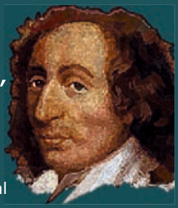
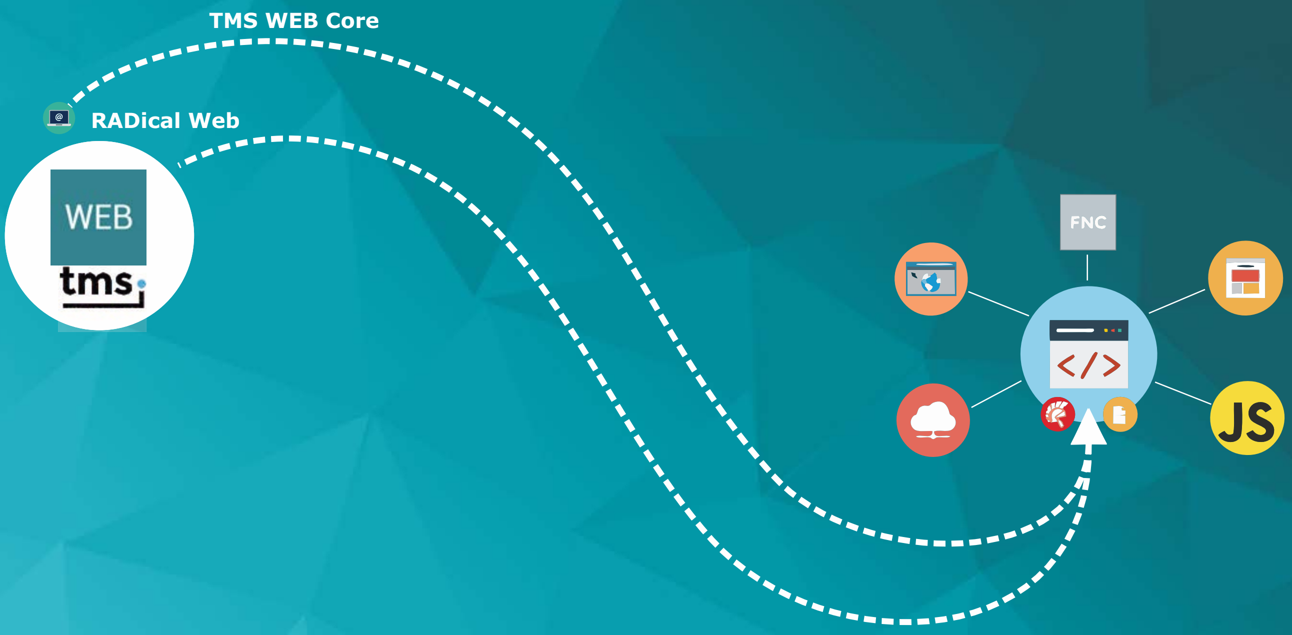
FOR DELPHI, LAZARUS, AND PASCAL RELATED LANGUAGES / WEB APPS, INTERNET, ANDROID, IOS, MAC, WINDOWS & LINUX



BLAISE PASCAL MAGAZINE 69/70



barnsten

• development tools
• consultancy
• components
• training
• hands-on workshops
• support

LAST OPPORTUNITY

TO UPGRADE FROM ANY PREVIOUS VERSION

of

Delphi / C++Builder/Rad Studio

AMNESTY

for everybody till 31 march 2018

After 31 march 2018 there are no upgrade products anymore.

ACT NOW!

https://www.barnsten.com/default/promotions/amnesty

BLAISE PASCAL MAGAZINE 67/68

DELPHI, LAZARUS, SMARTMOBILE STUDIO, AND PASCAL RELATED LANGUAGES FOR ANDROID, IOS, MAC, WINDOWS & LINUX

CONTENTS

ARTICLES

ADVERTISERS

Publisher: Foundation for Supporting the Pascal Programming Language
in collaboration with the Dutch Pascal User Group (Pascal Gebruikers Groep)
© Stichting Ondersteuning Programmeertaal Pascal

Stephen Ball
http://delphiaball.co.uk
@DelphiABall

Miguel Bebensee
mbebensee@ibexpert.biz
http://devstructor.com

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
www.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

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
www.romplesoft.de

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

Siegfried Zuhr
siegfried @ zuhr.nl

Editor - in - chief
Detlef D. Overbeek, Netherlands Tel.: +31 (0)30 890.66.44 / 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.

Member and donator of WIKIPEDIA

Subscriptions (2017 prices)

	Internat. excl. VAT	Internat. incl. VAT	Shipment
Printed Normal Issue 44 pages	€ 100	€ 106,50	€ 75
Printed Extended Issue 80 pages	€ 150	€ 159	€ 100
Electronic Download Issue 80 pages	€ 50	€ 60,50	——

Printed magazine edition
10 issues per annum, 44-page Delphi-only section: € 100.-- This includes postage, VAT at 6 % and all code and programs accompanying the articles.
Excluding postage the 44-page edition is € 75.-- per annum.
10 issues per annum 80-page Delphi + Lazarus sections: € 150 plus € 100 for postage.
Digital magazine edition (PDF format)
10 issues per annum 80-page Delphi + Lazarus sections: € 50.-- (excluding VAt at 21%).

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

barnsten

- development tools
- consultancy
- components
- training
- hands-on workshops
- support

Celebrate Tokyo Release 3 with Exclusive Promo

We're celebrating the latest release of RAD Studio, Delphi, and C++Builder:
Tokyo Release 3!
With powerful new features, now's a great time to start using
the best visual IDE for Delphi and C++Builder.
Join the celebration with this exclusive promotion worth up to 20% Off.

UP TO 20% OFF THE BEST TOOLS
FREE 12 MONTHS OF UPDATE SUBSCRIPTION

What Can You Do with Release 3?

Delphi and C++Builder Professional Editions Now Include Mobile Support
(Previously available in Enterprise, Architect or as additional purchase)
Develop for Windows and macOS Design, code, test, and publish robust cross-platform applications (plus build Linux apps in Delphi!)
Design Stunning Applications
Visual Design is a breeze with our IDE, hundreds of components, and **HighDPI** support for VCL.
Modernize Apps and Improve Code Readability.
We've added C++ Rename Refactoring to help you quickly and safely update identifiers throughout your project. **Faster Builds.** Enhancements for CMake Command Line Support allows for fast parallel builds. **Expanded RAD Server Support for Ext JS** Rapidly build services based applications, now with support for javascript web clients.

10.2.3 Special Offer: Professional versions now with the mobile Pack for FREE and

Save 10% on Professional Editions of RAD Studio, Delphi, C++Builder.
Save 15% on Enterprise Editions
Save 20% on Architect Editions.

Offers end 31 March 2018.

https://www.barnsten.com/default/promotions

TMS WEB CORE AND RADICAL WEB INTRODUCTION

AUTHORS: DETLEF OVERBEEK/HOLGER FLICK - CORRECTOR HOWARD PAGE CLARK

starter expert

It finally arrived - on Valentines day.
The web framework we always wanted.

We now have a framework which is capable of what I had been looking for since I had a meeting in Paris with an Embarcadero official, about eight years ago. We talked about Delphi, the future what was missing in Delphi and how to move ahead.

I had already met with Michael van Canneyt - he is the author of many articles and a large book about Pascal-Lazarus and told him the web interface I wanted was missing from both Delphi and Lazarus.
He agreed and said he had the same idea and showed me software similar to what he would like to create for Pascal, (Morfik) but by that time it was in its infancy. He had been dreaming about realising this, but it was a huge task, it would take years, he predicted.
It did. Altogether 10 years.

About two years ago, Michael and I decided to ask others to help develop the Web Suite we had started.
One Developer of the Pascal Lazarus team was Mattias Gärtner (an IDE development expert in the Lazarus team) and I made contact and asked him if he was willing to help us. Michael and Mattias spoke in two ways the same language, (being Belgian and German) so it was a very successful contact and especially since these two guys had fallen into the bucket – like Obelix in his "strengthening Bouillon" the project suddenly got an enormous boost.
Michael had already done a hellish task, so now the project exploded. We decided to work together closely, and as fast as we possibly could. Once we had done that, we realized it would be necessary to find a third party for the Delphi aspect.
It was years ago that I had first met Bruno Fierens and encountered TMS, and, I knew immediately that TMS Software would be the only candidate... Does coincidence exist?
So I went to Bruno and asked him if he could possibly do that for us. This article is part of the outcome.
I have been knocked sideways by the quality and versatility of the product Bruno has developed.
He created a web framework which in my view is as important as the invention of Delphi itself.

Of course we have a roadmap and still lots of things need to be done: further functionality will be added to the compiler and various annoyances must be ironed out…
For the short term now there will be a Beta version that enables you to play around with the components and their capabilities. The trial will be available not only for Delphi but also for Lazarus.
Quite soon now we will present a fully working and well-tested framework of components that you can buy in various combinations with either basic or more comprehensive functionality.

1. **TMS WEB Core:**
   295 EUR introductory price
   `https://www.tmssoftware.com/site/tmswebcore.asp`
2. **TMS WEB Studio**
   (TMS WEB Core
   + TMS FNC controls
   + TMS XData) :
   595 EUR introductory price

https://www.tmssoftware.com/site/tmswebcore.asp
https://www.tmssoftware.com/site/tmsfncuipack.asp
https://www.tmssoftware.com/site/xdata.asp

The sheer number of components you will be able to use for the web is enormous. At the end of this article we append several examples designed for you to learn from.. Many of the examples have complete working project code, so once you have the components available you can immediately create any website you want. All you need to understand is: Object Pascal. We aim to help you understand this new framework from the inside, and offer an outline of its architecture and working. You will be able to appreciate the tremendous possibilities this TMS WEB FRAMEWORK has. Hopefully you will be as enthusiastic as we are, and agree this is next best thing that ever happened to Delphi. I have had guidance and great help from Holger Flick and you will find his explanations and diagrams very enlightening. I want to commend him for his excellent help, without which I would not have been able to understand the inner workings of this new framework myself.

WHAT DOES TMS WEB MEAN?

TMS Web is a comprehensive web application development framework. It requires only a good knowledge of Delphi or Object Pascal.
You do not have to use any other programming languages. All the functionality you need is encapsulated in components, designed in an object oriented way using Pascal classes.
Especially you do NOT require any knowledge of HTML or JavaScript.

You will find that, a web-app created with "TMS Web" interacts amazingly well with available frameworks or can be visually enhanced through traditional web design using HTML, CSS and other JavaScript frameworks - if you wish. Designs created by and for enterprises can be implemented or extended endlessly.

APPLICATION MODEL:

Web applications are based on the **SPA** (*Single Page Application*) paradigm, a very sophisticated way of application modelling.
(*Wiki: A single-page application (SPA) is a web application or web site that interacts with the user by dynamically rewriting the current page rather than loading entire new pages from a server. This approach avoids interruption of the user experience between successive pages, making the application behave more like a desktop application. In an SPA, either all necessary code – HTML, JavaScript, and CSS – is retrieved with a single page load, or the appropriate resources are dynamically loaded and added to the page as necessary, usually in response to user actions. The page does not reload at any point in the process, nor does control transfer to another page, although the location hash or the HTML5 History API can be used to provide the perception and navigability of separate logical pages in the application. Interaction with the single page application often involves dynamic communication with the web server behind the scenes.*)

The SPA paradigm offers an obvious benefit to the end user, who starts the application simply by opening a single **HTML** document in her browser.
The principal drawback to the **SPA** model is the **Fat Client** it produces, often with an extremely high load effort. However, this can be offset by intelligent modularisation. To do this requires configuring the generation of the web-server application appropriately.
So as "Johan Cruiff" used to say: every downside has it's benefits.

COMPONENTS:

Components are the core ingredients for development with Delphi and that is why they are the focus for the development of Web-apps with **TMS Web**. At design time all the components you need are dropped on your form and configured via the Object Inspector, exactly as you are used to.
As with the **VCL, TMS Web** includes components which are visual (*e. g. a label*) and non-visual (*e. g. a timer*).

DEBUGGING:

You can debug your application through the Delphi IDE as well as with the Web Browser. **Break points** and the evaluation of variables line by line are supported without any limitations. This differs markedly from most other web development tools. Other tools don't usually support debugging of the running application in anything other than Javascript, forcing you to evaluate JavaScript source in the browser. This may be painful and unfamiliar for **Delphi** developers. Simply said it's essential that this works as it does in **TMS Web**. The uninterrupted use of Object Pascal means that errors are recognised sooner, and solutions applied more quickly using Pascal. If you would like to analyse the ongoing process in the web browser step by step: it is of course possible without any problem. This step is of course vital as soon as the application is tethered to existing JavaScript solutions.

JAVASCRIPT COMPONENTS:

If you design using existing JavaScript framework components you own, they will be shown (without any preview) as frame placeholders in the form designer. This is consistent since there is no representation for them in Delphi. Nevertheless, you can still set events and properties for these components via the **Object Inspector**, and avoid **JavaScript** altogether (*if you wish*). The list of supported components and frameworks is growing.

JAVASCRIPT FRAMEWORKS:

You can even integrate a JavaScript framework into your application that has no visual components at all. For example you can incorporate design styles by **Bootstrap** to Standard components from **TMS Web**.
The excellent separation between application logic and web design interface built into **TMS Web** is characteristic of the high quality of this new framework.

INTEGRATION INTO THE IDE

TMS Web provides wizards to get you going quickly with app development, along with numerous dialogs related to the creation, set-up and configuration of **TMS Web** apps. It is also very important to emphasize the seamless integration with all the existing IDE tools which write and navigate in the code without any restriction.
Provided you have the **TMS Web** sources you can even extend the framework with third party libraries with your own components.

Screenshots in this article:
All the screenshots shown were made using **TMS Web** and **Embarcadero Rad Studio 10.2.2 Tokyo**. The presentation in other Delphi versions will of course be different, since Embarcadero has modernized all the IDE icons. So that might seem a bit strange for seasoned developers who don't use the very latest Delphi version.

BASICS

We want to explain some of the basics of the framework . But before that we will give you the system requirements and then explain the installation.
The most popular browsers are listed below, showing you which version is supported.

Product	Recommended Version
Internet Explorer	11
Google Chrome	
Mozilla Firefox	
Opera	
Apple Safari	(only available for Apple products)
Microsoft Edge	(Windows 10)

So it is wise to use one of the listed browsers to make sure that there will be no production issues to try to guarantee the compatibility. TMS Web applications can be used in any HTML5- compatible web browser.
But the **TMS** applications have been tested in all the browsers listed above.

INSTALLATION

Like all products from TMS Software it will be shipped with an installation program that comes with a wizard that will guide you through the process. You need to know that the TMS Web framework installs using a two-phase process - first the basic system and as a second part the installation which is dependent on the Delphi Version. So if you want to use several Delphi versions it will be done separately for all these versions.

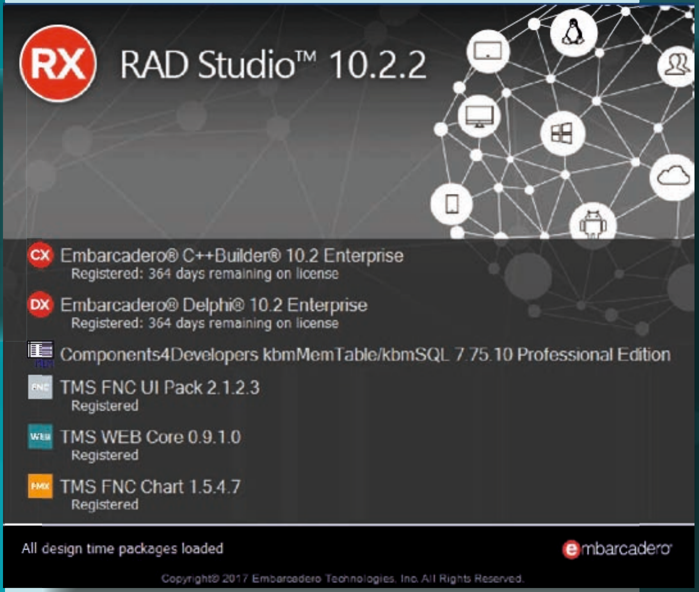
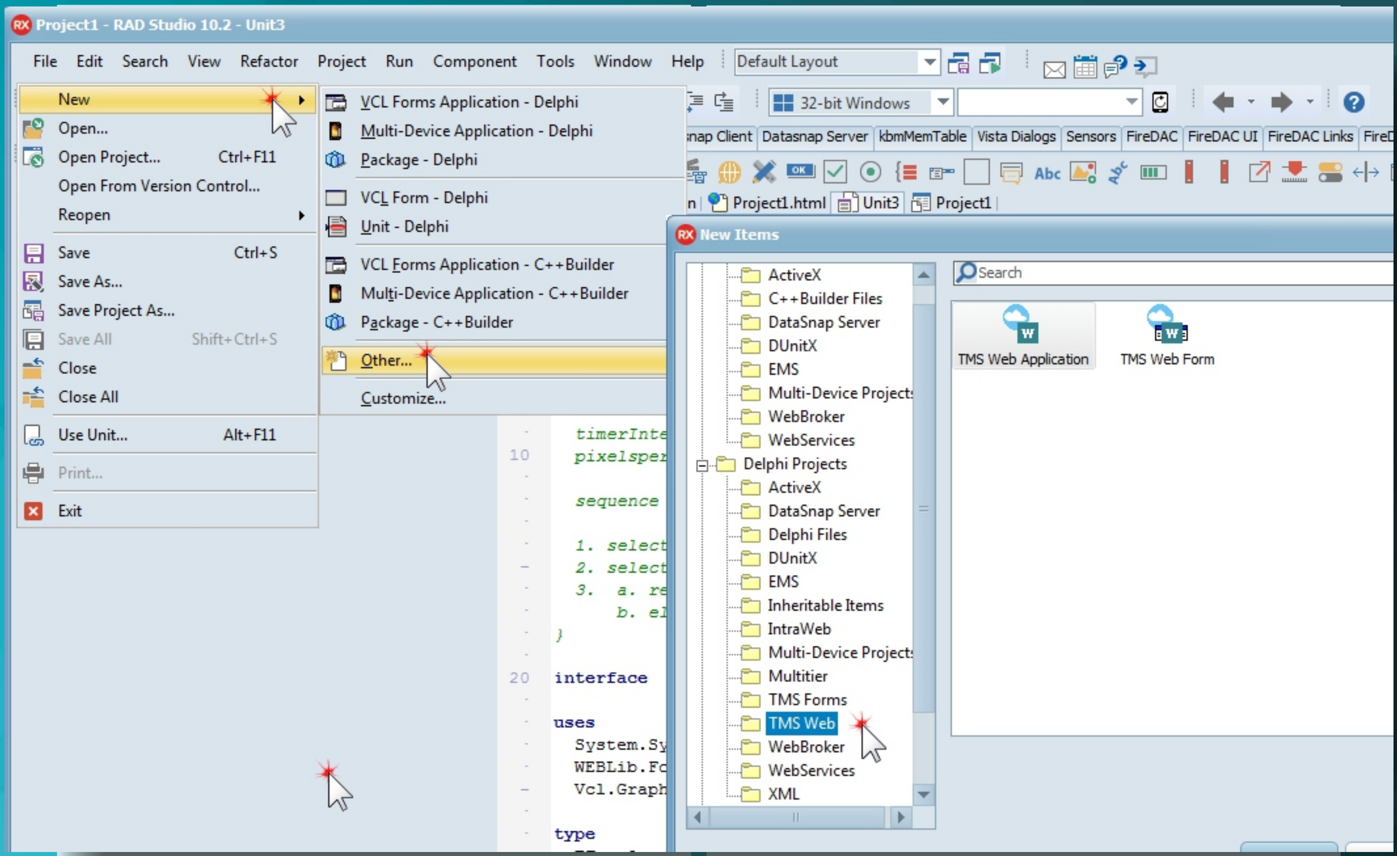

RAD Studio™ 10.2.2
Embarcadero® C++Builder® 10.2 Enterprise
Registered: 364 days remaining on license
Embarcadero® Delphi® 10.2 Enterprise
Registered: 364 days remaining on license
Components4Developers kbmMemTable/kbmSQL 7.75.10 Professional Edition
TMS FNC UI Pack 2.1.2.3
Registered
TMS WEB Core 0.9.1.0
Registered
TMS FNC Chart 1.5.4.7
Registered
All design time packages loaded
embarcadero
Copyright© 2017 Embarcadero Technologies, Inc. All Rights Reserved.

Figure 1. The opening splash screen.

The suite installs very smoothly. I tried it under Win7 and Win 10. Simple quick and easy. I did not need to ask for any help, as can be the case for installing a component suite.

Note 1:
If you have a **version with source code** the source code will be recompiled during the installation process and integrated in the IDE(Integrated Development Environment).
If you have a **version without source code** the pre-compiled "dcu"forms will be integrated.
If you have a so-called
"TMS All Access Subscription" you can make the installation by simply calling The TMS Subscription Manager. In that case you are always provided the latest version.



```
  timerInte
  pixelsper

  sequence

  1. select
  2. select
  3.   a. re
       b. el
}

interface

uses
  System.Sy
  WEBLib.Fo
  Vcl.Graph

type
```

Figure 2: This post-installation screenshot shows where the new TMS Web Form and Web Application can be found in the Delphi IDE.

THE FIRST PROJECT

Once installed, you have a fully-integrated IDE extension for generating code for Web applications. Whenever you start using something new it's useful to start with a very simple basic program that usually shows the text "Hello World". The IDE is extended by **TMS Web** and you can find that by **File → New → Other** and the wizard will show a list where you find the category 'Delphi Projects' ' **TMS Web**' . Just click on **TMS Web Application** (*See Figure 2*). There is nothing else to do. All application settings will be passed from the basic settings of TMS Web and you do not need to copy settings each time you want a new project for the web. Configurations we will discuss later.

The integration of the web design is done where it belongs: In HTML or CSS.

THE PROJECT SETTINGS

The project consists of a form, the **Main Form** in the unit `Unit1.pas` and the project file `Project1.dpr`. You will find an additional file: `Project.html`. This file is the **HTML** document for the web application. This file exists firstly as a reference for the web app which is created with **TMS Web**. You usually do not need to make many changes to it, however a web designer may wish to tweak or edit the overall design using **HTML.**

Each TMS Web Form consists of three Files: the `.pas`-file – including the code, the `.dfm`-file - containing the layout for the form and a **HTML** file. Editing this HTML form file lets you make possibly far-reaching changes to the form's design.can **integrate** other frameworks.

Note 2:
The TMS Web philosophy places all application logic in the framework's components.
The integration of the web design is done where it belongs: In **HTML** or **CSS.**
Many Web Frameworks trip themselves up by trying to integrate **HTML** or even **JavaScript** into the Application Component files. The code of these files is not easy to locate and maintain This because there is no clear separation between the Application logic and the design.

Now here is the overview of the **Project File** of the App: **(project1.dpr)**

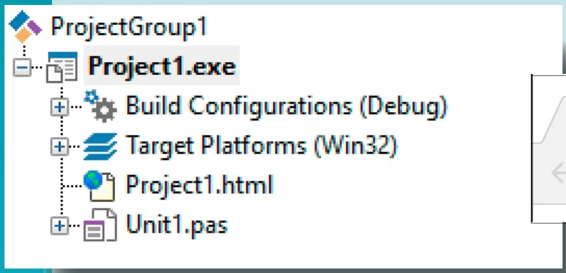


Figure3: the project group

```
program Project1;

uses
    Vcl.Forms,
    WEBLib.Runner,
    Unit1 in 'Unit1.pas'          {Form1: TWebForm} {*.html};

{$R *.res}

begin
    Application.Initialize;
    Application.MainFormOnTaskbar := True;
    Application.CreateForm(TForm1, Form1);
    Application.Run;
    TTMSWebRunner.Execute('http://localhost:8000/$(ProjectName)', tbnDefault);
end.
```

Code Listing 1

Code Listing 1

At first sight there aren't many differences from a classic form-based application, except that the uses clause includes the **WebLib.Runner** unit. As usual **TApplication** will be initialized but a call is made to the **WebRunner** - complete with a URL. The static **Execute Method** of the Class **TTMSWebRunner** opens a Browser

- conditioned by the constant tbnDefault for the standard workstation browser,
- with the specified address as the first parameter.

A place-holder for the name of the project **$(ProjectName)** is also used. Place-holders are all marked by a **$** and begin and end with brackets ().

So it is clear now that **TMS Web Apps** are actually very normal applications in Delphi. You will be running an executable (.exe). However, the Web app will not show its main form on your desktop.

If you run the program by pressing **Ctrl+Shift+F9** the program will run without Debugging and your default browser (*whatever your default browser is*) will show the app's output.

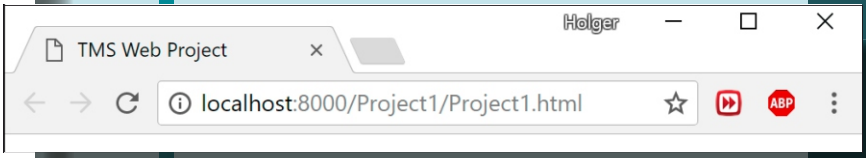


Figure 4 Here Google is the default browser

Alternatively **tbnDefault** can be replaced by one of the following constants:

Constant	Web Browser
tbnDefault	Standard Web Browser
tbnChrome	Google Chrome
tbnFirefox	Mozilla Firefox
tbnEdge	Microsoft Edge
tbnIExplore	Microsoft Internet Explorer
tbnOpera	Opera

So you can force Microsoft Edge to run the app by specifying the constant tbnEdge.

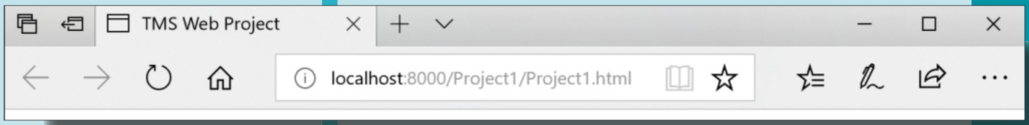


Figure 5 Microsoft Edge

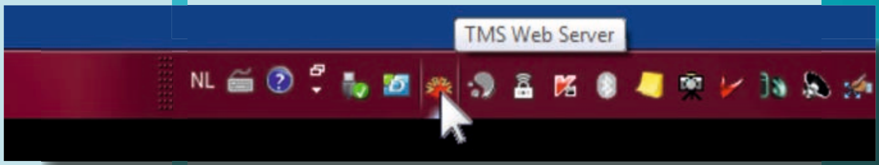


Figure 6: TMS Web Server

The executable file generated on our development machine is merely a launcher for the browser, which starts a **local Web Server** and navigates to the address so the application will be started. By default the internal web server is bound to **`localhost`** with **`port 8000`**.

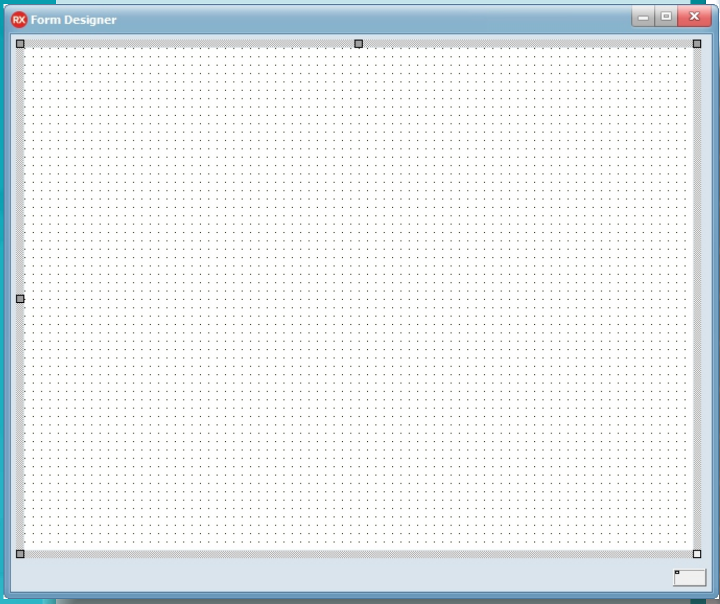
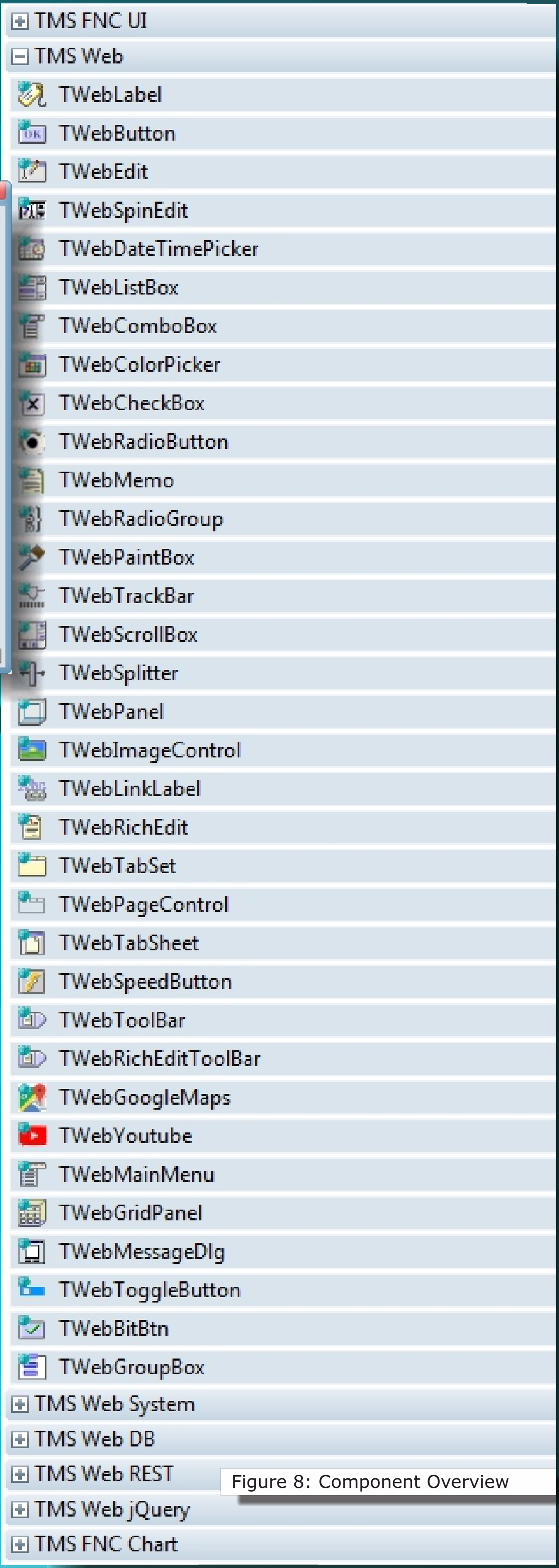


Figure 7: The Form Designer looks like one of the Data Modules

So far the application has no content so that we cannot yet verify the last step that the app will be started. So let's change that. Open the Main Form **`Unit1.pas`** in the **Form Designer** of the **IDE.** The **Form Designer** looks like a **Data Module** and is clearly distinct from the **VCL** or **FireMonkey** form. All visual tools for the use of Components in the designer are now available.

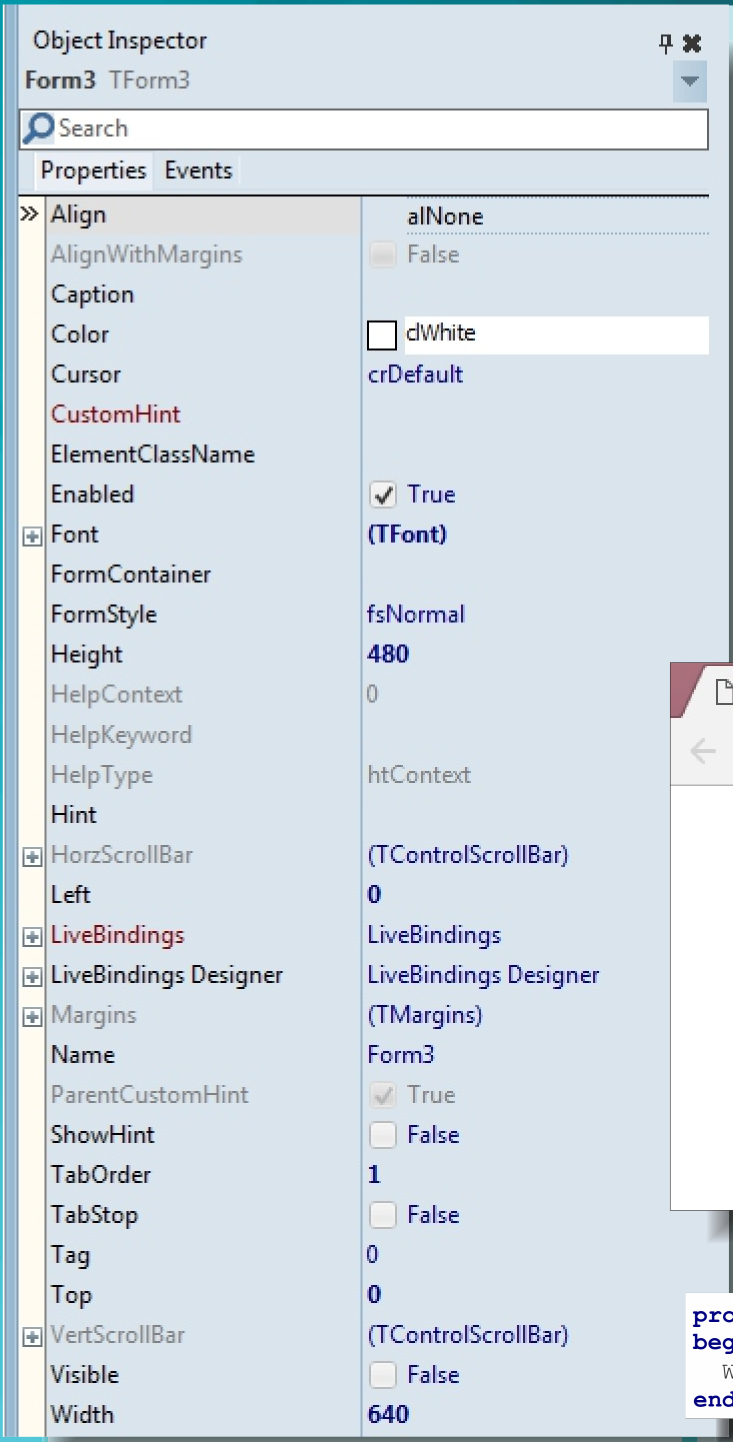
The component overview (*see the column to the right*) is now limited to those components that can be used in conjunction with **TMS Web**. This form inherits not from **`TForm`**, but from **`TWebForm`**. Many **`TWebForm`** properties have the identical name and functionality that their **`TForm`** counterpart shares. If you ever worked with other frameworks for the web to create apps you will appreciate the events of the forms immediately. It's like coming home. With TMS Web you will at first see no difference from creating a **VCL** form application at all. You can instinctively find the component you need: **`TWebLabel`** duplicates **`TLabel's`** functionality. **`TWebEdit`** is the equivalent of **`TEdit.`** Every **VCL** developer will be able to identify and use Standard-page components straight away. You will be pleased to know there are very few limitations on Web-component properties and events compared to their VCL analogues.



⊞ TMS FNC UI
⊟ TMS Web
- TWebLabel
- TWebButton
- TWebEdit
- TWebSpinEdit
- TWebDateTimePicker
- TWebListBox
- TWebComboBox
- TWebColorPicker
- TWebCheckBox
- TWebRadioButton
- TWebMemo
- TWebRadioGroup
- TWebPaintBox
- TWebTrackBar
- TWebScrollBox
- TWebSplitter
- TWebPanel
- TWebImageControl
- TWebLinkLabel
- TWebRichEdit
- TWebTabSet
- TWebPageControl
- TWebTabSheet
- TWebSpeedButton
- TWebToolBar
- TWebRichEditToolBar
- TWebGoogleMaps
- TWebYoutube
- TWebMainMenu
- TWebGridPanel
- TWebMessageDlg
- TWebToggleButton
- TWebBitBtn
- TWebGroupBox

⊞ TMS Web System
⊞ TMS Web DB
⊞ TMS Web REST
⊞ TMS Web jQuery
⊞ TMS FNC Chart

Figure 8: Component Overview



Object Inspector

Form3 TForm3

Search

Properties Events

Align	alNone
AlignWithMargins	False
Caption	
Color	clWhite
Cursor	crDefault
CustomHint	
ElementClassName	
Enabled	✓ True
Font	(TFont)
FormContainer	
FormStyle	fsNormal
Height	480
HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
HorzScrollBar	(TControlScrollBar)
Left	0
LiveBindings	LiveBindings
LiveBindings Designer	LiveBindings Designer
Margins	(TMargins)
Name	Form3
ParentCustomHint	✓ True
ShowHint	False
TabOrder	1
TabStop	False
Tag	0
Top	0
VertScrollBar	(TControlScrollBar)
Visible	False
Width	640

Figure 9: the Object Inspector

Double-clicking the button generates an **OnClick** handler as you would expect, and you are placed in the editor to complete the generated code skeleton:

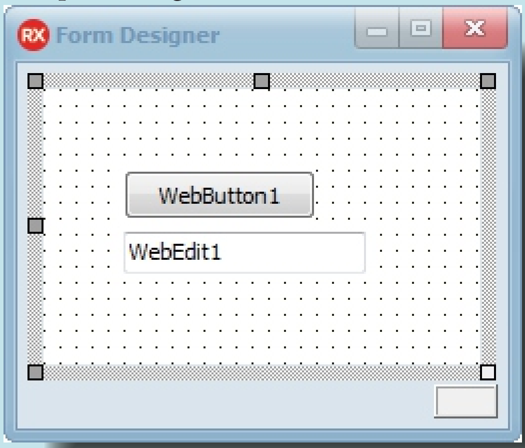


Figure 10: the Form and components

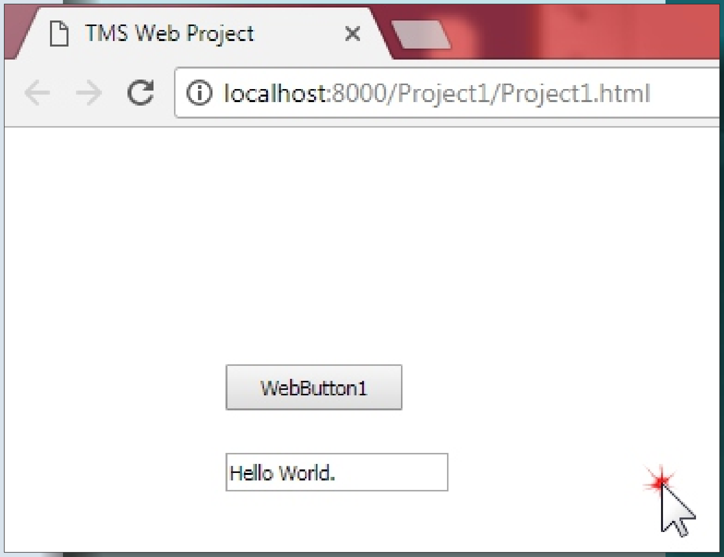


Figure 11: the Result after compilation

```
procedure TForm1.WebButton1Click(Sender: TObject);
begin
  WebEdit1.Text := 'Hello World.';
end;
```
Code Listing 2

For example, you can use **Anchors** and **Align** to put your components in place. You also can use the panel component to group your components. The adoption of so-called "Responsive Design" means it is supremely easy to use these components and set their properties.
Let’s go for the simple „Hello World“ app.
Drop a **TWebEdit** and a **TWebButton** component on the form.

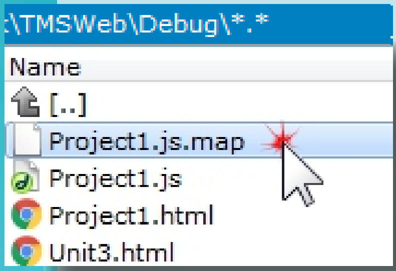
Again: no difference at all between writing a **Web App** or a **Desktop VCL** application.
The WebEdit1's Text property gets the value ‘Hello World’.
Of course you could also have renamed the component through the property Name.
After starting the app we can see the components are shown. One click on the button provides the desired result.

Let's see what files the compiler has generated. In addition to the **.exe** there is a new **.ini** file. Its contents are as follows:

```
[Paths]
HtmlPath=C:\tex\TMSWebBook\demos\minimal\.\TMSWeb\Debug
HtmlFile=Project1.html
DefaultURL=http://localhost:8000/Project1
```
Code Listing 3

The ini file contains the information for the app in the exe file to find the Web app. This is only meant to start the desired web browser. So where is the Web Application? By default a "TMS Web" directory has been created in a subdirectory. You can of course alter that according to your own wishes.
The web application consists of the **HTML** files for the forms and a **JavaScript** Form.
Opening the **HTML** file on the web server runs the corresponding **JavaScript** file. Furthermore the meta information for the debugger will be generated in a **.map** file. Where is that found?

Here the strength of TMS Web is shown: all the RTL functions used in the VCL work well in web apps thanks to TMS Web with very few limitations also in a Web App.


```
:\TMSWeb\Debug\*.*
Name
[..]
Project1.js.map
Project1.js
Project1.html
Unit3.html
```

By default it is placed in the project's **TMSWeb\Debug** directory. You can of course alter that.

Figure 12: Location of the . map file

Note 3:
If you place several apps on the same server, their overall size can be reduced by creating separate JavaScript files for each unit. I actually hardly needed the manual. Compared to using other web solutions, I quickly found an enormous improvement in speed and ease of working using TMS Web. I actually hardly needed the manual. To create web applications in this way is To create web applications in this way is for any developer that knows the VCL completely intuitive.

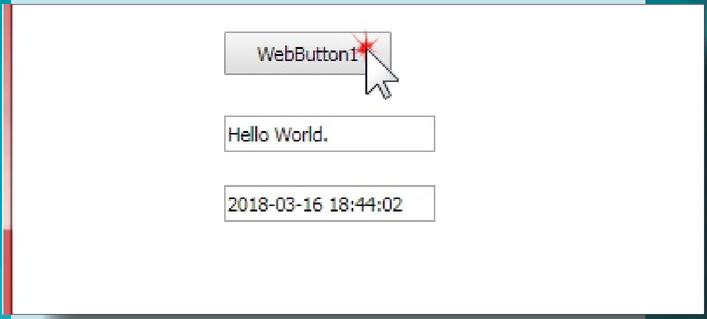
An further example:
Just to show how true the last words are we'll show two further examples. Let' s expand the form with another component of type **TWebEdit** and add to the **OnClick** of the Button as follows:

```
procedure TForm1.WebButton1Click(Sender: TObject);
begin
  WebEdit1.Text := 'Hello World.';
  WebEdit2.Text := DateTimeToStr(Now); // <-- new!
end;
```
Code Listing 3

We will naively try to use familiar, well-known Delphi RTL functions. Especially Date and Time functions are by using the Web a very popular subject for discussions which many developers love to hate. A click on the Button starts the WebApp and indeed shows what we expected. So our naivety and hope are fully successful. Here the strength of **TMS Web** is shown: all the **RTL** functions used in the **VCL** work well in web apps thanks to **TMS Web** with very few limitations also in a **Web App.**

LIVE CLOCK

So lets try the next step: we will create a little clock that is automatically updated in the background.



```
Hello World.
2018-03-16 18:44:02
```

Figure 13: Result

We want the correct time to be shown by being updated continuously. But we also want this to happen without sending a request to the server. Our goal is an implementation that is as easy as possible and that we are used to when we create a desktop application. We will drop two further WebButtons and call them **btnTimerStart** and **btnTimerStop**.
The name on the caption should be altered accordingly. You will find under TMS Web System a component called TWebTimer, which is the equivalent of TTimer known from the VCL. The timer's Interval property defaults to 1000, equivalent to 1 second.

The events for the buttons to start and stop the timer are implemented as follows:

```
procedure TForm1.btnTimerStartClick(Sender: TObject);
begin
  WebTimer1.Enabled := true;
end;

procedure TForm1.btnTimerStopClick(Sender: TObject);
begin
  WebTimer1.Enabled := false;
end;
```

Code Listing 5

The event **OnTimer** should show the actual time in on one of the edit fields

```
procedure TForm1.WebTimer1Timer(Sender: TObject);
begin
   WebEdit2.Text := TimeToStr( Now );
end;
```

Code Listing 6

That's all folks. We now can start the app.
The result is impressive.
A simple click on the start button shows the time immediately. Each second the clock is updated. Clicking „Stop“ causes the clock updates to cease. Anyone who ever tried to implement such a simple clock by writing a **JavaScript** implementation will appreciate what fantastic results **TMS Web** can produce in a fraction of the time.

FUNCTIONALITY

Now that we've seen TMS Web in action for the first time, it's time to move on and explain exactly what happens when Delphi source code becomes a runnable web application. You need to understand this so you can see how the few limitations of TMS Web arise. Most restrictions result from restrictions imposed by the target platform and the web itself, not from design shortcomings in **TMS Web.**

ARCHITECTURE

On the basis of the diagram in Figure 3.1, we can comprehend how the transforming of the source code of a JavaScript-dominated web application works. Any application will be built from several interdependent abstraction layers that encapsulate the desired layer functionality. You've already seen that you can use both **FNC** (Framework Neutral Components) standard components, and elements from **RTL.** You also can use visual components from other JavaScript frameworks. We now map the various components of a **TMS Web** application to the layers and explain how the compiler moves from layer to layer to build the final application.

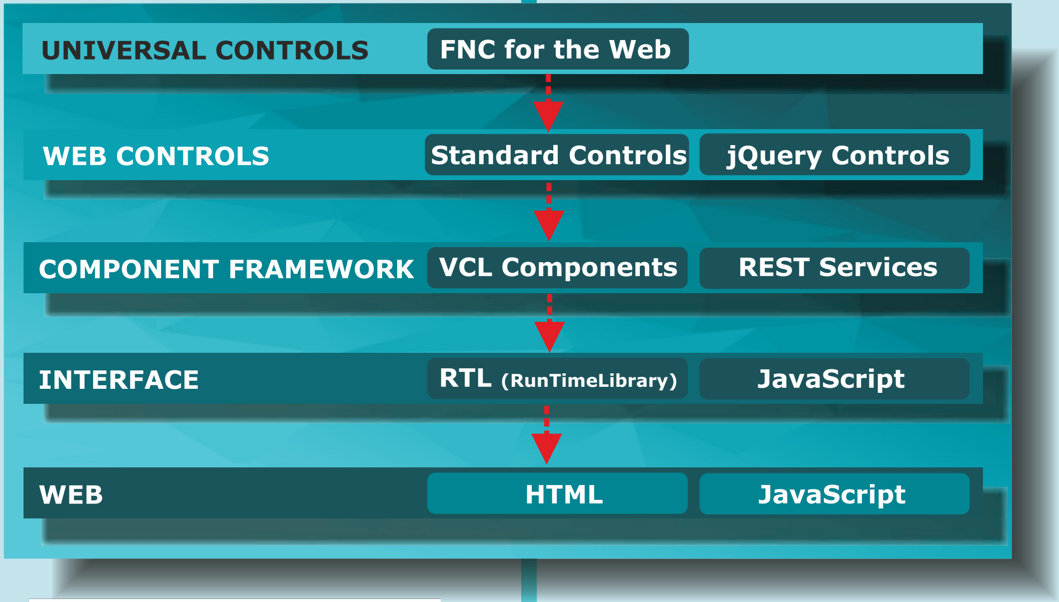


Figure 14: Architecture of TMS Web.

The final product on the web (*bottom layer*) is generated step by step.

For this purpose, e.g. **FNC components** (*top layer*) are mapped to **Web Standard** components, which in turn map to **VCL Standard** components.

Finally, the **Pascal2JavaScript** compiler forms the necessary components for a Web Application consisting of **HTML** and **JavaScript.**

- The top layer, at the highest level of abstraction, uses universal components which are independent of any implementing framework. The developer doesn't have to worry whether the components are used in a **VCL** application, **FireMonkey** application or Web Application. Even for the **Linux** target, enabled now via **fmxlinux** or via **LCL** and **Lazarus,** the **TMS FNC** controls can be used, no adjustments are necessary. The platform is perfectly abstracted. **TMS** has given this type of component the name **'FNC':** Framework Neutral Components.
  In **TMS Web Applications** these components are shown on the next, deeper abstraction layer. This means a framework neutral input field **`(TTMSFNCEdit)`** will be converted to a standard web component input field.

- The web components of TMS Web form the next layer. Here you can find the default components that start with **TWeb.**
  Examples are **`TwebLabel`, `TWebEdit`** etc. These Web components are broken down to components from the **VCL.** The web component abstraction layer ensures that the only properties and events available will be those appropriate to a web application. Any visual components used from other **JavaScript** frameworks must be wrapped as distinct components which can then be mapped to **VCL** equivalents.

- The next layer is the interface between the Delphi layer and the web Level.
  It also provides the RTL and JavaScript functions.
  That means here you can also find the implementation that ensures that the RTL will become **'Web - able'.**
  Of course, here are also internal JavaScript elements to be found for integration in the web applications.

- The bottom layer represents the end product: an HTML document containing one or more JavaScript file(s) with the web application. The visual components used are **HTML** elements that at this moment can be interpreted and displayed by any **HTML5** -able Web Browser.

For example a **`TWebEdit`** is converted into an **`<input>`** with parameters that represent the defined properties and events. Of course wrapped **JavaScript** components, e.g. **jQuery** Components, are replaced by the corresponding **JavaScript** components.

In summary it should be noted that TMS Web using the above 'Transformation chain' for web applications is an abstract, completely comprehensible representation of an object-oriented high-level language implemented concretely as a web application. How to implement this? From among the various object oriented possibilities open to **TMS,** they opted for inheritance abstraction.
The abstraction is based on inheritance of the respective classes from a **VCL** base class. All visual components in the upper layers **FNC** or the Web standard components thus descend from a base class in the **VCL.**

We now want to look at 3 classes as an example, to consolidate our understanding of the architecture of the **TMS Web** framework.

TWebForm
All components are arranged on a form.
The form is represented by the class **`TWebForm.`** This component is found in the standard components and is thus according to our model is mapped as a descendant of the VCL **`TCustomForm`** component.

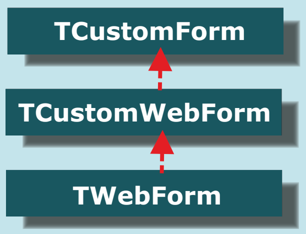


Figure 15

TWebEdit
The input field **`TWebEdit`** is also derived directly from the corresponding **VCL** components **`TCustomEdit`** forms the base class.

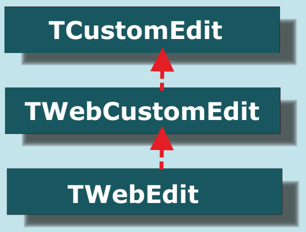


Figure 16

`TTMSFNCEdit` - an FNC component
Finally, an example of an FNC component. Analogous to the standard components is also derived here from VCL components. Important is the understanding that no web standard component is stored. It is directly derived from the VCL - without reference to the web.

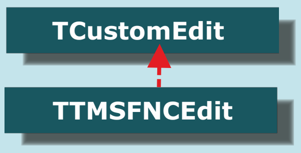


Figure 17

BUILDING BLOCKS

TMS Web has a modular structure. It consists of a basic building block designated as the '**TMS Web Core**'. Based on this core, additional components can be used. Which building blocks are available to the developer depends on the licensing model chosen, and which blocks that licence provides.

Figure 18 overleaf is a graphical representation of the way the different parts morph together and how modular it is. *However, it does not show the available modules as JavaScript has more things than are shown in the illustration.*

TMS WEB CORE

TMS Web Core is the foundation for all other modules. It contains both basic components and everything needed for full IDE integration, together with the special transpiler needed for the web applications this package creates.

TMS FNC FOR THE WEB

- **FNC** components for the Web.
With this package a variety of components from the **FNC UI** package come available for **TMS** Web. These types of components are of these components do not differ in principle from other **Delphi** frameworks.
You can use these FNC components on the Web - just as you might do for **VCL** or **FireMonkey** applications with **FNC** components normally.

jQuery COMPONENT

Components from the **jQuery JavaScript framework** can be used with help of the components from this module.
These components are shown as white frames in the form designer at design time.
WYSIWYG is not supported because the graphical representation is only available in JavaScript. **TMS Web** also uses the component package **`jQWidgets`**, to improve the interoperability between **jQuery** and **Delphi.**

CLOUD SERVICES

Components for direct use of cloud services, such as **Google Calendar**.
When using the components, you don't need to worry about the implementation of the communication interface between your application and the cloud.
In particular, aspects such as encryption, authentication and authorization are handled automatically in the background.

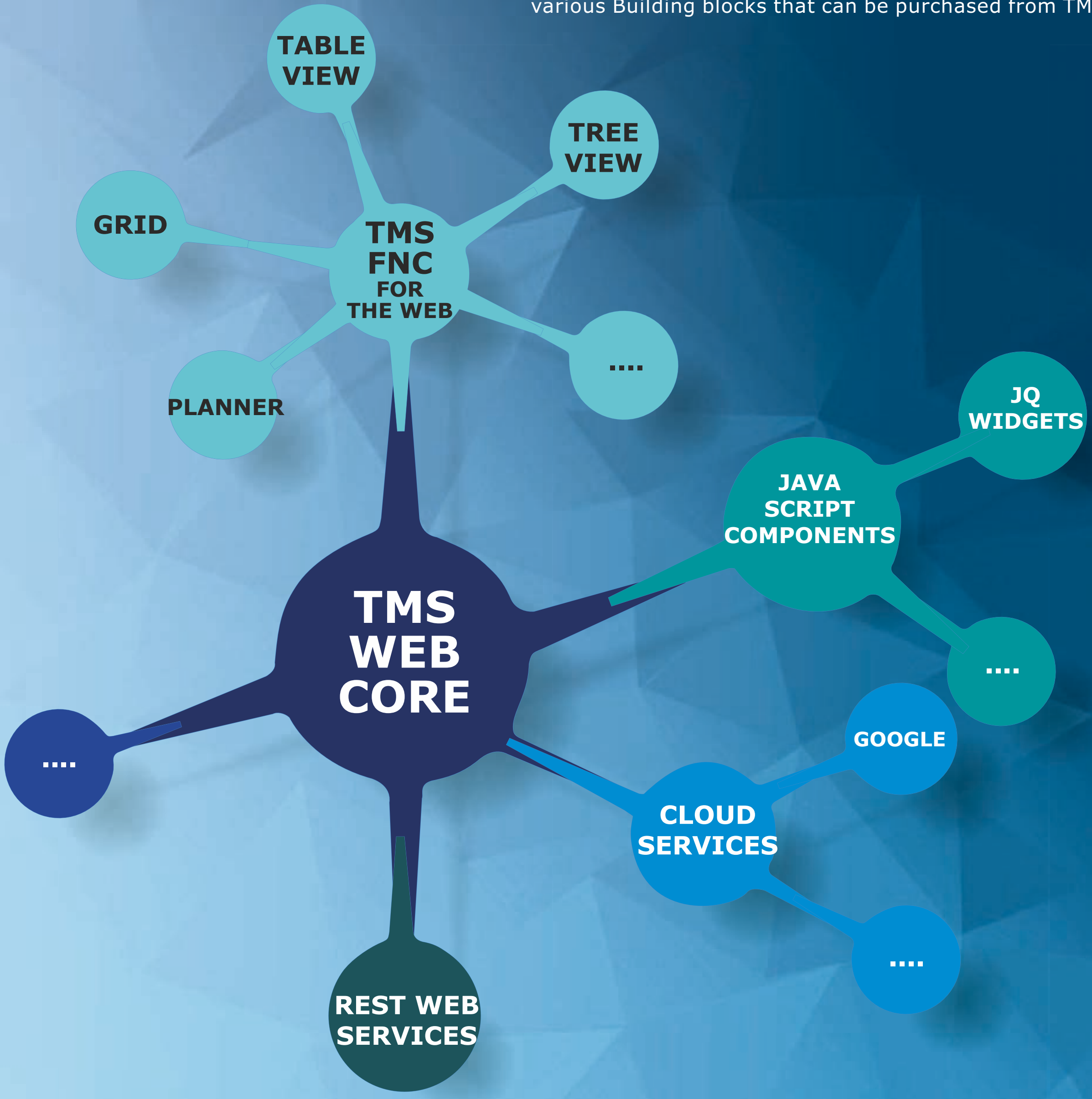
XData Server

The use of external Databases is handled using the **XData Server**. XData Server provides **REST Web Services**, which are set up automatically by the **TMS XData** engine
The final link in the web database app development chain is provided by TMS Aurelius. You develop your database structure, use **TMS Aurelius** to connect your database to your application logic and use then **XData,** to let the information become available. Your **Graphical User Interface (GUI)** on the web - if developed with **TMS Web** - can use this standardized interface to access the data and even new data, transfer records or changes to the database.

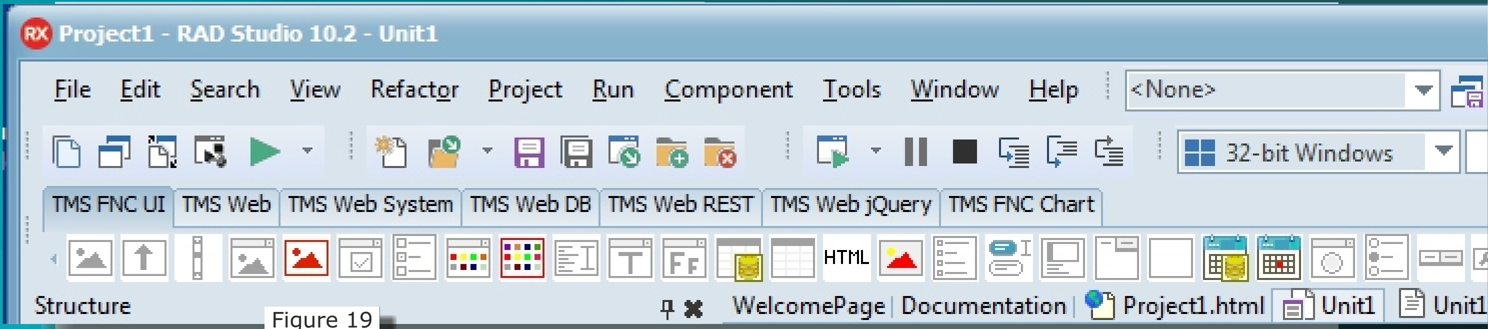
Starting at page 18 of the issue (page 13 of the article) you will find an exact overview of all the available components from TMS Web

About the CoAuthor
Dr. Holger Flick studied Informatics at the University of Dortmund and graduated at the faculty of mechanical engineering at the Ruhr-University Bochum.
He has been programming in Delphi since 1996, and has been very active in the wider Delphi community. Since his student days he has worked freelance on many projects for Borland, and was able to exchange his knowledge directly with lots of Delphi- experts from Silicon Valley. He mainly tested Delphi for the Q&A department, but also programmed database applications and Web Applications for the Borland Developer Network. He has also been a frequent seminar and conference speaker, covering a variety of Delphi-related topics.
His sincere engagement and his very extensive knowledge of Delphi and other programming languages C#, Objective-C) culminated in his being made a Delphi MVP in 2016.
Since 2013 Dr. Holger Flick has been responsible for all the Software- and Hardware -Architecture at Korfmann Lufttechnik GmbH in Witten.
In 2017 he became chief evangelist for TMS software, writing many technical articles, bi-lingual video tutorials and offering guidance via seminars.
He now writes for Blaise Pascal Magazine, and we welcome his contribution.

Figure 18: TMS Web Core can be extended. The Mind Map shows various Building blocks that can be purchased from TMS.

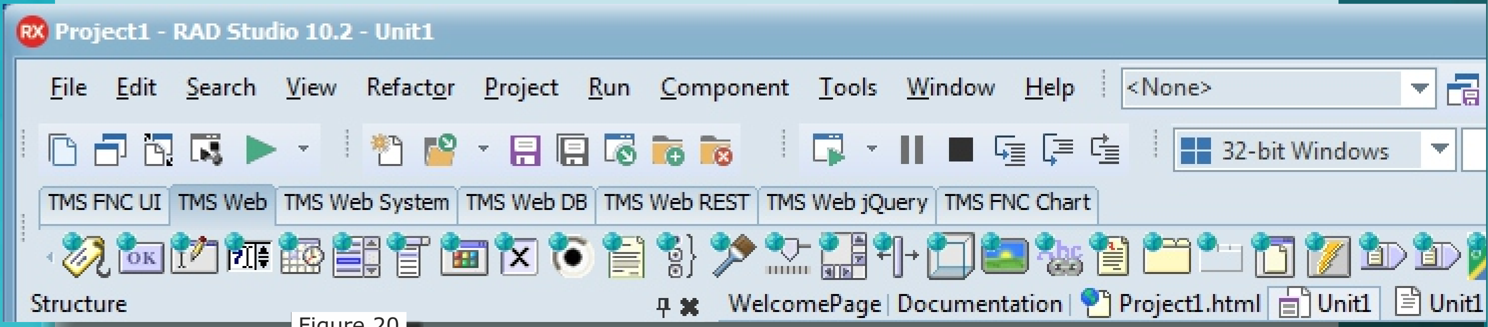


Note:
Even if it has only a building block for the jQuery JavaScript Framework, TMS Web can still work with other JavaScript frameworks, which offer visual components. In particular, it is advisable to consult with TMS for your interest in special frameworks. Without requests for support for a particular framework, TMS will not provide the needed supporting module.


Project1 - RAD Studio 10.2 - Unit1
File Edit Search View Refactor Project Run Component Tools Window Help
<None>
32-bit Windows
TMS FNC UI TMS Web TMS Web System TMS Web DB TMS Web REST TMS Web jQuery TMS FNC Chart
HTML
Structure
WelcomePage Documentation Project1.html Unit1 Unit1
Figure 19

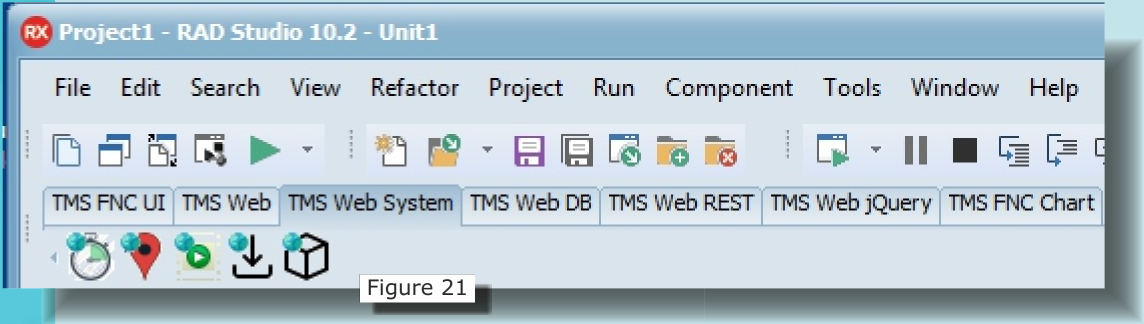
TMS FNC UI
TTMSFNC*

BitmapContainer	PopUp	ScrollBar	BitmapPicker
BitmapSelector	CheckGroupPicker	CheckGroup	ColorPicker
ColorSelector	Edit	FontnamePicker	FontSizePicker
GridDatabaseAdapter	Grid HTMLText	Image	CheckedListBox
ListEditor	NavigationPanel	PageControl	Panel
PlannerDatabaseAdapter	Planner	RadioGroupPicker	RadioGroup
TabSet/ToolBar	ToolBarButton	ToolBarSeparator	ToolBarFontNamePicker
ToolBarFontSizePicker	ToolBarColorPicker	ToolBarBitmapPicker	ItemPicker/DockPanel
TreeView	CheckedTree	ViewTaleView	


Project1 - RAD Studio 10.2 - Unit1
File Edit Search View Refactor Project Run Component Tools Window Help
<None>
32-bit Windows
TMS FNC UI TMS Web TMS Web System TMS Web DB TMS Web REST TMS Web jQuery TMS FNC Chart
Structure
WelcomePage Documentation Project1.html Unit1 Unit1
Figure 20

TMS WEB
TWEB*

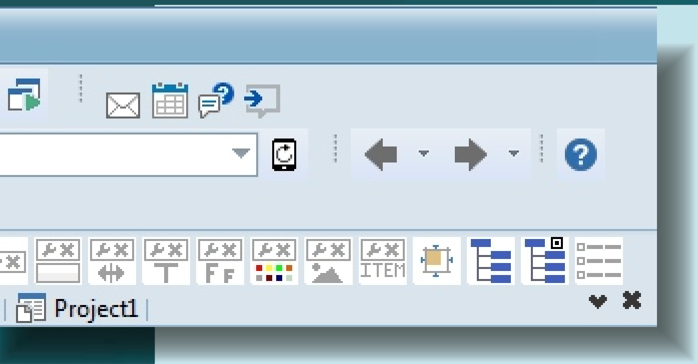
Label	WebButton	WebEdit	SpinEdit
DateTimePicker	ListBox	ComboBox	ColorPicker
CheckBox	RadioButton	Memo	RadioGroup
PaintBox	TrackBar	ScollBox	Splitter
Panel	ImageControl	LinkLabel	RichEdit
Tabset	PageControl	TabSheet	SpeedButton
ToolBar	RichEditToolbar	GoogleMaps	YouTube
MainMenu	GridPanel	MessageDlg	ToggleButton
BitBtn	GroupBox		

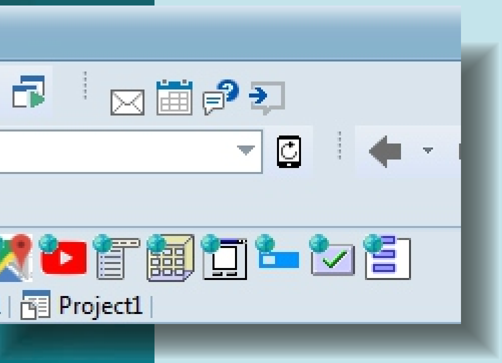

Project1 - RAD Studio 10.2 - Unit1
File Edit Search View Refactor Project Run Component Tools Window Help
TMS FNC UI TMS Web TMS Web System TMS Web DB TMS Web REST TMS Web jQuery TMS FNC Chart
Figure 21

TMS WEB SYSTEM
TWEB*

Timer	GeoLocation	MultiMediaPlyer	FileUpload
Code Manager			

Some of the **EXAMPLES ARE SPREAD OVER TWO PAGES** because they are to large.If you are using the **.pdf** file, you can change your **Acrobat Reader** settings to **two page view,** which gives you a better and more complete overview of what is all available: In **Acrobat Reader** open **Preferences** Choose in the left column top **Page Display**. then select at the top **Two Up**.





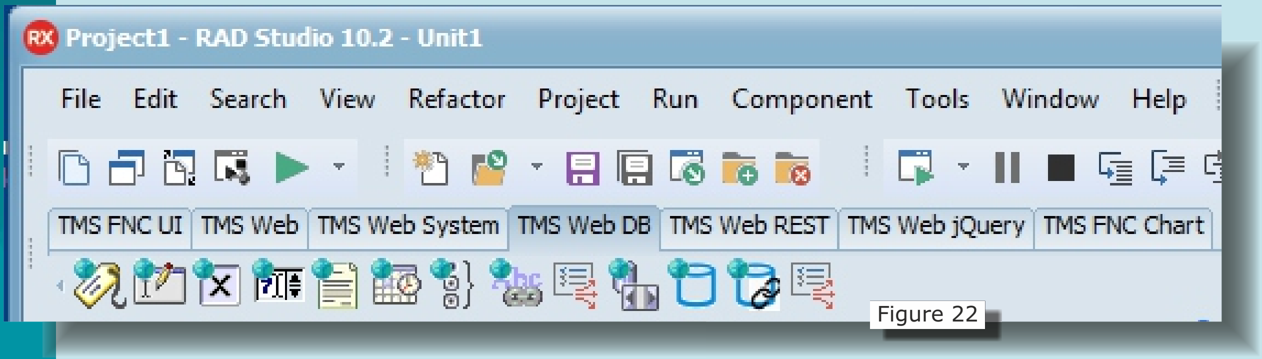


Figure 22

TMS WEB DB

TWEB*

DBLabel	DBEdit	DBCheckBox	DBSpinEdit
DBMemo	DBDateTimePicker	DBRadioGroup	DBLinkLabel
DataSource	DBNavigator	ClientDataSet	ClientConnection
ClientDataSource			

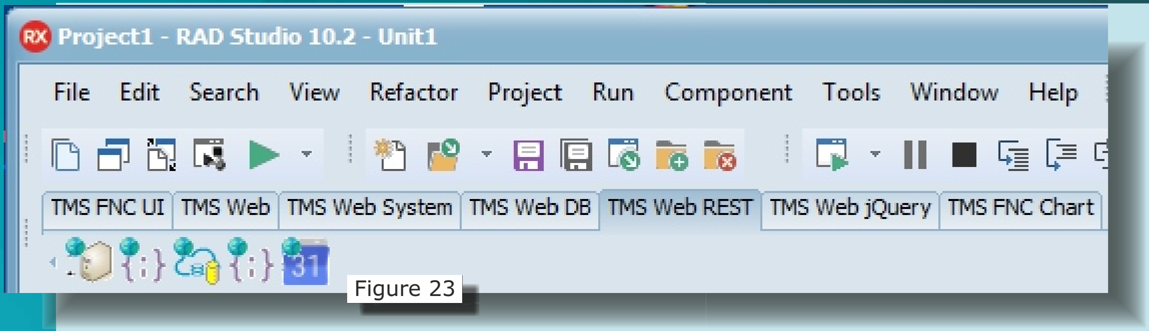

Project1 - RAD Studio 10.2 - Unit1
File Edit Search View Refactor Project Run Component Tools Window Help
TMS FNC UI
TMS Web
TMS Web System
TMS Web DB
TMS Web REST
TMS Web jQuery
TMS FNC Chart

Figure 23

TMS WEB DB

TMS Web REST*

HttpRequest	RESTClient	CloudData	GoogleClient
GoogleCalendar			

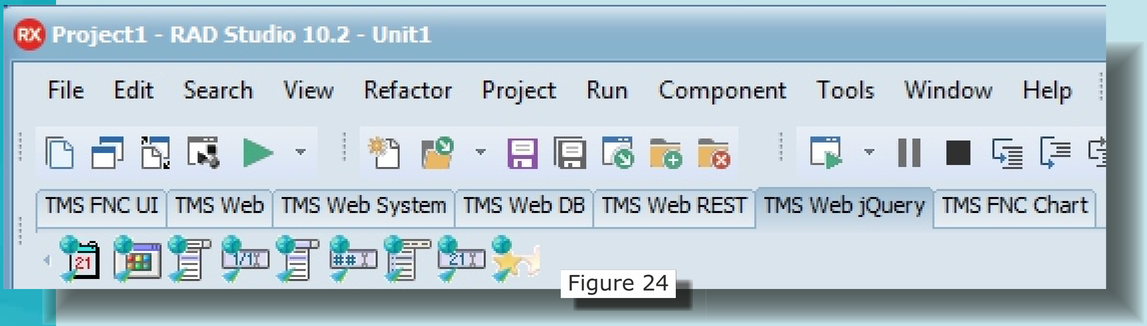

Project1 - RAD Studio 10.2 - Unit1
File Edit Search View Refactor Project Run Component Tools Window Help
TMS FNC UI
TMS Web
TMS Web System
TMS Web DB
TMS Web REST
TMS Web jQuery
TMS FNC Chart

Figure 24

TMS WEB jQuery

TWebJQX*

Calendar	ColorPicker	ComboBox	DateTimeInput
Dropdownlist	MaskedInput	Menu NumberInput	Rating

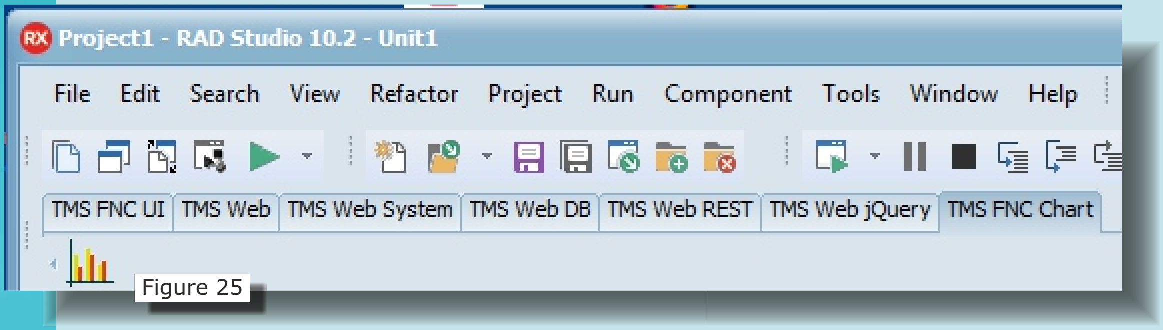

Project1 - RAD Studio 10.2 - Unit1
File Edit Search View Refactor Project Run Component Tools Window Help
TMS FNC UI
TMS Web
TMS Web System
TMS Web DB
TMS Web REST
TMS Web jQuery
TMS FNC Chart

Figure 25

TMS FNC CHART

TMSFNCChart

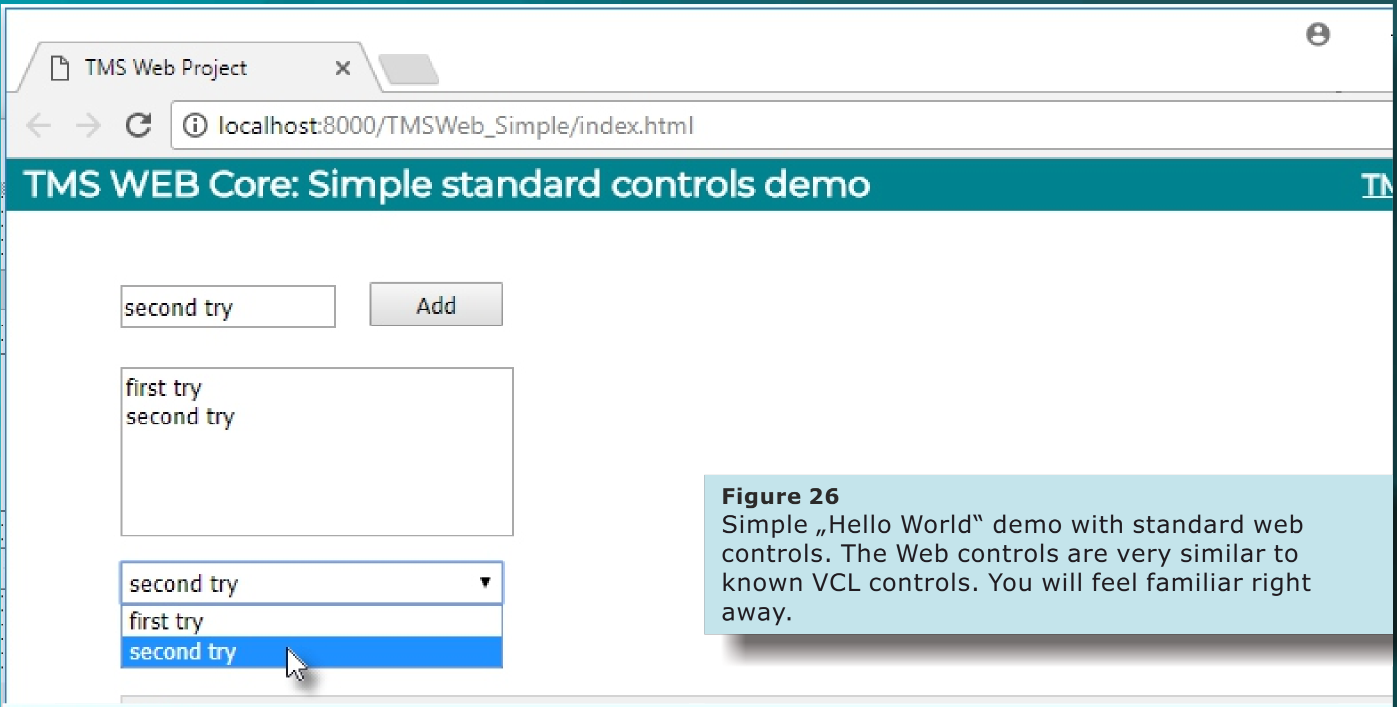
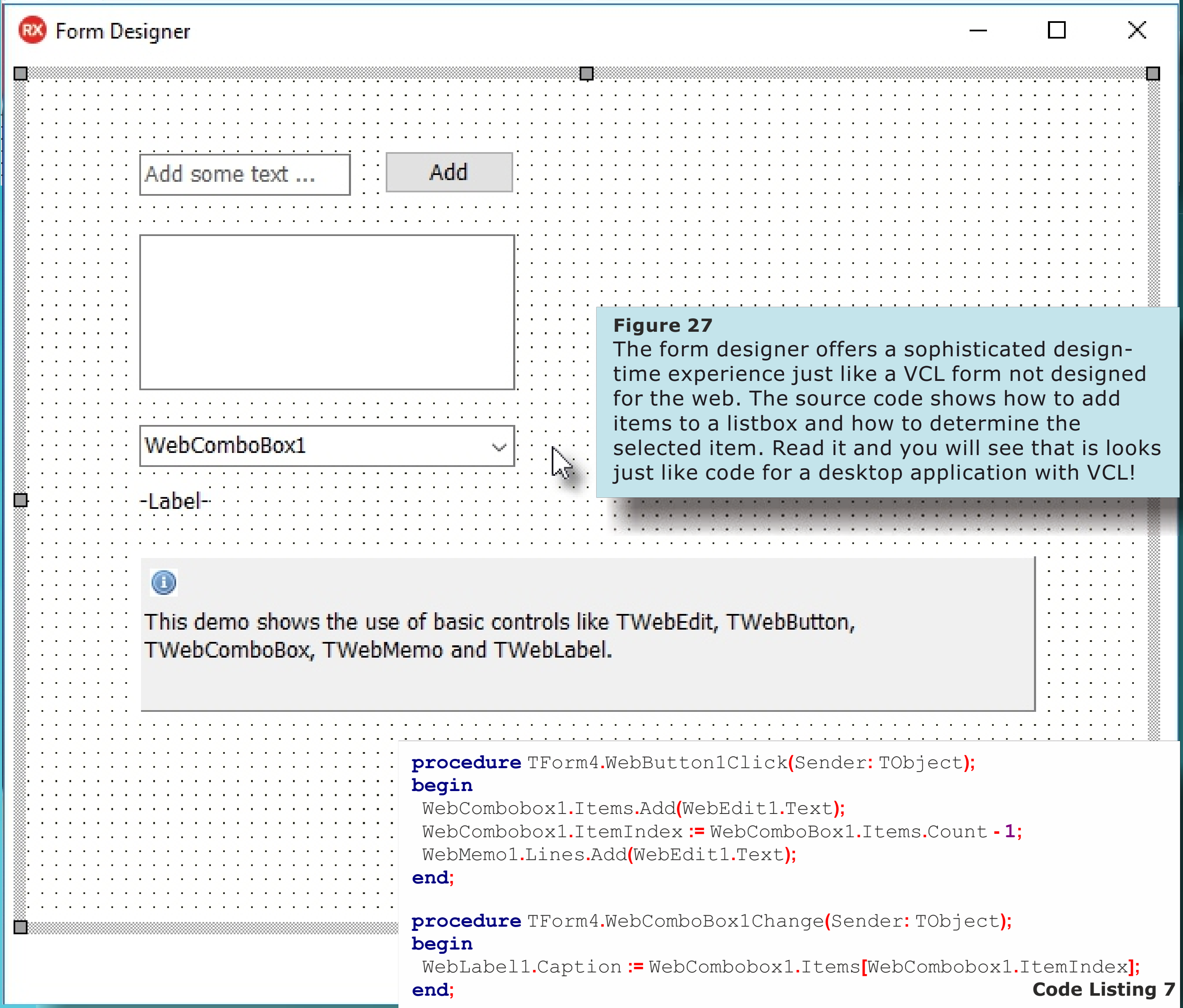

TMS Web Project
localhost:8000/TMSWeb_Simple/index.html
TMS WEB Core: Simple standard controls demo
second try
Add
first try
second try
second try
first try
second try

Figure 26
Simple „Hello World" demo with standard web controls. The Web controls are very similar to known VCL controls. You will feel familiar right away.

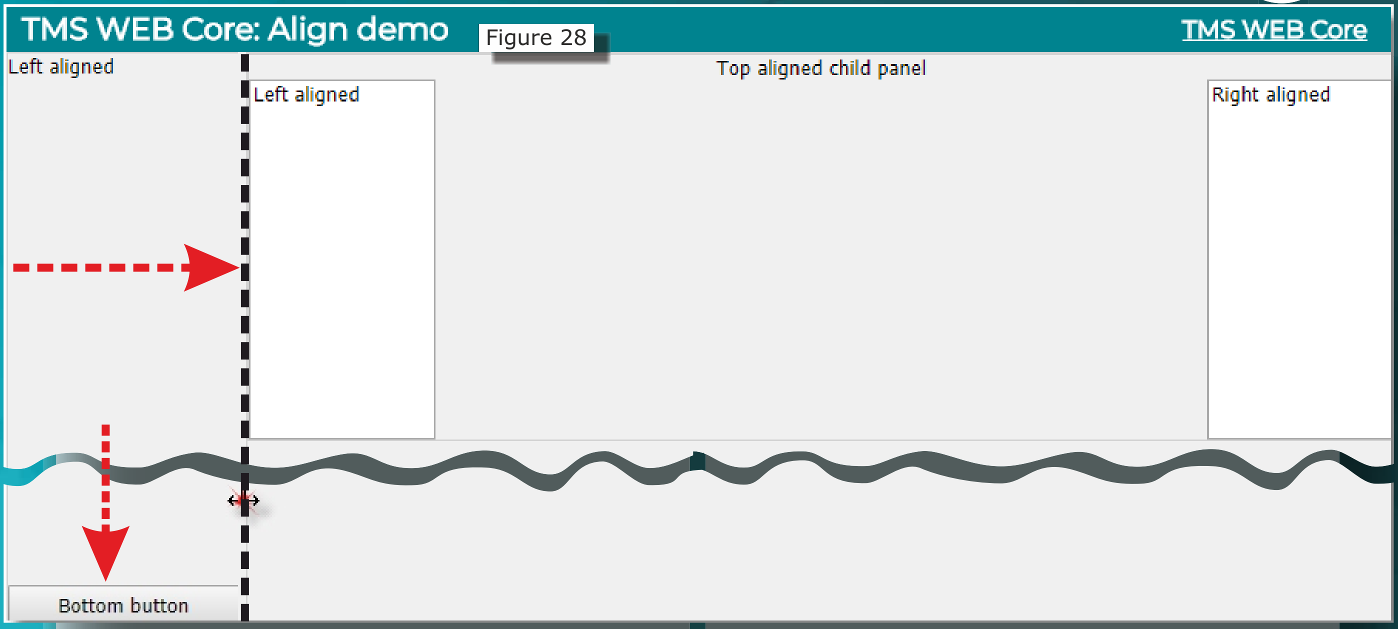


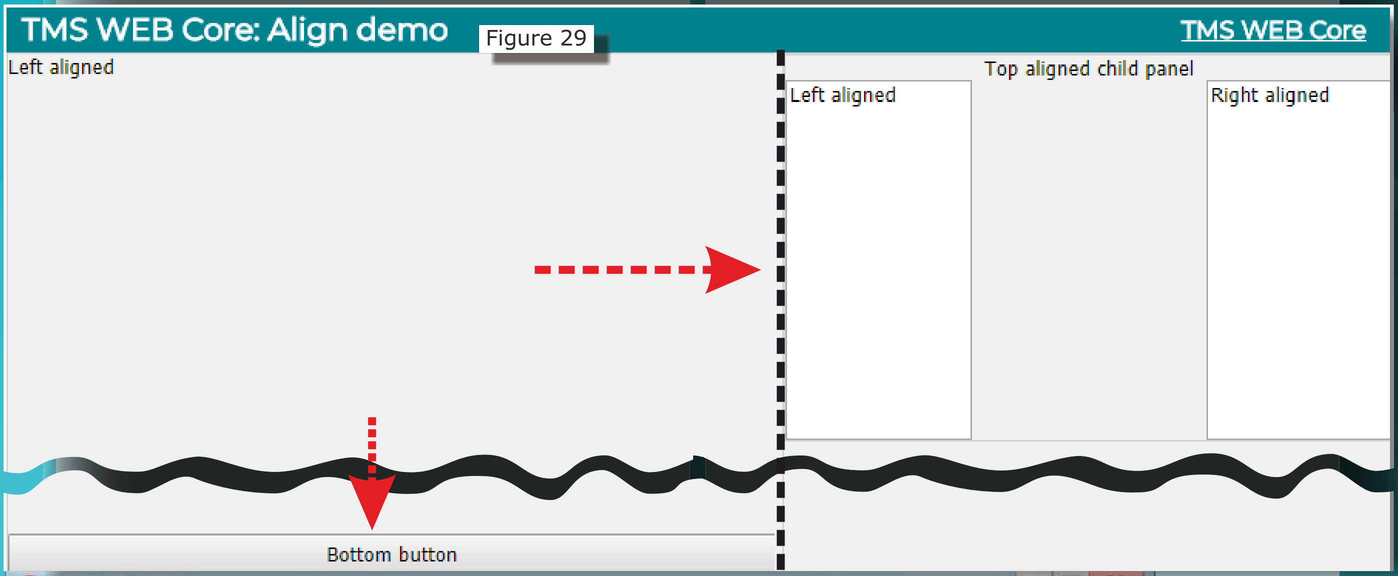
This demo shows the use of basic controls like TWebEdit, TWebButton, TWebComboBox, TWebMemo and TWebLabel.

Figure 27
The form designer offers a sophisticated design-time experience just like a VCL form not designed for the web. The source code shows how to add items to a listbox and how to determine the selected item. Read it and you will see that is looks just like code for a desktop application with VCL!

```
procedure TForm4.WebButton1Click(Sender: TObject);
begin
 WebCombobox1.Items.Add(WebEdit1.Text);
 WebCombobox1.ItemIndex := WebComboBox1.Items.Count - 1;
 WebMemo1.Lines.Add(WebEdit1.Text);
end;

procedure TForm4.WebComboBox1Change(Sender: TObject);
begin
 WebLabel1.Caption := WebCombobox1.Items[WebCombobox1.ItemIndex];
end;
```
Code Listing 7

TMSWeb_Align.dproj





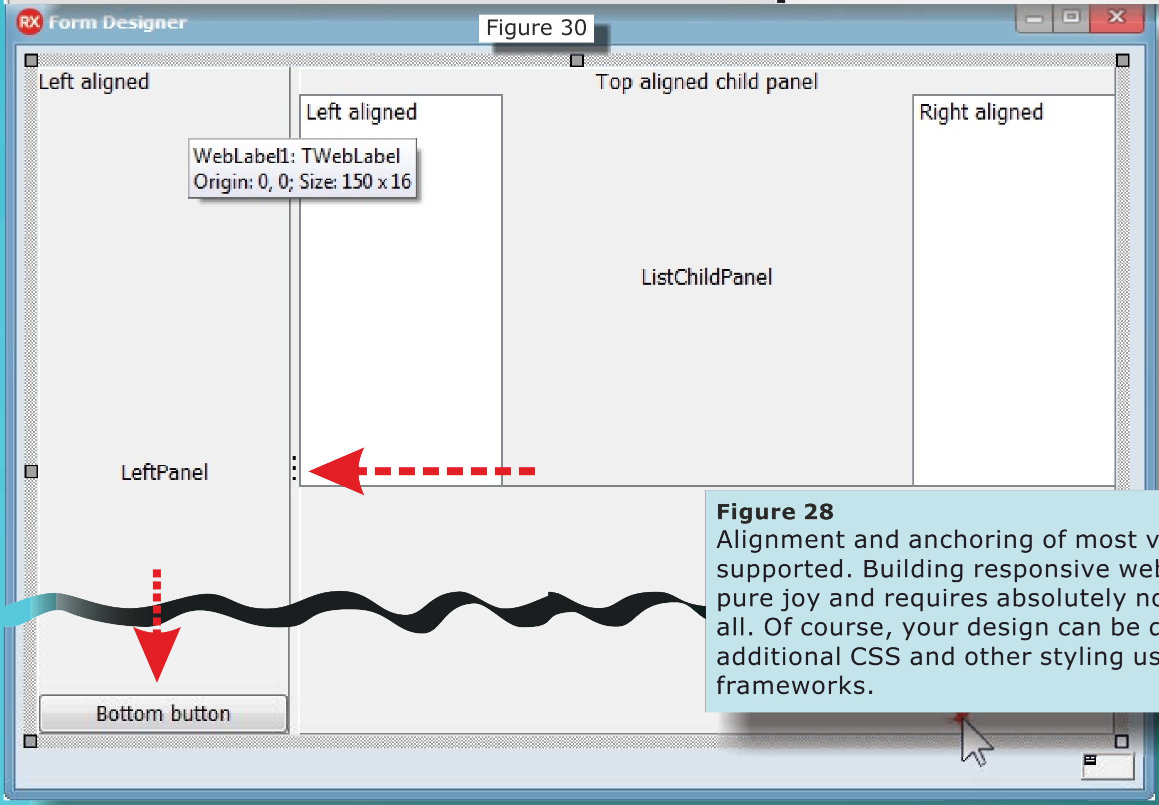
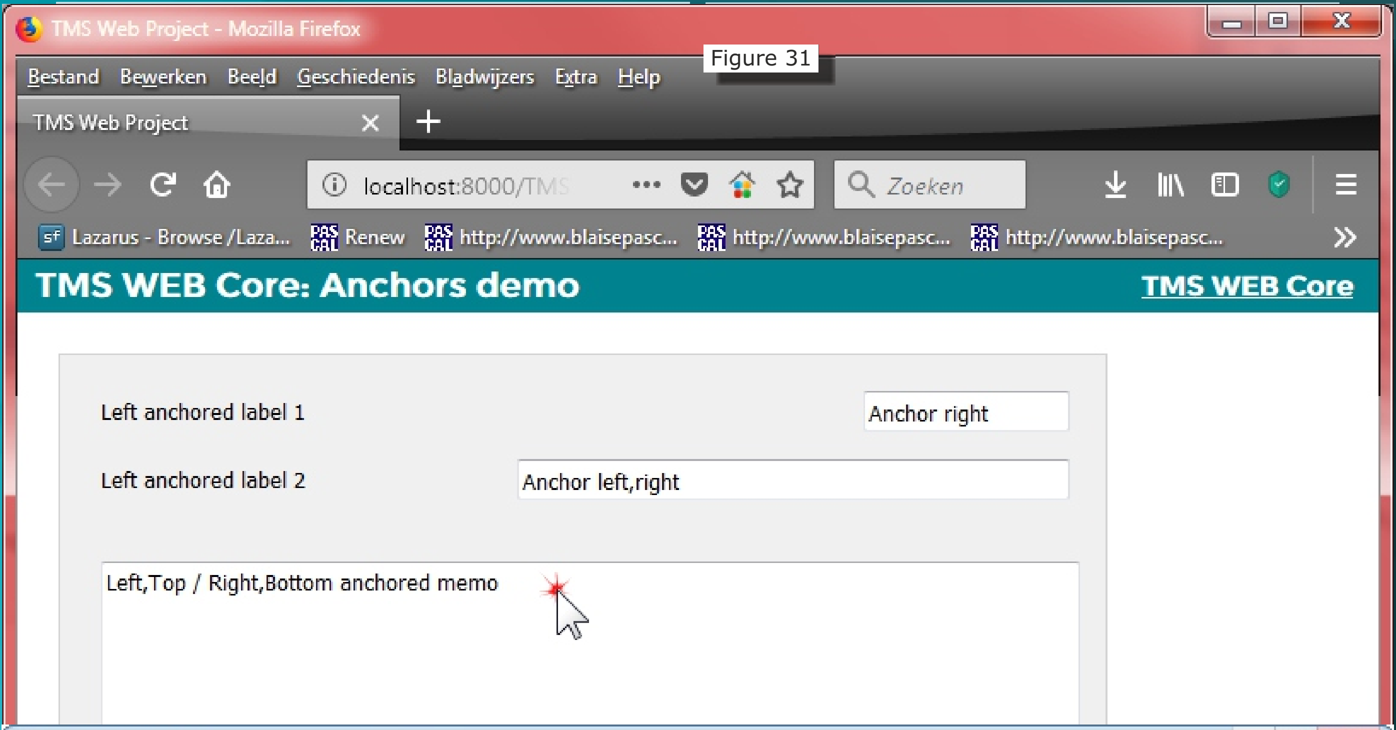


Figure 28
Alignment and anchoring of most visual control is supported. Building responsive web pages becomes pure joy and requires absolutely no source code at all. Of course, your design can be decorated with additional CSS and other styling using JavaScript frameworks.

TMSWeb_Anchors.dproj

Figure 31


TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS Web Project
localhost:8000/TMS
Zoeken
Lazarus - Browse /Laza...
Renew
http://www.blaisepasc...
http://www.blaisepasc...
http://www.blaisepasc...
TMS WEB Core: Anchors demo
TMS WEB Core

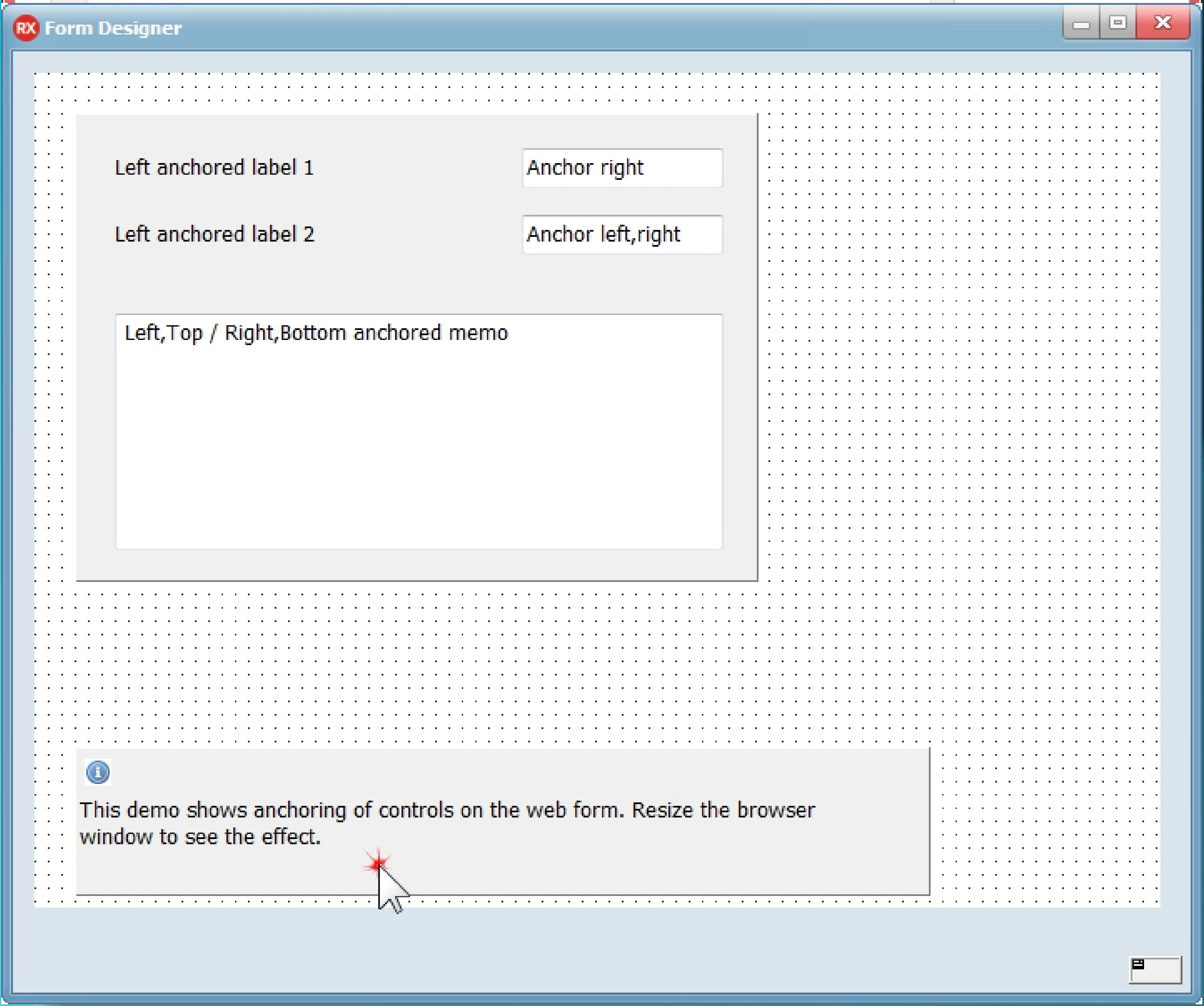
Left anchored label 1

Anchor right

Left anchored label 2

Anchor left,right

Left,Top / Right,Bottom anchored memo



Left anchored label 1

Anchor right

Left anchored label 2

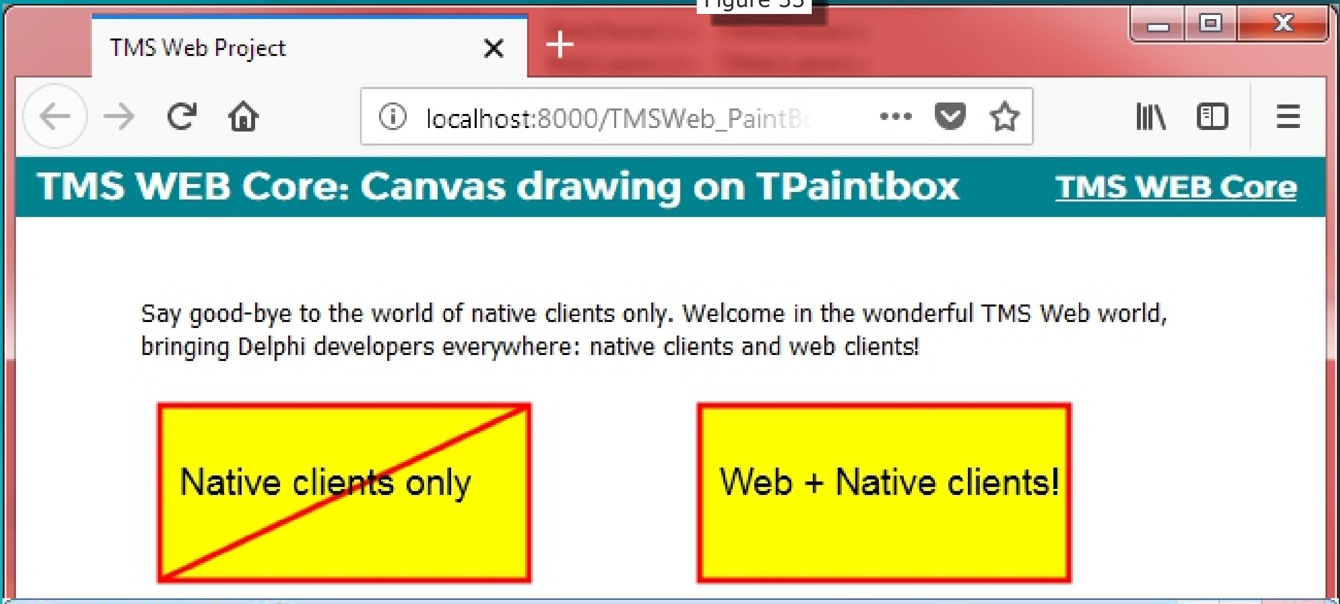
Anchor left,right

Left,Top / Right,Bottom anchored memo

This demo shows anchoring of controls on the web form. Resize the browser window to see the effect.

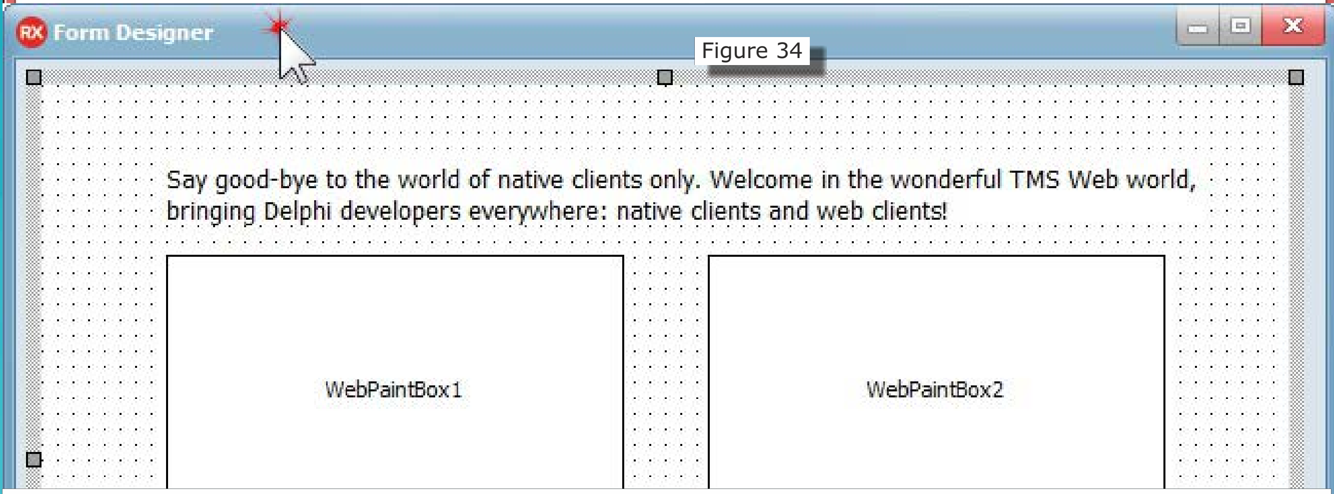
TMSWeb_PaintBox.dproj

Figure 33


TMS Web Project
localhost:8000/TMSWeb_PaintB
TMS WEB Core: Canvas drawing on TPaintbox
TMS WEB Core
Native clients only
Web + Native clients!

Say good-bye to the world of native clients only. Welcome in the wonderful TMS Web world, bringing Delphi developers everywhere: native clients and web clients!

Figure 34



Say good-bye to the world of native clients only. Welcome in the wonderful TMS Web world, bringing Delphi developers everywhere: native clients and web clients!

```
procedure TForm4.PaintSign(Control: TWebPaintBox; AText: string;  Cross: boolean);
begin
 Control.Canvas.Pen.Width := 3;
 Control.Canvas.Pen.Color := clRed;
 Control.Canvas.Brush.Color := clYellow;
 Control.Canvas.Brush.Style := bsSOlid;
 Control.Canvas.Rectangle(10,10,200,100);

 if Cross then
 begin
  Control.Canvas.MoveTo(10,100);
  Control.Canvas.LineTo(200,10);
  Control.Canvas.Font.Size := 14;
 end;

 Control.Canvas.Font.Name := 'Arial';
 Control.Canvas.Font.Size := 14;
 Control.Canvas.TextOut(20,40,AText);
end;

procedure TForm4.WebPaintBox1Paint(Sender: TObject);
begin
 PaintSign(WebPaintBox1, 'Native clients only', true);
end;

procedure TForm4.WebPaintBox2Paint(Sender: TObject);
begin
 PaintSign(WebPaintBox2, 'Web + Native clients!', false);
end;
```

Figure 34
Paintbox support in web applications! The paintbox allows you to paint on a canvas just like in any VCL, FMX or LCL application. The canvas class offers known methods to draw, paint and output texts. TMS Web will make sure that the box is rendered correctly in any browser.

Code Listing 8

Figure 35

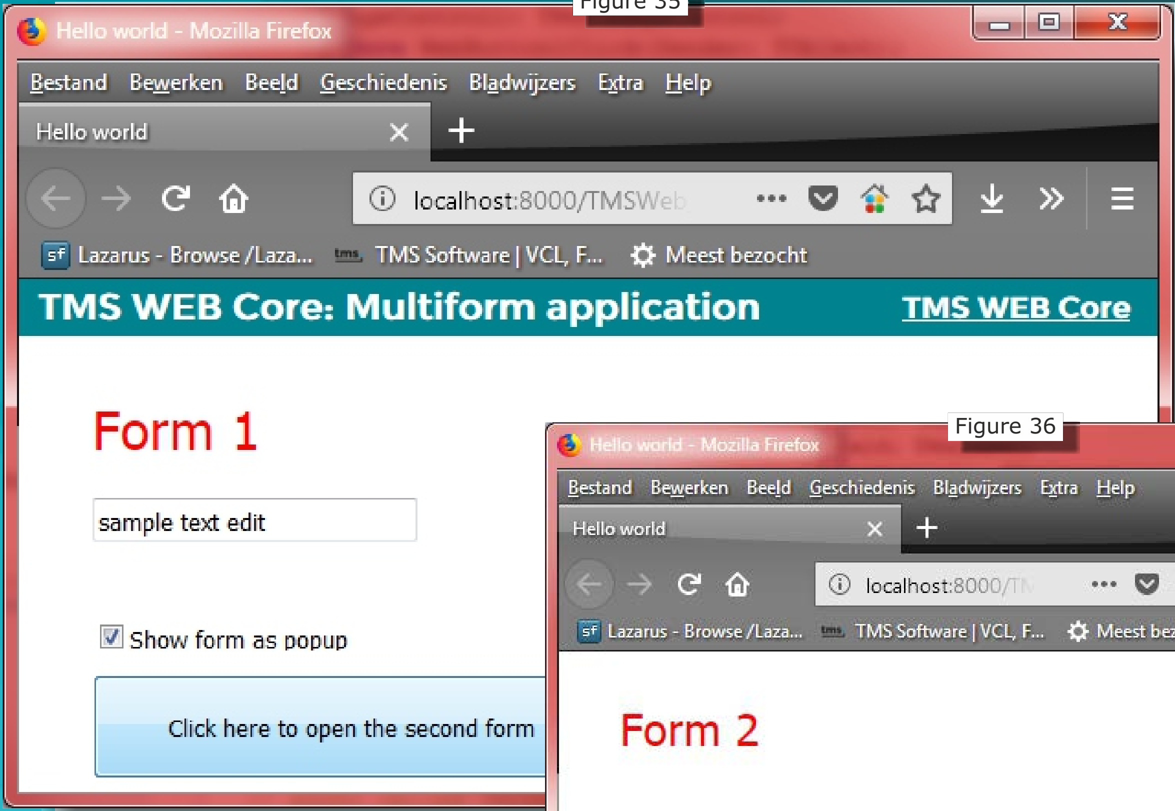

Hello world - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Hello world
localhost:8000/TMSWeb
Lazarus - Browse /Laza... TMS Software | VCL, F... Meest bezocht
TMS WEB Core: Multiform application
TMS WEB Core
Form 1
sample text edit
Show form as popup
Click here to open the second form

Figure 36

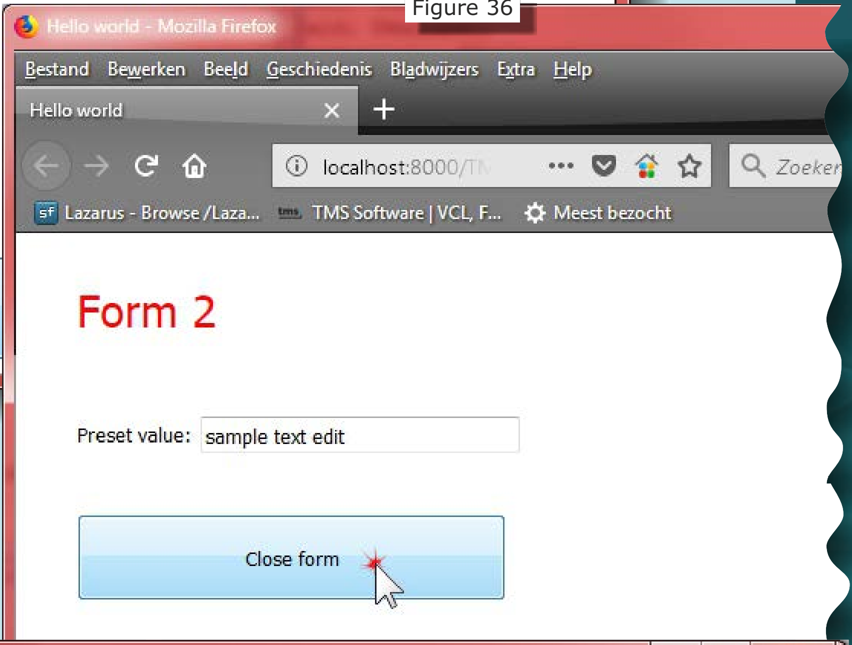

Hello world - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Hello world
localhost:8000/T
Zoeken
Lazarus - Browse /Laza... TMS Software | VCL, F... Meest bezocht
Form 2
Preset value: sample text edit
Close form

Figure 37

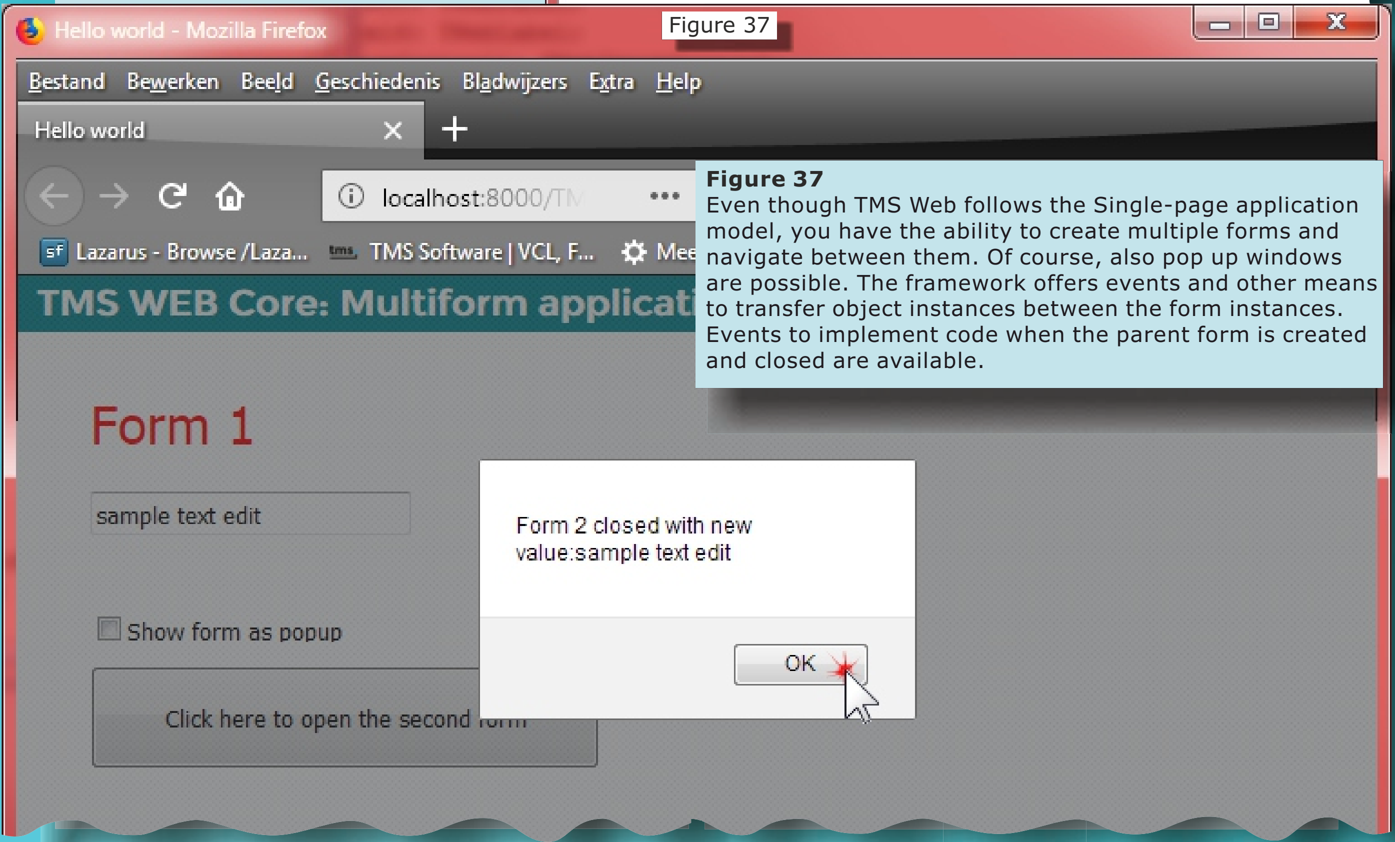
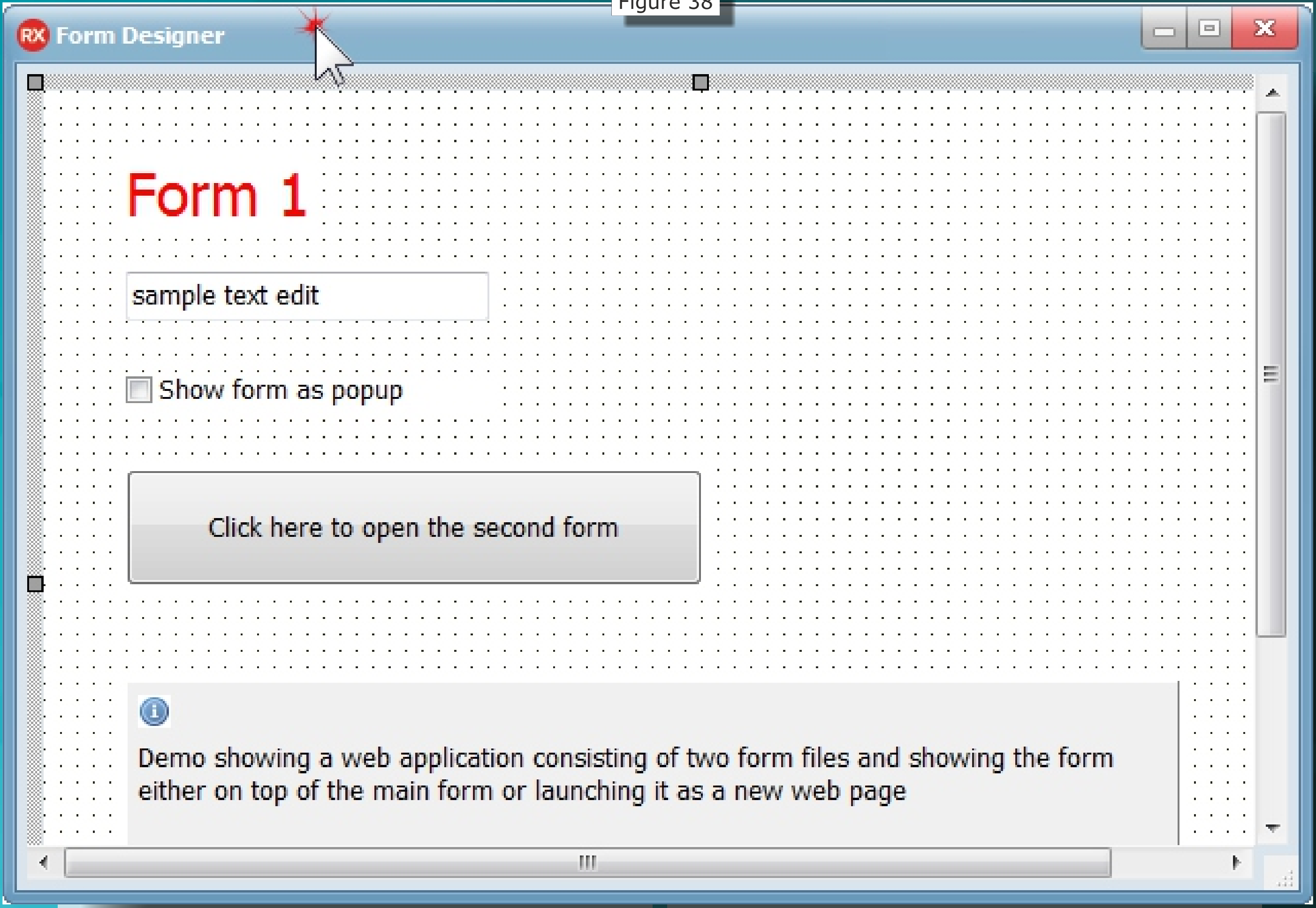

Hello world - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
Hello world
localhost:8000/T
Lazarus - Browse /Laza... TMS Software | VCL, F... Mee
TMS WEB Core: Multiform applicat
Form 1
sample text edit
Show form as popup
Click here to open the second form
Form 2 closed with new value:sample text edit
OK

Figure 37
Even though TMS Web follows the Single-page application model, you have the ability to create multiple forms and navigate between them. Of course, also pop up windows are possible. The framework offers events and other means to transfer object instances between the form instances. Events to implement code when the parent form is created and closed are available.

TMSWeb_Multiform.dproj

Figure 38



Demo showing a web application consisting of two form files and showing the form either on top of the main form or launching it as a new web page

```
var
 Form1: TForm1;

implementation

{$R *.dfm}

uses
 unit2, WebLib.WebTools;

procedure TForm1.WebButton1Click(Sender: TObject);
var
 newform: TForm2;

 procedure AfterShowModal(AValue: TModalResult);
 begin
  ShowMessage('Form 2 closed with new value:'+newform.frm2Edit.Text);
  WebEdit1.Text := newform.frm2Edit.Text;
 end;

 // async called OnCreate for TForm2
 procedure AfterCreate(AForm: TObject);
 begin
  (AForm as TForm2).frm2Edit.Text := WebEdit1.Text;
 end;

begin
 newform := TForm2.CreateNew(@AfterCreate);
 newform.Popup := WebCheckBox1.Checked;
 newform.ShowModal(@AfterShowModal);
end;

end.
```

Code Listing 9

Figure 39

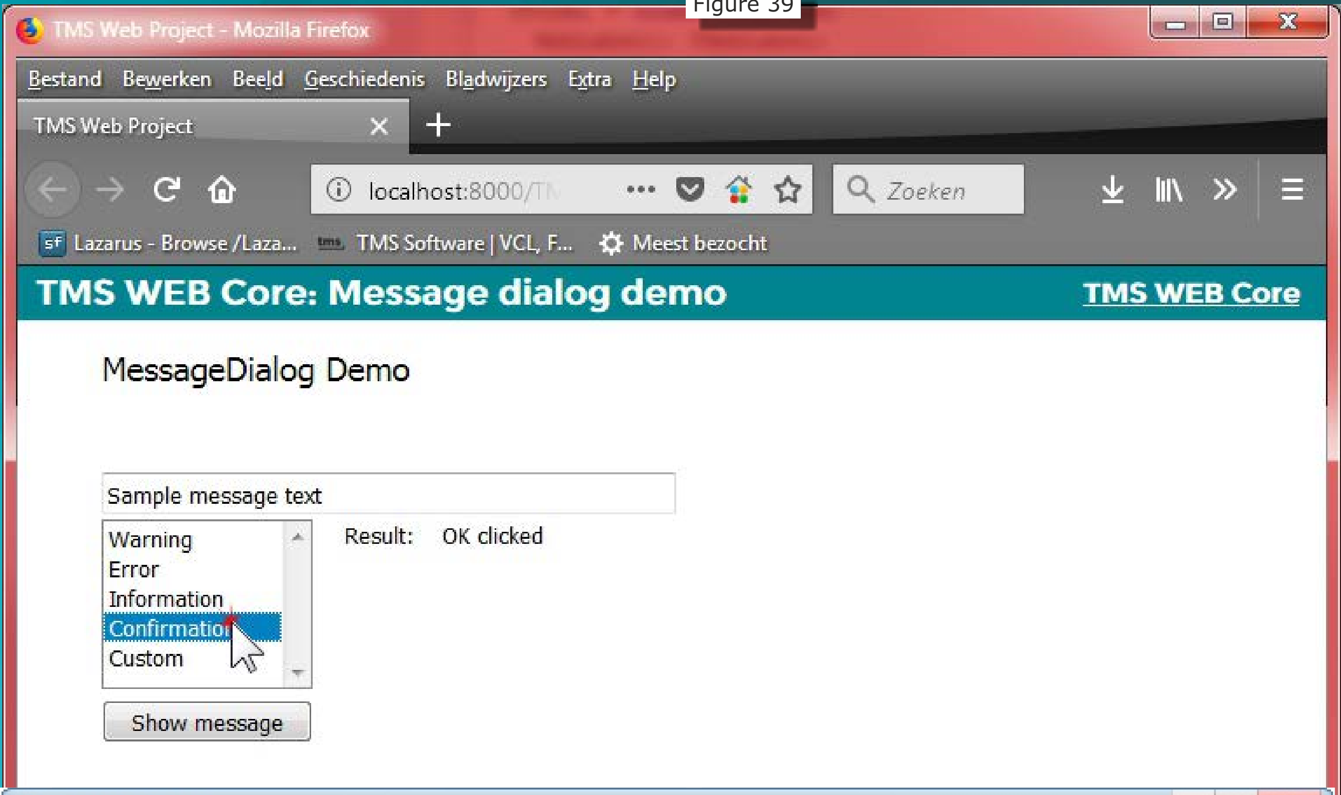


Figure 40

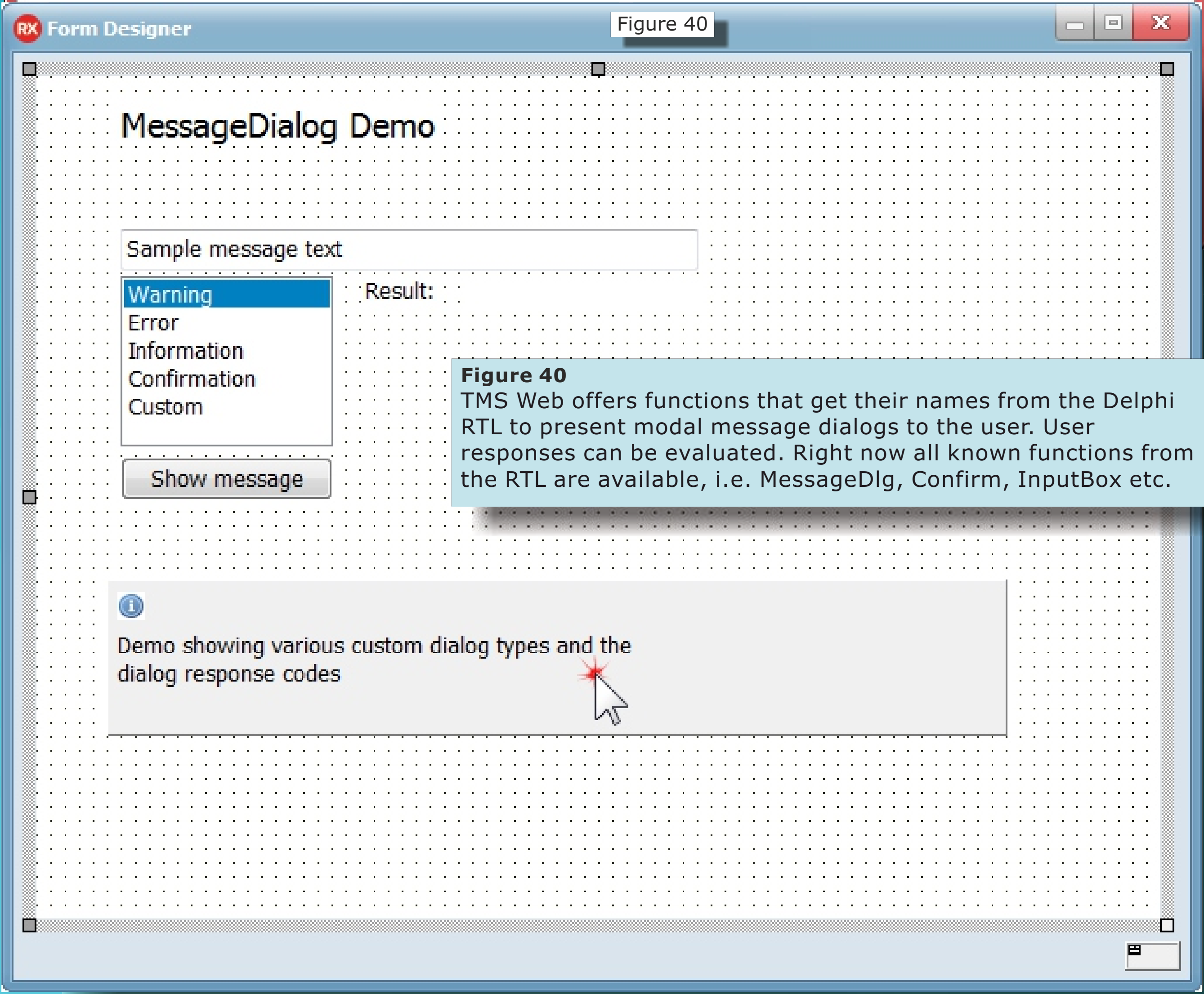
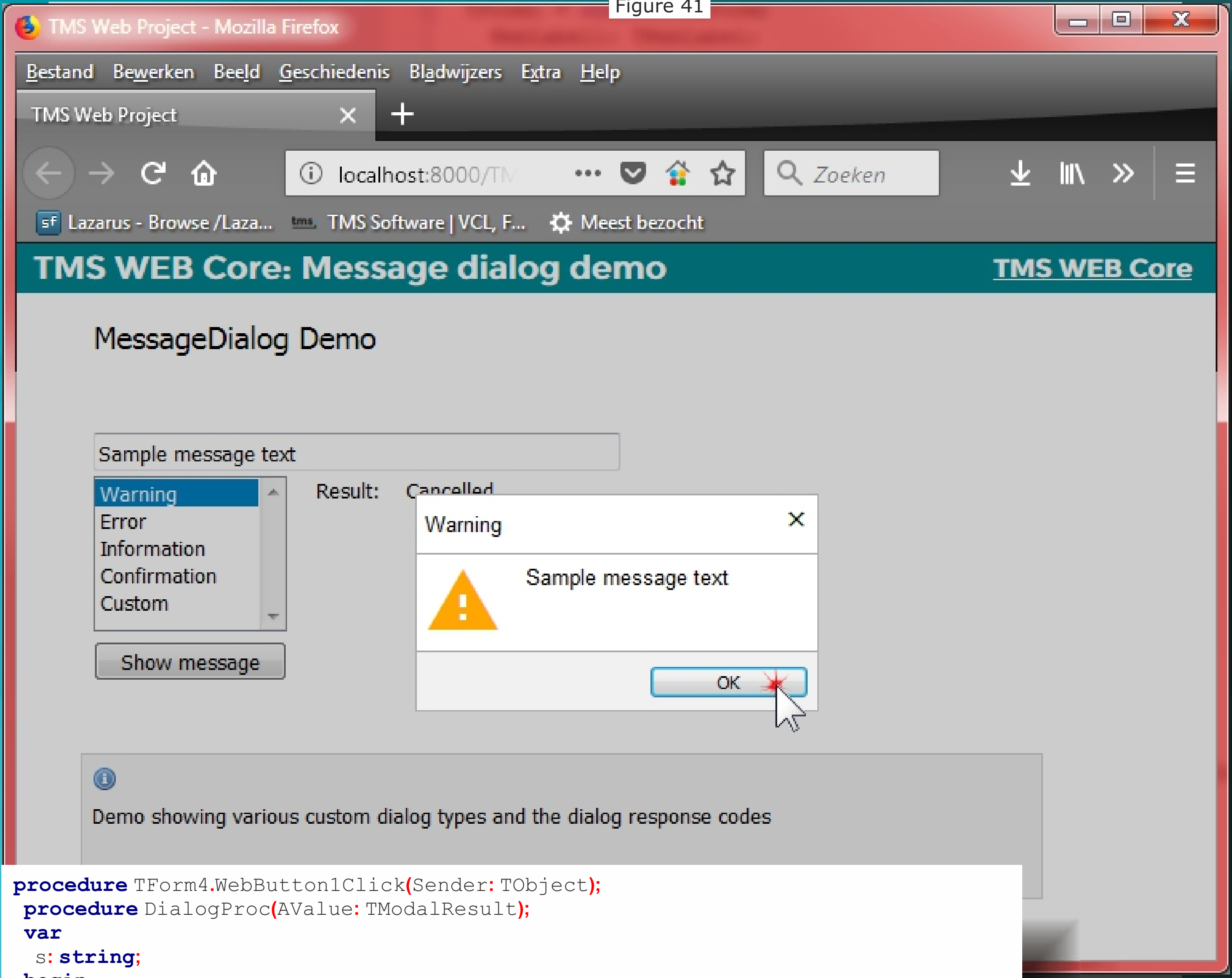


Figure 40
TMS Web offers functions that get their names from the Delphi RTL to present modal message dialogs to the user. User responses can be evaluated. Right now all known functions from the RTL are available, i.e. MessageDlg, Confirm, InputBox etc.

TMSWeb_Dialogs.dproj

Figure 41


TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS Web Project
localhost:8000/TM
Zoeken
Lazarus - Browse /Laza...
TMS Software | VCL, F...
Meest bezocht

TMS WEB Core: Message dialog demo   TMS WEB Core

MessageDialog Demo

Sample message text

Warning
Error
Information
Confirmation
Custom

Show message

Result: Cancelled

Warning

Sample message text

OK

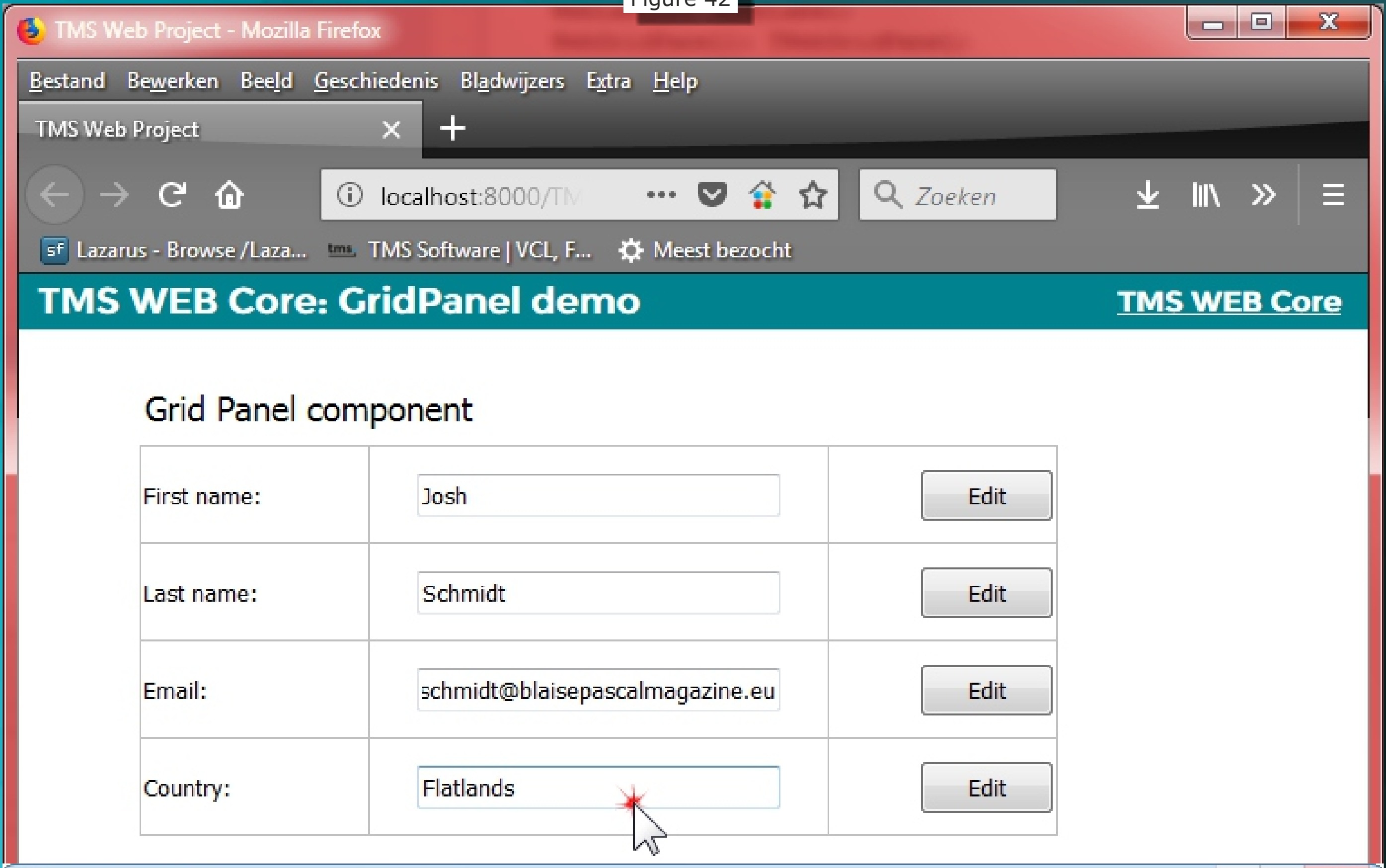
Demo showing various custom dialog types and the dialog response codes

```
procedure TForm4.WebButton1Click(Sender: TObject);
 procedure DialogProc(AValue: TModalResult);
 var
  s: string;
 begin
  if AValue = mrOk then
   s := 'OK clicked'
  else if AValue = mrYes then
   s := 'Yes clicked'
  else if AValue = mrNo then
   s := 'No clicked'
  else if AValue = mrAbort then
   s := 'Abort clicked'
  else if AValue = mrRetry then
   s := 'Retry clicked'
  else if AValue = mrClose then
   s := 'Close clicked'
  else if AValue = mrCancel then
   s := 'Cancelled';

  WebLabel2.Caption := s;
 end;

begin
 case WebListBox2.ItemIndex of
  0: MessageDlg(WebEdit1.Text, mtWarning, [], @DialogProc);
  1: MessageDlg(WebEdit1.Text, mtError, [], @DialogProc);
  2: MessageDlg(WebEdit1.Text, mtInformation, [], @DialogProc);
  3: MessageDlg(WebEdit1.Text, mtConfirmation, [mbYes, mbNo, mbCancel], @DialogProc);
  4: MessageDlg(WebEdit1.Text, mtCustom, [mbAbort, mbRetry, mbClose], @DialogProc);
 end;
end;
```

Code Listing 10

TMSWeb_GridPanel.dproj

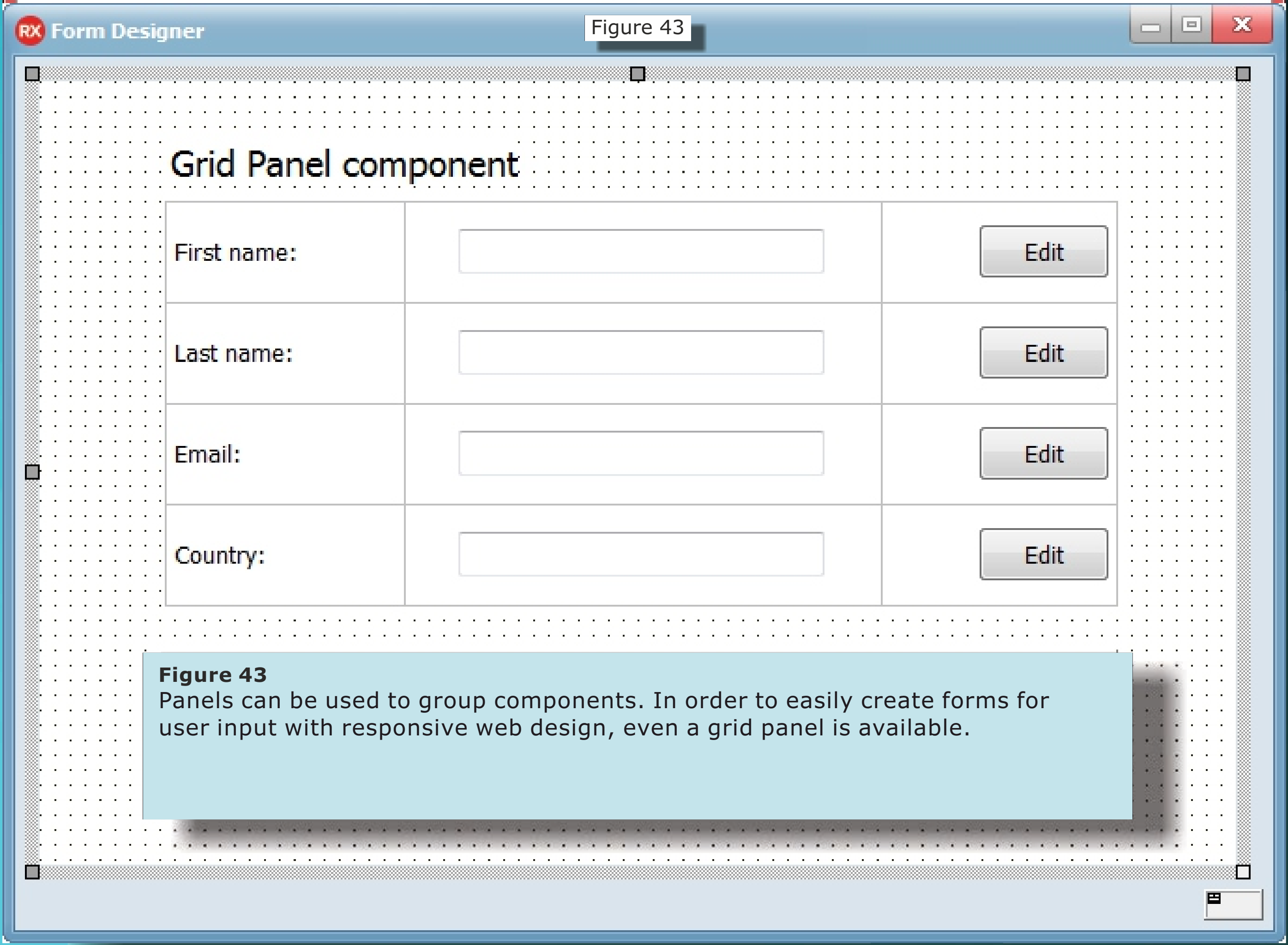
Figure 42


TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS Web Project
localhost:8000/TM
Zoeken
Lazarus - Browse /Laza... TMS Software | VCL, F... Meest bezocht
TMS WEB Core: GridPanel demo
TMS WEB Core

Grid Panel component

First name:	Josh	Edit
Last name:	Schmidt	Edit
Email:	schmidt@blaisepascalmagazine.eu	Edit
Country:	Flatlands	Edit

Figure 43



Grid Panel component

First name:		Edit
Last name:		Edit
Email:		Edit
Country:		Edit

Figure 43
Panels can be used to group components. In order to easily create forms for user input with responsive web design, even a grid panel is available.

Figure 44

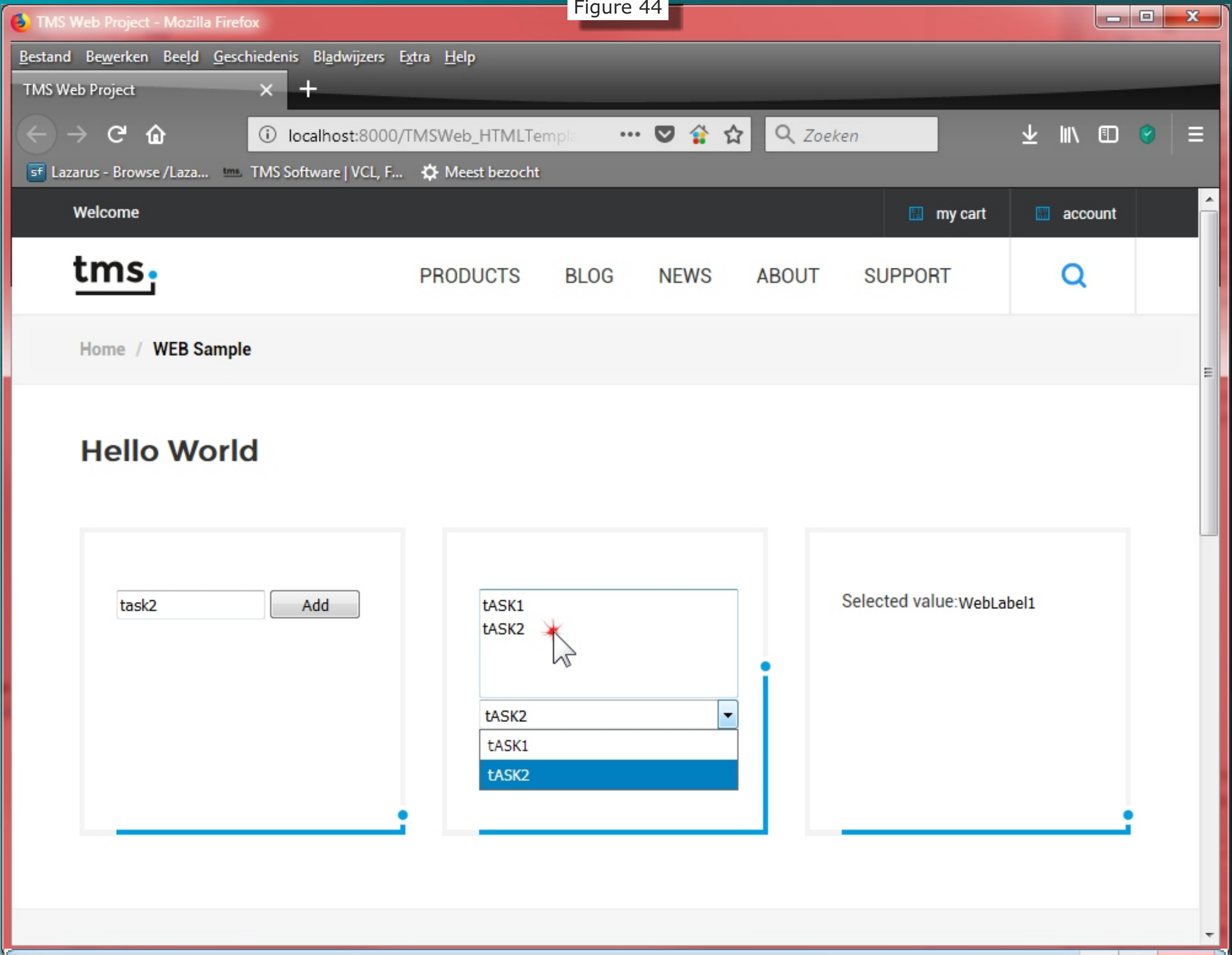

TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
localhost:8000/TMSWeb_HTMLTempl
Welcome
my cart
account
PRODUCTS
BLOG
NEWS
ABOUT
SUPPORT
Home / WEB Sample
Hello World
task2
Add
tASK1
tASK2
Selected value:WebLabel1

Figure 45

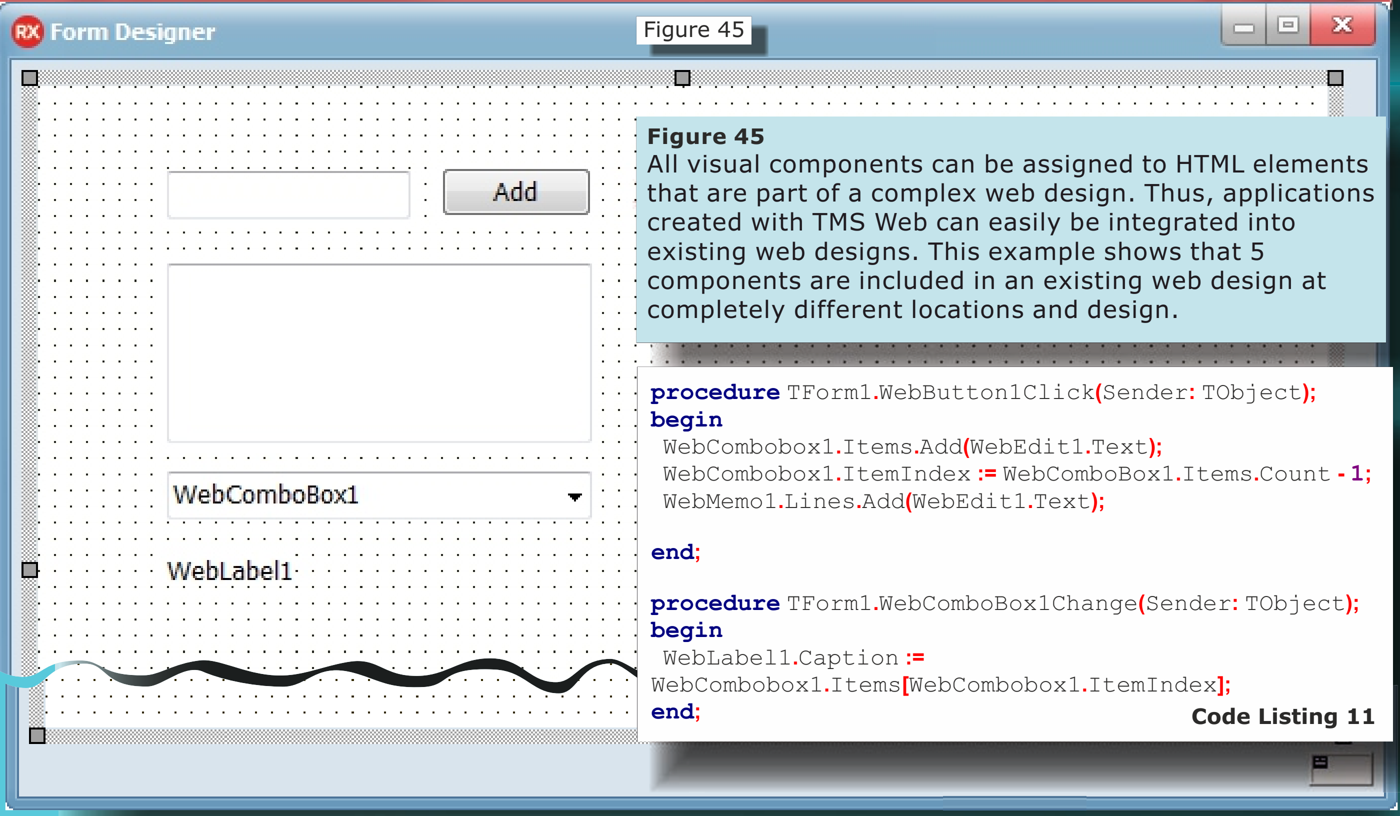
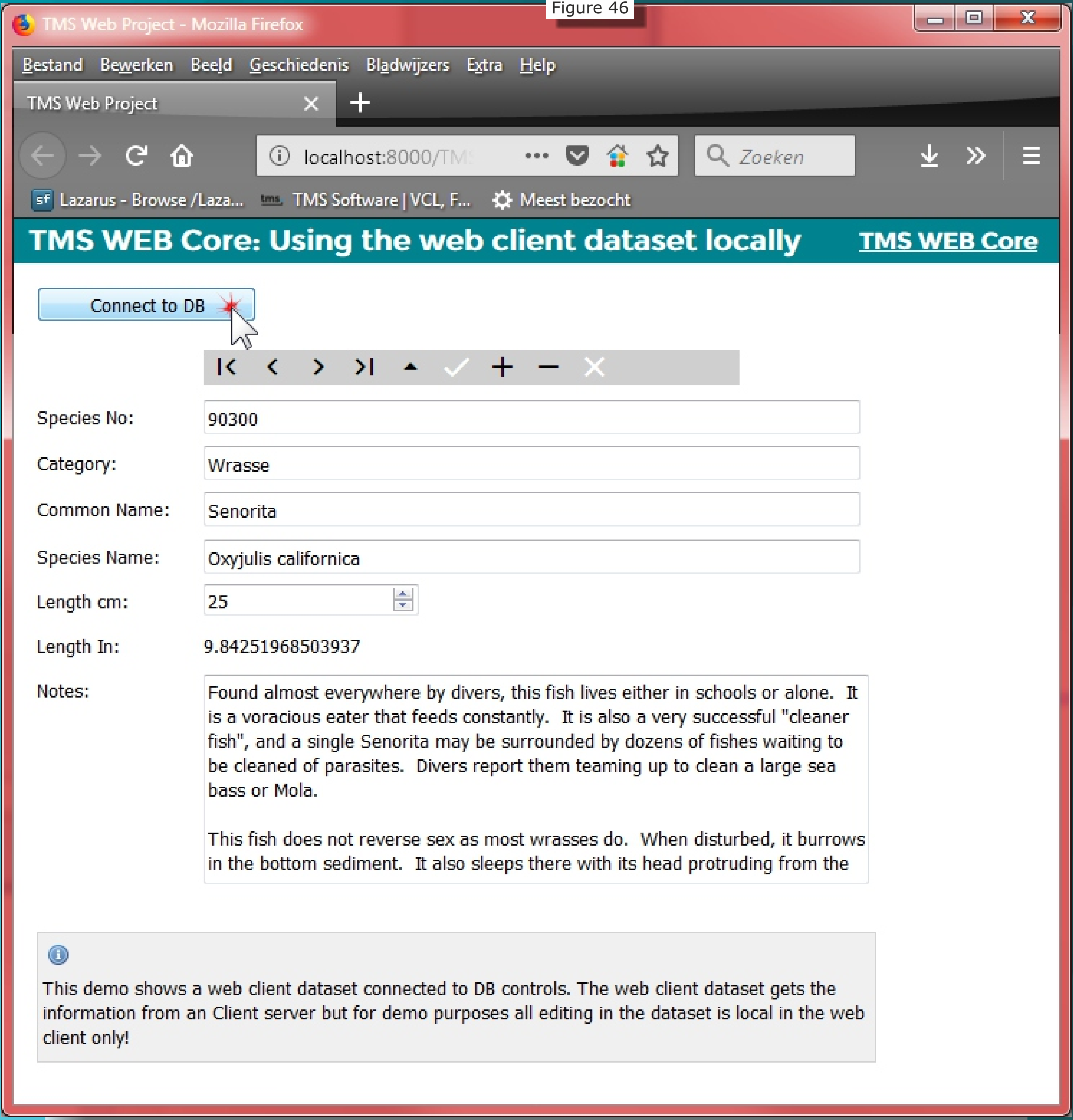


Figure 45
All visual components can be assigned to HTML elements that are part of a complex web design. Thus, applications created with TMS Web can easily be integrated into existing web designs. This example shows that 5 components are included in an existing web design at completely different locations and design.

```
procedure TForm1.WebButton1Click(Sender: TObject);
begin
 WebCombobox1.Items.Add(WebEdit1.Text);
 WebCombobox1.ItemIndex := WebComboBox1.Items.Count - 1;
 WebMemo1.Lines.Add(WebEdit1.Text);

end;

procedure TForm1.WebComboBox1Change(Sender: TObject);
begin
 WebLabel1.Caption :=
WebCombobox1.Items[WebCombobox1.ItemIndex];
end;
```

Code Listing 11

Figure 46


TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS Web Project
localhost:8000/TM
Zoeken
Lazarus - Browse /Laza...
TMS Software | VCL, F...
Meest bezocht
TMS WEB Core
Connect to DB

TMS WEB Core: Using the web client dataset locally

Species No:	90300
Category:	Wrasse
Common Name:	Senorita
Species Name:	Oxyjulis californica
Length cm:	25
Length In:	9.84251968503937
Notes:	Found almost everywhere by divers, this fish lives either in schools or alone. It is a voracious eater that feeds constantly. It is also a very successful "cleaner fish", and a single Senorita may be surrounded by dozens of fishes waiting to be cleaned of parasites. Divers report them teaming up to clean a large sea bass or Mola. This fish does not reverse sex as most wrasses do. When disturbed, it burrows in the bottom sediment. It also sleeps there with its head protruding from the

This demo shows a web client dataset connected to DB controls. The web client dataset gets the information from an Client server but for demo purposes all editing in the dataset is local in the web client only!

Figure 46
TMS Web can be used with all sorts of datasources. Its architecture is designed in particular for REST datasources. Components to consume REST web services are included in Web Core. Furthermore, you can directly use XData web services with database base components. Using XData makes it easy to implement web forms that not only display data, but also allow the user to add, edit and delete data. The communication with the web service is completely handled by the framework. You can concentrate on using Delphi components on the form and will not have to deal with tricky communication and protocol implementation scenarios.

TMSWeb_Dataset.dproj

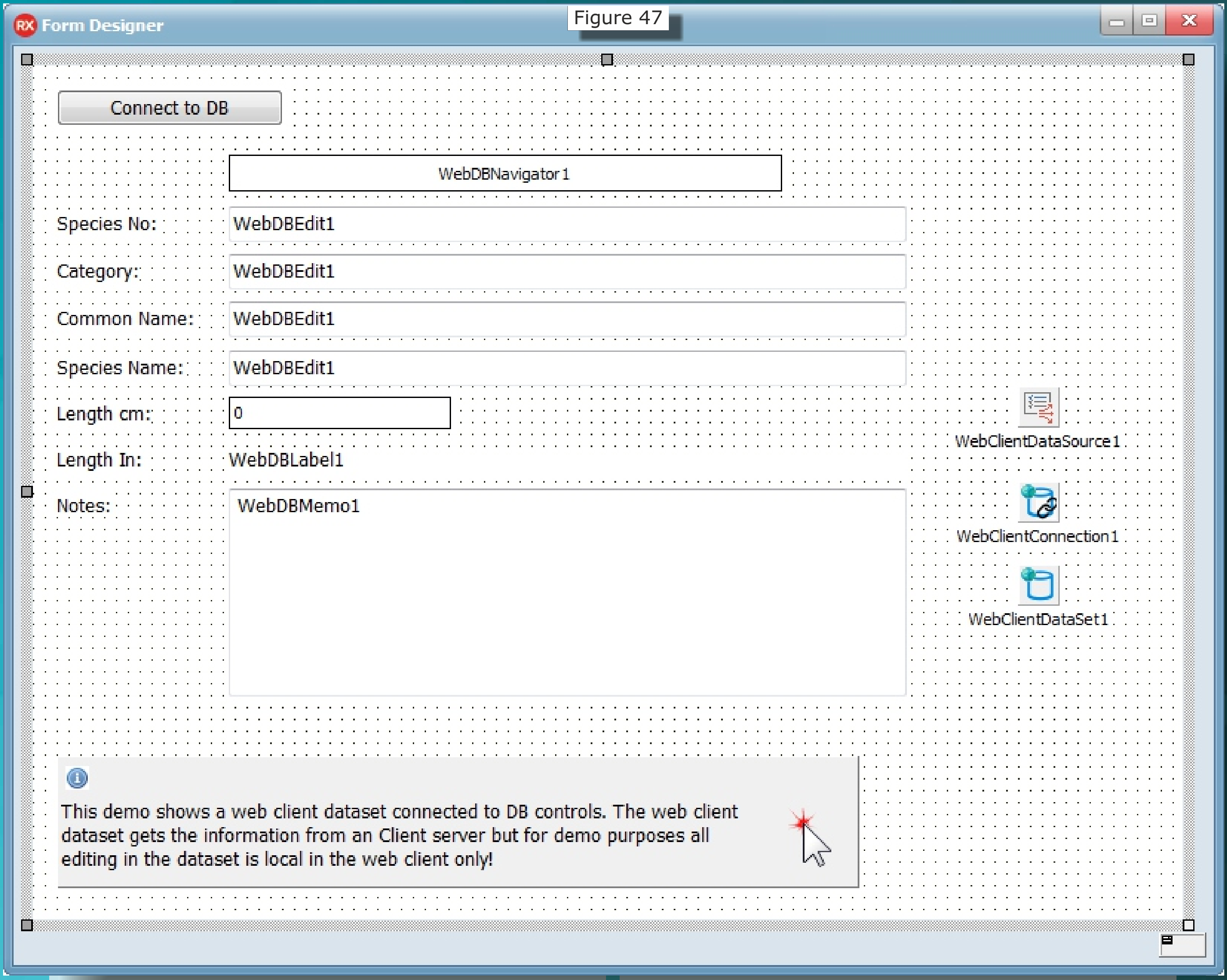
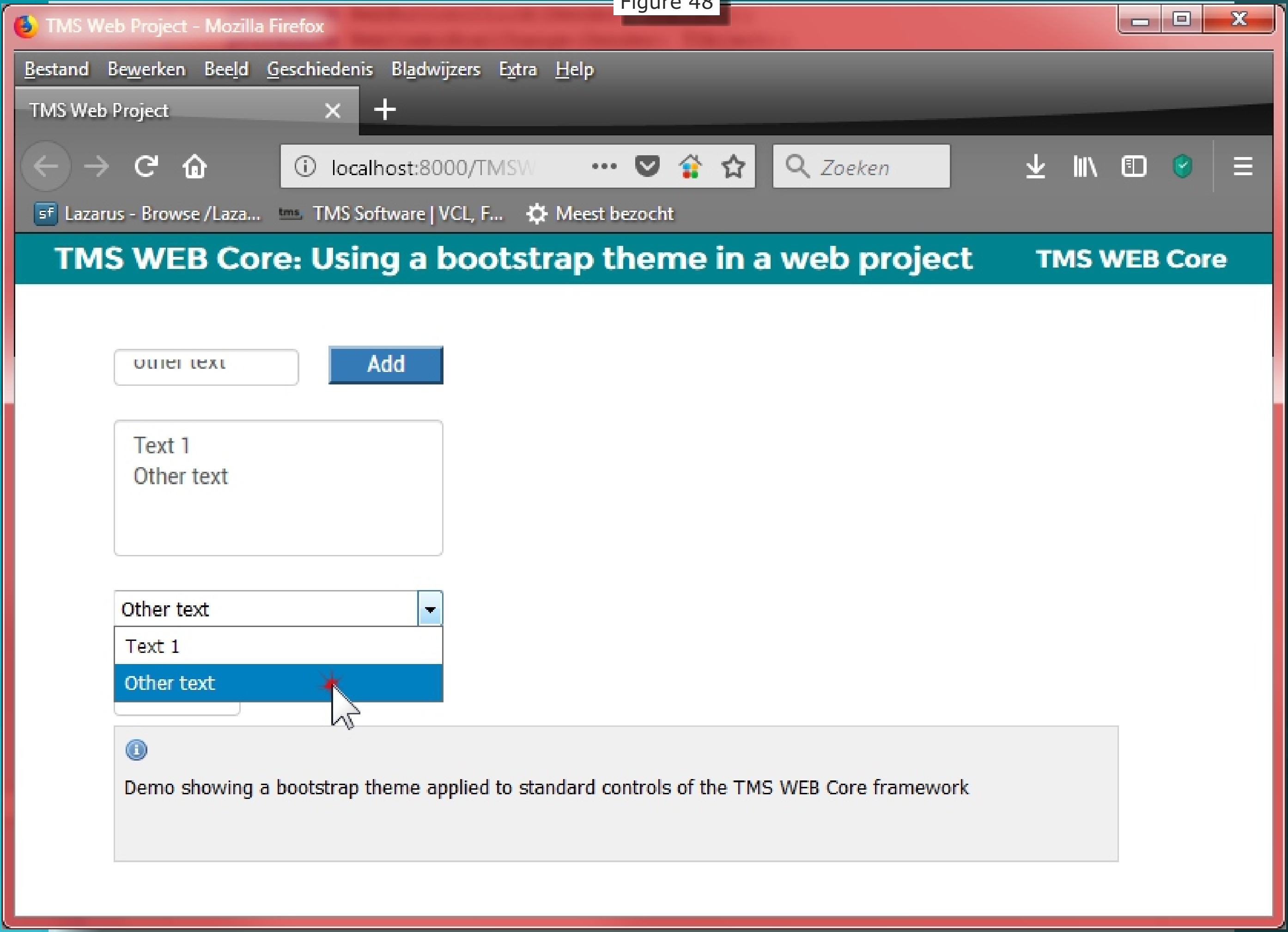


Figure 47

This demo shows a web client dataset connected to DB controls. The web client dataset gets the information from an Client server but for demo purposes all editing in the dataset is local in the web client only!

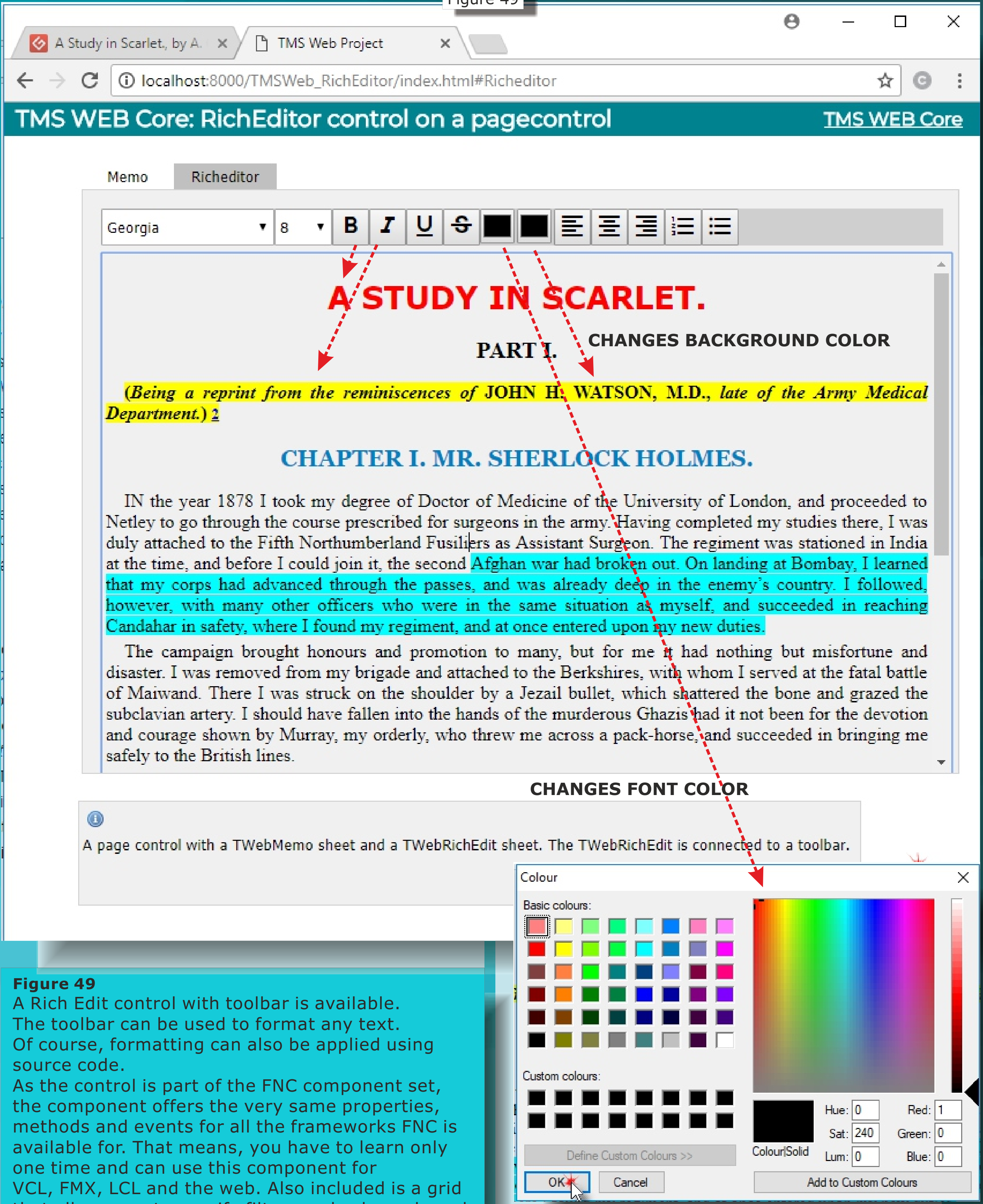
Figure 48


TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS Web Project
localhost:8000/TMSW
Zoeken
Lazarus - Browse /Laza...
TMS Software | VCL, F...
Meest bezocht
TMS WEB Core: Using a bootstrap theme in a web project
TMS WEB Core
other text
Add
Text 1
Other text
Other text
Text 1
Other text

Demo showing a bootstrap theme applied to standard controls of the TMS WEB Core framework

Figure 48
Interoperability with other JavaScript frameworks is unrestricted. You may import any JavaScript framework into your web project. This demo shows how to use Bootstrap to modify the design of the standard web controls. Litsbox and combobox get a different style using Bootstrap. Furthermore, additional styling is applied to the buttons on the form.

Figure 49



A STUDY IN SCARLET.

PART I.

(Being a reprint from the reminiscences of JOHN H. WATSON, M.D., *late of the Army Medical Department.)* 2

CHAPTER I. MR. SHERLOCK HOLMES.

IN the year 1878 I took my degree of Doctor of Medicine of the University of London, and proceeded to Netley to go through the course prescribed for surgeons in the army. Having completed my studies there, I was duly attached to the Fifth Northumberland Fusiliers as Assistant Surgeon. The regiment was stationed in India at the time, and before I could join it, the second Afghan war had broken out. On landing at Bombay, I learned that my corps had advanced through the passes, and was already deep in the enemy's country. I followed, however, with many other officers who were in the same situation as myself, and succeeded in reaching Candahar in safety, where I found my regiment, and at once entered upon my new duties.

The campaign brought honours and promotion to many, but for me it had nothing but misfortune and disaster. I was removed from my brigade and attached to the Berkshires, with whom I served at the fatal battle of Maiwand. There I was struck on the shoulder by a Jezail bullet, which shattered the bone and grazed the subclavian artery. I should have fallen into the hands of the murderous Ghazis had it not been for the devotion and courage shown by Murray, my orderly, who threw me across a pack-horse, and succeeded in bringing me safely to the British lines.

A page control with a TWebMemo sheet and a TWebRichEdit sheet. The TWebRichEdit is connected to a toolbar.

Figure 49
A Rich Edit control with toolbar is available.
The toolbar can be used to format any text.
Of course, formatting can also be applied using source code.
As the control is part of the FNC component set, the component offers the very same properties, methods and events for all the frameworks FNC is available for. That means, you have to learn only one time and can use this component for VCL, FMX, LCL and the web. Also included is a grid that allows you to specify filters and column-based styling. The filters are either offered automatically as drop-down lists or can be specified using a special expression syntax. Different cell styles are also supported, most notably checkboxes can be applied.

`TMSWeb_RichEditor.dproj`

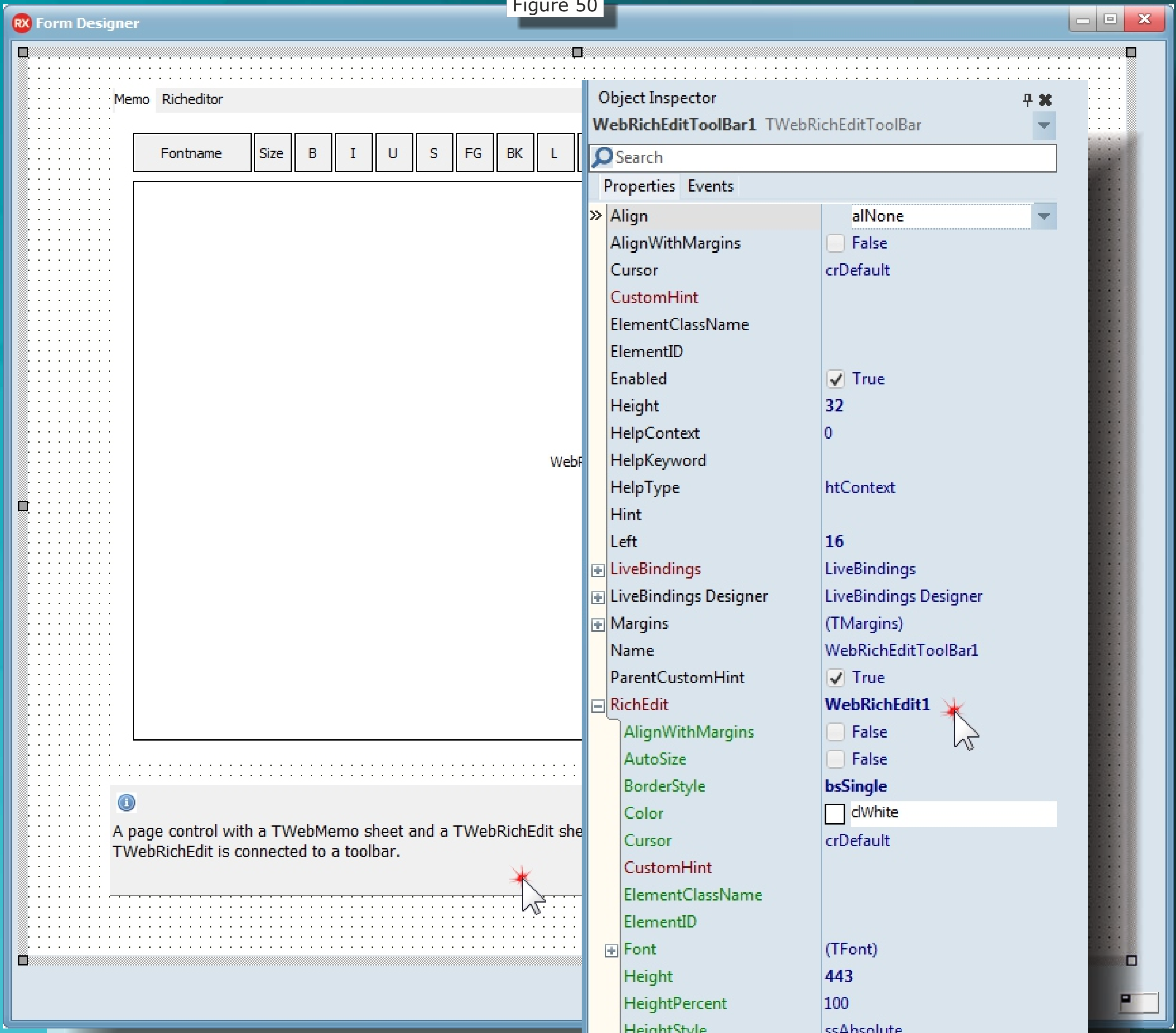
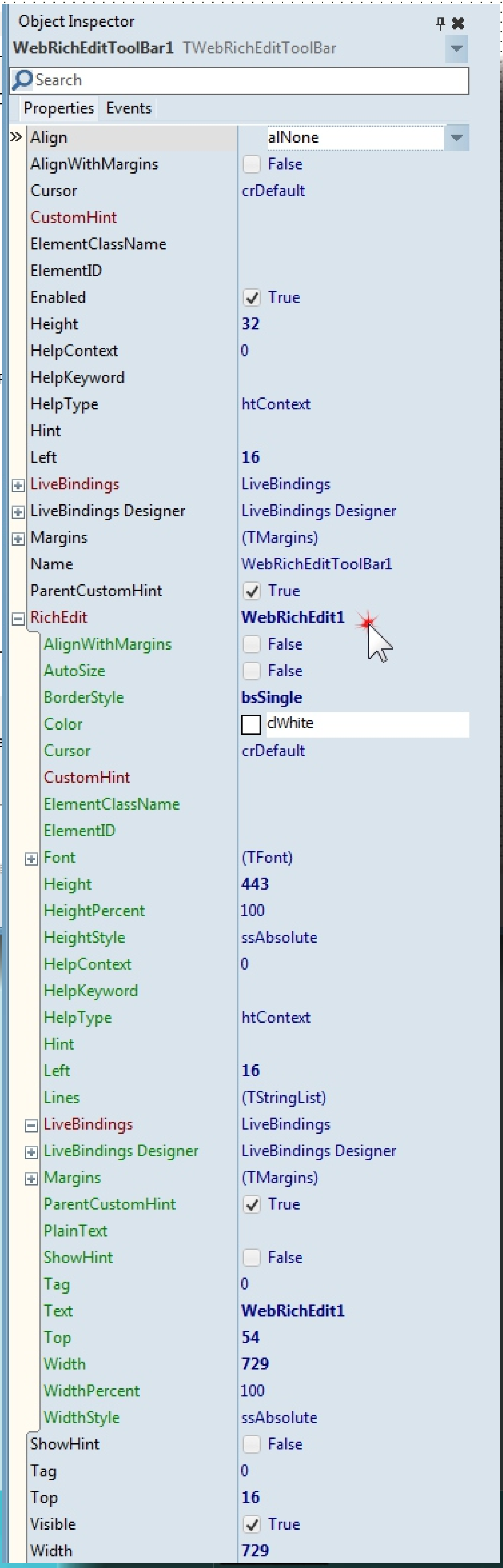


Figure 50

A page control with a TWebMemo sheet and a TWebRichEdit she
TWebRichEdit is connected to a toolbar.



Object Inspector

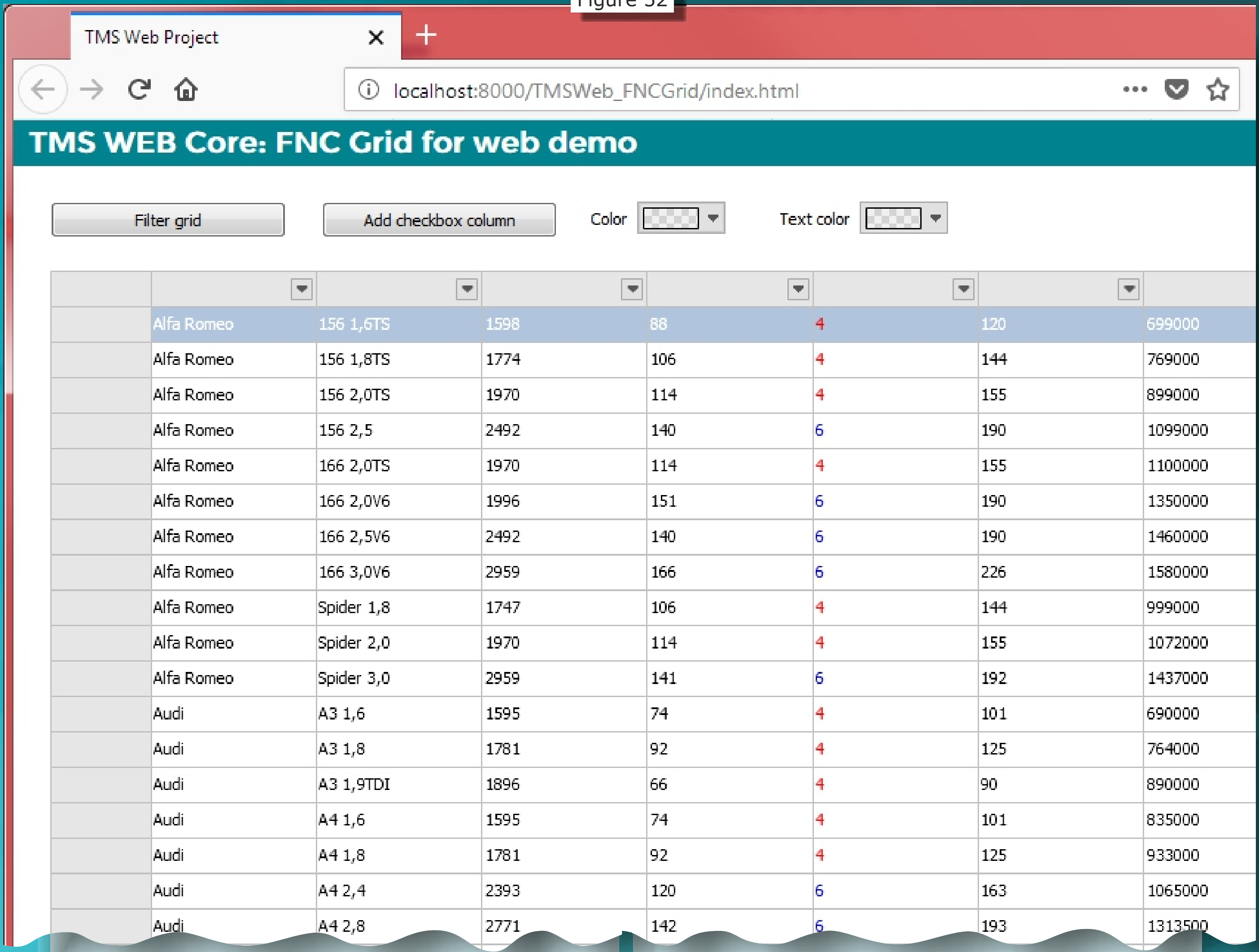
WebRichEditToolBar1 TWebRichEditToolBar

Align	alNone
AlignWithMargins	False
Cursor	crDefault
CustomHint	
ElementClassName	
ElementID	
Enabled	✓ True
Height	32
HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
Left	16
LiveBindings	LiveBindings
LiveBindings Designer	LiveBindings Designer
Margins	(TMargins)
Name	WebRichEditToolBar1
ParentCustomHint	✓ True
RichEdit	WebRichEdit1
AlignWithMargins	False
AutoSize	False
BorderStyle	bsSingle
Color	clWhite
Cursor	crDefault
CustomHint	
ElementClassName	
ElementID	
Font	(TFont)
Height	443
HeightPercent	100
HeightStyle	ssAbsolute
HelpContext	0
HelpKeyword	
HelpType	htContext
Hint	
Left	16
Lines	(TStringList)
LiveBindings	LiveBindings
LiveBindings Designer	LiveBindings Designer
Margins	(TMargins)
ParentCustomHint	✓ True
PlainText	
ShowHint	False
Tag	0
Text	WebRichEdit1
Top	54
Width	729
WidthPercent	100
WidthStyle	ssAbsolute
ShowHint	False
Tag	0
Top	16
Visible	✓ True
Width	729

Figure 51

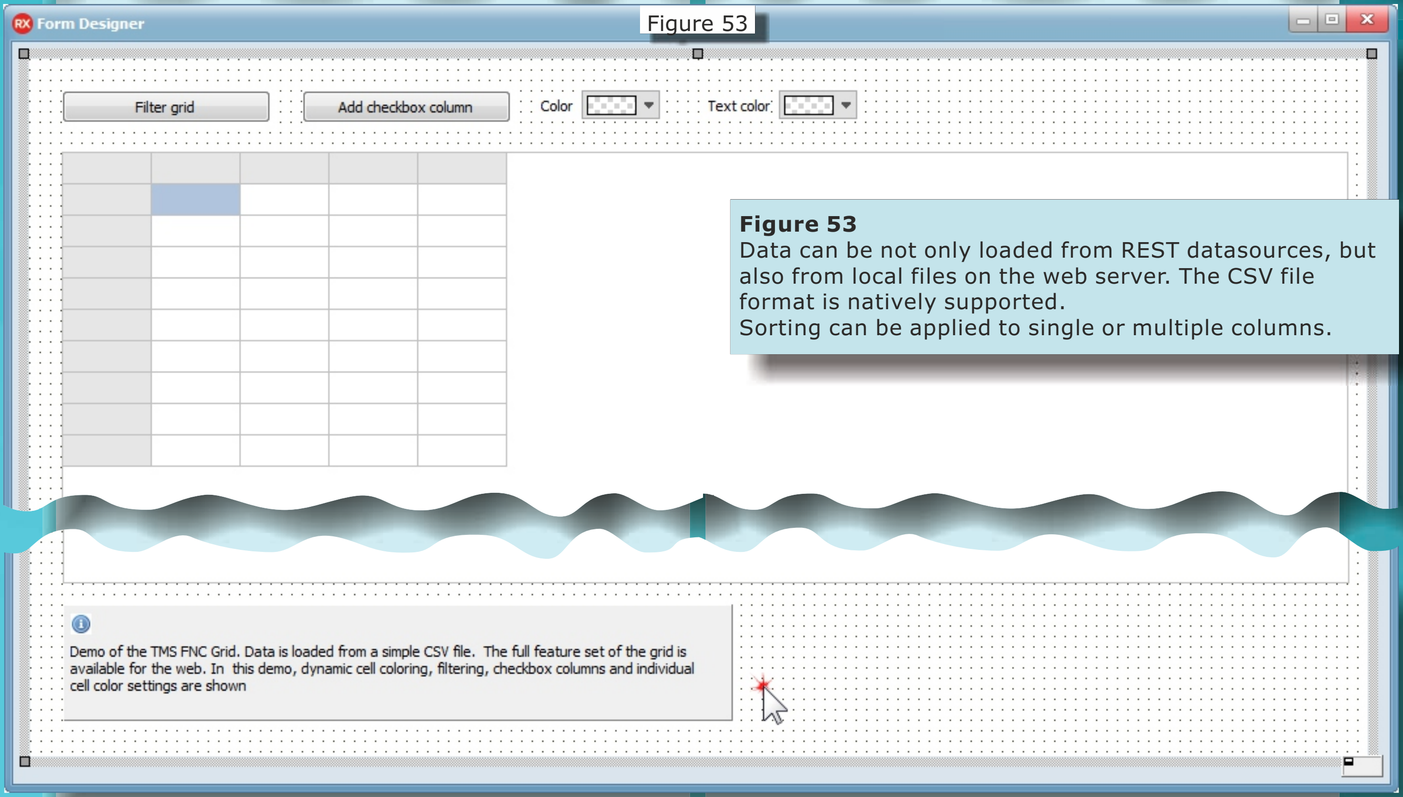
TMSWeb_FNCGrid.dproj

Figure 52


TMS Web Project
localhost:8000/TMSWeb_FNCGrid/index.html
TMS WEB Core: FNC Grid for web demo
Filter grid
Add checkbox column
Color
Text color

	Alfa Romeo	156 1,6TS	1598	88	4	120	699000
	Alfa Romeo	156 1,8TS	1774	106	4	144	769000
	Alfa Romeo	156 2,0TS	1970	114	4	155	899000
	Alfa Romeo	156 2,5	2492	140	6	190	1099000
	Alfa Romeo	166 2,0TS	1970	114	4	155	1100000
	Alfa Romeo	166 2,0V6	1996	151	6	190	1350000
	Alfa Romeo	166 2,5V6	2492	140	6	190	1460000
	Alfa Romeo	166 3,0V6	2959	166	6	226	1580000
	Alfa Romeo	Spider 1,8	1747	106	4	144	999000
	Alfa Romeo	Spider 2,0	1970	114	4	155	1072000
	Alfa Romeo	Spider 3,0	2959	141	6	192	1437000
	Audi	A3 1,6	1595	74	4	101	690000
	Audi	A3 1,8	1781	92	4	125	764000
	Audi	A3 1,9TDI	1896	66	4	90	890000
	Audi	A4 1,6	1595	74	4	101	835000
	Audi	A4 1,8	1781	92	4	125	933000
	Audi	A4 2,4	2393	120	6	163	1065000
	Audi	A4 2,8	2771	142	6	193	1313500

Figure 53



Demo of the TMS FNC Grid. Data is loaded from a simple CSV file. The full feature set of the grid is available for the web. In this demo, dynamic cell coloring, filtering, checkbox columns and individual cell color settings are shown

Figure 53
Data can be not only loaded from REST datasources, but also from local files on the web server. The CSV file format is natively supported.
Sorting can be applied to single or multiple columns.

TMSWeb_FNCGrid.dproj

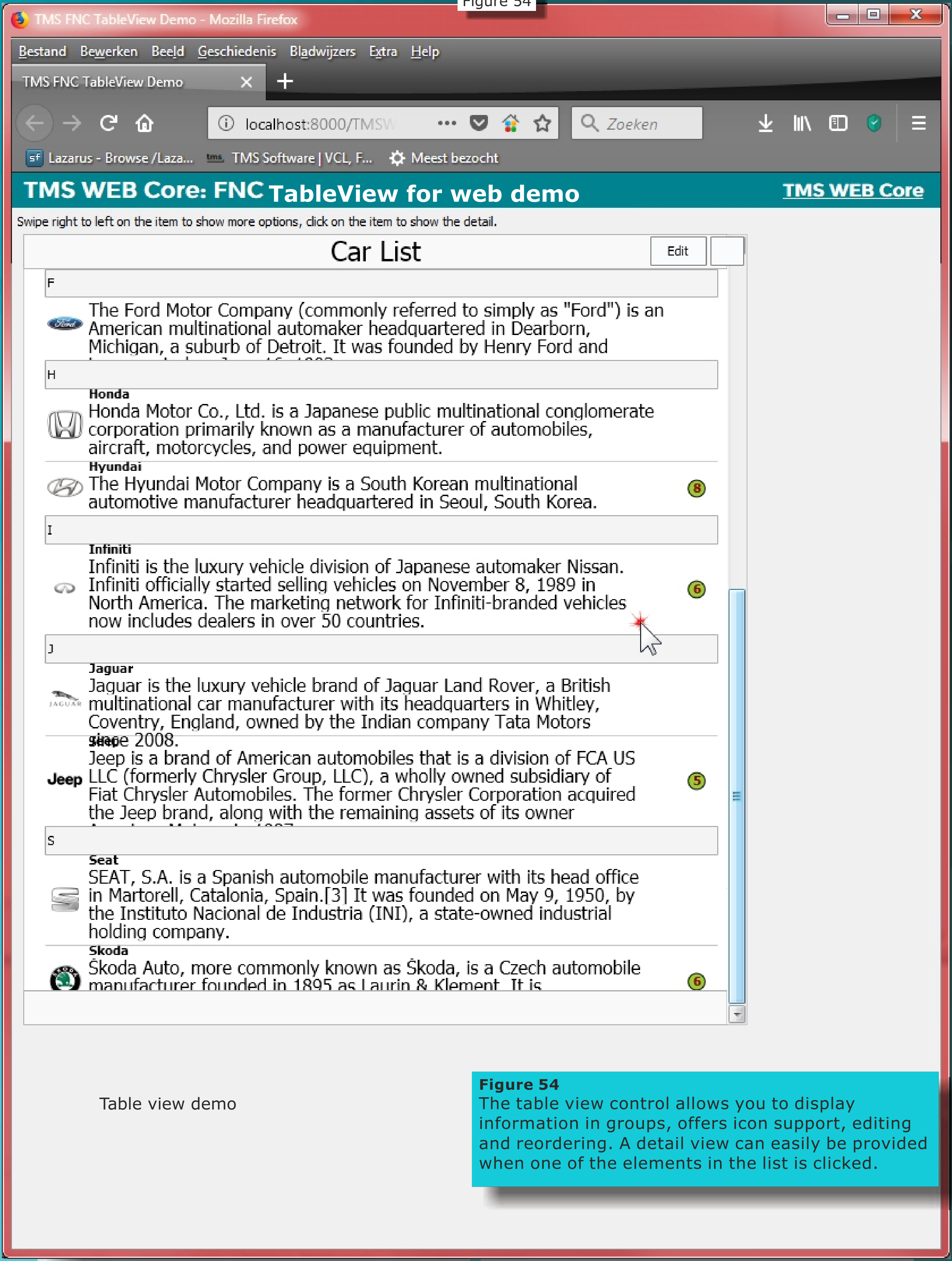
```
procedure TForm4.btnFilterClick(Sender: TObject);
var
 fd: TTMSFNCGridFilterData;
begin
 if not filtered then
 begin
  filtered := true;
  TMSFNCGrid1.Filter.Clear;
  fd := TMSFNCGrid1.Filter.Add;
  fd.Condition := 'B* | M*';
  fd.Column := 1;
  fd.CaseSensitive := false;
  TMSFNCGrid1.ApplyFilter;
  btnFilter.Caption := 'Remove filter';
 end
 else
 begin
  filtered := false;
  TMSFNCGrid1.RemoveFilter;
  btnFilter.Caption := 'Filter grid';
 end;
end;
```

Code Listing 12

```
procedure TForm4.TMSFNCGrid1SelectCell(Sender: TObject; ACol, ARow: Integer;
 var Allow: Boolean);
var
 rc,rr: integer;
begin
 rc := TMSFNCGrid1.FocusedCell.Col;
 rr := TMSFNCGrid1.FocusedCell.Row;
 rr := TMSFNCGrid1.DisplToRealRow(rr);

 TMSFNCColorPicker1.SelectedColor := TMSFNCGrid1.Colors[rc, rr];
 TMSFNCColorPicker2.SelectedColor := TMSFNCGrid1.FontColors[rc, rr];
end;

procedure TForm4.WebFormCreate(Sender: TObject);
begin
 filtered := false;
 {$IFNDEF WIN32}
 TMSFNCGrid1.LoadFromCSV('http://www.tmssoftware.biz/tmsweb/cars.csv');
 {$ENDIF}
 TMSFNCGrid1.Options.Sorting.Mode := gsmNormal;
 TMSFNCGrid1.Options.Filtering.DropDown := True;
 TMSFNCGrid1.Options.Selection.Mode := smRowRange;
 TMSFNCGrid1.Options.Editing.Enabled := true;
 TMSFNCGrid1.Options.ColumnSize.Stretch := True;
 TMSFNCGrid1.Options.ColumnSize.StretchAll := True;
end;
```

Code Listing 13

Figure 54


TMS FNC TableView Demo - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS FNC TableView Demo
localhost:8000/TMSW
Zoeken
Lazarus - Browse /Laza...
TMS Software | VCL, F...
Meest bezocht
TMS WEB Core: FNC TableView for web demo
TMS WEB Core
Car List
Edit
F
H
I
J
S
8
6
5
6
Jeep

Swipe right to left on the item to show more options, click on the item to show the detail.

The Ford Motor Company (commonly referred to simply as "Ford") is an American multinational automaker headquartered in Dearborn, Michigan, a suburb of Detroit. It was founded by Henry Ford and

Honda
Honda Motor Co., Ltd. is a Japanese public multinational conglomerate corporation primarily known as a manufacturer of automobiles, aircraft, motorcycles, and power equipment.

Hyundai
The Hyundai Motor Company is a South Korean multinational automotive manufacturer headquartered in Seoul, South Korea.

Infiniti
Infiniti is the luxury vehicle division of Japanese automaker Nissan. Infiniti officially started selling vehicles on November 8, 1989 in North America. The marketing network for Infiniti-branded vehicles now includes dealers in over 50 countries.

Jaguar
Jaguar is the luxury vehicle brand of Jaguar Land Rover, a British multinational car manufacturer with its headquarters in Whitley, Coventry, England, owned by the Indian company Tata Motors since 2008.

Jeep
Jeep is a brand of American automobiles that is a division of FCA US LLC (formerly Chrysler Group, LLC), a wholly owned subsidiary of Fiat Chrysler Automobiles. The former Chrysler Corporation acquired the Jeep brand, along with the remaining assets of its owner

Seat
SEAT, S.A. is a Spanish automobile manufacturer with its head office in Martorell, Catalonia, Spain.[3] It was founded on May 9, 1950, by the Instituto Nacional de Industria (INI), a state-owned industrial holding company.

Skoda
Škoda Auto, more commonly known as Škoda, is a Czech automobile manufacturer founded in 1895 as Laurin & Klement. It is

Table view demo

Figure 54
The table view control allows you to display information in groups, offers icon support, editing and reordering. A detail view can easily be provided when one of the elements in the list is clicked.

TMSWeb_FNCTableView.dproj

Figure 55

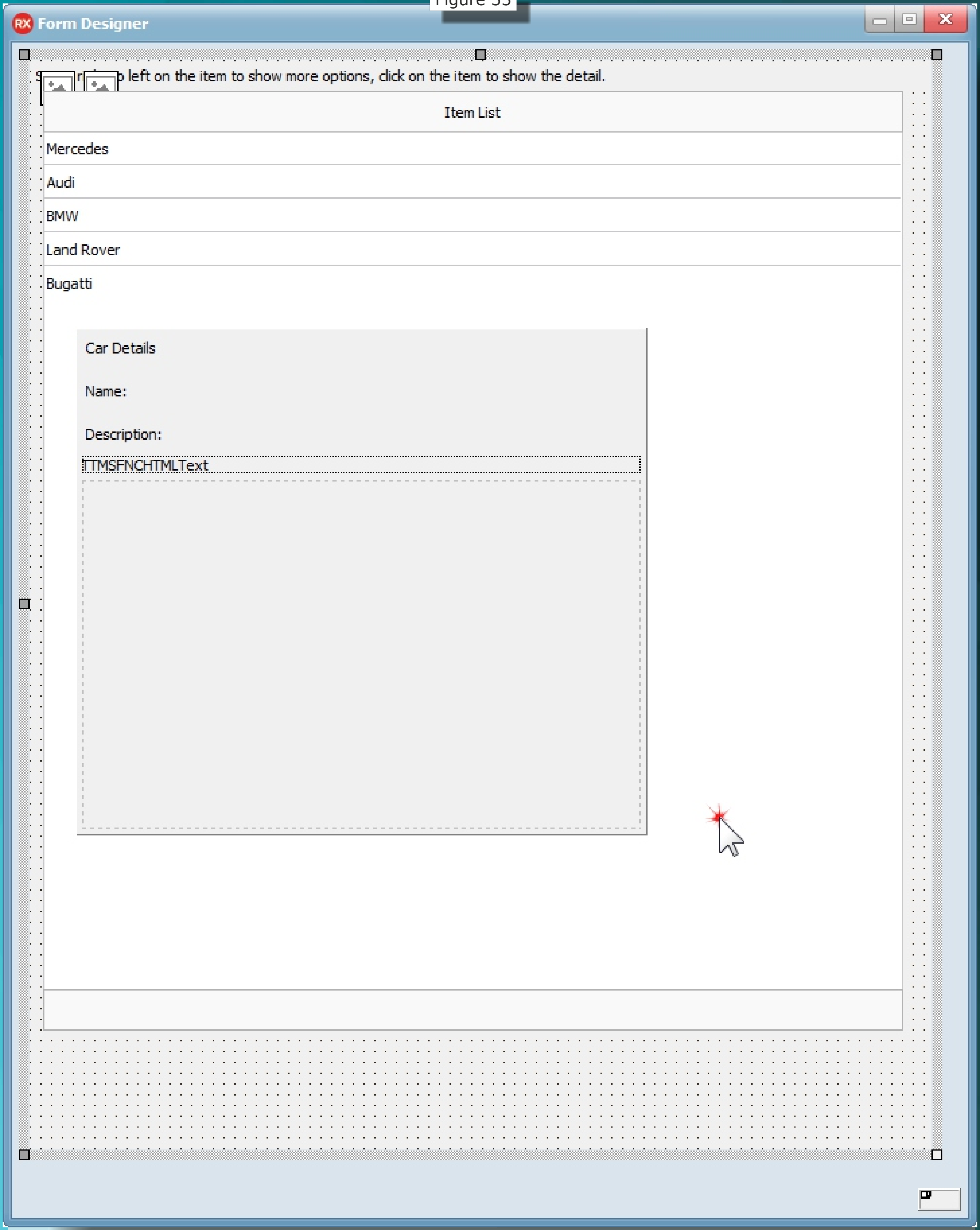
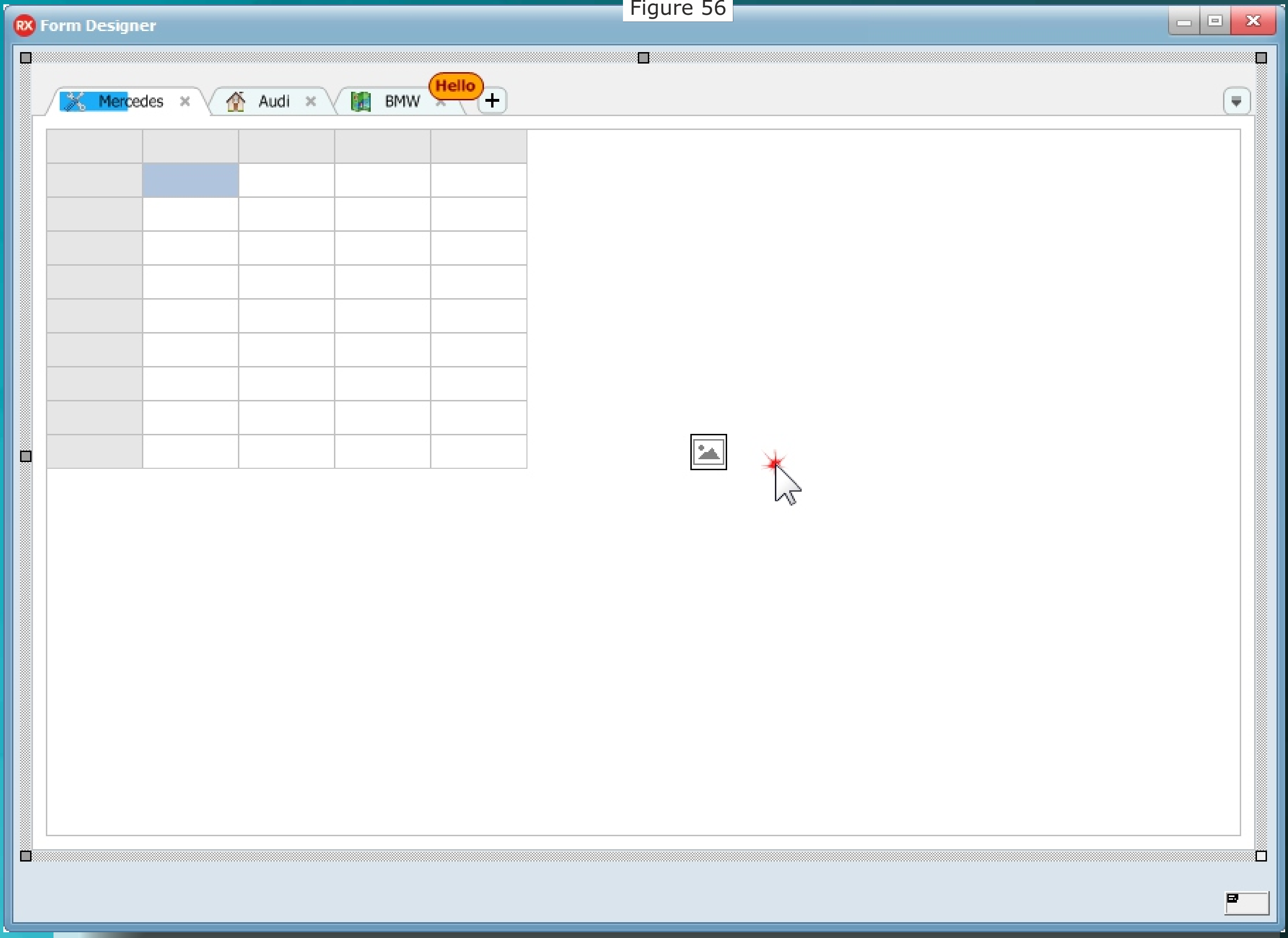


Figure 55.
This form shows the table view in the background and the detail view that is shown whenever an item is selected. A panel hosting all the other visual controls is used as a base component for the detail view and is directly linked to the table view. The component allows you to specify the data for the detail view before it is shown. This works completely the same way as it works on the desktop or mobile platforms.

TMSWeb_FNCTabSet_PageControl.dproj

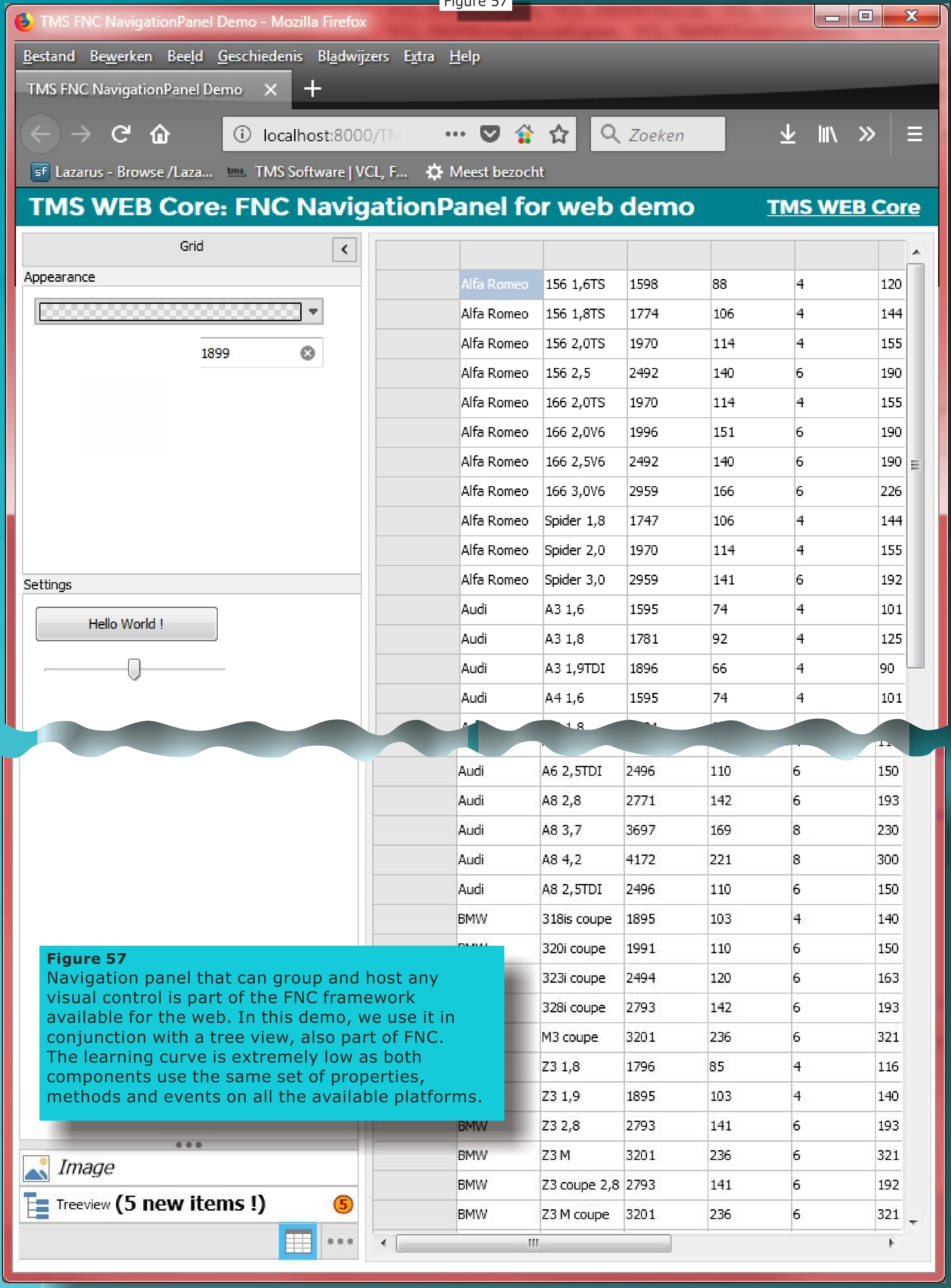
Figure 56



Form 56:
A page control with fully customizable tabs is included. The tabs can show icons and badges. The tabs can host any visual control and thus allows for very complex web forms with very little source code. Of course, the page controls supports the Anchor and Align property and is thus the perfect choice for responsive web design.

TMSWeb_FNCNavigationPanel.dproj

Figure 57

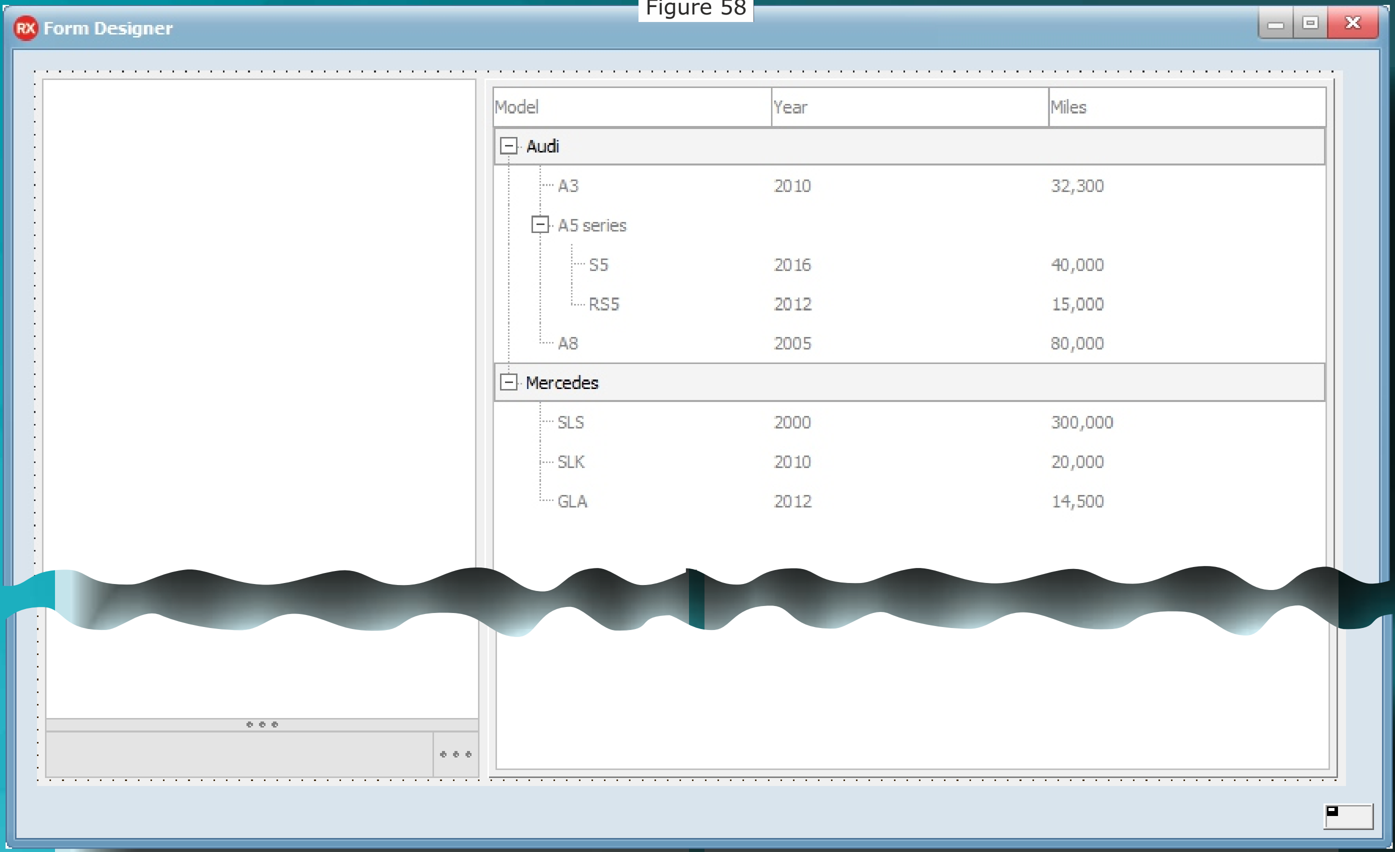

TMS FNC NavigationPanel Demo - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS FNC NavigationPanel Demo
localhost:8000/TM
Zoeken
Lazarus - Browse /Laza...
TMS Software | VCL, F...
Meest bezocht
TMS WEB Core: FNC NavigationPanel for web demo
TMS WEB Core
Grid
Appearance
1899
Settings
Hello World !
Image
Treeview (5 new items !)
5

Alfa Romeo	156 1,6TS	1598	88	4	120
Alfa Romeo	156 1,8TS	1774	106	4	144
Alfa Romeo	156 2,0TS	1970	114	4	155
Alfa Romeo	156 2,5	2492	140	6	190
Alfa Romeo	166 2,0TS	1970	114	4	155
Alfa Romeo	166 2,0V6	1996	151	6	190
Alfa Romeo	166 2,5V6	2492	140	6	190
Alfa Romeo	166 3,0V6	2959	166	6	226
Alfa Romeo	Spider 1,8	1747	106	4	144
Alfa Romeo	Spider 2,0	1970	114	4	155
Alfa Romeo	Spider 3,0	2959	141	6	192
Audi	A3 1,6	1595	74	4	101
Audi	A3 1,8	1781	92	4	125
Audi	A3 1,9TDI	1896	66	4	90
Audi	A4 1,6	1595	74	4	101
Audi	A6 2,5TDI	2496	110	6	150
Audi	A8 2,8	2771	142	6	193
Audi	A8 3,7	3697	169	8	230
Audi	A8 4,2	4172	221	8	300
Audi	A8 2,5TDI	2496	110	6	150
BMW	318is coupe	1895	103	4	140
	320i coupe	1991	110	6	150
	323i coupe	2494	120	6	163
	328i coupe	2793	142	6	193
	M3 coupe	3201	236	6	321
	Z3 1,8	1796	85	4	116
	Z3 1,9	1895	103	4	140
BMW	Z3 2,8	2793	141	6	193
BMW	Z3 M	3201	236	6	321
BMW	Z3 coupe 2,8	2793	141	6	192
BMW	Z3 M coupe	3201	236	6	321

Figure 57
Navigation panel that can group and host any visual control is part of the FNC framework available for the web. In this demo, we use it in conjunction with a tree view, also part of FNC. The learning curve is extremely low as both components use the same set of properties, methods and events on all the available platforms.

`TMSWeb_FNCNavigationPanel.dproj`

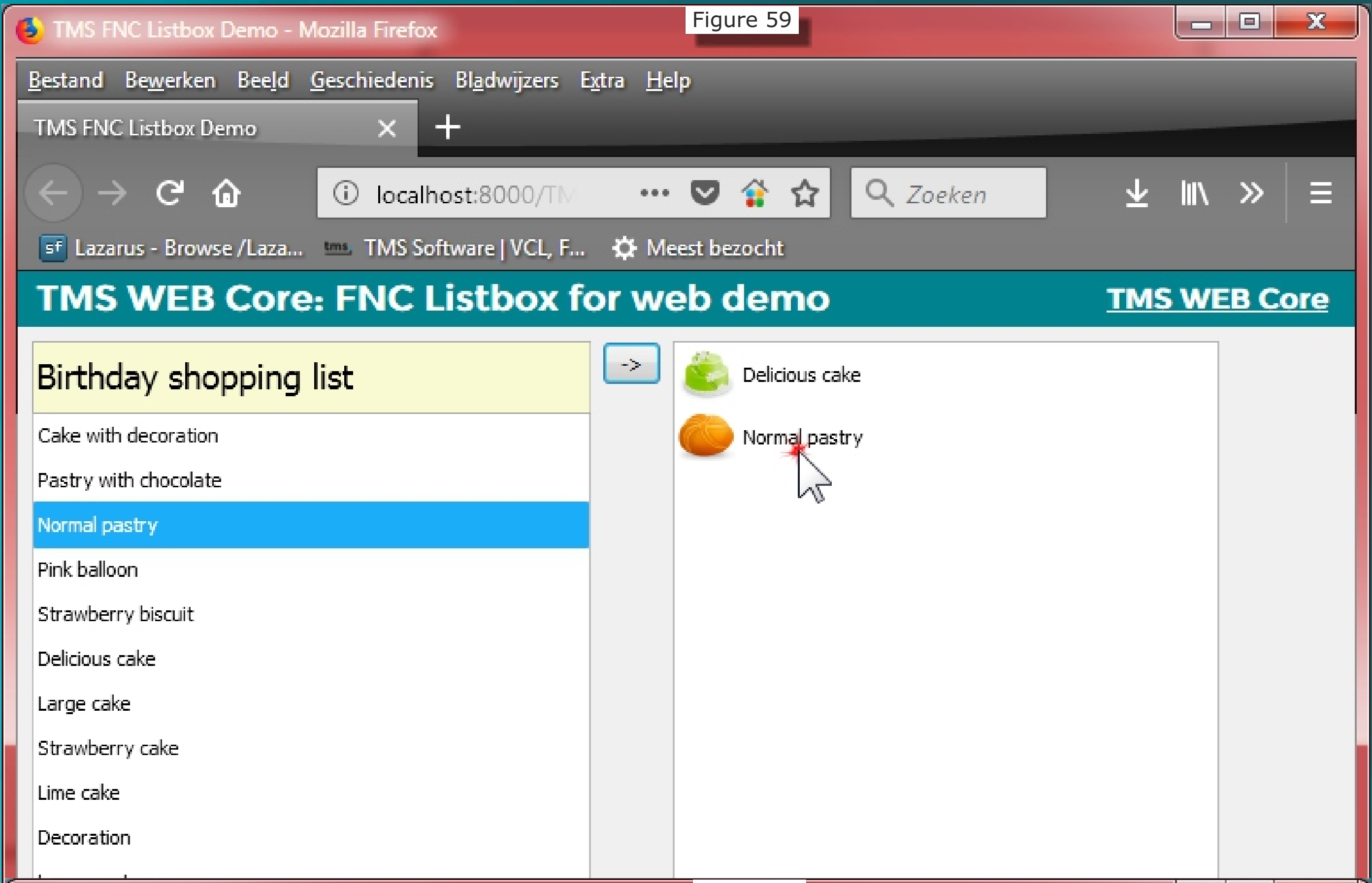
Figure 58



Model	Year	Miles
Audi		
A3	2010	32,300
A5 series		
S5	2016	40,000
RS5	2012	15,000
A8	2005	80,000
Mercedes		
SLS	2000	300,000
SLK	2010	20,000
GLA	2012	14,500

TMSWeb_FNCListbox.dproj

Figure 59


TMS FNC Listbox Demo - Mozilla Firefox

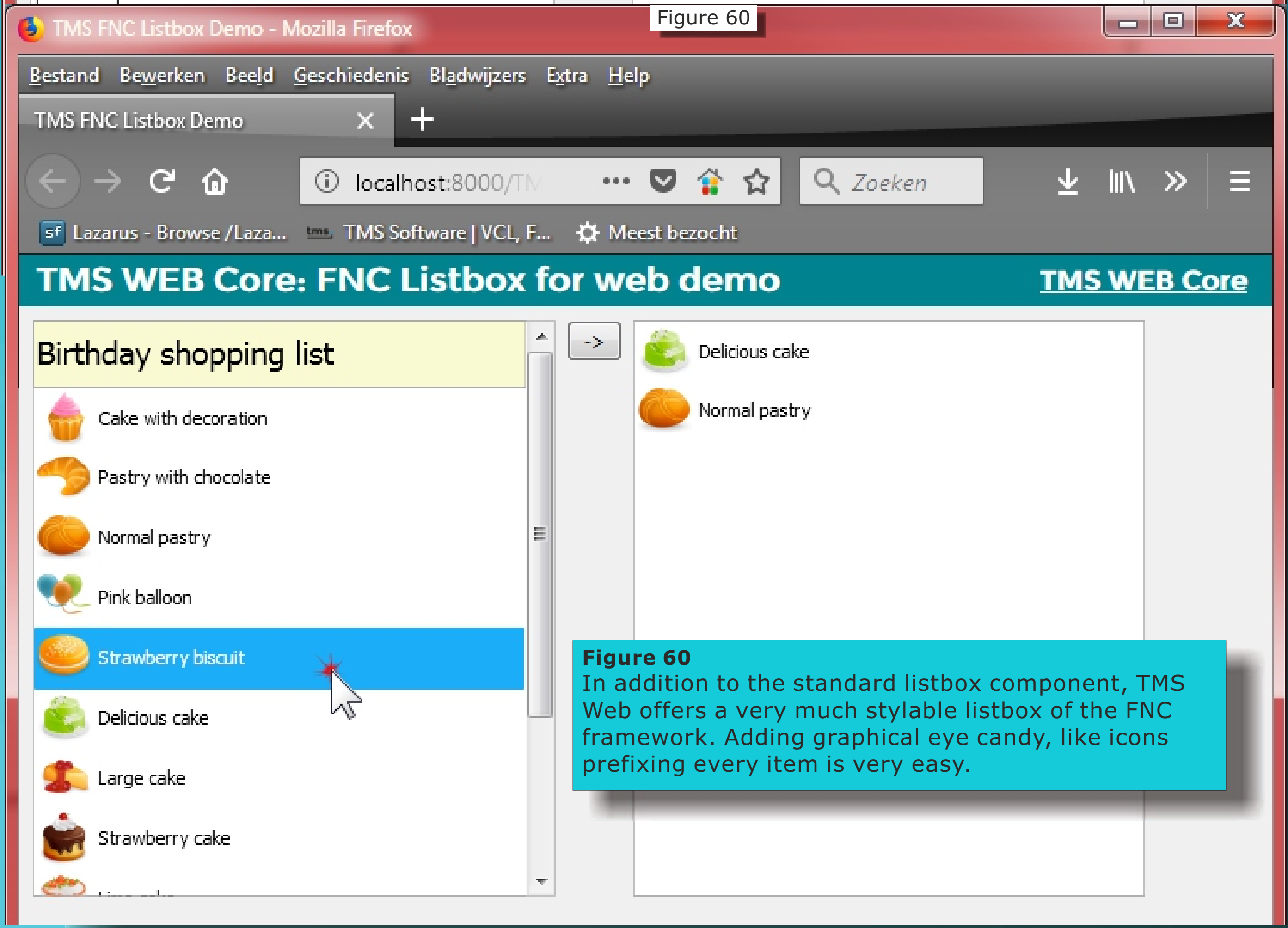
TMS WEB Core: FNC Listbox for web demo

Birthday shopping list

Cake with decoration
Pastry with chocolate
Normal pastry
Pink balloon
Strawberry biscuit
Delicious cake
Large cake
Strawberry cake
Lime cake
Decoration

Delicious cake
Normal pastry

Figure 60


TMS FNC Listbox Demo - Mozilla Firefox

TMS WEB Core: FNC Listbox for web demo

Birthday shopping list

Cake with decoration
Pastry with chocolate
Normal pastry
Pink balloon
Strawberry biscuit
Delicious cake
Large cake
Strawberry cake

Delicious cake
Normal pastry

Figure 60
In addition to the standard listbox component, TMS Web offers a very much stylable listbox of the FNC framework. Adding graphical eye candy, like icons prefixing every item is very easy.

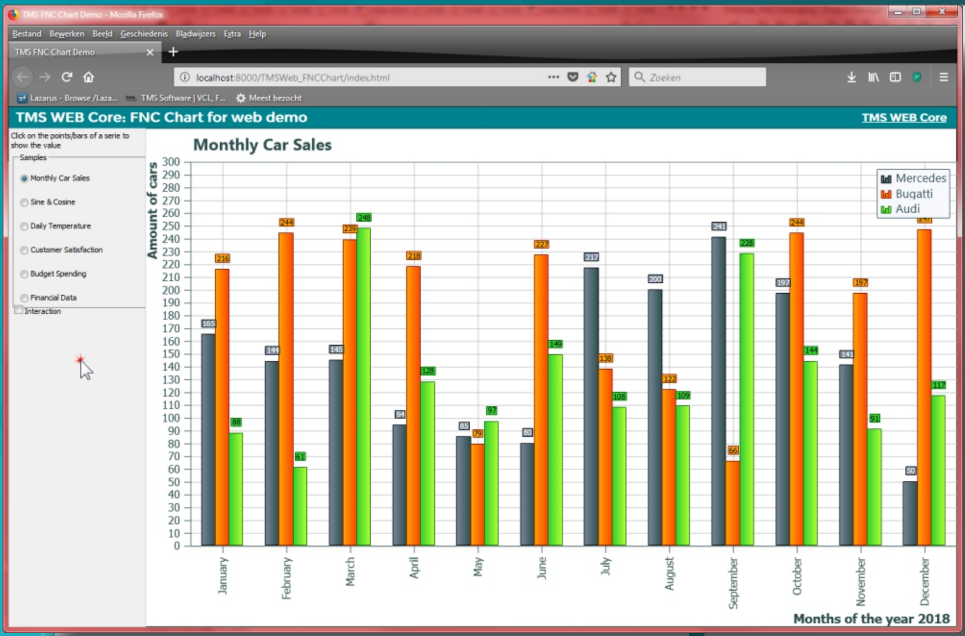


Figure 61

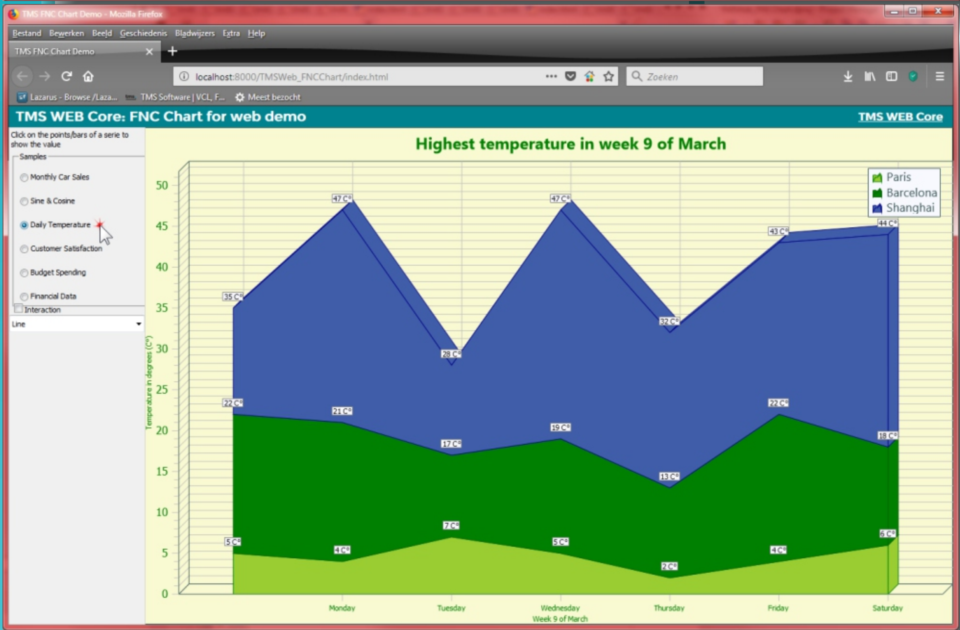


Figure 62

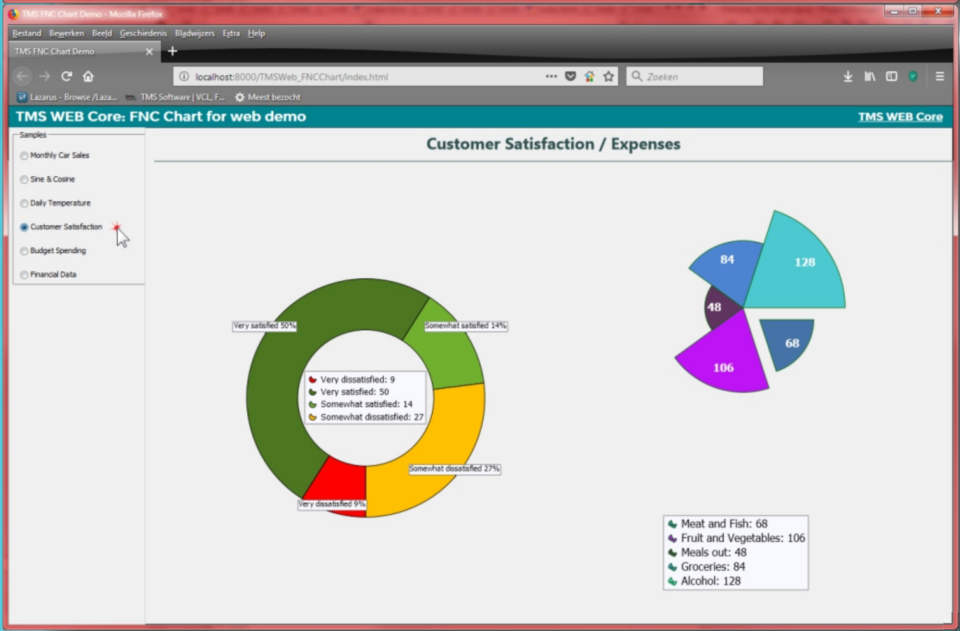
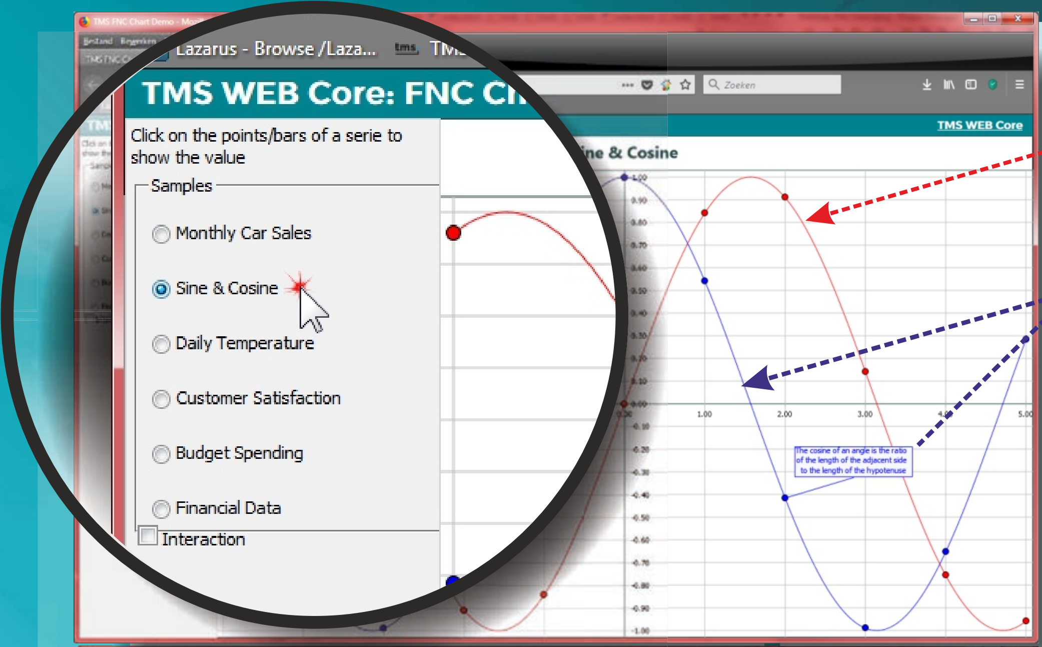


Figure 63


TMS WEB Core: FNC Chart for web demo
Click on the points/bars of a serie to show the value
Samples
Monthly Car Sales
Sine & Cosine
Daily Temperature
Customer Satisfaction
Budget Spending
Financial Data
Interaction
Sine & Cosine
TMS WEB Core
The cosine of an angle is the ratio of the length of the adjacent side to the length of the hypotenuse

The sine of an angle is the ratio of the length of the opposite side to the lengt of the hypotenuse

The cosine of an angle is the ratio of the length of the adjacent side to the lengt of the hypotenuse

Figure 64

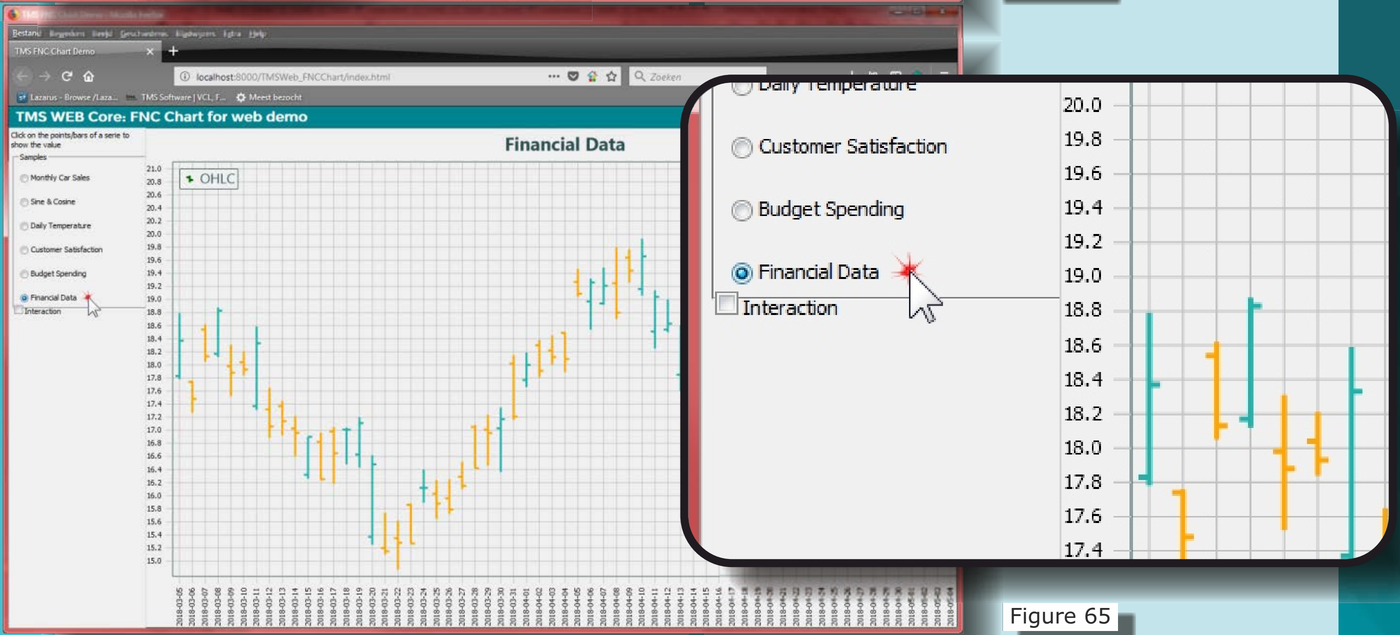


Figure 65

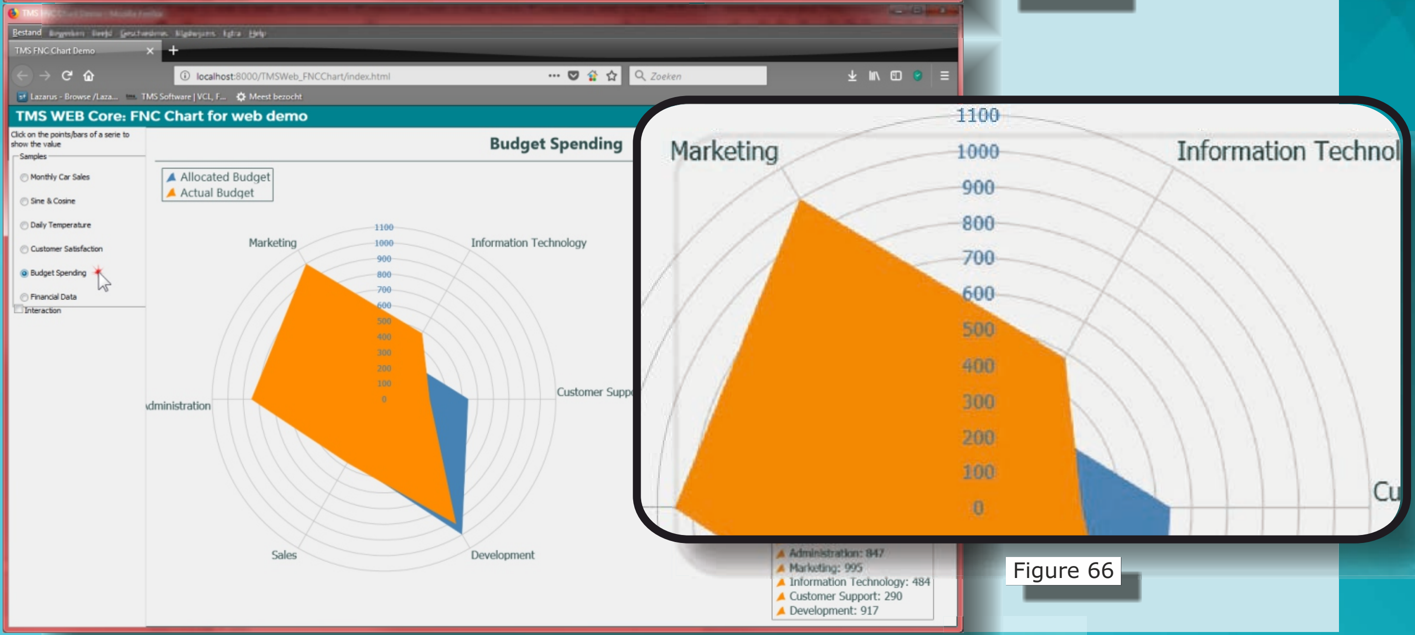


Figure 66

TMSWeb_FNCChart.dproj

Figure 67

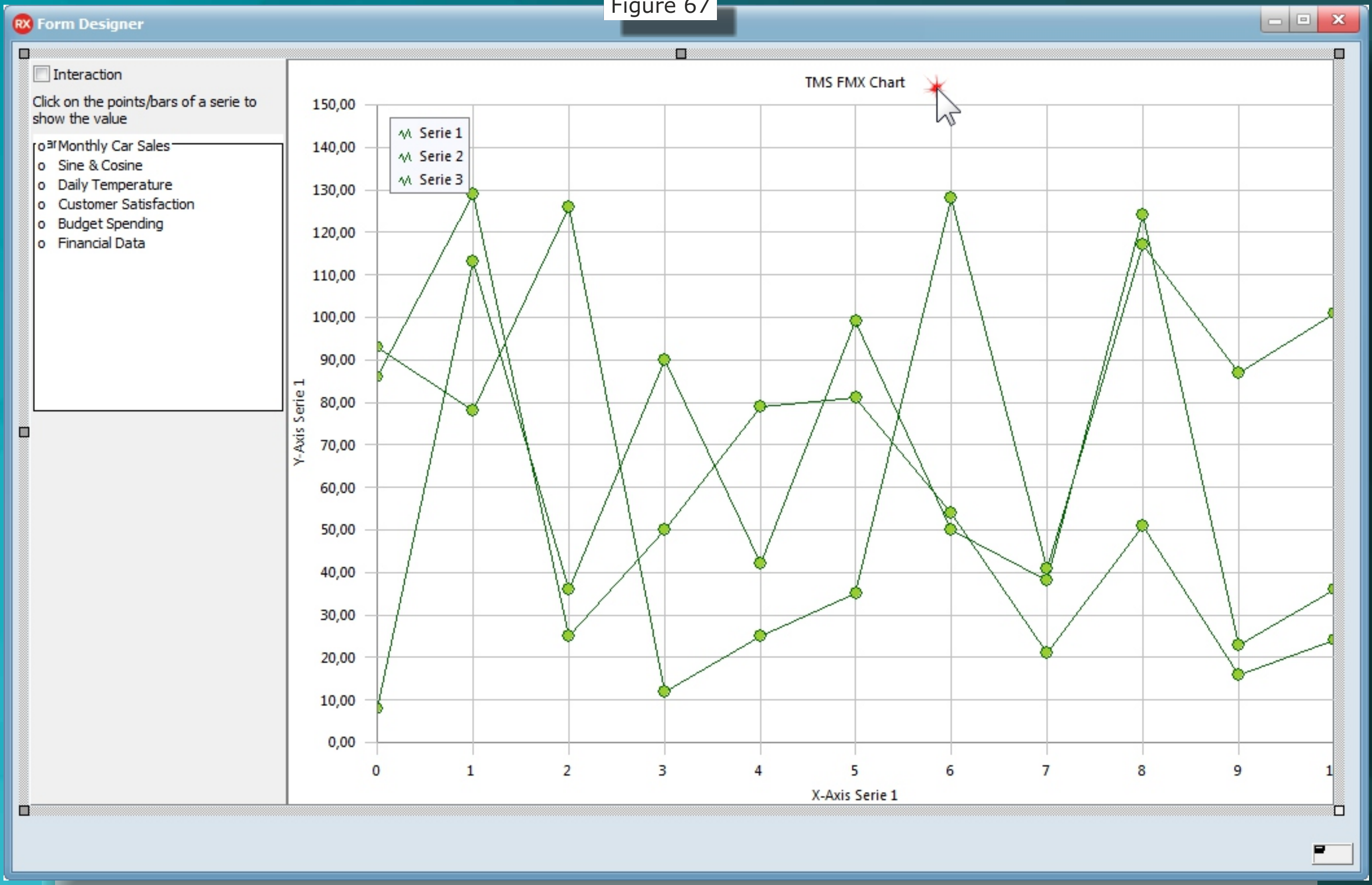
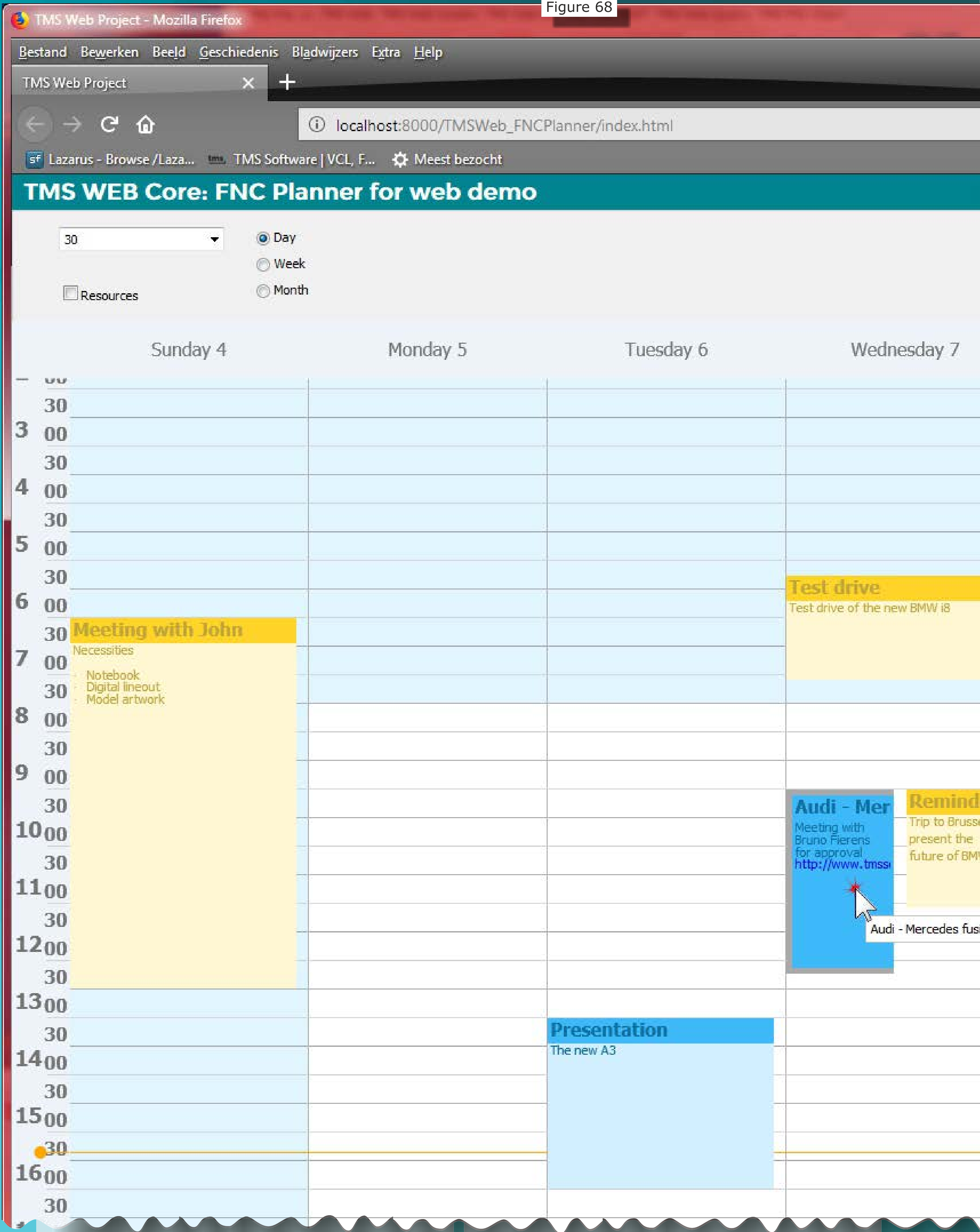


Figure 67
TMS Web offers extensive charting capabilities! There are almost no limits to the customization possibilities for charts! The complete charting abilities that you are used to from the desktop are now also available for the web. All the diagram types with all the customization options are available. Including user interaction and live charting.

Code Listing 21

Figure 68


TMS Web Project - Mozilla Firefox

localhost:8000/TMSWeb_FNCPlanner/index.html

TMS WEB Core: FNC Planner for web demo

30 · (●) Day · ( ) Week · ( ) Month · [ ] Resources

Sunday 4 · Monday 5 · Tuesday 6 · Wednesday 7

Meeting with John
Necessities
- Notebook
- Digital lineout
- Model artwork

Test drive
Test drive of the new BMW i8

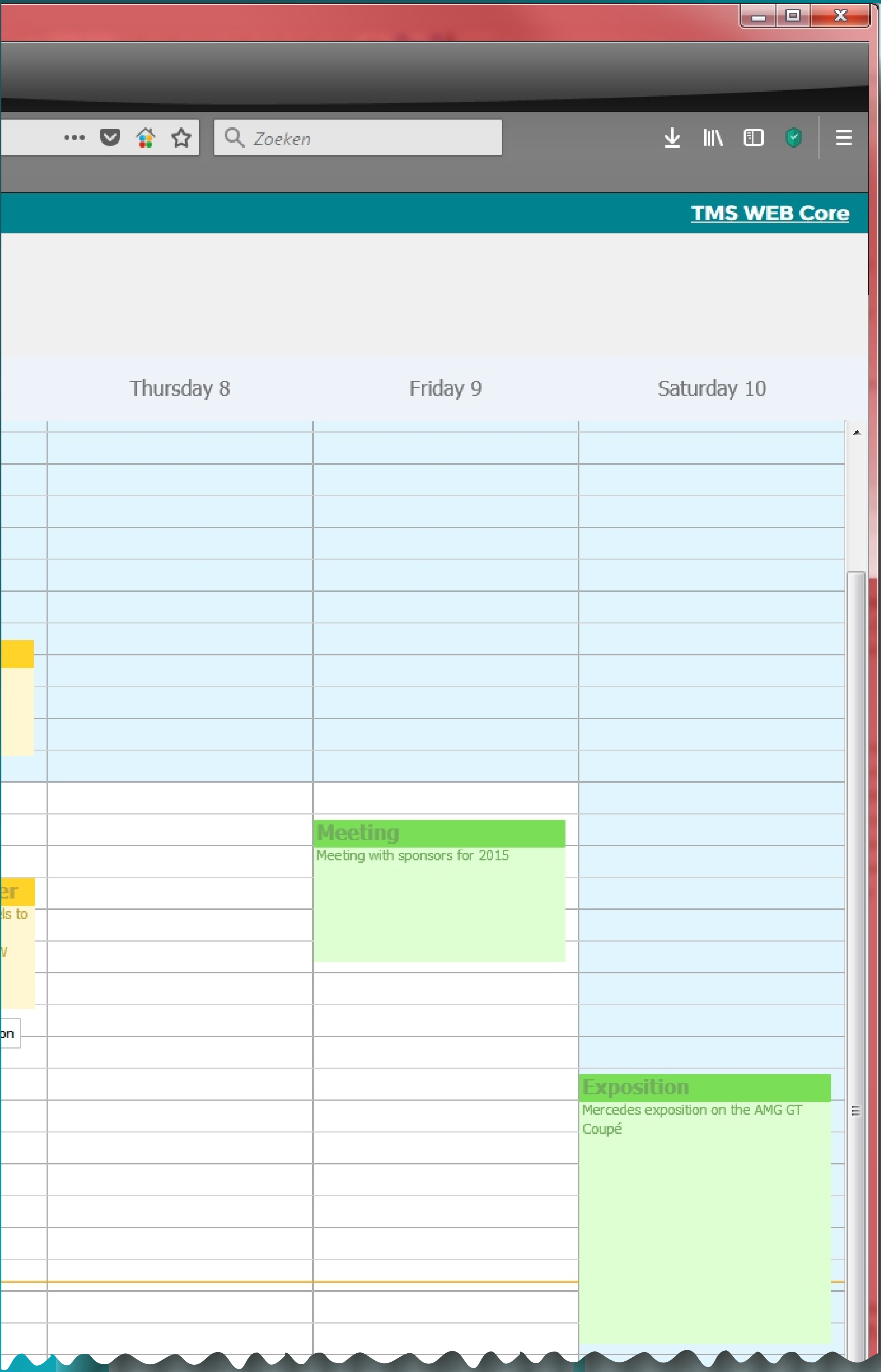
Audi - Mer
Meeting with Bruno Fierens for approval
http://www.tmss

Remind
Trip to Bruss present the future of BM

Audi - Mercedes fus

Presentation
The new A3

`TMSWeb_FNCPlanner.dproj`



TMSWeb_FNCPlanner.dproj

Figure 69

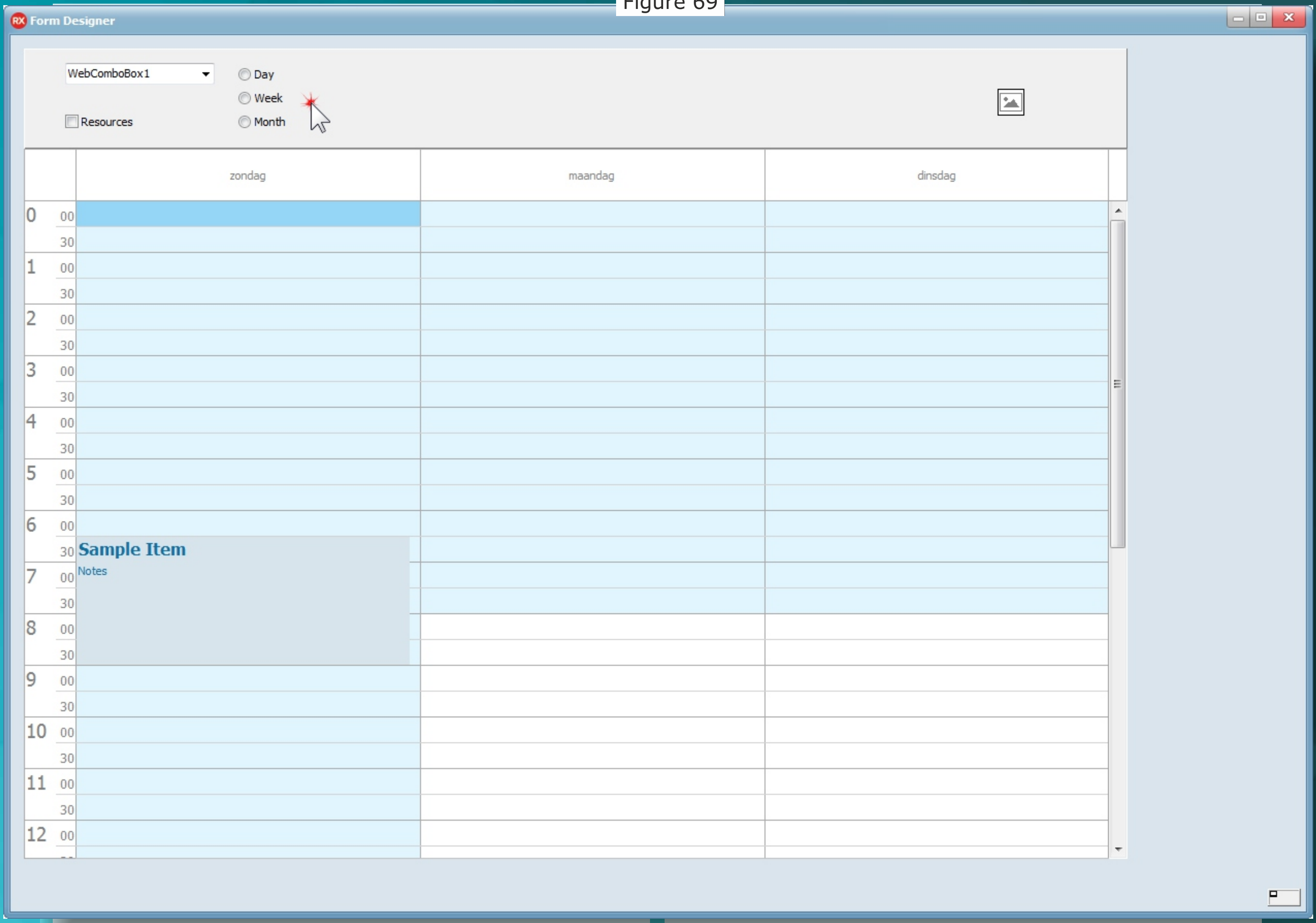


Figure 69
The planner component is very flexible and can be used for many purposes. In this demo, a REST service that provides local TV listings is consumed and the result is being transferred into the planner. This allows comfortable navigation of the TV programs and gives the user a truly incredible user experience to navigate the information. You can concentrate on reading the data from the web service and adding items to the planner. The whole visual part and the user interaction is handled by the frameworks. The best part about this: The very same source code can be shared for desktop and web!

FastReport VCL 6 is officially released!

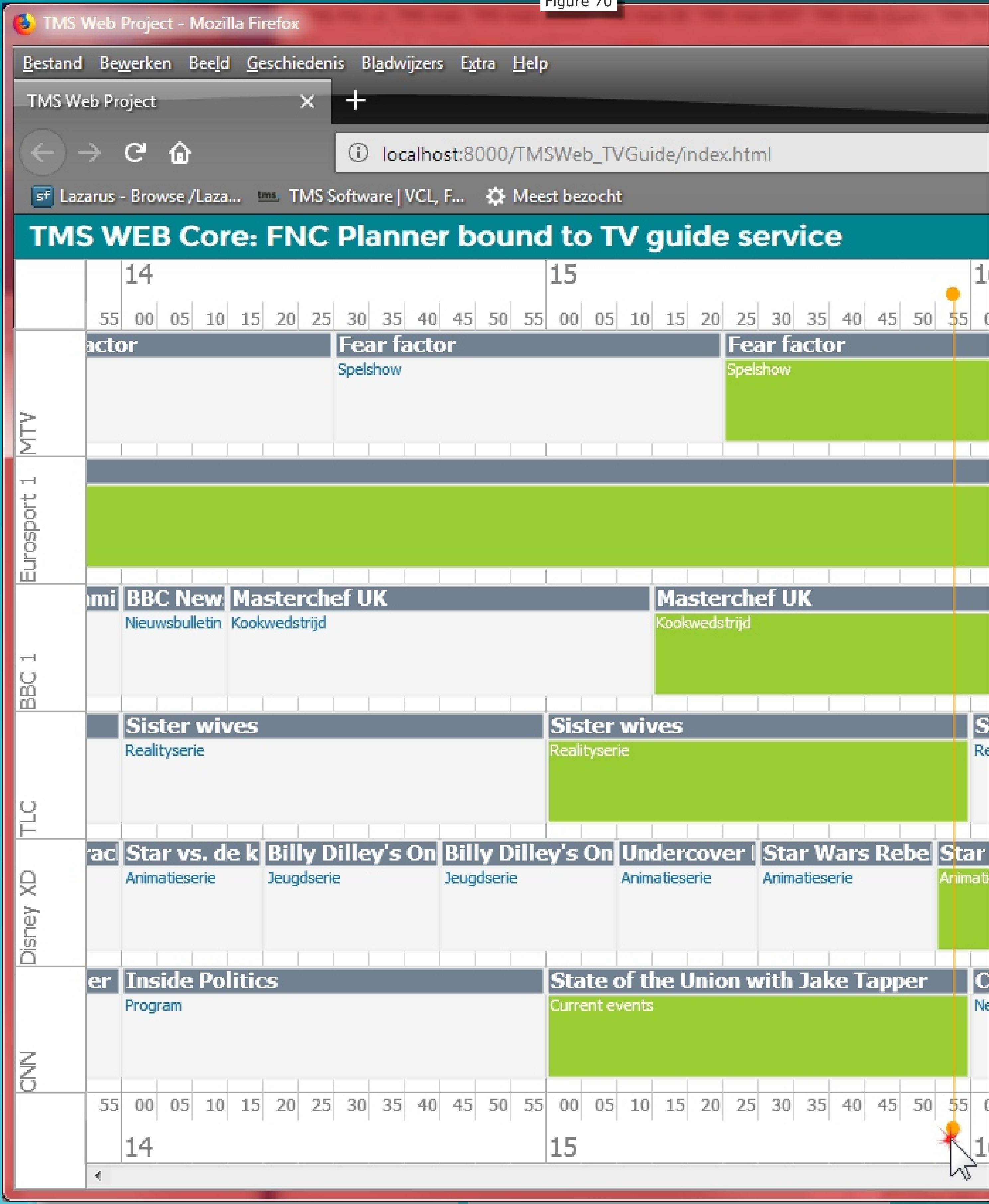
What's new in FastReport VCL 6?

Improved report engine expands editing and interactivity abilities. Report objects can be selected and edited instantly even from the preview
Expressions post processing and new duplicates processing.
Transport input-output filters: now you can save your reports to various cloud storages: DropBox, OneDrive, Box.com, Google Drive or send it by email
New report objects:
Table object – for super easy creating and editing of tabular reports
Map object that supports OSM, ESRI and GPX
Gauge object
New barcodes: Aztec, MaxiCode and linear USPS
Improved export filters to PDF, SVG, HTML5 will let you process complicated objects like RichText, Diagrams, Maps and exports them directly as vector/text format
And of course report designer couldn't be left without upgrade:
Improved Guide lines allow to move and resize docked objects.
Extended script debugger
Improved code completion
Copying and pasting of not only report objects, but their content as well
Enabling and disabling the quick editors

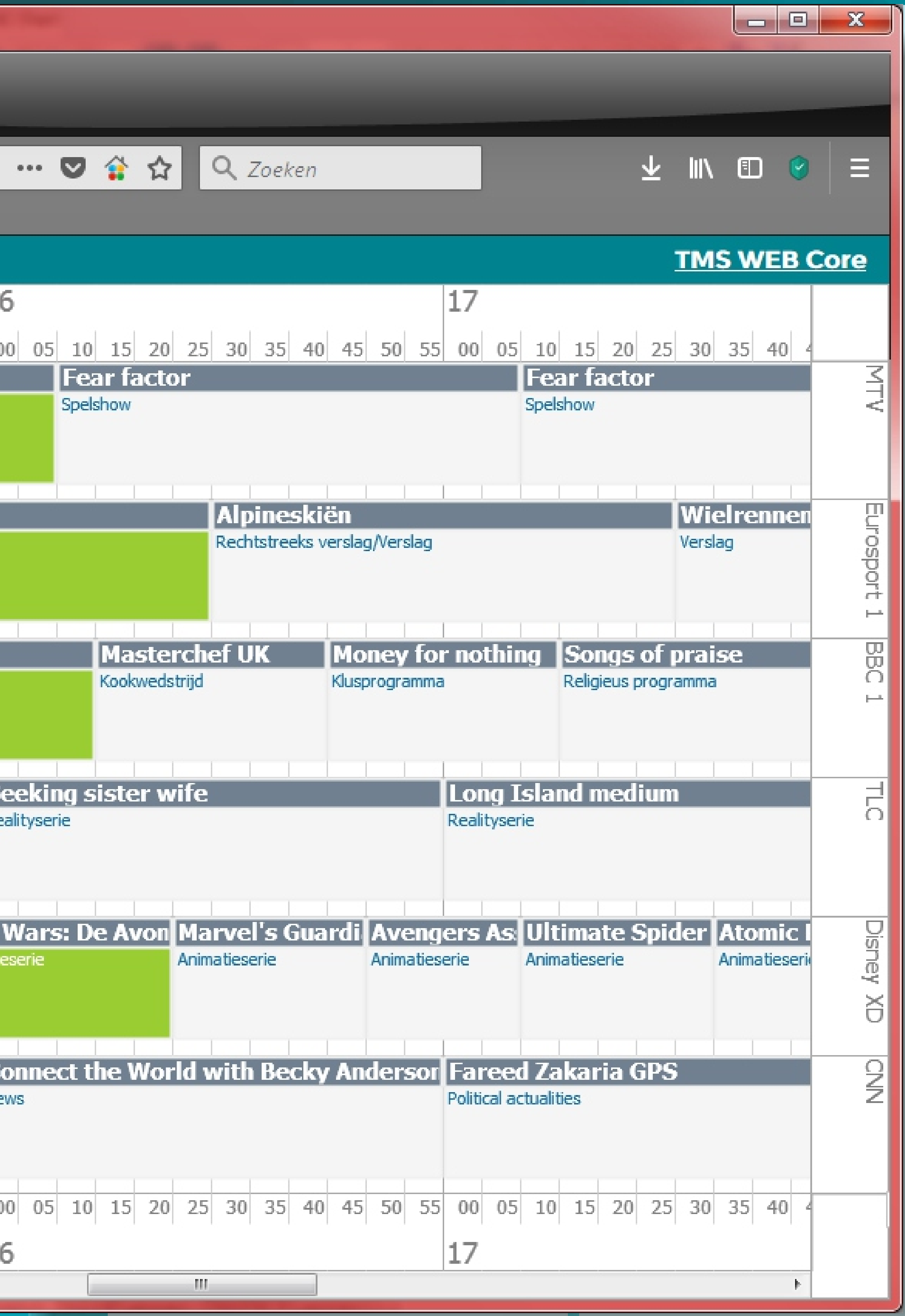
Fast Reports
Reporting must be Fast!

TVGuide\TMSWeb_TVGuide.dproj

Figure 70


TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS Web Project
localhost:8000/TMSWeb_TVGuide/index.html
Lazarus - Browse /Laza...
TMS Software | VCL, F...
Meest bezocht

TMS WEB Core: FNC Planner bound to TV guide service



MTV
Fear factor
Spelshow
Fear factor
Spelshow

Eurosport 1
Alpineskiën
Rechtstreeks verslag/Verslag
Wielrennen
Verslag

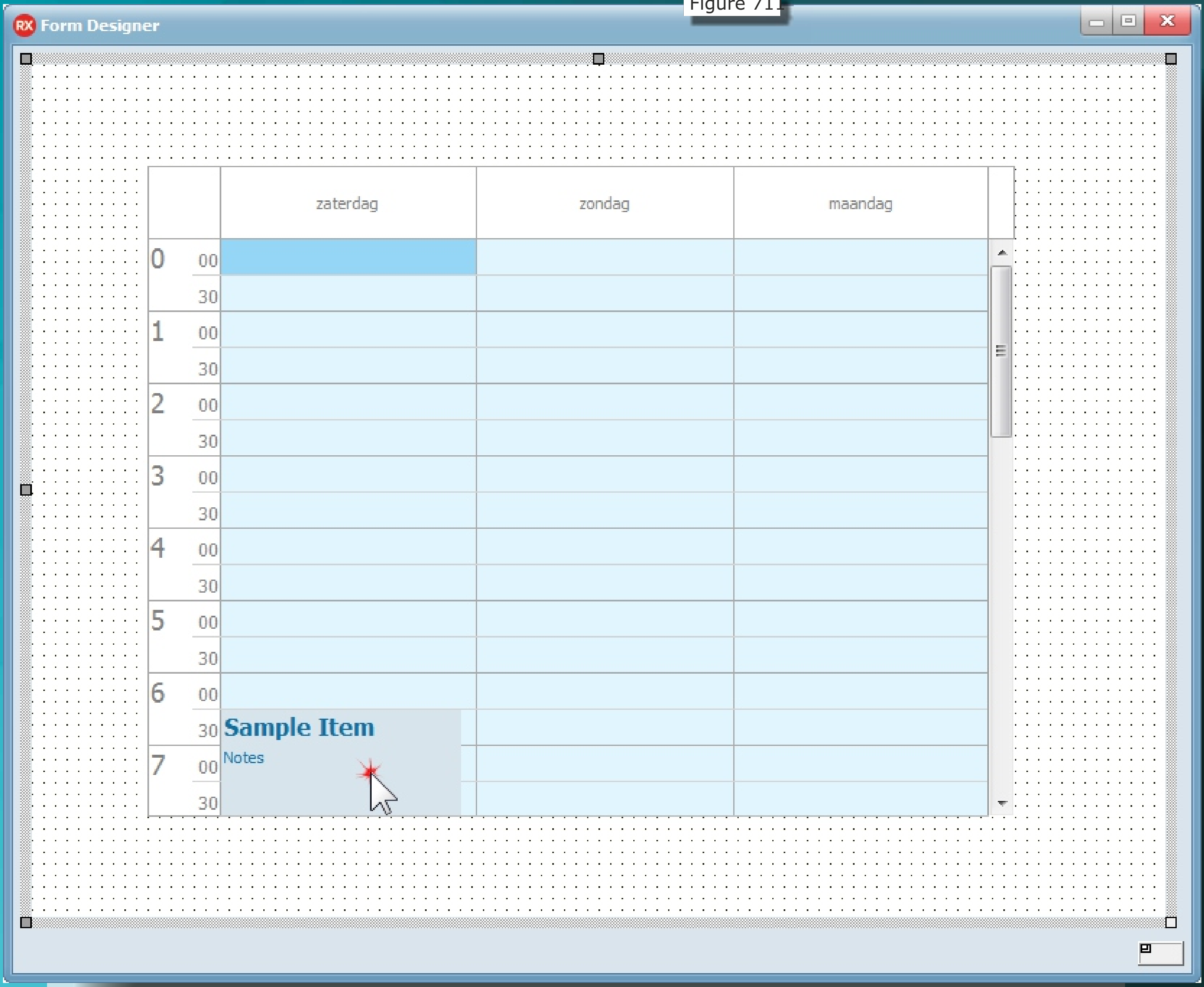
BBC 1
Masterchef UK
Kookwedstrijd
Money for nothing
Klusprogramma
Songs of praise
Religieus programma

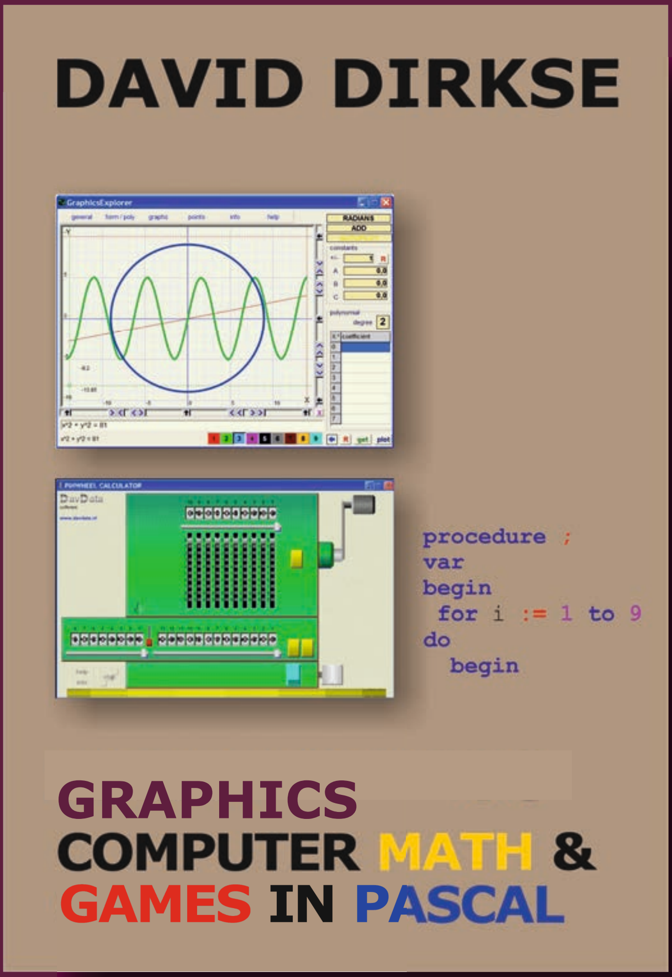
TLC
eeking sister wife
ealityserie
Long Island medium
Realityserie

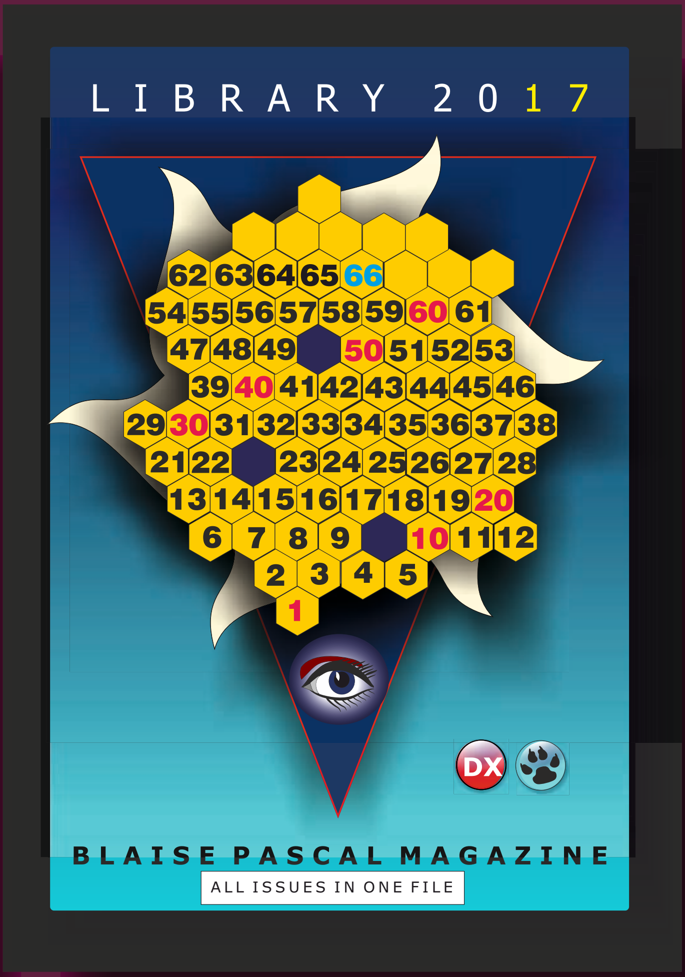
Disney XD
Wars: De Avon
eserie
Marvel's Guardi
Animatieserie
Avengers As
Animatieserie
Ultimate Spider
Animatieserie
Atomic
Animatieseri

CNN
onnect the World with Becky Anderson
ews
Fareed Zakaria GPS
Political actualities

Figure 71

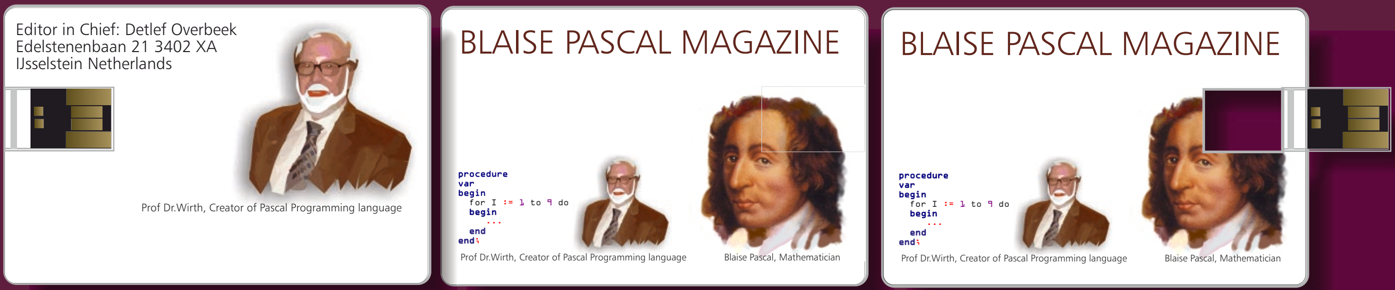

Form Designer
zaterdag
zondag
maandag
0
00
30
1
00
30
2
00
30
3
00
30
4
00
30
5
00
30
6
00
30
7
00
30
Sample Item
Notes





POCKET EDITION
Printed in full color.
A fully indexed PDF book
is included + 52 projects

CREDITCARD LIBRARY STICK 16 GB
All issues on the USB stick
complete searchable 3600 pages -
fully indexed


Editor in Chief: Detlef Overbeek
Edelstenenbaan 21 3402 XA
IJsselstein Netherlands
Prof Dr.Wirth, Creator of Pascal Programming language
BLAISE PASCAL MAGAZINE
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician
BLAISE PASCAL MAGAZINE
Prof Dr.Wirth, Creator of Pascal Programming language
Blaise Pascal, Mathematician

COMBINATION: 3 FOR 1

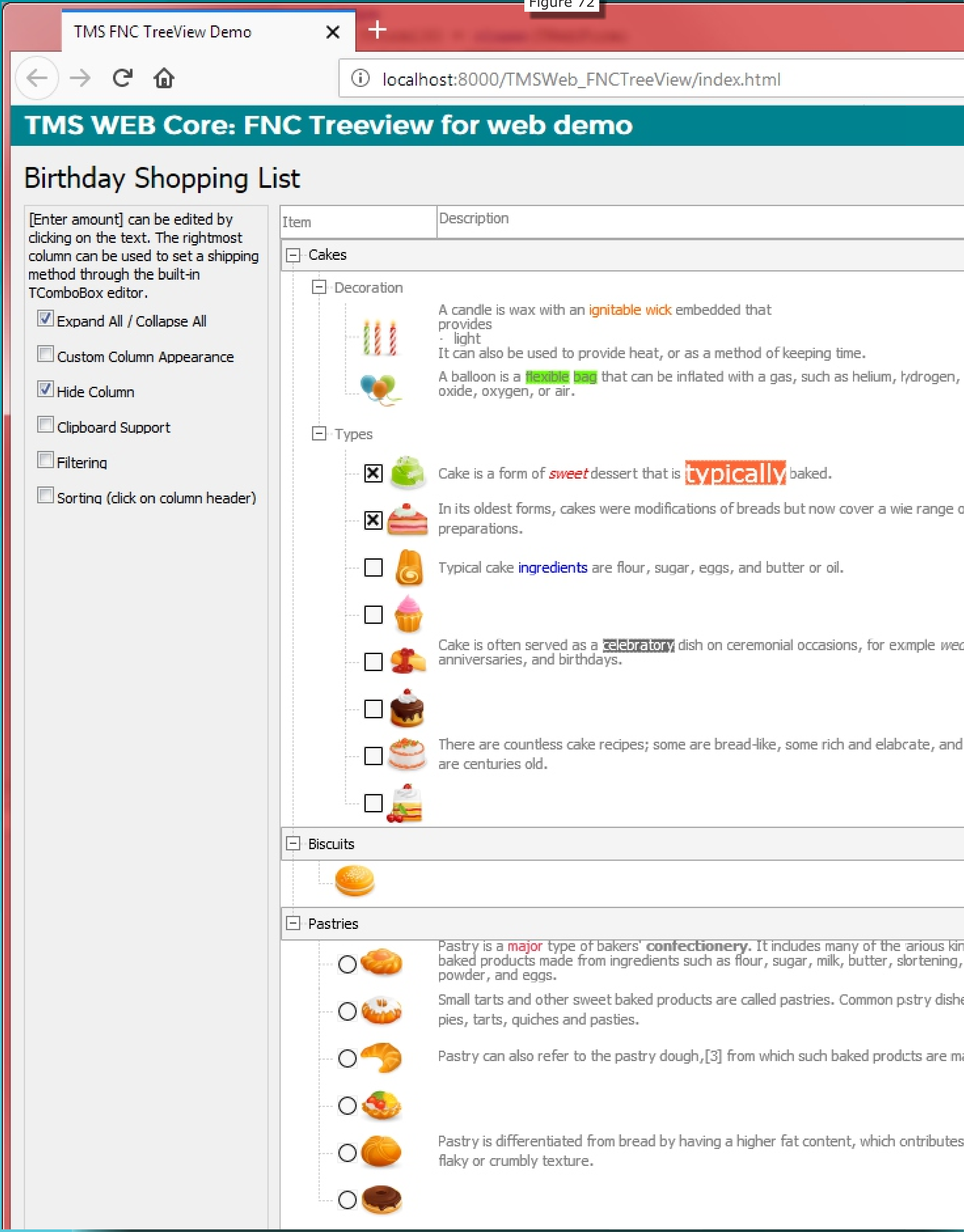
BOOK INCLUDING THE LIBRARY STICK EXCL. SHIPPING
INCLUDING 1YEAR DOWNLOAD FOR FREE
GET THE BOOK INCLUDING THE NEWEST LIBRARY STICK
INCLUDING 1 YEAR DOWNLOAD OF BLAISE PASCAL MAGAZINE

€ 100

https://www.blaisepascal.eu/subscribers/UK/UK_CD_DVD_USB_Department.html

TMSWeb_FNCTreeView.dproj

Figure 72



TMS FNC TreeView Demo

localhost:8000/TMSWeb_FNCTreeView/index.html

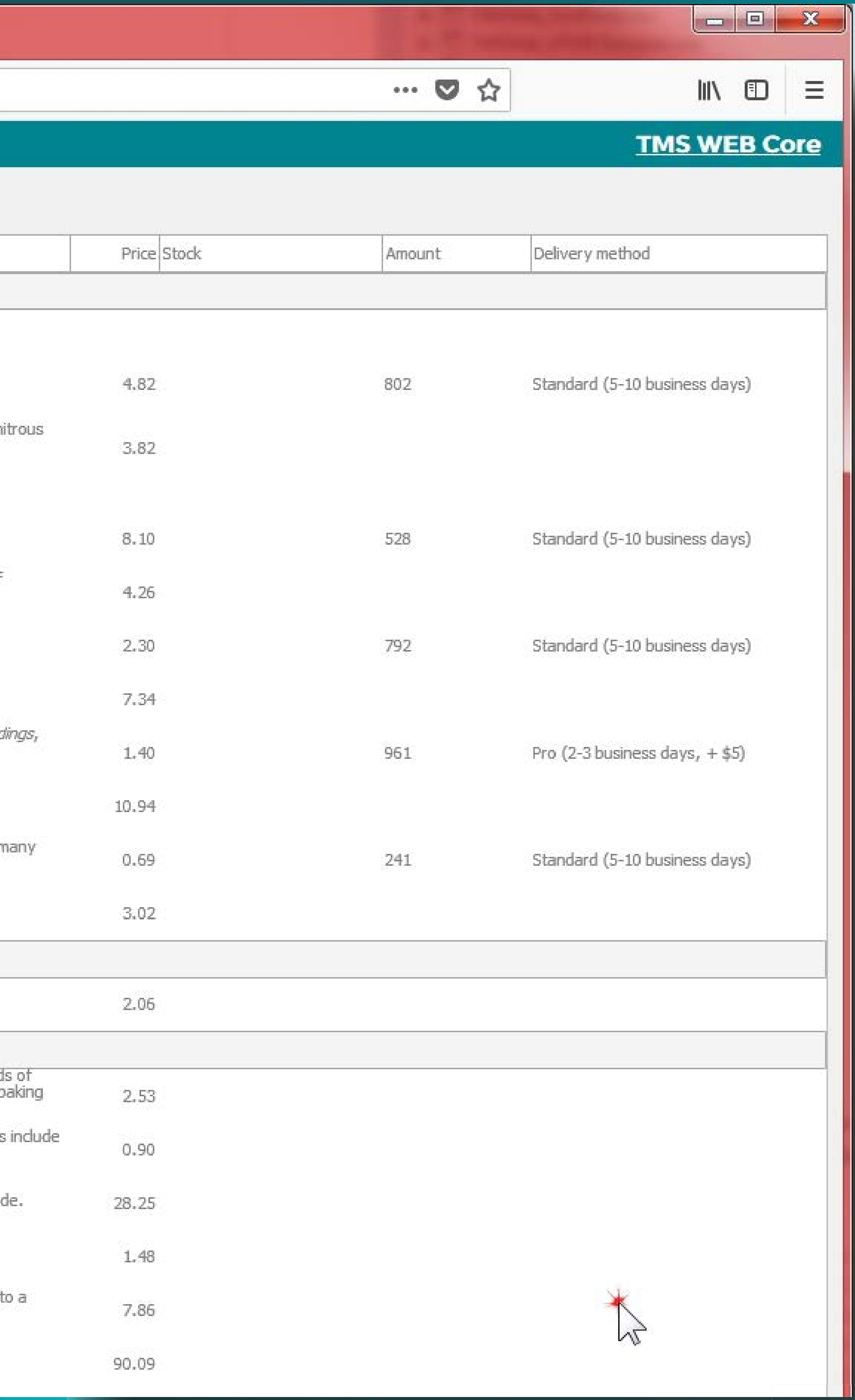
TMS WEB Core: FNC Treeview for web demo

Birthday Shopping List

[Enter amount] can be edited by clicking on the text. The rightmost column can be used to set a shipping method through the built-in TComboBox editor.

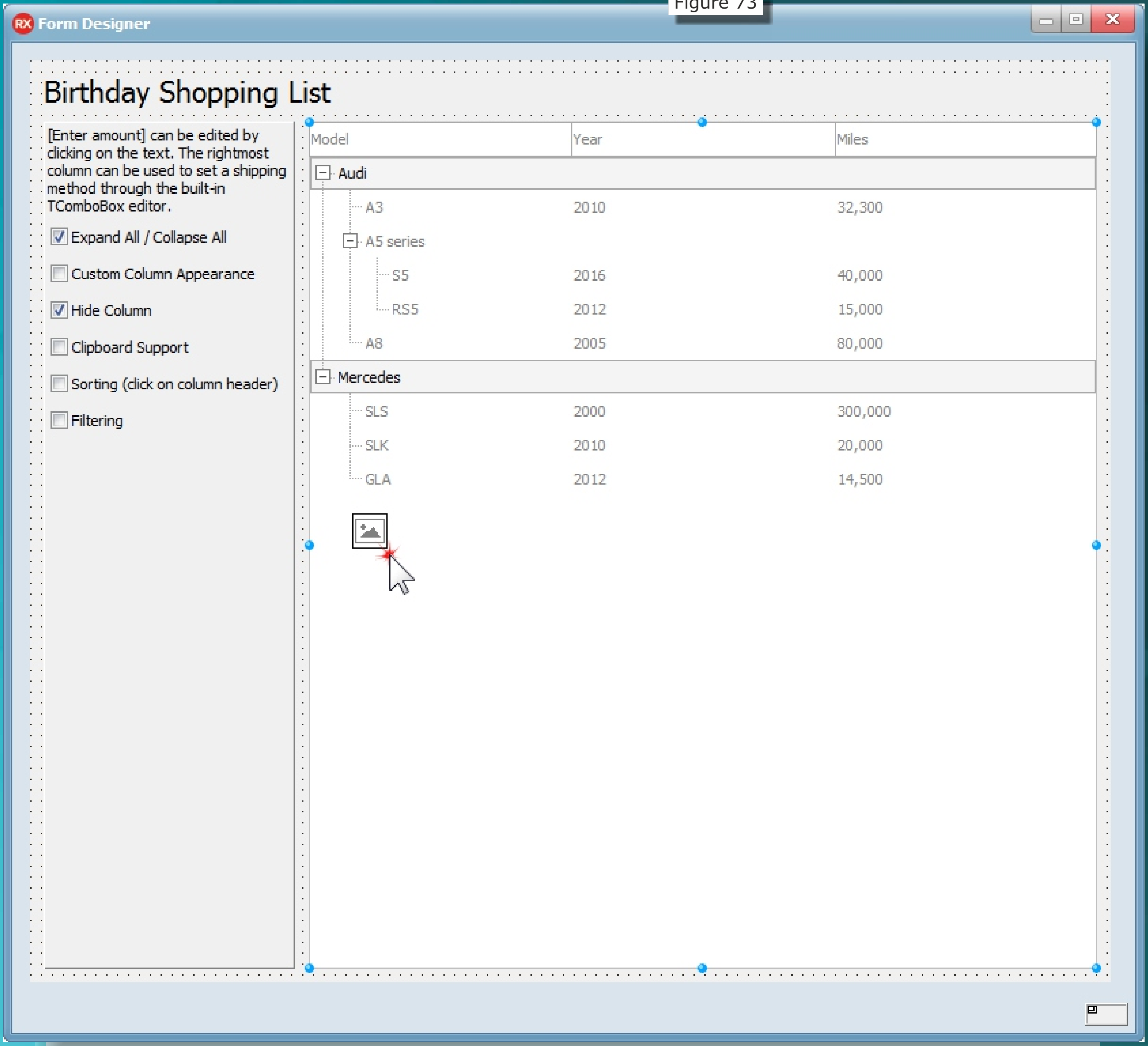
- [✓] Expand All / Collapse All
- [ ] Custom Column Appearance
- [✓] Hide Column
- [ ] Clipboard Support
- [ ] Filtering
- [ ] Sorting (click on column header)

Item	Description
Cakes	
Decoration	
	A candle is wax with an ignitable wick embedded that provides · light It can also be used to provide heat, or as a method of keeping time.
	A balloon is a flexible bag that can be inflated with a gas, such as helium, hydrogen, oxide, oxygen, or air.
Types	
☒	Cake is a form of *sweet* dessert that is typically baked.
☒	In its oldest forms, cakes were modifications of breads but now cover a wie range o preparations.
☐	Typical cake ingredients are flour, sugar, eggs, and butter or oil.
☐	
☐	Cake is often served as a celebratory dish on ceremonial occasions, for exmple wed anniversaries, and birthdays.
☐	
☐	There are countless cake recipes; some are bread-like, some rich and elabcrate, and are centuries old.
☐	
Biscuits	
Pastries	
○	Pastry is a major type of bakers' **confectionery**. It includes many of the arious kin baked products made from ingredients such as flour, sugar, milk, butter, slortening, powder, and eggs.
○	Small tarts and other sweet baked products are called pastries. Common pstry dishe pies, tarts, quiches and pasties.
○	Pastry can also refer to the pastry dough,[3] from which such baked prodicts are ma
○	
○	Pastry is differentiated from bread by having a higher fat content, which contributes flaky or crumbly texture.
○	



	Price	Stock	Amount	Delivery method
	4.82		802	Standard (5-10 business days)
itrous	3.82			
	8.10		528	Standard (5-10 business days)
F	4.26			
	2.30		792	Standard (5-10 business days)
	7.34			
dings,	1.40		961	Pro (2-3 business days, + $5)
	10.94			
many	0.69		241	Standard (5-10 business days)
	3.02			
	2.06			
ds of baking	2.53			
s include	0.90			
de.	28.25			
	1.48			
to a	7.86			
	90.09			

Figure 73



Birthday Shopping List

[Enter amount] can be edited by clicking on the text. The rightmost column can be used to set a shipping method through the built-in TComboBox editor.

- [✓] Expand All / Collapse All
- [ ] Custom Column Appearance
- [✓] Hide Column
- [ ] Clipboard Support
- [ ] Sorting (click on column header)
- [ ] Filtering

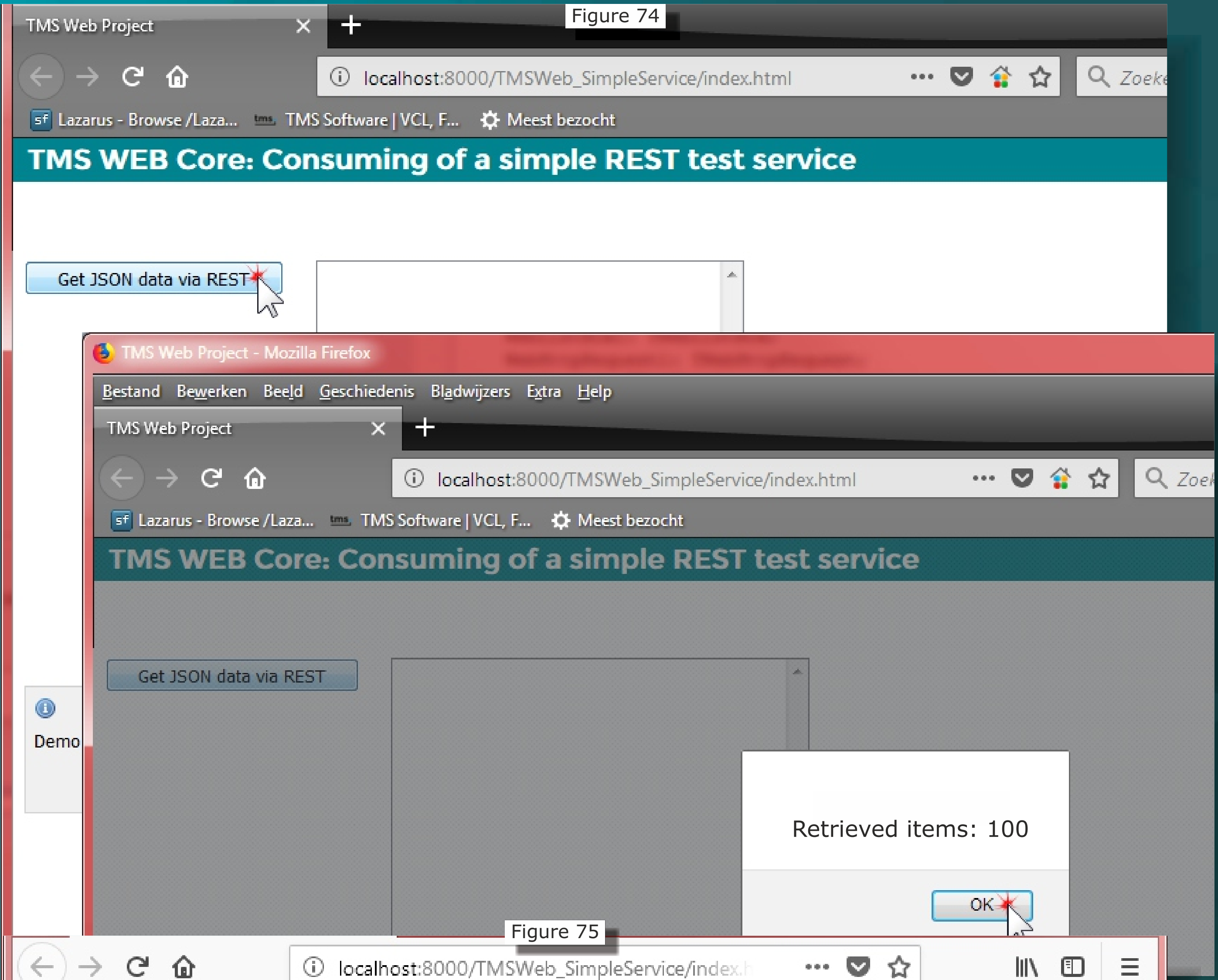
Model	Year	Miles
Audi		
A3	2010	32,300
A5 series		
S5	2016	40,000
RS5	2012	15,000
A8	2005	80,000
Mercedes		
SLS	2000	300,000
SLK	2010	20,000
GLA	2012	14,500

Figure 73
Another excellent example of the incredible possibilities to create an awesome user experience.
A treeview component offers many options to display data in a hierarchical fashion.
Design options include different styles and formats for any cell.
As with all data controls from TMS, there is support for filterting, grouping and sorting.
Data can be imported and exported to different file formats. TMS Web Applications can also easily interact with the clipboard, which is also demonstrated with this sample.

Code Listing 38

TMSWeb_SimpleService.dproj

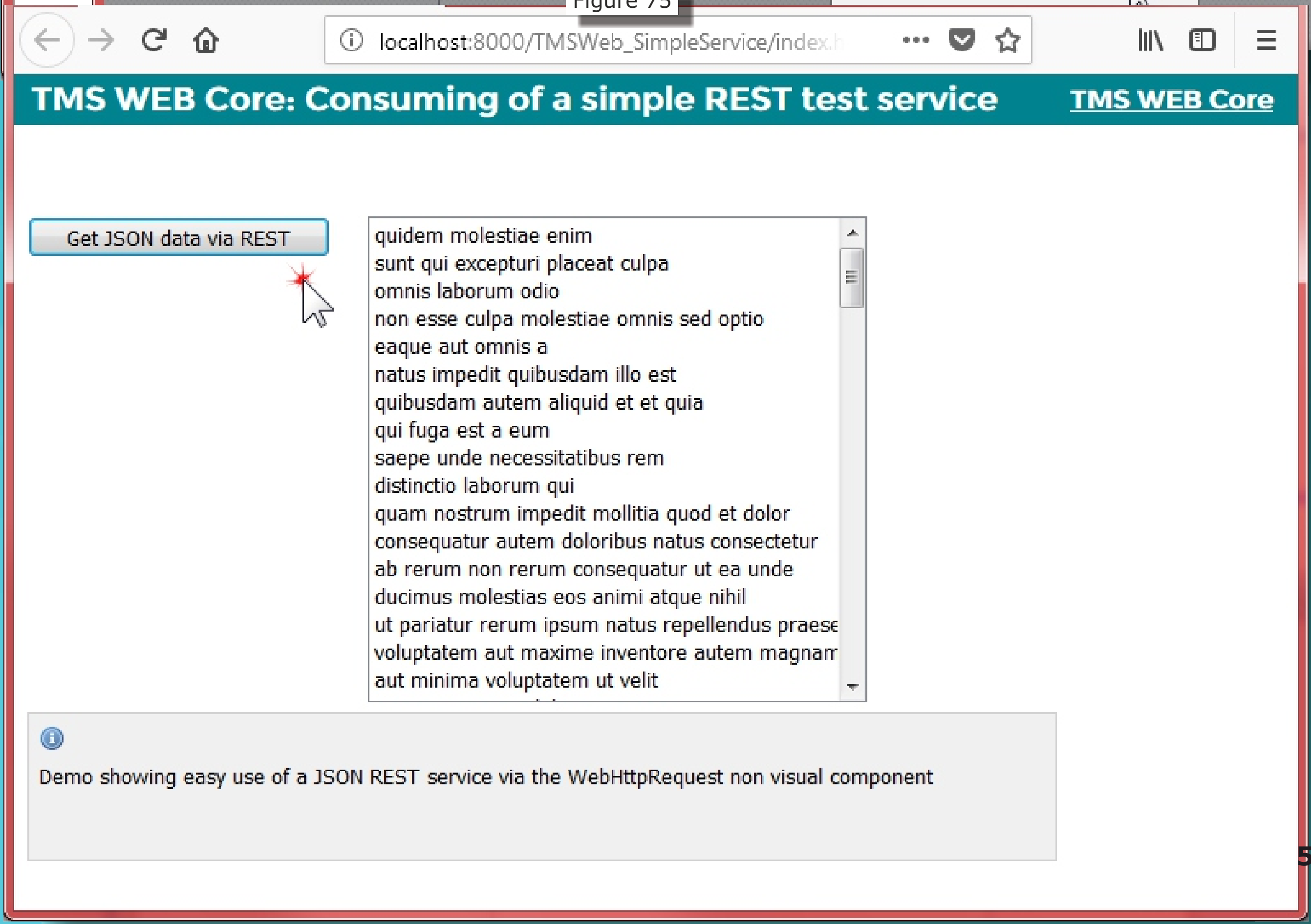
Figure 74



TMS WEB Core: Consuming of a simple REST test service

Retrieved items: 100

Figure 75

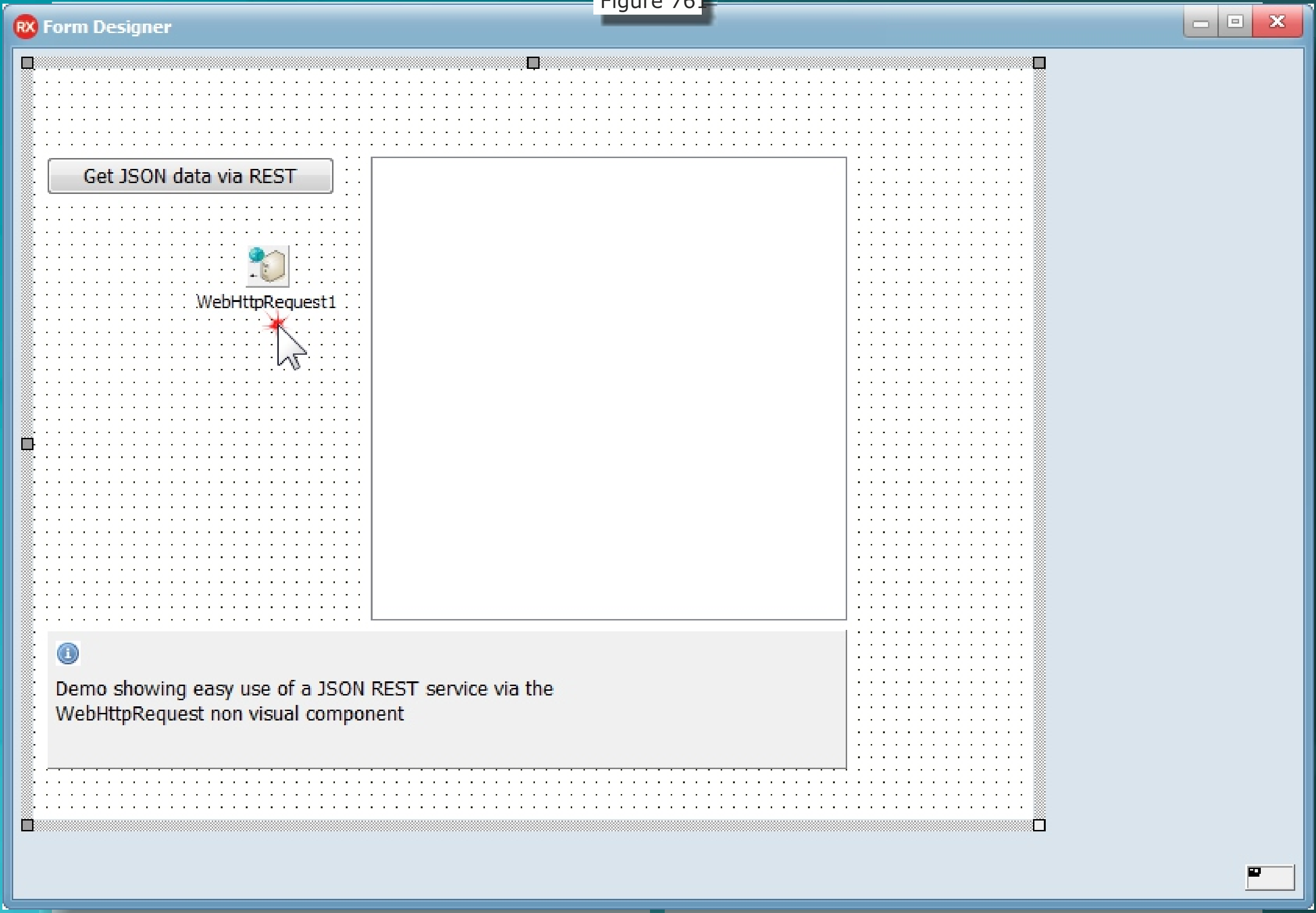


TMS WEB Core: Consuming of a simple REST test service

```
quidem molestiae enim
sunt qui excepturi placeat culpa
omnis laborum odio
non esse culpa molestiae omnis sed optio
eaque aut omnis a
natus impedit quibusdam illo est
quibusdam autem aliquid et et quia
qui fuga est a eum
saepe unde necessitatibus rem
distinctio laborum qui
quam nostrum impedit mollitia quod et dolor
consequatur autem doloribus natus consectetur
ab rerum non rerum consequatur ut ea unde
ducimus molestias eos animi atque nihil
ut pariatur rerum ipsum natus repellendus praese
voluptatem aut maxime inventore autem magnam
aut minima voluptatem ut velit
```

Demo showing easy use of a JSON REST service via the WebHttpRequest non visual component

Figure 761



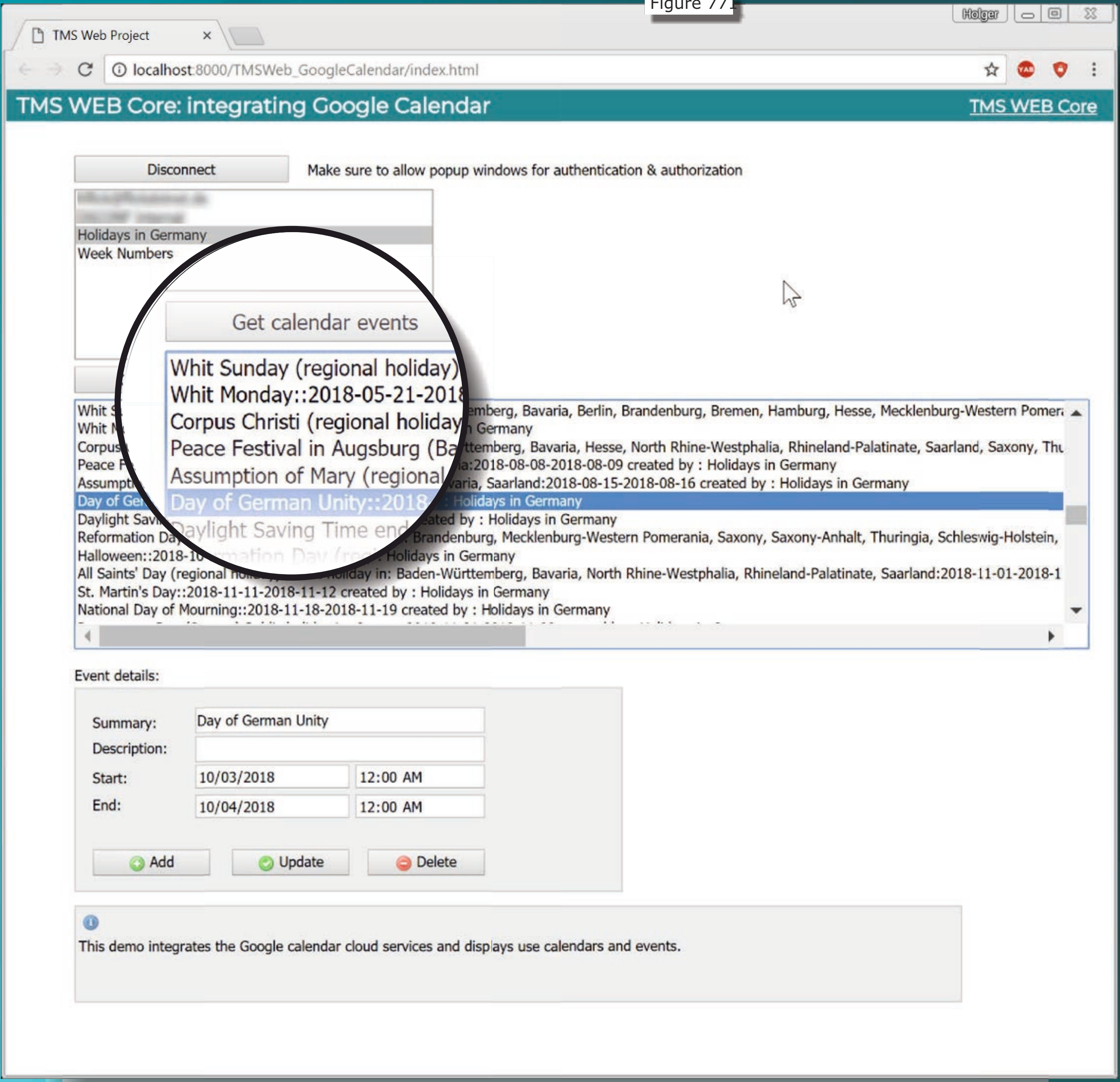
Demo showing easy use of a JSON REST service via the
WebHttpRequest non visual component

```
procedure TForm1.WebButton1Click(Sender: TObject);
begin
 WebHttpRequest1.URL :=
'https://jsonplaceholder.typicode.com/albums';
 WebHttpRequest1.Execute;
end;

procedure TForm1.WebHttpRequest1Response(Sender: TObject; AResponse: string);
var
 js: TJSON;
 ja: TJSONArray;
 jo: TJSONObject;
 i: integer;
begin
 js := TJSON.Create;

 try
  ja := TJSONArray(js.Parse(AResponse));

  ShowMessage('Retrieved items:' +inttostr(ja.Count));

  for i := 0 to ja.Count - 1 do
  begin
   jo := ja.Items[i];
   WebListBox1.Items.Add(jo.Get('title'));
  end;
 finally
  js.Free;
 end;
end;
```

Figure 76
Consuming web services is easy as a HTTP request component is available. Data can be processed using the Delphi RTL JSON classes.

Code Listing 14

Figure 771


TMS Web Project
localhost:8000/TMSWeb_GoogleCalendar/index.html
Holger
TMS WEB Core: integrating Google Calendar
TMS WEB Core
Disconnect
Get calendar events
Add
Update
Delete

Make sure to allow popup windows for authentication & authorization

[redacted]
Holidays in Germany
Week Numbers

Whit Sunday (regional holiday)
Whit Monday::2018-05-21-2018
Corpus Christi (regional holiday
Peace Festival in Augsburg (Ba
Assumption of Mary (regional
Day of German Unity::2018
Daylight Saving Time end
mation Day (regi

```
Whit S                                      emberg, Bavaria, Berlin, Brandenburg, Bremen, Hamburg, Hesse, Mecklenburg-Western Pomera
Whit M                                      n Germany
Corpus                                      ttemberg, Bavaria, Hesse, North Rhine-Westphalia, Rhineland-Palatinate, Saarland, Saxony, Thu
Peace F                                     ia:2018-08-08-2018-08-09 created by : Holidays in Germany
Assumpti                                    varia, Saarland:2018-08-15-2018-08-16 created by : Holidays in Germany
Day of Ger                                  : Holidays in Germany
Daylight Savi                               eated by : Holidays in Germany
Reformation Da                              : Brandenburg, Mecklenburg-Western Pomerania, Saxony, Saxony-Anhalt, Thuringia, Schleswig-Holstein,
Halloween::2018-10                          : Holidays in Germany
All Saints' Day (regional                   oliday in: Baden-Württemberg, Bavaria, North Rhine-Westphalia, Rhineland-Palatinate, Saarland:2018-11-01-2018-1
St. Martin's Day::2018-11-11-2018-11-12 created by : Holidays in Germany
National Day of Mourning::2018-11-18-2018-11-19 created by : Holidays in Germany
```

Event details:

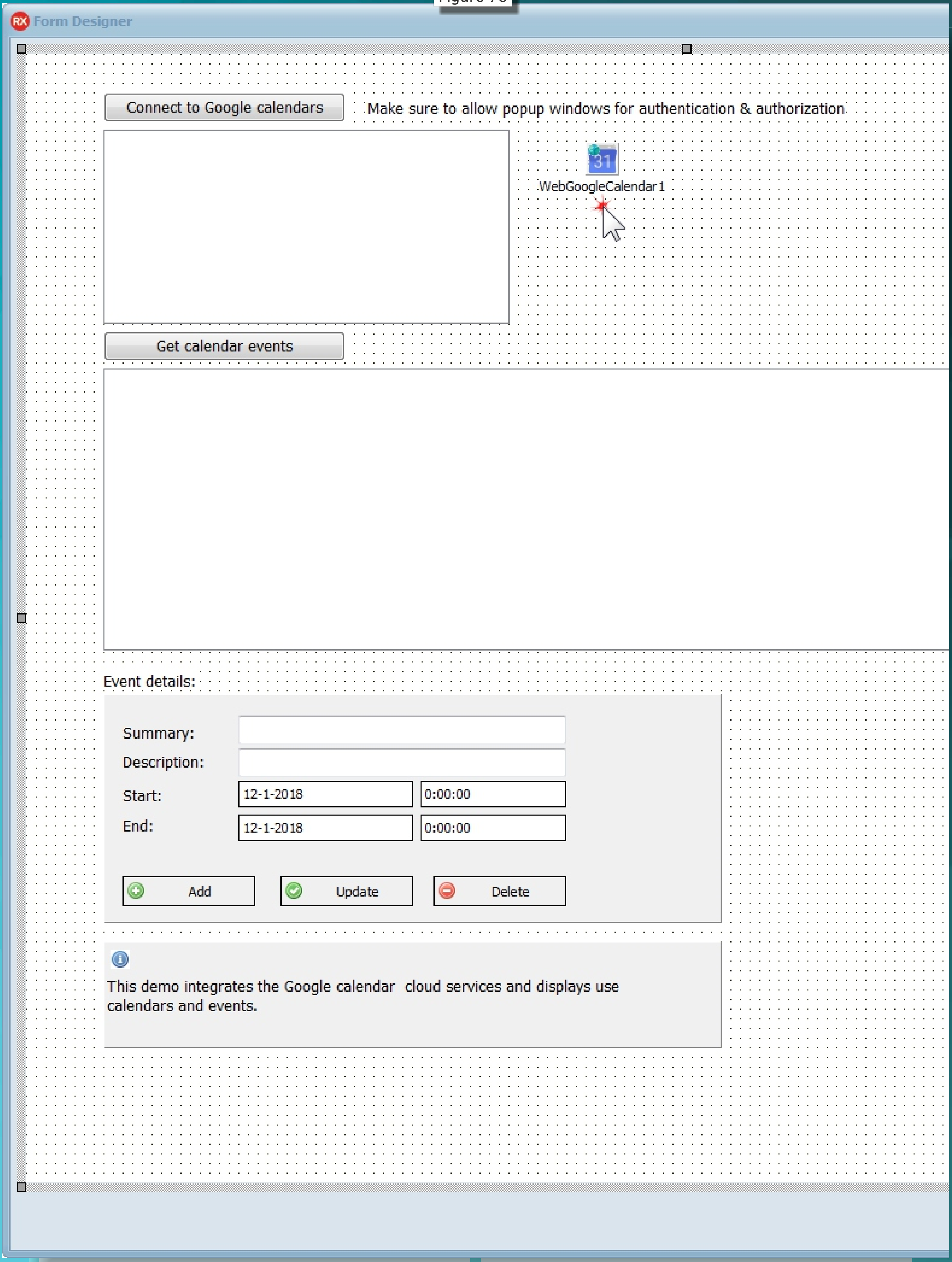
Summary:	Day of German Unity	
Description:		
Start:	10/03/2018	12:00 AM
End:	10/04/2018	12:00 AM

This demo integrates the Google calendar cloud services and displays use calendars and events.

Figure 77:
TMS Web can interact with many Cloud Services and the number of supported Cloud Services is increasing frequently. This demo shows how to access a Google Calendar without the hassle of thinking about anything. You simply drop the calendar component and provide you user credentials and the component offers properties and methods to work with any calendar stored in your profile.

TMSWeb_google_calendar.dproj

Figure 78



Make sure to allow popup windows for authentication & authorization

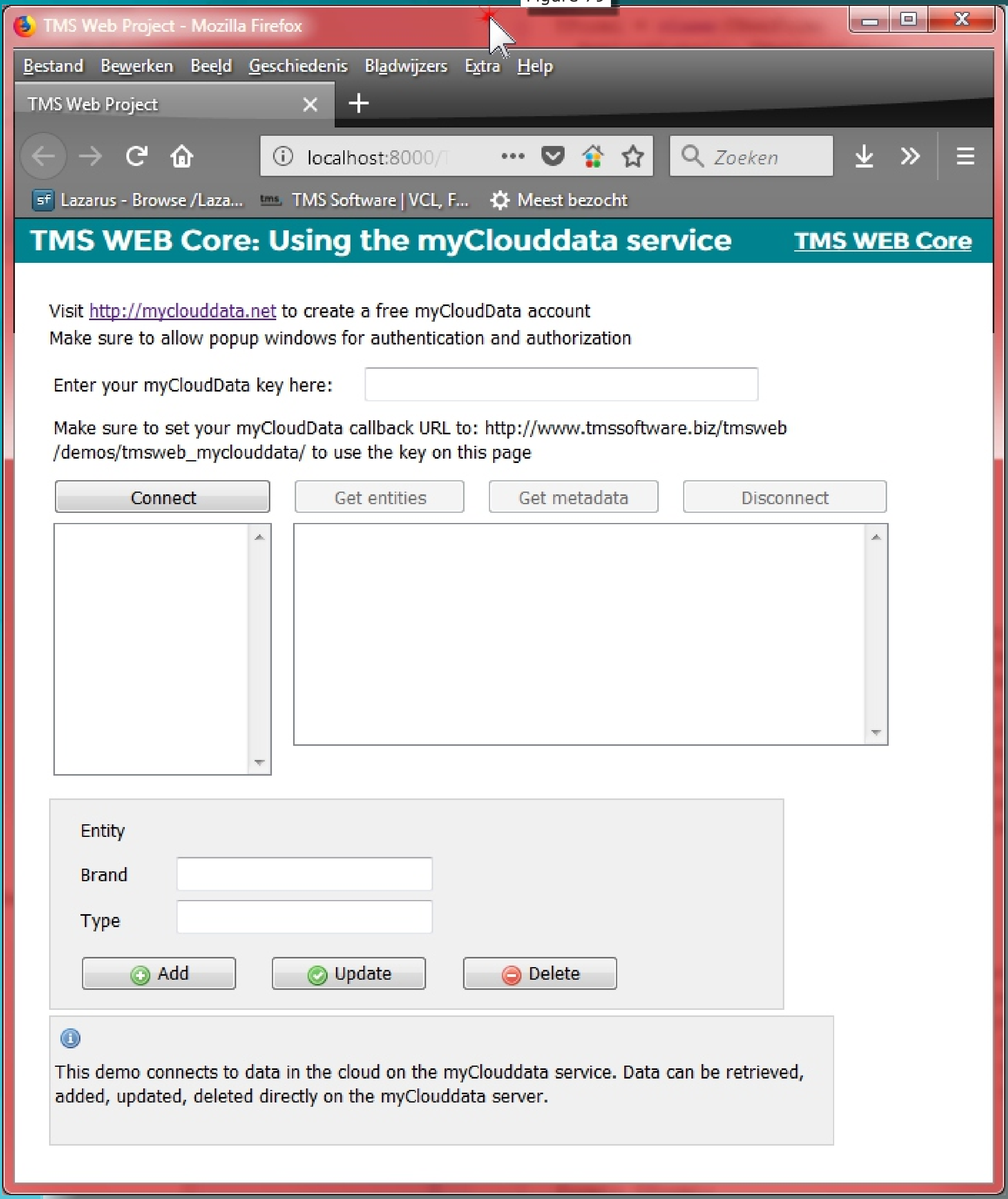
Event details:

Summary:
Description:
Start: 12-1-2018 0:00:00
End: 12-1-2018 0:00:00

This demo integrates the Google calendar cloud services and displays use calendars and events.

TMSWeb_myClouddata.dproj

Figure 79



TMS WEB Core: Using the myClouddata service

Visit http://myclouddata.net to create a free myCloudData account
Make sure to allow popup windows for authentication and authorization

Enter your myCloudData key here:

Make sure to set your myCloudData callback URL to: http://www.tmssoftware.biz/tmsweb /demos/tmsweb_myclouddata/ to use the key on this page

Entity

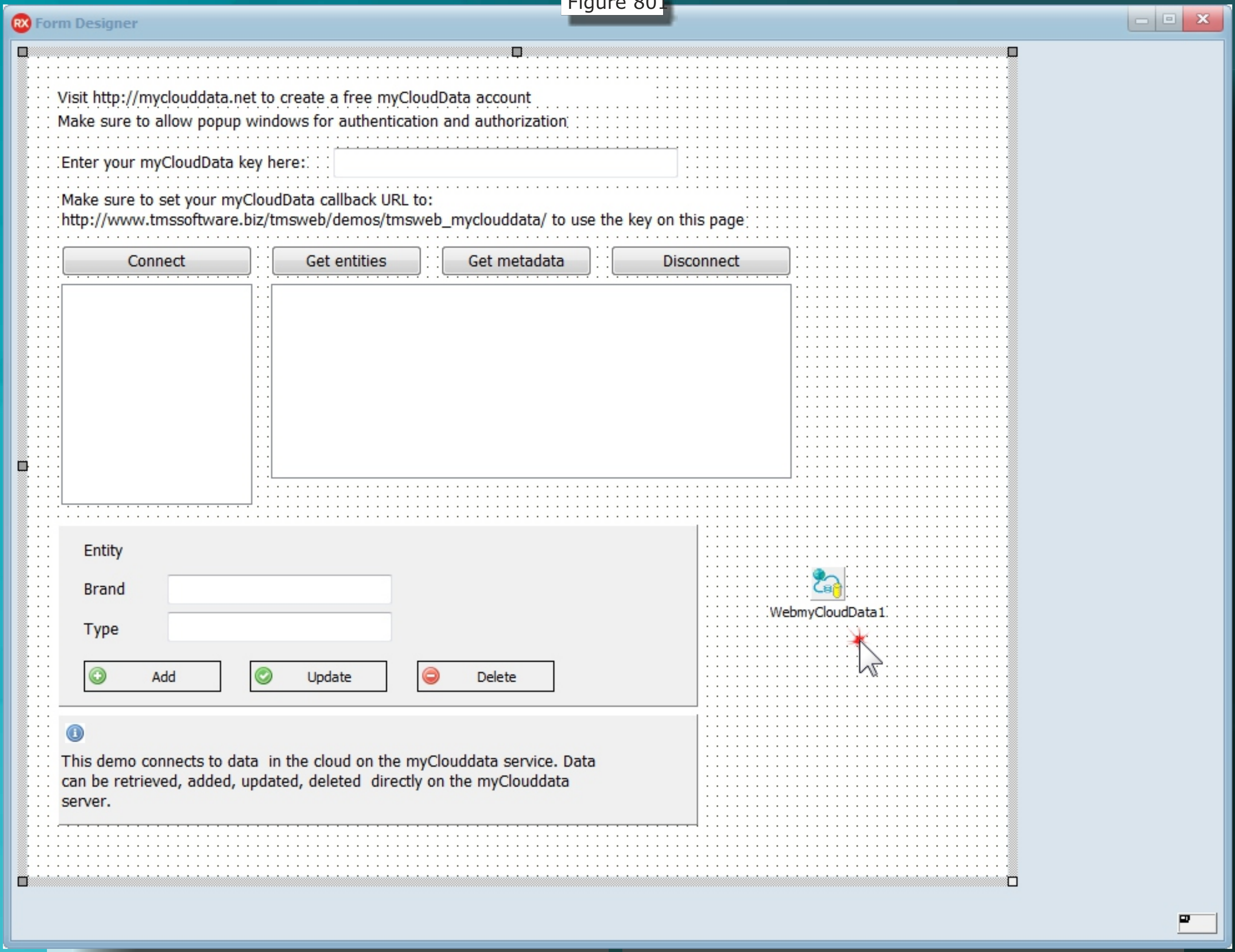
Brand

Type

This demo connects to data in the cloud on the myClouddata service. Data can be retrieved, added, updated, deleted directly on the myClouddata server.

Figure 91: Support for myCloudData is also already included in the first version.

Figure 80:



Visit http://myclouddata.net to create a free myCloudData account
Make sure to allow popup windows for authentication and authorization

Enter your myCloudData key here:

Make sure to set your myCloudData callback URL to:
http://www.tmssoftware.biz/tmsweb/demos/tmsweb_myclouddata/ to use the key on this page

Entity
Brand
Type

This demo connects to data in the cloud on the myClouddata service. Data can be retrieved, added, updated, deleted directly on the myClouddata server.

TMSWeb_Embedding.dproj

Figure 81

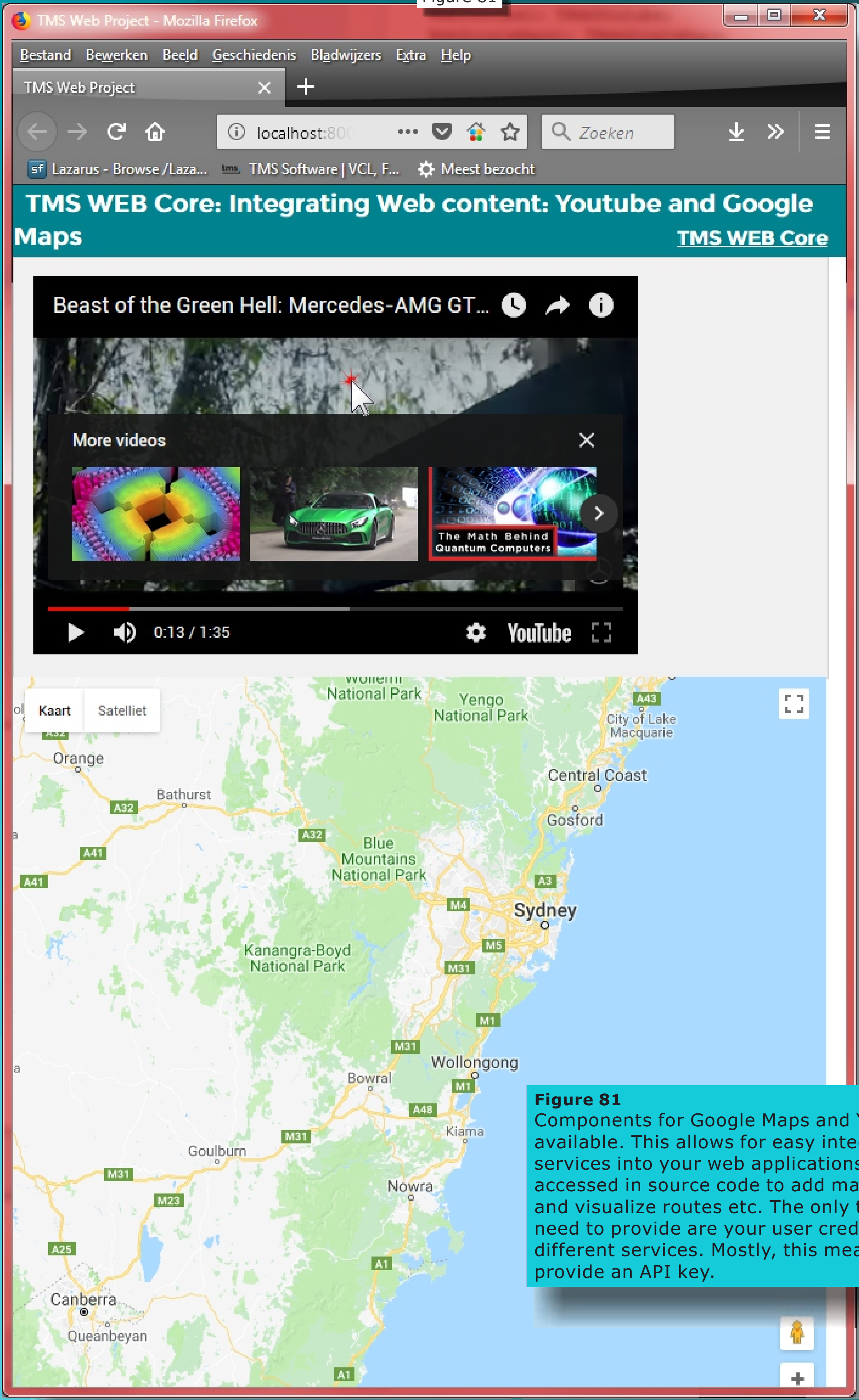
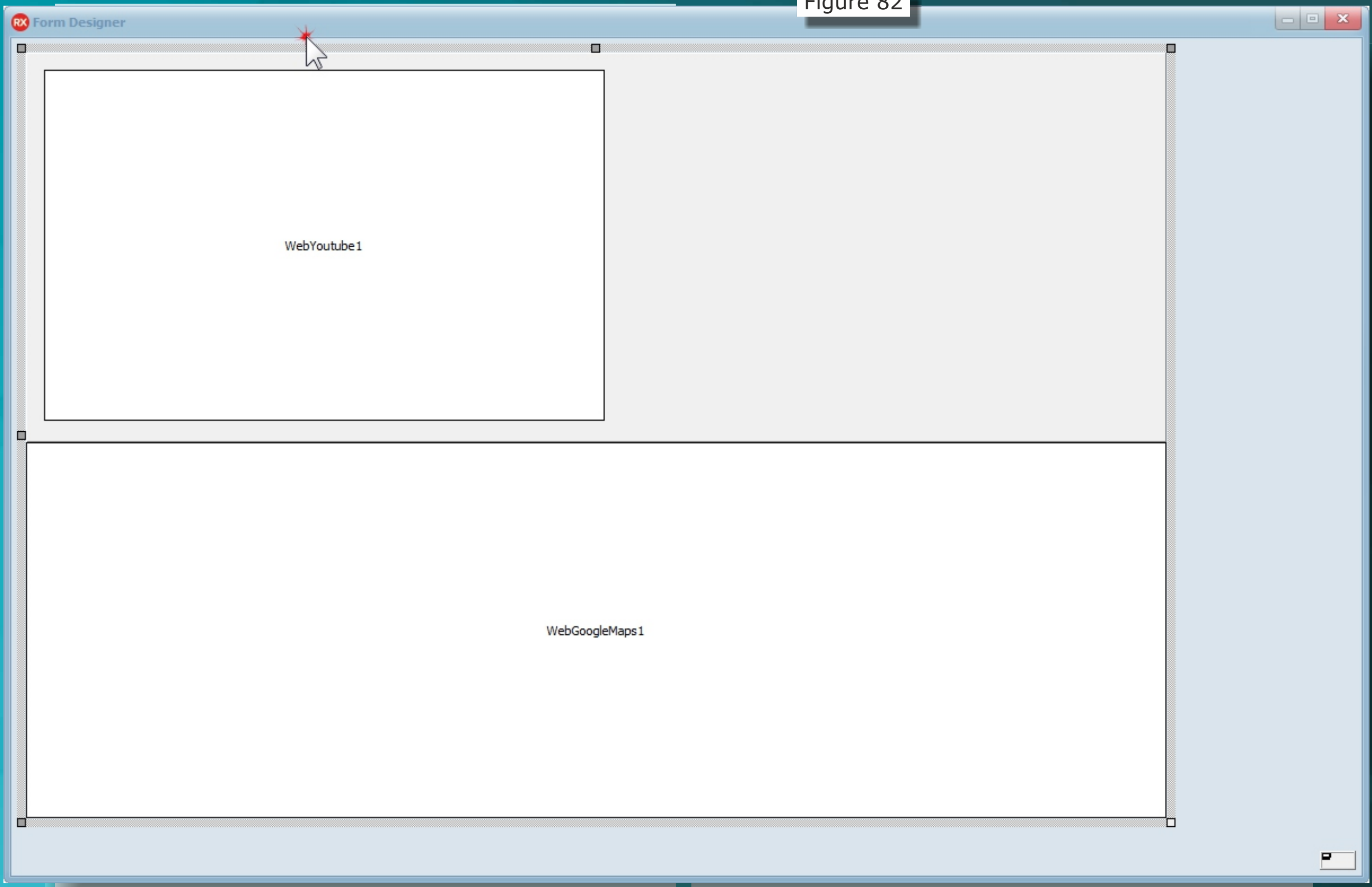

TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS Web Project
localhost:80
Zoeken
Lazarus - Browse /Laza...
TMS Software | VCL, F...
Meest bezocht
TMS WEB Core: Integrating Web content: Youtube and Google Maps
TMS WEB Core
Beast of the Green Hell: Mercedes-AMG GT...
More videos
The Math Behind Quantum Computers
0:13 / 1:35
YouTube
Kaart
Satelliet
Orange
Bathurst
Blue Mountains National Park
Yengo National Park
City of Lake Macquarie
Central Coast
Gosford
Sydney
Kanangra-Boyd National Park
Wollongong
Bowral
Kiama
Goulburn
Nowra
Canberra
Queanbeyan

Figure 81
Components for Google Maps and YouTube are available. This allows for easy integration of these services into your web applications. Maps can be accessed in source code to add markers, calculate and visualize routes etc. The only thing you always need to provide are your user credentials for the different services. Mostly, this means you have to provide an API key.

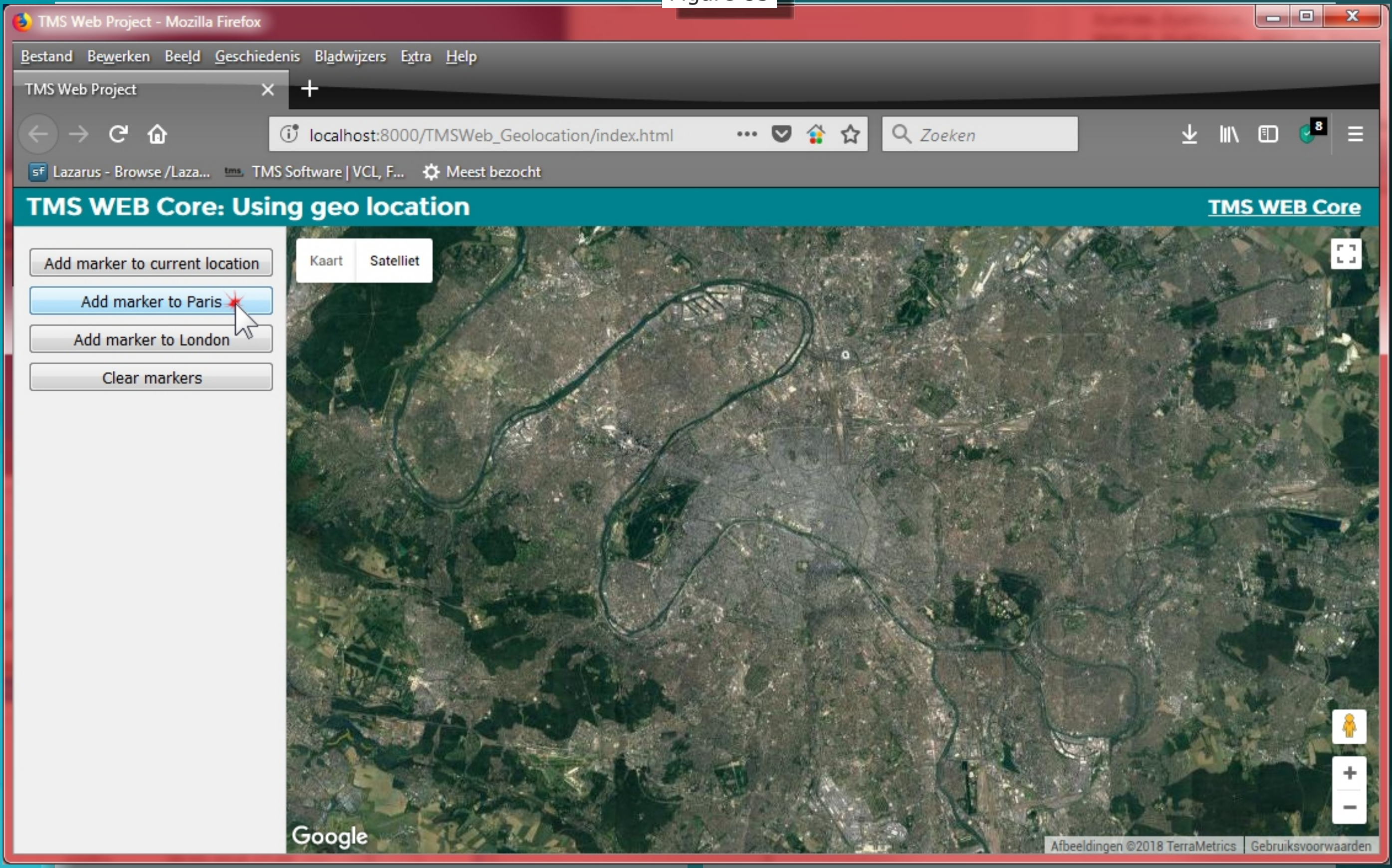
Figure 82



```
procedure TForm4.WebFormCreate(Sender: TObject);
begin
 WebGoogleMaps1.APIKey := '[redacted]';
 // provide your Google maps API key here;
 WebGoogleMaps1.Align := alClient;
end;
```

Code Listing 15

Figure 83


TMS Web Project - Mozilla Firefox
Bestand Bewerken Beeld Geschiedenis Bladwijzers Extra Help
TMS Web Project
localhost:8000/TMSWeb_Geolocation/index.html
Zoeken
Lazarus - Browse /Laza...
TMS Software | VCL, F...
Meest bezocht
TMS WEB Core: Using geo location
TMS WEB Core
Add marker to current location
Add marker to Paris
Add marker to London
Clear markers
Kaart
Satelliet
Google
Afbeeldingen ©2018 TerraMetrics
Gebruiksvoorwaarden

```
implementation

{$R *.dfm}

procedure TForm4.WebButton1Click(Sender: TObject);
begin
  if WebGeoLocation1.HasGeolocation then
    WebGeoLocation1.GetGeolocation;
end;

procedure TForm4.WebButton2Click(Sender: TObject);
var
  lat,lon: double;
begin
  lat := 48.8566;
  lon := 2.3522;
  WebGoogleMaps1.SetCenter(lat,lon);
  WebGoogleMaps1.AddMarker(lat,lon,'Paris');
end;

procedure TForm4.WebButton3Click(Sender: TObject);
var
  lat,lon: double;
begin
  lat := 51.5074;
  lon := 0.1278;
  WebGoogleMaps1.SetCenter(lat,lon);
  WebGoogleMaps1.AddMarker(lat,lon,'London');
end;

procedure TForm4.WebButton4Click(Sender: TObject);
begin
  WebGoogleMaps1.ClearMarkers;
end;

procedure TForm4.WebFormCreate(Sender: TObject);
begin
  WebGoogleMaps1.APIKey :=
 '[redacted]';
end;

procedure TForm4.WebGeoLocation1Geolocation(
Sender: TObject; Lat, Lon,
  Alt: Double);
begin
  WebGoogleMaps1.SetCenter(lat,lon);
  WebGoogleMaps1.SetZoom(11);
  WebGoogleMaps1.AddMarker(lat,lon,'Home');
end;

end.
```

Code Listing 16

Figure 84

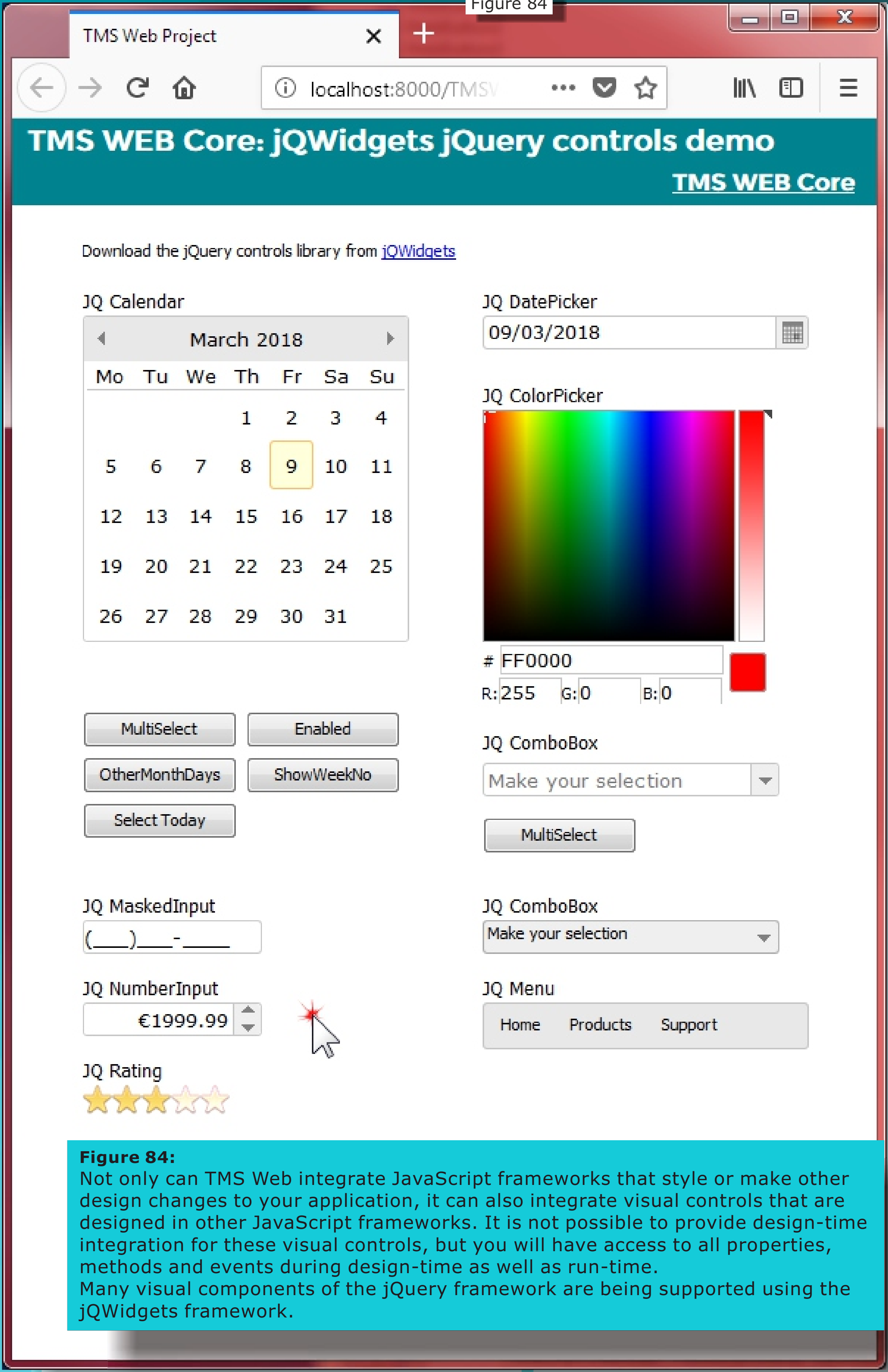
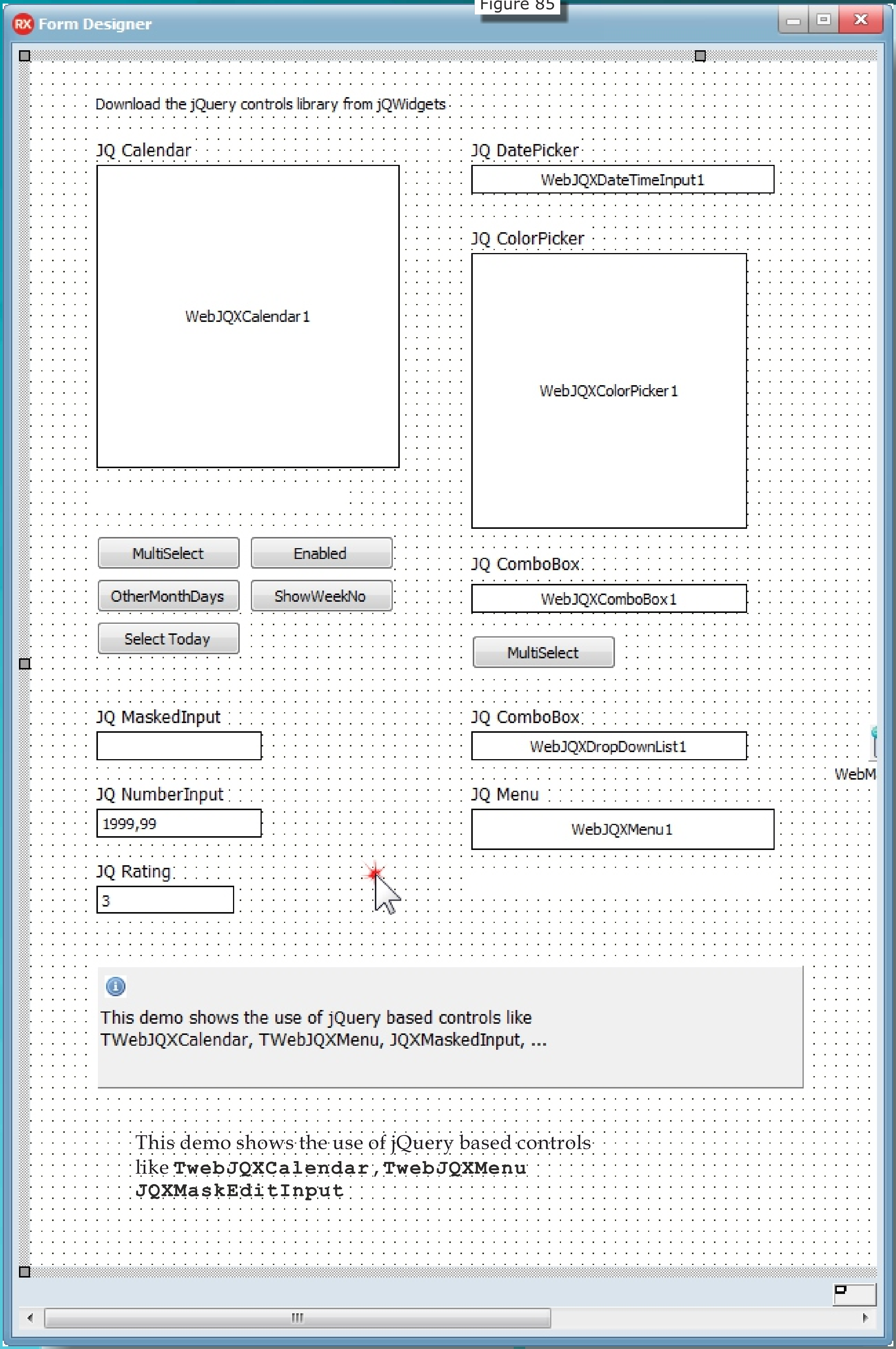


Figure 84:
Not only can TMS Web integrate JavaScript frameworks that style or make other design changes to your application, it can also integrate visual controls that are designed in other JavaScript frameworks. It is not possible to provide design-time integration for these visual controls, but you will have access to all properties, methods and events during design-time as well as run-time.
Many visual components of the jQuery framework are being supported using the jQWidgets framework.

TMSWeb_jQWidgets.dproj

Figure 85


Form Designer

Download the jQuery controls library from jQWidgets

JQ Calendar

WebJQXCalendar1

JQ DatePicker

WebJQXDateTimeInput1

JQ ColorPicker

WebJQXColorPicker1

MultiSelect | Enabled | OtherMonthDays | ShowWeekNo | Select Today

JQ ComboBox

WebJQXComboBox1

MultiSelect

JQ MaskedInput

JQ ComboBox

WebJQXDropDownList1

JQ NumberInput

1999,99

JQ Menu

WebJQXMenu1

JQ Rating

3

This demo shows the use of jQuery based controls like TWebJQXCalendar, TWebJQXMenu, JQXMaskedInput, ...

This demo shows the use of jQuery based controls like **TwebJQXCalendar**, **TwebJQXMenu** **JQXMaskEditInput**

TMSWeb_jQWidgets.dproj

```
var
 Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.WebButton1Click(Sender: TObject);
begin
 WebJQXCalendar1.MultiSelect := not WebJQXCalendar1.MultiSelect;
end;

procedure TForm2.WebButton2Click(Sender: TObject);
begin
 WebJQXCalendar1.OtherMonthDays := not WebJQXCalendar1.OtherMonthDays;
end;

procedure TForm2.WebButton3Click(Sender: TObject);
begin
 WebJQXCalendar1.WeekNumbers := not WebJQXCalendar1.WeekNumbers;
end;

procedure TForm2.WebButton4Click(Sender: TObject);
begin
 WebJQXCalendar1.Enabled := not WebJQXCalendar1.Enabled;
end;

procedure TForm2.WebButton5Click(Sender: TObject);
begin
 WebJQXComboBox1.MultiSelect := not WebJQXComboBox1.MultiSelect;

end;

procedure TForm2.WebButton6Click(Sender: TObject);
begin
 WebJQXCalendar1.Date := Now;
end;

procedure TForm2.WebFormCreate(Sender: TObject);
var
 I: Integer;
begin
 for I := 1 to 10 do
 begin
  WebJQXComboBox1.Items.Add('Item  ' + IntToStr(I));
  WebJQXDropDownList1.Items.Add('Item  ' + IntToStr(I));
 end;

end;

procedure TForm2.WebJQXCalendar1DateClick(Sender: TObject;
 Event: TJQXCalendarEventArgs);
begin
 WebLabel3.Caption := DateTimeToStr(Event.Date);
end;

procedure TForm2.WebJQXMenu1ItemClick(Sender: TObject;
 Event: TJQXMenuEventArgs);
begin
 WeblabelMenu.Caption := 'Item:  "' + Event.Source.Caption + '"  clicked';
end;

end.
```

Code Listing 17

ENUMERATED TYPES AND ASSOCIATED ATTRIBUTES PAGE 1/14

BY PAUL NAUTA

starter expert DX

ABSTRACT

Enumerated Types are a powerful tool to delimit the possible values for a variable or parameter. They are simple to understand, and simple to use in your code. But it becomes a bit more difficult when you want to expose them to your users in the User Interface. Then we need some conversion of an enumerated value to a value a user can understand. This brings the need for attributes to Enumerated Types. This article describes a possible implementation using generic lists of record-like structures, with inheritance functionality to make this useable for many

THE REQUIREMENT

I will use as an example an Enumerated Type for the State of a Change Request. The type definition could look like:

```
TCRState = ( csRegistered, csAnalysis,
csApproved, csDesign, csDevelopment,
csTesting, csReleased, csCancelled,
csRejected );
```

For each State there are some properties like Name (the Name to show in the UI), Description (more detailed definition of the State) and OpenState (is work still needed or already going on?). This could lead to following value matrix:

Enumeration	Name	OpenState	Description
csRegistered	Registered	True	The CR was registered but needs Analysis
csAnalysis	Analysis	True	The CR is being analyzed
csApproved	Approved	True	The CR is approved, budget available, work can start
csDesign	Design	True	The Design for the CR is going on
csDevelopment	Development	True	The Code Development was started
csTesting	Testing	True	The Code is being Tested
csReleased	Released	False	The Code was Released into Production
csCancelled	Cancelled	False	Work on the CR was stopped
csRejected	Rejected	False	After Analysis, the CR was rejected

Many more properties could be possible, like business logic when is it allowed to reach e.g. **crCancelled** (not from **csReleased**, **crRejected**, **crCancelled**, *but probably on all other States*), but I will use the set above as a starter. Name looks a bit curious as it could be retrieved from the Enumeration, but think of a different language: in Dutch you still have **csReleased** but Name could be 'Vrijgegeven'.
My first implementations used a record structure like:

```
TCRStateRecord = RECORD
 CRState      : TCRState;
 Name    : String;
 OpenState  : Boolean;
 Description : String;
END;
```

These records were stored in an array of records with associated functionality to find items and to retrieve the special attribute values.
Over time, the number of similar implementations grew considerably. The array of records was replaced by a **TList** of records, but each time special **TLists** were needed, with specific functionality for **GetItems**, **Finding**, **Sorting** etc. On the other hand, there was a great deal of similarity between them, so I started to look for a more generic approach using inheritance.

TENUMITEM

Records are simple, but they lack inheritance functionality. So, when you want to have a common ancestor for records representing an Enumerated Item, then you need to switch to a **CLASS**. This leads to following class definition:

```
TEnumBasic = CLASS
PRIVATE
 FDescription :String;
 FEnumName   :String;
 FEnumOrd    :Integer;
 FName      :String;
PUBLIC
 PROPERTY  Description:String READ FDescription WRITE FDescription;
 PROPERTY  EnumName   :String READ FEnumName;
 PROPERTY  EnumOrd    :Integer READ FEnumOrd;
 PROPERTY  Name      :String READ FName     WRITE FName;
END;
```

You will miss here the Enumerated Type itself as property. The problem is that the actual `Enumerated Type` cannot be known at this moment. The **Generics** Construct

```
TEnumItem< ET >= CLASS
```

could be a solution for that, but the ET (**Enumerated Type**) is not a class type so generic parameter references like

```
NewItem:TEnumItem< ET >
```

are not possible. You need to specialize them like

```
NewItem:TEnumItem< TCrState >
```

As I could not solve this in a straight forward way, I use the trick of a derived type:

```
TEnumItem< ET >= CLASS( TEnumBasic )
PRIVATE
 FEnumerator:ET;
PROTECTED
 PROCEDURE  SetEnumOrd( CONST Value : Integer ); VIRTUAL;
 PROCEDURE  SetName( CONST Value : String ); VIRTUAL;
PUBLIC
 PROPERTY  Enumerator :String READ FEnumerator;
END;
```

The purpose of **SetEnumOrd** and **SetEnumName** will be discussed later. Via this inheritance the **ET** can be specialized whereas parameter references could be to its ancestor, like:

```
NewItem:TEnumBasic;
```

Later we will need some more functionality on the **TEnumItem**, but let us first see how the **TCrState Enumerated Type** could be implemented:

```
TCrStateItem = CLASS( TEnumItem< TCrState > )
PRIVATE
 FOpenState :Boolean;
PUBLIC
 PROPERTY  CrState  :TCrState READ FEnumerator WRITE FEnumerator;
 PROPERTY  OpenState:Boolean  READ FOpenState  WRITE FOpenState;
END;
```

It inherits of course the attributes from its ancestors and has one additional attribute: **`OpenState`** plus a pseudo attribute: **`CrState`** which is just an easy representation of the Enumerator. Not really needed but sometimes very useful.

TENUMLIST

Now it is time to look how we can store the **`TEnumItems`**. By starting **`Generics`** for **`TEnumItem`**, it is logical to use Generics for the list of **`TEnumItems`** as well. Also here it came out that an inheritance trick was needed, similar to **`TEnumItem`**, so the basic setup looks as follows:

```
TEnumBasicList< EB : TEnumBasic, CONSTRUCTOR > = CLASS( TObjectList< EB > )
PUBLIC
 CONSTRUCTOR   Create; OVERLOAD;
END;
```

Where in most cases developers would use T as the class identifier, I use **`EB`** as abbreviation for **`TEnumBasic`** for clarity. The ancestor class is an **`TObjectList`** because compared to a **`TList`** it has the capability of automatically freeing object entries when they are removed from the list or when the list is freed.
Please notice the **`CONSTRUCTOR`** constraint which implies that the actual **`EB`** type must be a class that defines a default constructor (*a public parameterless constructor*). As a result, methods within **`TEnumBasicList`** may construct instances of **`EB`** using its default constructor, without knowing anything about **`EB`** itself (*no minimum base type requirements*). To reference this basic class, we need a NickName as follows:

```
TEnumBasicListExt = CLASS( TEnumBasicList< TEnumBasic >
```

Basically, it determines the basic classes as the ultimate ancestors. It is used in the following derived class definition:

```
TEnumList< ET; EI: TEnumItem< ET >, CONSTRUCTOR> = CLASS( TEnumBasicListExt )
PRIVATE
 PROCEDURE  Initialize;
 FUNCTION  GetEnumItem( Value : ET ) : EI;
 FUNCTION  GetItem( Index : Integer) : EI;
PUBLIC
 CONSTRUCTOR Create; OVERLOAD;
 PROPERTY  EnumItems[ Index : ET ] : EI READ GetEnumItem; DEFAULT;
 PROPERTY  Items[ Index : Integer ] : EI READ GetItem;
END
```

Here I use **`ET`** as abbreviation for Enumerated Type (like **`TCrState`**) and **`EI`** for **`TEnumItem`**. The Class Definition needs both and because they are of a different type, they must be separated by a semicolon. For **`EI`** the type is **`TEnumItem<ET>`**, thus an **`EnumItem`** specialized for **`ET`**. But **`ET`** is no type thus we cannot make that part of the definition constraint, so at this moment it is still undetermined. It will only become defined when we define the List for e.g. the **`TCrStates`**:

```
TCrStateList = CLASS( TEnumList< TCrState, TCrStateItem > );
```

But let us go back to the definition of the **`TEnumList`**. It is derived from **`TEnumBasicListExt`** which returns Items typed as **`EB`**. We need **`GetItems`** to perform a simple type casting to **`EI`**:

```
FUNCTION TEnumList< ET, EI >.GetItem( Index : Integer ) : EI;
BEGIN
 Result := EI( INHERITED Items[ Index ] );
END;
```

We want to use the **`TCrStateList`** by reference to an **`Enumerator`** instead of an Index, like:

```
PROCEDURE ShowCrStateDescription( Value : TCrState );
VAR
 AList  : TCrStateList;
BEGIN
 AList  := TcrStateList.Create;
 ShowMessage( AList[ Value ].Description );
END;
```

This requires the default **`EnumItems`** property using **`GetEnumItem`**. Before discussing in detail how **`GetEnumItem`** is implemented, let us first look how to Initialize the list.

INITIALIZATION

TEnumList is a list of **TEnumItem.** So, it is logical to create a **TEnumItem** for every **Enumerator** in the **Enumerated Type.**
We can create more, but why would you need two entries with the same **Enumerator?** Only for a different Name?
You could think of **csDevelopment** having Names like '**Internal Development**' and '**Outsourced Development**'. But in such case, there seems to be a functional difference between them, so it is better to split **csDevelopment** into **csDevelopmentInt** and **csDevelopmentExt.**
TEnumList is therefore a list of unique ordinalities,which is created via:

```
PROCEDURE TEnumList< ET, EI >.Initialize;
VAR  iEnumOrd : Integer;
     rEnumItem : EI; rEnumType : ET;
BEGIN
 FOR iEnumOrd := FEnumInfo.TypeData.MinValue TO
FEnumInfo.TypeData.MaxValue DO
  BEGIN
   rEnumItem  := EI.Create;
   rEnumItem.SetEnumOrd( iEnumOrd );
   rEnumItem.SetName( rEnumItem.FEnumName );
   INHERITED Add( rEnumItem );
 END;
END;
```

This procedure requires some more explanation.
One could expect to use

```
 FOR iEnumOrd := Low( ET ) TO High( ET ) DO
```

But the compiler has no notion what **ET** really means (**E2008: Incompatible types**) so we must resolve that during runtime. For this purpose, the **FEnumInfo** parameter is used which is

```
CONSTRUCTOR TEnumList< ET, EI >.Create;
BEGIN
 INHERITED Create;
 FEnumInfo  := TypeInfo( ET );
 Initialize;
END;
```

Via the **TypeInfo.TypeData** record we can determine the minimum and maximum ordinality values. In the Initialize procedure we set the **EnumOrd** and the Name properties. But because **EnumOrd** uniquely defines each element, **EnumName** and Enumerator can be derived from it. This is the purpose of the **SetEnumOrd** procedure on **TEnumItem:**

```
PROCEDURE TEnumItem< ET >.SetEnumOrd( CONST Value : Integer );
BEGIN
 FEnumName  := GetEnumName( TypeInfo( ET ), Value );
 FEnumOrd   := Value;
 FEnumerator := TRttiEnumerationType.GetValue<ET>( FEnumName );
END;
```

The determinitation of **FEnumerator** is the tricky point here because a simple type casting like:

```
FEnumerator := ET( FEnumOrd ;
```

does not compile (**E2089: Invalid Typecast**), for the same reason as given above. But the Run Time Type Information (**RTTI**) contains the solution as needed here.
With **FEnumName** determined, it is a simple step to give the Name the value of **FEnumName** as default value. Later we can always update to what we really want.
While working on this **Class**, it becomes more and more clear that the **TEnumList** should have full control over **TEnumItem**. That is also the reason that **EnumName**, **EnumOrd** and Enumerator are read-only values on **TEnumItem**. These values can only be set via protected procedures, in this case only from **TEnumList**. Once the list is created via **Initialize**, **Additions** and **Deletes** should no longer be possible. Therefore, we override these **functionalities** via:

```
FUNCTION TEnumList< ET, EI >.Add( CONST Value : ET ) : Integer;
BEGIN
 RAISE Exception.Create( 'Addition is not allowed' );
END;

PROCEDURE TEnumList< ET, EI >.Delete( Index : Integer );
BEGIN
 RAISE Exception.Create( 'Delete is not allowed' );
END;
```

Because of this, we must call **INHERITED Add** and not just **Add** in the **Initialize** procedure. By making Initialize part of the constructor, the list is automatically populated at create of the list.

RETRIEVING AND UPDATING ITEMS

It is now time to discuss the **GetEnumItem** function. It looks like:

```
FUNCTION TEnumList< ET, EI >.GetEnumItem( Value : ET ) : EI;
VAR
 iOrd : Integer;
 iEnum : Integer;
BEGIN
 iOrd  := ConvertEnumToOrd( Value );
 FOR iEnum := 0 TO Count - 1 DO
  IF ( Items[ iEnum ].EnumOrd = iOrd )THEN
   BEGIN
    Result  := Items[ iEnum ];
    Break;
   END;
END;
```

Basically, this is very straight forward, apart from the **ConvertEnumToOrd** function. Due to the same problems as noticed before, following does not compile:

```
iOrd >= Ord( Value );
```

As workaround I developed **ConvertEnumToOrd** which is defined as:

```
FUNCTION TEnumList< ET, EI >.ConvertEnumToOrd( Value: ET ): Integer;
BEGIN
 CASE System.TypInfo.GetTypeData( TypeInfo( ET ) )^.OrdType OF
  otUByte, otSByte : Result := PByte( @Value )^;
  otUWord, otSWord : Result := PWord( @Value )^;
  otULong, otSLong : Result := PInteger( @Value )^;
 END;
END;
```

This is not something you easily invent, but it is a customization of the **TRttiEnumerationType.GetName<T{: enum}>(AValue: T) function** in the **Rtti** unit.
With the **GetEnumItem** functionality it is possible to set e.g. the Description like:

```
VAR
 AList : TCrStateList;
BEGIN
 AList := TcrStateList.Create;
 AList[ csAnalysis ].Description := 'The CR is being analyzed';
```

The Name cannot be set in this way because it is read-only, mainly because you want Name to be unique. And the List needs to determine if a new Name value is acceptable. This can be done via:

```
PROCEDURE TEnumList< ET, EI >.ModifyNameDescr( Value : ET; Name : String;
  Description : String );
VAR
 rItem    : EI;
 rExisting : EI;
BEGIN
 rExisting := FindName( Name );

 IF ( rExisting = NIL ) THEN
  BEGIN
   rItem        := GetEnumItem( Value );
   rItem.SetName( Name );
   rItem.Description   := Description;
  END
 ELSE
  RAISE Exception.Create( 'Name ''' + Name + ''' is already in use with item ' + rExisting.EnumName );
END;
```

In this procedure also the **Description** is set, because that is what you would like to do in most of the simple implementations of **TEnumList.**
To check if the Name already exists, we need the **FindName** procedure:

```
FUNCTION TEnumList< ET, EI >.FindName( Name : String ) : EI;
VAR
 iEnum   : Integer;
 iCompare : Integer;
BEGIN
 Result  := NIL;

 FOR iEnum := 0 TO Count - 1 DO
  BEGIN
   iCompare  := AnsiCompareText( Items[ iEnum ].Name, Name );

   IF ( iCompare = 0 ) THEN
    BEGIN
     Result := Items[ iEnum ];
     Break;
    END;
  END;
END;
```

This procedure does a case insensitive check, which is logical in view of the nature of the Name field: registered and **REGISTERED** should mean the same **csRegistered.** With this functionality available, it is now possible to create our **TCrStateList** as follows:

```
CONSTRUCTOR TCrStateList.Create;

BEGIN
 INHERITED Create;

 ModifyNameDescr( csRegistered, 'Registered', 'The CR was registered but needs Analysis' );
 ModifyNameDescr( csAnalysis, 'Analysis', 'The CR is being analyzed' );
 ModifyNameDescr( csApproved, 'Approved', 'The CR is approved, budget available,
                                          work can start' );
 ModifyNameDescr( csDesign, 'Design', 'The Design for the CR is going on' );
 ModifyNameDescr( csDevelopment, 'Development', 'The Code Development was started' );
 ModifyNameDescr( csTesting, 'Testing', 'The Code is being Tested' );
 ModifyNameDescr( csReleased, 'Released', 'The Code was Released into Production' );
 ModifyNameDescr( csCancelled, 'Cancelled', 'Work on the CR was stopped' );
 ModifyNameDescr( csRejected, 'Rejected', 'After Analysis, the CR was rejected' );

 EnumItems[ csRegistered ].OpenState  := True;
 EnumItems[ csAnalysis ].OpenState   := True;
 EnumItems[ csApproved ].OpenState   := True;
 EnumItems[ csDesign ].OpenState    := True;
 EnumItems[ csDevelopment ].OpenState := True;
 EnumItems[ csTesting ].OpenState   := True;
END;
```

Of course, it is not necessary to make these configurations part of the **Creator.** You could also choose to load the information from an Ini File or from a **DataSet** but that would require some extra functionality.

NAME CASING

In most cases you want to control the casing of the Name, and maybe even switch the casing. This could look like:

```
TNameCasing = ( ncNone, ncUpper, ncLower, ncFirst );
```

ncNone means no requirements, **ncUpper** means we want **UpperCase, ncLower** means we want **LowerCase** while **ncFirst** means the first character is **UpperCase,** the rest is **LowerCase** (*in fact a nice example to use TEnumList for these definitions!*). To convert a Name to the required **NameCasing,** following function was developed:

```
FUNCTION ConvertNameCasing( Name : String; Casing : TNameCasing ) : String;
BEGIN
 CASE Casing OF
  ncUpper : Result := UpperCase( Name );
  ncLower : Result := LowerCase( Name );
  ncFirst : Result := UpperCase( Copy( Name, 1, 1 ) ) + LowerCase( Copy( Name, 2, Length( Name ) ) );
 ELSE
      Result := Name;
 END;
END
```

Which **NameCasing** should be applied, should be defined on the **TEnumList** and be propagated to each of its items. This leads to following additions to the definition of **TEnumBasicList:**

```
PRIVATE
 FNameCasing  : TNameCasing;
 PROCEDURE SetNameCasing( Value : TNameCasing );
PUBLIC
 PROPERTY NameCasing : TNameCasing READ FNameCasing WRITE
SetNameCasing;

where SetNameCasing gets following implementation:
PROCEDURE TEnumBasicList< EB >.SetNameCasing( CONST Value :
TNameCasing );
VAR
 iEnum    : Integer;
BEGIN
 FNameCasing := Value;
 FOR iEnum := 0 TO Count - 1 DO
  Items[ iEnum ].SetName( Items[ iEnum ].Name );
END;
```

In this way all **EnumItems** are adapted in the same way when the **NameCasing** is changed, to keep the casing uniform across the list. The actual setting is arranged in the **SetName** procedure on **TEnumItem:**

```
PROCEDURE TEnumItem< ET >.SetName( CONST Value : String );
BEGIN
 FName := ConvertNameCasing( Value, TEnumBasicListExt(FOwner).NameCasing );
END;
```

The **NameCasing** field is not available on **TEnumItem,** so we must get it from **TEnumBasicList.** Until now we did nothing to tell that an **TEnumItem** belongs to a specific **TEnumList.** Therefore I introduced on **TEnumBasic** the field **FOwner** as a **Pointer.** It must be set when we create the Item, which can be done in the Initialize procedure via:

```
 rEnumItem.FOwner := Self;
```

Because **FOwner** is just a Pointer, it must be type casted to **TEnumBasicListExt.** Defining **FOwner** directly as **TEnumBasicListExt** is not possible due to several mutual dependent type definitions.

SORTING

It is a general requirement that lists can be sorted. The **TCrState** could be sorted on **EnumOrd** (*natural ordering*) or on Name while several more possibilities could exist. To use the standard Sort procedure of **TObjectList,** you need to provide the **Comparer** functionality via its Interface to the **Sort** command. So how should this **Comparer** look like? Again, I want to be generic, therefore one compare function should be able to compare on arbitrary **Numeric** or **String Fields**, case sensitive or not, ascending or descending. The definition of the **Comparer** therefore becomes:

```
TCompareValues< EB : TEnumBasic > = CLASS( TComparer< EB > )
PRIVATE
 FCaseSensitive : Boolean;
 FSortAscending : Boolean;
PUBLIC
 CONSTRUCTOR Create; OVERLOAD;
 FUNCTION Compare( CONST Left, Right : EB ) : Integer; OVERRIDE;
 PROPERTY CaseSensitive : Boolean READ FCaseSensitive WRITE FCaseSensitive;
 PROPERTY SortAscending : Boolean READ FSortAscending WRITE FSortAscending;
END;
```

the **RTTI** sees the context as **TEnumItem<ET>** and not as an item possibly derived from **TEnumItem<ET>** like **TCrStateItem.** Consequently, all additional properties in derived items will lead to compiler errors like 'Property XXXX does not exist', when running this code. The solution is to determine the **ClassType** during runtime, because on that moment knowledge on the actual type of the items is available.

For sure you want some further explanation what is happening here. First, we determine the actual type of **TEnumType** via the **ClassType.** Second, we determine the property on which we want to compare, which is specified on the List. Third, we can determine the value of the Property. But the **GetValue** procedure causes a lot of problems: the result is of **TValue** which often leads to errors when used as **TValue.AsString.** Following procedure could resolve that as well:

```
FUNCTION ConvertValueToString( AProperty : TRttiProperty; Value : TValue;
  ForSorting : Boolean ) : String;
VAR
 iAdd     : Integer;
 rPropInfo  : PTypeInfo;
BEGIN
 CASE AProperty.PropertyType.TypeKind OF
  tkEnumeration :
   BEGIN
    rPropInfo := AProperty.PropertyType.Handle;
    IF ( rPropInfo = System.TypeInfo( Boolean )) THEN
     IF ( Value.AsOrdinal = 0 ) THEN Result:='False' ELSE Result:='True'
    ELSE
     Result := GetEnumName( rPropInfo, Value.AsOrdinal );     END;
  tkInteger :
   BEGIN
  iAdd    := 0;
  IF ForSorting THEN iAdd := 1000000;
  Result   := IntToStr( Value.AsInteger + iAdd    END;
 ELSE
  Result   := Value.AsString;
 END;
END;
```

The curious point here is that **Booleans** have **TypeKind** = **tkEnumerated** (*one would expect tkBoolean but that does not exist*) so **GetEnumName** is not useful. **iAdder** is used to avoid compare problems like **2 > 11**, because the **Comparer** does a straight forward string compare. **ForSorting** is a Parameter that was added for some other purposes. With all these additions we can do the actual Sort via:

It has following implementation for the **Compare function**:

```
FUNCTION TCompareValues< EB >.Compare( CONST Left, Right : EB ) : Integer;
BEGIN
 IF FCaseSensitive THEN
  Result := AnsiCompareStr( Left.GetCompareValue, Right.GetCompareValue )
 ELSE
  Result := AnsiCompareText( Left.GetCompareValue, Right.GetCompareValue );

 IF NOT FSortAscending THEN
  Result := - Result;
END;
```

I could not define this comparer for **EI** items because of the unknown dependency to **ET**. Instead, this function uses **EB** Items to compare (*in fact, this was one of the reasons to split between* **TEnumBasicList** *and* **TEnumList)**. But on that basic level, only the basic fields are present, not additional fields like 'OpenState'. The compiler simply rejects those additional fields. The trick is to introduce a 'calculated field' via a special **GetCompareValue** function on **TEnumItem**, using an **FCompareField**, defined on **TEnumBasicList**:

```
FUNCTION TEnumItem< ET >.GetCompareValue : String;
VAR
 rContext  : TRttiContext;
 rType     : TRttiType;
 rProperty : TRttiProperty;
 oValue    : TValue;
BEGIN
 rType    := rContext.GetType( ClassType );
 rProperty := rType.GetProperty(
TEnumBasicListExt(FOwner).FCompareField );
 oValue    := rProperty.GetValue(Self );
 Result    := ConvertValueToString( rProperty, oValue, True );
```

Because the **Compare** Function will call this procedure on **TEnumBasic**, we need to make **GetCompareValue** available also on **TEnumBasic**. A **VIRTUAL, ABSTRACT** function on **TEnumBasic** is sufficient to actually use the **GetCompareValue** on **TEnumItem** (*provided it is specified with* **OVERRIDE)**. The code snippet above again poses some implementation challenges because with the first guess:

```
sField := TEnumBasicListExt( FOwner ).FCompareField;
 Result := GetPropValue( Self, sField );
```

```
PROCEDURE TEnumBasicList< EB >.Sort;
BEGIN
 FComparer.FCaseSensitive := FCaseSensitive;
 FComparer.FSortAscending := FSortAscending;
 INHERITED Sort( FComparer AS IComparer< EB > );
END;
```

The **FComparer** is defined as

```
FComparer : TCompareValues< EB >;
```

and is created in the creator of

TEnumBasicList as:

```
FComparer := TCompareValues< EB >.Create;
```

In normal cases you would define **FComparer** as:

```
FComparer : IComparer< EB >;
```

because only the interface would be needed, but in this case, we added **FCaseSensitive** and **FSortAscending** as Fields, which we want to change on the fly. Therefore, we first need to create **TCompareValues< EB >** and then set the appropriate attributes. We also need to free it in Destroy. Obviously, such type is not qualified for automatic freeing by **TObjectList**, as it is not a list item.

EXPORT
One of the use cases could be to see the list of Names in a **StringList.** We can arrange this as follows:

```
PROCEDURE TEnumList< ET, EI >.PopulateStringList( List : TStrings;
   Field : String );
VAR
 rContext  : TRttiContext;
 rType     : TRttiType;
 rProperty : TRttiProperty;
 oValue    : TValue;
 iEnum     : Integer;
BEGIN
 List.Clear;
 rType     := rContext.GetType( TypeInfo( EI ) );
 rProperty := rType.GetProperty( Field );
 FOR iEnum := 0 TO Count - 1 DO
  BEGIN
   oValue  := rProperty.GetValue( TObject( Items[ iEnum ] )
   List.Add( ConvertValueToString( rProperty, oValue, False ) );
  END;
END;
```

Here we can specify the name of the Field to export. For the rest, this procedure uses functionality already discussed earlier.
For our example we will also create a list of **`Open States`** via:

```
PROCEDURE TCrStateList.PopulateOpenStates( List : TStrings );
VAR
 iEnum : Integer;
BEGIN
 List.Clear;
 FOR iEnum := 0 TO Count - 1 DO
  IF Items[ iEnum ].OpenState THEN
   List.Add( Items[ iEnum ].Name );
END;
```

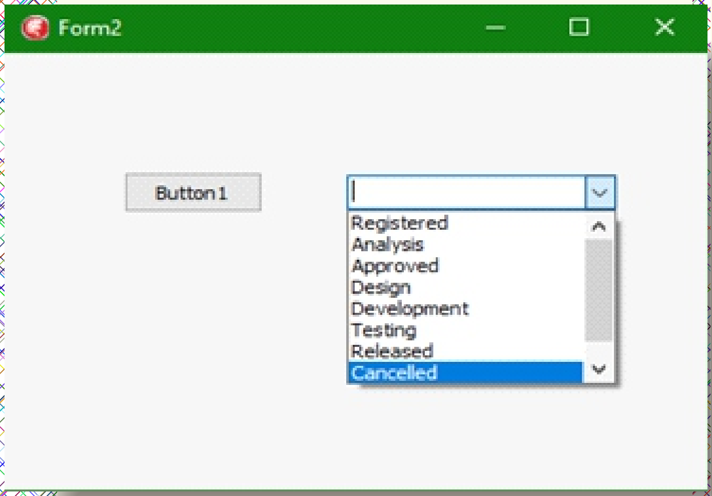
DEMONSTRATION
It is now time for a demonstration. You can create a Demo Application with a Button and 2 **`ComboBoxes`**. The button must have an **`OnClick`** event like:

```
PROCEDURE TForm1.Button1Click( Sender : TObject );
VAR
 lStates : TCrStateList;
BEGIN
 lStates := TCrStateList.Create;
 TRY
  lStates.PopulateStringList( ComboBox1.Items, 'Name' );
  lStates.NameCasing    := ncUpper;
  lStates.CompareField := 'Name';
  lStates.SortAscending := False;
  lStates.Sort;
  lStates.PopulateOpenStates( ComboBox2.Items );
  ShowMessage( 'Done' );
 FINALLY
  lStates.Free;
 END;
END;
```

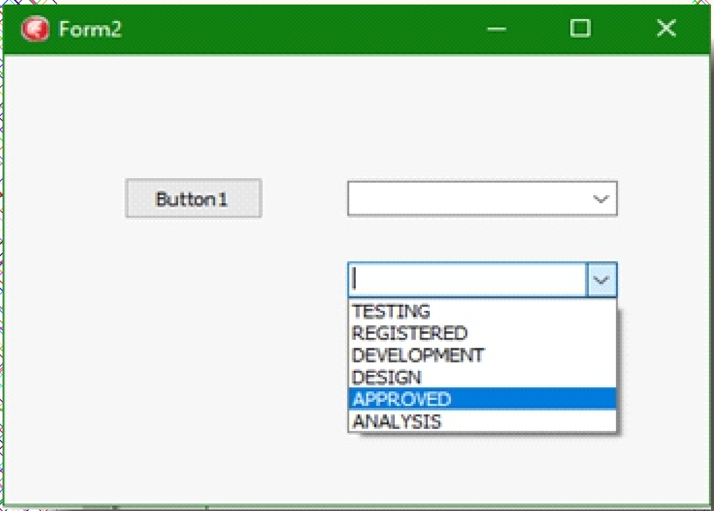
`ComboBox1` gets an **`OnDblClick`** event like:

```
PROCEDURE TForm1.ComboBox1DblClick(Sender: TObject);
VAR
 lStates : TCrStateList;
 sName : String;
BEGIN
 lStates := TCrStateList.Create;
 TRY
  sName := ComboBox1.Text;
  IF ( sName <> '' ) THEN
   ShowMessage( FStates.FindName( sName ).Description );
 FINALLY
  lStates.Free;
 END;
END;
```

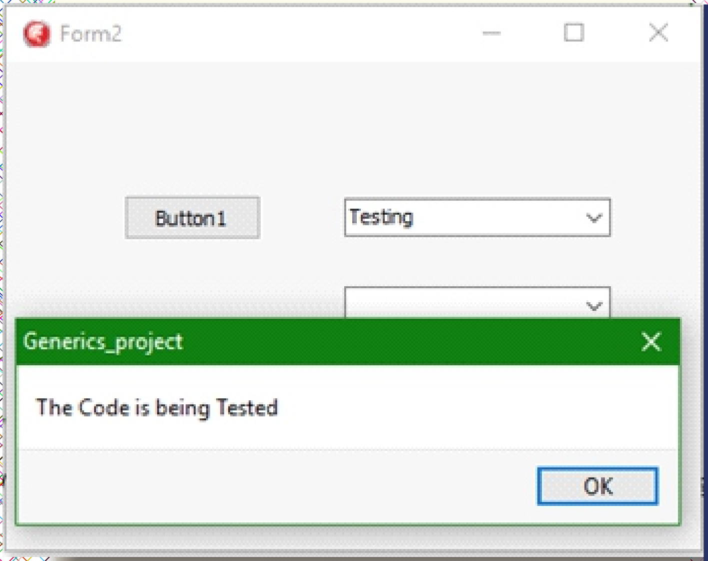
Pressing the button will populate **ComboBox1** with the Names in natural order:



ComboBox2 is populated with only the **Open States**, in **UpperCase** and in descending order:



When you double click on the selected item in **ComboBox1**, you get a message representing the Description of the selected item, like:



The Code is being Tested

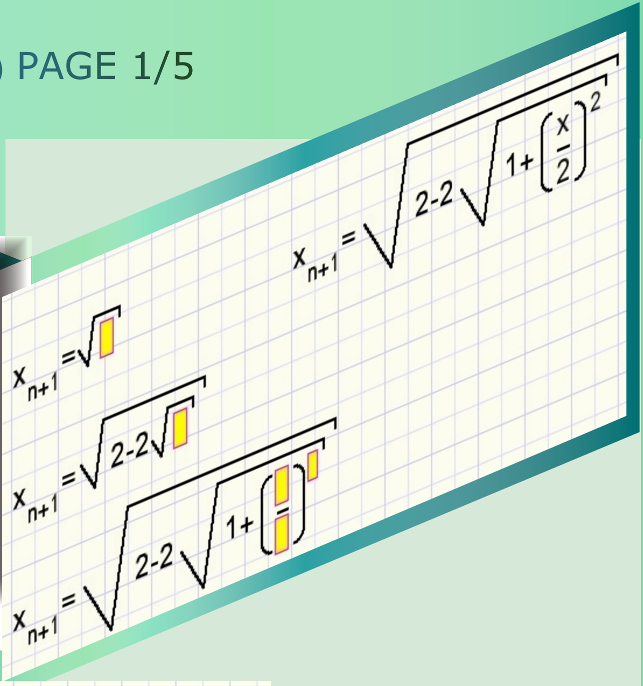
We see that **TCrStateList** can be used with only a few lines of additional coding; almost all the work is done via the generic **TEnumItem** and **TEnumList**.

CONCLUSION
With Generics, Inheritance and some RTTI functionality it is possible to develop generic functionality to maintain Attributes for Enumerated Types. The meaning of each Enumerator can simply be defined with the standard Description field. During development I gathered quite some knowledge on the functioning of Generics and RTTI and I needed to use some not obvious tricks. Also, the code used for Generics is more difficult to read/understand than 'normal' code. The reward however is, that the resulting generic functionality can be tailored easily for case specific needs via derived types. It will for example be possible to specify completely the associated business logics. This can be of great importance because all relevant settings can be maintained on one place, in one derived TEnumItem / TEnumList combination.

BY DAVID DIRKSE

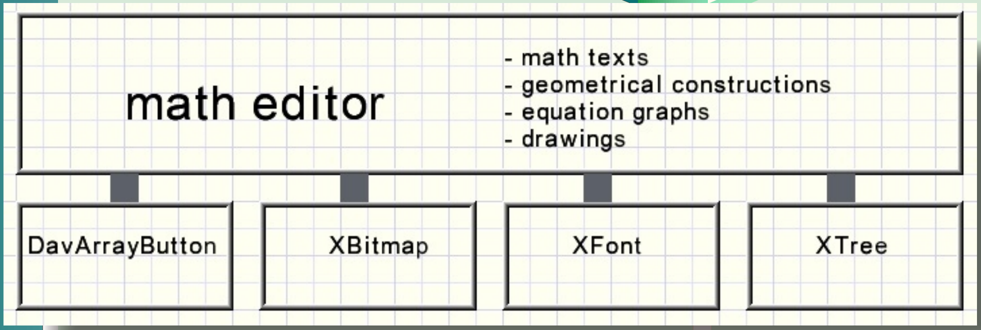
starter expert

For some years now, I have been working on a wysiwyg math editor. The main purpose is to supply a vehicle for easy communication of mathematical texts.
Such documents include:drawings:
(lines, circles, planes...)
geometrical constructions :
(bisectors, regular polygons, inscribed- and circumscribed circles....)
equation graphs :
(also parametric- and implicit functions)
text : (including fractions, roots, powers, indices...)



$x_{n+1}=\sqrt{2-2\sqrt{1+\left(\frac{x}{2}\right)^2}}$
$x_{n+1}=\sqrt{\square}$
$x_{n+1}=\sqrt{2-2\sqrt{\square}}$
$x_{n+1}=\sqrt{2-2\sqrt{1+\left(\frac{\square}{\square}\right)^{\square}}}$

THIS DELPHI-7 PROJECT RESTS ON FOUR PILLARS



DavArrayButton
Own component with rows and columns of buttons. Used in menus to select properties, elements and operation modes.

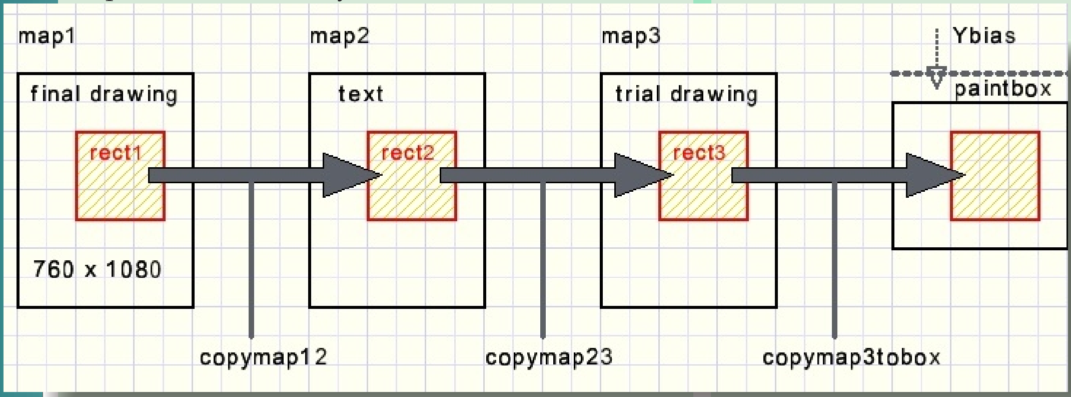
XBitmap
Own class. Extension of Tbitmap with clipping rectangle, improved stretchdraw and floodfill, dash-dot lines of larger penwidths, lines with arrows.

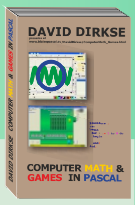
Xfont
Own class of scalable fonts with Greek characters and geometrical symbols.

Xtree
Own class with tree structure for text editing. Implements UNDO system as well.

These classes are described in my book: "computer math and games in Pascal".

GENERAL DATA FLOW.
All data is stored in vectorized form as array of records.
Drawing is done in three Xbitmaps with size 760 * 1080 (hor * vert) pixels.





Map1: drawings
Supplies background for

Map2: text

Map3: Trial drawings: during the drawing of lines etc. or geometrical constructions.

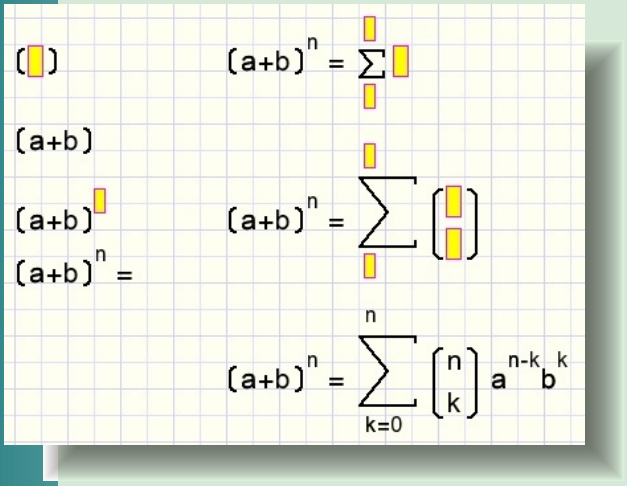
Part of a Xbitmap may be erased by copying that part from the left Xbitmap.
Finally, images are displayed by copying them to a paintbox.
This paintbox displays part of Xbitmap3, this part is selectable by a scrollbar.

During editing only modified rectangles of the Xbitmaps are transferred.
Cursors are painted in the paintbox and are erased by copying part of map3 to the paintbox.

The general idea is that elements are placed on a Xbitmap canvas.
Elements occupy rectangular spaces.
There are graphical elements and text elements.
The graphical elements include lines, circles, arcs etc. This article focusses on text.

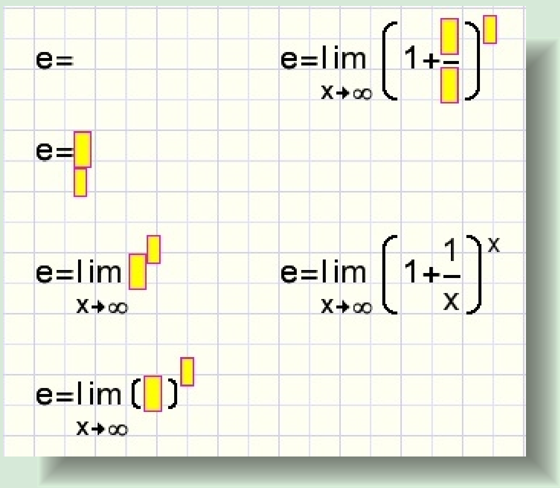
Text consists of lines holding characters and macro's. A macro holds one or more lines.
Some macro's add graphical symbols such as a root or fraction line.

To give an impression I show the steps in typing the Newton Binomium:
(look left top – down, than right-top down)
The editor paints empty lines with a yellow background.


$(\,)$
$(a+b)$
$(a+b)^{\,}$
$(a+b)^n =$
$(a+b)^n = \sum$
$(a+b)^n = \sum \left(\,\right)$
$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^k$

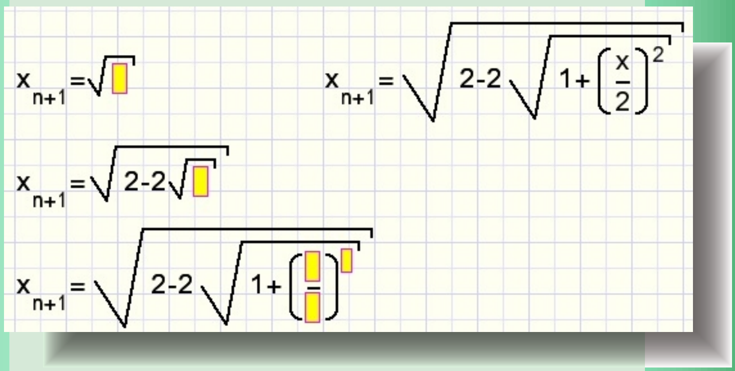
Starting left-top, a (...) macro is added, on its line a+b is typed.
After that a power macro is placed, n is typed in the line.
After the = character a sigma macro is added and its lines are filled with k=0, n, and a (n over k) macro. Etcetera.
The picture shows that the sigma macro automatically adjusts its size when the (n over k) macro was added.

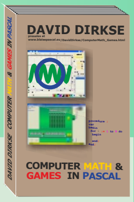
Second example is the constant e, base of the natural logarithm:


$e=$
$e=$
$e=\lim_{x\to\infty}$
$e=\lim_{x\to\infty}(\,)$
$e=\lim_{x\to\infty}\left(1+\frac{}{}\right)$
$e=\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^x$

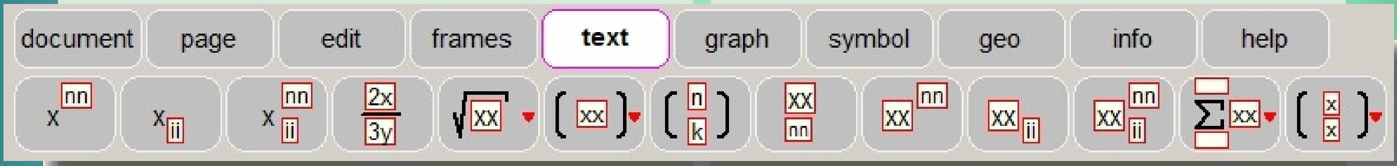
Please note the the (..) macro adjusts its size when the fraction macro is added.
Also the power x position is automatically adjusted.
Macro's are selected by a mouseclick on a (davarray) button.

Last example: the chord bisector formula:


$x_{n+1}=\sqrt{\,}$
$x_{n+1}=\sqrt{2-2\sqrt{\,}}$
$x_{n+1}=\sqrt{2-2\sqrt{1+\left(\frac{}{}\right)}}$
$x_{n+1}=\sqrt{2-2\sqrt{1+\left(\frac{x}{2}\right)^2}}$

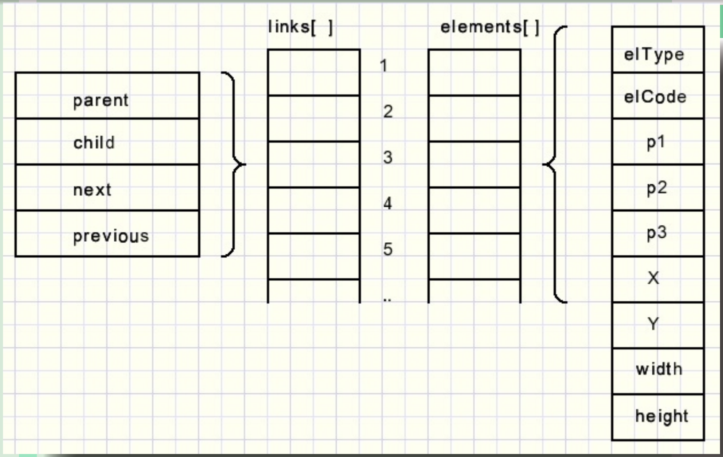


Below is the macro elements menu from with the macro elements were selected:



HOW IS THIS ALL DONE?
Elements that hold other elements within are called a parent. The elements within the parent are called children.
Each element type has three procedures which are element type specific:

1. creation : add its properties to a list
2. calculation of it's contents : postion of children, its own width and height
3. painting its contents : child lines with characters + graphics such as root symbols



The position of characters in a line are relative to the parent line.
The position of lines in a macro are relative to the parent macro.
So, when moving an element its children do not change.
Note that an element never calculates it's own position,the parent has to take care of that.

All other procedures such as adding, deleting, replacing elements are element type independent.
Inserting a character in a line or a line in a textbox is the same operation however the result is quite different as the parent recalculates the childrens positions differently.

When a child changes in size due to add or delete operations, it informs its parent which will recalculate itself.
If its size changes again, the parent calls its parent...etc.
The finally changed element is erased and repainted on the Xbitmap canvas.
That's all.

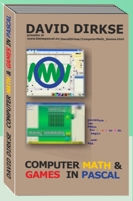
The properties of a text element are stored in an array called element[...]
The parent-child relation is stored in a separate array callind links[ ]
The indices to both arrays are the same.
Below is the record structure with links[3] and element[3] records expanded:

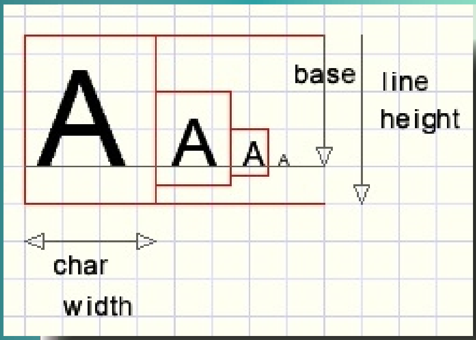
Element[1] is the document itself. All other elements are the children of element 1.

A parent may have many children.
The parent points to the first child, each child itself points to the next.
Links[ ] is a bidirectional linked list.
The elements records have the following meaning:

eltype: frame , macro, line, character
elCode: element specific property. Character code for characters. Frame code for frames.
P1: element dependent: font number for characters, alignment code for lines.
P2: element dependent: top symbol (` , ') for characters, subcode for macros.
P3: element dependent: base for characters,lines and macro's.
For frames: pointer to the first graphic element in the frame.
X : horizontal position relative to parent.
Y : vertical position relative to parent
width , height: element width and height.

The baseline takes care of vertical text alignment.
Characters of different font, height and color may occupy a line side by side.
Below characters are shown with height 80,40,20,10 placed on the base of the line.





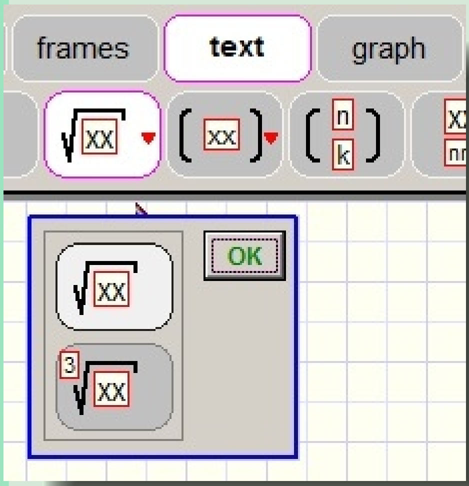
To show a little of the code: below is listed the procedure which is the core of the automatic resizing and recalculation schemes rectangles.

```
procedure processELchange(el : dword);
//element reports change to its parent el
//el can be macro,line,column (not char or block)
//purpose is
//1. to call for recalculation of parents
//2. set area to be erased
//3. set element that needs repainting
var oldw,oldh : smallInt;
   Done : boolean;
   r1 : Trect;
begin
 eraseflag := false;
 spaceflag := false;
 repeat
  repaintELnr := el;
  oldw := element[el].width;
  oldh := element[el].height;
  r1 := getElementRect(pageNr,el);
  updateRect(eraseflag,eraseRect,r1);
  recalculateElement(el);
  if (oldw <> element[el].width) or (oldh <>
element[el].height) then
   begin
    getParent(el,el);
    Done := element[el].elType = elBlock;
   end
    else Done := true;
 until Done;
end;
```

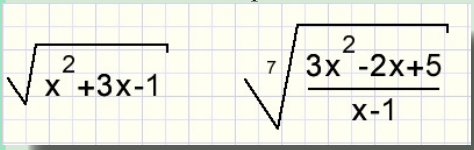
`repaintELnr` is the element that is finally repainted.
`recalculateElement(el)` goes to a big case statement which in turn calls the proper recalculation procedure for this element type.
If due to recalculation the elements' width or height changes, its parent is called. An elBlock type element is just a part of the document to hold lines or graphic symbols.
`getParent(el1,el2)` supplies the parent element of el2 in el1. El1,el2 are of type cardinal.
The updateRect procedure combines rectangles.

CURSOR MOVEMENT
This was an unexpected big coding effort requiring its own unit (textcontrol_unit).
Eventually a next article might illustrate that.

ELEMENT (RE)CALCULATION
As an example next I show how a root macro is (re)calculated.
At creation the start is the selected font height.
The root macro has two choices as shown in the elements submenu:



There is a normal square root and a root macro with extra line for 3rd and higher rank roots.
Here are two examples:


$$\sqrt{x^2+3x-1} \qquad \sqrt[7]{\frac{3x^2-2x+5}{x-1}}$$

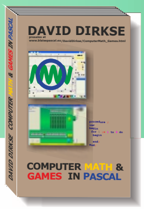
The parent of a macro such as root always is a line. Lines are the children of either other macro's or frames such as the document itself.

There are two procedures for the calcualtion:
1. recalculateRoot: for the width, height and (x,y) position of the line children.
2. calculateRoot: for the calculation of the root graphic symbol

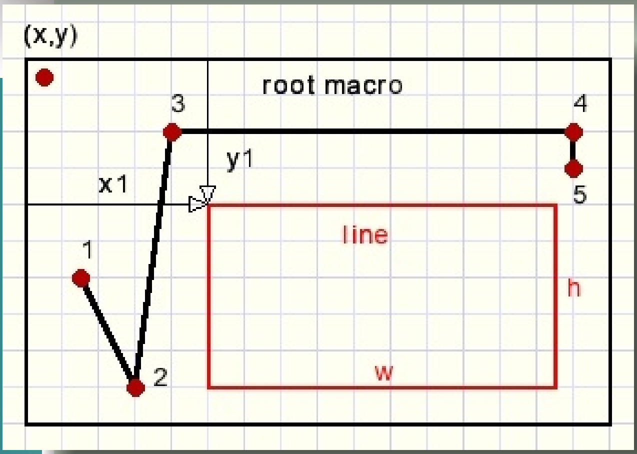
Calculateroot is only needed when painting the root, however the data supplied may be helpful for recalculateRoot. All macro's in general use this method, some recalculateXXX use the calculateXXX, others don't.

The type of root is found in the elements' p2 parameter which is equal to 0 for a simple square root and 1 for an extra line for higher rank roots.

When calculating the root macro it is important to remember that the line children's width and height are known. The goal is to calculate the lines (x,y) postions and the root macro's own width and height.



```
type  Txy = record
         x : smallInt;
         y : smallInt;
        end;
  TXY07 = array[0..7] of TXY;
.....
procedure calculateRoot(var rc:TXY07; pw:byte; el:dword);
//calculate the root symbol coordinates in rc
//pw: penwidth; el:element; lel: child line element
var lel : dword;
    h,w,x1,y1,d2,d3,d5 : smallInt;
    pw2,pw3 : byte;
begin
  pw2 := pw shl 1; //*2
  pw3 := pw2 + pw;
  getChild(lel,el);
  with element[lel] do
  begin
    x1 := x;
    y1 := y;
    h := height;
    w := width;
  end;
  d2 := round(h/2);
  d3 := round(h/3);
  d5 := round(h/5);
  rc[3].x := x1 - pw3;
  rc[2].x := rc[3].x - d5;
  rc[1].x := rc[2].x - d3;
  rc[2].y := y1 + h;
  rc[1].y := rc[2].y - d2;
  rc[3].y := y1 - pw3;
  rc[4].x := x1 + w + pw2;
  rc[4].y := rc[3].y;
  rc[5].x := rc[4].x;
  rc[5].y := rc[4].y + pw3;
end;
```

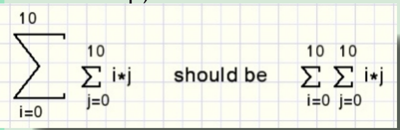


So above procedures asumes that the position of the child line has been calculated already.
This procedure is also called when the macro is painted.

Below is the recalculateroot procedure:

```
procedure recalcRoot(el : dword);
//calculate position of children,width,height
var i,pw,pw2,pw3: byte;
    lel,rel : dword;
    d10,dx,w2,h2,yr,x2,y2 : smallInt; rc : TXY07;
begin
  getChild(lel,el);
  pw := getMacroPenWidth(element[lel].height);

  //---> textpaint unit
  pw2 := pw shl 1;
  w3 := pw2 + pw;
  element[lel].y := 5*pw;
  element[lel].x := 0;
  calculateroot(rc,pw,el);
  d10 := round(0.1*element[lel].height);
  dx := -rc[1].x + d10;
  for i := 1 to 5 do  with rc[i] do x := x + dx;
    with element[lel] do x := rc[3].x + pw3;
      with element[el] do
      begin
        p3 := element[lel].y + element[lel].p3;
        width := rc[4].x + pw3;
        height := rc[2].y + pw;
      end;
  if element[el].p2 = 1 then
    begin //if higher rank root
      getNext(rel,lel);
      with element[rel] do
      begin
        w2 := width;
        h2 := height;
      end;
      yr := element[lel].y + (element[lel].height shr 1);
      x2 := rc[2].x - w2;
      if x2 < 0 then
        begin
          element[lel].x := element[lel].x - x2;
          element[el].width := element[el].width - x2;
          element[rel].x := 0;//d10;
        end
      else element[rel].x := rc[2].x - w2;
      y2 := yr - h2 - pw2;
      if y2 < 0 then
        begin
          element[lel].y := element[lel].y - y2;
          element[el].height := element[el].height - y2;
          element[el].p3 := element[el].p3 - y2;
          element[rel].y := pw2;
        end
      else element[rel].y := yr - h2;
    end;
end;
```

Above structure is very consistent however for the sigma macro an unexpected inconvenience showed up, see below



The second sigma blows up its parents line height and the first sigma recalculates it's height.
Extra code was needed to recognize this situation.
This is a good moment to conclude this description of my math editor. Insert , delete, backspace, cursor movement and UNDO work fine.
Much work has to be done. At present single independent lines are added to the document pages. There is no vertical alignment of lines. More frame elements will be implemented as parents for vertical alignment as well as horizontal centration.
Also copy-paste has to be implemented and insertion / deletion of full pages.

BY KIM MADSEN

starter expert Delphi

LINQ… what is LINQ? Well its a term used in C# which means Language Integrated Query.

The next version of kbmMW will support our own variant of LINQ. In reality we can't make true C# LINQ functionality, because it requires the compiler to be aware about the fundamentals of LINQ, and Delphi is blissfully unaware about such language integrated features.

However, my interpretation of the purpose of LINQ is that its designed to make certain programming tasks easier for the programmer. To get rid of boiler plate code, which is something that I have focused quite allot on in kbmMW v5 and continues to focus on.

So what does LINQ do for us?
It allows us to query, filter, order, calculate, compare, group etc. various types of data in an easy way using the same syntax regardless of what type the (supported) source data is.

Is kbmMW's LINQ fast? Yes and no. Since it hides much functionality from us, more CPU cycles are usually spend than would otherwise have been needed if you coded an optimized algorithm yourself. However since it uses optimized kbmMW features underneath, then some scenarios will probably perform just as well as manually written code.

So the advantage of using kbmMW's LINQ is not as such performance, but rather provides the ability to do quite complex and advanced things with very simple code.

To use kbmMW's LINQ, simply add kbmMWLinq to the uses clause. It will give you a global threadsafe object instance, named Linq, thru which all LINQ functionality originates.

In the following I will show various scenarios that are possible with kbmMW.

1

OPERATING A TSTRINGLIST USING LINQ
This is an example of using LINQ with a regular TStringList.

First we build a string list.

```
var
  sl,sl2:TStrings;
begin
   sl:=TStringList.Create;
   sl.Add('1');
   sl.Add('2');
   sl.Add('3');
   sl.Add('4');
   sl.Add('5');
   sl.Add('6');
   sl.Add('7');
   sl.Add('8');
   sl.Add('9');
   sl.Add('20');
```

Then we use Linq to give us the first 5 of them sorted descending and returning the result as a new TStringlist:

```
sl2:=Linq.Using(sl).First(5).Sort('value:D').AsStrings;
…
sl2.Free;
```

Next we show to how make multiple operations on the same data, without the overhead of re-parsing the source data. First we define the initial Linq stage (the one preparing the source data) as shared, then we return the last 8 items as a TStringList, looks for the Max value, and then calculates a SUM. Since the string list is strings, kbmMW's LINQ assumes that Max/Min functions should operate on a string level, not numeric. However the SUM function can only work on numeric data, and thus will always operate as such:

```
type
  lq:IkbmMWLinqStage;
  s:string;
  d:double;
…
   lq:=Linq.Using(sl).Shared;
   sl2:=lq.Last(8).AsStrings;
…
   sl2.Free;

   s:=lq.Max; // Returns the string 9
   d:=lq.Sum; // Returns 65
```

Operating class instances using LINQ
The next example shows how to use Linq on lists of class instances. For a class to be "Linqable" it must be tagged with the kbmMW_Linq attribute as seen below. In addition the class should be registered as a kbmMW known type and RTTI must be enabled for it.

```
[kbmMW_Linq]
 TMyData = class
 private
   FName:string;
   FAddress:string;
   FAge:integer;
 public
   constructor Create(const AName:string;
                const AAddress:string;
                const AAge:integer);
   property Name:string read FName write FName;
   property Address:string read FAddress write FAddress;
   property Age:integer read FAge write FAge;
 end;
```

One place to register the class to kbmMW is in the units initialization section. Notice that both TMyData and TObjectList are being registered, since we will use both types.

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

Lets prepare some data to play with:

```
var
  lst:TObjectList<TMyData>;
...
   lst:=TObjectList<TMyData>.Create;
   try
     lst.Add(TMyData.Create('Kim','Address 1',49));
     lst.Add(TMyData.Create('Hans','Address 2',22));
     lst.Add(TMyData.Create('Jens','Address 3',33));
     lst.Add(TMyData.Create('Joe','Address 3',77));
     lst.Add(TMyData.Create('John','Address 4',33));
```

If we want to locate the maximum age, we write

```
i:=Linq.Using(lst).Max('Age');
```

or the alphabetically smallest name

```
s:=Linq.Using(lst).Min('Name');
```

or we can return a key/value string list of sorted names with the age

And then we clean up. Also remember to free the returned **TStringList** (sl) when you don't need it any longer.

```
 finally
   lst.Free;
 end;
```

Operating JSON documents using LINQ
First lets prepare some data. This time we use the UsingJSON method. It can take a JSON string, a stream or you can use UsingJSONFile to load the JSON document from a file.

```
 sl:TStrings;
 lq:IkbmMWLinqStage;
begin
   lq:=Linq.UsingJSON('{"result":['+
              '{"ID":1,"name":"kim","date":"2018-01-05T19:05:00.000+08:00"},'+
              '{"ID":2,"name":"kim","date":"2018-01-05T20:05:30.000+01:00"},'+
              '{"ID":3,"name":"kim","date":"2018-01-05T20:05:45.000+01:00"},'+
              '{"ID":4,"name":"kim","date":"2018-01-05T20:06:15.000+01:00"},'+
              '{"ID":5,"name":"kim","date":"2018-01-05T20:06:30.000+01:00"},'+
              '{"ID":6,"name":"kim","date":"2018-01-05T20:06:45.000+01:00"},'+
              '{"ID":7,"name":"kim","date":"2018-01-05T20:07:15.000+01:00"},'+
              '{"ID":8,"name":"kim","date":"2018-01-05T21:07:30.000+01:00"},'+
              '{"ID":9,"name":"kim","date":"2018-01-05T21:07:45.000+01:00"},'+
              '{"ID":10,"name":"kim","date":"2018-01-05T21:08:00.000+01:00"}]}',
            '/result/.*',
            'ID,date');
```

UsingJSON takes 2 and optionally 3 arguments, the JSON data string, a subset pattern match('/result/.*'), and optionally a list of field names or expressions ('ID,date').

The subset is actually a regular expression which is applied to the JSON data, to only select the relevant data on which the Linq methods should operate. In this case we have one property (result) with a sub array, so we use the expression to accept everything starting (path wise) with /result/. We have also told kbmMW that we only want to access the ID and date fields of the JSON document. If we didn't specify that, kbmMW would automatically have figured out all relevant fields that should be accessible. The field names can include expressions, so its thus possible to add fields together or do calculations.

```
i:=lq.Max('ID');
  i:=lq.Min('ID');
  sl:=lq.Sort('date:D').AsStrings('date');
```

In similar way you can query YAML, BSON and MessagePack documents.

Operating XML documents using LINQ
XML documents are structurally more complex than JSON and YAML documents, in the sense

that each node in the document can have attributes in addition to child nodes and data. We must still specify a subset we want to operate on like above, but if we want access to the attributes, we must use the XMLAttr function, that takes two arguments: the node holding the attribute, and the attribute name itself. Since attributes by definition are strings, we have the ability to automatically have the values casted to some other types, like TEXT(size), INTEGER etc.

kbmMW supports the following casts:
INT/INTEGER, VARCHAR2(n), VARCHAR(n), CHAR(n), BOOL, BOOLEAN, AUTOINC, FLOAT, DOUBLE, NUMERIC, REAL, DATETIME, TIMESTAMP, DATE, TIME, LARGEINT, INT64, BLOB, GRAPHIC, CLOB, TEXT(n), CURRENCY, WORD, MEMO, WIDEMEMO and **GUID**.
If n is not given the default value is 20.

```
var
  sl:TStrings;
  lq:IkbmMWLinqStage;
begin
   lq:=Linq.UsingXML('<?xml version="1.0" ?>'+
             '<Dictionary>'+
             ' <Parameters>'+
             '   <Parameter SymbolName="CoDeviceType"'+
             '              ObjectType="VAR"'+
             '              Index="0x1000"'+
             '              SubIndex="0"'+
             '              DataType="UNSIGNED32"'+
             '              AccessType="const" />'+

             '   <Parameter SymbolName="CoErrorRegister"'+
             '              ObjectType="VAR"'+
             '              Index="0x1001"'+
             '              SubIndex="0"'+
             '              DataType="UNSIGNED8"'+
             '              AccessType="ro" />'+

             '   <Parameter SymbolName="CoClearErrorLog"'+
             '              ObjectType="VAR"'+
             '              Index="0x1003"'+
             '              SubIndex="0"'+
             '              DataType="UNSIGNED8"'+
             '              AccessType="rw"'+
             '              Remarks="Write 0 to clear"/>'+
             ' </Parameters>'+
             '</Dictionary>'
            ,'/Dictionary/Parameters/.*/'
            ,'XMLAttr(Parameter,"SymbolName") as "SymbolName->TEXT(40)"'+
            ,'XMLAttr(Parameter,"SubIndex") as "SubIndex→INTEGER"');

   sl:=lq.Sort('SymbolName').AsStrings('SymbolName','SubIndex');
```

sl will now contain a sorted key/value list:

```
CoClearErrorLog=0
CoDeviceType=0
CoErrorRegister=0
```

MORE LINQ FEATURES

KBMMW'S LINQ ALSO SUPPORTS:

- `Count` – Returns the number of items in the given Linq stage.
- `Distinct(fieldnames)` – Returns only items that have unique values in the fields specified by fieldnames.
- `GroupBy(groupfieldnames,aggregatefieldnames)` – Returns records grouped by the groupfieldnames (required), and optionally aggregated values on the fields specified in aggregatefieldnames. You specify aggregation method as a modifier to the field name. Eg. `field1:COUNT`, `field2:MAX`

The output of aggregated fields will be named
'originalfieldname_`COUNT/AVG/SUM/MIN/MAX/STDDEV`' (eg. `field1_COUNT`)

- `Select(fieldexpressions)` – Returns the items exposed by the given expressions. Eg. `Select('SIN(fld1) as fld1, fld2|fkd3 as newfield')`.
  When expressions are used, the resulting
  field will be named Fn where n is the index in the resulting item starting with 1.
  To ensure that you have full control over the names, you can specifically name them using the "as name" method as shown.
- `AsString(fieldname)` – Returns the first item's field value as a string.
- `AsInteger(fieldName)` – Returns the first item's field value as an integer.
- `AsFloat(fieldName)` – Returns the first item's field value as a double.
- `AsVariant(fieldname)` – Returns the first item's field value as a variant.
- `AsDataset` – Returns the data as a dataset.

The ownership of the dataset belongs to the **Linq** stage.
Thus when the **Linq** stage goes out of scope, the dataset is also destroyed.
Functions like `Min, Max, Avg, Sum` and `StdDev` can take zero or one field name.
If zero field names are given, the first known internal field column is used.
In functions like `Distinct, Sort` and `GroupBy` which takes multiple fields,
the fields must be separated by comma (,).
Functions like As…..(fieldname) can take 0 or 1 string argument, or alternatively an integer value.
If no argument is given, the first field is assumed.
If a string argument is given, the field with the given name is returned.
If an integer value is given, the values for the field with the given index (first field is 0) is returned.
Feel free to come with ideas and input for the new **Linq** look alike features in **kbmMW.**
Also remember that **Linq** works with all compilers supported by **kbmMW,**
so you can go "Linq nuts" on **Android, IOS, Linux, OSX, Windows** and **Linux.**

```
procedure TdmMain.MyLog(const ATime:TkbmMWDateTime; const AMessage:string);
var
 l:TMyLog;
begin
  try
   l:=TMyLog.Create;
   try
    l.Time:=ATime;
    l.Info:=AMessage;
    ORMLog.Persist(l);
   finally
    l.Free;
   end;
  except
   on E: Exception do
   begin
     SystemLog.Error('LogSystem',E);
   end;
  end;
end;
```

And at some point before the first time the database log will be used, the database tables needs to be prepared:

```
function TdmMain.PrepareLogDatabase:boolean;
begin
  Result:=false;

  // Open log database. dbLog is the (in this sample, SQLite) connection pool for the database.
  FORMLog.OpenDatabase(dbLog);

  // Create tables if they are not available.
  // Upgrade them if they exists and needs upgrade.
  if not FORMLog.CreateOrUpgradeTable([TMyLog]) then
  begin
    Log.Fatal('Unable to create or upgrade log database: '+dbLog.Database+'.
        Server not started!');
    exit;
  end;

  Result:=true;
end;
```

And finally I define the TkbmMWVirtualLogManager which in turn calls the MyLog method whenever there is something for this logmanager to handle.

Because I want to put timestamp in a separate field in the database, I redefine the logformatter of this logmanager to only include a few select type of information.

REST EASY WITH KBMMW #10 – LOGGING TO A DATABASE

BY KIM MADSEN

starter expert Delphi

In the upcoming release, the logging feature will have been improved in various ways. One of the new inclusions is the TkbmMWVirtualLogManager and its interface IkbmMWVirtualLogManager.
The virtual log manager can for example be used for logging select logs to a database, which this short blog will focus on.
I will in this sample, use kbmMW's ORM to handle the database access, however any traditional database access method could have been used instead.

10

n the upcoming release, the logging feature will have been improved in various ways.
One of the new inclusions is the **TkbmMWVirtualLogManager** and its interface **IkbmMWVirtualLogManager**.

The virtual log manager can for example be used for logging select logs to a database, which this short blog will focus on.

I will in this sample, use **kbmMW's ORM** to handle the database access, however any traditional database access method could have been used instead.

```
TdmMain = class(TDataModule)
private
 FDBLogManager:IkbmMWVirtualLogManager;
 FORMLog:TkbmMWORM;
public
 property ORMLog:TkbmMWORM read FORMLog;
```

Since I want to use the ORM for log storage handling, I need to define a class describing the storage.

```
[kbmMW_Table('name:myLog')]
 TMyLog= class
 private
   FID:kbmMWNullable<string>;

   FTime:TkbmMWDateTime;
   FInfo:kbmMWNullable<string>;
   FComments:kbmMWNullable<string>;

 public
   [kbmMW_Field('name:id, primary:true,
      generator:shortGUID',ftString,40)]
   property ID:kbmMWNullable<string> read FID write FID;

   [kbmMW_Field('name:time',ftDateTime)]
   [kbmMW_NotNull]
   property Time:TkbmMWDateTime read FTime write FTime;

   [kbmMW_Field('name:info',ftWideMemo)]
   property Info:kbmMWNullable<string> read FInfo write FInfo;

   [kbmMW_Field('name:comments',ftWideMemo)]
   property Comments:kbmMWNullable<string>
          read FComments write FComments
end;
```

The TSystemLog class needs to be registered:

```
initialization
  TkbmMWRTTI.EnableRTTI([TMyLog]);
  kbmMWRegisterKnownClasses([TMyLog]);
```

We add a method that we can call to persist the log entry. Notice that if the method is unable to persist the log due to some database issue, an error will be logged on the SystemLog, which is a standard, always existing, alternative logger in kbmMW. It will default output to debug view on Windows, or LogCat on Android.

```
// Prepare database oriented log manager.
  FDBLogManager:=TkbmMWVirtualLogManager.Create(
   procedure(const AType:TkbmMWLogType; const ALevel:TkbmMWLogLevel;
       const AOrigin:string; const ATime:TkbmMWDateTime; const AString:string)
   begin
     // Specifically do not accept messages comming from kbmMW's internals itself,
     // since those could be generated from the database layers, resulting in deadlock.
     if pos('kbmMW',AOrigin)=0 then
       MyLog(ATime,AString);
   end
  );

 // Setup the log formatter to only include a few things in the log string.
  FDBLogManager.LogFormatter:=TkbmMWSimpleLogFormatter.Create;
  FDBLogManager.LogFormatter.Columns := [mwlfcLogType,mwlfcLogString,mwlfcLogData];
  Log.LogManager:=FDBLogManager;
```

Now every time you use `Log.Info/Log.Error/Log.Warning/Log.Fatal` or any of the other log methods, the log will be appended to the myLog table in the database.

REST EASY WITH KBMMW PART 9 PAGE 1/4
DATABASE 4 BY KIM MADSEN

COMPONENTS 4 DEVELOPERS

starter expert DX Delphi

Data augmentation and XML

his blog post will focus on one way of augmenting data returned from a database using the ORM, serving this as a wellformed XML result to REST client's using as little code as possible.
kbmMW's ORM is pretty good at fetching data from a database based on a class.
Sometimes we want to augment the class with additional data, before returning the data to a client.
This we can use the virtual table attribute for.

8

WHAT DOES DATA AUGMENTATION MEAN?
Wiki *Data augmentation adds value to base data by adding information derived from internal and external sources within an enterprise.*
Data is one of the core assets for an enterprise, making data management essential. Data augmentation can be applied to any form of data, but may be especially useful for customer data, sales patterns, product sales, where additional information can help provide more in-depth insight.
Data augmentation can help reduce the manual interventation required to developed meaningful information and insight of business data, as well as significantly enhance data quality.

AN EXAMPLE:
We have a class TPerson, which is used by the ORM to persist and retrieve persons from the person database table. The person might refer to a company, via a companyId which is a GUID. This is all straight forward.

```
[kbmMW_Table('name:person')]
 TPerson = class
 private
   FID:kbmMWNullable&lt;string&gt;;
   FName:kbmMWNullable&lt;string&gt;;
   FCompanyID:kbmMWNullable&lt;string&gt;;
 public
   [kbmMW_Field('name:id, primary:true, generator:shortGUID',ftString,40)]
   property ID:kbmMWNullable&lt;string&gt; read FID write FID;

   [kbmMW_Field('name:name',ftString,50)]
   [kbmMW_NotNull]
   property Name:kbmMWNullable&lt;string&gt; read FName write FName;

   [kbmMW_Field('name:companyId',ftString,40)]
   property CompanyID:kbmMWNullable&lt;string&gt; read FCompanyID write FCompanyID;
 end;
```

However let's say we want to provide an augmented REST interface to the persons information, where we want to add additional fields, like company name.

```
[kbmMW_VirtualTable]
 [kbmMW_Root('person',[mwrfIncludePublic])]
 TAugmentedPerson = class
 private
   FID:kbmMWNullable<string>;
   FName:kbmMWNullable<string>;
   FCompanyID:kbmMWNullable<string>;
  <strong> FCompanyName:kbmMWNullable<string>;</strong>
 public
   [kbmMW_Field('name:id',ftString,40)]
   property ID:kbmMWNullable<string> read FID write FID;

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

   [kbmMW_Field('name:companyId',ftString,40)]
   property CompanyID:kbmMWNullable<string> read FCompanyID
        write FCompanyID;

   <strong>[kbmMW_Field('name:companyName',ftString,50)]
   property CompanyName:kbmMWNullable<string> read FCompanyName
        write FCompanyName;</strong>
 end;
```

What you can see is that an additional class is defined, which purpose primarely is to marshal augmented TPerson data to REST clients.

We have told that the TAugmentedPerson (*which by outset has no relation to TPerson class wise*), is a virtual table, which means it does in fact not live in any databases, but it can still be used as the output for queries.

So let's put together an augmented query:

```
kbmMW_Service('name:REST, flags:[listed]')]
[kbmMW_Rest('path:/rest')]
TsvcRest = class(TkbmMWCustomHTTPSmartService)
public
  [kbmMW_Rest('method:get, path:persons')]
  function GetPersons:TObjectList&lt;TAugmentedPerson&gt;;
end;
```

```
// Return augmented persons.
function TsvcRest.GetPersons:TObjectList&lt;TAugmentedPerson&gt;
begin
 Result:=dmMain.ORM.QueryList&lt;TAugmentedPerson&gt;(
     'SELECT p.ID as ID,p.Name as Name, '+
     ' CompanyID, c.Name as CompanyName '+
     'FROM uData.TPerson p, uData.TCompany c '+
     'WHERE c.ID=p.Company');
end;
```

This will make a query, augmenting the person data with a company name and returning it as a JSON object to the REST client.

This is all well and fine.

But lets say that the REST interface wants to return this list of TAugmentedPerson's as XML?

This is now easily done in kbmMW by adding a responseMimeType to the kbmMW_Rest attribute of the GetPersons method.

```
[kbmMW_Service('name:REST, flags:[listed]')]
[kbmMW_Rest('path:/rest')]
TsvcRest = class(TkbmMWCustomHTTPSmartService)
public
  [kbmMW_Rest('method:get, path:persons, <strong>responseMimeType:application/xml</strong>')]
  function GetPersons:TObjectList&lt;TAugmentedPerson&gt;;
end;
```

this will result in simple XML document representing the resulting list of TAugmentedPerson's. You can ask kbmMW to add XML declarations, namespaces and types if you want to by adding the properties declared:true, typed:true to the kbmMW_Rest method attribute.

But looking at the XML, it is still not perfect. It's outer node is called TObjectList<person>, which is not a terribly nice tag name for an XML node. We are missing a way to redefine how kbmMW is to name the TObjectList<TAugmentedPerson> class.

Usually one would use the kbmMW_Root attribute in combination with our own class definition, to specify any name changes to root elements when marshalling data into or out from object instances, similarly to how TAugmentedPerson was defined.

However since we do not define TObjectList<TAugmentedPerson> anywhere (it is implicitely being defined as the result from our ORM.QueryList call, we have no place to specify our settings/attributes for that particular class.

Next version of kbmMW provides a new kbmMW_Alias attribute which handles this issue.

Basically what it does is to declare any class as an attribute wise alias to any other class/classes, like this

```
[kbmMW_Root('persons',[mwrfIncludePublic])]
[kbmMW_Alias]
TAugmentedPersonList = class(TObjectList&lt;TAugmentedPerson&gt;);
```

The kbmMW_Alias can have zero or one argument. If an argument is given, it can be a class reference, or an array of class references. If no argument is given, kbmMW automaticallyh defines TAugmentedPersonList to be an alias to TObjectList<TAugmentedPerson> due to the class inheritance.

As we never define TObjectList<TAugmentedPerson> anywhere, we can not refer to it as a class reference, why we use kbmMW's way to implicitely determine the class by not providing any arguments for the kbmMW_Alias attribute.

In reality we will usually never instantiate any TAugmentedPersonList instances. It is only being used as a "placeholder" for defining attributes (on the class level) on types we don't directly declare ourselves, like the TObjectList<TAughmentedPerson>.

Now the xml will look pretty, with the outer node named <persons> containing a number of inner nodes named <person> which each of them includes the companyName in addition to other TPerson related data.

As a side note, the [kbmMW_VirtualTable] attribute can now also take an argument, namely the actual database class for which this class is a virtual class for.

It would be possible to define [kbmMW_VirtualTable(TPerson)]

It informs kbmMW about that any queries made for TAugmentedPerson (which is not really a table found in the database), where the ORM can not deduce from any kbmMW SQL query statement, where to pickup data from, then it should use TPerson as the goto data table.

So this is now legal:

```
var
 ap:TAugmentedPerson;
begin
 ap:=ORM.Query&lt;TAugmentedPerson&gt;(['Name'],['Kim']);
end;
```

It will return first found record in the person table, which matches the person named Kim and return that as a TAugmentedPerson instance.

Only fields matching will be filled. Hence in this case the CompanyName value is null since we did not provide any value for it via the query.

But we are getting an object instance which allows us to add our own value for CompanyName, thus in practice augmenting the TPerson look alike object with additional information.

New quantum dot could make quantum communications possible : http://www.extremetech.com/

KBMMW PROFESSIONAL AND ENTERPRISE EDITION V. 5.05.10 RELEASED!

NEW! TKBMMWISAPIRESTSERVERTRANSPORT REST CAPABLE ISAPI SERVER SIDE TRANSPORT.

- **RAD Studio 10.2 Tokyo support including Linux support** (in beta).
- **Huge number of new features** and improvements!
- **New Smart services and clients** for very easy publication of functionality and use from clients and REST aware systems without any boilerplate code.
- **New ORM OPF** (Object Relational Model Object Persistence Framework) to easy storage and retrieval of objects from/to databases.
- **New high quality random functions.**
- **New high quality pronouncable password generators.**
- **New support for YAML, BSON, Messagepack** in addition to JSON and XML.
- **New Object Notation framework which JSON, YAML, BSON and Messagepack** is directly based on, making very easy conversion between these formats and also XML which now also supports the object notation framework.
- **Lots of new object marshalling improvements,** including support for marshalling native Delphi objects to and from YAML, BSON and Messagepack in addition to JSON and XML.
- **New LogFormatter support** making it possible to customize actual logoutput format.
- **CORS support in REST/HTML services.**
- **High performance HTTPSys transport for Windows.**
- Focus on central performance improvements.
- Pre XE2 compilers no longer officially supported.
- Bug fixes
- **Multimonitor** remote desktop V5 (VCL and FMX)
- RAD Studio and Delphi XE2 to 10.2 Tokyo support
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OSX client and server support!
- **Native PHP**, Java, OCX, ANSI C, C#, Apache Flex client support!
- **High performance LZ4 and Jpeg compression**
- **Native high performance** 100% developer defined app server with support for loadbalancing and failover
- **Native improved XSD importer** for generating marshal able Delphi objects from XML schemas.
- **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
- **Full FastCGI hosting support.** Host PHP/Ruby/Perl/Python applications in kbmMW!
- **Native AMQP support** (Advanced Message Queuing Protocol) with AMQP 0.91 client side gateway support and sample.
- **Fully end 2 end secure brandable Remote Desktop** with near REALTIME HD video, 8 monitor support, texture detection, compression and clipboard sharing.
- **Bundled kbmMemTable Professional** which is the fastest and most feature rich in memory table for Embarcadero products.

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

COMPONENTS DEVELOPERS 4

EESB, SOA,MoM, EAI TOOLS FOR INTELLIGENT SOLUTIONS. kbmMW IS THE PREMIERE N-TIER PRODUCT FOR DELPHI / C++BUILDER BDS DEVELOPMENT FRAMEWORK FOR WIN 32 / 64, .NET AND LINUX WITH CLIENTS RESIDING ON WIN32 / 64, .NET, LINUX, UNIX MAINFRAMES, MINIS, EMBEDDED DEVICES, SMART PHONES AND TABLETS.