to specify server-side filters.
This facilitates not only multi-tenant scenarios but also increases performance by minimizing data returned from the server.

❿ Numerous smaller extensions & improvements to various components & IDE integration

Not only was the TMS WEB Core framework improved in almost any corner, the IDE integration got further fine-tuning as well. The capability to use environment variables in the compiler path with the $() syntax was added. The same is possible with custom compiler directives.
And finally, also the pas2js compiler & RTL were updated to the latest version v2.0.4.

Planning

The first step is that we will release the beta version of TMS WEB Core v1.7 in the next few days to all TMS ALL-ACCESS users and registered users of TMS WEB Core. We expect the upgrade process should be smooth and painless as we paid a huge amount of attention to backwards compatibility. This is proven by the fact that we did not have to make any modifications to the meanwhile over 100 demos included. However, we want to put the v1.7 beta into your hands first and listen closely to your feedback. That allows us to apply further polish where needed.

After this beta period, TMS WEB Core v1.7 Ancona will be officially released, and our team will continue to work hard on the next milestones.

TMS WEB Core for Visual Studio Code v1.3

The new version v1.3 of TMS WEB Core for Visual Studio Code is also around the corner. The major new feature of v1.3 will be extensibility via 3rd party components with a package system and with this, the whole portfolio of TMS FNC components will also become usable at design-time in Visual Studio Code. The TMS WEB Core framework will be at the same level as TMS WEB Core v1.7 for Delphi & Lazarus. The only feature that will not yet be included in TMS WEB Core for Visual Studio Code v1.3 is Miletus support. That is reserved for v1.4 and with this, it is expected to already bring Miletus for Windows, macOS, and Linux.

TMS WEB Core v1.8

There are several ongoing new developments for TMS WEB Core that we cannot reveal yet, but it is clear that v1.8 will come with Miletus support for macOS and Linux. We already have proof of concept versions working in the lab, but further work is needed before this is considered “feature complete”. This will very likely be the case in v1.8.

Get started today!

The only constant in software development is change. Web technologies open up exciting new capabilities for us Delphi, Lazarus, or Object Pascal developers in general. It allows us to create with RAD component-based techniques and a strongly-typed object-oriented language no-deploy web applications. Further, we may create PW As for iOS and Android that can be installed and run offline when needed. Best of all, we do not need to conform to Apple or Google store censorship. Alternatively, you can create cross-platform desktop applications with Electron or Miletus at the same time.

We are of course curious to hear what you like the most about TMS WEB Core v1.7 Ancona or what you wish to see in future versions of TMS WEB Core! We look forward to discussing all these great milestones with you!

Webinar

If you want to see the new capabilities in TMS WEB Core 1.7 Ancona demonstrated live and ask questions, we have organized this opportunity for you! Attend our free webinar on April 8, 2021, from 15h00 to 16h00 UTC (17h00 18h00 CEST).
Register for the webinar here `https://www.tmssoftware.com/site/tmswebacademy.asp?id=75`

Lazarus: Webform inside Lazarus

A small revolution for Pascal on the desktop

By Michaël Van Canneyt

ABSTRACT
The Chrome browser is by far the most popular browser of the moment. The technology underlying this browser is freely available: the Chromium project distributes the rendering and Javascript engine underlying this browser. In this article we show how to embed the Chromium browser in your Lazarus Application.

starter expert

INTRODUCTION

Survey after survey shows that the Chrome browser is by far the most popular browser of this moment. Google has made the technology underlying this browser freely available:
The CHROMIUM EMBEDDED FRAMEWORK, managed by the Chromium project, makes the RENDERING & JAVASCRIPT ENGINE available to any programmer.
This can be used to do simple matters such as show webpages in your application:
this can be used to embed „help" or even a complete „helpdesk" in your application.
If you're adventurous, you can try to build a complete browser.
But one can go even further: projects such as NODE.JS, ELECTRON, ATOM, SPOTIFY and many others use this technology since a long time to allow you to run Web Applications on the desktop or on a tablet:
The CHROMIUM EMBEDDED FRAMEWORK runs on all major platforms, so when using it, your program will run on all major programs, with just a single codebase.

We can now create a program that has a native part, programmed in Pascal, and a GUI part running in the browser - also programmed in Pascal.

Now why would one want to do this ?
Isn't it better to run native applications ?
It all depends on what your application is meant for. For applications with an extensive and rich UI, the choice to run the UI in a browser has considerable advantages:
There is a multitude of UI frameworks for the browser, and if you want to roll your own, the standards of HTML and CSS allow you to fine-tune your UI and add visually appealing effects with little or no effort.

Delphi's VCL or Lazarus' LCL and derived components are hopelessly outdated where it concerns visual effects and gadgets, simply because the Javascript world is meanwhile much bigger and more people contribute to its development. You may ask: Do these visual effects really matter, and is it worth the effort? Each must decide this for himself, but the technology exists, and the FREE PASCAL & LAZARUS FOUNDATION wants to make it available for Object Pascal programmers.

With the development of PAS2JS and TMS Web core, it is possible to program the browser itself.
This means that we can now create a program that has a native part, programmed in Pascal, and a GUI part running in the browser - also programmed in pascal. This opens new perspectives.

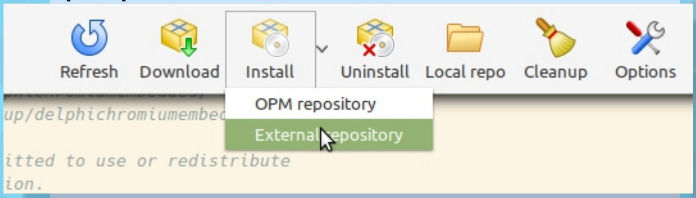


Figure 1: Install from external repository

So, you want to embed the browser in your application. How to start ?
The first thing to do is to download the Chromium Embedded Framework. You can find binaries for your platform on

```
https://cef-builds.spotifycdn.com/index.html
```

Second is to install **CEF4Delphi** in the Lazarus IDE. **CEF4Delphi** was initially developed as **DCEF3** by **Henri Gourvest**, and is continued by **Salvador Diaz Fau**.
The name suggests that the project was initially developed for Delphi and as such the support was best on Windows. There was preliminary support for Lazarus, and the **Free Pascal and Lazarus foundation** has sponsored some development by Martin Friebe to make it universally usable on all major Lazarus-supported platforms:
Windows, Linux and MacOS.

The **CEF4Delphi** project has a package file for installation in Lazarus (**CEF4Delphi_Lazarus**). The **Lazarus OnLine Package manager** can be used to install this package.
You will need to install the **DCPCrypt** package as well, as the **CEF4Delphi_Lazarus** package depends on this package.

Note that when installing through the **Online Package Manager,** you must install from the external repository, as shown in figure 1 on page 2.
The reason is that the changes needed to run in Lazarus are not yet packaged in the zip distributed by the people managing the Online Package manager.
Alternatively, you can also install directly from the **CEF4Delphi** sources.
Simply clone the official repository using your favourite git client:

```
https://github.com/salvadordf/CEF4Delphi
```

The Lazarus package is located in the packages directory. The package is a Lazarus package as any other and can be installed in the IDE with the usual **'Use → Install'** menu from the package dialog.
The Lazarus IDE will offer to rebuild itself, and if all goes well, 2 extra tabs will appear on the component palette, shown in Figure 2 on page 2/11 and Figure 3 on page2/11. (See below)
The following components are registered on the **'Chromium'** tab:

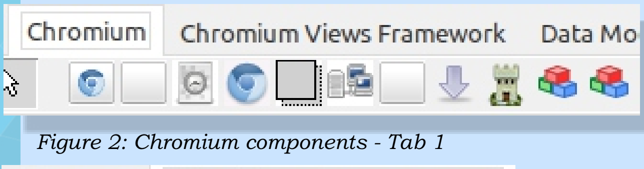


Figure 2: Chromium components - Tab 1

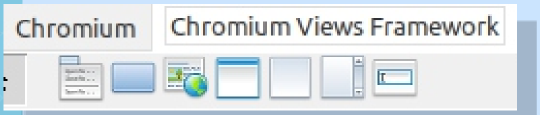


Figure 3: Chromium components - Tab 2

- **`TChromium`**
  The general interface to the Chromium Embedded Framework: It basically gives you access to the CEF API, and can be seen as the API for the browser.
- **`TBufferPanel`**
  A panel which can be used by the CEF browser to draw on, it provides double buffering (See the simple OSR (Off Screen Rendering) Browser demo)
- **`TCefServerComponent`**
  A small non-visual component that implements a small webserver and websocket server.
- **`TChromiumWindow`**
  A TWinControl descendent which can be used by the CEF browser to host its windows. Normally you will not use this, you should use `TBrowserWindow`
- **`TCEFWindowParent`**
  A TWinControl descendent which can be used by the CEF browser to host its windows. Normally you will not use this, you should use `TBrowserWindow`
- **`TCEFLinkedWindowParent`**
  A `TWinControl` descendent which can be used by the CEF browser to host its windows. Normally you will not use this, you should use `TBrowserWindow`
- **`TCEFURLRequestClientComponent`**
  a `TComponent` wrapper around a `TCustomCefUrlrequestClient`, used to handle request progress.
- **`TCEFWorkScheduler`**
  a component that can be used to control when the browser does work and when it is idle.
- **`TBrowserWindow`**
  A control that is essentially a combination of a `TCEFLinkedWindowParent` control and `TChromium` component. Usually this will be sufficient for most applications.
- **`TOSRBrowserWindow`**
  a control similar to `TBrowserWindow` that shows the browser window using **OSR (Off Screen Rendering).**
- **`TCEFSentinel`**
  a control that can be used to check the status of the CEF browser: You must be careful when to close the window, when the window is ready for you to send commands etc.

The components on the '**Chromium Views Framework**' tab are normally not necessary for embedding the browser. They are wrappers around **CEF** classes which - under normal circumstances - you will not need, but they are provided for completeness.
This might look like a lot of components just to implement a browser, but for simple cases, only a single component is needed: `TBrowserWindow`. This visual control has a `TChromium` component built-in and it uses this component to control the browser.

CREATING A SIMPLE BROWSER

To create a simple browser, not much code is needed. To show this, we create a small demo program. We drop an edit-control along the top border of the window `(edtAddress)`, and a `TBrowserWindow` control that occupies the rest of the window: `bwBrowser`.
Next to the address bar, we put a small button `btnGo`. In the `OnClick` handler of the button, we place the following code:

```
bwBrowser.LoadURL(UTF8Decode(edtAddress.Text));
```

The `UTF8Decode` is needed because the LCL uses `UTF8` for all texts in controls, but the **CEF API** expects a `WideString` (or Unicode String) in its APIs.
If this were all there was to it, it would of course be very simple.
Alas, some more work is needed.
The **Chromium browser** is a complex piece of software. To let it work correctly, some initialization is needed, and when closing down, also some extra code is needed to make sure the **CEF browser** close down properly.
Specifically, you can't just close the form on which the browser window is located. In the `OnCloseQuery` of the form, you must check whether the browser window was actually closed before allowing the form to be closed.

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  With bwBrowser do
    begin
      CloseBrowser(True);
      CanClose:=IsClosed;
    end;
end;
```

The first line tells the browser to close itself.
The second line checks if the browser window was actually closed, and if so, allows the form to be closed.
But what if the browser form was not yet closed ? How to close the form ? Well, the `TBrowserWindow` window as an `OnBrowserClosed` event that is triggered when the browser form is actually closed.
We can use this event to close the main form again:

```
procedure TForm1.bwBrowserBrowserClosed(Sender: TObject);
begin
  Close;
end;
```

The effect of these 2 event handlers is the following sequence of events:

❶ The user closes the main form.
❷ The `OnCloseQuery` event is triggered.
It tells the browser window to close.
If the window is not yet closed, the main form is prohibited from closing.
❸ When the browser window is closed, the `OnBrowserClosed` event is triggered.
It closes the main form.
❹ Again the `OnCloseQuery` event is triggered. But this time, the `IsClosed` will evaluate to `True`, and the main form is allowed to close.

Additionally, on Windows, some extra code is needed to handle the use of the menu. The menu uses a separate modal loop, and this must be communicated to the CEF browser.
The following code can be inserted in the form declaration:

```
{$IFDEF WINDOWS}
procedure WMEnterMenuLoop(var aMessage: TMessage); message WM_ENTERMENULOOP;
procedure WMExitMenuLoop(var aMessage: TMessage); message WM_EXITMENULOOP;
{$ENDIF}
```

And the methods must be implemented as follows:

```
uses
uCEFApplication;
{$IFDEF WINDOWS}
procedure TForm1.WMEnterMenuLoop(var aMessage: TMessage);
begin
inherited;
if (aMessage.wParam = 0) and (GlobalCEFApp <> nil) then
GlobalCEFApp.OsmodalLoop := True;
end;
procedure TForm1.WMExitMenuLoop(var aMessage: TMessage);
begin
inherited;
if (aMessage.wParam = 0) and (GlobalCEFApp <> nil) then
GlobalCEFApp.OsmodalLoop := False;
end;
{$ENDIF}
```

The `uCEFApplication` unit contains the `GlobalCEFApp` object instance which takes care of communication with the `CEF` library. Setting its `OsmodalLoop` property notifies the CEF library that a menu global event loop is active (or not).
The above code is not so difficult, and is the same for every application that uses CEF.
Alas, this is not yet everything you must do.

INITIALIZING THE CEF LIBRARY

The CEF library itself must also be initialized and stopped. Because the CEF browser runs in a separate process (this is a security measure), this takes time to set up and a lot of initialization needs to be done. Luckily, the CEF components take care of a lot of the details involved in setting up the library. In a Lazarus program, you don't need to code a lot to start the initialization process.
Unfortunately, this must be done at different points in time, depending on the OS you are using: on Linux, the initialization must happen before the LCL itself is initialized. On Windows and Mac, the LCL must be initialized before starting CEF.
The best way to do this is to put all code in a separate unit. We'll name it CEFControl:

```
unit CEFControl;
{$mode objfpc}

{$h+}

interface

uses
  uCEFApplication, uCEFWorkScheduler;
Procedure InitCEF;
Procedure StopCEF;

implementation

Procedure InitCEF;
var
  Dir : String;
begin
  if Assigned(GlobalCEFApp) then exit;
  {$IFDEF DARWIN}
  AddCrDelegate;
  {$ENDIF}
  CreateGlobalCEFApp;
  // Set the location of the CEF libs.
  {$IFDEF WINDOWS}
  Dir:='c:\CEF\Current\Dist':
  {$ELSE}
  Dir:='/opt/CEF/current/dist';
  {$ENDIF}
  GlobalCEFApp.FrameworkDirPath:=Dir;
  GlobalCEFApp.ResourcesDirPath:=Dir;
  GlobalCEFApp.LocalesDirPath:=Dir+PathDelim+'locales';
  if not GlobalCEFApp.StartMainProcess
  then
    begin
      StopCEF;
      halt(0); // exit the subprocess
    end;
end;

Procedure StopCEF;
begin
  if GlobalCEFWorkScheduler <> nil then
      GlobalCEFWorkScheduler.StopScheduler;
    DestroyGlobalCEFApp;
    DestroyGlobalCEFWorkScheduler;
end;

{$IFDEF LINUX}
initialization
  InitCEF;
{$ENDIF}
end.
```

On Linux, the InitCEF routine is called in the initialization section of the unit. If the CEFControl unit is inserted before the Interfaces unit in the program's `uses` clause, the result is that CEF will be initialized before the LCL is initialized.
To initialize CEF on MacOS andWindows, the InitCEF is called in the Initialization of the main form unit:

```
initialization
  InitCEF;
finalization
  StopCEF;
end.
```

Since the initialization section of the main form is only executed after the **LCL** was initialized,
the **CEF** library will be initialized only after the **LCL** is initialized, and **CEF** will be stopped before the **LCL** is finalized. Let’s see what happens in more detail during initialization:
The first line of the **InitCEF** routine checks if the global **CEF** application exists.
If it does, it exits: this ensures that even on **Linux,** the **CEF** library is initialized only once.
On **MacOS,** after this, an application delegate is created, this is a **MacOS-specific** routine, needed to handle communications with **CEF.**
The call to **CreateGlobalCEFApp** creates the actual **GlobalCEFApp** instance, and
does some configuration.

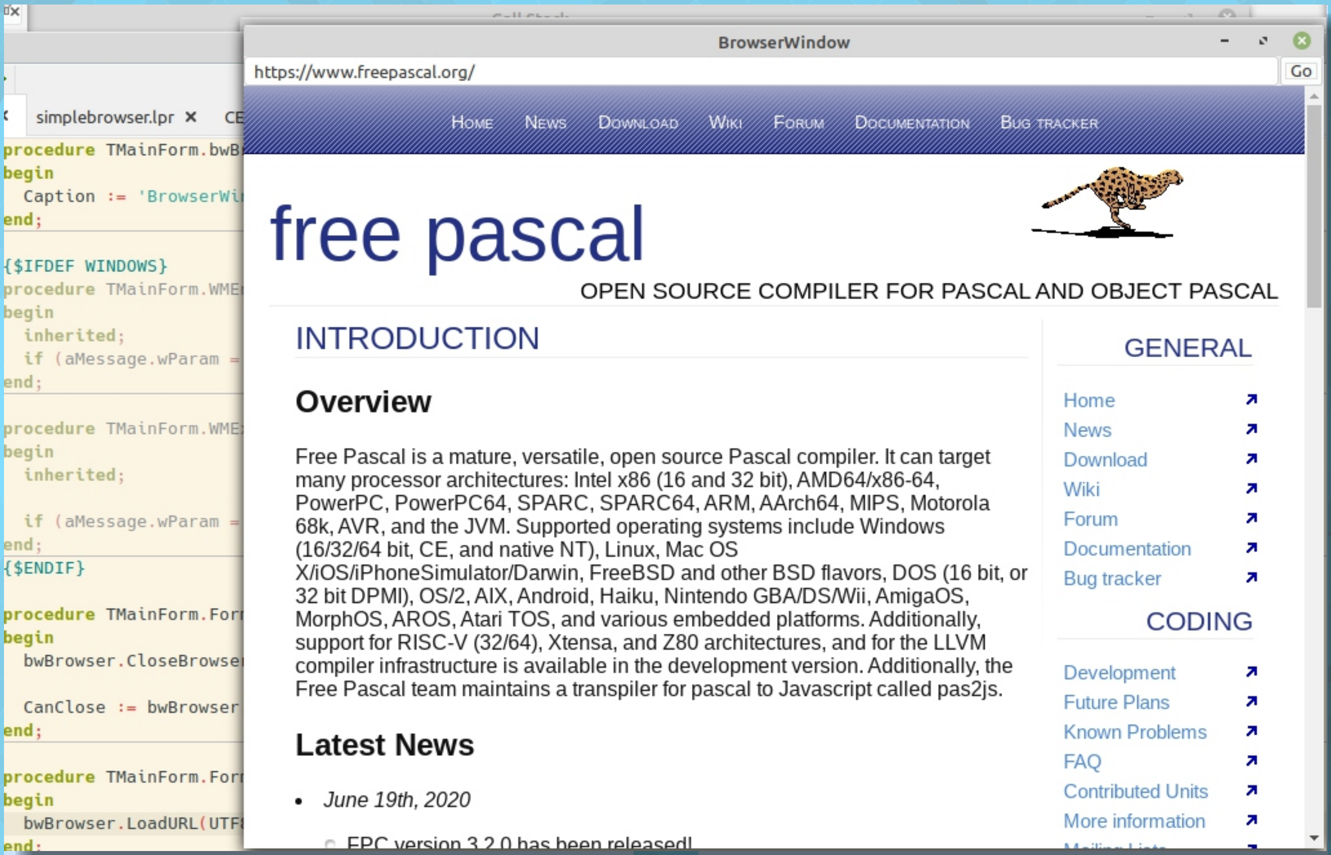
```
procedure PumpWork(const aDelayMS : int64);
begin
   if (GlobalCEFWorkScheduler <> nil) then
   GlobalCEFWorkScheduler.ScheduleMessagePumpWork(aDelayMS);
end;

procedure CreateGlobalCEFApp;
begin
   if GlobalCEFApp <> nil then  exit;
   GlobalCEFWorkScheduler := TCEFWorkScheduler.Create(nil);
   GlobalCEFApp:= TCefApplication.Create;
   GlobalCEFApp.ExternalMessagePump:= True;
   GlobalCEFApp.MultiThreadedMessageLoop:= False;
   GlobalCEFApp.OnScheduleMessagePumpWork:= @PumpWork;
   {$IFDEF LINUX}
   GlobalCEFApp.DisableZygote := True;
   {$ENDIF}
end;
```

The configuration is to handle messages; in the above case the **message pump** is handled in **PumpWork**. There are other ways to handle this, but it is outside the scope of this article to treat all possibilities: The above code can simply be used in most case.
After calling **CreateGlobalCEFApp**,
a couple of lines set some path properties of the global **GlobalCEFApp** instance.
This needs some explanation.
By default, **CEF4Delphi** looks for the **CEF** libraries and auxiliary files in the same directory as your application.

This means that you must copy the complete contents of the Resource and Release (*about 1.2 Gb*) to your application directory, and this for every time you create an application that uses CEF.
That’s of course not very convenient when you’re making various applications.
A better approach may be to copy the contents of these 2 directories to a single directory (*in the above example code this is CEF/current/dist*) and tell **CEF4Delphi** to use the libraries and support files in that directory.
That is what these lines in **InitCEF** do:

```
$IFDEF WINDOWS}
Dir:='c:\CEF\Current\Dist':
{$ELSE}
Dir:='/opt/CEF/current/dist';
{$ENDIF}
GlobalCEFApp.FrameworkDirPath:=Dir;
GlobalCEFApp.ResourcesDirPath:=Dir;
GlobalCEFApp.LocalesDirPath:=Dir+PathDelim+'locales';
```

Depending on the configuration this will start a second (sub) process. In the subprocess the return value of this function is False, in the main process the function returns True. The subprocess should stop at once, as actually a new program is started in the sub process.
That’s it.
Now the program is ready to go. Running the program will result in something like figure 4 on article page 6/11.



```
procedure TMainForm.bwB
begin
  Caption := 'BrowserWi
end;

{$IFDEF WINDOWS}
procedure TMainForm.WMEr
begin
  inherited;
  if (aMessage.wParam =
end;

procedure TMainForm.WMEx
begin
  inherited;

  if (aMessage.wParam =
end;
{$ENDIF}

procedure TMainForm.For
begin
  bwBrowser.CloseBrowse

  CanClose := bwBrowser
end;

procedure TMainForm.For
begin
  bwBrowser.LoadURL(UTF8
end;
```

free pascal

OPEN SOURCE COMPILER FOR PASCAL AND OBJECT PASCAL

INTRODUCTION

Overview

Free Pascal is a mature, versatile, open source Pascal compiler. It can target many processor architectures: Intel x86 (16 and 32 bit), AMD64/x86-64, PowerPC, PowerPC64, SPARC, SPARC64, ARM, AArch64, MIPS, Motorola 68k, AVR, and the JVM. Supported operating systems include Windows (16/32/64 bit, CE, and native NT), Linux, Mac OS X/iOS/iPhoneSimulator/Darwin, FreeBSD and other BSD flavors, DOS (16 bit, or 32 bit DPMI), OS/2, AIX, Android, Haiku, Nintendo GBA/DS/Wii, AmigaOS, MorphOS, AROS, Atari TOS, and various embedded platforms. Additionally, support for RISC-V (32/64), Xtensa, and Z80 architectures, and for the LLVM compiler infrastructure is available in the development version. Additionally, the Free Pascal team maintains a transpiler for pascal to Javascript called pas2js.

Latest News

- *June 19th, 2020*
  - FPC version 3.2.0 has been released!

Figure 4: Simple browser

Using these functions, it is easy to add 2 buttons to go back and forth in the browser history.
To demonstrate this, we add 2 buttons to the top panel, **`btnForward`** and **`btnBack`**.
Their **`OnClick`** handlers are quite simple:

```
procedure TMainForm.btnBackClick(Sender: TObject);
begin
  if bwbrowser.Chromium.CanGoBack then
   bwbrowser.Chromium.GoBack;
end;

procedure TMainForm.btnBackClick(Sender: TObject);
begin
  if bwbrowser.Chromium.CanGoBack then
   bwbrowser.Chromium.GoBack;
end;
```

The **`CanGoBack`** and **`CanGoForward`** functions can be used to create actions in an actionlist and let the buttons be enabled or disabled, depending on whether there is history to go forward or backward to. This is left as an exercise for the reader.

As developers, it is useful to be able to show the browser developer tools: this way we can inspect the HTML, CSS or debug the Javascript in the webpage. The chromium component also has a method to help with this: **`ShowDevtools`** and its counterpart **`HideDevTools`**. So, we create a menu button that shows a popup menu, and in the popup menu, we add menu items to show and hide the developer console.

The **`ShowDevtools`** call has 2 arguments: a point and a **`TWinControl`** instance.
The point is a coordinate in the browser window that will be inspected in the developer tools **HTML inspector**. The **`TWinControl`** instance is a **`TWinControl`** instance that will be replaced with the developer tools.
The latter can be **`Nil`**, in which case the developer tools are shown in a floating window.

We'll create a bottom-aligned panel, controller by a splitter, which will contain a second panel: the second panel will be replaced with the developer tools. Initially, the panel is invisible, and when the user asks to see the developer tools, we'll make the panel visible. When the user asks to close the developer tools, we ask the browser to close them and hide the panel.

```
procedure TMainForm.miHideDevToolsClick(Sender: TObject);
begin
   bwBrowser.Chromium.CloseDevTools(pnlDevToolsInner);
   splDevtools.Visible:=False;
   pnlDevtools.Visible:=False;
end;
procedure TMainForm.miSHowDevToolsClick(Sender: TObject);
begin
   splDevtools.Visible:=True;
   pnlDevtools.Visible:=True;
   pnlDevtools.Height:=ClientHeight div 3;
   bwBrowser.Chromium.ShowDevTools(Point(0,0),pnlDevToolsInner);
   bwBrowser.Chromium.OnDevToolsAgentDetached:=@DoDevtoolsDetached;
end;
```

The visibility of the panel can be used to enable/disable the appropriate menu items in our popup menu:

```
procedure TMainForm.pmMenuPopup(Sender: TObject);
begin
   miSHowDevTools.Enabled:=not pnlDevtools.Visible;
   miHideDevTools.Enabled:=pnlDevtools.Visible;
end;
```

EVENTS FROM THE BROWSER

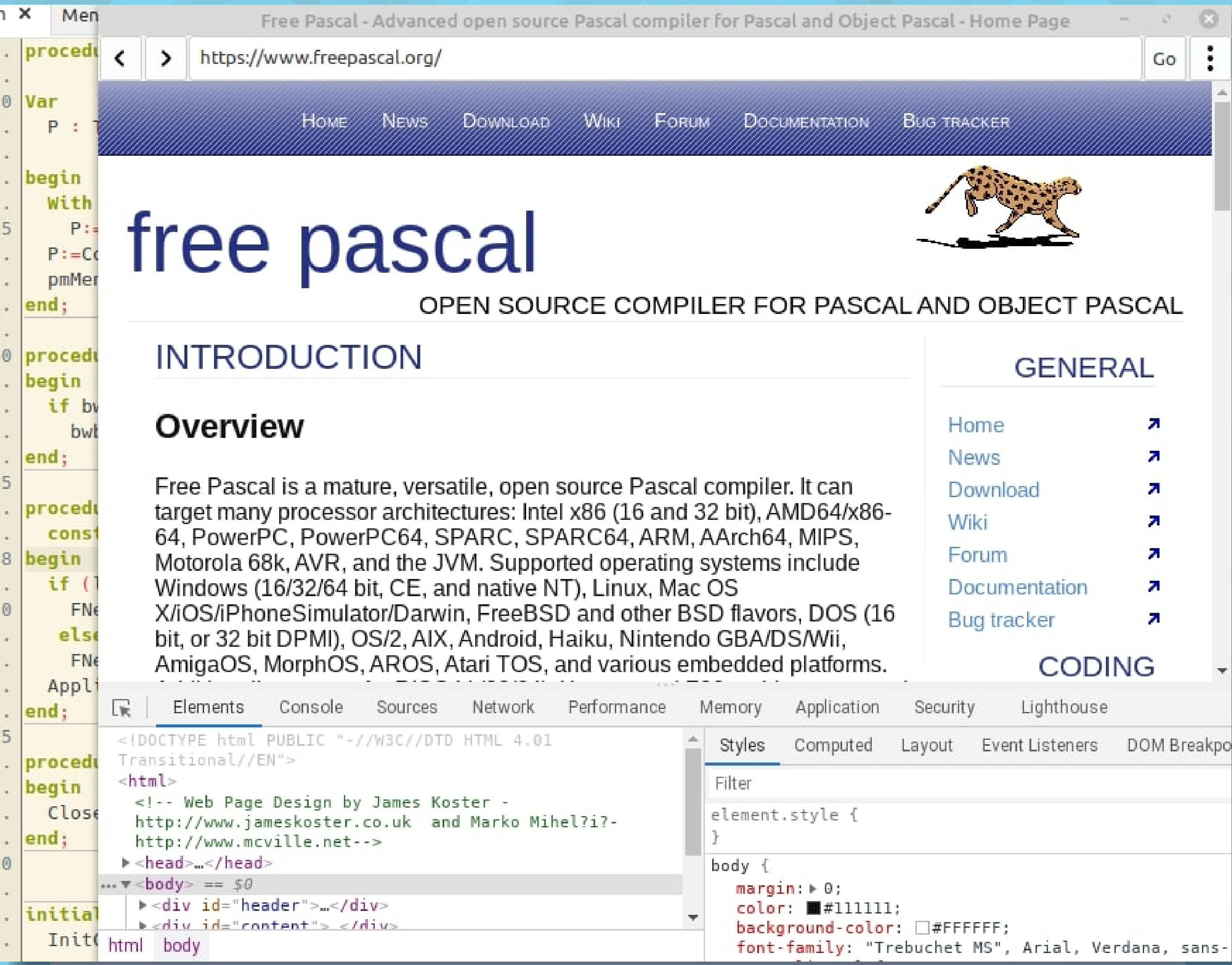
The **TChromium** component has a lot of event handlers. These are used by the CEF framework to notify your program of many things.
However, care must be taken when using these event handlers: The browser is running in a separate process, so all communication happens through the sending of messages, which are translated to events by CEF4Delphi. These events arrive in a thread which does not necessarily have to be the main UI thread. Since the widgetsets used by the LCL are not thread-safe this means that you cannot just update the GUI in the event handler. Luckily, Lazarus has a mechanism that can be used for this: **Appliction.QueueAsyncCall**. **QueueAsyncCall** schedules a callback, which will be called when the main application loop as finished processing pending messages. This callback is therefor called in the main application thread, and so it can be used to update the UI.
To demonstrate this, we'll use a simple event. The **OnTitleChange** event of the **TChromium** component notifies you when a HTML page has a Title tag in it, and it sends you the title. We can use this event to update the title bar of the form:

```
procedure TMainForm.ChromiumTitleChange(Sender: TObject;
const browser: ICefBrowser; const title: ustring);
begin
   if (length(title) > 0)
   then  FNewTitle := UTF8enCode(title)
   else
      FNewTitle:=UTF8enCode(bwBrowser.Chromium.DocumentURL);
   Application.QueueAsyncCall(@DoSetHTMLTitle,0);
end;
```

The new title is saved to a temporary variable, and an async call is scheduled: **DoSetHTMLTitle**:

```
procedure TMainForm.DoSetHTMLTitle(Data: PtrInt);
begin
   Caption:=FNewTitle;
end;
```

Since this call is executed in the main thread, it is safe to update the form caption. The result of this can be seen in figure 5 on page 11/11.



Men
Free Pascal - Advanced open source Pascal compiler for Pascal and Object Pascal - Home Page
https://www.freepascal.org/
Go
Home
News
Download
Wiki
Forum
Documentation
Bug tracker
free pascal

```
procedu
Var
  P : T

begin
  With
    P:=
  P:=Co
  pmMer
end;

procedu
begin
  if bw
    bwb
end;

procedu
  const
begin
  if (T
    FNe
   else
    FNe
  Appli
end;

procedu
begin
  Close
end;

initial
  Init(
```

OPEN SOURCE COMPILER FOR PASCAL AND OBJECT PASCAL

INTRODUCTION

Overview

Free Pascal is a mature, versatile, open source Pascal compiler. It can target many processor architectures: Intel x86 (16 and 32 bit), AMD64/x86-64, PowerPC, PowerPC64, SPARC, SPARC64, ARM, AArch64, MIPS, Motorola 68k, AVR, and the JVM. Supported operating systems include Windows (16/32/64 bit, CE, and native NT), Linux, Mac OS X/iOS/iPhoneSimulator/Darwin, FreeBSD and other BSD flavors, DOS (16 bit, or 32 bit DPMI), OS/2, AIX, Android, Haiku, Nintendo GBA/DS/Wii, AmigaOS, MorphOS, AROS, Atari TOS, and various embedded platforms.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN">
<html>
  <!-- Web Page Design by James Koster -
  http://www.jameskoster.co.uk  and Marko Mihel?i?-
  http://www.mcville.net-->
 ▸<head>…</head>
…▾<body> == $0
   ▸<div id="header">…</div>
   ▸<div id="content">…</div>
```

```
element.style {
}
body {
   margin: ▸ 0;
   color: ■#111111;
   background-color: □#FFFFFF;
   font-family: "Trebuchet MS", Arial, Verdana, sans-
```

USING YOUR OWN SCHEME TO SERVE LOCAL FILES

The browser normally uses the **'http(s)://'** URL scheme to access files on a webserver, if you double-click on a HTML file in the file explorer, the **'file://'** URL scheme.

If you want to create a GUI using a browser, and you want to distribute an application that can be only be used to load the files you want, you can register a custom URL scheme.

You can use this to serve a restricted list of files, only files from a ZIP file, or even not to distribute files, but have all files in resources embedded in the executable.

We'll demonstrate this by creating an application that plays the **Pacman** demo from the **pas2js demos**.
To do this is not very difficult. We'll use the first demo and extend it a little. When the **GlobalCEFApp** application is created in the **CEFControl** unit, we add a callback in which custom URL schemes can be registered:

```
GlobalCEFApp.OnRegCustomSchemes:=@CEFRegisterCustomSchemes;
```

This routine will be called when the browser is started, and is quite simple. When called, it gets a reference to a scheme registrar object , which can be used to register a scheme:

```
procedure CEFRegisterCustomSchemes(const registrar: TCefSchemeRegistrarRef);
Var  aOptions : TCefSchemeOptions;
begin
  aOptions:=CEF_SCHEME_OPTION_STANDARD or CEF_SCHEME_OPTION_LOCAL;
  registrar.AddCustomScheme('local', aOptions);
end;
```

We register here the URL Scheme name 'local'. The options mean that the scheme is a local one, which ensures that the browser will use the same safety mechanisms as when using the **file://** scheme.
The **TCefSchemeRegistrarRef** class is defined in unit **uCEFSchemeRegistrar**, so we must add that to the uses class of our **CEFControl** unit. Likewise we need the **uCEFTypes** unit for the **TCefSchemeOptions** and the **uCEFConstants** unit for the various constants. The above just tells the browser it can expect to see URLS starting with the **local://** scheme. It does not yet handle this scheme. To actually handle the 'local' scheme, we need to implement a descendent of the **TCefResourceHandlerOwn** class and register it. At least 3 methods of this class must be overridden:

```
TLocalResourceHandler = Class(TCefResourceHandlerOwn)
private
  FFileName : String;
  FStream : TStream;
Public
  function ProcessRequest(CEFRequest: ICefRequest;  callback: ICefCallback): Boolean; override;
  procedure GetResponseHeaders(CEFResponse: ICefResponse;
                               out responseLength: Int64;
                               out redirectUrl: ustring); override;
  function ReadResponse(const dataOut: Pointer; bytesToRead: Integer;
               var bytesRead: Integer;
               callback: ICefCallback): Boolean; override;
  Destructor destroy; override;
end;
```

To be able to add this definition, you need to add the **uCEFInterfaces**, and **uCEFResourceHandler** units to the **uses** clause.
This class must be registered with the browser, the **CefRegisterSchemeHandlerFactory** function from the **uCEFMiscFunctions** unit can be used for this:

```
CefRegisterSchemeHandlerFactory('local',",TLocalResourceHandler);
```

Now, when the browser needs to handle a URL scheme of 'local', it will create an instance of TLocalResourceHandler, and call **ProcessRequest**, **GetResponseHeaders** and **ReadResponse**, in that order.

The first method to be called is **ProcessRequest**:

```
function TLocalResourceHandler.ProcessRequest(const CEFRequest: ICefRequest;
  const callback: ICefCallback): Boolean;
Var
  S : String;
  P : Integer;
begin
  S:=CEFRequest.GetUrl;
  P:=Pos('://',S);
  if P>0 then
    Delete(S,1,P+2);
  FFileName:=SysUtils.SetDirSeparators(ExtractFilePath(ParamStr(0))+S);
  Result:=FileExists(FFileName);
  if Result then
    FStream:=TFileStream.Create(FFileName,fmOpenRead or fmShareDenyWrite);
  if assigned(CallBack) then
    Callback.Cont;
end;
```

For this example we just check if the file exists, and if it does, we return True, otherwise we return false. For a real application, you would probably check that the URL does not contain any strange or forbidden locations, or is restricted to a list of files. When the file exists and can be served, we create a file stream (it is destroyed in the destructor).
The browser passes a callback, which you must call in order to indicate that the process can continue.
After this call, the **GetResponseHeaders** is called. This method should simulate the headers of a HTTP Response, and return the size of the data:

```
procedure TLocalResourceHandler.GetResponseHeaders(const CEFResponse: ICefResponse;
  out responseLength: Int64; out  redirectUrl: ustring);
Var
  Ext : String;
begin
  Ext:=ExtractFileExt(FFileName);
  Ext:=Copy(Ext,2,Length(Ext)-1);
  if Assigned(FStream)
  then
    begin
      CefResponse.Status:=200;
      CefResponse.StatusText:='OK';
      responseLength:=FStream.Size;
    end
  else
    begin
      CefResponse.Status:=400;
      CefResponse.StatusText:='NOT FOUND';
      responseLength:=0;
    end;
    CefResponse.MimeType:=CefGetMimeType(Ext);
    // No other headers.
end;
```

Normally, you would also set other HTTP headers of the request, but for simplicity we've omitted that part in our implementation.
After these 2 calls, if there is data to be read, the browser will call **ReadResponse** to actually read the data. This is quite simple, we just read the data from the stream we created in **ProcessRequest**, taking care we do not read too much data:

```
function TLocalResourceHandler.ReadResponse(
   const dataOut: Pointer; bytesToRead: Integer;
   var bytesRead: Integer; const callback: ICefCallback
   ): Boolean;
Var
   aAvail : Integer;
begin
   Result:=Assigned(FStream) and Assigned(DataOut);
   if Result then
    begin
      aAvail:=FStream.Size-FStream.Position;
      If aAvail>BytesToRead then
        aAvail:=BytesToRead;
        BytesRead:=FStream.Read(DataOut^,aAvail);
      Result:=aAvail>0;
    end;
end;
```

Note that the function must return **`True`** if data was read, and **`False`** if there is no more data, or the browser will keep trying to read data. (*If the size of the data is not known in advance, this can happen*).
That's it. Our application is now ready to use the **`'local://'`** scheme.
We remove the address bar, the go button, and simply load the HTML file we wish to show in the browser with the local scheme:

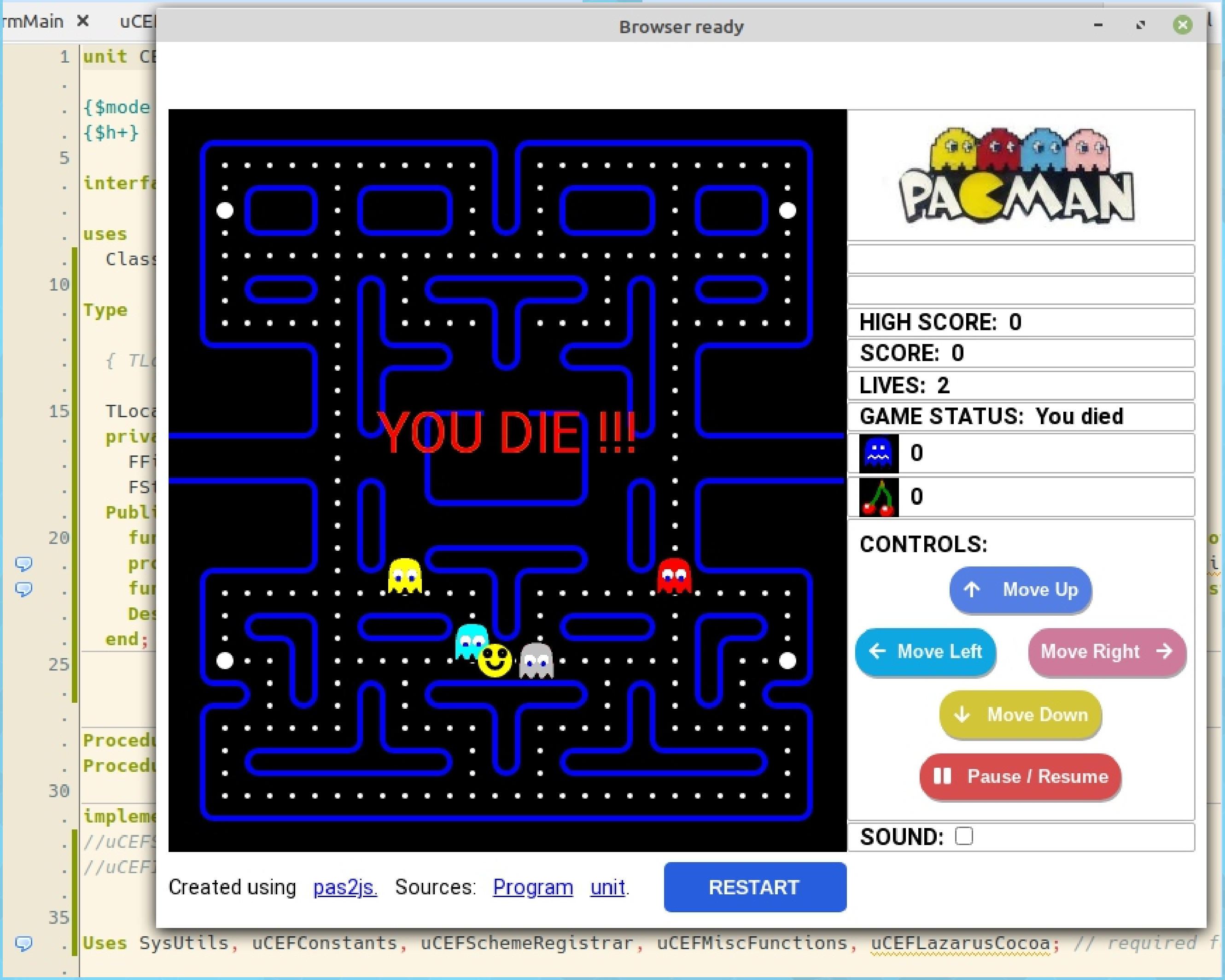
```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  CefRegisterSchemeHandlerFactory('local','',TLocalResourceHandler);
  bwBrowser.LoadURL('local://files/index.html');
end;
```

When using the **pacman** example from **pas2js**, this will look as in figure 6 on page 75

CONCLUSION

In this article, we showed how to embed a browser in your Lazarus application: due to recent improvements in the CEF4Delphi framework, this has been made really easy.
But the article has scratched only the surface of what is possible. In a next article, we'll show how to interact with the host application from within the browser application.

Figure 6: PacMan example



```
unit CE
{$mode
{$h+}
interfa
uses
  Class
Type
  { TL
  TLoca
  priva
    FF
    FS
  Publi
    fu
    pr
    fu
    Des
  end;
Procedu
Procedu
impleme
//uCEFS
//uCEF
Uses SysUtils, uCEFConstants, uCEFSchemeRegistrar, uCEFMiscFunctions, uCEFLazarusCocoa; // required f
```

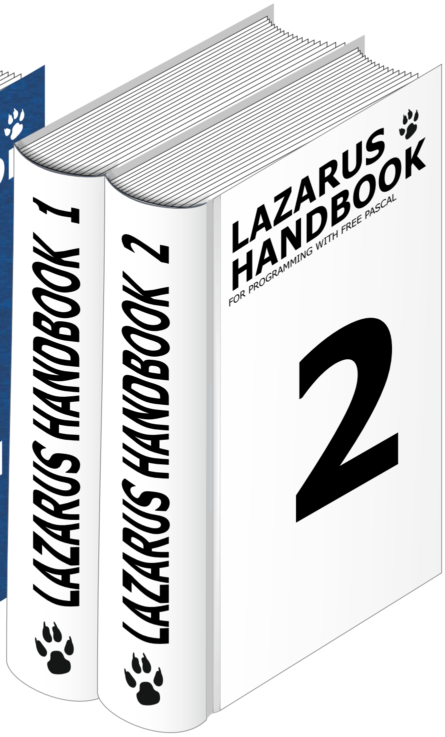
Browser ready

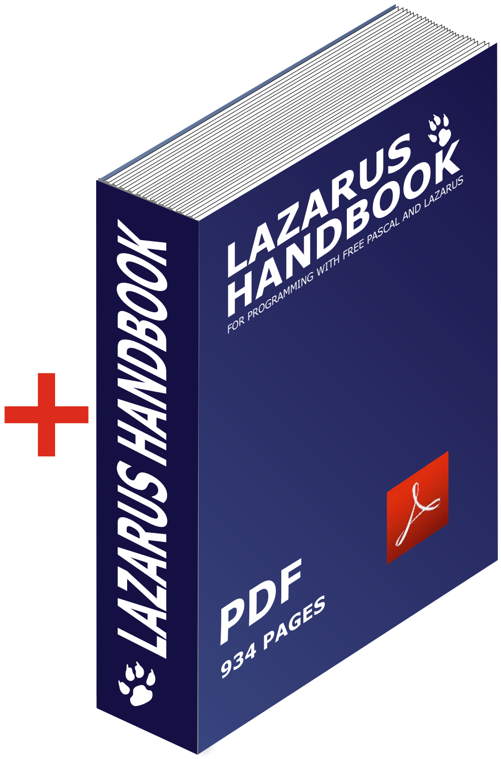
HIGH SCORE: 0
SCORE: 0
LIVES: 2
GAME STATUS: You died
0
0

Created using pas2js. Sources: Program unit.

LAZARUS HANDBOOK 2
LAZARUS HANDBOOK 1
LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL AND LAZARUS
1
+
LAZARUS HANDBOOK
LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL AND LAZARUS
PDF
934 PAGES

LAZARUS HANDBOOK 1
LAZARUS HANDBOOK 2
LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL
2


LAZARUS HANDBOOK 1
LAZARUS HANDBOOK 2
LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL
2


+
LAZARUS HANDBOOK
LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL AND LAZARUS
PDF
934 PAGES

LAZARUS HANDBOOK
FOR PROGRAMMING WITH FREE PASCAL AND LAZARUS
1
BLAISE PASCAL MAGAZINE
LAZARUS HANDBOOK 1
LAZARUS HANDBOOK 2

CODE SNIPPETS Part 10 Printing With Lazarus   Page 1/5

By Detlef Overbeek , project by Michael van Canneyt

starter   expert

INTRODUCTION

Printing is closely related to graphics and drawing: when you want to print something, you must draw it on a canvas – in this case a printer page. Printing is implemented in the **`Printer4Lazarus`** package. (There is a similar-sounding package called **`Printers4lazide`**, which enables a Print Menu in the IDE itself, which is not necessary for regular printing support in a LCL application).

THE TPRINTER OBJECT

The canvas on which to draw the pages you want to print, can be obtained from the Printer instance which has a Canvas property. There is always a global Printer object of type TPrinter available, if Printers is included in your form's uses clause. The TPrinter class encapsulates printer functionality for the LCL. TPrinter has the following public methods:

Method	Purpose
`Abort`	Stop the current printing job, discarding all pages.
`BeginDoc`	Start a new printing job.
`NewPage`	Starts a new page in the current print job, saving the current page and increasing PageNumber.
`EndDoc`	Ends the printing job, and sends the job to the printer.
`Refresh`	Refreshes the list of printers and fonts.
`SetPrinter`	Set the current printer.
`RestoreDefaultBin`	Restore the default bin on the printer.
`Write`	Write something directly to the printer (to be used only in raw printing mode).

To print something means using the following methods in a simple loop:

```
Printer.BeginDoc;
While not DonePrinting do
begin
// Draw the current page, setting DonePrinting if the last page is now drawn
If not DonePrinting do
  NewPage;
end;
Printer.EndDoc;
```

If something goes wrong (e.g. the user presses a Cancel button during the printing process) then

`Printer.Abort;`

stops the printing process.

Property	Type	Purpose
Aborted	Boolean	Indicates whether the current print job was aborted or not.
BinName	string	Name of the selected paper bin on the printer.
CanPrint	Boolean	Indicates whether the printer is ready to print.
CanRenderCopies	Boolean	Indicates whether the printer can make multiple copies of a document by itself.
Canvas	TCanvas	The page canvas.
CanvasClass	TPrinterCanvasRef	The TCanvas class to use for Canvas
Copies	Integer	Number of copies to print (only observed if CanRenderCopies is True)
DefaultBinName	string	Name of the default paper bin.
FileName	string	The file name to which the file will be printed.
Fonts	TStrings	A list of font names available for printing.
Orientation	TPrinterOrientation	Page orientation.
PageHeight	Integer	Printable page height in dots.
PageNumber	Integer	Current page number.
PageWidth	Integer	Printable page width in dots.
PaperSize	TPaperSize	Current paper size.
PrinterIndex	Integer	Index of currently selected printer in the list of printers.
PrinterName	string	Currently selected printer.
Printers	TStrings	A list of the names of available printers on the system.
PrinterState	TPrinterState	Current printer state.
PrinterType	TPrinterType	Current printer type.
Printing	Boolean	Is the printer printing?
RawMode	Boolean	Is the printer in RAW mode (usable for matrix printers etc.)?
SupportedBins	TStrings	Paper bins supported by this printer.
Title	string	Title of the current print job.
XDPI	Integer	Horizontal resolution of the selected printer.
YDPI	Integer	Vertical resolution of the selected printer.

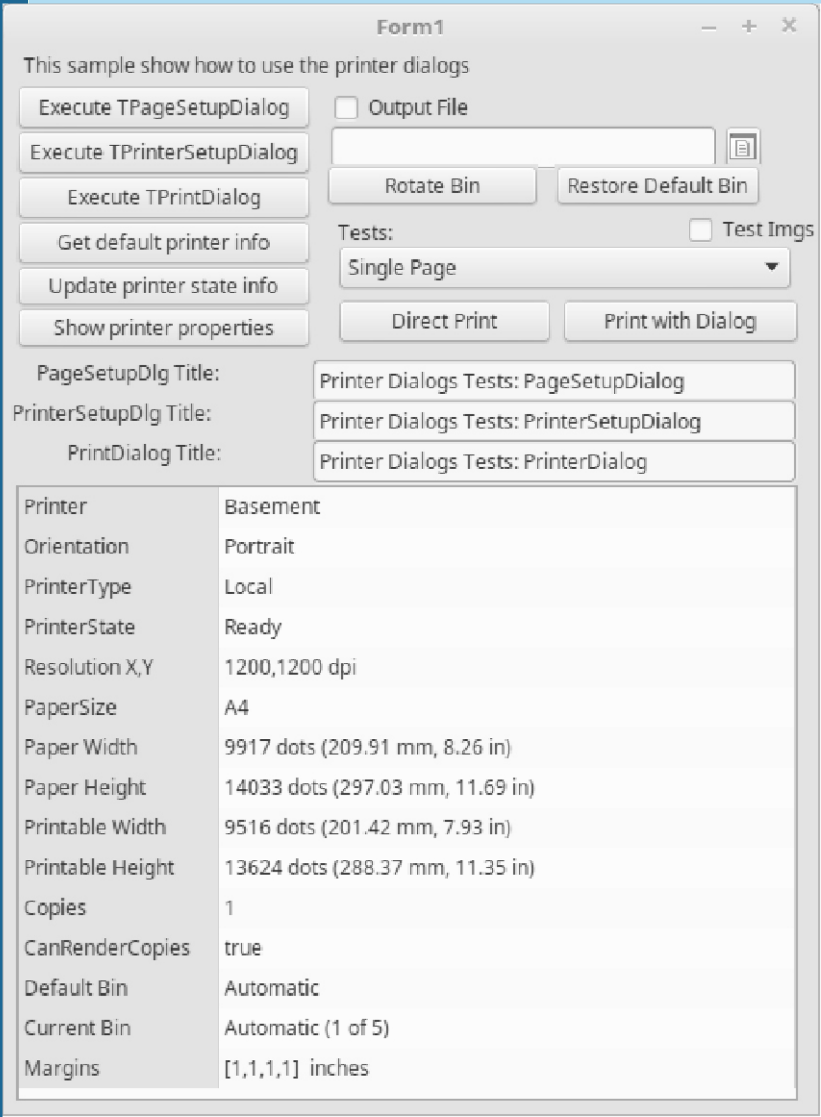
The page Orientation (of type **PrinterOrientation)** can take one of four self-descriptive values: **poPortrait, poLandscape, poReverseLandscape, poReversePortrait.**
The **PrinterType** property (of type **TPrinterType)** can take one of two self-descriptive values: **ptLocal** or **ptNetWork.**
The **PrinterState** property can have one of the **TPrinterState** enumeration values: **psNoDefine, psReady, psPrinting, psStopped.** These enumerations are self-explanatory, except **psNoDefine** which here means that printer state information is not available.
The TPaperSize class contains information about the currently selected printer paper size.
TPaperSize has the following properties:

Property	Type	Purpose
DefaultPaperName	String	The name of default paper size.
DefaultPapers	Boolean	Is this class using a default list of predefined papers?
Height	Integer	The paper height in dots.
PaperName	String	The name of the currently selected paper.
PaperRect	TPaperRect	A structure describing the (printable and actual) paper rectangles.
PaperRectOf	Array	An array of paper rectangles, indexed by paper name.
SupportedPapers	TStrings	A list of supported (or predefined) paper sizes.
Width	Integer	The paper width in dots.

The LCL provides several dialogs to manipulate these properties:

Class	Purpose
TPrintDialog	A printer selection dialog. This modifies the **Printer** instance, applying the properties of the selected printer.
TPrinterSetupDialog	A printer properties dialog, which modifies the **Printer** instance with the selected properties.
TPageSetupDialog	A page set-up dialog, which shows settings for printing a page, such as margins etc. This dialog's properties are not applied to the **Printer** object, but must be read and used via code.

The Lazarus SelectPrinter example demonstrates the use of these dialogs and how they affect the Printer object. A screenshot of this demo is shown below.



This sample show how to use the printer dialogs

□ Output File

Tests: Single Page □ Test Imgs

PageSetupDlg Title: Printer Dialogs Tests: PageSetupDialog

PrinterSetupDlg Title: Printer Dialogs Tests: PrinterSetupDialog

PrintDialog Title: Printer Dialogs Tests: PrinterDialog

Printer	Basement
Orientation	Portrait
PrinterType	Local
PrinterState	Ready
Resolution X,Y	1200,1200 dpi
PaperSize	A4
Paper Width	9917 dots (209.91 mm, 8.26 in)
Paper Height	14033 dots (297.03 mm, 11.69 in)
Printable Width	9516 dots (201.42 mm, 7.93 in)
Printable Height	13624 dots (288.37 mm, 11.35 in)
Copies	1
CanRenderCopies	true
Default Bin	Automatic
Current Bin	Automatic (1 of 5)
Margins	[1,1,1,1] inches

Figure 1: Win32 Check

It is important to note that the coordinate system of the printer canvas is in dots,
and that the actual size of all things printed or drawn will depend on the resolution of the printer. Also note that the actual printable area is often smaller than the whole page. Many laser, inkjet and dot-matrix printers cannot print across the entire paper surface. The actually available page width and height are therefore important, and the coordinate system of the page starts at the origin of the printable area.

The following short program demonstrates both the Printer Set-Up Dialogs, and how the printable area applies through printing the content of a memo on one or more pages. The program draws a border around the text, based on the margins set in the Page Set-Up Dialogs.
The coordinates of the border rectangle for the printer canvas coordinate system (taking into account the origin and usable print area of the page), are calculated as follows, where the WorkRect property holds critical information obtained from the selected printer:

CODE SNIPPETS Part 10 Printing With Lazarus   Page 1/5

```
procedure TMainForm.GetPageRect(Out PageRect : TRect);

  function ToPixels(aUnit : Double; ares : Integer) : Integer;
  begin
    if PSDemo.Units in [pmDefault,pmMillimeters] then
      Result:=MMtoPixel(aUnit/100,Ares)
    else
      Result:=InchToPixel(aUnit/100,Ares)
  end;

  var
    LM,RM,TM,BM,H,W : Integer;
    WR : TRect;
  begin
    LM:=ToPixels(PSDemo.MarginLeft,Printer.XDPI);
    RM:=ToPixels(PSDemo.MarginRight,Printer.XDPI);
    TM:=ToPixels(PSDemo.MarginTop,Printer.YDPI);
    BM:=ToPixels(PSDemo.MarginBottom,Printer.YDPI);
  // Work rect
    WR:=Printer.PaperSize.PaperRect.WorkRect;
    W:=Printer.PaperSize.Width-WR.Left;
    H:=Printer.PaperSize.Height-WR.Top;
    PageRect:=Rect(LM-WR.Left,TM-WR.Top, W-RM,H-BM);
  end;
```

The units for the printer set-up dialog margins are in 1/100 of a unit. They are converted to pixels (dots) by the following transformation functions:

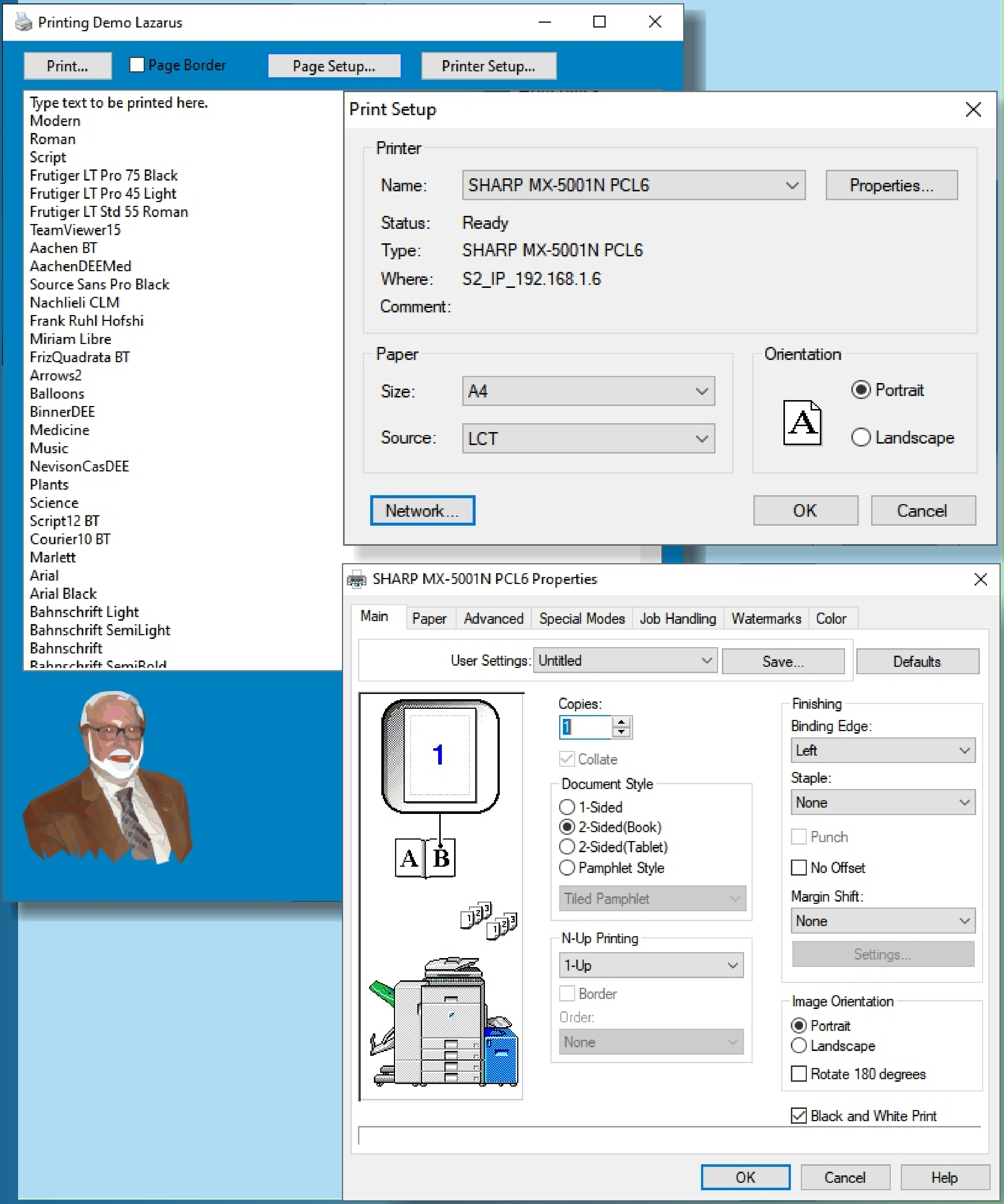
```
Function MMToPixel(aMM : Double; aRes : integer) : Integer;
begin
  Result:=Round((aMM/25.4) * aRes)
end;

Function InchToPixel(aInch : Double; aRes : integer) : Integer;
begin
  Result:=Round(aInch * aRes);
end;
```

The results of these calculations obviously depend on the X and Y resolution of the printer.

With this, the contents of a memo can be printed as follows:

```
procedure TMainForm.PrintMemo(DrawBorder : Boolean);
Var  PageRect : Trect;  I,D,T,L,Y : Integer;
begin  // Global options
  Printer.BeginDoc;
  Printer.Title:='Print demo page';
  Printer.PaperSize.PaperName:='A4';
   with Printer.Canvas do  begin  // How to draw and what font to use
     Pen.Color:=clBlack;
     Pen.Style:=psSolid;
     Brush.Style:=bsSolid;
     Brush.Color:=clNone;
     Font.Name:= 'Dejavu Sans';
     Font.Size:=10;
      // Get work area
     GetPageRect(PageRect);
     if DrawBorder then
     Rectangle(PageRect);
     // Prepare for the text printing loop
     L:=PageRect.Left+MMToPixel(5,Printer.XDPI);
     T:=PageRect.Left+MMToPixel(5,Printer.YDPI);
     D:=Round(TextHeight('M')*1.2);
     I:=0;
     Y:=T;
     While (I<MText.Lines.Count) do  begin
       TextOut(L,Y,MText.Lines[i]);
       Y:=Y+D;
       Inc(I);
       // Check whether the text will overflow the workable area, and start new page
       If (Y>=PageRect.Bottom-(D div 2)) and (I<MText.Lines.Count) then
         begin
           Printer.NewPage;
           Y:=T;
           if DrawBorder then  Rectangle(PageRect);
         end;
       end;
   end;
  Printer.EndDoc;  // Send everything to the printer
 end;
```

The running program and look like this:



Printing Demo Lazarus

Print... □ Page Border Page Setup... Printer Setup...

```
Type text to be printed here.
Modern
Roman
Script
Frutiger LT Pro 75 Black
Frutiger LT Pro 45 Light
Frutiger LT Std 55 Roman
TeamViewer15
Aachen BT
AachenDEEMed
Source Sans Pro Black
Nachlieli CLM
Frank Ruhl Hofshi
Miriam Libre
FrizQuadrata BT
Arrows2
Balloons
BinnerDEE
Medicine
Music
NevisonCasDEE
Plants
Science
Script12 BT
Courier10 BT
Marlett
Arial
Arial Black
Bahnschrift Light
Bahnschrift SemiLight
Bahnschrift
Bahnschrift SemiBold
```

Print Setup

Printer

Name: SHARP MX-5001N PCL6 — Properties...

Status: Ready

Type: SHARP MX-5001N PCL6

Where: S2_IP_192.168.1.6

Comment:

Paper

Size: A4

Source: LCT

Orientation

● Portrait

○ Landscape

Network... OK Cancel

SHARP MX-5001N PCL6 Properties

Main | Paper | Advanced | Special Modes | Job Handling | Watermarks | Color

User Settings: Untitled — Save... Defaults

Copies: 1

☑ Collate

Document Style

○ 1-Sided

● 2-Sided(Book)

○ 2-Sided(Tablet)

○ Pamphlet Style

Tiled Pamphlet

N-Up Printing: 1-Up

□ Border

Order: None

Finishing

Binding Edge: Left

Staple: None

□ Punch

□ No Offset

Margin Shift: None

Settings...

Image Orientation

● Portrait

○ Landscape

□ Rotate 180 degrees

☑ Black and White Print

OK Cancel Help

WEB SERVICE PART 2 - STORAGE

By Danny Wind

starter expert

This series of articles is about writing your own web services server and client in Delphi. The approach of all articles is pragmatic. The first article introduced some of the concepts you need to know and shows you how to create and consume your own web service in Delphi with just the GET request. This second article shows you how to update the data in the web service and how to create in-memory storage for the web service.

In the previous article we only used the HTTP GET request to return data from our web service. This time we will add the other three HTTP commands to our web service.

HTTP GET idempotent, cacheable	Retrieves data from the resource
usage in our web service	**SELECT** (get existing record, disallow caching so we get new data each time)
HTTP POST not idempotent, not cacheable/stale	Appends data to the existing resource
usage in our webservice	**UPDATE** existing (partial update of fields in a record, not updating the primary key)
HTTP PUT idempotent, not cacheable/stale	Replace the existing resource or inserts data as a new resource
usage in our web service	**INSERT** new (or **REPLACE**) (insert new record with new primary key, or replace entire record)
HTTP DELETE idempotent, not cacheable/stale	Deletes the resource
usage in our web service	**DELETE** (delete existing record or return error if it doesn't exist)

Before we do that however we need to know what the idempotent and cacheable properties of the HTTP commands mean for our web service.
Idempotency means that after the first request a second (or third and so on) request of an idempotent method should yield the same effect, unless there is an error or it has expired.

The cacheable property means that a requester is allowed to cache a request. So instead of sending a repeated request to the server again, it can just return a cached result. Both idempotency and cacheable combined in the GET request means that a web client that consumes your web service could be caching requests. It's easy to notice when you run into this.

If you get the same result from a GET request, even if the data in the server has changed, then your web client is caching. Taking a look at network traffic helps as well. If the web client only generates network traffic on the first net request, you know what's happening. It's easy to prevent this type of caching behaviour, by setting the Cache-Control and Expires elements in the HTTP header. Remember this if you're not getting the new data that you want to get from your GET.

IDEMPOTENT METHODS (RFC definition)
Methods can also have the property of "idempotence" in that (aside from error or expiration issues) the side-effects of N > 0 identical requests is the same as for a single request. The methods GET, HEAD, PUT and DELETE share this property. Also, the methods OPTIONS and TRACE SHOULD NOT have side effects, and so are inherently idempotent.

However, it is possible that a sequence of several requests is non- idempotent, even if all of the methods executed in that sequence are idempotent. (A sequence is idempotent if a single execution of the entire sequence always yields a result that is not changed by a reexecution of all, or part, of that sequence.) For example, a sequence is non-idempotent if its result depends on a value that is later modified in the same sequence.
A sequence that never has side effects is idempotent, by definition (provided that no concurrent operations are being executed on the same set of resources).

Looking at PUT, you see that PUT is considered idempotent and not-cacheable. So the same PUT request, when repeated, should yield the same result. However PUT is not considered cacheable, so if you PUT a resource, then DELETE that resource from another location and PUT it again it will result in the new resource. The second PUT is not cached on the client-side, it's always considered stale and thus sent to the server. In short, if you PUT something twice it should always successfully replace the existing resource. Your PUTs won't disappear.

For POST the defined behaviour is a bit different, as POST is not idempotent. So if you want to update a resource it could work the first time, but if someone else uses DELETE on that resource a subsequent POST (append) to the same resource could yield an error or just fail.

HTTP Commands
A good thing to know is that the definition of the HTTP commands in RFC allows for multiple usage scenarios of each command.
Because there is some leeway between definition and interpretation of its usage for PUT and POST in web services it's perfectly valid for us to create a web service that uses PUT as an equivalent for INSERT or REPLACE and POST as an equivalent for an UPDATE.

There is an interpretation that wants you to use POST to get data, when the GET request manipulates the data on the server. In our web service we return a random number with a GET request, and this interpretation would suggest using POST instead, as a POST is not idempotent and is allowed to change the server state.

Nevertheless in this article we will continue using GET with Cache-Control and Expires elements in the returned HTTP header to prevent caching.

Before we start coding and add PUT, POST and DELETE methods to our web service, let's first expand our toolkit and introduce the REST Debugger. This is a handy tool that comes bundled with the Delphi IDE. You can use it to test and debug your web service. You can find the REST debugger in the Delphi IDE under Tools ➔ REST Debugger.

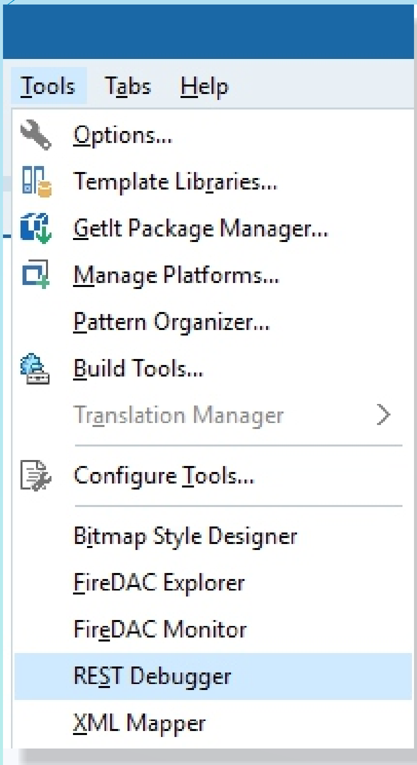
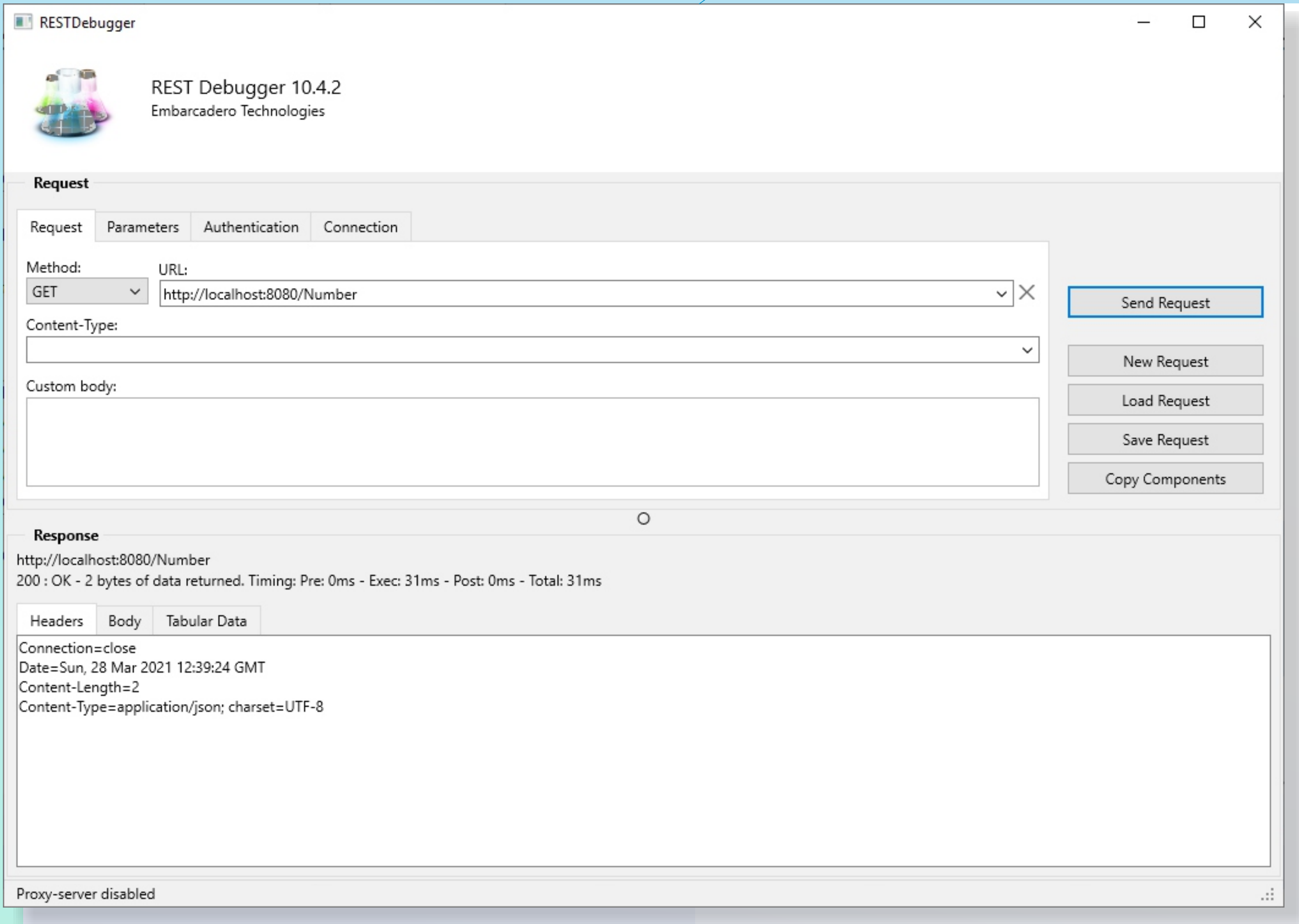


Figure 1: Tools ➔ REST Debugger

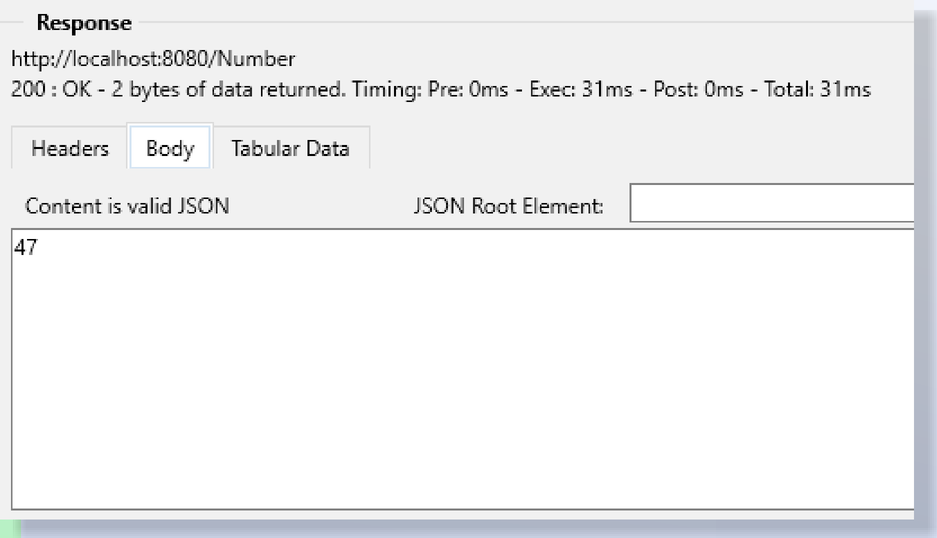
Let's run the REST Debugger and use it to test the web service we created in the previous article. We can use the GET request for Number.
`http://localhost:8080/Number`



```
http://localhost:8080/Number
200 : OK - 2 bytes of data returned. Timing: Pre: 0ms - Exec: 31ms - Post: 0ms - Total: 31ms
```

```
Connection=close
Date=Sun, 28 Mar 2021 12:39:24 GMT
Content-Length=2
Content-Type=application/json; charset=UTF-8
```

Figure 2:
In the REST Debugger we can see that the returned `Content-Type = application/json` and if we open the tab **Body** we see that the returned JSON is valid.



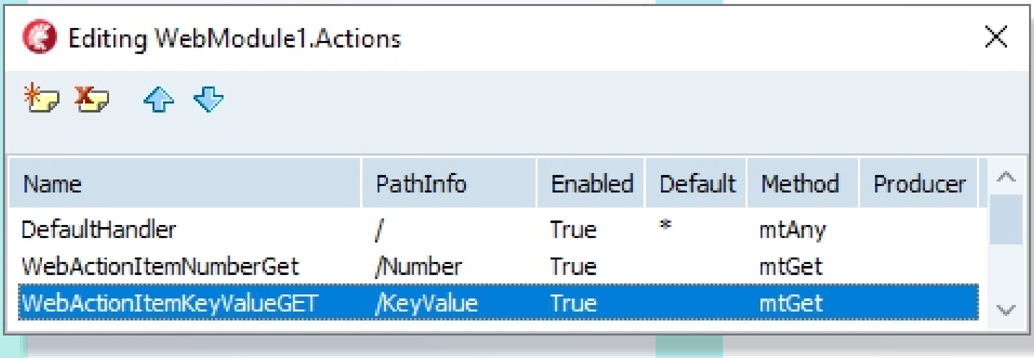
```
http://localhost:8080/Number
200 : OK - 2 bytes of data returned. Timing: Pre: 0ms - Exec: 31ms - Post: 0ms - Total: 31ms
Content is valid JSON
47
```

Figure3: Result

We are now ready to add some new methods to our web service.

OPEN THE WEB SERVER SERVICE (`WebServiceServerWithGUI`) we created in the previous article in Delphi. It's compatible with either Delphi 10 Community, Delphi 10 Professional, Enterprise or up.

❶ In the WebModuleUnit edit the Actions property and add a new handler with name WebActionItemKeyValueGET, MethodType mtGet and pathinfo /KeyValue. *Figure 4:*



Editing WebModule1.Actions

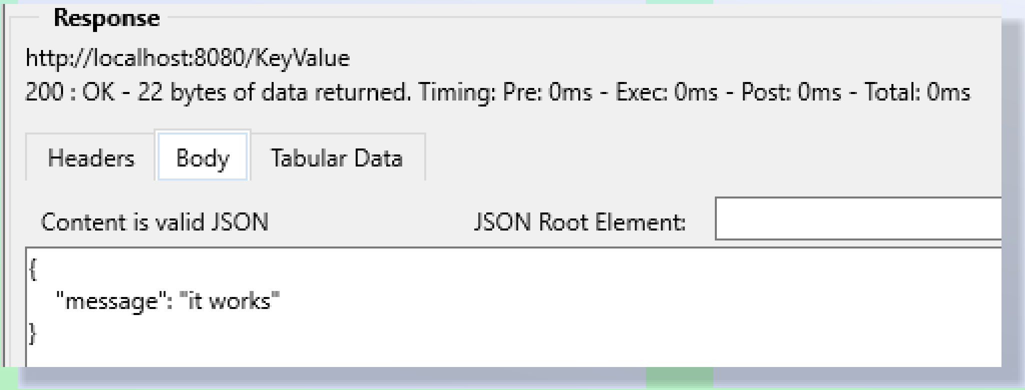
Name	PathInfo	Enabled	Default	Method	Producer
DefaultHandler	/	True	*	mtAny	
WebActionItemNumberGet	/Number	True		mtGet	
WebActionItemKeyValueGET	/KeyValue	True		mtGet	

❷ In the OnAction event-handler for this new item code a test response in the format of a JSON string.

```
procedure TWebModule1.WebModule1WebActionItemKeyValueGETAction(
  Sender: TObject; Request: TWebRequest; Response: TWebResponse;
  var Handled: Boolean);
begin
  Response.ContentType := 'application/json; charset=UTF-8';
  Response.Content := '{"message":"it works"}';
end;
```

❸ Run the web service again, click the Start button and start the REST Debugger and test if your web service still works and a GET request of this URL yields the expected valid JSON result.

`http://localhost:8080/KeyValue`



Response

http://localhost:8080/KeyValue

200 : OK - 22 bytes of data returned. Timing: Pre: 0ms - Exec: 0ms - Post: 0ms - Total: 0ms

```
{
  "message": "it works"
}
```

Figure 5: Response

The next step is creating an actual Key Value store in-memory to hold the data for this web service. We will be using a generic TDictionary to store the Key Value pairs.
To safely and successfully use this in-memory Key Value store we need to know how the WebModule handles incoming requests. A Web Broker application has only one WebModule class variable as you can see in the interface section of the WebModuleUnit

```
var
  WebModuleClass: TComponentClass = TWebModule1;
```

However for each request a new WebModule instance may be instantiated and each request is handled in its own thread. For us this means we will need to serialize access to our in-memory Key Value store to make it thread safe. Also the Key Value store will be created as a global variable to make it accessible to all WebModule instances.

❹ Declare the gLock and gKeyValueStore variables and add System.Generics.Collections and System.SyncObjs to the uses clause of your WebModule Unit.

```
var  WebModuleClass: TComponentClass = TWebModule1;

implementation

{%CLASSGROUP 'System.Classes.TPersistent'}
{$R *.dfm}

uses
 System.StrUtils, System.Generics.Collections, System.SyncObjs;

var
 gLock: TObject;
 gKeyValueStore: TDictionary<string, string>;
```

❺ At the end of the WebModule unit add an initialization section where you create both the locking object and the Tdictionary.

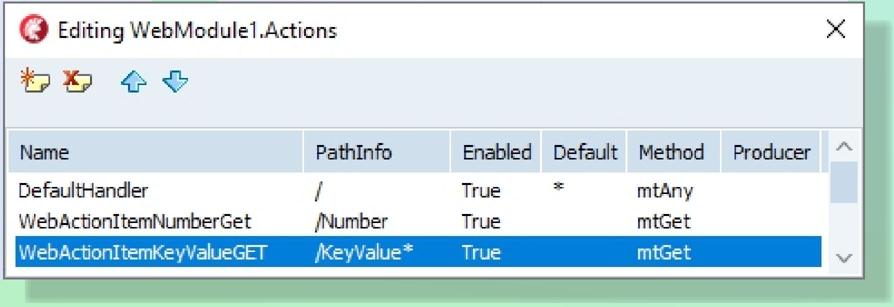
```
initialization

 gLock := TObject.Create;
 gKeyValueStore := TDictionary<string, string>.Create;
 gKeyValueStore.AddOrSetValue('0', 'Zero');

end.
```

We now have an in-memory Key Value store. To retrieve a value in response from a HTTP GET request we need to do some additional legwork.

First we need to parse the parameters in a HTTP request for our resource identifiers. In a HTTP request to a REST webservice a parameter is usually sent by using additional URL segments. So to get the Value for Key 0 in the KeyValue resource you'd use a URI like this
`http://localhost:8080/KeyValue/0`
For REST web services using URL segments is the preferred and also the most simple method. But if you want to you can also support specifying them as URL query parameters, starting with the question mark and separated by ampersands.
`http://localhost:8080/KeyValue?key=0`
This second method is already supported in the TWebRequest with the QueryFields method, but to support using the preferred URL segment parameters we need to add a bit of code as well as change the PathInfo of the Item for the KeyValue GET.

❻ Modify the PathInfo for the WebActionItem for the KeyValue GET, and add a * at the end like this (see figure 6)

Figure 6: Response



Editing WebModule1.Actions

Name	PathInfo	Enabled	Default	Method	Producer
DefaultHandler	/	True	*	mtAny	
WebActionItemNumberGet	/Number	True		mtGet	
WebActionItemKeyValueGET	/KeyValue*	True		mtGet	

This * makes sure that any URL that starts with /KeyValue, but continues with additional URL segments actually ends up in this WebActionItem handler. So **`http://localhost:8080/KeyValue/0`** is now also handled by this action handler.

❼ Next we create a function to parse both the URL query parameters as well as the URL segment parameters. Add a protected function declaration to the WebModule.

```
 private
  { Private declarations }
 protected
  function GetParameters(const aActionPath,
aRequestPath: string): TStringDynArray;
 public
  { Public declarations }
 end;
```

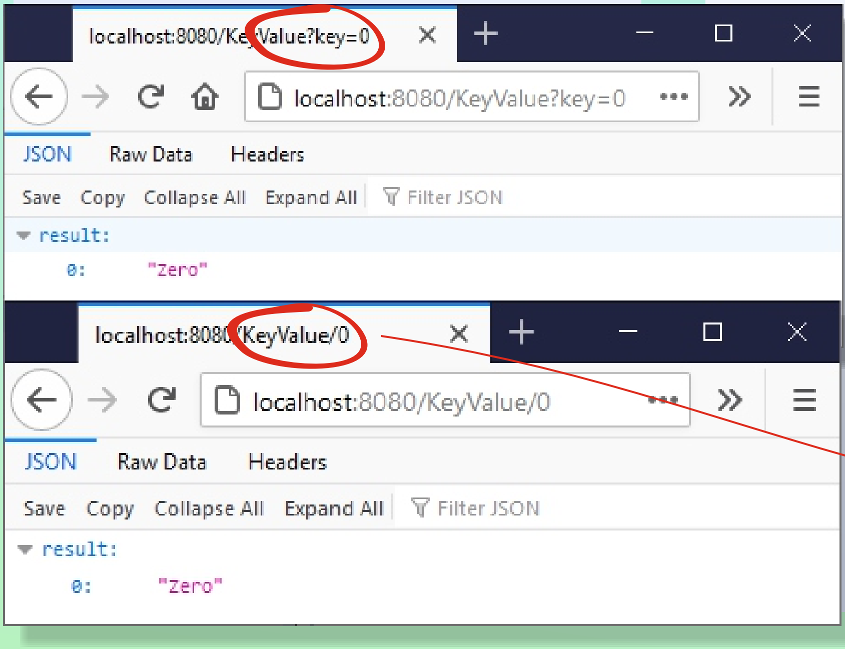
❽ And write the following code to parse both types of parameters from the URI

```
function TWebModule1.GetParameters(const aActionPath,
  aRequestPath: string): TStringDynArray;
var
  lActionPathLength, lRequestPathLength: Integer;
  lParameter: string;
  lParameters: TStringDynArray;
begin
  SetLength(Result, 0);
  lActionPathLength := aActionPath.Length;
  lRequestPathLength := aRequestPath.Length;
  if (lRequestPathLength > lActionPathLength) then
  begin
     lParameter := RightStr(aRequestPath,
          lRequestPathLength - lActionPathLength);
     lParameters := SplitString(lParameter,'/');
     if (Length(lParameters) > 0) then
     begin
       Result := lParameters;
     end
  end;
end;
```

❾ Then modify the OnAction event-handler for the WebActionItemKeyValue GET Action

```
procedure TWebModule1.WebModule1WebActionItemKeyValueGETAction(
  Sender: TObject; Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
  {GET  - "Select" / Idempotent}
var
  lParameters: TStringDynArray; lKey, lValue: string;
begin
  lKey := '';
  {parse URL query parameters - http://localhost:8080/KeyValue?key=0}
  lKey := Request.QueryFields.Values['key'];
  if (lKey.IsEmpty) then
  begin  {parse URL segment parameters - http://localhost:8080/KeyValue/0}
    lParameters := GetParameters((Sender as TWebActionItem).PathInfo, Request.PathInfo);
    if (Length(lParameters) > 0) then
      begin
        lKey := lParameters[0];
       end;
  end;
  if not(lKey.IsEmpty) then
  begin
    Response.ContentType := 'application/json; charset=UTF-8';
    gKeyValueStore.TryGetValue(lKey, lValue);
    if not(lValue.IsEmpty) then
    begin  // {"result":["string"]}
      Response.Content := '{"result":["' + lValue + '"]}';
    end
    else
      begin  // {"error":"Item not found"}
        Response.Content := '{"error":"Item not found"}';
      end;
    Handled := True;
    end
    else {No parameters on URL for GET request}
    Handled := False;
end;
```

Test if this works, either with the REST debugger or using the web browser. The results should look like this.



```
result:
    0:      "Zero"
```

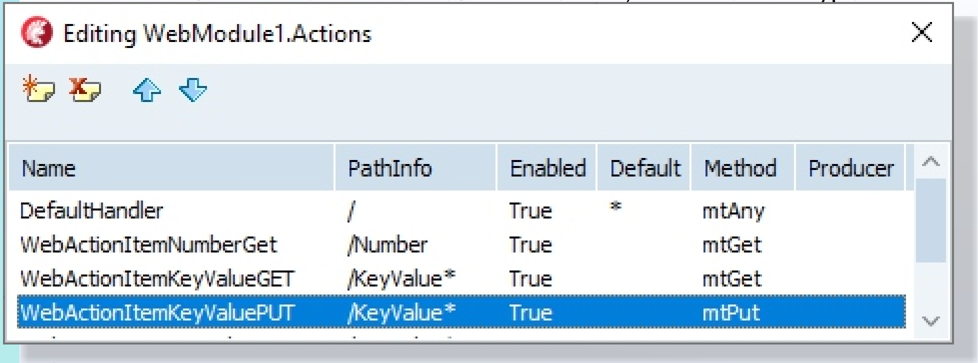
```
result:
    0:      "Zero"
```

Figure 7: Zero

prefered

Our next major step is to add both a PUT and a POST handler.
When data is sent to a web service, the data can be sent as part of a URL segment like this `http://localhost:8080/KeyValue/1/One` but his method has some limitations, one obvious one being that not all characters are allowed in URL segments as they have a special meaning. If you want to send larger or more complex items you would use the HTTP requests body. We want to support both methods of sending data to our web service.

⓾ Add a PUT handler to the WebModule unit, with method type mtPut.



Editing WebModule1.Actions

Name	PathInfo	Enabled	Default	Method	Producer
DefaultHandler	/	True	*	mtAny	
WebActionItemNumberGet	/Number	True		mtGet	
WebActionItemKeyValueGET	/KeyValue*	True		mtGet	
WebActionItemKeyValuePUT	/KeyValue*	True		mtPut	

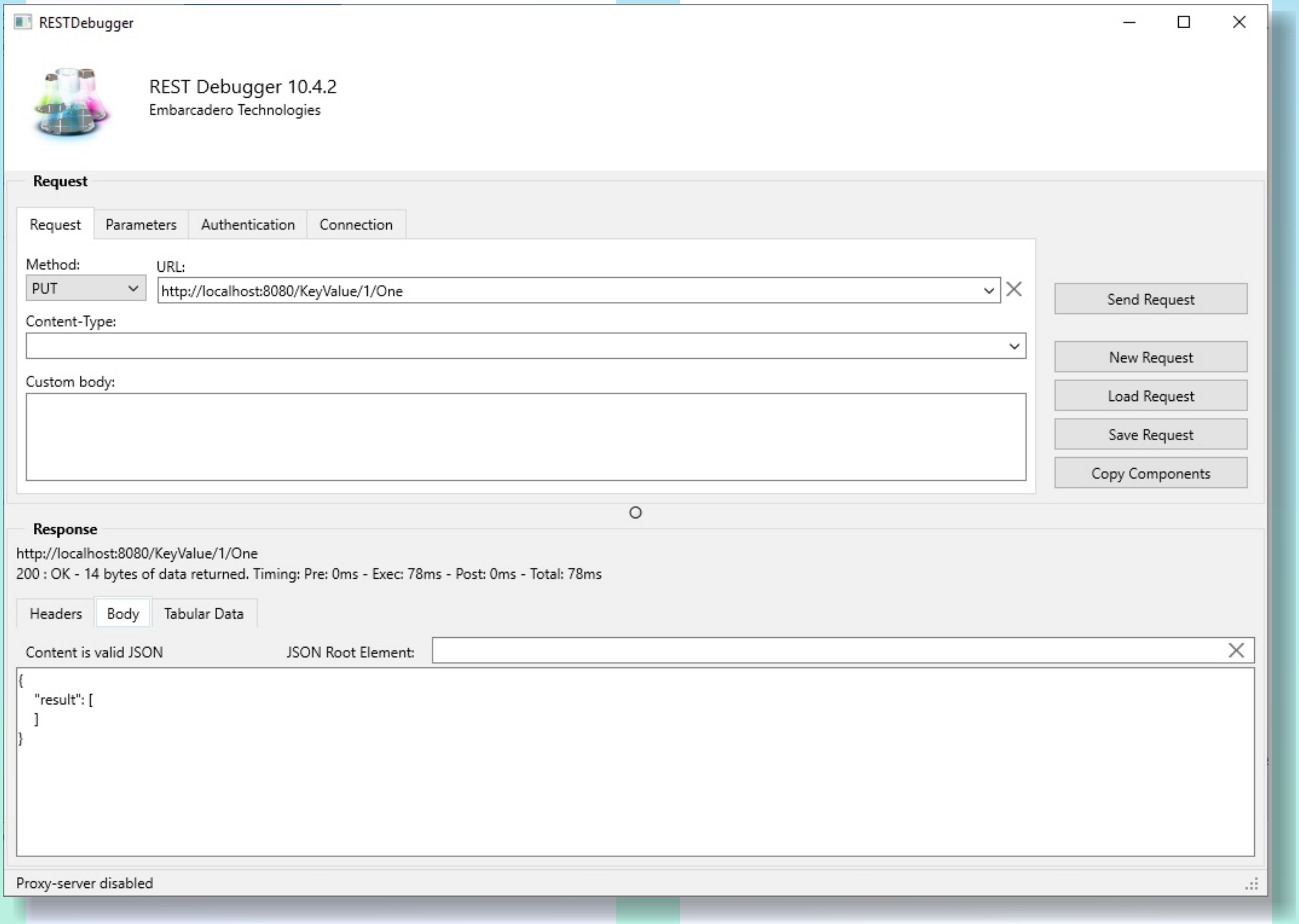
Figure 8: Actions

⓫ Then code the handler

```
procedure TWebModule1.WebModule1WebActionItemKeyValuePUTAction(
  Sender: TObject; Request: TWebRequest; Response: TWebResponse;
  var Handled: Boolean);
  {PUT  - "Insert or Update" / Idempotent}
var
  lParameters: TStringDynArray;
  lKey: string;
  lValue: string;
begin
  lParameters := GetParameters((Sender as TWebActionItem).PathInfo, Request.PathInfo);
  if (Length(lParameters) > 0) then
  begin
    lKey := lParameters[0];
    lValue := '';
    if (Length(lParameters) > 1) then
      begin {Value is part or URL and a simple string}
        lValue := lParameters[1]
      end
  else
    begin {Value is send as content in the request and possibly a JSON or other complex string}
      lValue := Request.Content;
    end;
  if not(lValue.IsEmpty) then
    begin
      Response.ContentType := 'application/json; charset=UTF-8';
      gKeyValueStore.AddOrSetValue(lKey, lValue);
      Response.Content := ' {"result":[]}';
      Handled := True;
    end
  else
    begin
      Handled := False;
    end;
  end
  else
    begin
      {Do not reply}
      Handled := False;
    end;
end;
```

Now test this new PUT method and insert some data into your webservice using the REST Debugger.

Figure 9: Result



Method: PUT

URL: http://localhost:8080/KeyValue/1/One

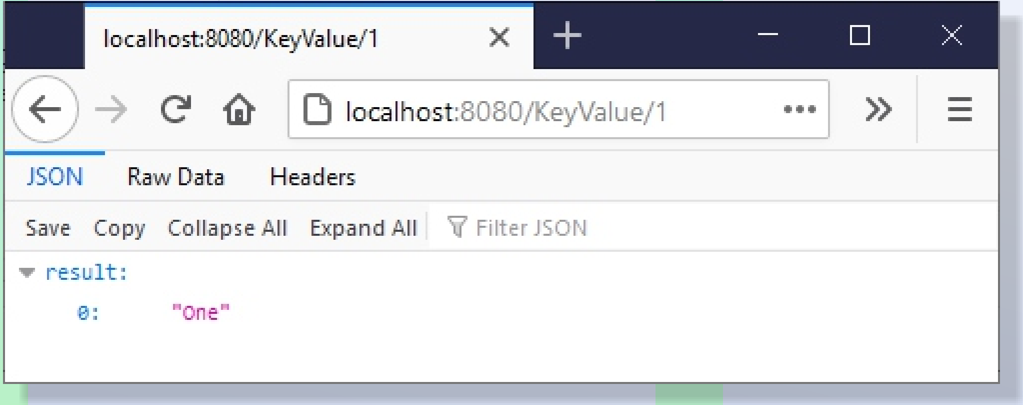
Content-Type:

Custom body:

```
http://localhost:8080/KeyValue/1/One
200 : OK - 14 bytes of data returned. Timing: Pre: 0ms - Exec: 78ms - Post: 0ms - Total: 78ms
```

```
{
   "result": [
   ]
}
```

And request the new item to see if it was saved correctly.

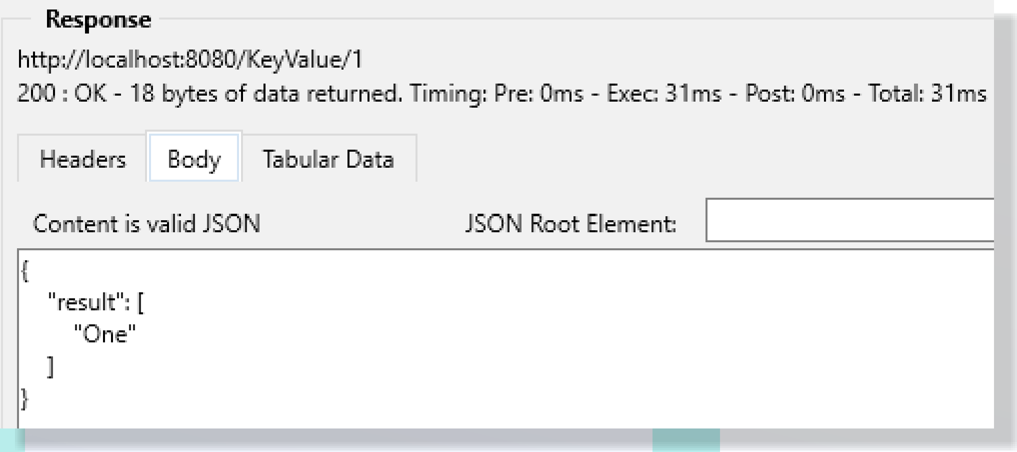


localhost:8080/KeyValue/1

```
result:
    0:    "One"
```

Figure 10: Result One

At first glance the result might seem odd, but that is because the result in this case is returned as a JSON array, and the first item of an array has index 0. If you look at it in the REST debugger you'll see that this is valid JSON.



Response

http://localhost:8080/KeyValue/1

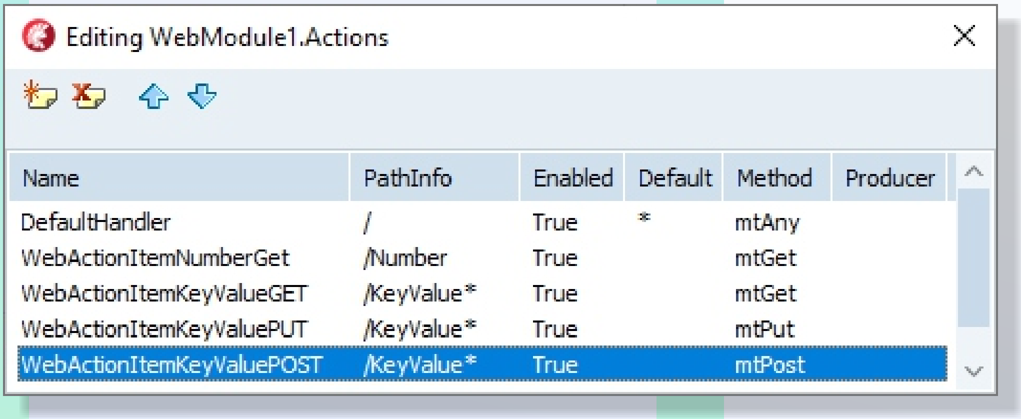
200 : OK - 18 bytes of data returned. Timing: Pre: 0ms - Exec: 31ms - Post: 0ms - Total: 31ms

Content is valid JSON JSON Root Element:

```
{
   "result": [
      "One"
   ]
}
```

Figure 11: Result One

⓬ We will do the same for POST. We add a handler and use method type mtPOST.

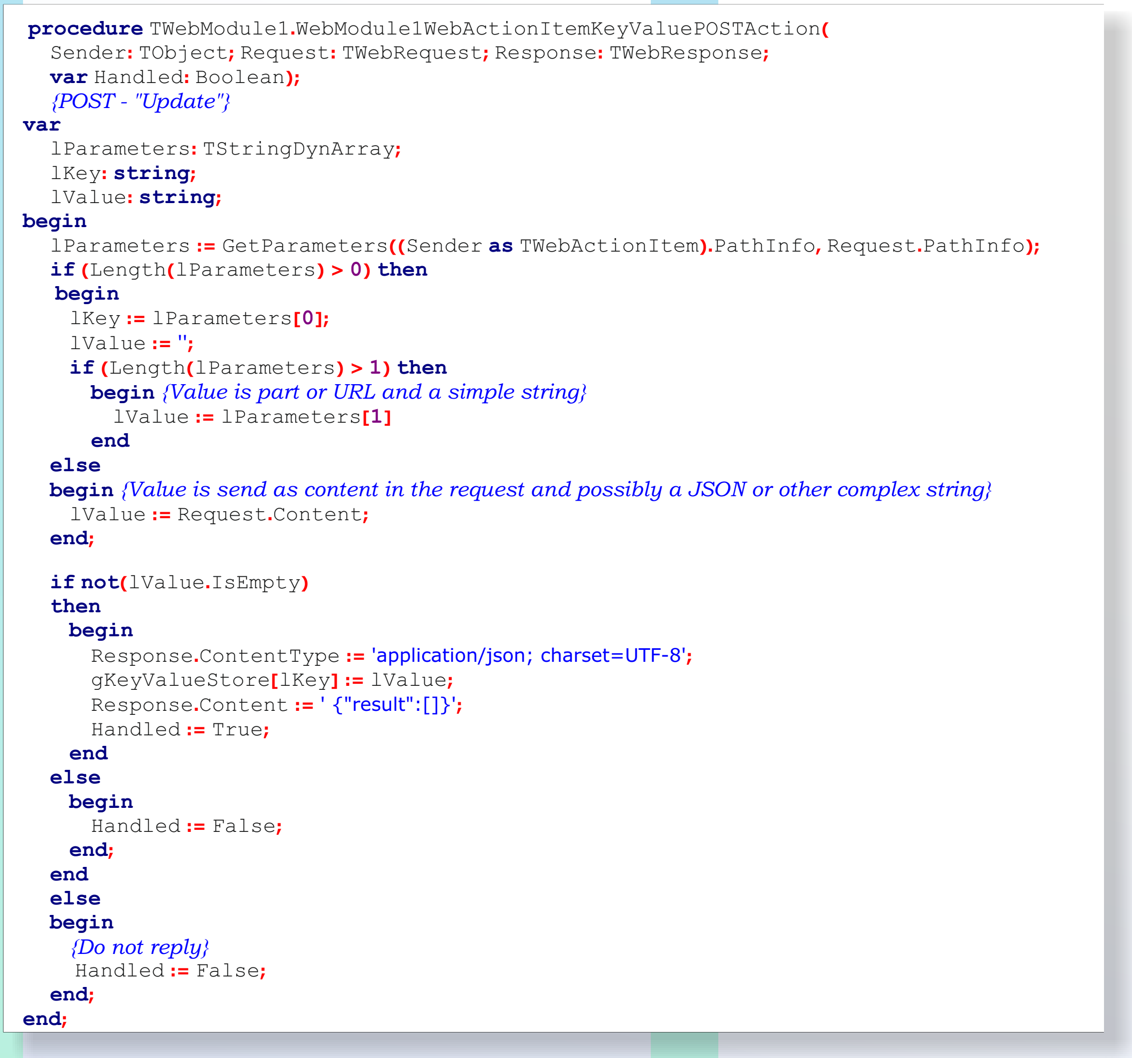


Editing WebModule1.Actions

Name	PathInfo	Enabled	Default	Method	Producer
DefaultHandler	/	True	*	mtAny	
WebActionItemNumberGet	/Number	True		mtGet	
WebActionItemKeyValueGET	/KeyValue*	True		mtGet	
WebActionItemKeyValuePUT	/KeyValue*	True		mtPut	
WebActionItemKeyValuePOST	/KeyValue*	True		mtPost	

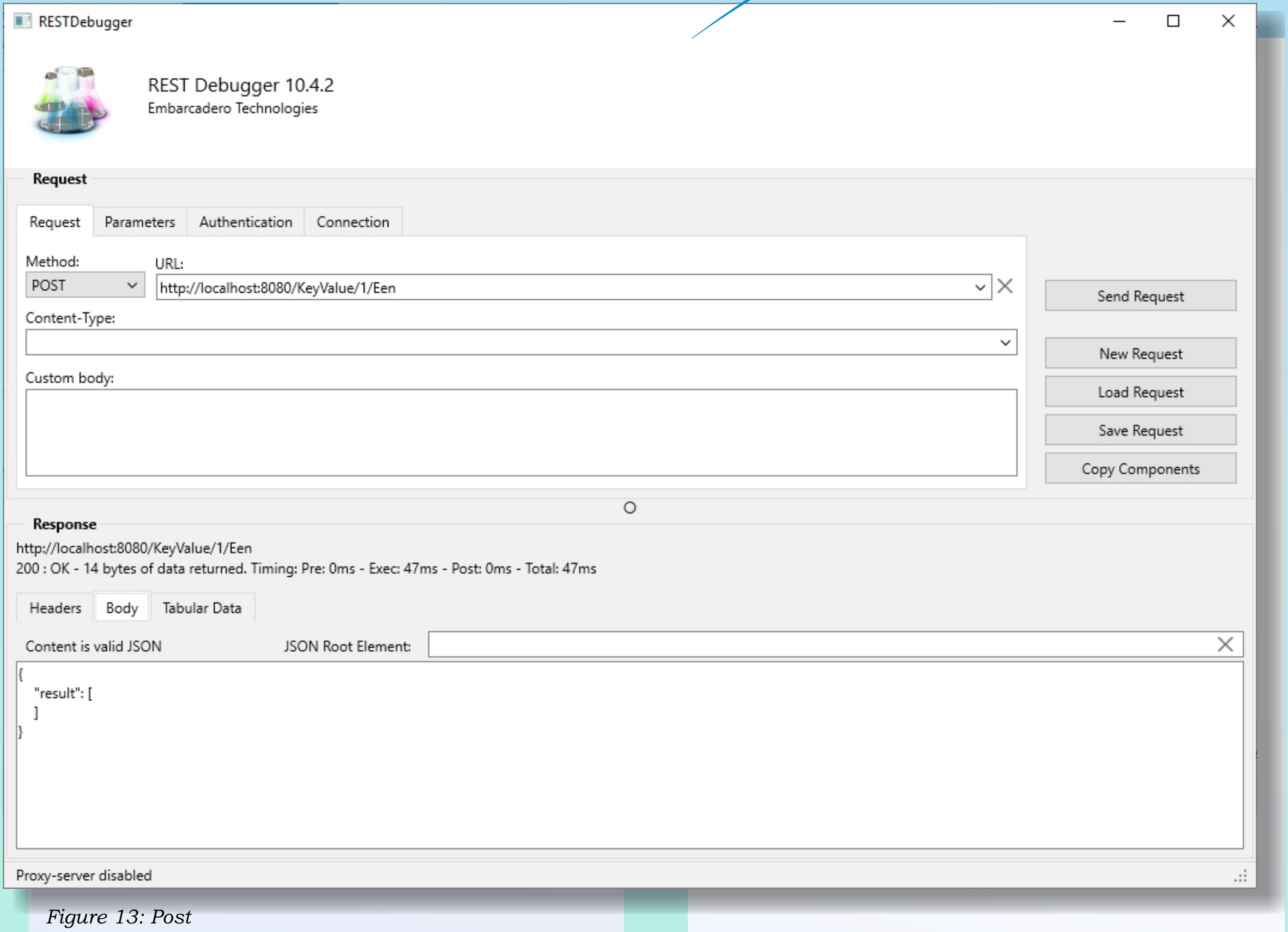
Figure 12: Post

13 And the code



```
procedure TWebModule1.WebModule1WebActionItemKeyValuePOSTAction(
  Sender: TObject; Request: TWebRequest; Response: TWebResponse;
  var Handled: Boolean);
  {POST - "Update"}
var
  lParameters: TStringDynArray;
  lKey: string;
  lValue: string;
begin
  lParameters := GetParameters((Sender as TWebActionItem).PathInfo, Request.PathInfo);
  if (Length(lParameters) > 0) then
   begin
    lKey := lParameters[0];
    lValue := '';
    if (Length(lParameters) > 1) then
      begin {Value is part or URL and a simple string}
        lValue := lParameters[1]
      end
  else
  begin {Value is send as content in the request and possibly a JSON or other complex string}
    lValue := Request.Content;
  end;

  if not(lValue.IsEmpty)
  then
    begin
      Response.ContentType := 'application/json; charset=UTF-8';
      gKeyValueStore[lKey] := lValue;
      Response.Content := ' {"result":[]}';
      Handled := True;
    end
  else
    begin
      Handled := False;
    end;
  end
  else
  begin
    {Do not reply}
    Handled := False;
  end;
end;
```

Test it with the REST Debugger. First add an item with Key 1 and Value One and then update this existing item using POST, changing the Value to something other than One. In my example I used Een, which is the dutch word for One. You probably never guessed you'd learn a dutch word from reading this article.



Method: POST

URL: http://localhost:8080/KeyValue/1/Een

Content-Type:

Custom body:

http://localhost:8080/KeyValue/1/Een

200 : OK - 14 bytes of data returned. Timing: Pre: 0ms - Exec: 47ms - Post: 0ms - Total: 47ms

Content is valid JSON

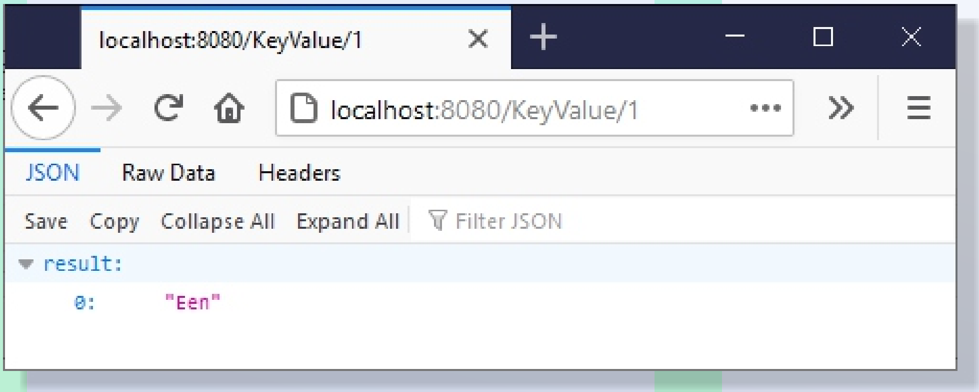
JSON Root Element:

```
{
   "result": [
   ]
}
```

Proxy-server disabled

Figure 13: Post

And a request in the web browser to verify it worked.



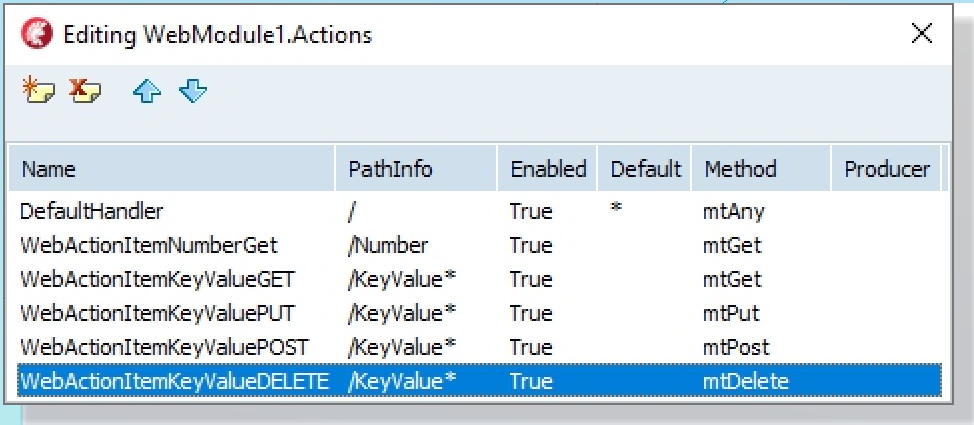
```
▼ result:
    0:     "Een"
```

Figure 14: Result: Een

At this point you may have already run into a potential issue we just introduced when adding the `POST` method. What happens if you try to `POST` a Value for a non-existing key?

Just try it out with the REST Debugger.
We need to improve on this, but we will do so in our next article.

⓮ Our last **`HTTP Command`** will be the **`DELETE`**. Just add it to the handlers as before, this time with **`mtDELETE`**.



Editing WebModule1.Actions

Name	PathInfo	Enabled	Default	Method	Producer
DefaultHandler	/	True	*	mtAny	
WebActionItemNumberGet	/Number	True		mtGet	
WebActionItemKeyValueGET	/KeyValue*	True		mtGet	
WebActionItemKeyValuePUT	/KeyValue*	True		mtPut	
WebActionItemKeyValuePOST	/KeyValue*	True		mtPost	
WebActionItemKeyValueDELETE	/KeyValue*	True		mtDelete	

Figure 15: mtDelete

⑮ The code is straightforward:

```
procedure TWebModule1.WebModule1WebActionItemKeyValueDELETEAction(Sender: TObject; Request: TWebRequest;
  Response: TWebResponse; var Handled: Boolean);
  DELETE - "Delete"}
var
  lParameters: TStringDynArray;
  lKey: string;
begin
  lParameters := GetParameters((Sender as TWebActionItem).PathInfo, Request.PathInfo);
  if (Length(lParameters) > 0) then
  begin
    lKey := lParameters[0];
    gKeyValueStore.Remove(lKey);
    Response.ContentType := 'application/json; charset=UTF-8';
    Response.Content := '{"result":[]}';
    Handled := True;
  end
  else {No parameters on URL for GET request}
    Handled := False;
end;
```

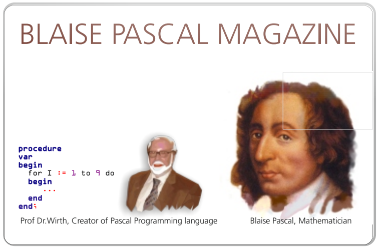
You may have noticed that we left out URL query parameter parsing in the PUT, POST and DELETE. This was intentional, as using URL segment parameters is the preferred method for a REST web service. However due to the limitations of URL segment parameters, you may wish to add URL query parameters for the PUT/POST and DELETE as well. This could be done quite easily with a duplicate of the sample code from the GET request handler.

A short recap of the things we have done in this article. We created a GET request that correctly parses both URL query parameters and URL segment parameters and returns a value from the in-memory Key Value store. We also created PUT and POST requests that handle passing content as part of the URL segment, for short values, as well as from the HTTP content stream, for larger more complex values.

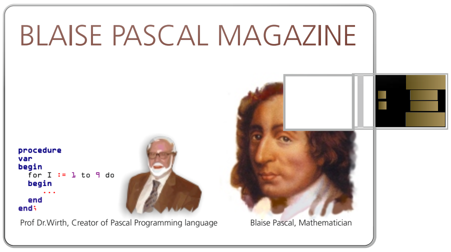
As a teaser we also added a gLock object, but we did not actually write any code for it. We will do that in our next article, where we'll add some code to make sure any access to our in-memory Key Value store is handled in a thread safe manner. In that next article we will also add error handling and of course expand our Web Service client to consume our web service. There might even be some JavaScript code to consume our web service and further on some additional JSON serialization. Stay tuned!

Advertisement

BLAISE PASCAL MAGAZINE 93
Multi Platform / Object Pascal / Internet / JavaScript / WebAssembly / Pas2Js / Databases
CSS Styles / Progressive Web Apps
Android / IOS / Mac / Windows & Linux
Blaise Pascal
Delphinium
First impressions with Delphi FMX / By Detlef Overbeek
Cartoon / By Jerry king
Max Box RSS Feeds of BBC News Sentiment Analysis / By Max Kleiner
Delphi 10.4.2 / new features / By Detlef Overbeek
Migration Tool - How to keep your settings in Delphi / By Detlef Overbeek
New Number Component inside Delphi 10.4.2 / By Jim Mckeeth (Embarcadero)
New ControlList Component in Delphi 10.4.2 / By Jim Mckeeth (Embarcadero)
Updated Konopka Components / By Detlef Overbeek
Code Snippets 11/12 / By Detlef Overbeek
Web Service Part 2 / By Danny Wind
Lazarus: Webform inside lazarus with WYSIWYG / Michael van Canneyt


BLAISE PASCAL MAGAZINE
procedure
var
begin
for I := 1 to 9 do
begin
...
end
end;
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician

Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands
Prof Dr.Wirth, Creator of Pascal Programming language
editor@blaisepascalmagazine.eu
https://www.blaisepascalmagazine.eu


BLAISE PASCAL MAGAZINE
procedure
var
begin
for I := 1 to 9 do
begin
...
end
end;
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician

1 year Subscription+ the new LibStick

(on USB Card - 90 Issues)

€ 70

ex vat / inc. shipment

ORM in kbmMW #3 – Copying a table from one database to another

By Kim Bo Madsen

PREFACE

In 5.15.xx kbmMW's ORM got even better since it allow us to easily copy table structure and data from one table in one database to another database.

THE TABLE STRUCTURE

ORM tables can be defined in a couple of ways, where a usual way is to define a Delphi class and provide attributes for it that makes the class a template for a table in the ORM, which in turn then can be used to automatically create a table in the datastore which the current ORM is connected to.

But what if the table in the database do not exist as a Delphi class? Well the ORM is fortunately able to create one for us, based on just about any existing table in any of the supported databases.

```
procedure TForm1.CreateDelphiClass(const ATableName:string; const AStrings:TStrings);
var
 t:TkbmMWORMTable;
begin
   t:=FORMSrc.SurfaceDynamicTable(ATableName,[mwsdtoPromoteFieldDefaultsAsGenerators]);
   if t<>nil then
     AStrings.Text:=FORMSrc.GenerateDelphiClass(t,[mwocgoNoSchemaNameInClassName]);
end;
```

The above code shows how to easily have the **ORM,** at runtime, generate the needed Delphi code that describes a table, and thus can be used for creating tables in all sorts of **ORM** connected databases.

In the above example, **`FORMSrc`** is a variable/field of type **`TkbmMWORM`**, which has been connected to a connection pool, that represents the database.

```
FORMSrc:=TkbmMWORM.Create(cpSrc);
```

`chSrc` is the connection pool, defined at designtime, and in this case hooked up to a **FireDAC** database instance connected to an old **Firebird 1.5 database** (which happens to be our old newsgroup database).

All we need to do is to ask the **ORM** to surface the table based on the table name, and then produce the class for that table.

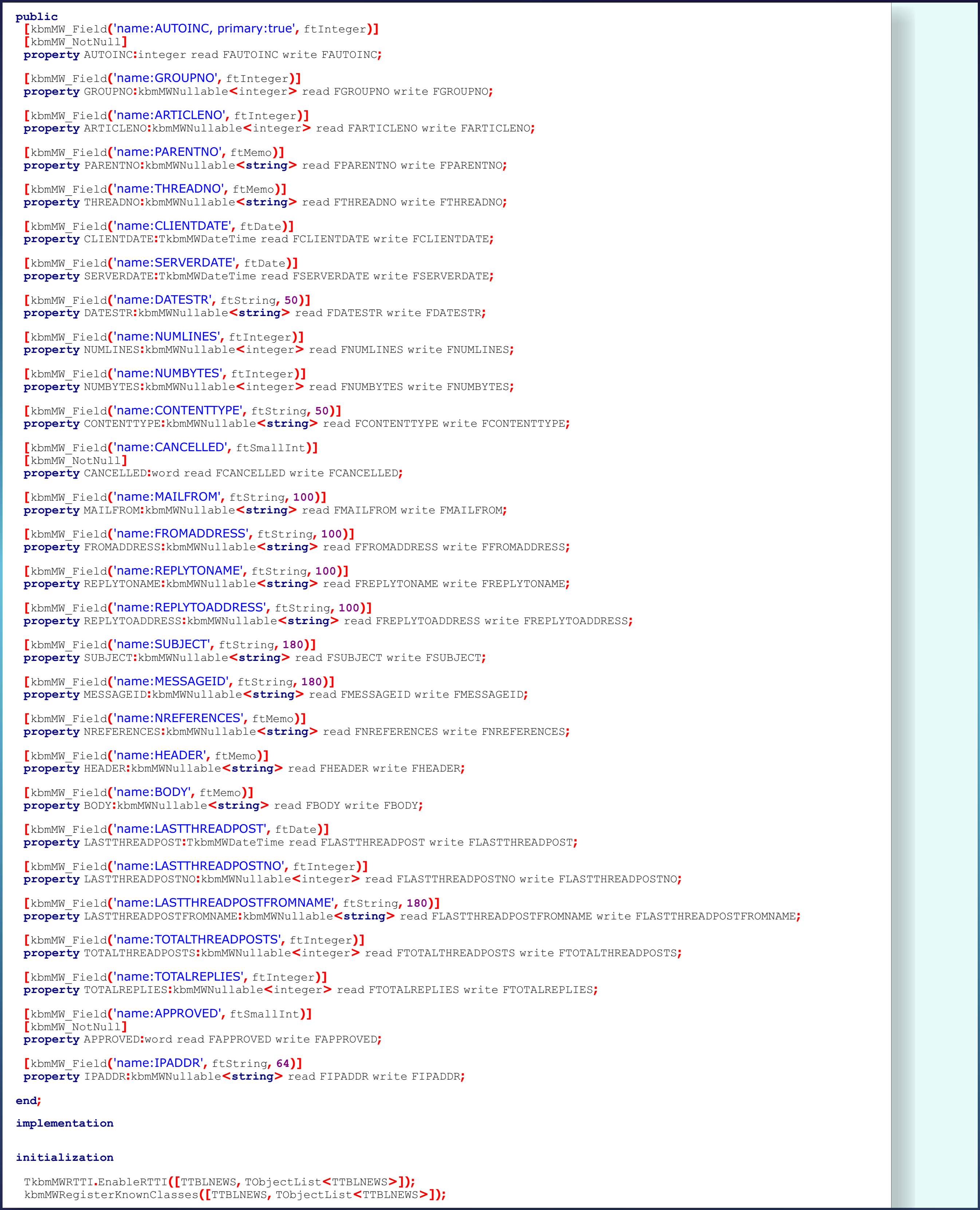
Surfacing a table means that the **ORM** gathers very detailed information about the table structure, indexes, generators, constraints etc. and provide us with a **`TkbmMWORMTable`** instance that we can use for accessing that table via the **ORM.**

In fact enough detail is collected to be able to reproduce the table (*within the scope of the ORM*) elsewhere, which we utilize by asking the **ORM** to generate the Delphi code for us for that particular surfaced table using the **`GenerateDelphiClass`** method.

The output of the calling the method with the table name **`TBLNEWS`** looks like this

```
interface

uses
 DB, Math, Classes, System.Generics.Collections,
 kbmMWGlobal, kbmMWObjectMarshal, kbmMWNullable,
 kbmMWORM, kbmMWRTTI;

type

[kbmMW_Table('name:TBLNEWS')]
TTBLNEWS = class

private
 FAUTOINC:integer;
 FGROUPNO:kbmMWNullable<integer>;
 FARTICLENO:kbmMWNullable<integer>;
 FPARENTNO:kbmMWNullable<string>;
 FTHREADNO:kbmMWNullable<string>;
 FCLIENTDATE:TkbmMWDateTime;
 FSERVERDATE:TkbmMWDateTime;
 FDATESTR:kbmMWNullable<string>;
 FNUMLINES:kbmMWNullable<integer>;
 FNUMBYTES:kbmMWNullable<integer>;
 FCONTENTTYPE:kbmMWNullable<string>;
 FCANCELLED:word;
 FMAILFROM:kbmMWNullable<string>;
 FFROMADDRESS:kbmMWNullable<string>;
 FREPLYTONAME:kbmMWNullable<string>;
 FREPLYTOADDRESS:kbmMWNullable<string>;
 FSUBJECT:kbmMWNullable<string>;
 FMESSAGEID:kbmMWNullable<string>;
 FNREFERENCES:kbmMWNullable<string>;
 FHEADER:kbmMWNullable<string>;
 FBODY:kbmMWNullable<string>;
 FLASTTHREADPOST:TkbmMWDateTime;
 FLASTTHREADPOSTNO:kbmMWNullable<integer>;
 FLASTTHREADPOSTFROMNAME:kbmMWNullable<string>;
 FTOTALTHREADPOSTS:kbmMWNullable<integer>;
 FTOTALREPLIES:kbmMWNullable<integer>;
 FAPPROVED:word;
 FIPADDR:kbmMWNullable<string>;
```

By Kim Bo Madsen



```
public
 [kbmMW_Field('name:AUTOINC, primary:true', ftInteger)]
 [kbmMW_NotNull]
 property AUTOINC:integer read FAUTOINC write FAUTOINC;

 [kbmMW_Field('name:GROUPNO', ftInteger)]
 property GROUPNO:kbmMWNullable<integer> read FGROUPNO write FGROUPNO;

 [kbmMW_Field('name:ARTICLENO', ftInteger)]
 property ARTICLENO:kbmMWNullable<integer> read FARTICLENO write FARTICLENO;

 [kbmMW_Field('name:PARENTNO', ftMemo)]
 property PARENTNO:kbmMWNullable<string> read FPARENTNO write FPARENTNO;

 [kbmMW_Field('name:THREADNO', ftMemo)]
 property THREADNO:kbmMWNullable<string> read FTHREADNO write FTHREADNO;

 [kbmMW_Field('name:CLIENTDATE', ftDate)]
 property CLIENTDATE:TkbmMWDateTime read FCLIENTDATE write FCLIENTDATE;

 [kbmMW_Field('name:SERVERDATE', ftDate)]
 property SERVERDATE:TkbmMWDateTime read FSERVERDATE write FSERVERDATE;

 [kbmMW_Field('name:DATESTR', ftString, 50)]
 property DATESTR:kbmMWNullable<string> read FDATESTR write FDATESTR;

 [kbmMW_Field('name:NUMLINES', ftInteger)]
 property NUMLINES:kbmMWNullable<integer> read FNUMLINES write FNUMLINES;

 [kbmMW_Field('name:NUMBYTES', ftInteger)]
 property NUMBYTES:kbmMWNullable<integer> read FNUMBYTES write FNUMBYTES;

 [kbmMW_Field('name:CONTENTTYPE', ftString, 50)]
 property CONTENTTYPE:kbmMWNullable<string> read FCONTENTTYPE write FCONTENTTYPE;

 [kbmMW_Field('name:CANCELLED', ftSmallInt)]
 [kbmMW_NotNull]
 property CANCELLED:word read FCANCELLED write FCANCELLED;

 [kbmMW_Field('name:MAILFROM', ftString, 100)]
 property MAILFROM:kbmMWNullable<string> read FMAILFROM write FMAILFROM;

 [kbmMW_Field('name:FROMADDRESS', ftString, 100)]
 property FROMADDRESS:kbmMWNullable<string> read FFROMADDRESS write FFROMADDRESS;

 [kbmMW_Field('name:REPLYTONAME', ftString, 100)]
 property REPLYTONAME:kbmMWNullable<string> read FREPLYTONAME write FREPLYTONAME;

 [kbmMW_Field('name:REPLYTOADDRESS', ftString, 100)]
 property REPLYTOADDRESS:kbmMWNullable<string> read FREPLYTOADDRESS write FREPLYTOADDRESS;

 [kbmMW_Field('name:SUBJECT', ftString, 180)]
 property SUBJECT:kbmMWNullable<string> read FSUBJECT write FSUBJECT;

 [kbmMW_Field('name:MESSAGEID', ftString, 180)]
 property MESSAGEID:kbmMWNullable<string> read FMESSAGEID write FMESSAGEID;

 [kbmMW_Field('name:NREFERENCES', ftMemo)]
 property NREFERENCES:kbmMWNullable<string> read FNREFERENCES write FNREFERENCES;

 [kbmMW_Field('name:HEADER', ftMemo)]
 property HEADER:kbmMWNullable<string> read FHEADER write FHEADER;

 [kbmMW_Field('name:BODY', ftMemo)]
 property BODY:kbmMWNullable<string> read FBODY write FBODY;

 [kbmMW_Field('name:LASTTHREADPOST', ftDate)]
 property LASTTHREADPOST:TkbmMWDateTime read FLASTTHREADPOST write FLASTTHREADPOST;

 [kbmMW_Field('name:LASTTHREADPOSTNO', ftInteger)]
 property LASTTHREADPOSTNO:kbmMWNullable<integer> read FLASTTHREADPOSTNO write FLASTTHREADPOSTNO;

 [kbmMW_Field('name:LASTTHREADPOSTFROMNAME', ftString, 180)]
 property LASTTHREADPOSTFROMNAME:kbmMWNullable<string> read FLASTTHREADPOSTFROMNAME write FLASTTHREADPOSTFROMNAME;

 [kbmMW_Field('name:TOTALTHREADPOSTS', ftInteger)]
 property TOTALTHREADPOSTS:kbmMWNullable<integer> read FTOTALTHREADPOSTS write FTOTALTHREADPOSTS;

 [kbmMW_Field('name:TOTALREPLIES', ftInteger)]
 property TOTALREPLIES:kbmMWNullable<integer> read FTOTALREPLIES write FTOTALREPLIES;

 [kbmMW_Field('name:APPROVED', ftSmallInt)]
 [kbmMW_NotNull]
 property APPROVED:word read FAPPROVED write FAPPROVED;

 [kbmMW_Field('name:IPADDR', ftString, 64)]
 property IPADDR:kbmMWNullable<string> read FIPADDR write FIPADDR;

end;

implementation


initialization

 TkbmMWRTTI.EnableRTTI([TTBLNEWS, TObjectList<TTBLNEWS>]);
 kbmMWRegisterKnownClasses([TTBLNEWS, TObjectList<TTBLNEWS>]);
```

Obviously we can simply use

```
FORMDst.CreateOrUpgradeTable(TTBLNEWS);
```

to create the table structure in a different database, pointed to by the **FORMDst TkbmMWORM** instance.
After that we can copy the data over.
However we can do it even simpler, without the need to compile an application containing our specific **TTBLNEWS** class.

```
tSrc:=FORMSrc.SurfaceDynamicTable(ATableName,[mwsdtoPromoteFieldDefaultsAsGenerators]);
  if tSrc<>nil then
  begin
     // If source table surfaced, clone it to the destination ORM.
     tDst:=tSrc.Clone(FORMDst);
     FORMDst.CreateOrUpgradeTable(tDst);
  end;
```

Again we surface the original table, but this time we do not create a Delphi class. Instead we ask the **ORM** to clone the **TkbmMWORMTable** instance onto another **ORM (FORMDst)** connected to another database.
As the **FORMDst ORM** now knows about our table, we can directly use it to **CreateOrUpgradeTable**, and thus very easily have copied the table structure completely dynamically without linking in any Delphi classes describing the tables.

THE DATA
Now all that is missing is copying the data. This can fortunately also be done easily. Simply query the source **ORM**, make sure that the data is marked as to be inserted and persist the data in the destination **ORM**.

And in our case, try to move the data in batches, to avoid loading loads of records into memory. This is not important for smaller tables (*few tens of thousands of records*), but for a very large amount of records, using the new batch features in **kbmMW v. 5.15.xx** may be of good assistance.

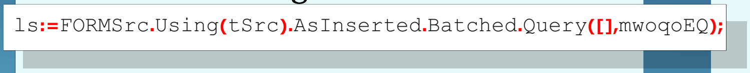
NOTICE though that I have had limited success getting FireDAC to actually not fetching all data from the **Firebird 1.5 database**, so even if **kbmMW** may batch, any **DB API** may not always do it.
A final **MoveTable** method could look like this

```
procedure TForm1.MoveTable(const ATableName:string);
var
  tSrc,tDst:TkbmMWORMTable;
  ds:TkbmMWCustomPooledCursor;
  ls:IkbmMWORMQueryStage<TkbmMWORMQueryStageTable>;
begin
   // Check if destination table exists, then drop it.
   tDst:=FORMDst.SurfaceDynamicTable(ATableName,
           [mwsdtoPromoteFieldDefaultsAsGenerators]);
   if tDst<>nil then
     FORMDst.DeleteTable(tDst);

   // Now create table from source.
   tSrc:=FORMSrc.SurfaceDynamicTable(ATableName,
           [mwsdtoPromoteFieldDefaultsAsGenerators]);
   if tSrc<>nil then
   begin
      // If source table surfaced, clone it to the destination ORM.
      tDst:=tSrc.Clone(FORMDst);
      FORMDst.CreateOrUpgradeTable(tDst);

      // Copy data.
      ls:=FORMSrc.Using(tSrc).
                 AsInserted.Batched.Query([],mwoqoEQ);
      repeat
         ds:=TkbmMWCustomPooledCursor(ls.AsDataset);
         if ds<>nil then
          FORMDst.PersistDataset(tDst,ds);
      until ds=nil;
   end;
end;
```

In this case we first check to see if the destination database already contains a table with same name. If so, we delete it. You may not like to do that and could thus simply avoid the call to DeleteTable.

Next we establish information about the source table and clone it to the destination **ORM** after which we ask the **ORM** to ensure that a compatible table with that name and structure exists or will be made to exist on the database represented by the destination **ORM.**

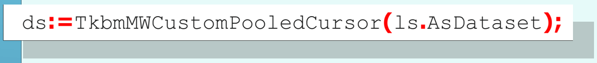
Finally we use a new `5.15.xx` feature allowing us to query records and have them returned ready for insert elsewhere, and in a batched mode in the following statement



```
ls:=FORMSrc.Using(tSrc).AsInserted.Batched.Query([],mwoqoEQ);
```

We use the fluid variant of the query syntax to ask for data from the source table, matching any key value, returned ready for insertion, and we only want a batch at a time. A batch is currently 100 records and as is not configurable.

When we execute the line



```
ds:=TkbmMWCustomPooledCursor(ls.AsDataset);
```

we will be given a batch, or if there are no more batches the value nil will be returned to us.

While ds has a value we persist the dataset using the destination **ORM,** and loop until we get the value nil at which point no more records can be copied.

Do not free the returned ds instance. That is entirely handled internally by the **ORM** query stage logic when it goes out of scope.

PROLOGUE

The above demonstrates one new **ORM** feature in `5.15.xx`.

Another new feature in v. `5.15.xx` is that it is able to understand partial **Firebird/Interbased intermediate language (BLR)** and thus able to determine if default column values has been defined and which they are, as long as they have been defined to a constant.

The Bumble Bee...

By Panoramedia - Own work, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=20441726

POPULATION DECLINE

Bumblebee species are declining in Europe, North America, and Asia due to a number of factors, including land-use change that reduces their food plants. In North America, pathogens are possibly having a stronger negative effect especially for the subgenus Bombus.
A major impact on bumblebees was caused by the mechanisation of agriculture, accelerated by the urgent need to increase food production during the Second World War.
Small farms depended on horses to pull implements and carts.
The horses were fed on clover and hay, both of which were permanently grown on a typical farm. Little artificial fertiliser was used.
Farms thus provided flowering clover and flower-rich meadows, favouring bumblebees.
Mechanisation removed the need for horses and most of the clover; artificial fertilisers encouraged the growth of taller grasses, outcompeting the meadow flowers. Most of the flowers, and the bumblebees that fed on them, disappeared from Britain by the early 1980s.
The last native British short-haired bumblebee was captured near Dungeness in 1988. The significant increase in pesticide and fertilizer use associated with the industrialization of agriculture has had adverse effects on the genus Bombus.
The bees are directly exposed to the chemicals in two ways: by consuming nectar that has been directly treated with pesticide, or through physical contact with treated plants and flowers.
The species Bombus hortorum in particular has been found to be impacted by the pesticides;
their brood development has been reduced and their memory has been negatively affected. Additionally, pesticide use negatively impacts colony development and size.

BOMBUS BALTEATUS, BOMBUS TERRICOLA, BOMBUS AFFINIS, BOMBUS OCCIDENTALIS, ARE IN SERIOUS DECLINE

Bumblebees are in danger in many developed countries due to habitat destruction and collateral pesticide damage. The European Food Safety Authority ruled that three neonicotinoid pesticides (**clothianidin, imidacloprid, and thiamethoxam**) presented a high risk for bees. While most work on neonicotinoid toxicity has looked at honeybees, a study on B. terrestris showed that "field-realistic" levels of imidacloprid significantly reduced growth rate and cut production of new queens by 85%, implying a "considerable negative effect" on wild bumblebee populations throughout the developed world. However, in another study, following chronic exposure to field-realistic levels of the **neonicotinoid** pesticide **thiamethoxam,** colony weight gain was not affected, nor were the number or mass of sexuals produced.

Low levels of neonicotinoids can reduce the number of bumblebees in a colony by as much as 55%, and cause dysfunction in the bumblebees' brains. The Bumblebee Conservation Trust considers this evidence of reduced brain function "particularly alarming given that bumblebees rely upon their intelligence to go about their daily tasks.“

A study on B. terrestris had results that suggests that use of neonicotinoid pesticides can affect how well bumblebees are able to forage and pollinate.

Bee colonies that had been affected by the pesticide released more foragers and collected more pollen than bees who had not been dosed with neonicotinoid. Although the bees affected by the pesticide were able to collect more pollen, they took a longer amount of time doing so.

Of 19 species of native nestmaking bumblebees and six species of cuckoo bumblebees formerly widespread in Britain, three have been extirpated, eight are in serious decline, and only six remain widespread. Similar declines have been reported in Ireland, with four species designated endangered, and another two considered vulnerable to extinction.
A decline in bumblebee numbers could cause large-scale changes to the countryside, resulting from inadequate pollination of certain plants.

Some bumblebees native to North America are also vanishing, such as Bombus balteatus, Bombus terricola, Bombus affinis, and Bombus occidentalis, and one, Bombus franklini, may be extinct.
In South America, Bombus bellicosus was extirpated in the northern limit of its distribution range, probably due to intense land use and climate change effects.

KBMMW PROFESSIONAL AND ENTERPRISE EDITION V. 5.14 RELEASED!

- **RAD Studio XE5 to 10.4.1 Sydney supported**
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OSX client and server support
- Native high performance 100% developer defined application server
- Full support for centralized and distributed load balancing and failover
- Advanced ORM/OPF support including support of existing databases
- Advanced logging support
- Advanced configuration framework
- Advanced scheduling support for easy access to multithread programming
- Advanced smart service and clients for very easy publication of functionality
- High quality random functions.
- High quality pronouncable password generators.
- High performance LZ4 and Jpeg compression
- Complete object notation framework including full support for YAML, BSON, Messagepack, JSON and XML
- Advanced object and value marshalling to and from YAML, BSON, Messagepack, JSON and XML
- High performance native TCP transport support
- High performance HTTPSys transport for Windows.
- CORS support in REST/HTML services.
- Native PHP, Java, OCX, ANSI C, C#, Apache Flex client support!

kbmMemTable is the fastest and most feature rich in memory table for Embarcadero products.

- **Easily supports large datasets with millions of records**
- **Easy data streaming support**
- **Optional to use native SQL engine**
- **Supports nested transactions and undo**
- **Native and fast build in M/D, aggregation/grouping, range selection features**
- **Advanced indexing features for extreme performance**

- ♦ **New HTTP and kbmMW server status message handling support in Smart services**
- ♦ **New support for TkbmMWGenericMagneticStripeReaderHID and TkbmMWGenericBarcodeReaderHID**
- ♦ **New support for pivot based counters in the ORM**
- ♦ **New support for transport native file sending for HTTP.Sys transport**
- ♦ **New support for automatic GZIP compression of responses from HTTP Smart services**
- ♦ **Several feature improvements and fixes.**
- ♦ **More features improvements and fixes.**

Please visit
`http://www.components4developers.com`
for more information about kbmMW

- High speed, unified database access (35+ supported database APIs) with connection pooling, metadata and data caching on all tiers
- Multi head access to the application server, via REST/AJAX, native binary, Publish/Subscribe, SOAP, XML, RTMP from web browsers, embedded devices, linked application servers, PCs, mobile devices, Java systems and many more clients
- Complete support for hosting FastCGI based applications (PHP/Ruby/Perl/Python typically)
- Native complete AMQP 0.91 support (Advanced Message Queuing Protocol)
- Complete end 2 end secure brandable Remote Desktop with near realtime HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- Bundling kbmMemTable Professional which is the fastest and most feature rich in memory table for Embarcadero products.

COMPONENTS 4 DEVELOPERS

EESB, SOA,MoM, EAI TOOLS FOR INTELLIGENT SOLUTIONS. kbmMW IS THE PREMIERE N-TIER PRODUCT FOR DELPHI / C++BUILDER