

BLAISE PASCAL MAGAZINE 110 / 111

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



```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
begin
  Result := True;
  if ATestIndex > Length(FoundPrimes) then
    exit;
  Result := APrimeCandidate mod FoundPrimes[ATestIndex] <> 0;
  if not Result then
    exit;
  Result := (APrimeCandidate mod FoundPrimes[ATestIndex + 1]) <> 0;
end;

procedure FindPrimes(AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 0) then
      AddPrime(i);
end;
```

Bildklassifizierer mit Laden und Testen eines vortrainierten Modells

Das Zahlen-Raten Projekt

Debuggen mit dem neuen Debugger in Lazarus - Lektionen Teil 2

KI-fähiger Gehirnschanner liest Gedanken

Lazarus kompiliert Delphi Code

BLAISE PASCAL MAGAZIN BIBLIOTHEK Per Internet und auf USB-Stick

Das Library-Kit für BPM wurde um neue Funktionen erweitert:

Suche über ALLE 111 Ausgaben und pro Ausgabe.

Raise SoftWare DropMaster

Lazarus für Visual Studio

Delphi Community Version für Delphi 11

Jim McKeeth verlässt Embarcadero/Delphi

FastReport für Lazarus unter LINUX
in einer Trial und als Professional Version



CONTRIBUTORS

Stephen Ball http://delphiaball.co.uk DelphiABall	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 mail: David @ davdata.nl	Benno Evers b.evers @ everscustomtechnology.nl	Bruno Fierens www.tmssoftware.com bruno.fierens @ tmssoftware.com	Holger Flick holger @ fixments.com
Mattias Gärtnernc- gaertnma@netcologne.de	Max Kleiner www.softwareschule.ch max @ kleiner.com	John Kuiper john_kuiper @ kpnmail.nl	Wagner R. Landgraf wagner @ tmssoftware.com
Vsevolod Leonov vsevolod.leonov@mail.ru	Andrea Magni www.andreamagni.eu andrea. magni @ gmail.com www.andreamagni.eu/wp		
		Paul Nauta PLM Solution Architect CyberNautics paul.nauta @ cybernautics.nl	
Kim Madsen www.component4developers.com kbmMW		Boian Mitov mitov @ mitov.com	
	Jeremy North jeremy.north @ gmail.com	Detlef Overbeek - Editor in Chief www.blaisepascal.eu editor @ blaisepascal.eu	
Anton Vogelaar ajv @ vogelaar-electronics.com	Danny Wind dwind @ delphicompany.nl	Jos Wegman Corrector / Analyst	Siegfried Zuhr siegfried @ zuhr.nl

Chefredakteur

Detlef D. Overbeek, Niederlande Tel.: Mobil: +31 (0)6 21.23.62.68

Nachrichten und Pressemitteilungen nur per E-Mail an editor@blaisepascal.eu

Abonnemente können online unter <https://www.blaisepascalmagazine.eu/deutsche-Ausgabe/> oder per schriftlicher Bestellung abgeschlossen werden oder indem Sie eine E-Mail an office@blaisepascal.eu senden. Das Abonnement kann zu einem beliebigen Zeitpunkt beginnen. Alle Ausgaben, die im Kalenderjahr des Abonnements veröffentlicht werden, werden geschickt. Das Abonnement hat eine Laufzeit von 365 Tagen. Abonnemente werden nicht ohne Vorankündigung verlängert

Der Zahlungseingang wird per E-Mail verschickt. Sie können das Abonnement bezahlen, indem Sie die Zahlung an folgende Adresse senden: ABN AMRO Bank Konto Nr. 44 19 60 863 oder per Kreditkarte oder Paypal Name: Stiftung Pro Pascal (Stichting Programmeertaal Pascal) IBAN: NL82 ABNA 0441960863 BIC ABNANL2A Umsatzsteuer-Nr.: 81 42 54 147 (Stichting Programmeertaal Pascal) Abonnementsabteilung Edelstenenbaan 21 / 3402 XA IJsselstein, Niederlande Mobil: + 31 (0) 6 21.23.62.68 office@blaisepascal.eu

Markenzeichen Alle verwendeten Markenzeichen sind Eigentum der jeweiligen Inhaber. Vorbehalt Obwohl wir uns bemühen sicherzustellen, dass die in der Zeitschrift veröffentlichten Informationen korrekt sind, können wir keine Verantwortung für Fehler oder Auslassungen übernehmen. Wenn Sie etwas bemerken, das möglicherweise nicht korrekt ist, wenden Sie sich bitte an den Herausgeber, und wir werden gegebenenfalls eine Korrektur veröffentlichen.



Mitglied der **Königlich Niederländischen Bibliothek**

KB

Mitglied und Spender von **WIKIPEDIA**

Subscriptions (2022 prices)

	Internat. excl. VAT	Internat. incl. 9% VAT	Shipment	TOTAL
Printed Issue (8 per year) ±60 pages :	€ 200	€ 218	€ 130	€ 348
Electronic Download Issue (8 per year) ±60 pages :	€ 64,20	€ 70		

COPYRIGHT-HINWEIS

Das gesamte in Blaise Pascal veröffentlichte Material unterliegt dem Copyright © SOPP Stichting Ondersteuning Programmeertaal Pascal, sofern nicht anders angegeben, und darf nicht ohne schriftliche Genehmigung kopiert, verbreitet oder neu veröffentlicht werden. Die Autoren erklären sich damit einverstanden, dass der zu ihren Artikeln gehörende Code nach der Veröffentlichung den Abonnenten zur Verfügung gestellt wird, indem er auf der Website der PGG zum Download angeboten wird, und dass Artikel und Code auf verteilbaren Datenträgern gespeichert werden. Die Nutzung von Programmlisten durch Abonnenten zu Forschungs- und Studienzwecken ist erlaubt, jedoch nicht zu kommerziellen Zwecken. Die kommerzielle Nutzung von Programmlistings und Code ist ohne die schriftliche Genehmigung des Autors untersagt.



von Ihrem Redakteur

Hallo,
bei der Erstellung dieser neuen Ausgabe unseres Magazins hatte ich den Sommer ganz fest im Blick.
Ich nehme an, Sie haben das gleiche Bedürfnis nach Wärme und Sonnenschein. Meine Wünsche scheinen jetzt erfüllt zu sein: Es ist warm und schön.

Ich hatte Ihnen so viel zu erzählen, dass ich eine Doppelausgabe daraus machen musste. Es gibt noch mehr Extras, aber die sind für die nächste Ausgabe 112.

Wir hatten bereits über die KI (*Künstliche Intelligenz*) und Bildklassifizierer gesprochen. Jetzt sind weitere beunruhigende Nachrichten aufgetaucht:

Ich schreibe darüber in dem Artikel KI-fähiger Gehirnschanner.

Wir müssen darüber nachdenken, was noch kommen wird, denn das ist schon ein bisschen gruselig. Lassen Sie uns darüber diskutieren, wo das alles enden soll.

Ich persönlich bin davon überzeugt, dass wir alles, was Ihnen in den Sinn kommt, auch schaffen können, wie seltsam es auch immer sein mag: Lesen Sie den Artikel auf Seite 31.

Um etwas zu schaffen, an das ich nie gedacht hätte: **Lazarus für Visual Studio.**

Dafür gibt es gute Gründe.

Lazarus ist die fortschrittlichste Umgebung, die es gibt oder geben wird. Alles, was Sie wollen, können Sie mit Lazarus machen. Sogar Sie selbst, denn die Quellen sind offen und die Industrie will dies.

Alle wichtigen Programmiersprachen haben eine direkte Verbindung zu **Visual Studio**, also brauchte Lazarus das auch.

Natürlich gibt es noch viel zu tun - aber wir sind auf dem besten Weg dahin.

Mein zukünftiges Ziel ist es, Pascal für Kinder und Jugendliche zugänglich zu machen.

Dazu habe ich einige besondere Ideen.

Ich denke, das Team und ich sollten - durch Spielen und Herumalbern - die Umgebung so freundlich für sie gestalten, dass sie lernen, ohne es zu merken, indem sie die Probleme lösen, die sie selbst schaffen...

Die KI wird dabei eine große Rolle spielen, denn sie hat bereits gezeigt, dass sie sehr nützlich sein kann.

Ich wäre ohne sie nicht in der Lage, ein Thema in andere Sprachen zu übersetzen, weil ich so viel Zeit verliere.

Martin Friebe hat die **Debugger-Story** erstellt und geschrieben und dies ist eine sehr gute Möglichkeit, um zu lernen, was er kann.

Er hat sogar noch mehr und bessere Funktionen für ihn erweitert.

Lazarus hat jetzt den Debugger, den wir so sehr gebraucht haben:

einfach in der Anwendung, aber sehr vielseitig.

- Wenn Sie eine Anfrage dazu haben, lassen Sie es mich wissen...

Ich hatte **Michael van Canneyt** gebeten, das neue **PDF Kit** für Blaise Pascal Magazine zu schreiben, so dass wir in allen Seiten und Ausgaben nach einem bestimmten Text suchen können. Das ist eine enorme Aufgabe, aber er hat es geschafft.

Es ist jetzt verfügbar und wir können Ihnen einen noch besseren Service als bisher bieten. Wir werden einige zusätzliche Beispiele erstellen...

Jim McKeeth hat **Embarcadero** verlassen, um etwas für ihn sehr Neues zu tun: Er arbeitet an der web3 Version... (ich habe bereits in der letzten Ausgabe über web 3 geschrieben)

Ich habe mit seinem Nachfolger **Ian Barker** gesprochen, und wir haben einige neue Aktionen für **Delphi** für die Zukunft besprochen.

Eine davon ist, dass wir die Neuigkeiten von **Delphi** früh genug erhalten werden, um sie zu veröffentlichen, so dass es ECHTE Neuigkeiten sein werden.

Dafür danke ich ihm im Voraus.

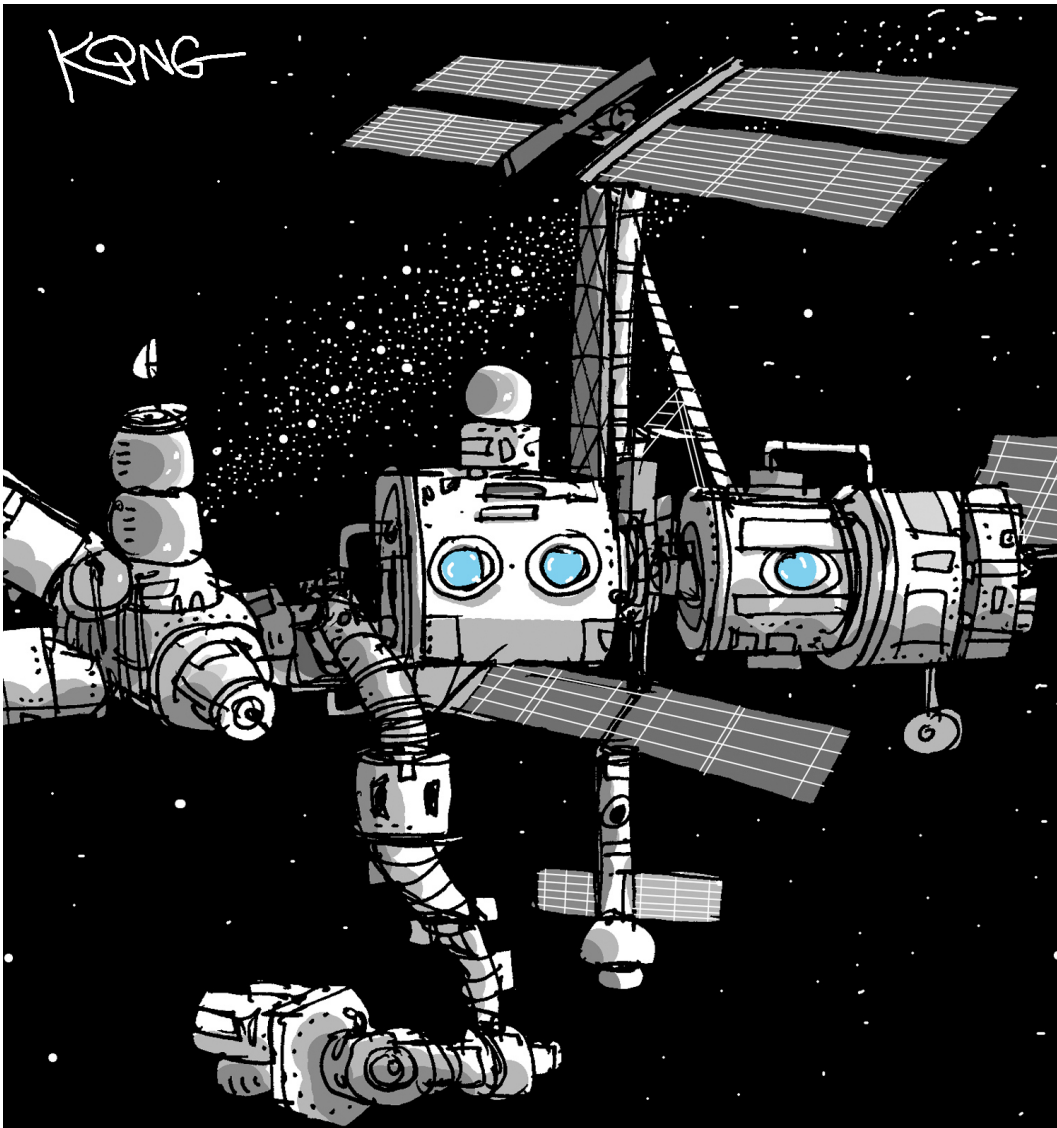
So werden wir Sie in Zukunft noch mehr mit unserer Lieblingssprache Pascal überraschen können und versuchen, junge Leute zu finden, die über und mit ihr lernen.

Ihr

Detlef



Von unserem technischen Ratgeber Jerry King



*Vati ist bei der Arbeit, Schätzchen.
Las Mami das Wifi-Passwort herausfinden*



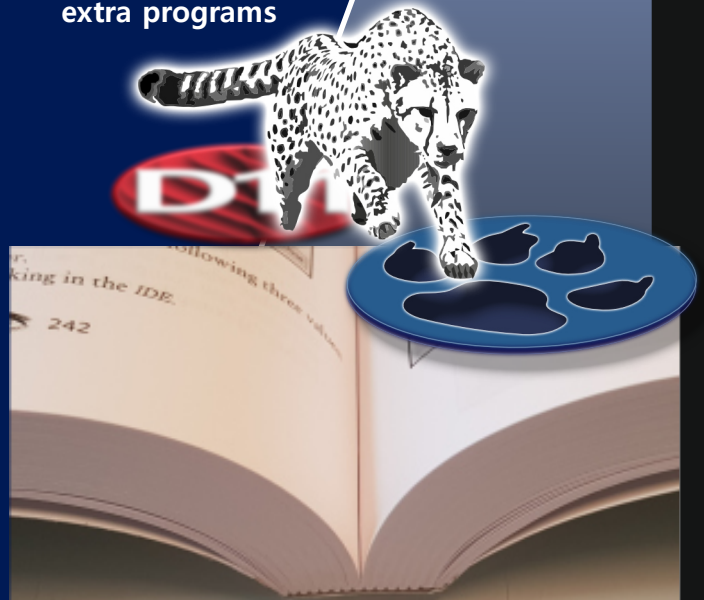
LAZARUS HANDBOOK

POCKET PACKAGE (2BOOKS)

Price: € 35,00
Excluding VAT and Shipping

934 PAGES
PDF & INDEX
INCLUDED

- English
- Printed black & white
- 2 Volumes
- PDF included
- 934 Pages
- Weight: 2kg
- Extra protected
- Including
40 Example
projects and
extra programs



BlaisePascalMagazine
PDF viewer included

CLASSIFY CIFAR10

maXbox Starter 105 - Bildklassifikator mit dem Laden und Testen eines vortrainierten Modells.

maXbox

Starter Expert



EINFÜHRUNG

Dieser Leitfaden zum maschinellen Lernen erklärt einen Klassifikator, der auf dem sogenannten CIFAR-10 Image Classifier mit einem vortrainierten Modell basiert.


Das vortrainierte Modell ist eine Datei: `ClassifyCNNModel_70.nn`

Wie der Name schon sagt, handelt es sich um ein CNN-Modell. Ein **Convolutional Neural Network** (CNN*) ist eine Art von Deep Learning-Algorithmus, der insbesondere für Bilderkennungs- und Objekterkennungsaufgaben geeignet ist. Es besteht aus mehreren Schichten, darunter **Faltungsschichten**, Pooling-Schichten und vollständig verbundene Schichten.

* CNNs werden auch als Shift Invariant oder **Space Invariant Artificial NN** bezeichnet.

Werfen wir nun einen Blick auf die untenstehende App/Skript mit einzelnen Bildern aus Cifar-Testdaten. Hierfür haben wir zwei nützliche Funktionen geschrieben. Die erste gibt das Label zurück, das mit den Vorhersagen des Modells verbunden ist. Die zweite akzeptiert ein Bild als Argument. Dann zeigt sie das Bild, die Vorhersage des Modells und die tatsächliche Klasse an, zu der das Bild gehört. Auch andere Wahrscheinlichkeiten werden im Multi-Klassifizierungsgitter angezeigt:

Form1 maXbox CAI_Classify 1.5

predicts: ship  load: .\model\ClassifyCNNModel_70.nn
dropout:

.\data\ship1.bmp

Classify

type	probability +[-60,90]
airplane	4.73834943771362
automobile	3.57388186454773
bird	-2.46890497207642
cat	-4.34820127487183
deer	-5.9130539894104
dog	-5.9897141456604
frog	-5.56393814086914
horse	-4.31041717529297
ship	15.2397747039795
truck	-0.136169910430908

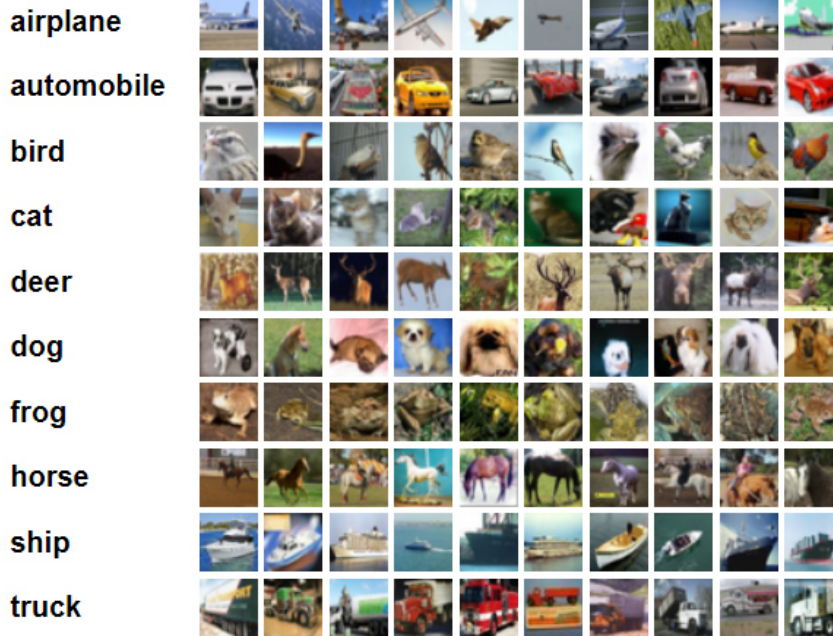
Mit dieser App können Sie Bilder von einem Flugzeug bis zu einem LKW oder einem Zug klassifizieren. Und Sie sehen, dass z.B. ein Schiff (15.2) einige Elemente eines Flugzeugs oder Autos (4.7, 3.5) in seiner Merkmalskarte hat... Die Modelle bestehen aus kleinen linearen Filtern und dem Ergebnis der Anwendung von Filtern, die als Aktivierungskarten oder allgemeiner als Merkmalskarten bezeichnet werden. Wenn Sie sich den folgenden Datensatz ansehen, werden die Merkmale in einem konstanten Punktprodukt extrahiert, auch wenn die Bilder Schatten haben oder in verschiedenen Winkeln angeordnet sind. Es ist wichtig zu wissen, dass die Filter als Merkmalsdetektoren aus dem ursprünglichen Eingabebild, in unserem Fall 32*32 Bitmaps, fungieren.

```
Const PICPATH = '\.data\';
TRAINPATH = '\.model\ClassifyCNNModel_70.nn';
```

Die richtige Art, ein CNN zu verwenden, gibt es nicht. Der Ratschlag für einen hässlichen Score lautet, eine kleinere Lernrate oder eine größere Batchgröße für die Gewichte zu verwenden, die fein abgestimmt werden, und eine höhere für die zufällig initialisierten Gewichte (z.B. die im Softmax-Klassifikator) TNetSoftMax. Vortrainierte Gewichte (in `ClassifyCNNModel_70.nn`) sind bereits gut, sie müssen nur noch feinabgestimmt und nicht verzerrt werden.



Here are the classes in the dataset, as well as 10 random images from each:



Die Lernrate ist der entscheidende Hyperparameter, der beim Training von Deep Convolution Neural Networks (DCNN) verwendet wird, um die Modellgenauigkeit zu verbessern; Auf diese Weise können Sie ein CNN-Modell erstellen, das eine Validierungsgenauigkeit von mehr als 95 % aufweist.

Die Frage ist jedoch, wie spezifisch oder relevant diese Validierung ist.

In unserem Beispiel bedeuten Werte kleiner als 0,7 falsch, während Werte größer als 0,7 richtig bedeuten.

Dies wird als monopolare Kodierung bezeichnet. CAI unterstützt auch bipolare Kodierung (-1, +1).

Werfen wir einen Blick direkt in den Quellcode für die Labels und die classify-Methode:

```
var cs10Labels: array[0..9] of string;  
  
procedure setClassifierLabels;  
begin  
  cs10Labels[0] := 'airplane';  
  cs10Labels[1] := 'automobile';  
  cs10Labels[2] := 'bird';  
  cs10Labels[3] := 'cat';  
  cs10Labels[4] := 'deer';  
  cs10Labels[5] := 'dog';  
  cs10Labels[6] := 'frog';  
  cs10Labels[7] := 'horse';  
  cs10Labels[8] := 'ship';  
  cs10Labels[9] := 'truck';  
end;
```

Der CIFAR-10-Datensatz besteht aus 60.000 32x32-Farbbildern in 10 Klassen, mit 6.000 Bildern pro Klasse. Es gibt 50.000 Trainingsbilder und 10.000 Testbilder. Der Datensatz ist in fünf Trainingsstapel und einen Teststapel mit jeweils 10.000 Bildern unterteilt.

Wir bauen nun das neuronale Faltungsnetzwerk auf, indem wir 1 Faltungsschicht, 4 Relu-Aktivierungsfunktionen, Dropout- und Pooling-Schichten, 1 vollständig verbundene Schicht und eine SoftMax-Finanzierungsfunktion verwenden. Nachfolgend finden Sie die Liste aller Schichten, für die wir auch den Optimierer und eine Verlustfunktion für den Optimierer definieren:




```
Debug TNNNet.Struct.LoadFromString ST:
-1) TNNNetInput:32;32;3;0;0;0;0;0
0) TNNNetConvolutionLinear:64;5;2;1;1;0;0;0
1) TNNNetMaxPool:4;4;0;0;0;0;0;0
2) TNNNetConvolutionReLU:64;3;1;1;1;0;0;0
3) TNNNetConvolutionReLU:64;3;1;1;1;0;0;0
4) TNNNetConvolutionReLU:64;3;1;1;1;0;0;0
5) TNNNetConvolutionReLU:64;3;1;1;1;0;0;0
6) TNNNetDropout:2;0;0;0;0;0;0;0
7) TNNNetMaxPool:2;2;0;0;0;0;0;0
8) TNNNetFullConnectLinear:10;1;1;0;0;0;0;0
9) TNNNetSoftMax:0;0;0;0;0;0;0;0
```

Die Hauptprozedur zur Klassifizierung eingehender Bilder lädt das Modell, entscheidet über Dropout oder nicht (dazu später mehr) und erstellt Input- und Output-Volumen mit der Form 32;32;3 oder ein 32x32x3 Volumen:

Begin

```
NN:= THistoricalNets.create; //TNNNet.Create();
NN.LoadFromFile(TRAINPATH);
label2.caption:= 'load: '+TRAINPATH;

if chkboxdrop.checked then
  NN.EnableDropouts(true) else
  NN.EnableDropouts(false);
pInput:= TNNNetVolume.Create0(32, 32, 3, 1);
pOutput:= TNNNetVolume.Create0(10, 1, 1, 1);

LoadPictureIntoVolume(image1.picture, pinput);
pInput.RgbImgToNeuronalInput(csEncoderRGB);
NN.Compute65(pInput,0);
NN.GetOutput(pOutput);
writeln('result get class type: '+itoa(pOutput.GetClass()));
```

Dann müssen wir `RgbImgToNeuronalInput` hinzufügen und mit Hilfe von `SoftMax` können wir die Klassenwahrscheinlichkeiten ausgeben, um sie im Stringgrid anzuzeigen. Die `*.nn`-Datei in `TRAINPATH` dient als vortrainierte Datei (`FAvgWeight`) zur Klassifizierung/Vorhersage von Bildern, auf die wir trainiert haben. Auch die CIFAR-10 Klassifizierung

Beispiele mit `experiments/testcnnalgo/testcnnalgo.lpr` und eine Reihe von CIFAR-10 Klassifikationsbeispielen sind unter `experiments` verfügbar.

Stellen Sie sich vor, die Genauigkeit steigt und die Verlustfunktion (*Fehlerrate*) sinkt.

Die Verlustfunktion ist das A und O des modernen maschinellen Lernens. Sie macht Ihren Algorithmus von der Theorie zur Praxis und verwandelt die Matrixmultiplikation in Deep Learning.



CLASSIFY CIFAR10

maXbox Starter 105 - Bildklassifikator mit dem Laden und Testen eines vortrainierten Modells.

EXPERIMENTE AUSFALLEN LASSEN

Es ist normalerweise sehr schwer, Neuron für Neuron zu verstehen, wie ein neuronales Netzwerk, das für die Bildklassifizierung eingesetzt wird, intern funktioniert. Bei dieser Technik muss ein beliebiges Neuron aktiviert werden und dann wird dieselbe Back-Propagation-Methode, die für das Lernen verwendet wird, auf ein Eingabebild angewandt, das ein Bild erzeugt, das dieses Neuron zu sehen erwartet.

Das Hinzufügen von Neuronen und neuronalen Schichten ist oft eine Möglichkeit, künstliche neuronale Netzwerke zu verbessern, wenn Sie über genügend Hardware und Rechenzeit verfügen. Wenn Sie sich Zeit, Parameter und Hardware nicht leisten können, werden Sie nach Effizienz mit separierbaren Konvolutionsverfahren (SCNN) suchen. Aber es gibt noch einen weiteren, für mich interessanten Punkt: die Dropout-Regularisierung.

Die Dropout-Schicht ist eine Maske, die den Beitrag einiger Neuronen zur nächsten Schicht aufhebt und alle anderen unangetastet lässt. In unserem Modell sehen Sie Schicht 6 als Dropout-Schicht:

6) TNNetDropout:2;0;0;0;0;0;0;0;0

Wir können eine Dropout-Schicht auf den Eingabevektor anwenden, in diesem Fall werden einige seiner Merkmale annulliert. Wir können sie aber auch auf eine versteckte Schicht anwenden, in diesem Fall werden einige versteckte Neuronen annulliert.

type	probability +[-60,90]
airplane	4.50879859924316
automobile	3.85027289390564
bird	-1.3220397233963
cat	-4.75964832305908
deer	-6.82196760177612
dog	-5.9468469619751
frog	-5.84444665908813
horse	-4.98815011978149
ship	15.6644458770752
truck	0.755887031555176

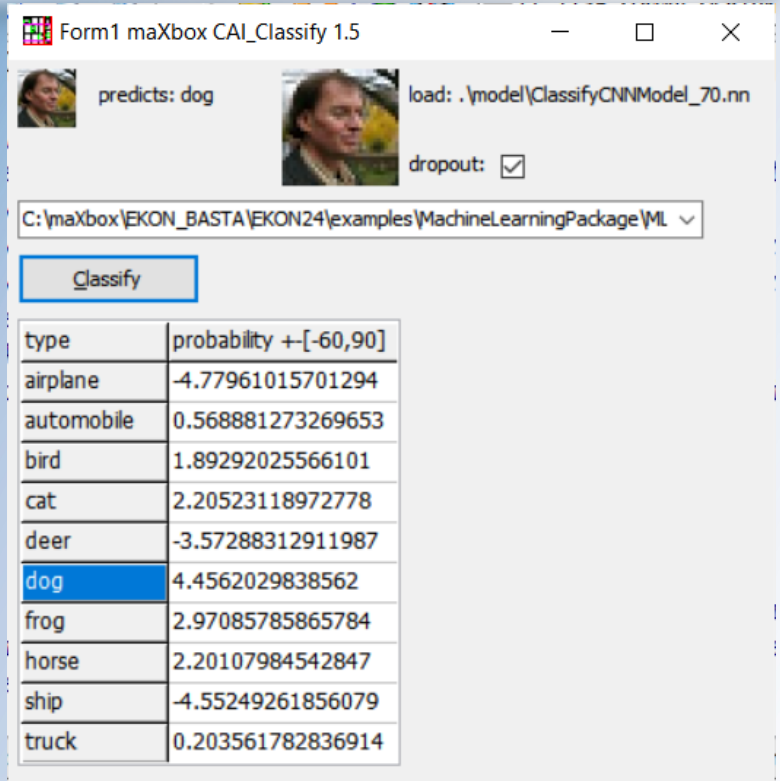
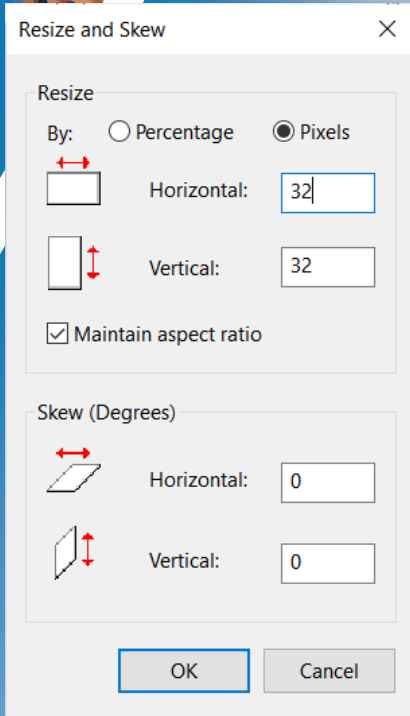
Dropout ist eine Technik, bei der zufällig ausgewählte Neuronen beim Training ignoriert werden. Sie werden nach dem Zufallsprinzip "herausgenommen". Jedes Mal, wenn Sie auf Klassifizieren klicken, erhalten Sie ein anderes Ergebnis in kleinen Änderungen. Das Schiff im ersten Bildschirm ist mit 15.2 klassifiziert, jetzt oben mit 17.1. Das bedeutet, dass ihr Beitrag zur Aktivierung der nachgelagerten Neuronen beim Vorwärtsdurchlauf zeitlich entfernt wird und dass alle Gewichtsaktualisierungen beim Rückwärtsdurchlauf nicht auf das Neuron angewendet werden. Wenn Sie ein vergleichbares Ergebnis wünschen, deaktivieren Sie das Kontrollkästchen. Was ist also der Vorteil von Dropout? Sie können sich vorstellen, dass, wenn Neuronen während des Trainings zufällig aus einem Netzwerk herausfallen, andere Neuronen einspringen und die Repräsentation ergänzen müssen, die erforderlich ist, um Vorhersagen für diese fehlenden Neuronen zu treffen.

Der Effekt ist, dass ein CNN (oder welches Deep Learning auch immer) weniger empfindlich auf bestimmte Gewichte der Neuronen reagiert. Dies wiederum führt zu einem Netzwerk, das besser generalisieren kann und weniger anfällig für die Spezialisierung von Trainingsdaten ist. Das bedeutet, dass Sie im Durchschnitt mit neuen oder im Training nicht gesehenen Bildern ein besseres Ergebnis erzielen. Nehmen wir zum Beispiel ein neues Bild aus den bekannten Klassifizierungsetiketten. Dazu konvertieren wir das Bild zunächst in eine cifar 32*32 24-Bit-Bitmap:

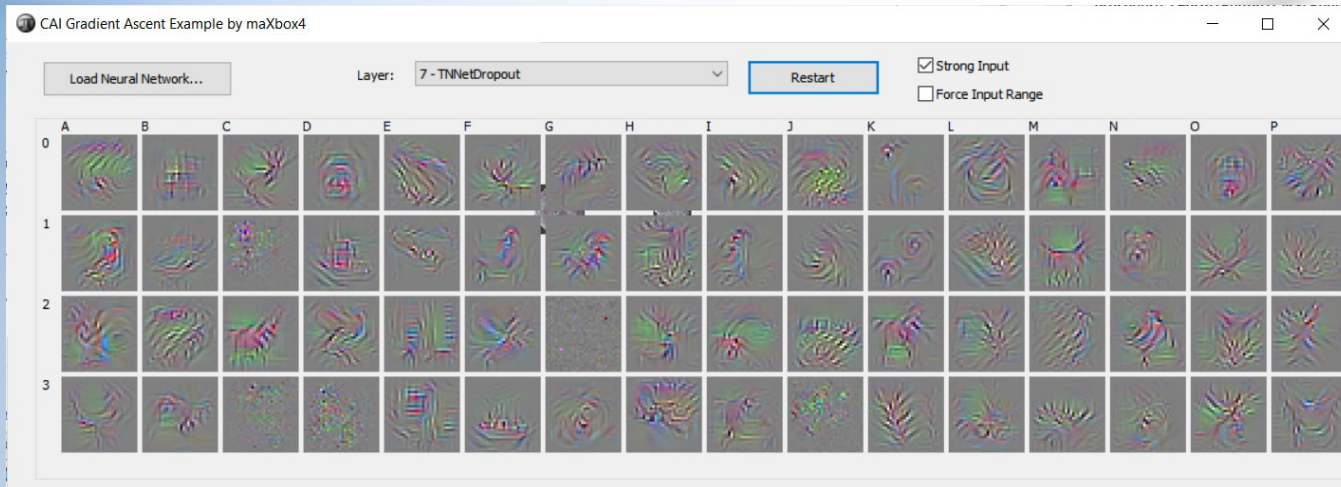




Dann laden wir das Bild als *.bmp (legen Sie es einfach in das Verzeichnis ./data) und versuchen, eine unbekannte Klasse mit ungesehenem Training zu klassifizieren, aber - und das ist irgendwie überraschend - wir erhalten ein Ergebnis:



Das Ergebnis ist niederschmetternd und verblüffend zugleich, etwas zwischen Hund und Pferd ist die Art der Bionik! Aber dies kann eine Grundlage für Ähnlichkeiten in einem Empfehlungssystem sein oder Sie können das Alter oder Geschlecht einer Person klassifizieren. einer Person, auch in einer Gender-Gap-Forschung ermöglichen. In unserem Modell wurde eine neue Dropout-Schicht zwischen **TNNetConvolutionReLU** (Aktivierungsschicht) und der versteckten Schicht **TNNetMaxPool** eingefügt. Sie können sie auch sichtbar machen, aber das ist mehr Kunst als Wissenschaft oder mehr Wissenschaft als Fiktion:





Diese visuelle Technik, die oben verwendet wurde, um zu verstehen, was einzelne Neuronen darstellen, wird Gradientenaufstieg genannt. Mehr über Gradient Ascent finden Sie unter <http://yosinski.com/deepvis>.

In dem Archiv MachineLearningPackage.zip finden Sie das Skript, das Modell und die Daten, die Sie benötigen. Es funktioniert auch mit Lazarus, Jupyter und maXbox:

The screenshot shows two windows. The left window is Lazarus IDE v2.0.8, displaying the Object Inspector for a form with components like TButton, TImage, TLabel, and TStringGrid. The right window is maXbox4 ScriptStudio, showing a Pascal procedure for classifying an image. A 'CAIForm1' window is overlaid on the code, showing a 'dog' label and a table of classification results.

type	score
airplane	-92.159187
automobile	-291.19982
bird	-499.12570
cat	417.233154
deer	-308.49899
dog	438.274780
frog	101.278960
horse	111.407750
ship	-139.78335
truck	-22.937343

Debug TNNet
picname: csdog1.bmp
picsize1 32x32
picsize2 3072
reset class type: 5





maxbox

14 1135_classify_cfar10images2tutor3.pas

Form1 maxbox CAI_Classify

predicts: dog

C:\Program Files\Streaming\maxbox4\maxbox47590\maxbox4\resized\csc

classify

type	probability +[-600,900]
airplane	-43.9228477478027
automobile	-177.024063110352
bird	-551.393493652344
cat	394.038208007813
deer	-337.506622314453
dog	424.327239990234
frog	22.5182991027832
horse	120.184234619141
ship	-43.5722160339355
truck	-82.7495727539063

```
Interface List: 1135_classify_cfar10ma
*****
procedure TForm1Button1Click(Sender: TObject);
procedure TForm1ComboBox1Change(Sender: TObject);
procedure TForm1FormCreate(Sender: TObject);
procedure setClassifierLabels;
procedure SetColumnFullWidth(Grid: TGrid);
procedure AutoSizeGridColumns(Grid: TGrid);
procedure TForm1Button1Click(Sender: TObject);
procedure TForm1ComboBox1Change(Sender: TObject);
procedure TForm1FormCreate(Sender: TObject);
procedure loadAIForm;
Locs: 257 - code blocks: 10

Compiled: 09/0 Runtime: 0:0:1.61 Threads: 5
```

2°C 16:08 09/02/2023



CLASSIFY CIFAR10

maxbox Starter 105 - Bildklassifikator mit dem Laden und Testen eines vortrainierten Modells.

maxbox

Die Funktion `FormCreate()` kann auch mit diesen wenigen Codezeilen ausgelöst werden:

```

procedure TForm1FormCreate(Sender: TObject);
var k,t: integer;
    items: TStringList;
begin
    items:= TStringList.create;
    for k:= 0 to 9 do
        StringGrid1.Cells[0, k+1]:= cs10Labels[k];

    //FindAllFiles(ComboBox1.Items, 'csdata');
    FindFiles(exepath+'data', '*.bmp',items);
    writeln(items.text);
    for t:= 1 to items.count-1 do
        ComboBox1.Items.add(items[t]);
    if ComboBox1.Items.Count > 0 then begin
        ComboBox1.text:= ComboBox1.Items[0];
        if FileExists(ComboBox1.text) then begin
            Image1.Picture.LoadFromFile(ComboBox1.text);
            Image2.Picture.LoadFromFile(ComboBox1.text);
            label1.Caption:= extractfilename(ComboBox1.text);
        end;
    end;
end;

```

`FindFiles(exepath+'data', '*.bmp', items)` ist eine Übernahme aus Lazarus.

Für den Fall, dass ein Eingabebild nicht 32x32 ist, können Sie die Größe ändern (durch Kopieren):

`TVolume.CopyResizing(Original: TVolume; NewSizeX, NewSizeY: integer);`

Und da Sie ein trainiertes NN haben, können Sie dies aufrufen:

`TNeuralImageFit.ClassifyImage(pNN: TNNNet; pImgInput, pOutput: TNNNetVolume);`

The screenshot shows the maxbox4 ScriptStudio IDE with the following code in the main editor:

```

24 TForm1 = {class(TForm;
25 var
26     Button1: TButton;
27     ComboBox1: TComboBox;
28     chkbxdrop: TCheckbox;
29     Image1, image2: TImage;
30     Label1, label2, lbldropout: TLabel;
31     StringGrid1: TStringGrid;
32     procedure TForm1Button1Click(Sender: TObject);
33     procedure TForm1ComboBox1Change(Sender: TObject);
34     procedure TForm1FormCreate(Sender: TObject);
35     //private
36     //public
37
38     const PICPATH = './data\';
39     TRAINPATH = './model\ClassifyCNNModel_70.nn\';
40
41 var
42     Form1: TForm1;

```

The interface list on the right shows the following procedures:

```

Interface List: 1135_classify_cifar10images1_5.pa
*****
procedure TForm1Button1Click(Sender: TObject);
procedure TForm1ComboBox1Change(Sender: TObject);
procedure TForm1FormCreate(Sender: TObject);
procedure setClassifierLabels;
procedure SetColumnFullWidth(Grid: TStringGrid;
procedure AutoSizeGridColumns(Grid: TStringGrid);
procedure TForm1Button1Click(Sender: TObject);
procedure TForm1ComboBox1Change(Sender: TObject);
procedure TForm1FormCreate(Sender: TObject);
procedure loadAllForm;
Locs: 298 - code blocks: 10

```

The debug console at the bottom shows the loaded neural network structure:

```

Debug TNNNet.Struct.LoadFromStrng ST:
-1)TNNNetInput:32;32;3;0;0;0;0;0
0)TNNNetConvolutionLinear:64;5;2;1;1;0;0;0
1)TNNNetMaxPool:4;4;0;0;0;0;0;0
2)TNNNetConvolutionReLU:64;3;1;1;0;0;0;0
3)TNNNetConvolutionReLU:64;3;1;1;0;0;0;0
4)TNNNetConvolutionReLU:64;3;1;1;0;0;0;0
5)TNNNetConvolutionReLU:64;3;1;1;0;0;0;0
6)TNNNetDropout:2;0;0;0;0;0;0;0

```



Fazit:

Die Neural-API oder CAI API (Conscious Artificial Intelligence) ist so etwas wie TensorFlow für Pascal und ist eine plattformunabhängige Open-Source-Bibliothek für künstliche Intelligenz oder maschinelles Lernen in den Bereichen Spracherkennung, Bildklassifizierung, OpenCL, Big Data, Data Science und Computer Vision2.

Um dieses Beispiel ausführen zu können, müssen Sie eine bereits trainierte neuronale Netzwerkdatei laden und dann das Bild auswählen, das Sie klassifizieren möchten. CAI speichert sowohl die Architektur als auch die Gewichte in der gleichen *.nn-Datei! Dropout ist eine einfache und leistungsstarke Regularisierungstechnik für neuronale Netzwerke und Deep Learning-Modelle.

Die Verlustfunktionen unterscheiden sich je nach Problemstellung, auf die Deep Learning angewendet wird. Die Kostenfunktion ist ein weiterer Begriff, der austauschbar für die Verlustfunktion verwendet wird, aber er hat eine andere Bedeutung. Eine Verlustfunktion bezieht sich auf ein einzelnes Trainingsbeispiel, während eine Kostenfunktion ein durchschnittlicher Verlust über den gesamten Trainingsdatensatz ist.

<https://github.com/joaopauloschuler/neural-api>

<https://sourceforge.net/projects/cai/files/>

<https://github.com/maxkleiner/neural-api>

REFERENZ:

Wie ein Jupyter Notebook:

https://github.com/maxkleiner/maxbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb

und das gleiche in colab.research:

https://colab.research.google.com/github/maxkleiner/maxbox/blob/master/EKON24_SimpleImageClassificationCPU.ipynb

Das ganze Paket mit App, Skript, Tutorial, Daten und Modell:

<https://github.com/maxkleiner/neural-api/blob/master/examples/SimpleImageClassifier/MachineLearningPackage.zip>

Dokument und Werkzeug: <https://maxbox4.wordpress.com>

Script Ref: 1135_classify_cifar10images1_5.pas



THE NEW FUTURE BLAISE PASCAL LIBRARY 2023

ON USB STICK INCLUDING THE INDEXER FOR ALL ITEMS AND PER ITEM ON CREDIT CARD USB STICK

Blaise Magazine Library

File | D:/SPP/Blaise/Blaise_UK_110_111_2023/Authors/Michael%20Van%20Canney/indexedbpm/app/index.html

Other bookmarks

Issue: Enter issue number... Open

Search: Search in all articles... Search

Search in PDF

Dark mode

Login

NO ISSUE SELECTED

BLAISE PASCAL MAGAZINE

Search in results

Page 1: ickThe Lib rary kit for BPM has been exte

Page 2: PDF Viewer 2023 Blaise Pascal Lib rary USB stick PageLazarus Han

Page 3: hor.Member of the Royal Dutch Lib rary KONINKLIJKE BIBLIOTHEEKCON

Page 14: tform-independent open source Lib rary for artificialintelligenc

Page 15: TISEMENTTHE NEW BLAISE PASCAL Lib RARYON USB STICK INCLUDINGTHE

Page 16: STICK INCLUDINGBLAISE PASCAL Lib RARYADVERTISEMENT

Page 17: STICK INCLUDINGBLAISE PASCAL Lib RARYADVERTISEMENT

Page 21: eth leaving Embarcadero/Delphi Lib rary kitThe Library kit for BP

Page 48: eth leaving Embarcadero/Delphi Lib rary kitThe Library kit for BP

Page 49: Magazine offers subscribers a Lib rary:a collection of all issu

Page 49: nce theBlaise Pascal Magazine Lib rary.◆INTRODUCTIONIn several

Page 49: ite the Blaise PascalMagazine Lib rary as a Pas2js application t

Page 49: of the Blaise Pascal Magazine Lib rary willneed to have the foll

Page 49: GE 1/20BLAISE PASCAL MAGAZINE Lib RARYBY INTERNET AND ON USB STI

Page 50: GE 2/18BLAISE PASCAL MAGAZINE Lib RARYBY INTERNET AND ON USB STI

BLAISE PASCAL MAGAZINE 110 / 111

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

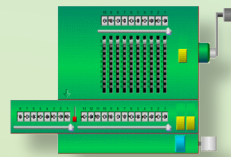
Blaise Pascal

```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): Boolean;
begin
  Result := True;
  if ATestIndex < 2 then
    if (FoundPrimes) then
      exit;
  Result := APremeCandidate mod ATestIndex <= 0;
  if not Result then
    exit;
  Result := (APremeCandidate mod ATestIndex < 1);
end;

var
  AMaxNum: integer;
  ATestIndex: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 0*1) then
      AddPrime(i);
  end;
```

AI-enabled brain scanner reads thoughts
Image Classifier with loading and testing a pre-trained model
Delphi Community version for Delphi 11
Jim McKeeth leaving Embarcadero/Delphi
The Number guessing project
BLAISE PASCAL MAGAZINE LIBRARY By internet and on USB Stick
The Library kit for BPM has been extended with new features:
Search over ALL 111 issues and per issue.
Lazarus compiling Delphi code
Lazarus for Visual Studio
Debugging with the new debugger in Lazarus - lessons part 2
FastReport for Lazarus on LINUX
in a Trial and as Professional version

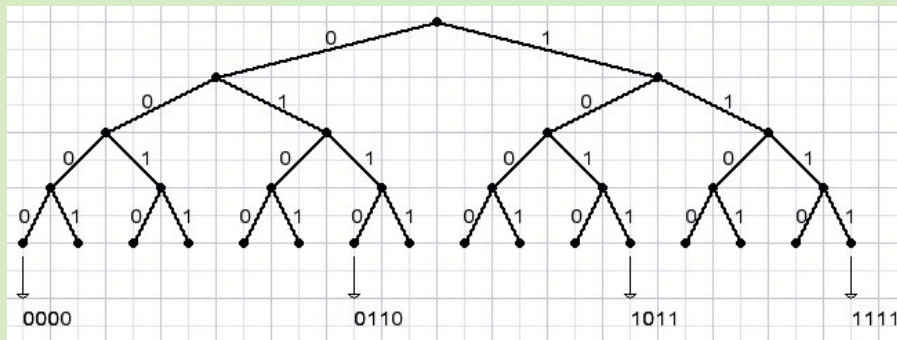
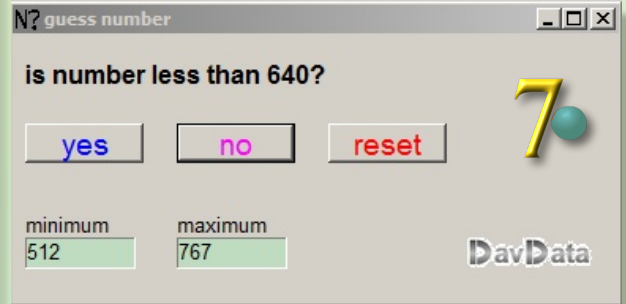
GUESS: NUMBER GUESSING PROJECT.



EINFÜHRUNG

Computer sind informationsverarbeitende Maschinen. Der Empfang von Informationen ist die Antwort auf eine Frage. Die kleinste Menge an Information ist also die Antwort Ja oder Nein. Wenn wir "Nein" = 0 und "Ja" = 1 nennen, haben wir eine binäre Ziffer, BIT genannt. Ein Bit ist die Einheit der Information.

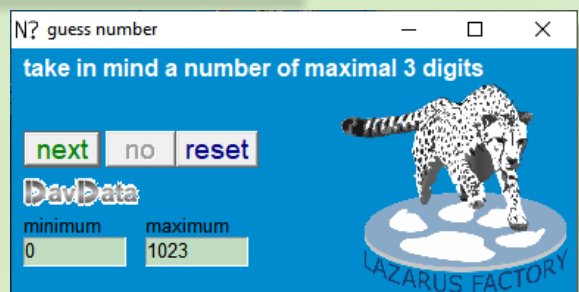
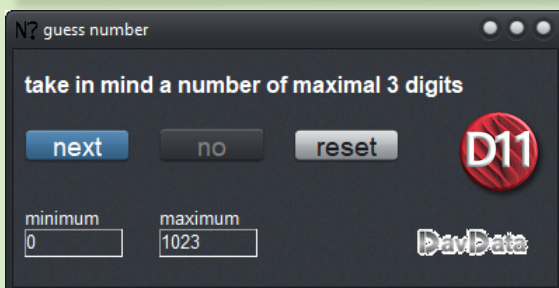
Wenn wir n Bits an Informationen erhalten, eines pro Wegspaltung, können wir den Weg zu 2^n verschiedenen Zielen finden.

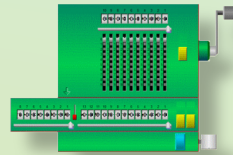


Jedes Informationsbit reduziert die Anzahl der Ziele um die Hälfte. Im Allgemeinen gilt: Wenn die Auswahl aus N_1 Zielen vor und N_2 Zielen nach Erhalt der Informationen besteht, ist die Anzahl der Informationsbits $I = \log_2(N_1/N_2)$. Von der Spitze des Baums aus können wir 16 Ziele erreichen. Unten ist diese Zahl 1. Wir haben also $\log_2(16/1) = 4$ Bits Informationen erhalten.

DAS PROGRAMM

Um zu zeigen, wie Informationen die Auswahlmöglichkeiten einschränken, wurde ein kleines Programm geschrieben. Der Benutzer wird aufgefordert, sich eine Zahl kleiner als 1000 (eigentlich kleiner als 1024) vorzustellen. Der Computer stellt dann Fragen, um die Zahl zu ermitteln. Bei jeder Ja/Nein-Antwort des Benutzers wird die Auswahl an Zahlen halbiert. Es sind 10 Fragen nötig, denn $2^{10} = 1024$.





ALGORITHM

Zu Beginn ist die kleinstmögliche Zahl 0, die größte Zahl ist 1023.

Basis=0, Bereich=1024-0=1024.

Bei jeder Antwort halbiert sich der Bereich.

Die Frage lautet $N \geq \text{base} + \text{range}/2$.

Wenn wahr, ist $\text{base} = \text{base} + \text{range}/2$.

Dies wiederholt sich, bis $\text{range}=1$.

Dann ist die Basis die Zahl.

Anschaulicheres Spiel

Immer die gleiche Art von Fragen zu stellen ist langweilig.

Also werden die Fragen zufällig ausgewählt aus

$N \geq \text{Bereich} + \text{Basis}/2$ und

$N < \text{Bereich} + \text{Basis}/2$

Beschreibung des Programms

Entscheidungen werden in der Prozedur `gamecontrol` getroffen

Das Spiel wird durch Nachrichten an `gamecontrol` gesteuert.

```
type TGameStatus = (gsStart,gsQuestion,gsEnd);
   TControlMessage = (cmStart,cmYes,cmNo,cmNextQuestion,cmEnd);

var gameState : TGameStatus;
    base,range : word;
    QGR : boolean;
.....
.....

procedure gamecontrol(cm : TControlmessage);
```

`gamecontrol` ruft die Prozeduren `procStart`, `procQuestion`(var Q: boolean);

for action auf.

Q gibt die Art der Frage zurück, die gestellt wurde.

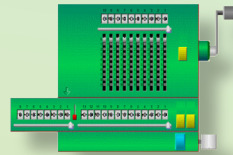
`cmYes` und `cmNo` sind Meldungen von Klicks auf die Buttons "Ja" und "Nein".

Einzelheiten entnehmen Sie bitte dem Quellcode.

Delphi 7/11 und Lazarus Code ist verfügbar

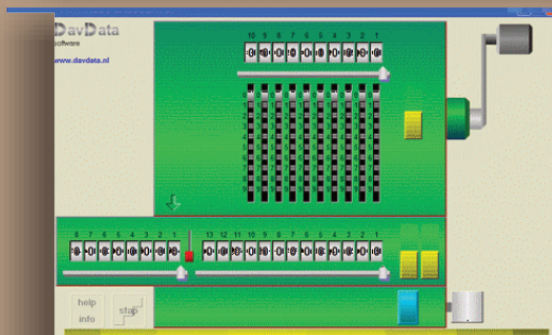
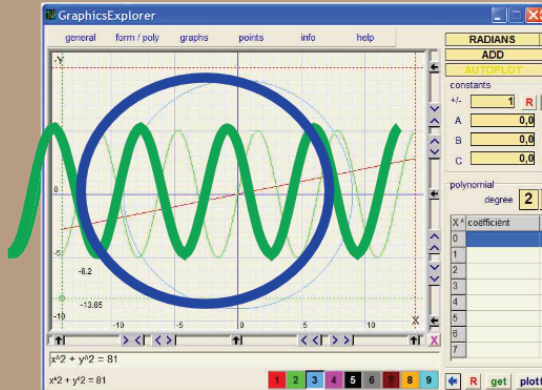


ADVERTISEMENT



DAVID DIRKSE

including 50 example projects



COMPUTER (GRAPHICS) MATH & GAMES IN PASCAL

<https://www.blaisepascalmagazine.eu/product-category/books/>



LAZARUS HANDBOOK (PDF) + SUBSCRIPTION 1 YEAR

- **Lazarus Handbook**
- Printed in black and white
- PDF Index for keywords
- Almost 1000 Pages
- Including 40 Examples
- **Blaise Pascal Magazine**
- English and German
- Free Lazarus PDF Kit Indexer
- 8 Issues per year
- minimal 60 pages
- Including example projects and code

SPECIAL OFFER € 75



+

BLAISE PASCAL MAGAZINE 110
Multi platform / Object Pascal / Internet / JavaScript / Web Assembly / Pac2s /
 Databases / CSS Styles / Progressive Web Apps
 Android / iOS / Mac / Windows & Linux



Blaise Pascal

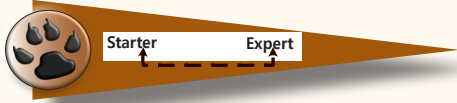


```

function CheckPrime(APrimeCandidate, ATestIndex: Integer): Boolean;
begin
  Result := True;
  if ATestIndex <= 1 then
    exit;
  Result := APrimeCandidate mod ATestIndex < 0;
  if not Result then
    exit;
  Result := !APrimeCandidate div ATestIndex < 1;
end;

procedure AddPrime(Maximum: Integer);
var
  i: Integer;
begin
  for i := 2 to Maximum do
    if CheckPrime(i, 0) then
      AddPrime(i);
end;
          
```

Image Classifier with loading and testing a pre-trained model
 Delphi Community version for Delphi 11
 Jim McKeeth leaving Embarcadero/Delphi
 Library kit
 The Library kit for BPM has been extended with new features: Search over ALL 109
 issues and per issue.
 Lazarus compiling Delphi code
 Lazarus for Visual Studio
 Debugging with the new debugger in Lazarus - lessons part 2



TEIL 2: NÄCHSTE SCHRITTE - SCHRITTWEISE

VERFOLGEN DES CODES

Im vorherigen Artikel haben wir Breakpoints verwendet, um bei bestimmten Zeilen zu pausieren. Das gibt einen ziemlich eingeschränkten Blick auf nur ein paar Schnappschüsse. Der Debugger bietet Funktionen wie das Ausführen einer einzelnen Zeile und das automatische Anhalten bei der nächsten Zeile, ohne dass dort ein Breakpoint erforderlich ist. Dies wird als Stepping bezeichnet. In diesem Artikel werden wir verschiedene Methoden des Steppens untersuchen.

Wenn wir den Beispielcode ausführen:

```

1. program primes;
2.
3. const
4.   MAX_NUM = 100;
5.
6. var FoundPrimes: Array of integer;
7.
8. procedure AddPrime(APrimeNum: Integer);
9. var
10.  l: SizeInt;
11. begin
12.  l := Length(FoundPrimes);
13.  SetLength(FoundPrimes, l+1);
14.  FoundPrimes[l] := APrimeNum;
15. end;
16.
17. function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
18. begin
19.  Result := True;
20.  if ATestIndex >= Length(FoundPrimes) then
21.    exit;
22.
23.  Result := APrimeCandidate mod FoundPrimes[ATestIndex] <> 0;
24.  if not Result then
25.    exit;
26.
27.  Result := CheckPrime(APrimeCandidate, ATestIndex + 1);
28. end;
29.
30. procedure FindPrimes(AMaxNum: integer);
31. var
32.  i: Integer;
33. begin
34.  for i := 2 to AMaxNum do
35.    if CheckPrime(i, 1) then
36.      AddPrime(i);
37. end;
38.
39. procedure PrintPrimes;
40. var
41.  i: Integer;
42. begin
43.  for i := 0 to Length(FoundPrimes) - 1 do
44.    WriteLn(FoundPrimes[i]);
45. end;
46.
47. begin
48.  FindPrimes(MAX_NUM);
49.  PrintPrimes;
50.  readln;
51. end.

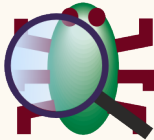
```



DER LAZARUS DEBUGGER

TEIL 2: NÄCHSTE SCHRITTE

- SCHRITTWEISE



SEITE 2/8



Wir würden die Ausgabe zu Beginn erwarten:

2
3
5
7
11
13

Aber stattdessen beginnt es mit einer Reihe, die die Zahl "4" enthält.

2
3
4
5
7
11
13

STEP OVER

Im letzten Artikel haben wir einen Breakpoint verwendet und uns die Daten angesehen, um vorherzusagen, was der Code tun würde. Um die Debug-Sitzung zu starten, setzen wir einen Breakpoint in Zeile 34 und starten mit F9. (Abbildung 1)

```
30 procedure FindPrimes(AMaxNum: integer);
.   var
.     i: Integer;
.   begin
.     for i := 2 to AMaxNum do
34       if CheckPrime(i, 1) then
35         AddPrime(i);
.     end;
```

Abbildung 1

Sobald der Debugger die Zeile erreicht hat, wird die Anwendung angehalten. Anstatt das Programm zu starten, um den BreakPoint erneut zu erreichen, führen wir es Zeile für Zeile aus.

Wenn wir F8 drücken, wird der Debugger "übergehen". Er führt die aktuelle Zeile aus und hält in der nächsten Zeile an. Der Pfeil, der im roten Punkt des Breakpoints angezeigt wurde, befindet sich nun vor der nächsten Zeile. (Abbildung 2).

```
.   begin
.   for i := 2 to AMaxNum do
.   if CheckPrime(i, 1) then
→35   AddPrime(i);
.   end;
```

Abbildung 2

Da der Debugger nun den Befehl "for i := 2 to AMaxNum" ausgeführt hat, ist die Variable "i" nun initialisiert, und wir können ihren Wert im Fenster Locals beobachten. (Strg-Alt-L).

Die durch den Pfeil als aktuell markierte Zeile 35 ist noch nicht ausgeführt worden. Wir werden diese Zeile ausführen, indem wir erneut F8 drücken. Der Debugger führt die gesamte Zeile aus, d.h. er führt den Aufruf von "CheckPrime" in einem einzigen Schritt aus. Er wird nicht innerhalb von "CheckPrime" aufhören.

Da "i" den Wert 2 hat und dies eine Primzahl ist, sollte die nächste Zeile die bedingte Anweisung "then" sein. Wir können sehen, dass dies tatsächlich der Fall ist, denn der grüne Pfeil befindet sich jetzt vor Zeile 36. Der nächste "Schritt über" (F8) führt "AddPrime" aus und bringt uns zurück an den Anfang der Schleife. "i" ist immer noch 2 und wird auf 3 erhöht, wenn wir die Zeile "for i := " ausführen.



DER LAZARUS DEBUGGER

TEIL 2: NÄCHSTE SCHRITTE

- SCHRITTWEISE



SEITE 3/8



STEP INTO (EINSTEIGEN)

Wir können die Schleife erneut durchlaufen (mit F8), und bei der 3. Iteration der Schleife wird "i" zu 4. (Abbildung 3)

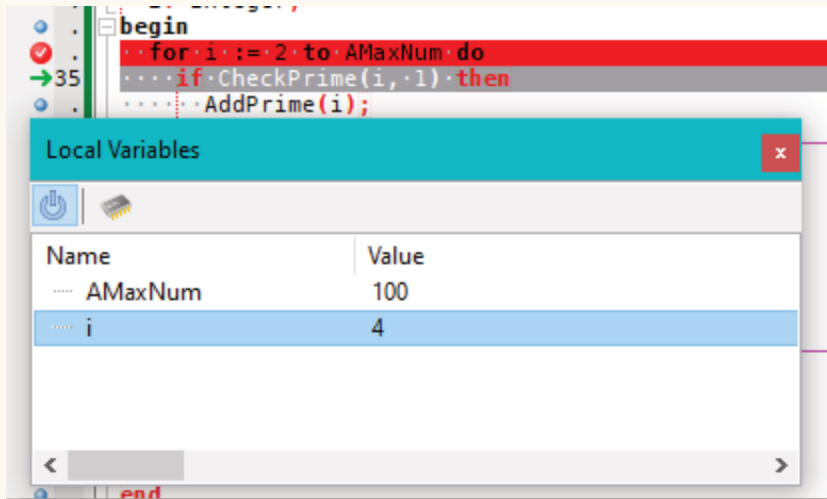


Abbildung 3

Wie wir in der Ausgabe gesehen haben, wird die Zahl 4 fälschlicherweise zur Liste der Primzahlen hinzugefügt. Das bedeutet, dass wir überprüfen müssen, was in "CheckPrime" passiert.

Anstatt einen BreakPoint zu setzen (wie im letzten Artikel), verwenden wir "step into" (F7), um die Funktion aufzurufen

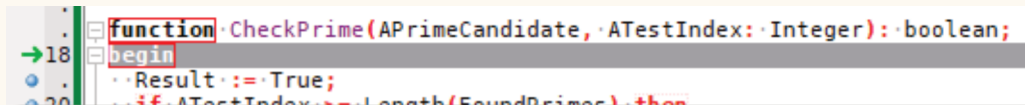
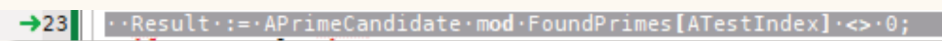


Abbildung 4

Der Debugger springt zur ersten Zeile in "CheckPrime" (Abbildung 4). Von dort aus können wir mit "step over" (F8) Zeile für Zeile weitergehen. Die Funktion "CheckPrime" prüft rekursiv, ob eine der bereits gefundenen Primzahlen ein natürlicher Teiler des aktuellen APrimeCandidate ist. Wenn der Kandidat nicht durch eine dieser Primzahlen teilbar ist, dann ist er selbst eine Primzahl.

Wenn wir "step over" (F8) verwenden, gehen wir über die Zeilen, die prüfen, ob "ATestIndex" innerhalb der Länge des Arrays FoundPrimes liegt. Wir erreichen die Zeile



An dieser Stelle sollten wir eine Überwachung für "FoundPrimes[ATestIndex]" hinzufügen, damit wir sehen können, was der Code testet. (Abbildung 5)

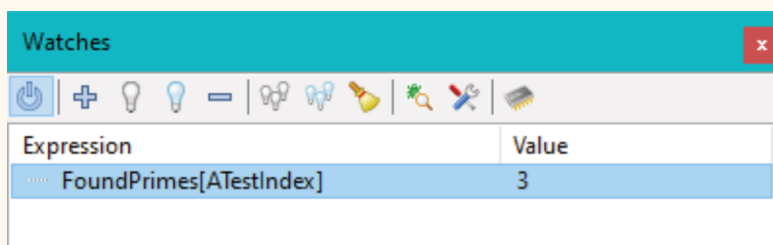


Abbildung 5





Wenn Sie über die Zeile 23 gehen, wird der Wert für "Result" berechnet und im "Lokalen Fenster" angezeigt. 3 ist kein natürlicher Teiler von 4.

Und Result ist wahr, da kein natürlicher Teiler gefunden wurde. Der bisherige Code nimmt an, dass 4 eine Primzahl sein könnte.

Wir verwenden einen weiteren "step over", um über das "if not result" zu der Codezeile mit dem rekursiven Aufruf von "CheckPrime" zu gelangen. Hier verwenden wir "step into" (F7), um den verschachtelten Funktionsaufruf aufzurufen und zu sehen, wie er weiter prüft, ob 4 eine Primzahl ist oder nicht.

Der Debugger bringt uns wieder zu Zeile 18, dem "Begin" von "CheckPrime".

Wir verwenden "step over", um die Ausführung des Codes zu verfolgen.

Dieses Mal gehen wir über die "if ATestIndex >= Length(FoundPrimes)" bringt uns zur Zeile "end" der Funktion. Das bedeutet, dass das "then exit" ausgeführt wurde. Aber aufgrund von Code-Optimierungen durch den Compiler wurde die Zeile "exit" übersprungen.

HINWEIS: Wenn wir das Projekt mit "Optimierungsstufe 0 - keine Optimierungen" statt mit der Standardeinstellung "Stufe 1" kompiliert hätten kompiliert hätten, dann hätte das Stepping die Zeile 21 "exit" enthalten.

Der Code gibt also zurück, dass alle Primzahlen als Teiler geprüft wurden und keine den aktuellen Kandidaten "4" geteilt hat. Wir haben aber nur gesehen, dass er mit 3 als Divisor getestet wurde. Wir wissen jedoch, dass auch 2 als Primzahl gefunden wurde. Wir müssen also unseren Code daraufhin untersuchen, warum er 2 übersprungen hat.

In der Schleife in "FindPrimes" rufen wir "if CheckPrime(i, 1) then" auf. Die 1 als 2. Parameter sollte die Primzahlprüfung mit der ersten gefundenen Primzahl beginnen. Wir verwenden jedoch den Parameter "ATestIndex" als Index für das Array "FindPrimes" und dynamische Array-Indizes sind nullbasiert. Diese Zeile sollte also lauten

```
35. if CheckPrime(i, 0) then
```

Wenn wir das Projekt neu kompilieren und es ausführen (nachdem wir den BreakPoint entfernt haben), erhalten wir die korrekte Ausgabe. Wir haben das Projekt also erfolgreich mit "step over" und "step into" debuggt.

STEP OUT

Wir werden uns nun einige andere Methoden des Steppens ansehen.

Wir werden keine weiteren Fehler finden, aber wir werden denselben Beispielcode verwenden (mit der oben beschriebenen Korrektur)

Wir haben "step in" verwendet, um ein Unterprogramm aufzurufen.

Jetzt machen wir es umgekehrt. Setzen wir einen BreakPoint in Zeile 24

"if not Result then" (nach "Result := APrimeCandidate mod ..") und führen das Projekt aus.

Wenn wir in dieser Zeile pausieren, können wir die Locals der Funktion überprüfen.

Wenn wir mehrmals auf "run" (F9) drücken, sehen wir die Locals

"ATestIndex"=1 und "APrimeCandidate"=9.

Jetzt, da wir diese Werte haben, möchten wir wissen, was der Aufrufer der Funktion mit dem Ergebnis tun wird.

Es ist möglich, mehrmals F8 zu drücken, bis wir zur "end;"-Zeile kommen,

und dann würde uns F8 zum Code des Aufrufers führen. Wir können aber auch einen einfacheren Weg wählen. Wir können Shift-F8 ("step out") drücken.

Dies bewirkt dasselbe, es wird der Rest der aktuellen Funktion ausgeführt (einschließlich des Codes, der von der aktuellen Funktion aufgerufen wird), und dann gestoppt, sobald die Ausführung zum Aufrufer zurückkehrt.





Nachdem wir "step out" aufgerufen haben, fährt der Debugger mit der Zeile 27 fort. Dies ist jedoch die Zeile 27 im aufrufenden Code. Sehen Sie die grünen Pfeile in Abbildung 6 hat der Debugger den verschachtelten Aufruf von "CheckPrime" in Zeile 27 überflogen, das "Ende" in Zeile 28 erreicht, ist zurückgekehrt und hat in Zeile 27 pausiert, die die Funktion aufgerufen hatte.

Wir können auch sehen, dass wir uns im aufrufenden Code befinden, da der Wert von "ATestIndex" von 1 auf 0 geändert wurde.

Hinweis: Je nach aufrufendem Code kann "step out" zu der Zeile zurückkehren, die den Aufruf enthält, in diesem Fall Zeile 27, die "CheckPrime(APrimeCandidate, ATestIndex + 1)" enthält. Oder es kann zu der Zeile nach dem Aufruf zurückkehren, was das "Ende" in Zeile 28 wäre. Beides ist ein gültiges Ergebnis.

Da wir zu Zeile 27 zurückgekehrt sind, ist diese Zeile noch nicht vollständig ausgeführt worden. Die Variable "Ergebnis" des Aufrufers ist möglicherweise noch nicht gesetzt. Aber sobald wir in die nächste Zeile gehen (mit F8), wird sie auf das gesetzt, was die Funktion zurückgegeben hat.

```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): boolean;
begin
  Result := True;
  20  if ATestIndex >= Length(FoundPrimes) then
  21  exit;
  22
  Result := APrimeCandidate mod FoundPrimes[ATestIndex] <> 0;
  24  if not Result then
  25  exit;
  26
  Result := CheckPrime(APrimeCandidate, ATestIndex + 1);
end;

30 procedure FindPrimes(AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
  35  if CheckPrime(i, 0 * 1) then
  36  AddPrime(i);
end;
```

Abbildung 6

Wir können nun im Code des Aufrufers weiter debuggen. Wir können auch wieder "aussteigen", was uns zur Zeile 35 in "FindPrimes" führen würde. Dies wird durch die rote Linie in Abbildung 6 angezeigt.

BREAKPOINTS VERSUS STEPPING

Es gibt einige Unterschiede zwischen dem Erreichen eines Breakpoints und dem Springen zur nächsten Zeile, in oder aus einer Funktion.

Bei rekursivem Code hält ein BreakPoint Ihren Code immer an, wenn Sie diese Zeile erreicht haben.

```
1. procedure foo;
2. begin
3.   if recurse_done then exit;
4.   foo();
5.   writeln;
6. end;
```





Wenn Sie sich in der Zeile 4 "foo()" befinden und zu Zeile 5 "writeln" gehen möchten, können Sie "step over" verwenden. Dies führt den gesamten verschachtelten (und unterverschachtelten) Aufruf von "foo" aus und hält bei Zeile 5 im aktuellen Aufruf von "foo" an.

Wenn Sie jedoch in Zeile 5 einen BreakPoint "writeln" setzen und mit F9 zum BreakPoint laufen, halten Sie innerhalb des (unter)verschachtelten Aufrufs von "foo" an. Es handelt sich um dieselbe Codezeile, aber in einem ganz anderen Kontext. Das Gleiche gilt für "step out".

Wenn Sie sich innerhalb eines verschachtelten Aufrufs von "foo" und in Zeile 4 befinden, führt step out alle weiteren verschachtelten Aufrufe von foo aus, es läuft bis zum Ende von foo auf der aktuellen Ebene und kehrt dann zum Aufrufer zurück. Wenn Sie einen Haltepunkt in Zeile 5 setzen (die Zeile im Aufrufer, in der Sie nach der Rückkehr vom aktuellen Aufruf von "foo" eine Pause einlegen müssten), können Sie dies nicht tun. Der BreakPoint wird innerhalb des verschachtelten Aufrufs von "foo" ausgelöst, oder wenn dieser mit "recurse_done"=true beendet wird, wird der BreakPoint in der nächsten Zeile ausgelöst, bevor Sie die Anweisung "end" erreichen und von dort aus zurückkehren.

BREAKPOINTS STOP STEPPING

In allen Beispielen haben wir entweder run (F9) zum BreakPoint oder stepping verwendet. Es sollte erwähnt werden, dass Breakpoints auch Vorrang vor Stepping haben.

Wenn Sie über eine Prozedur schreiten ("step over") und innerhalb dieser Prozedur ein BreakPoint vorhanden ist, wird der BreakPoint getroffen. Das bedeutet, dass der Schritt NICHT bis zu der Zeile ausgeführt wird, die beim Aufrufen des Schritts die nächste war.

Wenn dies passiert, können Sie eine Reihe von "step out" verwenden, um zu dem Code zurückzukehren, mit dem Sie begonnen haben.

In ähnlicher Weise kann ein "step out" durch einen BreakPoint unterbrochen werden.

Entweder, wenn sich der BreakPoint in einer Unterroutine befindet, die beim Verlassen der aktuellen Routine aufgerufen wird, oder wenn sich der BreakPoint in der aktuellen Routine befindet, bevor das "Ende" erreicht wird.

STEPPING LINES, NOT STATEMENTS

Als Sprache ist Pascal anweisungsbasiert. Aber im Debugger funktioniert die Ausführung zeilenweise. Dies kann sich in verschiedenen Szenarien manifestieren. Sie können eine Zeile mit mehreren Anweisungen darin haben.

```
1.   a := 1; b := 2; c := a+b;
```

In diesem Fall führt stepping alle 3 Anweisungen auf einmal aus.

Sie können auch eine einzige Anweisung haben, die sich über mehr als eine Zeile erstreckt;

```
1.                                     writeln(
2.                                     random(1),
3.                                     random(2),
4.                                     random(3)
5.                                     );
```

In diesem Fall kann jede Zeile separat ausgeführt werden. Ihr Projekt führt zunächst den Code für jeden der 3 Parameter aus (Aufrufe von "random") und sobald es die Werte hat, ruft es "writeln" mit den Werten auf. Da FPC die Funktionsargumente von rechts nach links (rückwärts) auswertet, wird, wenn Sie den obigen Code schrittweise ausführen, als erste Zeile "random(3)", dann "random(2)", random(1) und schließlich "writeln" ausgeführt.





Wenn es sich dabei um Funktionen in Ihrem eigenen Code handelt und diese mit Debug-Informationen versehen sind, können Sie in jeder Zeile entscheiden, ob Sie mit "step in" (F7) in die Funktionen gehen wollen.

Beachten Sie, dass der Code zwischen der Ausführung der "random"-Anweisungen bis zur "writeln"-Zeile gehen kann, da er das Ergebnis für den späteren Aufruf speichert.

AUSSTEIGEN - UND WIEDER EINSTEIGEN

"step out", das kann weder zeilenweise noch anweisungsweise bedeuten. Weiter oben in diesem Artikel wurde erwähnt, dass "step out" entweder zu der Zeile zurückkehren kann, die die eigentliche aufrufende Anweisung enthielt, oder zur nächsten Zeile nach dieser Anweisung zurückkehren kann. Im Allgemeinen kehrt es in die Mitte der Zeile zurück, direkt an die Position nach dem Aufruf. Der Rest der Zeile kann dann zum Beispiel die Zuweisung des zurückgegebenen Wertes an eine Variable sein. Oder es können weitere Anweisungen oder weitere Aufrufe sein.

1. DoSomething (GetFoo (), GetBar ());

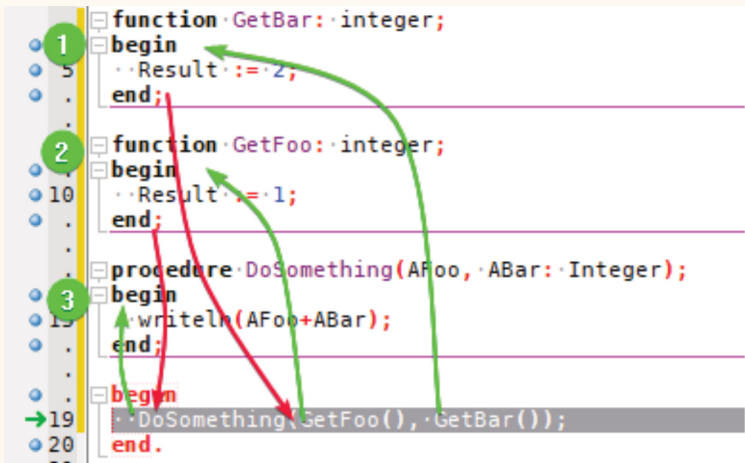


Abbildung 7

Wenn Sie in der obigen Zeile einsteigen "Step in", geben Sie "GetBar" ein (beachten Sie, dass Parameter von rechts nach links ausgewertet werden). Wenn Sie nun aus "GetBar" aussteigen, können Sie immer noch "GetFoo" oder "DoSomething" oder beide eingeben. Wenn "step out" in die nächste Zeile nach der Anweisung gehen würde, dann wären diese beiden Funktionen bereits ausgeführt worden.

Da aber "step out" (rote Pfeile) in der Mitte der Zeile stoppt, können Sie, sobald Sie aus "GetBar" herausgetreten sind, wieder "Step in" verwenden und "GetFoo" und danach "DoSomething" eingeben. Wenn Sie nur "GetFoo" eingeben wollen, müssen Sie in "GetBar" einsteigen. Aber Sie können dann sofort "Step out" und "Step into" "GetFoo".

Beachten Sie, dass dies nur funktioniert, wenn Sie "step out" verwenden. Wenn Sie "GetBar" eingeben und einen einzelnen Schritt bis zur Anweisung "end" machen und dann mit "step over" zurückkehren, führt der Debugger den Rest der aufrufenden Zeile aus und hält erst bei der übernächsten Zeile an.

Wenn Sie den gesamten Code von "GetBar" durchlaufen haben und bei der "End"-Anweisung angelangt sind, können Sie mit "Step into" (F7) in die nächste Routine, nämlich "GetFoo", wechseln. In diesem Fall kehrt "Step into" in die Mitte der aufrufenden Zeile zurück und springt sofort in die nächste Routine. (Bei einigen Debugger-Einstellungen kann es sein, dass er nur aussteigt und einen 2.)





“RUN TO CURSOR” ODER “STEP OVER TO CURSOR”

Manchmal kann es notwendig sein, mehrere Zeilen auf einmal zu durchlaufen.

Vielleicht, um eine Schleife zu verlassen, die noch zu viele Iterationen zu durchlaufen hat und nicht mit **F8** einzeln durchlaufen werden konnte.

Das Setzen eines **Breakpoints** ist zwar möglich, aber nur, wenn er nicht rekursiv ist. Hierfür können Sie "step over to cursor" verwenden.

Positionieren Sie den Textcursor auf die Zeile, in der Sie anhalten möchten. Drücken Sie dann **F4**. Der Code wird ausgeführt, bis er entweder diese Zeile erreicht oder die aktuelle Funktion zu ihrem Aufrufer zurückkehrt. Letzteres ist ein Sicherheitsnetz, falls die Zeile aus irgendeinem Grund nicht erreicht wurde (z.B. wenn sie sich in einem bedingten Block befand). "Step over to cursor" macht nur dann eine Pause auf dieser Zeile, wenn sie in der aktuellen Aufrufebeine erreicht wird. Sie wird ignoriert, wenn sie in einer Rekursion erreicht wird.

Da "Step over to cursor" die Anwendung nur in der aktuellen Aufrufebeine anhält, kann es nicht verwendet werden, um zu einer Zeile außerhalb der aktuellen Funktion zu laufen.

Um dies zu tun, können Sie "run to cursor" aus dem Menü verwenden. Dies ist dasselbe wie das Setzen eines **Breakpoints**, Ausführen und Entfernen des **Breakpoints**. "Run to cursor" hält auch an, wenn die Zeile innerhalb rekursiver Aufrufe getroffen wird.

Hinweis 2: In Lazarus vor Version 2.4 hat der FpDebug basierte Debugger ein Problem, das in einigen Fällen dazu führen kann, dass "Step out" zu früh pausiert. Je nach der von Ihnen verwendeten Debugger-Konfiguration (z.B. FP-Lldb auf Mac) sind diese beiden Befehle möglicherweise nicht verfügbar.

ZUSAMMENFASSUNG

In diesem Artikel haben wir 3 Methoden zur schrittweisen Bearbeitung des Codes kennengelernt.

- **Step over**
Führen Sie den Code in der aktuellen Zeile aus und pausieren Sie bei der nächsten Zeile.

- **F8**
- **Menü:** Run → Step over
- **Toolbutton:**
- **Step into**



Wenn der aktuelle Code einen Funktionsaufruf enthält, springen Sie in die erste Zeile der aufgerufenen Routine.

Dient auch als Abkürzung für "Step out" + "Step into", wenn Sie sich in der "End"-Zeile einer Routine befinden und eine andere Routine von derselben Zeile aus aufgerufen wird, von der aus diese Routine aufgerufen wurde.

- **F7**
- **Menü:** Run → Step into
- **Toolbutton:**
- **Step out**



Führt den Rest der aktuellen Routine aus und pausiert in der aufrufenden Zeile.

Hält in der Mitte der aufrufenden Zeile an, so dass weitere Routinenaufrufe in dieser Zeile "hineingeschaltet" werden können.

- **Shift-F8**
- **Menü:** Run → Step out
- **Toolbutton:**
- **Step over to cursor**



Führt den Code aus, bis er die Zeile erreicht, in der sich der Textcursor befindet.

Reagiert nur in der aktuellen Funktion und ignoriert Treffer bei Rekursionen.

Nur in **Lazarus** seit Version 2.2 verfügbar.

- **F4**
- **Menü:** Run → Step over to cursor
- **Run to cursor**

Führt den Code aus, bis er die Zeile erreicht, in der sich der Textcursor befindet.

Verhält sich so, als würden Sie zu einem **BreakPoint** in dieser Zeile laufen.

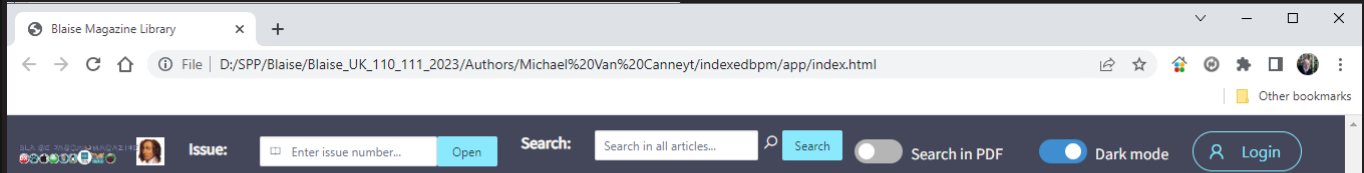
- **Menü:** Run → Run to cursor

Die Bilder für die **Tool-Buttons** haben sich zwischen den Lazarus-Versionen geändert. Sie können sich in Ihrer Version unterscheiden.



THE NEW FUTURE BLAISE PASCAL LIBRARY 2023

ON USB STICK INCLUDING THE INDEXER FOR ALL ITEMS AND PER ITEM ON CREDIT CARD USB STICK



NO ISSUE SELECTED
BLAISE PASCAL MAGAZINE

← < > > → 100 Load PDF...

BLAISE PASCAL MAGAZINE LIBRARY ARTICLE 1 SERVER PAGE 1/20
BY INTERNET AND ON USB STICK Written by Michael van Canneyt

Starter Expert

ABSTRACT
Blaise Pascal Magazine offers subscribers a library : a collection of all issues available till now. In this article we show how the PDF indexer application presented in the previous articles about indexing PDF files will be used to rewrite and enhance the Blaise Pascal Magazine library.

INTRODUCTION
In several earlier contributions, we showed how to show a PDF file in an offline PAS2JS application, and how to index PDF files and use that index to search and download PDF files. In this article, we'll show how to combine all these techniques to rewrite the Blaise Pascal Magazine library as a Pas2js application that works both offline and online. The new edition of the Blaise Pascal Magazine library will need to have the following features:

- It must work as a **local application**, distributed on a **USB stick**.
- It must work as a web application, deployed on the Blaise Pascal Magazine website.
- When working locally, issues must be loaded from the local disk: usually a **USB stick**.
- The issues must be search-able.
- When working locally, and no Internet connection is available, then a (*limited*) search must be performed locally in the list of articles. If an internet connection is available, the application must be able to search globally and download a PDF if needed.
- **PDF Downloads** are limited to the downloads purchased by the magazine subscriber.
- the issue will be displayed.

All of the techniques needed to satisfy these requirements have been presented in previous articles.
In this article we bring everything together: this will require some refactoring of the code presented in the previous articles. Since we'll be needing a server part and a client part, we'll start by discussing the code changes needed on the server.

BLAISE PASCAL MAGAZINE

BLAISE PASCAL MAGAZINE

Editor in Chief: Delft Overboek
Eindhovenstraat 21 5022 KA
Eindhoven Netherlands

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

Blaise Pascal Magazine 110 2023

49

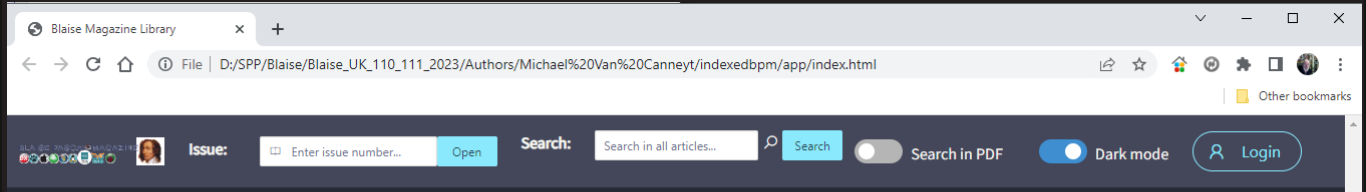
AVAILABLE ON YOUR OWN USB STICK

IMMEDIATE SEARCH

OVER ALL FILES AND ISSUES

THE NEW FUTURE BLAISE PASCAL LIBRARY 2023

ON INTERNET INCLUDING THE INDEXER FOR ALL ITEMS AND PER ITEM ON CREDIT CARD USB STICK



BLAISE PASCAL MAGAZINE 110 / 111

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



```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): Boolean
begin
  Result := True;
  if ATestIndex > 1 then
    with FoundPrimes do
    begin
      if Contains(APrimeCandidate) then
        exit;
    end;
  end;
  Result := APPrimeCandidate mod ATestIndex = 0;
  if not Result then
    exit;
  Result := (APPrimeCandidate mod ATestIndex) <> 0;
end;

procedure AddPrime(APrimeCandidate: Integer; ATestIndex: Integer);
var
  i: Integer;
begin
  for i := 2 to APrimeCandidate div ATestIndex do
    if CheckPrime(i * ATestIndex) then
      AddPrime(i * ATestIndex);
  end;
end;
```

AVAILABLE ON INTERNET FOR
**IMMEDIATE
SEARCH**
OVER ALL FILES AND ISSUES

Von Detlef Overbeek

Der Originalartikel wurde in Nature, Ausgabe 11 May 2023/Vol.617 veröffentlicht.

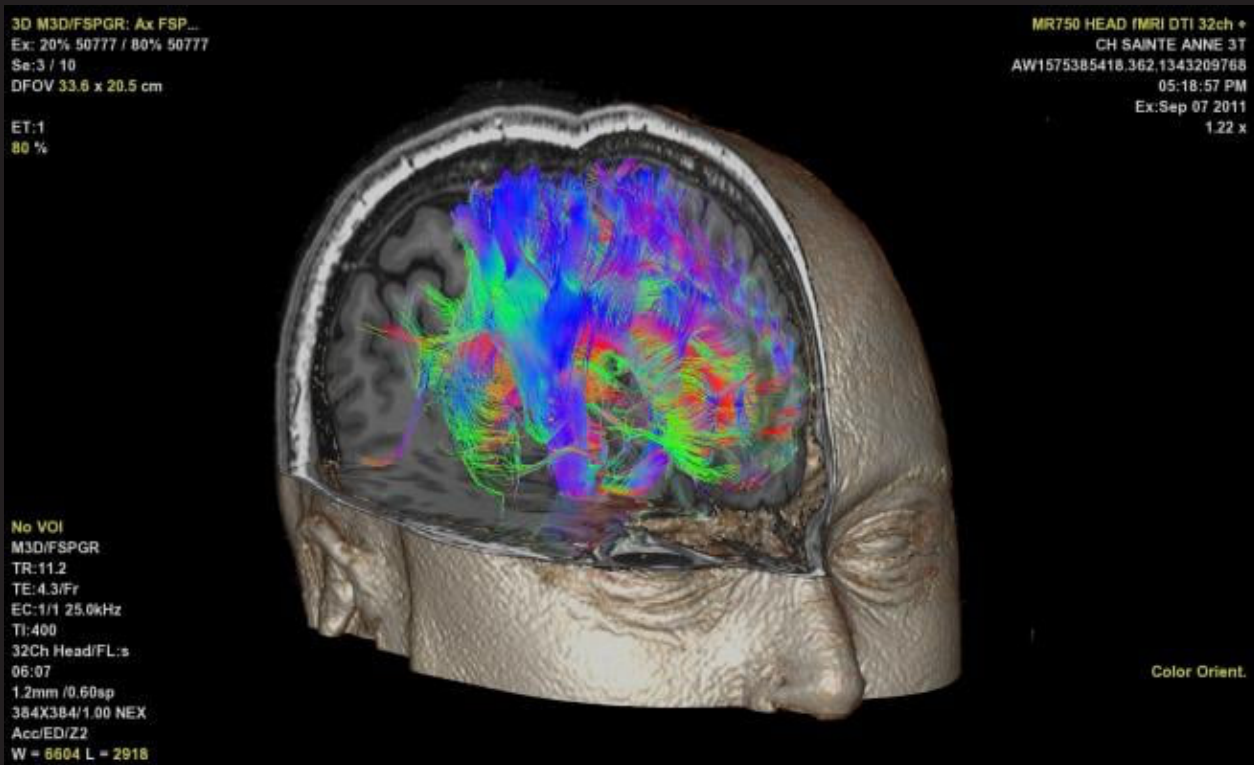


Abbildung 1 von <https://www.itnonline.com/content/machine-learning-uncovers-new-insights-human-brain-through-fmri>

Ein Gehirnschanner kann jetzt, zumindest zeitweise, die Hintergrundstimme in Ihrem Kopf entschlüsseln.

Forscher haben die erste nicht-invasive Technik entwickelt, um den Inhalt imaginärer Sprache zu analysieren. Sie kann ein potenzielles Kommunikationsmittel für Menschen sein, die nicht sprechen können. Aber wie nah ist die derzeit verfügbare Technologie, die nur eine geringe Genauigkeit aufweist, an der Verwirklichung eines echten Gedankenlesens?

Wie können die politischen Entscheidungsträger sicherstellen, dass diese Fortschritte (*in der Zukunft*) nicht falsch angewandt werden?

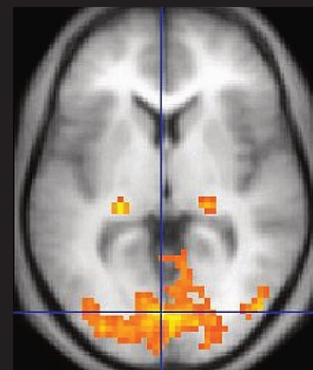
Die meisten der heute verwendeten Spracherkennungssysteme nutzen Gehirnimplantate, um die Aktivität der motorischen Hirnrinde des Benutzers zu verfolgen und die Worte zu erraten, die er mit seinen Lippen auszusprechen versucht. Alexander Huth und Jerry Tang, Informatiker an der University of Texas in Austin, haben gemeinsam mit anderen Forschern Algorithmen der Künstlichen Intelligenz (KI) mit der funktionellen Magnetresonanztomographie (fMRI), einer nicht-invasiven Methode zur Messung der Gehirnaktivität, kombiniert, um die wahre Bedeutung hinter den Gedanken zu verstehen.

(Die funktionelle Magnetresonanztomographie oder funktionelle MRT misst die Hirnaktivität, indem sie Veränderungen im Zusammenhang mit dem Blutfluss erkennt. Diese Technik beruht auf der Tatsache, dass der zerebrale Blutfluss und die neuronale Aktivierung gekoppelt sind. Wenn ein Bereich des Gehirns aktiv ist, erhöht sich auch der Blutfluss in dieser Region).

Abbildung 2 (aus Wikipedia) Ein fMRI-Bild mit gelben Bereichen, die eine erhöhte Aktivität im Vergleich zu einer Kontrollbedingung aufweisen.

Zweck Misst die Gehirnaktivität und erkennt Veränderungen durch den Blutfluss.

Die Gehirne von niemandem sollte entschlüsselt werden ohne deren Zustimmung.



Diese Algorithmen, die auch als LARGE LANGUAGE MODELS (LLMs) bekannt sind, ermöglichen ChatGPT und sind darauf trainiert, das nächste Wort in einer Textpassage vorausszusehen. In einer Studie, die in J. Tang et al. Nature Neuroscience 26, 858-866; 2023 beschrieben wird, ließen die Forscher drei Freiwillige in einem fMRI-Scanner liegen und ihre Gehirnaktivität aufzeichnen, während sie Podcasts hörten.

Die Forscher erstellten eine kodierte Karte davon, wie das Gehirn jedes Einzelnen auf verschiedene Wörter und Ausdrücke reagiert, indem sie dieses Wissen mit der Fähigkeit des LLM kombinierten, zu verstehen, wie Wörter miteinander in Beziehung stehen.

Die Teilnehmer hörten dann entweder eine Geschichte, stellten sich das Erzählen einer Geschichte vor oder sahen sich einen Stummfilm an, während die Forscher die fMRI-Aktivität aufzeichneten. Die Forscher versuchten dann, diese neue Hirnaktivität zu entschlüsseln, indem sie eine Kombination aus den Mustern, die sie zuvor für jede Person kodiert hatten, und Algorithmen verwendeten, die herausfinden, wie ein Satz auf der Grundlage anderer darin enthaltener Wörter wahrscheinlich aufgebaut sein wird.

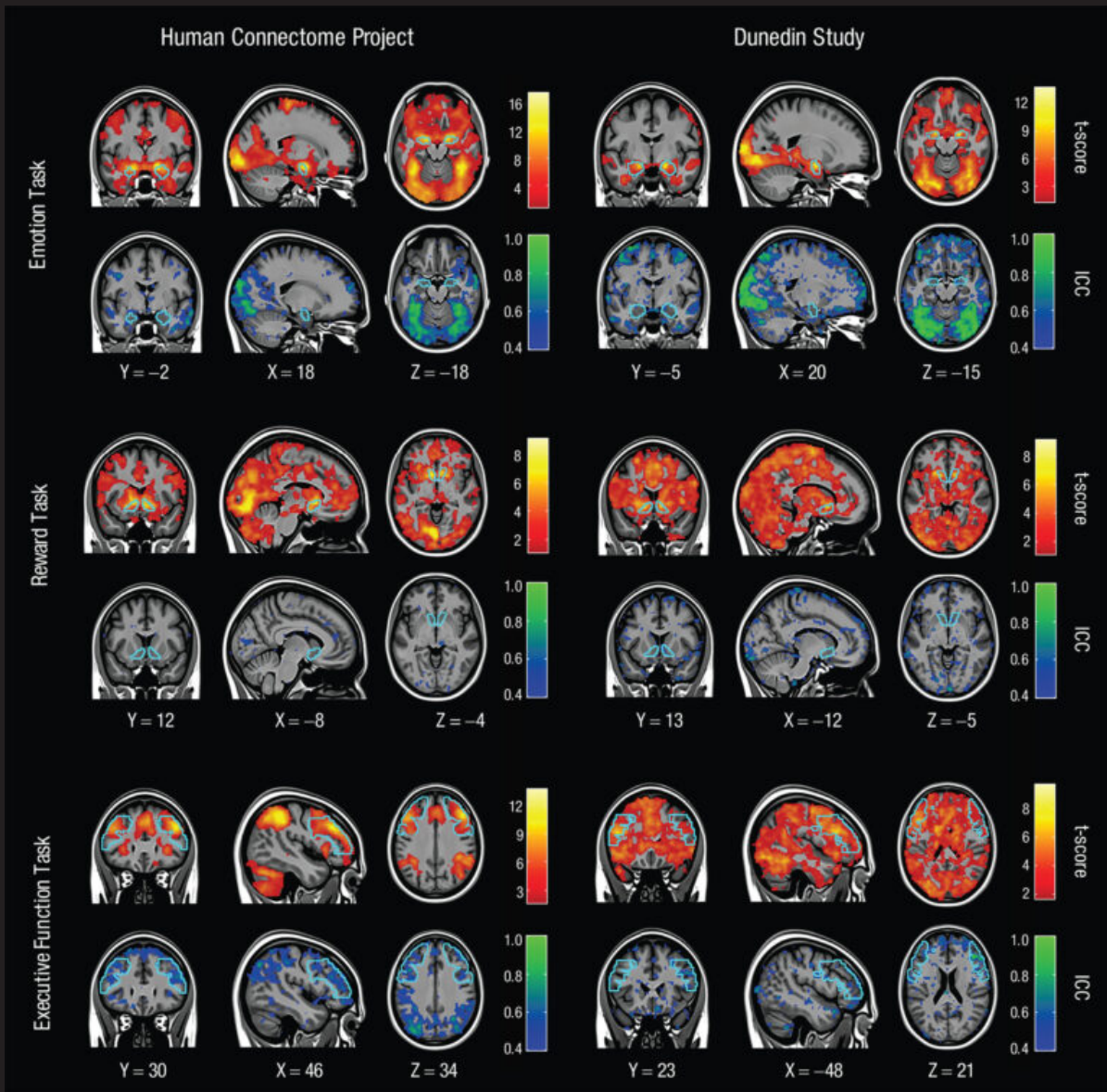


Abbildung 3 Gehirnskans mit MRT-Abbildung für 3 Aufgaben an 2 verschiedenen Tagen. Warme Farben zeigen, wie sich die Ergebnisse in Gruppen verhalten. Kühle Farben zeigen, dass die Ergebnisse von Person zu Person weniger zuverlässig sind. (Annchen Knodt/Duke University)



Außerdem hat es eine gute Arbeit geleistet, indem es genau erklärt hat, was die Zuschauer in den Filmen sehen. Aber viele der Sätze, die es lieferte, waren falsch. Die Forscher entdeckten auch, dass die Technologie leicht zu überlisten war. Der Decoder war nicht in der Lage, die Wörter zu erkennen, die die Teilnehmer hörten, wenn sie sich eine andere Geschichte vorstellten, während sie sich eine aufgezeichnete Geschichte anhörten. Außerdem variierte die kodierte Karte von Person zu Person, was es den Forschern unmöglich machte, einen universellen Decoder zu entwickeln.

WAKE UP CALL

Bei der Frage, ob die jüngste Entwicklung eine Bedrohung für die geistige Privatsphäre darstellt, sind die Neuroethiker anderer Meinung. Der BIOETHIKER Gabriel Lázaro-Muoz von der Harvard Medical School in Boston, Massachusetts, sagte:

"Ich rufe nicht zur Panik auf, aber die Entwicklung ausgeklügelter, nicht-invasiver Technologien wie dieser scheint näher am Horizont zu sein, als wir erwartet haben. Ich denke, das ist ein wichtiger Weckruf für die Öffentlichkeit und die politischen Entscheidungsträger.

Adina Roskies, ein Wissenschaftsphilosoph an der Dartmouth University in Hanover, in New Hampshire, ist anderer Meinung. Er behauptet, dass die Technologie derzeit zu fehlerhaft und schwierig zu handhaben ist, um eine Gefahr darzustellen. Da fMRI-Geräte nicht tragbar sind, ist es eine Herausforderung, das Gehirn einer Person ohne deren Zustimmung zu scannen. Sie fragt sich auch, ob sich die Ausbildung eines Decoders für eine Person zeitlich und finanziell lohnen würde, wenn das Ziel etwas anderes wäre als die Wiedererlangung der Kommunikationsfähigkeit.

Es ist (noch) nicht an der Zeit, sich Sorgen zu machen. Es gibt zahlreiche zusätzliche Möglichkeiten für die Regierung, zu erfahren, was wir denken. Für Greta Tuckute, eine kognitive Neurowissenschaftlerin am Massachusetts Institute of Technology in Cambridge, ist es ermutigend, dass Menschen das Dekodierungssystem leicht überlisten können, indem sie an andere Dinge denken, und dass es nicht auf alle Personen anwendbar ist. Es ist eine fantastische Illustration des Grades an Handlungsfähigkeit, den wir tatsächlich besitzen, sagt sie. Tun Sie dies mit Vorsicht.

Roskies warnt davor, dass es zu Problemen kommen kann, wenn Anwälte oder Richter den Decoder nutzen, ohne sich seiner technischen Grenzen bewusst zu sein. Zum Beispiel wurde der Satz "*Ich bin gerade [aus dem Auto] gesprungen*" in der aktuellen Studie in "*Ich musste sie aus dem Auto schubsen*" übersetzt. Die Unterschiede sind so eklatant, dass sie den Ausgang eines Rechtsstreits erheblich beeinflussen können.

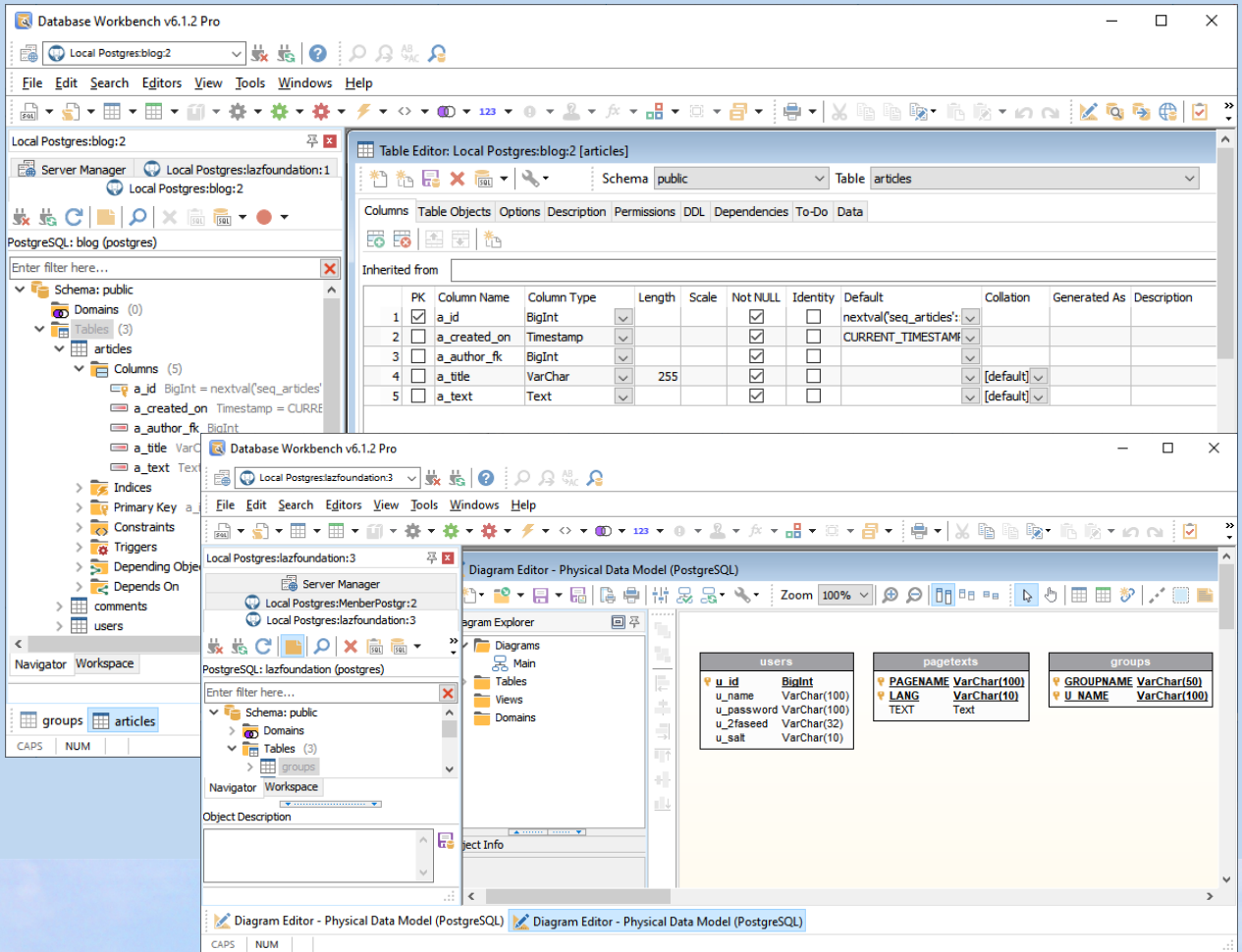
Tang erklärte auf einer Pressekonferenz, dass "der Lügendetektor nicht genau ist, sondern negative Folgen hat." Die Gehirne von niemandem sollten ohne seine Zustimmung entschlüsselt werden. Er und Huth forderten die Behörden auf, sich intensiv mit den rechtlichen Aspekten des Einsatzes von Technologien zum Gedankenlesen zu befassen.

Laut Lázaro-Muoz könnte diese Regelung nach dem Vorbild eines US-Gesetzes erfolgen, das es Versicherern und Arbeitgebern verbietet, genetische Informationen zu diskriminierenden Zwecken zu nutzen. Er ist besonders besorgt über die Auswirkungen des Decoders auf Personen, die aufdringliche, unwillkommene Ideen haben könnten, anderen zu schaden, obwohl sie niemals eine solche Tat begehen würden.

Ein Neurowissenschaftler namens Francisco Pereira am US NATIONAL INSTITUTE OF MENTAL HEALTH in Bethesda, Maryland, ist der Meinung, dass es unklar ist, wie präzise die Decoder werden oder ob sie jemals universell und nicht personenbezogen werden können. Auch wenn der Decoder letztendlich besser wird, wenn er das nächste Wort in einer Reihe vorhersagen kann, könnte er Schwierigkeiten haben, Metaphern oder Sarkasmus zu entschlüsseln.

Laut Pereira sind das Zusammensetzen von Wörtern und das Verständnis dafür, wie das Gehirn die Beziehungen zwischen ihnen kodiert, zwei sehr unterschiedliche Dinge."

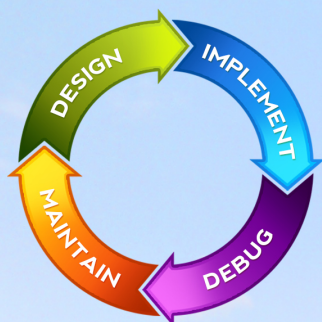




Introducing

Database Workbench 6

database development environment



Consistent user interface, modern code editors, Unicode enabled, HighDPI aware, ER designer, reverse engineering, meta data browsing, visual object editors, meta data migration, meta data compare, stored routine debugging, SQL plan visualizer, test data generator, meta data printing, data import and export, data pump, Grant Manager, DBA tasks, code snippets, SQL Insight, built in VCS, report editor, database meta data search, numerous productivity tools and much more...

for SQL Server, Oracle, MySQL, MariaDB, Firebird, InterBase, NexusDB and PostgreSQL



Database tools for developers

www.upsene.com

VERWENDUNG DES DELPHI-COMPILERS IN DER LAZARUS-IDEE

SEITE 1/7

By Michaël Van Canneyt

ZUSAMMENFASSUNG

Die Lazarus IDE hat einige Vorteile gegenüber der Delphi IDE. Zum einen funktioniert sie plattformübergreifend: Sie können Lazarus auf Ihrem Mac oder unter Linux verwenden.

Können Sie damit also an Ihrem Delphi-Projekt arbeiten und das Ergebnis mit dem Delphi-Compiler kompilieren?
das Ergebnis mit dem Delphi-Compiler kompilieren?
In diesem Artikel zeigen wir Ihnen, wie.



1 EINLEITUNG

Die Delphi IDE ist nur für Windows geeignet.

Lazarus kann auf vielen Plattformen verwendet werden.

Was also, wenn Sie an Ihrem Delphi-Code auf Ihrem Mac arbeiten, aber trotzdem mit Delphi kompilieren möchten?

Theoretisch ist das möglich: Der Delphi-Befehlszeilen-Compiler kann problemlos in wine ausgeführt werden, einer Plattform, die eine Windows-kompatible API auf UNIX-Systemen bereitstellt, die Windows-Binärdateien direkt auf LINUX (und auf MAC) ausführen kann.

Allerdings ist zu beachten, dass nicht alle Versionen von **Wine** für die Ausführung des Delphi-Befehlszeilen-Compilers geeignet sind (*geschweige denn für die vollständige IDE*). Selbst unter Windows kann es Gründe geben, die Lazarus IDE der Delphi IDE vorzuziehen:

Die Lazarus IDE ist schneller und verfügt über bessere Code-Tools als die Delphi IDE:

Die Code-Vervollständigung ist weitaus besser.

Sie könnten in Lazarus editieren und den Delphi-Befehlszeilen-Compiler in einem Terminal- oder Konsolenfenster ausführen.

Aber Lazarus kann noch mehr.

Obwohl es natürlich in erster Linie für Pascal-Code konzipiert ist, kann es auch zum Bearbeiten und Kompilieren von C-Code oder JAVASCRIPT verwendet werden - oder auch für jeden anderen Compiler.

Lazarus bietet eine API, die Ihnen hilft, einen Compiler aufzurufen und die Ausgabe dieses Compilers zu analysieren.

So gibt es zum Beispiel ein Package, das die Ausgabe des GCC (*GNU C Compiler*) Compilers analysieren und bei GCC-Fehlermeldungen an die richtige Stelle im Code springen kann.

2 DAS DELPHI TOOL

Delphi verfügt über einen Befehlszeilen-Compiler.

Kann diese API nicht verwendet werden, um den Delphi-Compiler auszuführen?

Die Antwort lautet: Ja, natürlich!

Seit einiger Zeit gibt es ein Paket, das genau das tut:
das **Delphitool-Package**.

Es befindet sich im Lazarus Quellbaum und wird mit der nächsten Hauptversion ausgeliefert. Es ermöglicht Ihnen den Aufruf des Delphi-Compilers, analysiert die Ausgabemeldungen des Delphi-Compilers, und zeigt sie im Nachrichtenfenster an und zwar so, dass Sie auf die Meldung klicken können und die IDE springt zu dem entsprechenden Quellcode.

Das Paket macht aber noch mehr als das:

Optional übernimmt es die Compiler-Befehlszeilenoptionen, die für den FPC-Compiler verwendet werden, und konvertiert sie in entsprechende Delphi-Befehlszeilenoptionen: Pfade für Units, Generierung von Debug-Informationen oder Optimierungen sind nur einige der Optionen, die in Delphi-Optionen umgewandelt werden.



Die Optionen werden in eine Konfigurationsdatei geschrieben und Sie können diese Konfigurationsdatei in Ihrer Kommandozeile verwenden, um den Delphi Compiler aufzurufen.

Da die einzige praktische Möglichkeit den Delphi-Befehlszeilen-Compiler unter Linux oder Mac auszuführen, die Verwendung von Wine ist, bietet Ihnen die Lazarus IDE an, die Dateinamen in den Ausgabemeldungen von der Windows-Notation in die Unix-Notation zu konvertieren:

Sie ordnet die Laufwerksbuchstaben den Verzeichnissen auf Ihrem Linux- oder Mac-Rechner zu und ändert in slashes in slashes.

Es analysiert die Zuordnung der **Wine**-Laufwerksbuchstaben, um die Pfade korrekt zuordnen zu können.

Die Installation dieses Pakets erfolgt auf die übliche Weise:

Das Package befindet sich im Quellbaum von Lazarus, im Entwicklungszweig des Lazarus Git-Repositorys. Es befindet sich im Verzeichnis

`components/compilers/delphi` Verzeichnis, und die Package-Datei ist `lazdelphi.lpk`.

Wenn Sie keine Git-Version von Lazarus auf Ihrem System haben, können Sie auch einfach die Dateien für dieses Package von Gitlab herunterladen und das Package in einer älteren IDE installieren: Die APIs, die von diesem Package verwendet werden, existieren schon seit geraumer Zeit.

Öffnen Sie die Package-Datei `lazdelphi.lpk` und installieren Sie sie über das Kontextmenü 'Verwenden - Installieren'. Nach dem Neuaufbau der IDE und dem Neustart der IDE sollten Sie neue Seiten im Dialog IDE-Werkzeuge - Optionen und im Dialog Projektoptionen haben.

③ IDE-KONFIGURATION

Bevor Sie den Delphi Compiler verwenden können, müssen Sie die Lazarus IDE konfigurieren. Im Dialog Extras - Optionen gibt es einen neuen Rahmen 'Delphi Compiler', siehe Abbildung 1 auf Seite 3 dieses Artikels. Auf dieser Seite sind die folgenden Einstellungen verfügbar:

Delphi Compiler ausführbare Datei

Dies ist der vollständige Pfad des Delphi-Befehlszeilen-Compilers.

Erweiterung der Delphi-Konfigurationsdatei

Wenn die IDE eine Konfigurationsdatei für den Delphi

Kommandozeilen-Compiler erzeugt, verwendet sie den gleichen Namen wie die Lazarus Projektdatei, jedoch mit der hier angegebenen Erweiterung.

Dateinamen von Windows in Unix-Notation umwandeln

Diese Option ist nur unter Linux oder Mac verfügbar.

Wenn diese Option aktiviert ist, wird die IDE alle Dateinamen in der Ausgabe des Delphi Compilers in UNIX Notation um und ersetzt die von Wine zugewiesenen Laufwerksbuchstaben durch das richtigen Verzeichnis auf Ihrem UNIX-System.

Zusätzliche Compiler-Optionen

diese Optionen werden in der Befehlszeile an den Compiler übergeben, zusätzlich zur Konfigurationsdatei, für jedes Projekt, das Sie kompilieren möchten.

Unter Linux und Mac-OS kann der Compiler über **Wine** aufgerufen werden. Zum Lieferumfang des Delphi-Tools gehört ein Skript `dcc.sh`, das dies für Sie erledigt:

Es wandelt Pfade um und macht die erzeugte Binärdatei ausführbar macht (was der delphi Compiler nicht tut). Sie können die IDE auf dieses Skript verweisen, anstatt auf wine und die eigentliche DCC-Binärdatei.

Das Skript kann auch über die Befehlszeile verwendet werden, es ist nicht spezifisch für Lazarus.



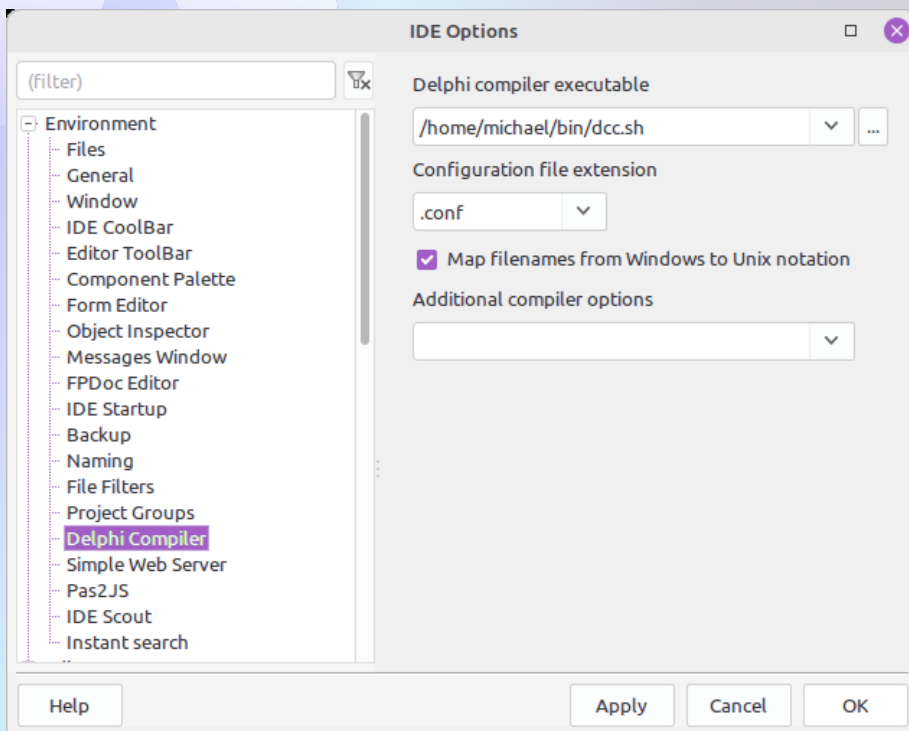


Abbildung 1: Das Delphi-Tool Konfigurationsseite

4 PROJEKTKONFIGURATION

Um den Delphi-Compiler für ein Projekt verwenden zu können, müssen Sie das Projekt dafür konfigurieren. Dies geschieht natürlich im Dialogfeld Projektoptionen.

Es gibt 2 Schritte zur Konfiguration. Der erste ist allgemeiner Natur und kann im Rahmen 'Delphi Compiler' konfiguriert werden, siehe Abbildung 2 auf Seite 4 dieses Artikels - hier können die Optionen eingestellt werden:

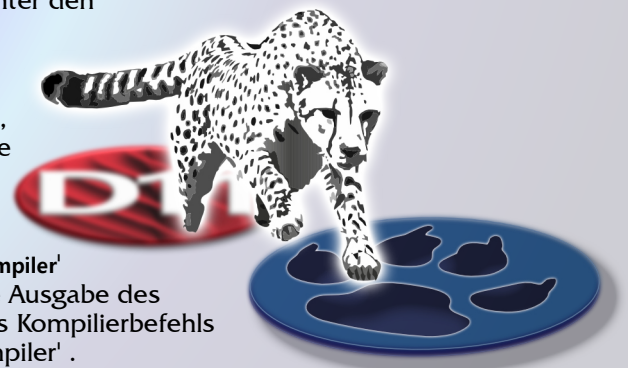
Delphi-Konfigurationsdatei basierend auf den FPC-Compiler-Optionen generieren, wenn diese Option aktiviert ist, generiert die IDE bei jeder Kompilierung eine Konfigurationsdatei mit den aktuellen Optionen aus der FPC-Compilerkonfiguration.

Zusätzliche Compiler-Optionen

diese Optionen werden zusätzlich zur Konfigurationsdatei auf der Kommandozeile an den Compiler übergeben, allerdings nur für dieses Projekt.

Die zweite Konfiguration besteht darin, den Compileraufruf anzupassen. Dies geschieht im Rahmen 'Compiler-Befehle', dem letzten Rahmen unter den Compiler-Optionen. In Abbildung 3 auf Seite 4 dieses Artikels sehen Sie, was Sie tun müssen:

1. Geben Sie in das Feld 'Ausführen vor' den Delphi Compile Befehl ein, den Sie ausführen möchten, wenn Sie den Befehl 'compile' oder 'build' Befehle ausführen wollen und markieren Sie die 'compile' 'build' und 'run' Kontrollkästchen. Mehr über den Befehl weiter unten.
2. Markieren Sie im Feld 'Ausführen vor' den Delphi Compiler' in der Liste der Parser. Dies weist die IDE an, die Ausgabe des Kompilierbefehls zu parsen und die Ausgabe des Kompilierbefehls analysieren soll mit dem Parser-Tool 'Delphi Compiler' . Dieses Werkzeug wurde durch das Paket 'lazdelphi' registriert.
3. Deaktivieren Sie im Feld 'Compiler' die Kontrollkästchen 'Kompilieren', 'Erstellen' und 'Ausführen' die Markierungen.



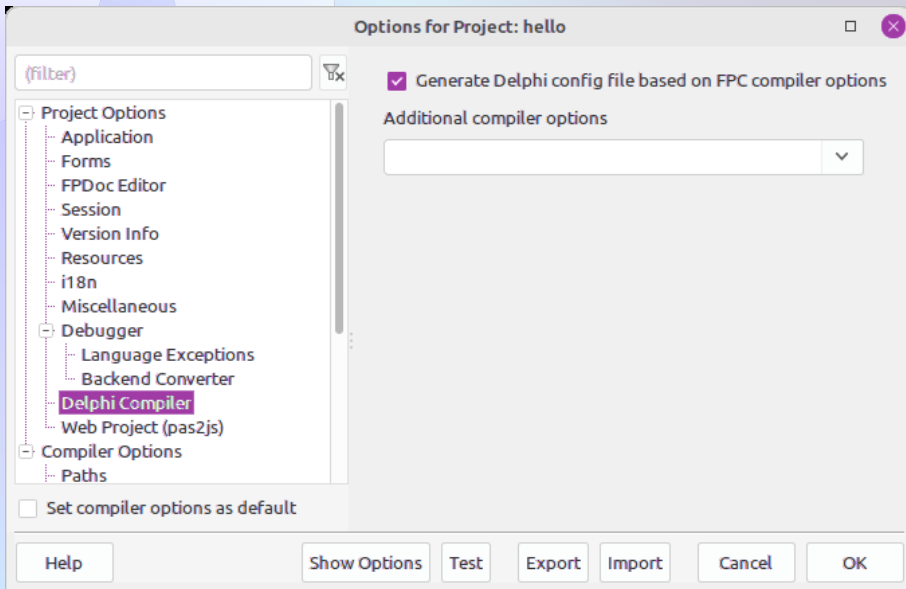


Abbildung 2: Die Projekt-Delphi-Konfigurationsseite

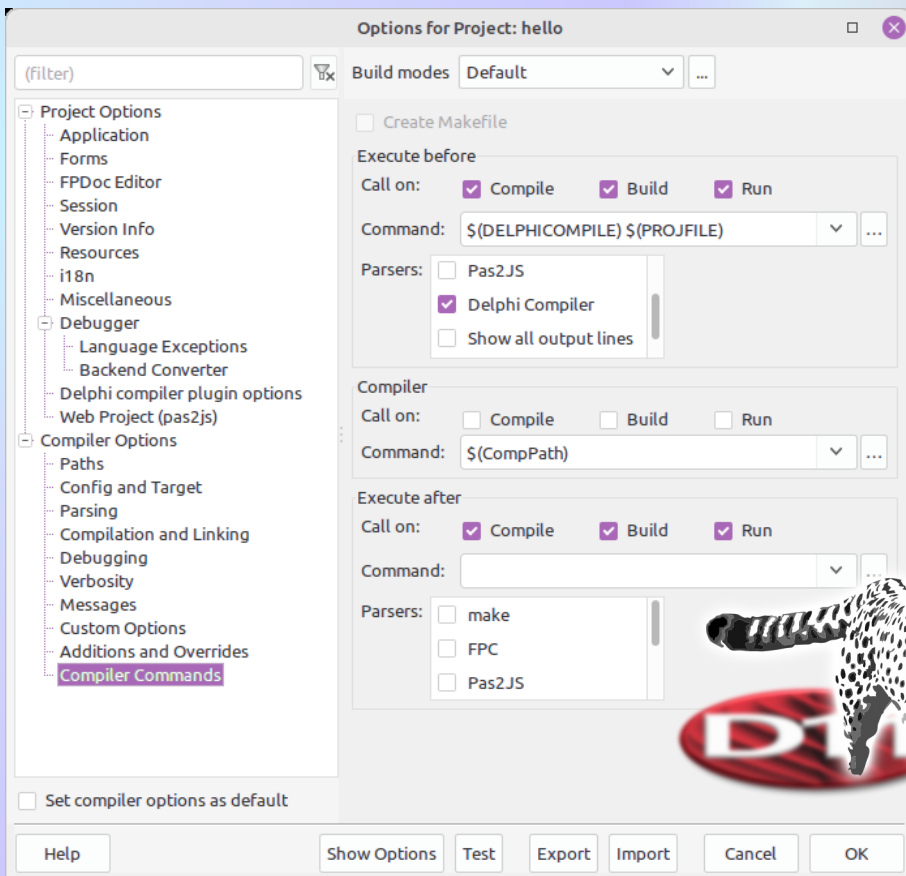
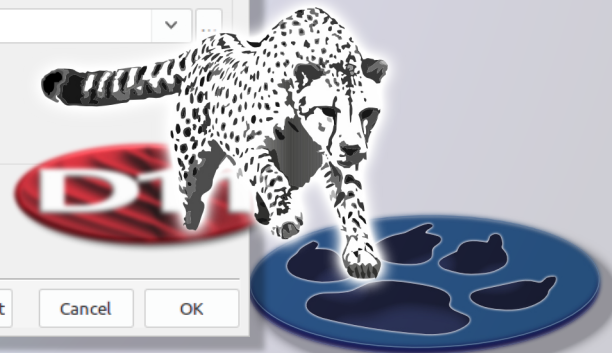


Abbildung 3: Die Seite mit den Compiler-Befehlen für das Projekt



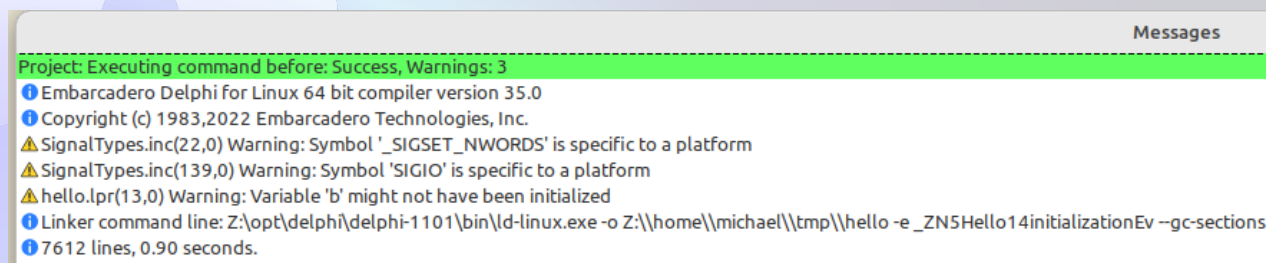


Abbildung 4: Ausgabe beim Kompilieren des Projekts

Danach wird beim Kompilieren oder Erstellen des Projekts in der IDE die Befehlszeile 'execute before' ausgeführt und der FPC-Compiler wird nicht aufgerufen.

Beachten Sie, dass Sie durchaus 2 Build-Modi haben können:

einen, um Ihr Projekt mit Delphi zu kompilieren,

einen, um Ihr Projekt mit FPC zu kompilieren.

Wie geben Sie also den Delphi-Kompilierbefehl im Bearbeitungsfeld 'Ausführen vor' an?

Sie können den Kompilierbefehl vollständig selbst angeben, zum Beispiel:

```
c:\delphi\bin\dcc32.exe -V c:\projects\myproject.dpr
```

Oder Sie können Makros verwenden. Das lazdelphi-Paket definiert mehrere Makros, die Sie verwenden können, um Ihren Befehl nach Ihren Vorstellungen zusammenzustellen:

DCC

Dieses Makro wird zum Binärpfad des Delphi-Compilers erweitert, wie er in den IDE-Optionen eingestellt ist.

DCCCONFIG

Dieses Makro wird zu der Delphi-Compiler-Konfigurationsdatei erweitert die von der IDE für Ihr Projekt erzeugt wurde. Dem Dateinamen wird ein @-Zeichen (*at-sign*) vorangestellt, wenn er nicht leer ist (das @-Zeichen ist die Art, wie die Konfigurationsdatei auf der Kommandozeile des Delphi-Compilers angegeben)

DCCARGS

Dies wird zur Verkettung der 'zusätzlichen Compiler-Optionen' die in den globalen und projektspezifischen Einstellungen angegeben sind.

DELPHICOMPILE

Dieses Makro entspricht in der Tat den drei oben genannten Makros zusammen:

```
$(DCC) $(DCCARGS) $(DCCCONFIG)
```

es ist der Einfachheit halber vorhanden.

Der einfachste Kompilierbefehl lautet also:

```
$(DELPHICOMPILE) $(PROJFILE)
```

Genau wie in *Abbildung 3 auf Seite 4 dieses Artikels* dargestellt. Sobald dies alles erledigt ist, können Sie Ihr Projekt mit Delphi kompilieren, und das Ergebnis kann so aussehen, wie Sie es in *Abbildung 4 auf Seite 5* sehen. Wenn Sie auf die Warnung klicken, gelangen Sie an die richtige Stelle im Projekt.

⑤ DEBUGGING



Die Lazarus IDE bietet natürlich Debugging, und der Debugger verfügt über viele erweiterte Funktionen (*siehe die jüngsten Beiträge von Martin Friebe im Blaise Pascal Magazin*).



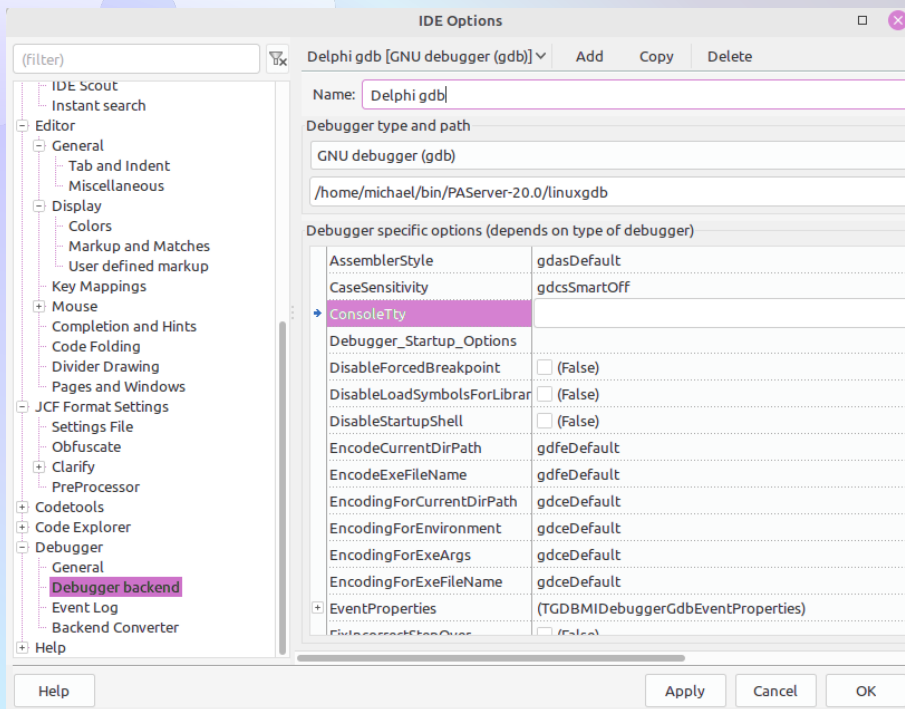


Abbildung 5: Definieren des Delphi-Debugger

Daher ist die Frage, ob die kompilierte ausführbare Datei in der Lazarus IDE debugged werden kann, natürlich relevant.

Die Antwort auf diese Frage lautet 'Ja, aber...':

Unter Linux ist es definitiv möglich, die von Delphi erzeugte ausführbare Datei zu debuggen. Dazu muss GDB (der GNU-Debugger) verwendet werden.

Um die besten Ergebnisse zu erzielen, wird eine speziell modifizierte Version von GDB benötigt. Delphi liefert diese GDB-Version als Teil seines Plattform-Assistenten 'PAServer' für Linux aus.

Glücklicherweise kann Lazarus dieses GDB-Executable verwenden. Sie können es in der Lazarus IDE im Dialog Werkzeuge - Optionen unter 'Debugger - Debug Backend' einrichten. Die IDE kann mit mehreren Debug-Backends arbeiten, daher müssen wir hier ein neues Debugger-Backend definieren. Dazu müssen die folgenden Aktionen durchgeführt werden:

1. Klicken Sie oben im Dialogfeld auf "Hinzufügen"(Add).
2. Geben Sie einen neuen Namen in das Eingabefeld 'Name' ein.
3. Als Debugger-Typ wählen Sie 'GNU Debugger (gdb)'.
4. Wählen Sie die 'linuxgdb' Binärdatei, die Teil von PAServer ist.

Das Ergebnis sieht dann so aus wie in *Abbildung 5 auf Seite 6 dieses Artikels*.

Sobald dies erledigt ist, können Sie Ihre Anwendung im Debugger ausführen: Sie können Breakpoints setzen, Sie können Variablen beobachten, wie in *Abbildung 6 auf Seite 7* gezeigt. Das **Debugging-Erlebnis** ist noch nicht einwandfrei: Delphi kodiert bestimmte Pascal-Typen anders als FPC, und die Lazarus IDE kennt diese Information (noch) nicht. Um also den Wert einiger Typen zu sehen, können einige Typecasts notwendig sein.

Der Autor hat das Debugging unter Mac-OS und Windows nicht getestet. Basierend auf dem verfügbaren Wissen über die Werkzeuge unter Windows und Mac-OS ist es sehr wahrscheinlich, dass: → *siehe nächste Seite*



- Unter Mac-OS kann Lazarus zum Debuggen mit Ildb verwendet werden (*dem Debugger auf dem Mac*) mit einer ähnlichen Technik wie unter Linux.
- Unter Windows wird das Debuggen voraussichtlich nicht möglich sein, da Delphi ein proprietäres Format für Debug-Informationen unter Windows verwendet.

```
. uses System.SysUtils;
.
5 {$Warnings on}
.
. function DoSomethingNice(a : string) : string;
.
. var
10  b : Integer;
.
. begin
.   b:=b+33;
14  Result:=IntToStr(B);
15 end;
.
. begin
.   Writeln(DoSomethingNice('soso'));
.   Writeln('Hello, world');
20 end.
21
```

Name	Value
Result	@0x7fffffff258:
b	34

Abbildung 6: Fehlersuche in einer mit Delphi erzeugten ausführbaren Datei

6 SCHLUSSFOLGERUNG

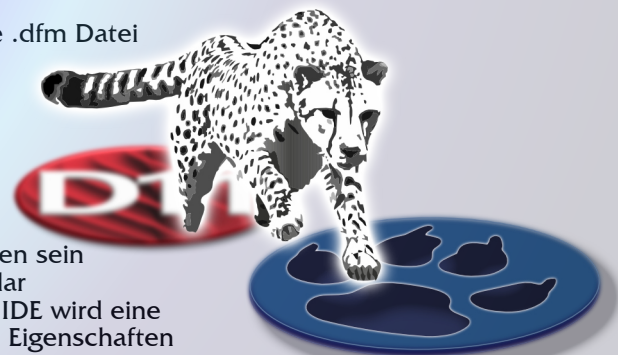
Das Delphi-Tool kann verwendet werden, um Ihren FreePascal- oder Delphi-Code mit dem Delphi-Compiler zu kompilieren, direkt aus der Lazarus IDE heraus.

Das Tool funktioniert so wie es jetzt ist, aber einige Verbesserungen sind noch geplant: Die Schnellkorrekturen, die für den FPC-Compiler existieren, können auch für den Delphi-Compiler implementiert werden. Weitere Verbesserungsvorschläge sind den Lazarus-Maintainern natürlich immer willkommen.

Ein Wort der Warnung:

Während die Codierung kein Problem darstellt, sollten Sie bei der Bearbeitung visueller Formulare vorsichtig sein: Wenn Sie eine Komponente (z.B. ein TEdit) zu einem Formular hinzufügen, gibt es in der LCL Eigenschaften, die es in Delphi nicht gibt.

Diese Eigenschaften werden von der Lazarus IDE in die .dfm Datei geschrieben. Wenn Sie das Programm ausführen und das Formular zur Laufzeit erstellt wird, kommt es zu einem Streaming-Fehler: Die Komponenten, die tatsächlich instanziiert werden, sind die Delphi-Komponenten, denen möglicherweise einige einige Eigenschaften fehlen. Umgekehrt, wenn Sie ein Delphi Formular in der Lazarus IDE laden, können VCL-Eigenschaften vorhanden sein (oder sogar komplette Komponenten) in dem Formular die kein Gegenstück in der LCL haben und die Lazarus IDE wird eine Fehlermeldung anzeigen und Sie fragen was mit diesen Eigenschaften gemacht werden soll.



CODE BETTER IN DELPHI

<https://leanpub.com/codebetterindelphi>

ALISTER CHRISTIE



Introduction

Programming better is somewhat unclear; better than what? I don't know how good at coding you are, but I'm sure you will find plenty to allow you to write better code by the end of this book.

My first book, *Code Faster in Delphi*, was much easier to write. "Faster" is something easily measured. For instance: I used to do something some way, but now I know a shortcut, and it's much quicker. "Better" is far more subjective and much harder to measure. It's also a much broader topic, and I could have written *much* more. I had to stop somewhere; otherwise, you'd never get the opportunity to enjoy this book. If you are concerned some topics are not as complete as they could be, don't worry. I have provided references to many excellent resources on various topics if you want to pursue things in more detail.

I aim to give you a book that helps your Delphi code be more maintainable, reliable, and scalable.

Maintainable can best be thought of as:

- Being able to return to your code six months or six years later and quickly understand the code's intention.
- Having projects that others can efficiently work on either with or after you.
- Being able to leverage your code to save you time in the future.

Reliable is building code that:

- Avoids bugs by its very nature.
- Is thoroughly tested.
- Will save you time in the future.

Scalable relates to:

- As your source code grows, it's still easy to maintain.
- You can reuse large portions of your codebase.
- Multiple people can work on the same codebase.

I have all 3 of these ideas in one book because they are often interlinked. More importantly, lacking one will often cause problems in the others. If your code is not maintainable, don't be surprised when it isn't reliable and won't scale.

Of these three topics, I've focused on maintainability for this book. This will help significantly with reliability and scalability, and you can get them just about for free. If you work in a large team, you will still need to consider how everyone can safely work on the code while still being productive.

<https://leanpub.com/codebetterindelphi>



Sections in this Book

This book has been divided into four parts.

Part One - Coding Tips looks at coding practices. There are many short code examples in this section.

Part Two - The Right Tool includes the Delphi IDE and other third-party tools that can help improve your code.

Part Three - Working with Code is a more detailed look at specific coding practices. The examples here are longer and go into more depth.

Part Four - Beyond Delphi covers topics that aren't Delphi specific but are fundamental to achieving maintainable, scalable and reliable Delphi code. It covers source control and project management, among other things.

begin

Make Code Maintainable

You write code once, but you'll read and maintain it forever. Writing code is usually the easy part (although it might not feel that way). It's the maintenance that's painful.

Maybe you've experienced this? Have you returned to the code you've written the day before and thought, "I have no idea what this does!"? If you can forget what your code does after 24 hours, what will it be like after five, ten, or even twenty years? Twenty years might sound like a long time, and yes, it is - particularly if you've just started your software development career. Many large projects developed in Delphi have lifetimes measured in decades.

It can be a terrifying thought - the code you are writing today might still be in use 20 years later. So as you are coding, take time to think, "if I came back to this procedure in 20 years, having no memory of writing it - will I be cursing or praising its author?" Code I created as a newbie is now legacy code, and I'm still working with some of it. I'm not exactly singing my own praises when I attempt to decipher what I was trying to do back then.

Code that lasts for decades is the code you want to write. Code that's useful and can be reused is successful code. However, successful code is not always good code. It's great to see that an application you hacked together five years earlier is still doing

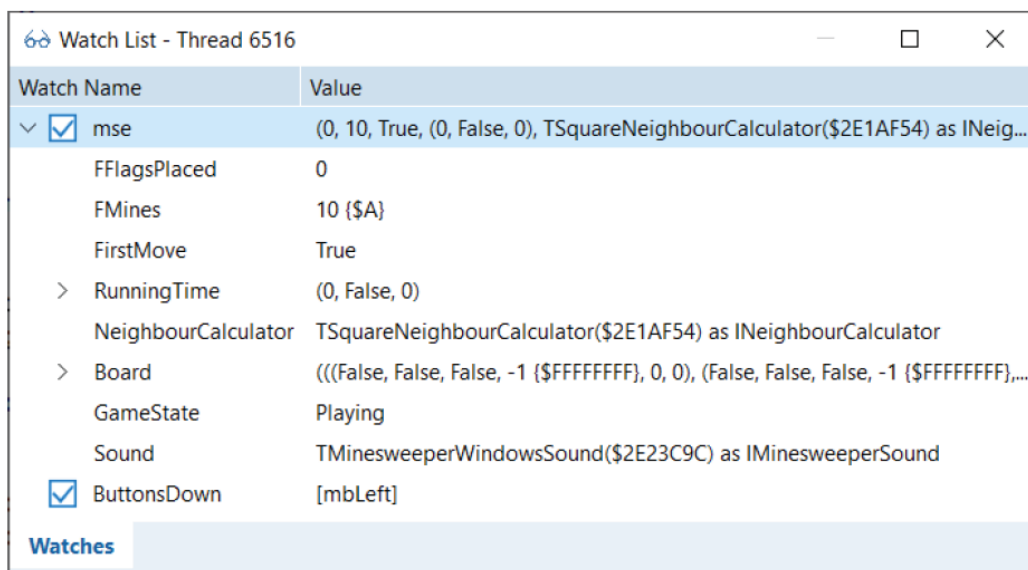
<https://leanpub.com/codebetterindelphi>



View and **Edit Source** (*Ctrl+V* and *Ctrl+S*) will show you the source code in the code editor, **View** will keep focus on the Call Stack, and **Edit** will shift it to the editor. **View Locals** (*Ctrl+L*) will select the method from the call stack list in the **Local Variables** window. **Toggle Breakpoint** (*F5*) is the same as clicking the blue dot.

Often when an exception occurs, it will happen within library code (VCL, RTL or FMX), but the exception's cause will be somewhere in your code. This means the source file you will initially see differs from where you want to be. Scan down the call stack until you see the first file you are responsible for, and *Double-click* on it to start looking for the cause of the exception there.

Watch List



The **Watch List** (**View|Debug Windows|Watches** *Ctrl+Alt+W*) allows you to specify various things you want evaluated while using the debugger. The check box indicates if the watch is enabled or disabled, and you can click to toggle between the two.

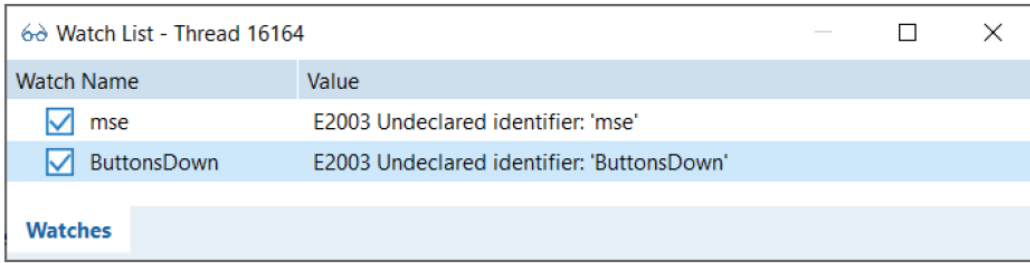
You can add a watch in many ways. I often drag something I want to monitor directly from the code editor, this is especially helpful for an expression. You can add one by pressing *Ctrl+F5* in the editor (Add watch at Cursor). You can also add one from a *Right-click* in the **Local Variables** window or manually from a *Right-click* in the **Watch List**.

Items are evaluated in the context of the current execution point. The watches might not be evaluated if you are stopped at a breakpoint in a different area of your code.

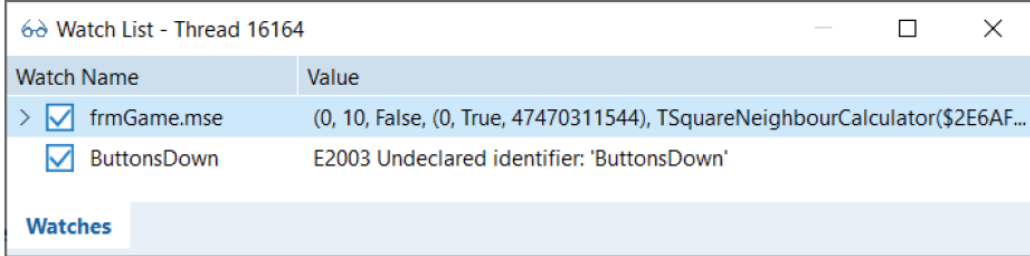
<https://leanpub.com/codebetterindelphi>



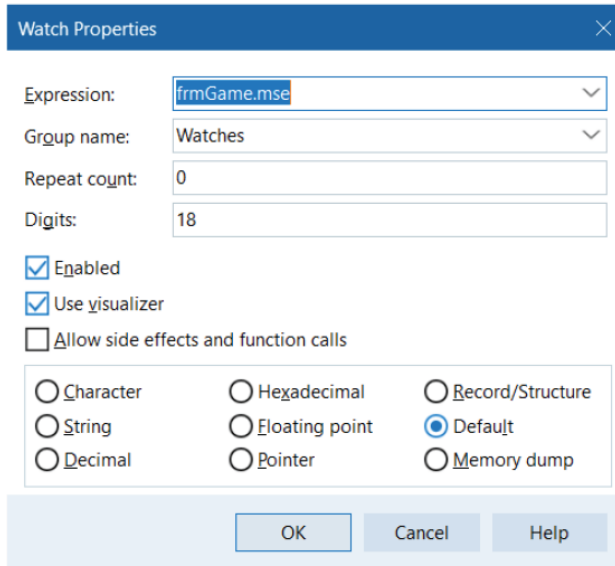
10000110



You can view the watch from anywhere in your application using a fully qualified name.



Double-click on a watch (or add a watch manually), and it will show you the **Watch Properties** window.



There are several options here. The **Use visualizer** option enables a much nicer visualisation of expressions for certain types like `TStringList` and `TDateTime`. **Allow side effects and function calls** can be quite interesting. Mostly you will enable it to see the result of a function. However, you can have an expression such as `MyStringList.SaveToFile('c:\temp\list.txt')`. This has the interesting side effect of saving a copy of `MyStringList` to a file (every time it's evaluated) that y



<https://leanpub.com/codebetterindelphi>



If you want to show only the assembly code in the Disassembly window, you can *Right-click* and untick **Show Addresses** (*Ctrl+D*), **Show Op codes|Never** (*Ctrl+E*), and potentially untick **Mixed Source** (*Ctrl+X*). However, copying and pasting that assembly code into an ASM block in your application is well outside the scope of this book (it's more of an anti-maintainability operation).

Using the Debugger:

The debugger in Delphi is very helpful, and I've included some interesting techniques for using it beyond the usual inspecting and stepping.

Simultaneous Debugging

Sometimes, it is handy to be able to debug multiple applications at the same time. This is especially true of client-server applications where you will want to debug both the request and response from both sides. You can do this by having multiple copies of Delphi open (one debugging the client and the other the server, both of which could be on separate machines) or using a project group.

Within your project group, you can set multiple breakpoints in different applications. You can't compile an application while debugging, so you must precompile all the applications. To do this, *Right-click* on your project group and select **Compile All**. You can then run multiple applications (you can *Right-click* on each in the project window and select **Run**).

Breakpoints and exceptions will now work for all the applications that you are running.

Work Out the Pass Count

You might have a loop from which an exception is being raised, and you want to know how many times through the loop before that exception is raised. This is pretty easy on a regular `for` loop as you can inspect the iteration variable. It might not be as easy for a `while`, `repeat`, or a `for..in` loop. Rather than modifying the code, you can use the pass count on a breakpoint to work this out.

Take the following highly contrived code:

```
var
  MyArray : TArray<integer>;
  z : double;
begin
  SetLength(MyArray, 1000);
  for var i := 0 to Length(MyArray)-1 do
    MyArray[i] := 1;
```



<https://leanpub.com/codebetterindelphi>





```
function CheckPrime(APrimeCandidate, ATestIndex: Integer): Boolean;
begin
  Result := True;
  if ATestIndex >= Length(FoundPrimes) then
    exit;
  Result := APrimeCandidate mod FoundPrimes[ATestIndex] = 0;
  if not Result then
    exit;
  Result := (APrimeCandidate mod FoundPrimes[ATestIndex] < 0);
end;

procedure FindPrimes(AMaxNum: integer);
var
  i: Integer;
begin
  for i := 2 to AMaxNum do
    if CheckPrime(i, 0) then
      AddPrime(i);
  end;
end;
```

- AI-enabled brain scanner reads thoughts
- Image Classifier with loading and testing a pre-trained model
- Delphi Community version for Delphi 11
- Jim McKeeth leaving Embarcadero/Delphi
- The Number guessing project
- BLAISE PASCAL MAGAZINE LIBRARY By internet and on USB Stick
- The Library kit for BPM has been extended with new features:
Search over ALL 111 issues and per issue.*
- Lazarus compiling Delphi code
- Lazarus for Visual Studio
- Debugging with the new debugger in Lazarus - lessons part 2
- FastReport for Lazarus on LINUX
in a Trial and as Professional version

SUBSCRIPTION FOR 2 YEAR BLAISE PASCAL MAGAZINE ONLY €120 ex Vat



Neuling Experte

KURZFASSUNG

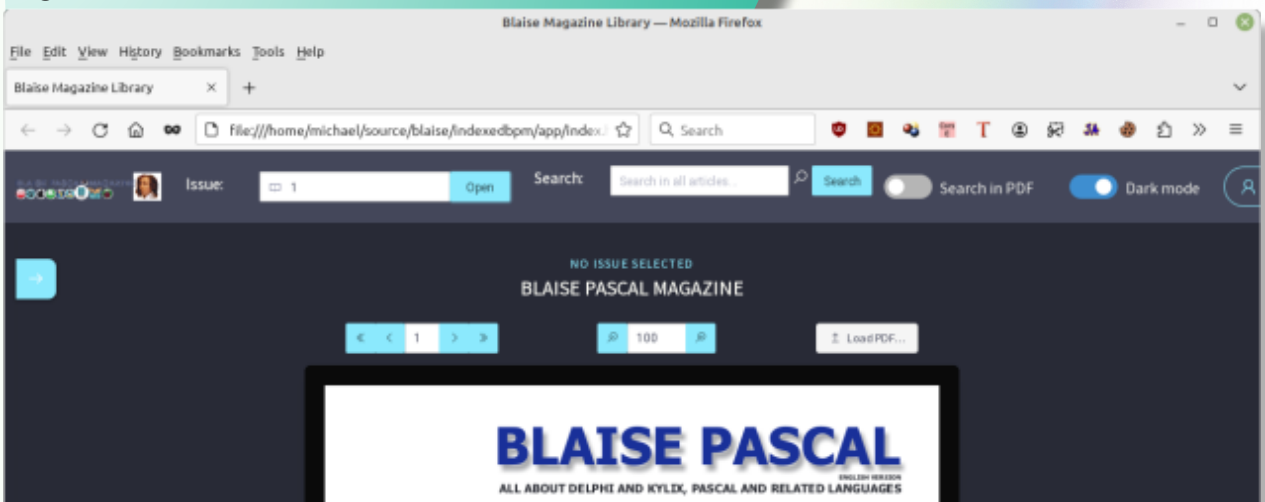
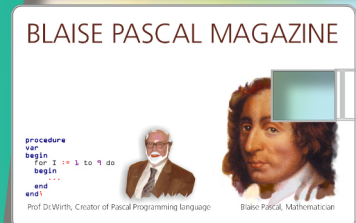
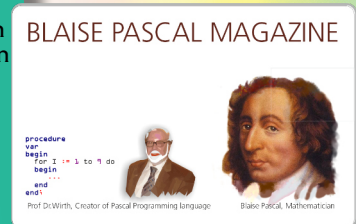
Das Blaise Pascal Magazine bietet seinen Abonnenten eine Bibliothek: eine Sammlung aller bisher erschienenen Ausgaben. In diesem Artikel zeigen wir, wie die PDF-Indexer-Anwendung, die in den vorangegangenen Artikeln über die Indizierung von PDF-Dateien vorgestellt wurde, verwendet wird, um die Bibliothek des **Blaise Pascal Magazine** neu zu schreiben und zu erweitern.

1 EINLEITUNG

In mehreren früheren Beiträgen haben wir gezeigt, wie Sie eine PDF-Datei in einer Offline-PAS2JS-Anwendung anzeigen und wie Sie PDF-Dateien indizieren und diesen Index zum Suchen und Herunterladen von PDF-Dateien verwenden können. In diesem Artikel zeigen wir wie Sie all diese Techniken kombinieren können, um die Blaise Pascal Magazine Bibliothek in eine Pas2js Anwendung umzuschreiben, die sowohl offline und online funktioniert. Die neue Ausgabe der Blaise Pascal Magazine Bibliothek muss die folgenden Funktionen haben:

- Es muss als **lokale Anwendung** funktionieren, die auf einem **USB-Stick** verteilt wird.
- Sie muss als **Webanwendung** funktionieren, auf der **Website des Blaise Pascal Magazines**.
- Wenn Sie lokal arbeiten, müssen die Hefte von der lokalen Festplatte geladen werden: in der Regel ein **USB-Stick**.
- Die Hefte müssen **durchsuchbar** sein.
- Wenn Sie lokal arbeiten und keine Internetverbindung verfügbar ist, dann muss eine (*begrenzte*) **Suche** lokal in der Liste der Artikel durchgeführt werden.
Wenn eine Internetverbindung verfügbar ist, muss die Anwendung in der Lage sein, **global zu suchen** und bei Bedarf ein **PDF Heft herunterzuladen**.
- **PDF-Downloads** sind auf die **Downloads** beschränkt, die der Abonnent bei der Zeitschrift erworben hat.
- Es wird das **Heft angezeigt**.

Alle Techniken, die erforderlich sind, um diese Anforderungen zu erfüllen, wurden bereits in früheren Artikeln vorgestellt. In diesem Artikel führen wir alles zusammen: Dies erfordert eine Überarbeitung des Codes, der in den vorherigen Artikeln vorgestellt wurde. Da wir einen **Server**- und einen **Client-Teil** benötigen, beginnen wir mit den Codeänderungen, die auf dem Server vorgenommen werden müssen.





② ZUFÜGEN VON SICHERHEIT AUF DEM SERVER

In früheren Artikeln über die Indizierung von PDF-Dateien haben wir zusätzlich zum Suchmechanismus und zur Wortliste einen einfachen Mechanismus zum Herunterladen von Heften vom Server implementiert.

Alle Hefte konnten vom Server heruntergeladen werden.

Dies muss geändert werden, damit der Benutzer nur die Ausgaben herunterladen kann, für die er ein Abonnement abgeschlossen hat.

Das bedeutet, dass wir einen Anmeldemechanismus hinzufügen müssen:

Der Server muss wissen, wer versucht, ein Heft herunterzuladen.

Außerdem brauchen wir einen Mechanismus, um festzustellen, welche Hefte ein Benutzer herunterladen darf. Um dies zu implementieren, müssen wir die Datenbank um diese Informationen erweitern.

Zunächst benötigen wir eine Liste der Benutzer: eine Tabelle mit mindestens einem Benutzernamen und einem Passwort. Das SQL zum Erstellen einer solchen Tabelle (*nennen wir sie Benutzer*) kann in Postgres zum Beispiel so aussehen:

```
create sequence seq_users;

create table users (
  u_id bigint not null default nextval('seq_users'),
  u_firstname varchar(50) NOT NULL,
  u_lastname varchar(50) NOT NULL,
  u_login varchar(127) not null,
  u_password varchar(127) not null,
  constraint pk_users primary key (u_id)
);

create unique index idx_users on users (u_login);
```

Die Feldnamen sprechen für sich selbst, und der Index sorgt dafür, dass jede Anmeldung eindeutig ist.

In ähnlicher Weise benötigen wir eine Tabelle mit allen verfügbaren Heften. Wir könnten die in den vorangegangenen Artikeln vorgestellte Tabelle `articles` nehmen, aber jedes Heft taucht mehr als einmal in dieser Tabelle auf, so dass es schwierig ist, einen Fremdschlüssel auf dieser Tabelle für die referentielle Integrität zu erstellen.

Stattdessen erstellen wir eine neue Tabelle mit dem treffenden Namen "issues" Hefte:

```
create sequence seq_issues;

create table issues (
  i_id bigint not null default nextval('seq_issues'),
  i_issue varchar(10) NOT NULL,
  i_filename varchar(127) NOT NULL,
  constraint pk_issues primary key (i_id)
);
```

Das Feld `I_issue` ist keine Zahl, die für doppelte Hefte geeignet ist, daher können wir eine Notation wie `81 82` für die kombinierten Hefte 81 und 82 unterstützen. Um zu wissen, auf welche Hefte ein bestimmter Benutzer zugreifen kann, benötigen wir eine dritte Tabelle (*genannt userissues*):

```
create sequence seq_userissues;
create table userissues (
  ui_id bigint not null default nextval('seq_userissues'),
  ui_user_fk bigint NOT NULL,
  ui_issue_fk bigint NOT NULL,
  constraint pk_userissues primary key (ui_id)
);
```





Für jeden Benutzer und jedes Heft, auf das der Benutzer zugreifen kann, wird ein Datensatz in diese Tabelle eingefügt, der mit der Tabelle der Benutzer und der Tabelle der Hefte verknüpft ist.

Für die referenzielle Integrität erzwingen wir einen Fremdschlüssel zu diesen Tabellen;

```
alter table userissues add constraint fk_userissues_users
foreign key (ui_user_fk) references users(u_id) on delete cascade;
alter table userissues add constraint fk_userissues_issues
foreign key (ui_issue_fk) references issues(i_id) on delete cascade;
```

Außerdem stellen wir sicher, dass es für jede Kombination aus Benutzer und Heft nur einen Datensatz gibt.

```
create unique index idx_userissue on userissues (ui_user_fk,ui_issue_fk);
```

Mit diesen Tabellen bewaffnet können wir nun einige Sicherheitsmechanismen implementieren. Wir werden nicht beschreiben, wie die Daten in diese Tabellen gelangen: Wir gehen davon aus, dass die Tabellen über einen externen Mechanismus gefüllt wurden, über eine Verbindung mit dem Abonnement-System.

Als erstes müssen wir eine Art Login-Routine implementieren:

Wir implementieren einen einfachen HTTP-Endpunkt, der ein JSON mit einem Benutzernamen und einem Passwort empfängt (*dies ist kein JSON-RPC-Mechanismus*):

```
{
  "username": "michael",
  "password": "verysecret"
}
```

Der Server antwortet mit einem Token (einer einfachen GUID):

```
{
  "token" : "{F5A07A5B-5184-4256-8EE6-20E2DE987AF5}",
  "expires" : "2023-06-03T17:12:09.489Z"
}
```

Dieses Token kann an den Server weitergegeben werden, wenn eine PDF-Datei heruntergeladen wird: Der Server prüft dann dieses Token und erlaubt den Download, wenn es gültig ist.

Die Token werden auch in der Datenbank gespeichert, in einer Tabelle namens Tokens:

```
CREATE TABLE tokens
(
  tk_id bigint NOT NULL DEFAULT nextval('seqTokens'::regclass),
  tk_token character varying(38) NOT NULL
  DEFAULT (upper((' '::text || uuid_generate_v4() ||
  ' '::text))::character varying(38)
  tk_user_fk bigint NOT NULL,
  tk_expires_on timestamp without time zone NOT NULL
  DEFAULT (now() + '00:30:00'::interval),
  CONSTRAINT pktokens PRIMARY KEY (tk_id)
);

create index idx_token_expires on tokens(tk_expires_on);
create unique index idx_tokens on tokens(tk_token);
```

Das Feld `tk_expires_on` wird standardmäßig mit einem Zeitstempel gefüllt, der 30 Minuten nach dem Zeitpunkt des Einfügens des Datensatzes liegt. Das Token ist also 30 Minuten lang gültig.

Die Funktion `uuid_generate_v4` ist Teil einer Postgres-Erweiterung und generiert eine GUID, die als unser eindeutiges Session-Token dient.

Die Erweiterung muss mit der folgenden SQL-Anweisung aktiviert werden:





```
CREATE EXTENSION IF NOT EXISTS "uuid-osspl";
```

Um dieses Schema zu implementieren, erstellen wir eine neue Klasse TSessionManager:

```
TSessionManager = Class(TComponent)  
Public  
  constructor Create(aOwner : TComponent); override;  
  destructor destroy; override;  
  // Public API  
  Procedure GetToken(aRequest : TRequest; aResponse : TResponse);  
  Function CheckToken(const aToken : string) : int64;  
  function CheckFileAllowed(aUserID : int64; const aFileName : string) : Boolean;  
  Property DB : TSQLConnection Read FDB Write FDB;  
end;
```

Der GetToken-Aufruf wird beim Start der Anwendung als Handler für die /token-Anmeldung registriert:

```
aSession:=TSessionmanager.Create( Nil );  
aSession.DB:=aSearch.Connection;  
HTTPRouter.RegisterRoute('/token', @aSession.GetToken, False);
```

Der GetToken-Aufruf wird beim Start der Anwendung als Handler für die /token-Anmeldung registriert:

```
aSession:=TSessionmanager.Create( Nil ); aSession.DB:=aSearch.Connection;  
HTTPRouter.RegisterRoute('/token', @aSession.GetToken, False);
```

Die Eigenschaft DB ist die von der Suchklasse verwendete SQL-Verbindung:
Sie wurde im Code der vorherigen Artikel eingerichtet und wird hier einfach wiederverwendet.

Der Aufruf von GetToken ist ziemlich einfach.
Er beginnt mit der Überprüfung auf eine CORS-Anfrage, ähnlich wie beim Download der PDF-Datei im vorherigen Artikel.
Dies ist erforderlich, da sich der Ursprung vom Server unterscheidet, wenn die Client-Anwendung von einer lokalen Festplatte aus gestartet wird.

Wenn die CORS-Anfrage anzeigt, dass alles in Ordnung ist, beginnt der Code mit der Dekodierung der Nutzdaten der Anfrage als JSON-Struktur.

Wenn etwas schief geht, wird die Routine ReportInvalidParam verwendet, um einen HTTP 400 Rückgabewert zu melden.
Wenn die JSON-Struktur korrekt dekodiert wurde, werden der Benutzername und das Kennwort extrahiert.
Auch hier wird geprüft, ob Werte für den Benutzernamen und das Kennwort übergeben wurden.
Wenn nicht, wird wieder ein HTTP 400-Fehler gemeldet:





```

procedure TSessionManager.GetToken(aRequest: TRequest; aResponse: TResponse);
Var Req, Resp : TJSONData;
    Obj : TJSONObject absolute Req;
    token,UserName, UserPW : String;
    aExpiresOn : TDatetime;
begin
    if FCors.HandleRequest(aRequest,aResponse,[hcDetect,hcSend]) then exit;
    Req:=Nil;
    Resp:=Nil;
    try Req:=GetJSON(aRequest.Content);
        if not (Req is TJSONObject) then
            ReportInvalidParam(aResponse)
        else
            begin
                userName:=Obj.Get('username','');
                userPW:=Obj.Get('password','');
                if (UserName='') or (userPW='') then
                    ReportInvalidParam(aResponse)
                else
                    begin token:=ValidateUser(UserName,userPW,aExpiresOn);
                        if Token="" then
                            ReportForbidden(aResponse)
                        else
                            begin
                                Resp:=TJSONObject.Create([
                                    'token',token,
                                    'expires',DateToISO8601(aExpiresOn)
                                ]);
                                SendJSON(aResponse,200,'OK',Resp);
                            end;
                        end;
                    end;
                except
                    on E : Exception do
                        ReportException(aResponse,E);
                    end;
                Resp.Free;
                Req.Free;
            end;
    end;

```

Der Benutzername und das Passwort werden mit dem Aufruf ValidateUser überprüft: Er gibt ein Token zurück, wenn das Paar Benutzername/Kennwort gültig war. Wenn das Token leer ist, bedeutet dies, dass die Kombination nicht gültig war und ein Fehler gemeldet wird. Wenn wir schließlich ein Token und ein Ablaufdatum erhalten haben, senden wir beides mit SendJSON in einer JSON-Struktur an den Client zurück:

```

procedure TSessionManager.SendJSON(aResponse : TResponse;aCode : Integer; aText : String; aJS
begin
    if aResponse.ContentSent then
        exit;
    aResponse.Code:=aCode;
    aResponse.CodeText:=aText;
    aResponse.ContentType:='application/json';
    aResponse.Content:=aJSON.FormatJSON();
    aResponse.SendContent;
end;

```

Die gleiche SendJSON-Methode wird z.B. in der ReportInvalidParam-Methode verwendet:

```

procedure TSessionManager.ReportInvalidparam(aResponse: TResponse); Var J : TJSONObject;
begin
    J:=TJSONObject.Create(['message','need username /password']);
    try
        SendJSON(aResponse,400,'INVALID PARAM',J);
    finally
        J.Free;
    end;
end;

```





Die `ValidateUser`-Methode ist wiederum sehr einfach: Das einzig Bemerkenswerte ist, dass das `Password` verschlüsselt in der Datenbank gespeichert wird. Dazu verwenden wir die nativen kryptografischen Mechanismen der PostgreSQL-Datenbank: Die Funktion `crypt` verschlüsselt einen Wert mit einem `Salt`, und dieselbe Funktion kann auch zum Einfügen der Daten verwendet werden. Die Krypto-Funktionalität muss mit der folgenden SQL-Anweisung aktiviert werden:

```
CREATE EXTENSION pgcrypto;
```

Mit dieser Funktion sieht die SQL-Anweisung zur Überprüfung eines Benutzerkennworts wie folgt aus:

```
SELECT
    U_password=crypt(:password,U_password) as PasswordOK, *
from
    Users where
        (U_login=:login);
```

Wenn die Benutzeranmeldung nicht gefunden wird, gibt die Abfrage keine Datensätze zurück. Wenn der Benutzer gefunden wird, gibt es einen einzigen Datensatz (*weil die Anmeldung eindeutig ist*) und das Feld `PasswordOK` ist `True`, wenn das im Parameter: `password` übergebene Passwort mit dem in der Datenbank gespeicherten übereinstimmt, und andernfalls ist es `False`.

Mit dieser SQL-Anweisung können wir ganz einfach den Aufruf von `ValidateUser` erstellen. Sie beginnt mit der Erstellung einer Datenbanktransaktion: Jeder Vorgang wird in einer eigenen Transaktion ausgeführt.

Nachdem die Transaktion erstellt wurde, wird sie verwendet, um einen `TSQLQuery`-Datensatz zu erstellen und die SQL-Anweisung auszuführen. Die Funktionen `CreateTransaction` und `CreateQuery` sind trivial und werden hier nicht vorgestellt:

```
function TSessionManager.ValidateUser(const aUser, aPassword: String;
out aExpires: TDateTime): String;
```

Const

```
    SQLSelectUser =
    'SELECT U_password=crypt(:password,U_password) as PasswordOK,*' +
    'from Users '+
    'where (U_login=:login)';
```

Var

```
    Tr : TSQLTransaction;
    Qry : TSQLQuery; Res,OK : Boolean;
    aID : Int64;
```

begin

```
    OK:=False;
    Result:=""; qry:=Nil;
    Tr:=CreateTransaction;
    try Qry:=CreateQuery(SQLSelectUser,['LOGIN',aUser,'PASSWORD',aPassword],Tr);
        Qry.Open;
        // If we have a user and the password matches
        Res:=(Not Qry.IsEmpty) and (Qry.FieldName('PasswordOK').AsBoolean);
        aID:=Qry.FieldName('u_id').AsLargeInt;
        if Res then
            // We get a token
            Result:=CreateToken(aID,aExpires,Qry.SQLTransaction);
            Tr.Commit;
            OK:=True;
```

finally

```
    if not OK then
        Tr.Rollback;
        ReleaseQuery(Qry,Tr);
```

```
end;
end;
```





Wenn der Benutzer verifiziert ist, werden die Benutzer-ID und die Transaktion an `CreateToken` übergeben, das einen neuen Token erstellt. Beachten Sie, dass der Token in der gleichen Transaktion erstellt wird: Die Funktion `CreateToken` ist wiederum recht einfach. Sie macht sich die Tatsache zunutze, dass die 'Standardwerte' in den Tabellenspalten-Definitionen brauchbare Werte erzeugen, und gibt einfach die von Postgres erstellten Werte zurück.

```
function TSessionManager.CreateToken(aUser: Int64; out Expires: TDateTime;
    aTransaction : TSQLTransaction): String;

Const
    SQLInsert =
    ' insert into tokens (tk_user_fk) values (:USER) ' +
    ' returning tk_token, tk_expires_on';

Var
    Qry : TSQLQuery; OK : Boolean;

begin OK:=False;
    Qry:=CreateQuery(SQLInsert,['USER',aUser],aTransaction);
    try
        Qry.Open;
        if Qry.IsEmpty then
            DatabaseError(SErrFailedToCreateToken, self);
            Result:=Qry.FieldName('tk_token').AsString;
            Expires:=Qry.FieldName('tk_expires_on').AsDateTime; OK:=True;
        finally
            if Not OK then
                Qry.SQLTransaction.Rollback; ReleaseQuery(Qry);
        end;
    end;
end
```

Mit diesen Routinen haben wir einen HTTP-Endpoint erstellt, der in der Anwendung verwendet werden kann, um nach einem Token zu fragen.

3 SECURING THE DOWNLOAD

Wenn der Benutzer eine PDF-Datei herunterladen möchte, muss das Token angegeben werden, damit der Server überprüfen kann, wer den Download durchführt und ob der Benutzer berechtigt ist, die angeforderte PDF-Datei herunterzuladen. Das Token kann auf eine von 2 Arten angegeben werden:

- Als URL-Abfrageparameter, genannt 'token':

```
http://localhost:3010/pdf/BlaisePascalMagazine_61_UK.pdf?
token=%7BF5A07A5B-5184-4256-8EE
```

- Als HTTP-Header, genannt 'X-Access-Token':

```
X-Access-Token: {F5A07A5B-5184-4256-8EE6-20E2DE987AF5}
```

Das bedeutet, dass wir das Download-Modul so anpassen müssen, dass es zuerst das Token überprüft und dann prüft, ob der Benutzer, der das Token besitzt, die Datei herunterladen kann. Die Änderung ist trivial:





```
procedure TCorsFileModule.HandleRequest(ARequest: TRequest;
    AResponse: TResponse);
begin
    Cors.Enabled:=true;
    if Cors.HandleRequest(aRequest,aResponse) then exit;
    if not CheckToken(aRequest,aResponse) then exit;
    inherited HandleRequest(ARequest, AResponse);
end;
```

Die Funktion `CheckToken` erledigt die eigentliche Arbeit. Sie verwendet die Funktion `CheckToken` aus der Klasse `TSessionManager`, um das Token zu überprüfen. Wenn das Token OK ist, wird die Benutzer-ID zurückgegeben, wenn das Token nicht OK ist, wird -1 zurückgegeben. Die zurückgegebene Benutzer-ID wird dann verwendet, um zu prüfen, ob der Benutzer die angeforderte PDF-Datei herunterladen darf (die Funktion `GetRequestFileName` ist eine Methode des `Dateidownload-Datenmoduls`, das mit FPC geliefert wird):

```
function TCorsFileModule.CheckToken(ARequest: TRequest; AResponse: TResponse): Boolean;
var
    aToken,aFileName : String;
    aID : int64;
begin
    // Check URL parameter and HTTP header for token.
    aToken:=aRequest.QueryFields.Values["token"];
    if aToken="" then
        aToken:=aRequest.CustomHeaders.Values['x-access-token']; Result:=(aToken<>"");
    if Result then
        begin
            // Check token in database
            aID:=aSession.CheckToken(aToken); Result:=aID<>-1;
            if Result then
                begin
                    // Get requested filename.
                    aFileName:=ExtractFileName(GetRequestFileName(aRequest));
                    // Check if user is allowed to download this file.
                    Result:=aSession.CheckFileAllowed(aID,aFileName)
                end;
            end;
        end;
    if not Result then
        begin
            aResponse.Code:=403;
            aResponse.CodeText:='FORBIDDEN';
            aResponse.SendContent;
        end;
    end;
```

Beachten Sie, dass, wenn das Token ungültig ist oder der Benutzer das PDF nicht herunterladen darf, ein **403 FORBIDDEN** HTTP-Rückgabecode an den Browser gesendet wird. Die Funktion `CheckToken` des `Sitzungsmanagers` führt die eigentliche Prüfung des Tokens durch. Sie ist wiederum recht einfach:

```
function TSessionManager.CheckToken(const aToken: string): int64;
const
    SQLSelect = 'select tk_user_fk,tk_expires_on from tokens where (tk_token=:token)';
var
    Tr : TSQLTransaction;
    Qry : TSQLQuery;
    OK : Boolean;
begin
    OK:=False;
    Qry:=nil; Result:=-1;
    TR:=CreateTransaction;

    try Qry:=CreateQuery(SQLSelect,['token',aToken],Tr); Qry.Open;
        if Not Qry.IsEmpty and (Qry.FieldByName('tk_expires_on').asDateTime>Now) then
            Result:=Qry.FieldByName('tk_user_fk').asLargeInt;
            if Result<>-1 then
                UpdateToken(aToken,Tr);
        finally
            if not OK then TR.Rollback; ReleaseQuery(Qry,Tr);
        end;
    end;
```





Wenn der Token noch nicht abgelaufen ist, wird er mit dem UpdateToken um 30 Minuten verlängert:

```
function TSessionManager.UpdateToken(const aToken: String;
    aTrans: TSQLTransaction): TDateTime;

Const
    SQLUpdate =
        'update tokens set ' +
        'tk_expires_on = clock_timestamp() + interval "30 minutes" '+
        'where (tk_token=:TOKEN) returning tk_vervalt_op;';

Var
    Qry : TSQLQuery;

begin
    Qry:=CreateQuery(SQLUpdate,['TOKEN',aToken],aTrans);
    try
        Qry.Open;
        if not Qry.IsEmpty then
            Result:=Qry.Fields[0].AsDateTime
        else Result:=0;
    finally
        ReleaseQuery(Qry);
    end;
end;
```

Damit soll vermieden werden, dass sich der Benutzer alle 30 Minuten anmelden muss, aber nach mehr als 30 Minuten Inaktivität läuft der Token ab.

Der obige Mechanismus ist ein einfacher, in der Praxis können auch fortgeschrittenere Strategien verwendet werden.

Schließlich wird mit dem Aufruf CheckFileAllowed geprüft, ob der Benutzer berechtigt ist, die angeforderte PDF-Datei herunterzuladen.

Dies geschieht anhand der Tabelle userissues:

Wenn ein Datensatz für das angeforderte Heft und den Benutzer vorhanden ist, darf der Benutzer die PDF-Datei herunterladen. Die Prüfung ist dann sehr einfach:

```
function TSessionManager.CheckFileAllowed(aUserID: int64;
    const aFileName: string): Boolean;

Const
    SQLSelect =
        'select '+' ui_id '+' from '+' userissues '+'
        ' inner join issues on (i_id=ui_issue_fk) '+' where '+'
        '( i_filename=:filename) '+' and (ui_user_fk=:uid)';

var
    Tr : TSQLTransaction;
    Qry : TSQLQuery;

begin
    Tr:=CreateTransaction;
    try
        Qry:=CreateQuery(SQLSelect,['uid',aUserID,
            'filename',lowercase(aFileName)],Tr);
        Qry.Open;
        Result:=Not Qry.IsEmpty;
    finally
        ReleaseQuery(Qry,Tr);
    end;
end;
```





4 ERMÖGLICHT DAS HERUNTERLADEN EINES HEFTES NACH NUMMER

Eine der Anforderungen war, dass der Benutzer ein Heft herunterladen und ansehen kann, indem er die Nummer des Heftes eingibt. Um dies einem PDF-Dateinamen zuzuordnen, ist eine Routine erforderlich, die die Tabelle der Hefte überprüft und den entsprechenden Dateinamen zurückgibt. Um dies zu implementieren, lassen wir die Anwendung, wenn sie ein Heft anzeigen muss, eine PDF-Datei mit einer speziellen URL herunterladen:

```
http://localhost:3010/issue/45
```

In der obigen URL muss die Zahl 45 durch das tatsächliche Heft ersetzt werden. Um dies auf dem Server zu codieren, müssen wir einen Handler für die obige URL implementieren. Wir registrieren ihn beim HTTP-Router wie folgt:

```
HTTPRouter.RegisterRoute('/issue/:Issue', @IssueToPDF);
```

Die IssueToPDF ist eine einfache Routine. Der Suchmechanismus hat die Liste der Artikel eines Heftes in einem Array im Speicher. Auf diese Liste kann zugegriffen werden, um den Dateinamen des Heftes zu finden.

Wenn ein Dateiname gefunden wird, wird die Datei nicht gesendet, sondern eine Umleitungsantwort an den Browser mit dem 'normalen' PDF-Downloadort gesendet: Die Redirect-Antwort bedeutet, dass ein **307 HTTP-Returncode** gesendet wird und der Speicherort der PDF-Datei im Location HTTP-Header zurückgegeben wird.

Die Antwort des Servers sieht in etwa so aus:

```
HTTP/1.1307 Temporary Redirect
Location: /pdf/BlaisePascalMagazine_45_46_UK.pdf
```

Upon receiving the 307 return code, the browser will immediately do a second request to the new location. To the user, this is transparent.

To code this is quite simple:

```
Procedure IssueToPDF(ARequest: TRequest; AResponse: TResponse);
```

```
var Cors : TCORSSupport;
    PDF : String;

begin
    Cors:=TCORSSupport.Create;
    try
        Cors.Enabled:=true;
        if Cors.HandleRequest(aRequest,aResponse) then exit;
    finally
        Cors.Free;
    end;
    PDF:=aSearch.IssueToPDF(aRequest.RouteParams['Issue']);
    if (PDF<>'') then
        aResponse.SendRedirect('/pdf/'+PDF)
    else
        begin aResponse.Code:=404;
            aResponse.CodeText:='Not Found';
        end;
    aResponse.SendContent;
end;
```

Damit haben wir die Erweiterung des Servers abgeschlossen. Wir können uns nun den Änderungen in der Client-Anwendung **PAS2JS** zuwenden.

server   server





5 SUCHEN IM KUNDEN - zweiter Teil

In den vorangegangenen Artikeln, in denen die Suche in einer PDF-Datei in einer Browser-Anwendung behandelt wurde, wurden 3 Mechanismen behandelt, bei denen eine Suche an 3 verschiedenen Stellen durchgeführt wurde:

- In der angezeigten PDF-Datei, unter Verwendung der Mechanismen, die das pdf.js-Paket im Browser bereitgestellten Mechanismen.
- In einer Artikelliste, unter Verwendung einer speicherinternen Kopie der Artikelliste.
- In einer Datenbank, die mit einem PDF-Indexer erstellt wurde.

Wenn unsere Anwendung **online und offline funktionieren soll**, müssen wir überlegen, ob jeder der Mechanismen nutzbar ist:

Die Suche in der angezeigten PDF-Datei ist natürlich immer möglich. Wenn Sie offline sind, ist die Suche in der Datenbank nicht verfügbar und das Beste, was wir tun können, ist, sie transparent durch eine Suche in der Liste der Artikel zu ersetzen.

Zu diesem Zweck müssen wir den Suchmechanismus anpassen: In unserer letzten Iteration der PDF-Anwendung wurden die Suchalgorithmen (PDF und Datenbank) durch das `TPDFSearchControl` verwaltet. Jetzt müssen wir die Suche in der Liste der Artikel hinzufügen (*wie im ersten Artikel über die Anzeige einer PDF demonstriert*) zu dieser Klasse hinzufügen.

Um den Code einfach zu halten, werden wir die Suchmechanismen in separate Klassen aufteilen. Das `TPDFSearchControl` wählt dann abhängig von der Benutzereinstellung und dem Online-/Offline-Status des Browsers einen Suchmechanismus aus.

Die 3 Suchklassen sind für die Suche und die Anzeige der Ergebnisse unterhalb eines bestimmten HTML-Tags zuständig. Wenn der Benutzer ein Ergebnis auswählt, wird ein spezielles Ereignis ausgelöst, das genügend Informationen enthält, um das ausgewählte Ergebnis anzuzeigen.

Das `TPDFSearchControl` tut dann das Nötige, um die PDF-Datei anzuzeigen und zu der Seite zu springen, die das Ergebnis enthält.

Die Suchklasse ist auch für die Rückgabe einer Liste von Wörtern für die Autovervollständigung im Suchfeld verantwortlich:

Der Mechanismus, der für die Suche in der indizierten Datenbank entwickelt wurde, hat eine Implementierung.

Für die Liste der Artikel kann eine Liste von Wörtern *on-the-fly* erstellt werden, und auch eine Liste von Wörtern in der aktuellen PDF-Datei kann erstellt werden.

Da die drei Mechanismen die gleichen Funktionen ausführen müssen, definieren wir die folgende Schnittstelle, um die Anforderungen zu kapseln:
→ *nächste Seite*





```

TPageInfo = record
    Issue, Title, FileName : String; Page: Integer;
    useIssue : Boolean;
end;

TShowPDFPageEvent = procedure(aPage : TPageInfo) of object;
TWordListCallBack = reference to procedure(List : TStrings);

{ ISearchEngine }

ISearchEngine = Interface
    // Property getters & setters
    function GetOnShowResultPanel: TNotifyEvent;
    procedure SetOnShowResultPanel(AValue: TNotifyEvent);
    procedure SetResultsElement(aValue : TJSHTMLElement);
    function GetResultsElement : TJSHTMLElement;
    procedure SetShowPageEvent(aValue : TShowPDFPageEvent);
    function GetShowPageEvent : TShowPDFPageEvent;

    // Actual interface
    procedure Search(const aTerm : string; const aIssue : String);
    procedure GetWordList(aTerm : string; aOnResults : TWordListCallBack);

    // Easy access using properties
    property ResultsElement : TJSHTMLElement
        Read GetResultsElement Write SetResultsElement;
    property ShowPDFPageEvent : TShowPDFPageEvent
        Read GetShowPageEvent Write SetShowPageEvent;
    property OnShowResultPanel : TNotifyEvent
        Read GetOnShowResultPanel Write SetOnShowResultP
end;
    
```

Der Aufruf von Search zeigt die Liste der gefundenen Vorkommen des Suchbegriffs unter ResultsElement an. Das Ereignis OnShowResultPanel kann verwendet werden, um dem Aufrufer mitzuteilen, dass es Ergebnisse gab und dass das Ergebniselement angezeigt werden muss (das Ergebnispanel ist standardmäßig geschlossen, es muss geöffnet werden, wenn Ergebnisse vorliegen). Wenn der Benutzer auf ein Ergebnis klickt, wird das Ereignis ShowPDFPageEvent mit einem TPageInfo-Datensatz ausgelöst: Dieser Datensatz enthält genügend Informationen, um bei Bedarf ein PDF herunterzuladen und zur richtigen Seite zu springen. Die GetWordList-Methode ist ebenfalls eindeutig: Sie muss eine Liste von Wörtern anzeigen. Wenn eine Wortliste verfügbar ist, wird der Callback aOnResults aufgerufen, dem die Liste der Wörter übergeben wird. Ein Callback muss verwendet werden, da die Suche asynchron sein kann: Die Abfrage der Datenbank auf dem Server ist ein asynchroner Aufruf.

In der vorherigen Iteration der Anwendung zum Anzeigen und Indizieren von PDF-Dateien, enthielt das TPDFSearchControl 2 Suchmechanismen. Wir werden diese in eigene Klassen ausgliedern, so dass wir 4 Klassen haben, die zusammenarbeiten, um die Suchfunktionalität zu implementieren. Die ersten 3 Klassen sind lediglich eine Umstrukturierung der bestehenden Klassen

TPDFSearchControl

Damit wird nur die Benutzeroberfläche des Suchmechanismus verwaltet: Sie verwaltet die Buttons zum Bearbeiten und Suchen, zeigt die Wortliste zur Vervollständigung an und blendet das Ergebnisfenster ein oder aus. Die eigentliche Suche wird von den anderen 3 Komponenten durchgeführt. Wenn ein PDF angezeigt werden muss, wird ein Event-Handler aufgerufen.

TServerSearch

Implementiert die obige Schnittstelle unter Verwendung des im vorherigen Artikel beschriebenen Server-Suchmechanismus.





TPDFSearch

Implementiert die oben genannte Schnittstelle für die Suche in der angezeigten PDF-Datei. Sie verwendet den PDF-Suchmechanismus, der im ersten Artikel dieser Serie besprochen wurde.

TArticleSearcher

Implementiert die obige Schnittstelle unter Verwendung eines Suchmechanismus in einer Artikelliste, die in die Anwendung aufgenommen wird, wenn diese von der Festplatte geladen wird. Wenn die Klasse TPDFSearchControl erstellt wird, erzeugt sie Instanzen der 3 Suchmechanismen:

```

constructor TPDFSearchControl.Create(aOwner: TComponent);
begin
    Inherited;
    FLocalsearch:=TArticleSearcher.Create(Self);
    FServerSearch:=TServerSearch.Create(Self);
    FSearch:=TPDFSearch.Create(Self);
end;
    
```

Und er initialisiert sie in seiner BindElements-Methode:

```

procedure TPDFSearchControl.BindElements;
begin
    // ... other code...
    PrepareEngines(True);
end;

procedure TPDFSearchControl.PrepareEngines(Full : boolean);
begin
    PrepareEngine(FLocalsearch as ISearchEngine,Full);
    PrepareEngine(FServerSearch as ISearchEngine,Full);
    PrepareEngine(FSearch as ISearchEngine,Full);
end;

procedure TPDFSearchControl.PrepareEngine(aEngine : ISearchEngine;
    Full : Boolean);
begin
    aEngine.ShowPDFPageEvent:=FOnShowPDFPage;
    if Full then
        begin
            aEngine.OnShowResultPanel:=@HandleShowResultPanel;
            aEngine.ResultsElement:=pnlResults;
        end;
end;
    
```

Die PrepareEngine wird für alle 3 Suchmaschinen aufgerufen: Sie initialisiert die relevanten Eigenschaften, damit die Klassen ihre Arbeit tun können. FOnShowPDFPage ist ein Event-Handler, der von der Hauptanwendungsklasse gesetzt wird: Die Hauptanwendungsklasse ist für das Laden einer PDF-Datei verantwortlich. Der 'Klick'-Ereignishandler des Buttons Suche in TPDFSearchControl ist jetzt ganz einfach:

```

procedure TPDFSearchControl.onSearch(aEvent: TJSEvent);
var
    aTerm : string;
begin
    aTerm:=SearchTerm;
    if Length(aTerm)<=1 then exit;
    CurrentSearchEngine.Search(aTerm,FIssueFilter);
end;
    
```

Die Eigenschaft CurrentSearchEngine gibt eine ISearchEngine-Schnittstelle zurück.

Der Getter dieser Eigenschaft entscheidet anhand der Eigenschaft PDFSearch (im Grunde der Wert des Kontrollkästchens 'Search PDF') und einer Eigenschaft OffLine, welche Suchmaschine zurückgegeben werden soll:





```
function TPDFSearchControl.GetSearchEngine: ISearchEngine;
begin
  if PDFSearch then
    Result:=FSearch as ISearchEngine
  else
    if OffLine then
      Result:=FLocalsearch as ISearchEngine
    else
      Result:=FServerSearch as ISearchEngine;
end
```

Die Eigenschaft OffLine wird durch den Online- oder Offline-Status des Navigators bestimmt. Sie wird beim Starten der Anwendung festgelegt und während der gesamten Lebensdauer der Anwendung beibehalten. Wir zeigen Ihnen später, wie Sie dies tun können. Die Anwendung verfügt über eine Funktion, bei der das Bearbeitungsfeld eine Liste von Wörtern zur Vervollständigung anzeigt, die auf den Rückgaben des Servers basiert. Dieser Mechanismus muss überarbeitet werden, damit die Liste der Wörter von der aktuellen Suchmaschine geholt wird. Dieser Mechanismus wurde in einem Timer-Ereignis implementiert, das nun recht kurz wird:

```
function TPDFSearchControl.DoCompleteWord(Event: TEventListenerEvent): boolean;

  procedure DoServerSearchWord;
  begin
    if Length(edtSearch.Value)>1 then
      CurrentSearchEngine.GetWordList(edtSearch.Value,@DoShowWordList);
    end;
  end;

begin
  Result:=False;
  if FSearchTimerID<>0 then
    window.clearTimeout(FSearchTimerID);
  FSearchTimerID:=window.SetTimeout(@DoServerSearchWord,200);
end;
```

GetWordList ruft DoShowWordList auf, sobald die Liste der Wörter abgerufen wurde. Die DoShowWordList-Routine, die die Liste der Wörter anzeigt, muss nun lediglich alle Wörter in der Liste durchlaufen:

```
procedure TPDFSearchControl.DoShowWordList(List : TStrings);
Var
  S : String;
  P : TJSHTMLLElement;
  A : TJSHTMLAnchorElement;
begin
  mnuAutoComplete.style.setProperty('display','none');
  mnuAutoComplete.InnerHTML:='<div class="dropdown-content"></div>';
  P:=TJSHTMLLElement(mnuAutoComplete.firstElementChild);
  For S in List do
    begin
      a:=TJSHTMLAnchorElement(Document.createElement('a'));
      a.href:='#';
      a.classlist.Add('dropdown-item');
      a.innerText:=s; a.dataset['value']:=s;
      a.addEventListener('click',@DoWordSelected); P.appendChild(a);
    end;
  mnuAutoComplete.style.setProperty('display','block');
end;
```

Die Implementierung des Algorithmus zum Abrufen einer Wortliste wurde in die Klasse TServerSearch verschoben. Sie bleibt fast unverändert, mit der Ausnahme, dass sie eine TStringlist füllt.

Die Schnittstelle ISearchEngine enthält einige Boilerplate (*standardisierte vorbereitete*)-Codes zur Definition von 3 Eigenschaften (*sie müssen über Getter und Setter definiert werden*).

Um den Code, der in den 3 Suchklassen zur Implementierung dieser Schnittstelle benötigt wird, zu reduzieren, werden wir alle 3 Klassen von einem gemeinsamen Vorfahren ableiten: TSearchBase.

Hier ist die Definition:
Blaise Pascal Magazine 110 2023





```

TSearchBase = class(TComponent)
Private
    FShowPDFPageEvent : TShowPDFPageEvent;
    FOnShowResultPanel : TNotifyEvent;
    FResultsElement: TJSHTMLElement;
Protected
    // Property getters & setters
    Function GetOnShowResultPanel: TNotifyEvent;
    Procedure SetOnShowResultPanel(AValue: TNotifyEvent);
    Procedure SetResultsElement(aValue : TJSHTMLElement);
    Function GetResultsElement : TJSHTMLElement;
    Procedure SetShowPageEvent(aValue : TShowPDFPageEvent);
    Function GetShowPageEvent : TShowPDFPageEvent;
    // Easy access for descendents
    procedure ShowPDFPage(aInfo : TPageInfo);
    // Show results panel.
    procedure ShowResultsPanel;
    // Clear results panel.
    procedure ClearResultPanel;
    // Append a HTML node to the results panel.
    procedure AppendToResults(aElement : TJSHTMLElement);
Public
    // Easy access using properties
    Property ResultsElement : TJSHTMLElement Read GetResultsElement
                                                Write SetResultsElement;
    Property ShowPDFPageEvent : TShowPDFPageEvent Read GetShowPageEvent
                                                Write SetShowPageEvent
    Property OnShowResultPanel : TNotifyEvent Read GetOnShowResultPanel
                                                Write SetOnShowResultP
end;
    
```

Die verschiedenen Get*- und Set*-Methoden tun nichts anderes, als die Werte der privaten Felder für die Eigenschaften festzulegen und zu setzen. Die Methoden für den einfachen Zugriff sind dazu da, die Ereignishandler aufzurufen, wenn sie gesetzt wurden. Dadurch wird vermieden, dass alle Nachkommen eine Prüfung implementieren müssen. Die Klasse TServerSearch wird als Nachkomme dieser Klasse überarbeitet, enthält aber keinen neuen Code im Vergleich zur vorherigen Iteration unserer Anwendung, so dass wir ihn hier nicht wiederholen. Das Gleiche gilt für die Klasse TPDFSearch: Bei dieser Klasse ändert sich nichts, außer dass sich die Signatur der Methode etwas ändert.

6 OFFLINE ARBEITEN

Wenn wir offline arbeiten, können wir den Server nicht kontaktieren und eine Suche in einer Datenbank durchführen. Wir können die Datenbank auch nicht über den Browser verteilen und darauf zugreifen.

Was wir tun können, ist eine eingeschränkte Suche anzubieten:

Die Liste der Artikel und Hefte wird mit der Offline-Version der Anwendung verteilt.

Im Grunde erstellen wir eine Javascript-Datei, die eine 'Datenbank' mit Artikeln enthält.

Die Datenbank ist dann einfach ein Javascript-Array mit Datensätzen, das wie folgt aussieht (Formatierung zu Anzeigezwecken hinzugefügt):

```

var BPMArticles = [
{
    "i" : "1", "p" : 6,
    "a" : "Representing graphics for math functions",
    "u" : "Peter Bijlsma",
    "c" : ""
},
{
    "i" : "1", "p" : 8,
    "a" : "Client Dataset Toolkit", "u" : "Detlef Overbeek",
    "c" : ""
},
// ....
]
    
```





Diese Javascript-Datei wird mithilfe eines Skript-Elements in die HTML-Seite eingebunden:

```
<script src="js/articles.js"></script>
```

Der Zugriff auf dieses Array aus einem Pascal-Programm ist einfach: Ein Datensatz in diesem Array kann in Pascal als externe Klasse deklariert werden:

```
Type
TArticle = Class external name 'Object' (TJSObject)
Issue : String; external name 'i';
Page : Integer; external name 'p';
Title : String; external name 'a';
Author : String; external name 'u';
Code : string; external name 'c';
end;
TArticleArray = Array of TArticle;
```

Beachten Sie die Verwendung von 'external name', um die von Menschen lesbaren Felder (*Heft, Seite usw.*) auf die in Javascript verwendeten Namen der Mitglieder abzubilden. Das Array selbst ist dann wie folgt definiert:

```
var
    BPMArticles : TArticleArray; external name 'BPMArticles';
```

Wie Sie sehen können, ist die Variable als extern deklariert: Das bedeutet, dass sie tatsächlich außerhalb des Pascal-Codes definiert ist.

Mit diesem Wissen können wir nun eine Klasse erstellen, die einen lokalen Suchmechanismus und einen Mechanismus zum Abrufen einer Wortliste implementiert:

die Klasse TArticleSearcher in der Einheit articlesearch.

Diese Klasse ist als Abkömmling von TSearchBase definiert und verfügt über 2 Hauptmethoden. Die erste dient dazu, eine Liste von Wörtern zu erhalten:

```
procedure TArticleSearcher.GetWordList(aTerm: string; aOnResults: TWordListCallback);
var
    L : TStringList; aArticle : TArticle; S : String;
    R : TJSRegexp;
begin
    if not assigned(aOnResults) then exit;
    aTerm:=UpperCase(aTerm);
    L:=TStringList.Create;
    try
        L.Sorted:=True;
        L.Duplicates:=dupIgnore;
        R:=TJSRegexp.New('\b(?:\w|-)+\b','g');
        For aArticle in BPMArticles do
            for S in TJSString(aArticle.Title).match(R) do
                if pos(aTerm,UpperCase(S))>0 then L.Add(s);
                aOnResults(L);
    finally
        L.Free;
    end;
end;
```

Dies ist eine sehr einfache Schleife über das Array der Artikeldatensätze: Für jeden Artikel wird das Feld Titel mithilfe der Javascript-Methode 'match' des Typs String in Wörter zerlegt: Wenn das Wort den Suchbegriff enthält (*wir prüfen dies unabhängig von der Groß-/Kleinschreibung*), fügen wir es der Liste hinzu:

Die Liste ignoriert Duplikate, so dass wir jedes Wort nur einmal erhalten. Am Ende rufen wir den Callback auf.

Der Suchmechanismus funktioniert auf ganz ähnliche Weise. Er löscht alle vorherigen Ergebnisse, läuft in einer Schleife über die Artikelliste und wenn ein Artikel mit dem Suchbegriff übereinstimmt, wird er in das Ergebnis aufgenommen.





```

procedure TArticleSearcher.Search(const aTerm: string; const aIssue: String);
Var
    aIdx : integer;
    aArticle : TArticle;
    V, IssueFilter : string;
    I : integer;

begin
    if Not Assigned(ResultsElement) then exit;
    IssueFilter:='';
    For I:=1 to Length(aIssue) do
        if Pos(aIssue[I],'0123456789_')>0 then
            IssueFilter:=IssueFilter+V[I];
    ClearResultPanel;
    For aIdx:=0 to Length(BPMArticles)-1 do
        begin
            aArticle:=BPMArticles[aIdx];
            if aArticle.IsMatch(aTerm,IssueFilter) then
                ResultsElement.AppendChild(CreateArticleRow(aIdx,aArticle));
            end;
        ShowResultsPanel;
    end;
    
```

Am Ende wird das Ergebnisfenster angezeigt.
 Die IsMatch-Prozedur, die verwendet wird, um festzustellen, ob ein Artikel übereinstimmt, ist eine Hilfsmethode für TArticle:

```

TArticleHelper = class helper for TArticle
    Function IsMatch (aTerm : String; aIssue : string) : Boolean;
end;

function TArticleHelper.IsMatch(aTerm: String; aIssue: string): Boolean;
begin aTerm:=UpperCase(aTerm);
    Result:=(aTerm='') or ((Pos(aTerm,UpperCase(Author))>0)
        or (Pos(aTerm,UpperCase(Title))>0));
    if Result and (aIssue<>'') then
        Result:=(aIssue=Issue);
end;
    
```

Diese muss als Hilfsmethode implementiert werden, da die Klasse TArticle als externe Klasse definiert ist und ihre Definition daher keine Pascal-Methoden enthalten kann.

Die Methode CreateArticleRow verwendet eine String-Konstante DefaultPanel mit einer HTML-Vorlage, um das eigentliche HTML mithilfe eines einfachen Such- und Ersetzungsmechanismus zu konstruieren:

```

function TArticleSearcher.CreateArticleRow(aIdx: Integer; aArticle: TArticle):
    TJSHTMLElement;
Var
    Panel : String;

begin
    Result:=TJSHTMLElement(Document.createElement('div'));

    Panel:=StringReplace(DefaultPanel,'{{issue}}',aArticle.Issue,[rfReplaceAll]);
    Panel:=StringReplace(Panel,'{{page}}',IntToStr(aArticle.Page),[rfReplaceAll]);
    Panel:=StringReplace(Panel,'{{title}}',aArticle.Title,[rfReplaceAll]);
    Panel:=StringReplace(Panel,'{{author}}',aArticle.Author,[rfReplaceAll]);

    Result.dataset['issue']:=aArticle.Issue; Result.dataset['page']:=IntToStr(aArticle.Page);
    Result.dataset['articleid']:=intToStr(aIdx); Result.dataset['title']:=aArticle.Title;
    Result.AddEventListener('click',@OnArticleClick); Result.innerHTML:=Panel;
end;
    
```





Schließlich sammelt der OnClick-Handler für das Ergebniselement einige Daten zur Erstellung eines TPageInfo-Datensatz zu erstellen, der dann verwendet wird, um die richtige PDF-Seite anzuzeigen:

```
procedure TArticleSearcher.OnArticleClick(aEvent : TJSEvent);
var
    aPage : TPageInfo;
begin
    aPage:=Default(TPageInfo);
    With TJSHTMLLElement(aEvent.currentTargetElement) do
        begin
            aPage.Issue:=dataset['issue'];
            aPage.Page:=StrToIntDef(dataset['page'],-1);
            aPage.Title:=dataset['title']; aPage.useIssue:=True;
        end;
    ShowPDFPage(aPage);
end;
```

ShowPDFPage verwendet den ShowPDFPageEvent-Event-Handler, um die PDF-Datei tatsächlich auf der richtigen Seite anzuzeigen.

Und damit ist unser Offline-Suchmechanismus fertig.

7 ERKENNEN DES OFFLINE-STATUS UND ANZEIGEN EINES PDF

Der Offline-Suchmechanismus muss aktiviert werden, wenn der Navigator keinen Zugang zum Internet hat: Wir haben dafür die Eigenschaft OffLine im TPDFSearchControl implementiert. Aber diese Eigenschaft wurde noch nicht auf einen korrekten Wert gesetzt. Glücklicherweise verfügt der Browser über eine Eigenschaft, die angibt, ob er gerade online oder offline ist: Die Eigenschaft window.Navigator.onLine zeigt an, ob der Browser gerade online oder offline ist. Darüber hinaus implementiert die Klasse Window zwei Ereignisse 'online' und 'offline', die ausgelöst werden, wenn der Browser online bzw. offline geht. Wir können also AddEventListener verwenden, um einen Event-Handler zu installieren und auf Änderungen im Online- oder Offline-Status zu reagieren.

Dies geschieht in der Routine DetectOffline in der Anwendungsklasse.

Sie tut zwei Dinge:

Sie erkennt, ob die Anwendung durch einen Doppelklick auf die Datei index.html im Datei-Explorer gestartet wurde oder ob sie von einer Website aus gestartet wurde.

Das Ergebnis wird in der Eigenschaft IsLocal gespeichert und der Online-Status wird in der Eigenschaft IsOffline gespeichert:

```
procedure TBPMLibraryApplication.DetectOffline;
    Procedure updateOnlineStatus(event : TJSEvent);
    begin
        IsOffline:=not window.Navigator.onLine;
    end;
begin
    IsLocal:=Copy(window.location.protocol,1,4)='file';
    IsOffline:=not window.Navigator.onLine;
    window.addEventListener('online',@updateOnlineStatus);
    window.addEventListener('offline',@updateOnlineStatus);
end;
```

HINWEIS: Der Online- und Offline-Eventhandler wird zur Aktualisierung der Eigenschaft IsOffline verwendet.

Die IsOffline-Eigenschaft des Anwendungsobjekts verfügt über einen Setter und wird verwendet, um den Wert an das searchcontrol weiterzugeben:

```
procedure TBPMLibraryApplication.SetIsOffline(AValue: Boolean);
begin
    if FIsOffline=AValue then Exit;
    FIsOffline:=AValue;
    FSearchPane.Offline:=FIsOffline;
end;
```





So weiß der Suchmechanismus, ob er lokal oder remote suchen soll. Wie wir gesehen haben, hat der Suchmechanismus nur ein Ereignis, das er verwenden muss, um eine PDF-Datei zu öffnen und eine bestimmte Seite anzuzeigen. Das Ereignis wird in der Hauptanwendung auf den folgenden Ereignishandler gesetzt:

```
procedure TBPMLibraryApplication.HandleShowPDFPage(aPage: TPageInfo);

Function IsSameAsLastPDF : Boolean;
begin
  if aPage.useIssue then
    Result:=(FLastIssue<>"") and (FLastIssue=aPage.Issue)
  else
    Result:=(FLastPDF<>"") and (FLastPDF=aPage.FileName)
  end;
begin
  // local search or PDF already loaded
  if IsSameAsLastPDF then
    FViewer.ShowPage(aPage.Page)
  else if Not HaveLocalFile(aPage) then
    LoadRemotePDF(aPage)
  else
    LoadLocalPDF(aPage);
end;
```

Wenn die PDF-Datei bereits geladen ist (*markiert in IsSameAsLastPDF*), wird das Anzeigefenster einfach angewiesen, die neue Seite anzuzeigen. Wenn die PDF-Datei noch nicht geladen ist, muss sie erst geladen werden, bevor die richtige Seite angezeigt werden kann. Wenn die Seite von der lokalen Festplatte geladen wurde (*wie es bei der USB-Stick-Version des Blaise Pascal Magazins der Fall ist*), dann wird sie mit `LoadLocalPDF` von der Festplatte geladen, andernfalls mit `LoadRemotePDF`.

Die Funktion `HaveLocalFile` verwendet die Eigenschaft `IsLocal`, die von der Anwendung initialisiert wurde, um zu entscheiden, ob eine Datei von der Festplatte geladen werden kann oder nicht:

Wenn `IsLocal` falsch ist, wissen wir, dass die Seite von einer Website geladen wurde und die PDF-Dateien nicht lokal verfügbar sind. Aber wenn `IsLocal` wahr ist, kann es immer noch sein, dass das PDF lokal nicht verfügbar ist:

Wenn eine Online-Suche durchgeführt wurde, könnte das Suchergebnis eine PDF-Datei ergeben haben, die lokal nicht verfügbar ist. In diesem Fall müssen wir in der Liste der verfügbaren Artikel nachsehen, ob das gesuchte Heft vorhanden ist, indem wir die Heftnummer in der Artikelliste überprüfen:

```
function TBPMLibraryApplication.HaveLocalFile(aPage : TPageInfo) : Boolean;

Var
  aArticle : TArticle;

begin
  Result:=IsLocal;
  if Result then
    begin
      // Determine if we have the PDF locally
      Result:=False;
      for aArticle in BPMArticles do
        if (aPage.Issue=aArticle.Issue) then
          Exit(True);
        end;
      end;
    end;
end;
```

Wenn das PDF lokal verfügbar ist, laden wir es mit dem im ersten Artikel dieser Serie gezeigten Trick: Es wird ein Skript-Tag eingefügt, das den Inhalt des PDFs als Javascript-Variable (`pdfData`) definiert:





```

var
  pdfData : String; external name 'pdfData';

procedure TBPMLibraryApplication.LoadLocalPDF(aPage: TPageInfo);

  Procedure DoShowPage;
  begin
    FViewer.ShowPage(aPage.Page);
  end;

  function DoLoaded(Event : TEventListenerEvent) : Boolean;
  var
    Src : TPDFSource;

  begin
    Src:=TPDFSource.new;
    Src.Data:=pdfData;
    Fviewer.StartPDFRender(Src,@DoShowPage);
  end;

Var
  Script : TJSHTMLScriptElement;
  FN : String;

begin
  Script:=TJSHTMLScriptElement(document.CreateElement('script'));
  FN:=aPage.FileName;
  if Pos('file://',FN)=1 then
    Delete(FN,1,7);
  else if FN="" then
    FN:='issues/issue'+aPage.Issue+'.js';
  Script.Src:=FN;
  Script.Onload:=@DoLoaded;
end;
    
```

Schließlich lädt die Routine LoadRemotePDF eine PDF-Datei vom Server. Sie muss unterscheiden zwischen einem Aufruf, bei dem nur die Heftnummer angegeben ist (*der Benutzer wollte einfach nur ein Heft sehen*) oder wenn der PDF-Dateiname bekannt ist. Im ersten Fall wird die URL /issue/ verwendet, die wir am Anfang des Artikels gezeigt haben, im zweiten Fall wird die URL /pdf/ verwendet:

```

procedure TBPMLibraryApplication.LoadRemotePDF(aPage: TPageInfo);

  Procedure DoShowPage;
  begin
    FViewer.ShowPage(aPage.Page);
  end;

var
  Src : TPDFSource;

begin
  Src:=TPDFSource.new;
  if aPage.useIssue then
    Src.url:=ServerURL+'issue/'+aPage.Issue
  else
    Src.url:=ServerURL+'pdf/'+aPage.FileName+'?token='+ encodeURIComponent(FLogin.Token);
  FLastPDF:=aPage.FileName;
  FViewer.StartPDFRender(Src,@DoShowPage);
end;
    
```

Und mit dieser Routine ist die Anwendung einsatzbereit. Die Anwendung mit einem lokal geladenen Heft sehen Sie in *Abbildung 1 auf Seite 21* auf der nächsten Seite.





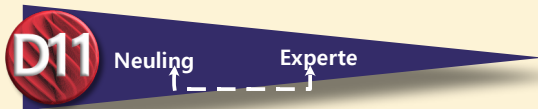
8 SCHLUSSFOLGERUNG

In diesem Artikel haben wir gezeigt, wie Sie mit **PAS2JS** eine reale Anwendung erstellen, die sowohl online als auch offline funktionieren kann und sich selbst anpasst: Wir haben die in den vorherigen Artikeln dieser Serie vorgestellten Techniken überarbeitet. Die Anwendung kann noch verbessert werden:

Zum Beispiel kann der Online-/Offline-Status visuell gestaltet werden - z.B. durch Ändern der Hintergrundfarbe.

Die Anwendung schaltet nach 30 Minuten ab, aber es wird keine Warnung ausgegeben: auch dies kann verbessert werden. Wie bei jeder Software ist die Arbeit nie zu Ende...





KURZFASSUNG

DropMaster von Ray Konopka ist eine Lösung für die Unterstützung von Drag & Drop in Anwendungen für Microsoft Windows. Die Komponenten und Demos wurden von Jim O'Brien von UnitOOPS Software entwickelt, die ursprünglich OLE Drag and Drop Components hießen.

Ray war der Meinung, dass sie eine großartige Ergänzung für die Produktlinie von Raize Software sein würden. Er benannte das Produkt in DropMaster um, erstellte die Trial Edition, und nutzte die One-Step-Installer-Technologie von Raize Software, um den Installationsprozess erheblich zu vereinfachen. Dies ist ein sehr nützliches und oft notwendiges Werkzeug, um die Benutzeroberfläche für den Endbenutzer interaktiv und logisch zu gestalten.

Es sollte in jeder Anwendung vorhanden sein, denn auch innerhalb einer Anwendung kann es sehr praktisch oder notwendig sein.

EINLEITUNG

Dies ist eine Sammlung von Komponenten, die Delphi-Programmen, die unter Microsoft Windows laufen, anwendungsübergreifende Drag-and-Drop-Funktionen verleihen. Drag-and-Drop von textbasierten Daten, Bildern und benutzerdefinierten Formaten wird unterstützt.

Sie enthält mehr als 40 Beispielanwendungen, die die Ergebnisse einer eingehenden Untersuchung der Drag-and-Drop-Funktionen zahlreicher bekannter kommerzieller Anwendungen darstellen. Das ist sehr hilfreich, insbesondere für Anfänger.

Das aktuelle Release 2.5.2, das mit RAD Studio (Delphi) 2009 gestartet wurde, unterstützt die neuesten Delphi 11 Alexandria 11.3 sowie wahrscheinlich auch zukünftige Versionen.

Das ist ein großer Wert. Vor allem, wenn Sie sich der Taktik bewusst sind, die angewendet wird: extrem kompliziert: für 99 \$ erhalten Sie schnell einen Gegenwert für Ihr Geld.

INSTALLIEREN

Als Gefallen für potenzielle Kunden, die die Komponenten des Produkts testen möchten, bietet diese Komponentensuite eine Testversion.

Wie bei den meisten Testversionen von VCL-basierten Komponenten üblich, sind die Komponenten in dieser Testversion dieselben wie die veröffentlichten Versionen, mit der Ausnahme, dass sie nur gleichzeitig mit Delphi verwendet werden können.

Das bedeutet, dass eine Anwendung, die eine der Komponenten der Testversion verwendet, nur von Delphi aus gestartet werden kann.

Stellen Sie sicher, dass Sie über Administrator-rechte verfügen, und legen Sie die Projekte vorzugsweise auf einer anderen Festplatte als C: ab. Der Installationsvorgang ist denkbar einfach; Sie müssen nicht die üblichen Schritte befolgen.

Sie können DropMaster in RAD Studio, Delphi verwenden sobald das Installationsprogramm abgeschlossen ist. Die Seite "DropMaster erscheint in der Komponentenpalette nach dem Neustart von RAD Studio. Siehe Abbildung 1:

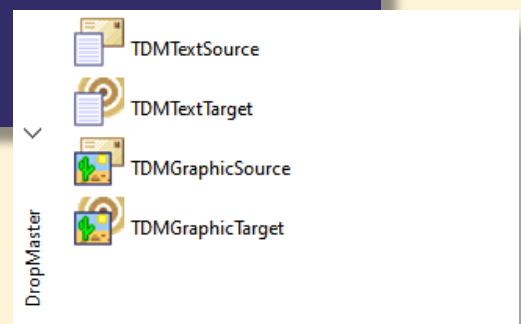


Abbildung 1: Überblick über die Komponentengruppe.



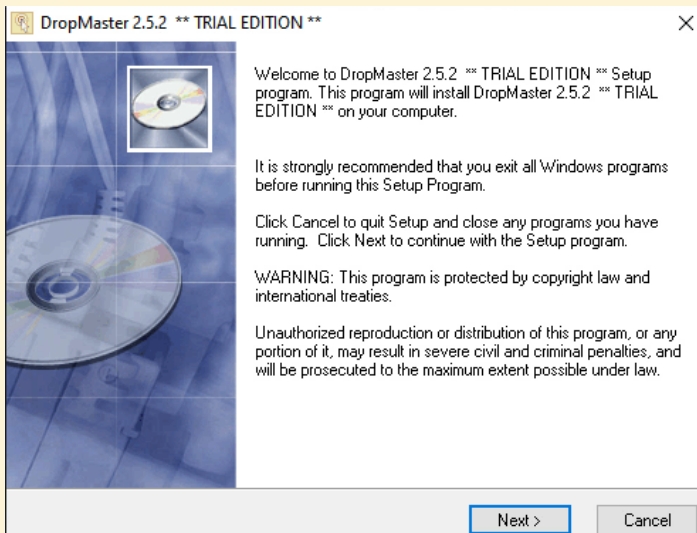


Abbildung 2: Installation der Trial Edition - voll funktionsfähig

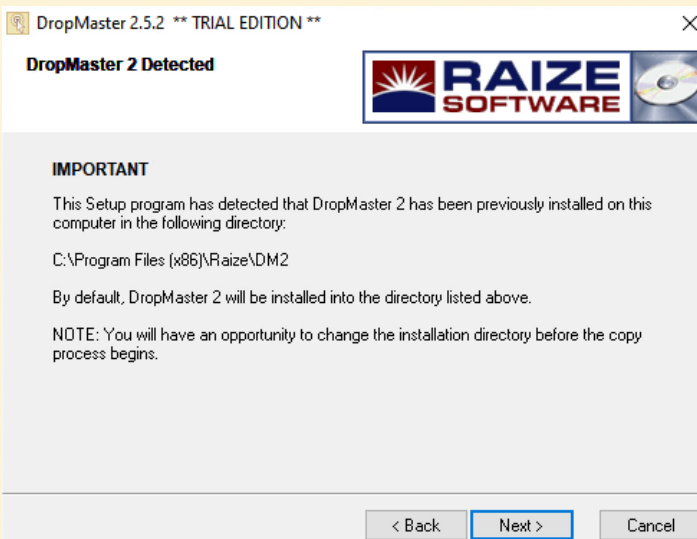


Abbildung 3: Es merkt, wenn Sie es schon einmal versucht haben

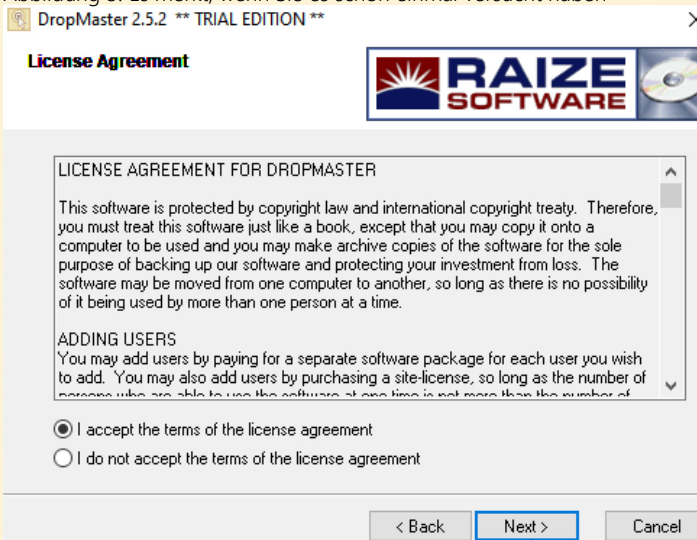


Abbildung 3: Lizenz



Wie bei Rays Produkten üblich, ist es unglaublich effektiv und hilfreich. Das macht Sinn, wenn man bedenkt, dass er das Buch über die Herstellung von Komponenten geschrieben hat. Er ist nicht nur unglaublich kreativ, sondern auch der eigentliche Erfinder.

Die Bilder in diesem Beitrag veranschaulichen das Verfahren und werden Sie davon überzeugen, wie einfach es zu benutzen ist.

DE-INSTALLATION

Entfernen der Komponenten aus der Komponentenpalette von RAD Studio:

Schließen Sie alle Dateien und Projekte und wählen Sie Komponente|Pakete installieren..., um die Seite Pakete im Dialogfeld Projektoptionen anzuzeigen.

Wählen Sie das Paket "DropMaster 2.x" aus der Liste Design Packages und klicken Sie auf den Button Entfernen.

Ein Meldungsfenster wird angezeigt, in dem Sie Ihre Anfrage bestätigen müssen - klicken Sie auf OK.

Als nächstes werden Sie je nach IDE (Integrierte Entwicklungsumgebung) gefragt, ob ein Laufzeitpaket aus der Liste Laufzeitpakete entfernt werden soll.

Klicken Sie in diesem Fall auf OK, um das Laufzeitpaket zu entfernen. Schließen Sie das Dialogfeld Projektoptionen, indem Sie auf den Button OK klicken.

Wiederholen Sie die obigen Schritte für jede IDE, die DropMaster verwendet.

Und um es ganz vollständig zu machen: Entfernen aller Komponentendateien von Ihrer Festplatte

Zu diesem Zeitpunkt verwenden alle IDE's DropMaster nicht mehr.

Um die DropMaster-Dateien von Ihrer Festplatte zu entfernen, wählen Sie die Ikone "Software" in der Systemsteuerung.

Wählen Sie dann den Eintrag "DropMaster 2.x" aus der Liste der installierten Programme und klicken Sie auf den Button Entfernen. (Versionen von Delphi).

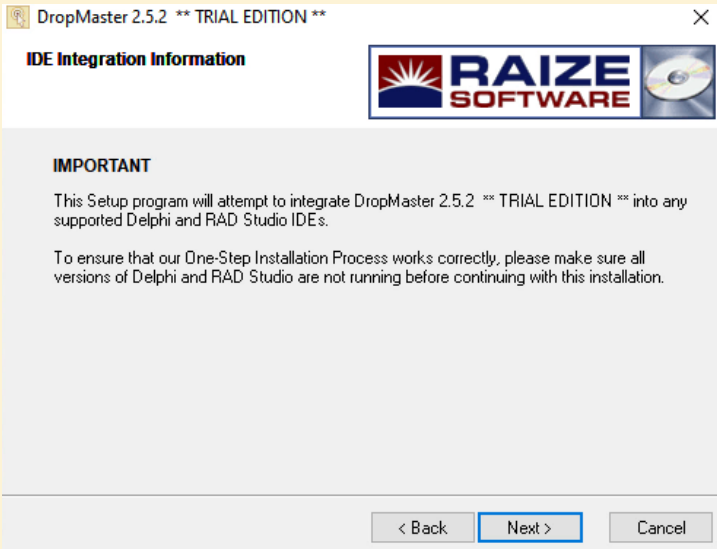


Abbildung 5: Führen Sie während des Installationsvorgangs KEIN RAD STUDIO oder Delphi aus

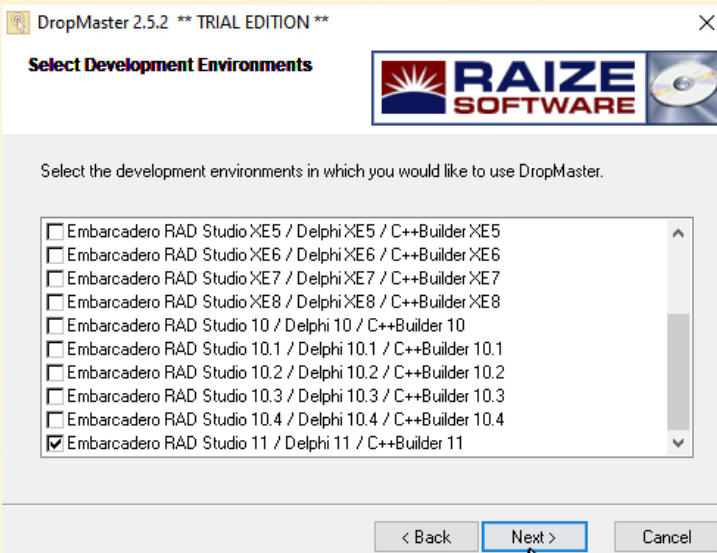


Abbildung 6: Das Menü

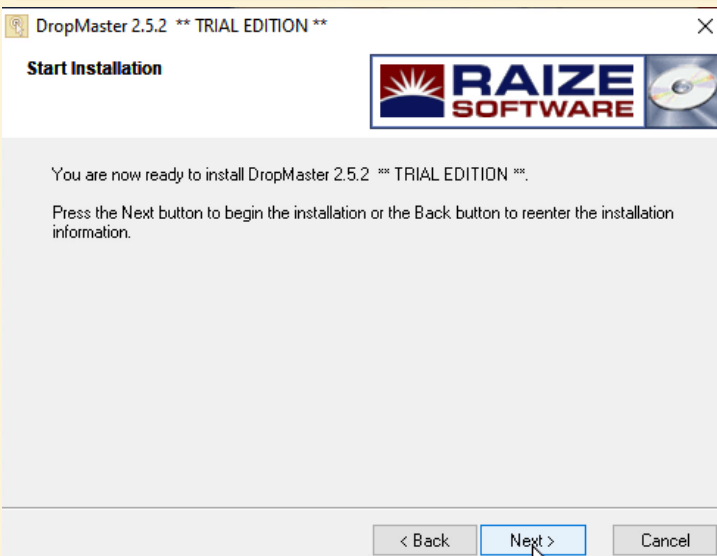


Abbildung 7: Letzte Warnung vor der Installation



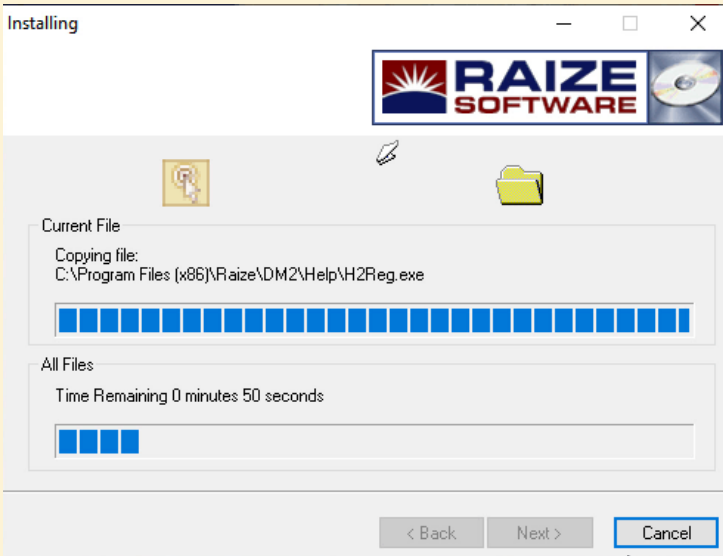


Abbildung 8: Der Fortschritt der Installation

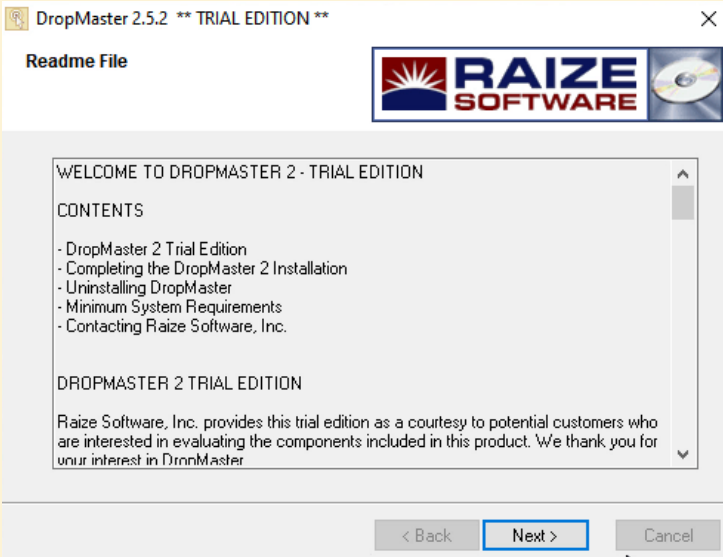


Abbildung 9: Überblick über den Inhalt

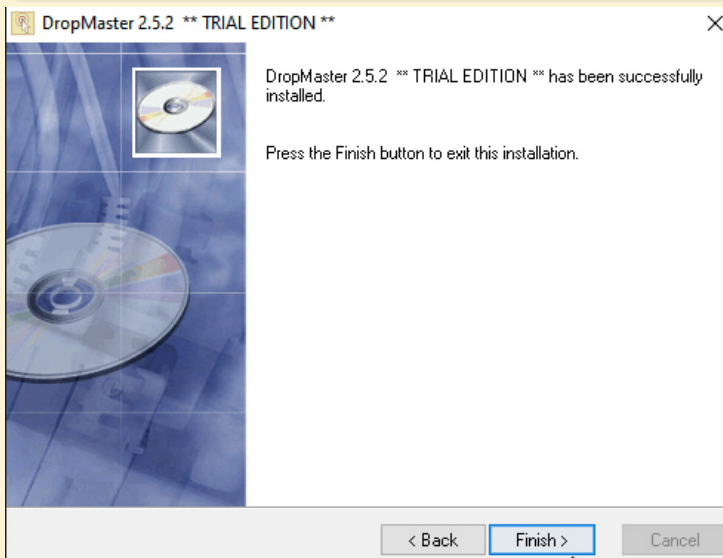


Abbildung 10: Bereit zur Verwendung



THE COMPONENTS

TDMTextTarget

Die Hauptkomponente zur Annahme von Daten, die von einem anderen Programm in Ihr Programm gezogen werden, heißt `TDMTextTarget`. Um auf abgelegte Daten zu reagieren, weisen Sie einfach die Eigenschaft `AcceptorControl` zu und erstellen einen `OnDrop-Event-Handler`. Text, RTE, HTML, Dateilisten und URLs können von `TDMTextTarget` akzeptiert werden. Diese Komponente kann nicht nur Text, sondern auch andere beliebige Formen akzeptieren. Alle abgelegten Formate sind über das `OnDrop-Ereignis` zugänglich. Auf den nächsten Seiten werde ich einige Beispielprojekte zeigen. Natürlich können Sie diese Beispiele an Ihre Bedürfnisse anpassen. Stellen Sie jedoch sicher, dass Sie das Projekt in einem anderen Verzeichnis speichern: NICHT C: - Windows schützt diesen Datenträger und lässt nicht zu, dass Sie normale Dinge tun. Sie werden schlechte Ergebnisse erhalten!

TDMTextSource

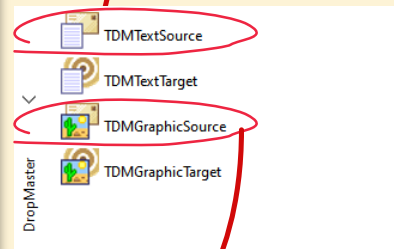
Daten, die in eine andere Anwendung gezogen werden müssen, werden über die Komponente `TDMTextSource` bereitgestellt. Geben Sie einfach einem Steuerelement die Eigenschaft `DonorComponent` und rufen Sie die Methode `Execute` auf, wenn das `OnMouseDown-Ereignis` des Steuerelements das Ziehen erkannt hat. Um benutzerdefiniertes Material bereitzustellen, verwenden Sie das Attribut `Text`. Die gleichzeitige Ausgabe mehrerer Formate ist eine weitere Funktion dieser Komponente.

TDMGraphicTarget

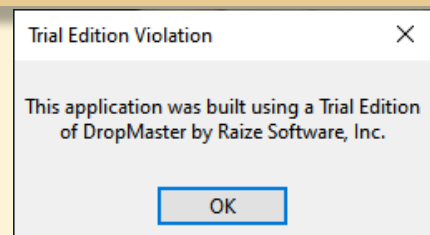
Die Komponente `TDMGraphicTarget` wird verwendet, um Fotos und Bilder zu akzeptieren, die aus einer anderen Anwendung gezogen werden. Der Entwickler muss lediglich ein Platzhalter-Steuerelement für die empfangenen Grafikdaten bereitstellen, da die Komponente mit zahlreichen Formaten interagiert, darunter DIBs, Bitmap-Handles und Metadateien.

TDMGraphicSource

Beim Ziehen und Ablegen werden in der Regel Grafiken als Datentyp verwendet. Die `TDMGraphicSource` ist vergleichbar mit der `TDMTextSource`, mit der Ausnahme, dass sie die Attribute `DonorImage` und `Picture` anstelle der Eigenschaften `DonorComponent` und `Text` der `TDMTextSource` besitzt. Es ist einfach, ein Bild in den Drag-daten bereitzustellen, indem Sie es einer Eigenschaft zuweisen, das Ziehen erkennen und dann `Execute` aufrufen.



Führen Sie die Projekte innerhalb von Delphi im Debug-Modus aus, oder aber



BEISPIELPROJEKTE: ÜBERSICHT ÜBER ALLE PROJEKTE

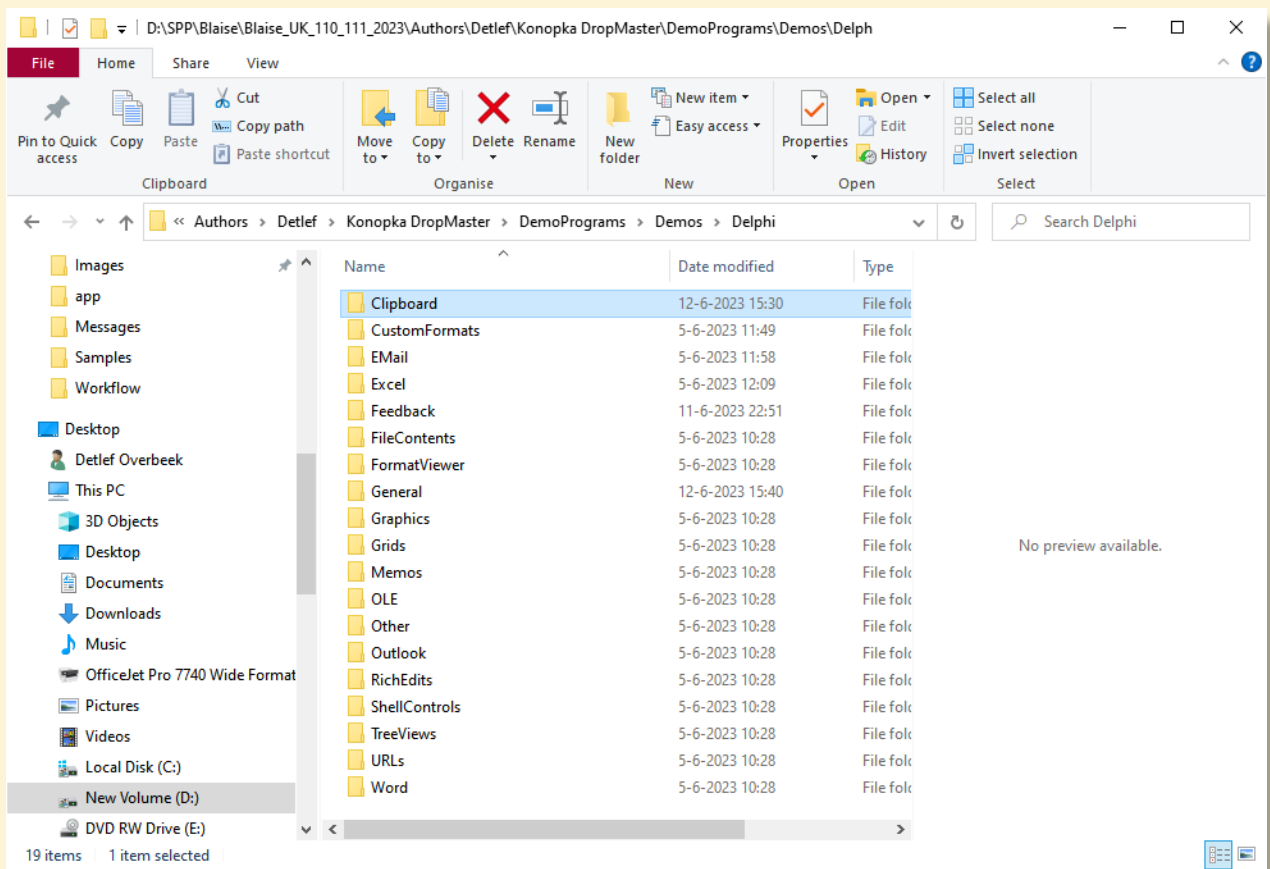


Abbildung 11: Übersicht der verfügbaren Projekte

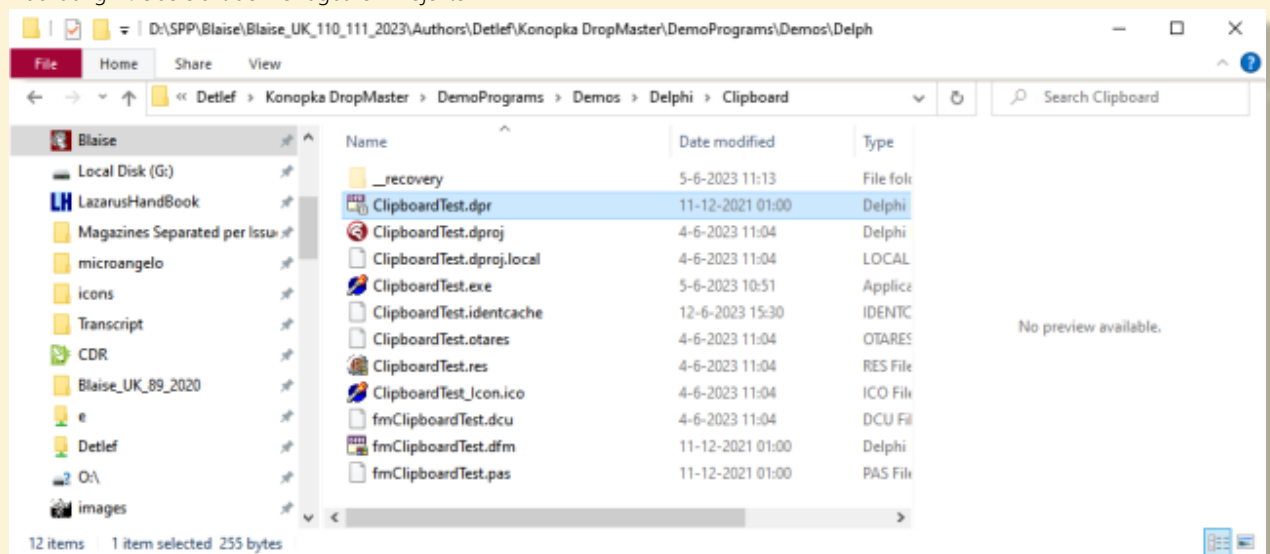


Abbildung 12: der Inhalt dieses Verzeichnisses.

Hier sehen Sie den Inhalt des Verzeichnisses des CLIPBOARD PROJECT.
Wenn Sie diese Komponentensuite nur für diese App kaufen würden, ist sie es schon wert. Sie könnten eine kleine App erstellen, in der Sie sehen können, wie Sie eine Liste von Produkten und Namen erstellen können, die Sie automatisch in den Betreff Ihrer E-Mail einfügen können



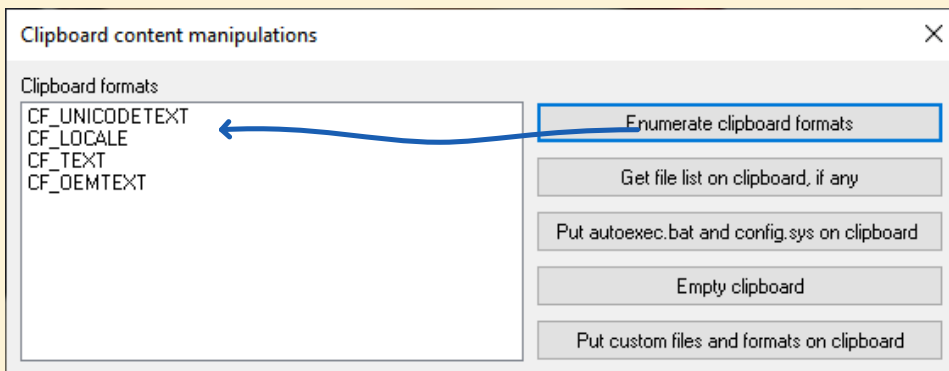


Abbildung 13: Auflistung der möglichen Textsorten

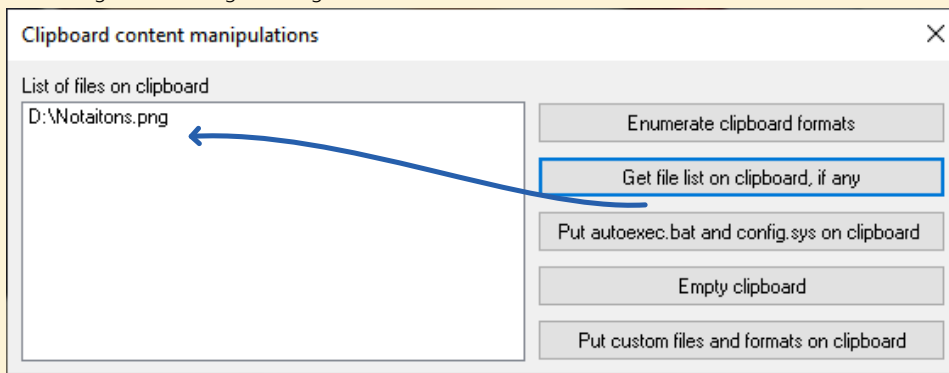


Abbildung 14: Eine PNG-Datei gefunden Im Speicher

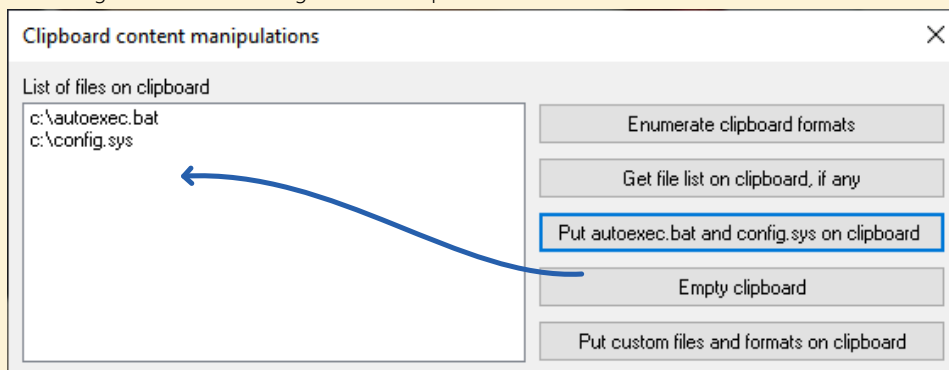


Abbildung 15: Kann sehr praktisch sein, wenn Sie sie brauchen

Project Overview Form

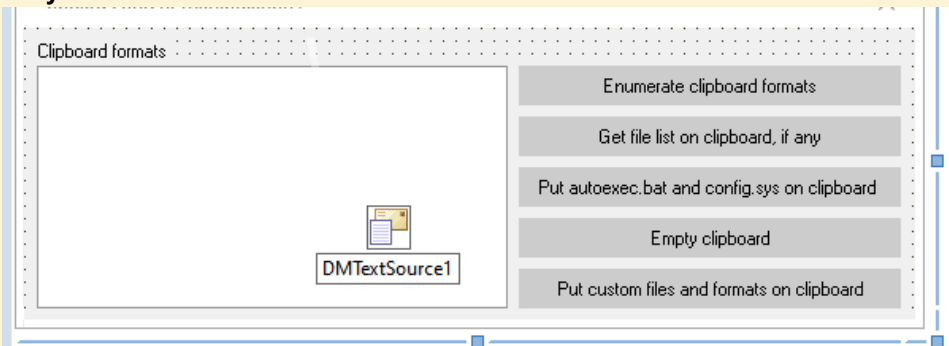


Abbildung 16: Projektübersicht Zwischenablage-Formular in Delphi




```
unit fmClipboardTest;
```

```
{ Example application for DropMaster.
```

Demonstrates using the DMUtil clipboard format helper functions to manipulate actual clipboard data rather than drag-and-drop data. In fact, this example has nothing to do with drag-and-drop, and contains no DropMaster components!

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  
StdCtrls, DMComps;
```

```
type
```

```
TForm1 = class(TForm)  
  ListBox1: TListBox;  
  btnEnumClipboard: TButton;  
  btnCFHDROP: TButton;  
  btnPutClipboard: TButton;  
  btnEmptyClipboard: TButton;  
  Label1: TLabel;  
  btnCustomFiles: TButton;  
  DMTextSource1: TDMTextSource;  
  procedure btnEnumClipboardClick(Sender: TObject);  
  procedure btnCFHDROPClick(Sender: TObject);  
  procedure btnPutClipboardClick(Sender: TObject);  
  procedure btnEmptyClipboardClick(Sender: TObject);  
  procedure btnCustomFilesClick(Sender: TObject);  
private  
  { Private declarations }  
public  
  { Public declarations }  
end;
```

```
var
```

```
Form1: TForm1;
```

```
implementation
```

```
uses
```

```
ClipBrd, DMUtil, ActiveX;
```

```
{ $R *.DFM }
```

```
procedure TForm1.btnEnumClipboardClick(Sender: TObject);  
// Fill the list box with the name of every format currently available  
// on the clipboard
```

```
var
```

```
aFmt: DWORD;  
j: Integer;
```

```
begin
```

```
ListBox1.Clear;
```

```
for j := 1 to Clipboard.FormatCount do // Iterate
```

```
begin
```

```
  aFmt := Clipboard.Formats[j-1];  
  ListBox1.Items.Add(Clipboard.FormatDisplayname(aFmt));
```

```
end; //for
```

```
Label1.Caption := 'Clipboard formats';
```

```
end;
```



```
procedure TForm1.btnCFHDROPClick(Sender: TObject);
// Get the list of files, if any, available on the clipboard
// You'd use this, e.g., to do a paste operation for files.
var
  aSL: TStringList;
  s: AnsiString;
begin
  Clipboard.Open;
  try
    s := GetHandleDataToString(Clipboard.GetAsHandle(CF_HDROP));
    aSL := FileListFromHDROP(s);
    // Now we have the file list. We could paste these, or make shortcuts.
    // Here we just put the names into a listbox.
    try
      ListBox1.Clear;
      ListBox1.Items.AddStrings(aSL);
    finally
      aSL.Free;
    end;
  finally
    Clipboard.Close;
  end;

  Label1.Caption := 'List of files on clipboard';
end;

procedure TForm1.btnPutClipboardClick(Sender: TObject);
// Put a list of files on the clipboard for pasting or pasting as shortcut
var
  aSL: TStringList;
  s, t: AnsiString;
begin
  aSL := TStringList.create;
  try
    // Put two file names in the list
    aSL.Add('c:\autoexec.bat');
    aSL.Add('c:\config.sys');
    // Generate both CF_HDROP and Shell IDList Array formats
    s := HDropFromFileList("", aSL);
    t := ShellIDListFromFileList("", aSL);
  finally
    aSL.Free;
  end;

  // Put the formats on the clipboard
  Clipboard.Open;
  try
    // CF_HDROP, so we can paste files
    Clipboard.SetAsHandle(CF_HDROP,
      SetHandleDataFromStrings(s));
    // Shell IDList Array, so we can paste shortcuts also
    Clipboard.SetAsHandle(ClipboardFormatFromStrings('Shell IDList Array'),
      SetHandleDataFromStrings(t));
  finally
    Clipboard.Close;
  end;
end;

procedure TForm1.btnEmptyClipboardClick(Sender: TObject);
// Clear the clipboard
begin
  Clipboard.Clear;

  ListBox1.Clear;
  Label1.Caption := 'Clipboard formats';
end;
```



```
procedure TForm1.btnCustomFilesClick(Sender: TObject);
// Put some custom stuff on the clipboard (to paste files that don't really exist)
// Works just like the FileContentsTest* demos
var
  sizes: array of integer;
  aSL: TStringList;
  bSL: TStringList;
  i: Integer;
  j: integer;
  aDataObject: IDataObject;
begin
  aSL := TStringList.Create;
  bSL := TStringList.Create;

  try
    // File names
    aSL.Add('first file.txt');
    aSL.Add('second file.txt');

    // File contents
    bSL.Add('contents of first file');
    bSL.Add('contents of second file');

    // Get sizes of contents
    SetLength(sizes, bSL.Count);
    for I := 0 to bSL.Count - 1 do // Iterate
    begin
      sizes[i] := length(bSL[i]);
    end; //for

    // Set up the data object. Use the internal support in TDMTextSource to do this.
    With DMTextSource1, CustomFormatData do
    begin
      Clear; // Empty the data formats

      // Pasting nonexistent files needs FileGroupDescriptor and FileContents formats.
      Add-format(DMFileGroupDescriptorFormatName,
        FileGroupDescriptorFromFileListEx(", aSL, sizes));

      // FileContents is an indexed format
      for j := 0 to bSL.Count-1 do
      begin
        AddFormatEx('FileContents', bSL[j], TYMED_HGLOBAL, j);
        // Make sure we don't get a trailing null in the content items
        // Items[Count-1] is the TCustomFormatData we just added.
        Items[Count-1].AllowTrailingNull := false;
      end;

    end;

    // Make a data object and put it on the clipboard.
    aDataObject := TTextDataObject.CreateExWithFormats(DMTextSource1.CustomFormatData,
      DMTextSource1);

    OleSetClipboard(aDataObject);

  finally
    // Clean up
    aSL.Free;
    bSL.Free;
  end;
end;

end.
```



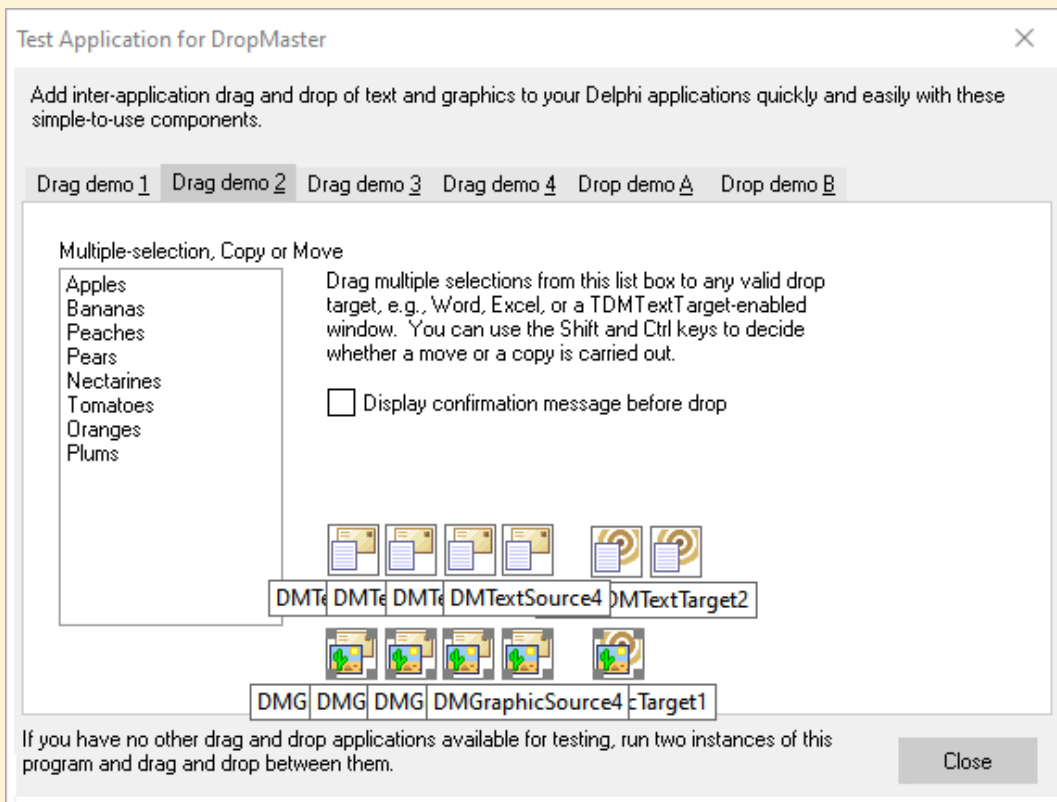


Abbildung 17: Projektübersicht TEST APPLICATION der verschiedenen Beispiele. Formular innerhalb von Delphi

Demonstriert verschiedene Techniken bei der Verwendung von TDMDTextSource, TDMDTextTarget, TDMGraphicSource, TDMGraphicTarget.

Der wichtigste Punkt hier: Sehen Sie, wie wenig Code erforderlich ist.
 Der meiste Code hier wurde von Delphi selbst geschrieben!
 Und der Teil, der manuell kodiert wurde, dient hauptsächlich der Benutzerschnittstelle.

```

interface

uses
    Windows, Types, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    StdCtrls, ComCtrls, Buttons, Grids, ExtCtrls, DMComps;

...

var
    Form1: TForm1;

implementation

{$SR *.dfm}

uses
    TypInfo;
    
```



```
procedure TForm1.ListBox1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
// Detect the start of a drag for ListBox1, and begin the drag operation.
begin
  if DragDetect(ListBox1.Handle, POINT(X,Y)) then
    DMTextSource1.Execute;
end;

procedure TForm1.ListBox2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
// Detect the start of a drag for ListBox2, and begin the drag operation.
begin
  if DragDetect(ListBox2.Handle, POINT(X,Y)) then
    DMTextSource2.Execute;
end;

procedure TForm1.DMTextTarget1Drop(Sender: TObject; Acceptor: TWinControl;
  const dropText: String; X, Y: integer);
// Handle drops on memo1. Notify the user whether files or just non-file text
// has been dropped.
begin
  label24.caption := Format('%s: (%d,%d)', ['Drop', X, Y]);

  if ((Sender as TDMTextTarget).DroppedTextFormat = dtfFiles) then
    label24.caption := label24.caption + ' [files]'
  else
    label24.caption := label24.caption + ' [text]';

  // If this is a URL, the actual URL address is in dropText
  // [also in droppedLines[0] and Text]. The title is in URLTitle.
  if ((Sender as TDMTextTarget).DroppedTextFormat = dtfURL) then
    label24.caption := label24.caption
      + Format(' [URL, title=%s]', [(Sender as TDMTextTarget).URLTitle]);

  // The only REQUIRED part of this handler. Do something with the text
  // that was just dropped. If this is missing, the drop won't do anything.
  // You have to decide what you want to do with the text you get!
  memo1.lines.add(dropText);
  // The following line is equivalent to the preceding line.
  // memo1.lines.add((Sender as TDMTextTarget).Text);

end;

procedure TForm1.btnCloseClick(Sender: TObject);
// Close down
begin
  Application.Terminate;
end;

procedure TForm1.DMTextSource2BeforeDrop(Sender: TObject;
  Donor: TComponent; var dropText: String; var cancelDrop: Boolean);
// Called before the drop of text from DMTextSource2. Allow the user to cancel.
// You can modify dropText here if you want. This handler is shared by DMTextSource1
// and DMTextSource2. The particular one in question is identified by looking at Sender.
var
  aString: string;
  showConfirmation: boolean;
begin
  showConfirmation := ((Sender = DMTextSource1) and (CheckBox1.Checked))
    or ((Sender = DMTextSource2) and (CheckBox2.Checked));
  if showConfirmation then
    begin
      if (deMove = (Sender as TDMTextSource).ReturnedEffect) then
        aString := 'Do you really want to move the string "%s"?'
      else
        aString := 'Do you really want to copy the string "%s"?'

      aString := Format(aString, [dropText]);

      cancelDrop := (MessageDlg(aString, mtConfirmation, [mbYes, mbNo], 0) = mrNo);
    end;
end;
```



```

procedure TForm1.DMTextSource2AfterDrop(Sender: TObject; Donor: TComponent;
droppedOK: Boolean);
var
  j: integer;
begin
  // A drop has been done from DMTextSource2. If the drop was a move, we
  // have to remove the text from ListBox2. You have to make sure that
  // droppedOK is true; if you end the drag while the "no drag" cursor is
  // showing, this event will still fire, but with droppedOK = false.
  //
  // In this handler, the expressions (Sender as TDMTextSource).DonorComponent,
  // Donor, and ListBox2 all refer to the same thing.

  if droppedOK then
  begin
    if (deMove = (Sender as TDMTextSource).ReturnedEffect) then
    begin
      // It's a move. Delete from the top down to avoid messing
      // up the Selected[] property.
      for j := ListBox2.items.count downTo 1 do // Iterate
      begin
        if ListBox2.Selected[j-1] then
          ListBox2.items.delete(j-1);
        end; //for
      end;
    end;
  end;

procedure TForm1.DMTextTarget2Drop(Sender: TObject; Acceptor: TWinControl;
  const dropText: String; X, Y: integer);
// Handle drops on ListBox3
//var
// j: integer;
// aSL: TStringList;
begin
  // Bring the form to the top, so that the message dialog won't accidentally
  // be hidden.
  SetForegroundWindow(Handle);

  if CheckBox3.Checked and
    (MessageDlg('Do you want to clear the list before dropping?', mtConfirmation,
      [mbYes, mbNo], 0) = mrYes) then
    ListBox3.Items.Clear;

  // Show a sign that files were dropped (rather than text)
  Label7.visible := ((Sender as TDMTextTarget).DroppedTextFormat = dtfFiles);

  // Handle the drop. Show the coordinates, just for fun.
  ListBox3.Items.Add(Format(' Drop at client pos (%d,%d):', [X, Y]));
  ListBox3.Items.AddStrings((Sender as TDMTextTarget).DroppedLines);

  // The following is equivalent to the preceding line, i.e., DroppedLines
  // and dropText contain the same information.
  //aSL := TStringList.create;
  //try
  // // Get the dropped text into dropText
  // aSL.text := dropText;
  // for j := 1 to aSL.count do // Iterate
  // begin
  //   ListBox3.Items.Add(aSL[j-1]);
  // end; //for
  //finally
  // aSL.free;
  //end;

end;

```

Für die Fortsetzung
laden Sie bitte das Projekt herunter:
<https://raize.com/>



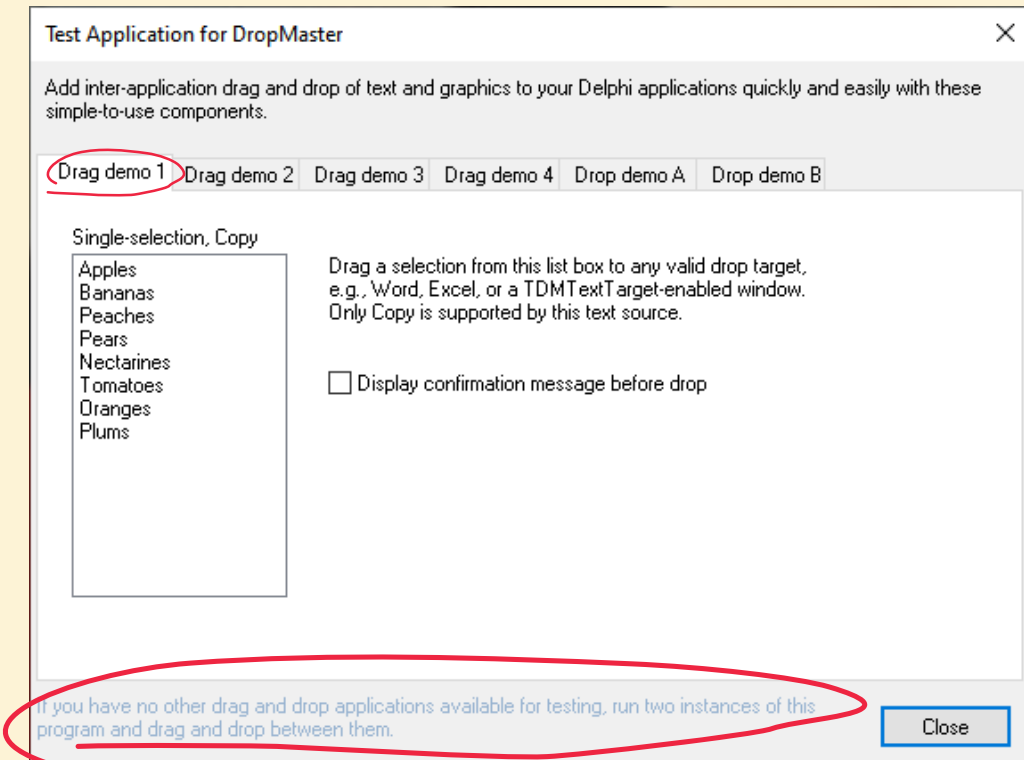


figure 18: Tab 1

If you have no other drag and drop applications available for testing, run two instances of this program and drag and drop between them.

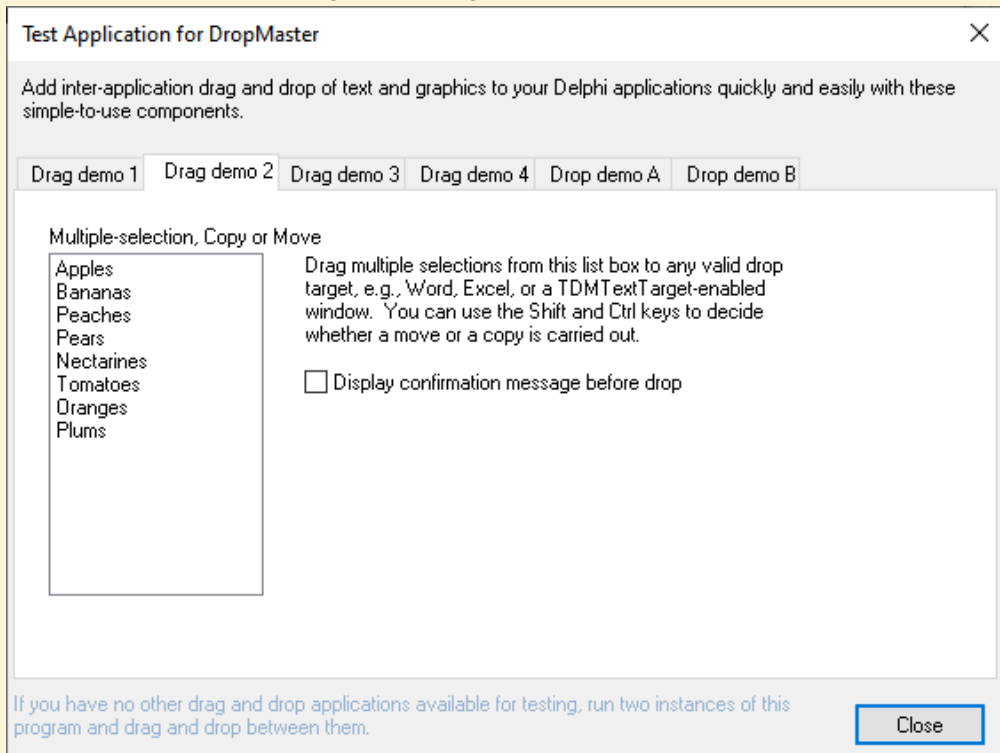


figure 19: Tab2 Interactions are possible



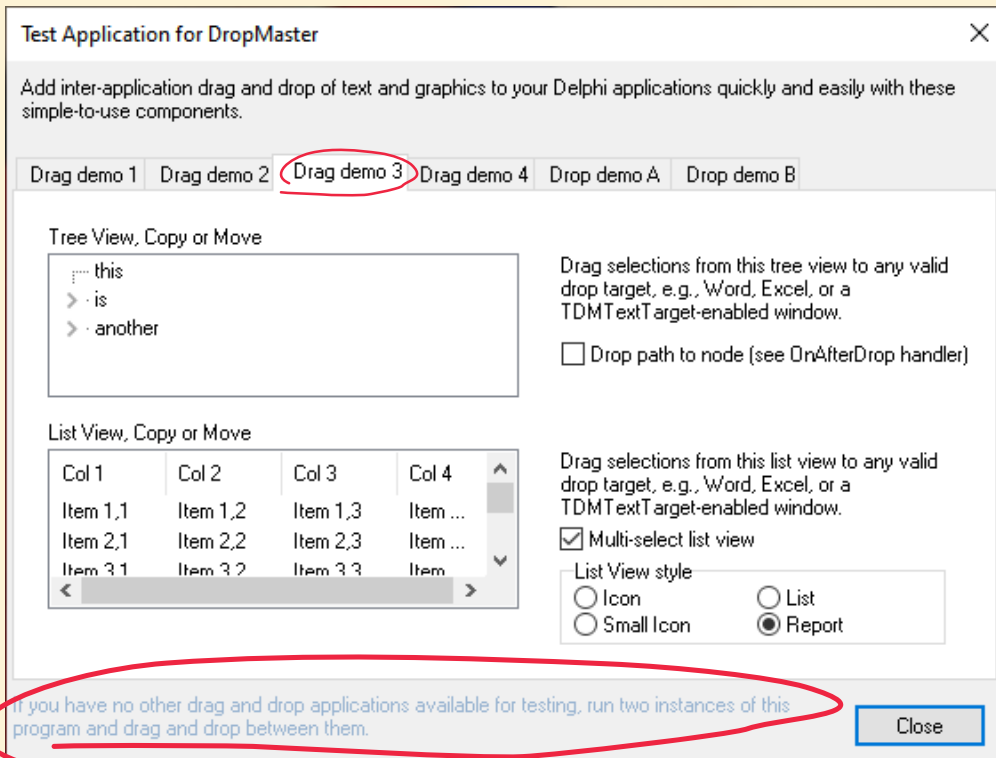


Abbildung 20: Tab 3

Wenn Sie keine anderen Drag&Drop-Anwendungen zum Testen zur Verfügung haben, führen Sie zwei Instanzen dieses Programms aus und ziehen Sie per Drag&Drop zwischen ihnen.

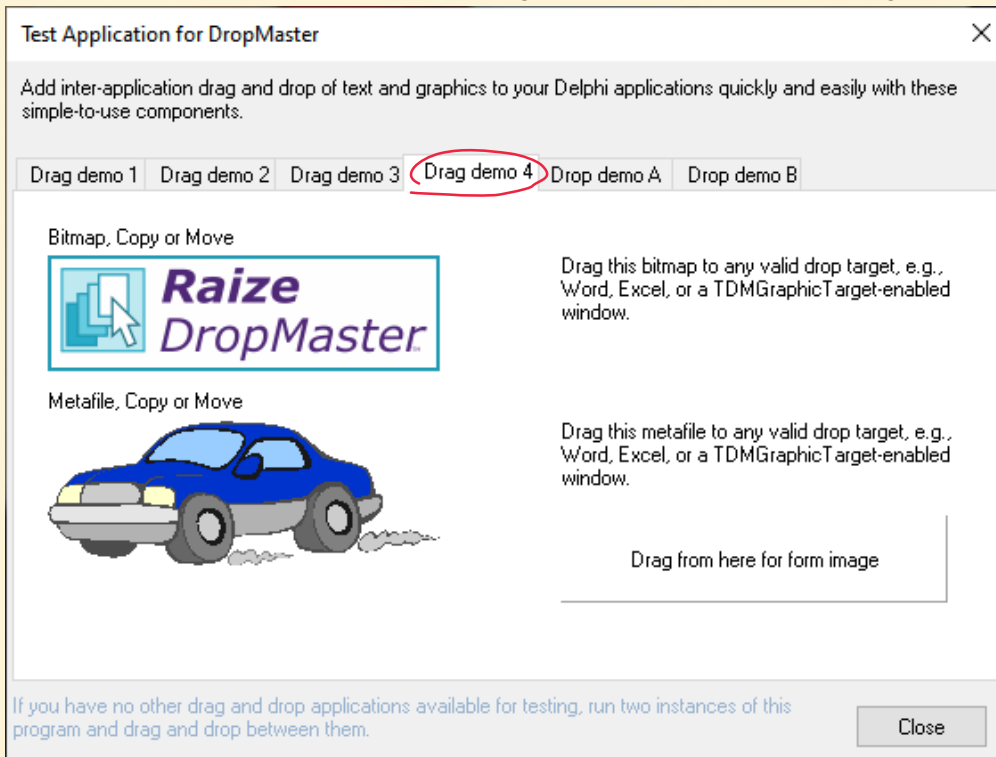


Abbildung 21: Tab 4



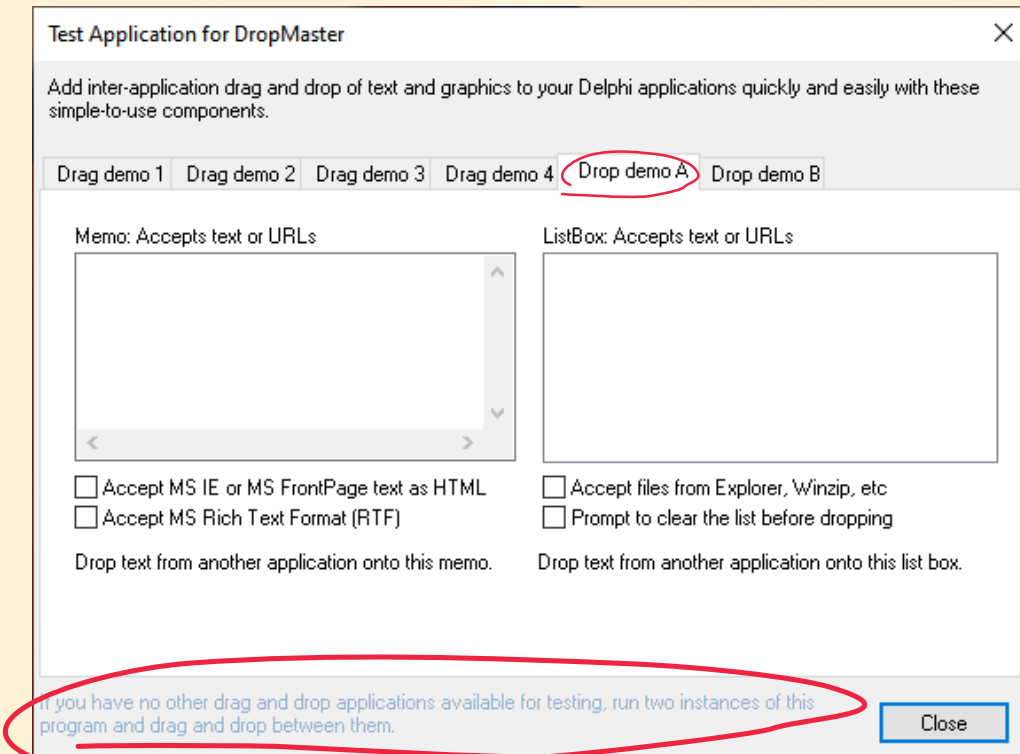


Abbildung 22: Tab A

Wenn Sie keine anderen Drag&Drop-Anwendungen zum Testen zur Verfügung haben, führen Sie zwei Instanzen dieses Programms aus und ziehen Sie per Drag&Drop zwischen ihnen.

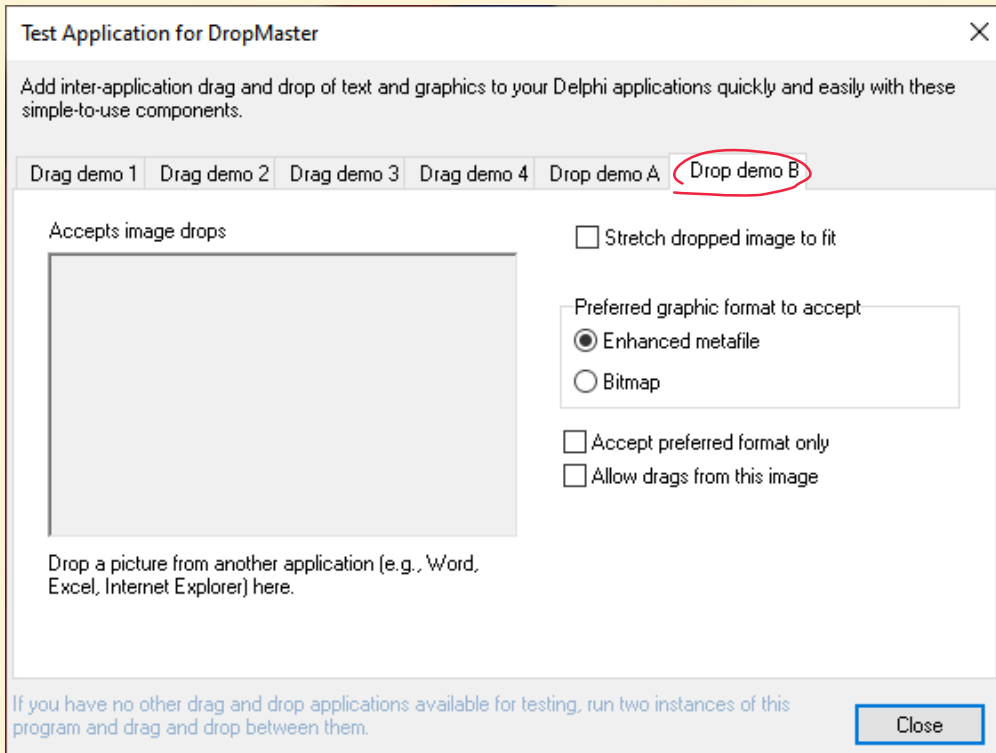


Abbildung 23: Tab B



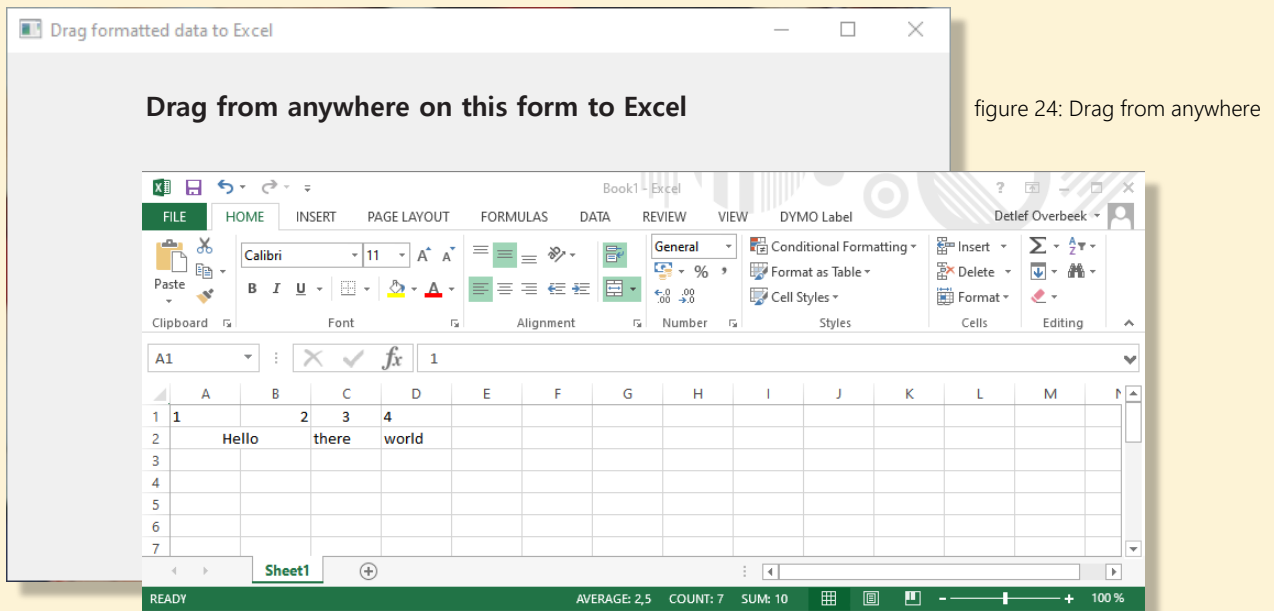


figure 24: Drag from anywhere

Abbildung 25: In das Blatt Te eingefügt

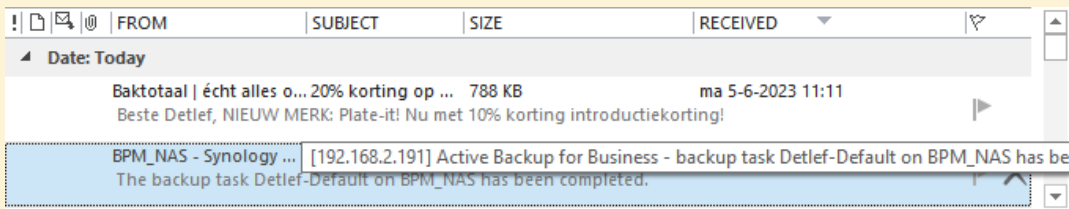


Abbildung 26: Aus einer E-Mail kopiert

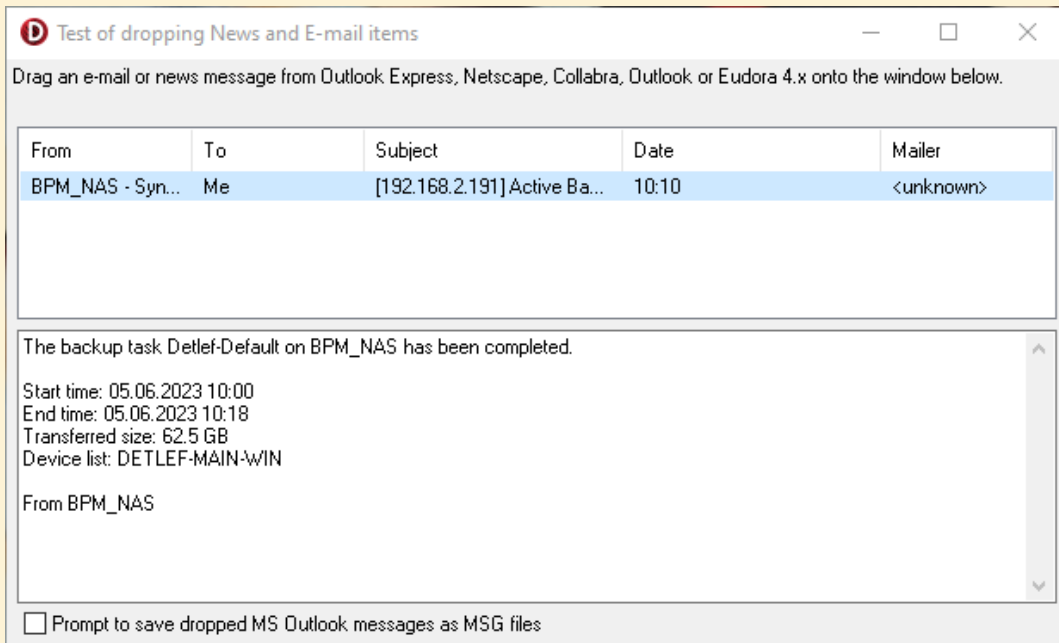


Abbildung 27: In Textfelder



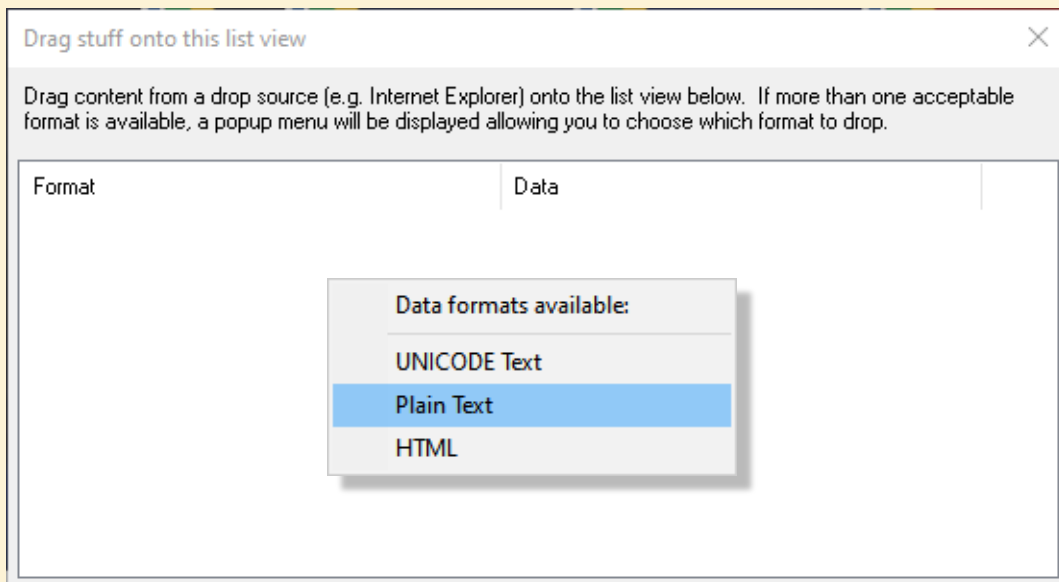
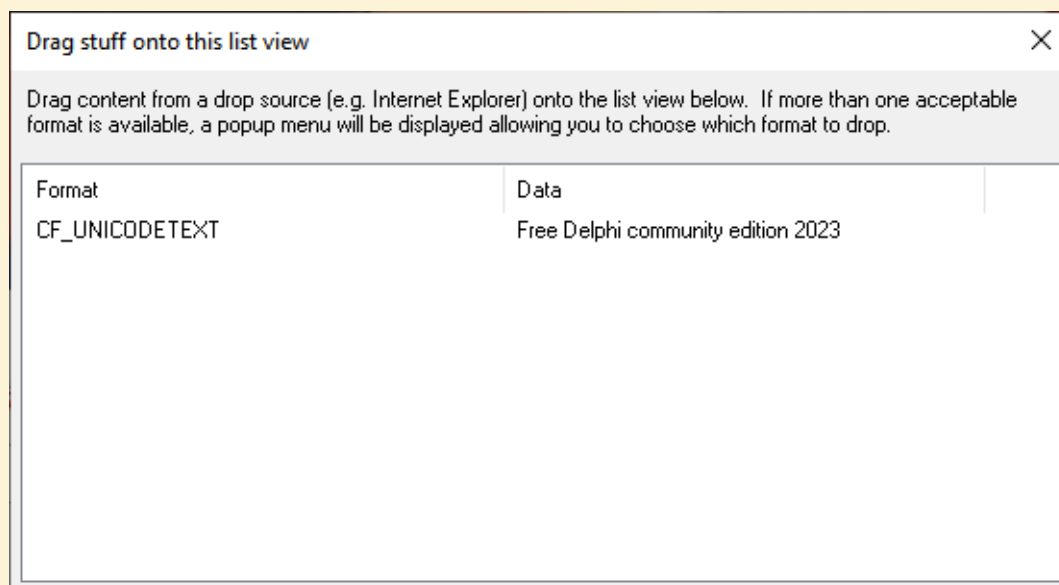


figure 27: Into text fields



CONCLUSION:

All in all: this is a very interesting component group which provides especially for the starter a lot of ease to do things he probably would never had dared to. Ray is always very original and creative in making things easy and better.



LAZARUS-KONFERENZ BACKNANG

in der Nähe von **Stuttgart** vom **22.09.2023 - 24.09.2023**
Die Konferenz findet zum 15. Mal statt.

Willkommen ist jeder, der sich für die Freepascal - IDE Lazarus und/oder eine der anderen Freepascal - IDE's interessiert.
Natürlich sind auch alle anderen Interessierten / Neugierigen, herzlich willkommen.

Die **Lazaruskonferenz** findet in den Räumen des **Technikforum Backnang** (*Wilhelmstraße 32, 71522 Backnang*) statt.

Für Fragen rund um die Konferenz steht das www.lazarusforum.de zur Verfügung.

Konferenz-Beschreibung:

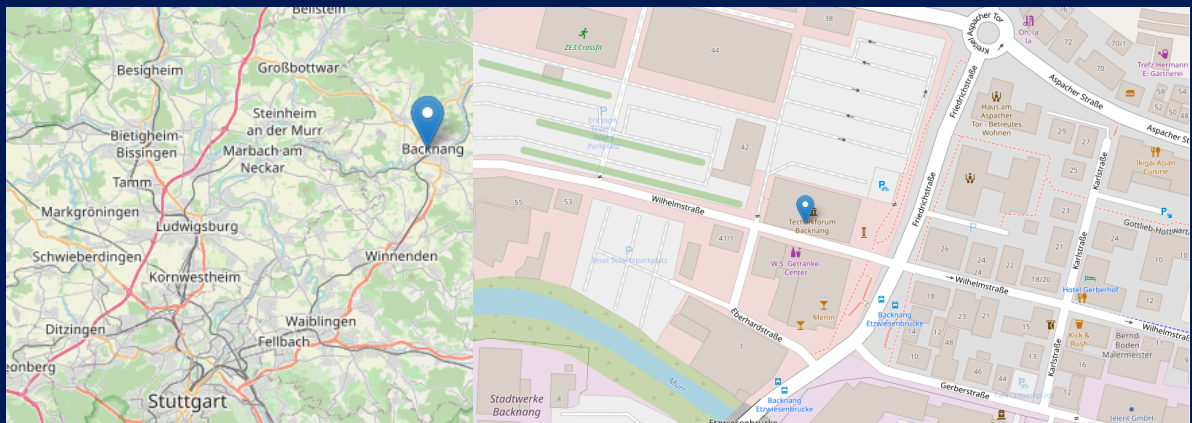
Seit 2008 treffen sich Entwicklerinnen und Entwickler aus dem Umfeld von Lazarus und Freepascal jährlich zum gemeinsamen Austausch über ihre Fachgebiete. Im Mittelpunkt steht dabei die Programmiersprache Pascal, aber auch weitere **Themengebiete aus dem Development-Bereich** sind hier vertreten.

Die Konferenz ist bewusst als **offene Veranstaltung** angelegt und besitzt **keine Rollenaufteilung in „Sprecher“ und „Zuhörer“**.

Die Agenda bestimmen die Teilnehmer selbstständig, anstatt einem festen Programm zu folgen. Neben den eingereichten Vorträgen und Workshops ist immer Zeit für einen freien Austausch, spontane Projektvorstellungen oder auch Hilfestellungen bei offenen Fragen.

Aus den Diskussionen entwickeln sich oftmals neue Projekte oder Kooperationen

- die Interaktion zwischen den Teilnehmern geht weit über das hinaus, was bei reinen **Lehrveranstaltungen** und Vorträgen üblich ist.
- Willkommen sind alle mit einem Interesse an Pascal, FPC oder Lazarus.
Die Teilnahme ist kostenfrei, anmelden könnt ihr euch auf lazarus-konferenz.de.





Neuling

Experte

KURZFASSUNG

Lazarus verfügt über ausgezeichnete Code-Tools. VS Code verfügt über ein Framework zum Hinzufügen von Unterstützung für neue Sprachen. In diesem Artikel zeigen wir, wie erstklassige Pascal-Unterstützung in Visual Studio Code mit Hilfe der Codetools der Lazarus IDE implementiert werden kann.

1 EINFÜHRUNG

Es ist kein Geheimnis, dass die Code-Tools der Lazarus IDE hervorragend sind und sogar die der Delphi IDE übertreffen. Für die Codierung in Object Pascal ist die Lazarus IDE also eine ausgezeichnete Wahl.

Aber manchmal müssen Sie mehr als nur Pascal programmieren. Vielleicht möchten Sie Markdown, HTML, CSS, C bearbeiten oder Makefiles oder Shell-Skripte erstellen. Das können Sie auch im Lazarus-Editor tun, aber dann ist die Unterstützung, die der Editor Ihnen über die grundlegende Bearbeitung hinaus bietet, sehr begrenzt:

Im besten Fall haben Sie Syntax-Highlighting.

Wenn Sie mehr als das wollen, müssen Sie einen anderen Editor für die Bearbeitung öffnen.

Viele moderne Editoren bieten Unterstützung für viele Sprachen: nicht nur Syntaxhervorhebung, sondern auch fortgeschrittenere Funktionen, die man in einem Editor erwartet: Code-Vervollständigung, Vervollständigung von Bezeichnern (Intellisense), Auffinden von Referenzen auf ein Symbol, Refactorings wie das Umbenennen eines Symbols und so weiter. Ein solcher Editor ist Visual Studio Code (eine Weiterentwicklung des inzwischen nicht mehr existierenden Atom-Editors): <https://code.visualstudio.com/>

Er ist sehr populär geworden und verfügt über eine erstaunliche Anzahl von Erweiterungen - darunter mehrere für Pascal.

Der Visual Studio Code-Editor wird von Microsoft verwaltet, und Microsoft hat einen Standard für die Erweiterung seines Editors mit Unterstützung für neue Sprachen eingeführt:

Das Language Server Protocol

<https://microsoft.github.io/language-server-protocol/>
Dieser Standard wurde von mehreren anderen Editoren übernommen, einschließlich Emacs, Vim, Delphi, Sublime Text, IntelliJ und der KDE-Editorsuite (Kate & KDevelop). Eine ausführlichere Liste finden Sie hier:

<https://microsoft.github.io/language-server-protocol/implementors/tools/>

Leider ist die Lazarus IDE nicht in dieser dieser Liste, da sie das LSP-Protokoll jetzt erst unterstützt.

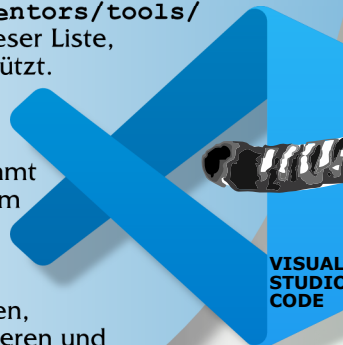
LSP-Protokoll unterstützt:

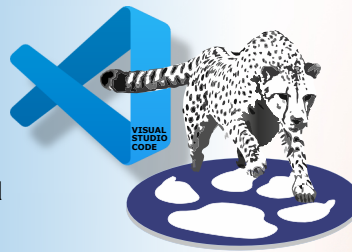
Es wird die Verwendung jeder Sprache in der der Lazarus IDE zu verwenden.

Wenn der Berg nicht zu Mohammed kommt, muss Mohammed zum Berg gehen:

Die Unterstützung für das LSP-Protokoll in Lazarus steht noch aus, jetzt können die Lazarus Code Tools verwendet werden, um das das LSP-Protokoll zu implementieren und andere Editoren zu erweitern mit Erstklassige Pascal Unterstützung.

Vielleicht möchten Sie Markdown, HTML, CSS, C bearbeiten oder Make-Dateien oder Shell- Skripte erstellen.





Mehrere LSP-Implementierungen unter Verwendung der Lazarus Codetools sind auf Github verfügbar, aber in diesem Artikel werden wir uns auf eine konzentrieren:

<https://github.com/genericptr/pascal-language-server>

② DAS LSP PROTOCOL

Das Language Server-Protokoll basiert auf einem JSON-RPC-Kommunikationsmechanismus.

Der Editor startet ein Programm, das als Language Server fungiert, und sendet JSON-RPC Nachrichten über die Standardeingabe an den Prozess. Er liest die Ergebnisse und mögliche Befehle vom LSP-Server über die Standardausgabe. Dieser Austausch sieht wie folgt aus. Der Editor sendet eine Anfrage:

```
{
  "jsonrpc" : "2.0",
  "method" : "textDocument/didOpen",
  "params" : {
    "textDocument" : {
      "uri" : "file:///home/michael/source/testio/testio.lpr",
      "languageId" : "pascal",
      "version" : 1,
      "text" : "program testio;\n\n ... end.\n"
    }
  }
}
```

In diesem Fall antwortet der Server mit einem Befehl:

```
{
  "jsonrpc" : "2.0",
  "method" : "textDocument/publishDiagnostics",
  "params" : {
    "diagnostics" : [],
    "uri" : "file:///home/michael/source/testio/testio.lpr"
  }
}
```

Die vollständige Liste der Befehle, die ein Server implementieren kann, ist in dem LSP-Protokoll dokumentiert (*siehe die obige URL*). Wenn der Editor den Server startet, wird ein Handshake durchgeführt: der Befehl `initialize`.

Im Initialisierungsbefehl (*dem ersten Befehl, den der Editor an den Server sendet*) gibt der Client (*der Editor*) an, über welche Fähigkeiten er verfügt, und der Sprachserver antwortet mit den Fähigkeiten, über die er verfügt:

Dies ist normalerweise eine Liste von 'Providern' bestimmter Funktionen.

Dieser Handshake ist wichtig, denn nicht alle Server unterstützen alle Befehle und nicht alle Clients unterstützen alle Befehle.





Ein Server kann auch benutzerdefinierte Befehle angeben: Dies sind Befehle, die nicht Teil des LSP Protokolls sind, für die das LSP-Protokoll aber einen speziellen Befehl mit der treffenden Bezeichnung 'executeCommand' zur Verfügung steht. Der oben erwähnte Pascal Language Server implementiert verschiedene der vom Language Server Protokoll erwarteten Funktionalitäten:

<code>textDocument/declaration</code>	Springe zur Deklaration
<code>textDocument/implementation</code>	Springe zur Implementierung
<code>textDocument/references</code>	Verweise auf ein Symbol finden
<code>textDocument/signatureHelp</code>	Funktions- oder Methodensignatur (Parameter) anzeigen.
<code>textDocument/documentSymbol</code>	Liste der Symbole in einem Projekt.
<code>textDocument/documentHighlight</code>	Ähnlich wie bei <code>textDocument/references</code> finden Sie Referenzen auf ein Symbol.
<code>textDocument/vervollständigung</code>	Identifizierende Vervollständigung
<code>textDocument/hover</code>	Intelligente Hinweise zu Ihrem Code
<code>window/showMessage</code>	erlaubt es dem LSP-Server, eine Nachricht im Editor anzuzeigen.
<code>workspace/symbol</code>	Liste aller Symbole, die zu einem Textstück passen.
<code>workspace/executeCommand</code>	Führen Sie einen benutzerdefinierten Befehl aus.
<code>diagnostics</code>	Ermöglicht es dem Server, eine Liste von Diagnosen an den Client zu senden: Dies können Warnungen, Fehler usw. sein. Sie werden im Editor angezeigt (unter 'Probleme' in VS Code)

Zusätzlich implementiert der Pascal Language Server einige benutzerdefinierte Befehle:

<code>pasls.completeCode</code>	Codevervollständigung: vervollständigt die aktuelle Klasse, definiert eine Variable usw. Das Äquivalent zur Code-Vervollständigung in der IDE.
<code>pasls.formatCode</code>	ruft den Jedi-Code-Formatierer für die Pascal-Datei auf.
<code>pasls.invertAssignment</code>	ein Refactoring, das die Zuweisungsanweisungen invertiert in der Auswahl umkehrt.
<code>pasls.removeEmptyMethods</code>	ein Refactoring, das alle leeren Methoden aus der aktuellen Datei entfernt.

More custom methods are being added to the server: theoretically, all code tools offered by Lazarus can be implemented.

What does the language server not do ? It does not compile the pascal code, it also does not offer functionality to edit form files. There are also several commands in the LSP that it does not implement. It also does not do syntax highlighting. (The 'pascal magic' extension in the VS extension marketplace does this for you).

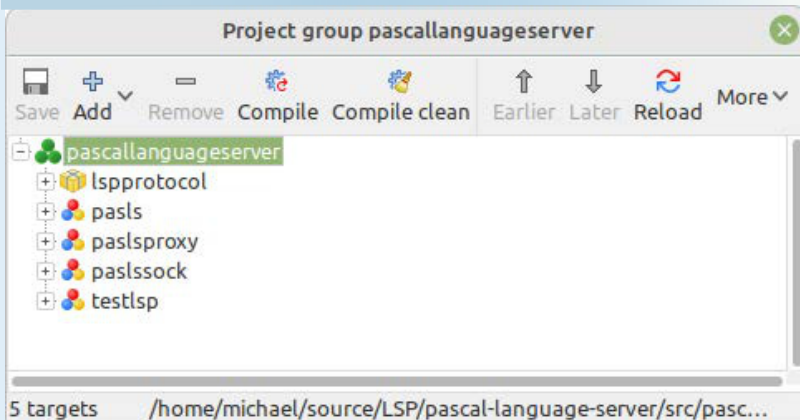
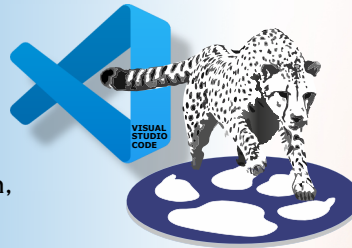


Abbildung 1: Die Projektgruppe Sprachserver





③ VERWENDUNG DES LSP-SERVERS

Um den LSP-Server in VS Code zu verwenden, sind 2 Dinge erforderlich:

- Kompilieren Sie den LSP-Server.
- Installieren Sie eine Erweiterung in VS Code, die den LSP-Server registriert, und die von ihm zur Verfügung gestellten zusätzlichen Befehle registriert.

Um den LSP-Server zu kompilieren, können Sie das offizielle Repository unter der obigen URL klonen. Wenn Sie dies tun, erhalten Sie unterhalb des Src-Verzeichnisses eine Projektgruppendatei `pascalanguageserver.lpg` mit 4 Projekten und einem Lazarus-Paket (siehe Abbildung 1 auf Seite 3 dieses Artikels):

`lspprotocol.lpk`

Ein Package mit den Units, die das Protokoll des Sprachenservers bilden. Es hängt von dem Lazarus Codetools Package ab.

`pasls.lpi`

Das eigentliche LSP-Serverprogramm. Dies ist das Programm, das Sie kompilieren und verwenden müssen. Um es zu kompilieren, müssen Sie zuerst das `lspprotocol` Package öffnen und kompilieren.

`paslsproxy.lpi`

Ein Proxy-Programm, das das JSON-RPC-Protokoll auf Standard-Ein-/Ausgabe implementiert und die Nachrichten über einen TCP/IP-Socket unter Verwendung eines speziellen Hochgeschwindigkeits-Nachrichtenschemas weiterleitet.

`paslssock.lpi`

Ein Sprachserverprogramm, das an einem TCP/IP-Socket lauscht und das JSON-RPC-Protokoll unter Verwendung desselben Nachrichtenschemas wie `paslsproxy` implementiert.

`testlsp.lpi`

Ein minimales Unit-Testprogramm.

Die Programme `paslsproxy` und `paslssock` werden nur zum Debuggen des Sprachprozess-Servers verwendet: Da der Editor den LSP-Prozess startet, ist es schwierig, das Starten und den Nachrichtenfluss zu debuggen. Wenn Sie `paslssock` im Lazarus Debugger ausführen und den Editor das Programm `paslsproxy` starten lassen, können Sie den Sprachserverprozess debuggen. Für den normalen Gebrauch benötigen Sie jedoch nur das Programm `pasls`, das Sie kompilieren sollten. Es wird auf allen Plattformen kompiliert, die Lazarus unterstützt.

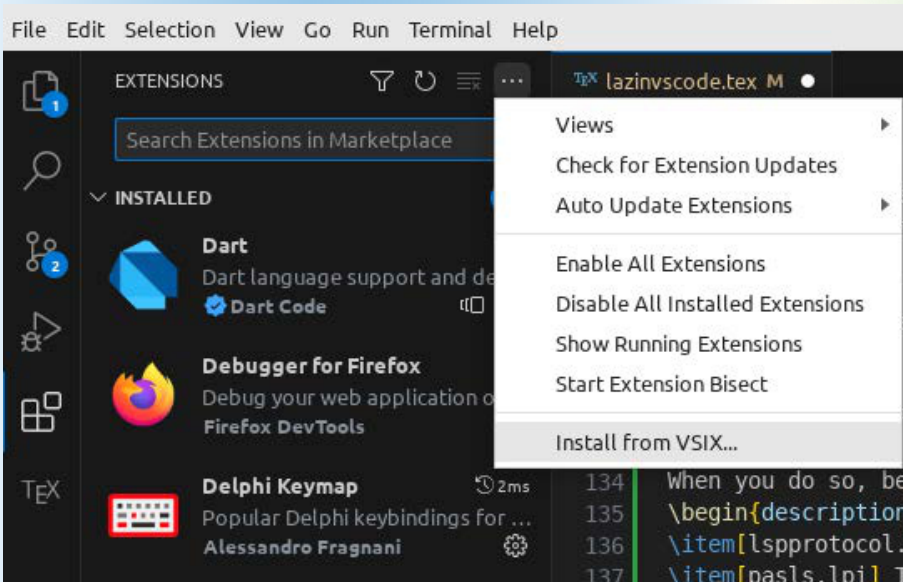
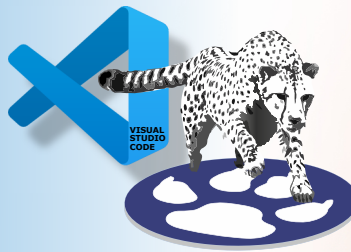


Abbildung 2: Das VSIX-Installationsmenü in VS Code





Sie können es mit der veröffentlichten Version von Free Pascal (3.2.2) oder mit der Entwicklungsversion von git kompilieren.

Wenn Sie Letzteres tun, haben Sie eine bessere Syntaxprüfung aufgrund von Verbesserungen im PAS2JS-Parser (*der für die Syntaxprüfung verwendet wird*).

Um die Pasls-Binärdatei zu erstellen, öffnen Sie das Projekt in der Lazarus IDE und drücken Sie die Tastenkombination Compile oder verwenden Sie den Menüpunkt Run-Compile in Lazarus.

Sie können auch versuchen, die pasls-Binärdatei zu kompilieren, ohne Lazarus installiert zu haben.

Sie können die Lazarus Quellen aus dem Git Repository auschecken unter <https://gitlab.com/freepascal.org/lazarus/lazarus>

Wenn Sie dies tun, müssen Sie die Pfade zum lazarus codetools Package und allen anderen Packages angeben, von denen letzteres abhängt:

- codetools (*Komponenten/codetools*)
- jcfbase (*components/jcf2*) - der Jedi Code Formatierer.
- lazutils (*Komponenten/lazutils*)

Sobald Sie eine pasls-Binärdatei haben, können Sie sie in einem Editor verwenden. Für VS Code gibt es ein Erweiterungspaket auf github:

<https://github.com/genericptr/pasls-vscode>

Sie können die Erweiterung in VS Code selbst paketieren und installieren, aber eine .vsix-Paketdatei ist verfügbar.

Dies ist ein Erweiterungspaket für Visual Studio Code, das über einen Menüpunkt installiert werden kann:

In der Registerkarte Erweiterungen auf der linken Seite der IDE enthält das Menü am oberen Rand einen Eintrag 'Von VSIX installieren' (siehe Abbildung 2 auf Seite 4 dieses Artikels). Damit können Sie die .vsix Datei auswählen und installieren.

Nach der Installation müssen einige Einstellungen in der VS Code IDE konfiguriert werden (*siehe Abbildung 3 auf Seite 5*).

Wichtige Einstellungen, die für die korrekte Funktion der Codetools erforderlich sind, sind die folgenden: (*Fahren Sie mit Seite 7 dieses Artikels fort*)



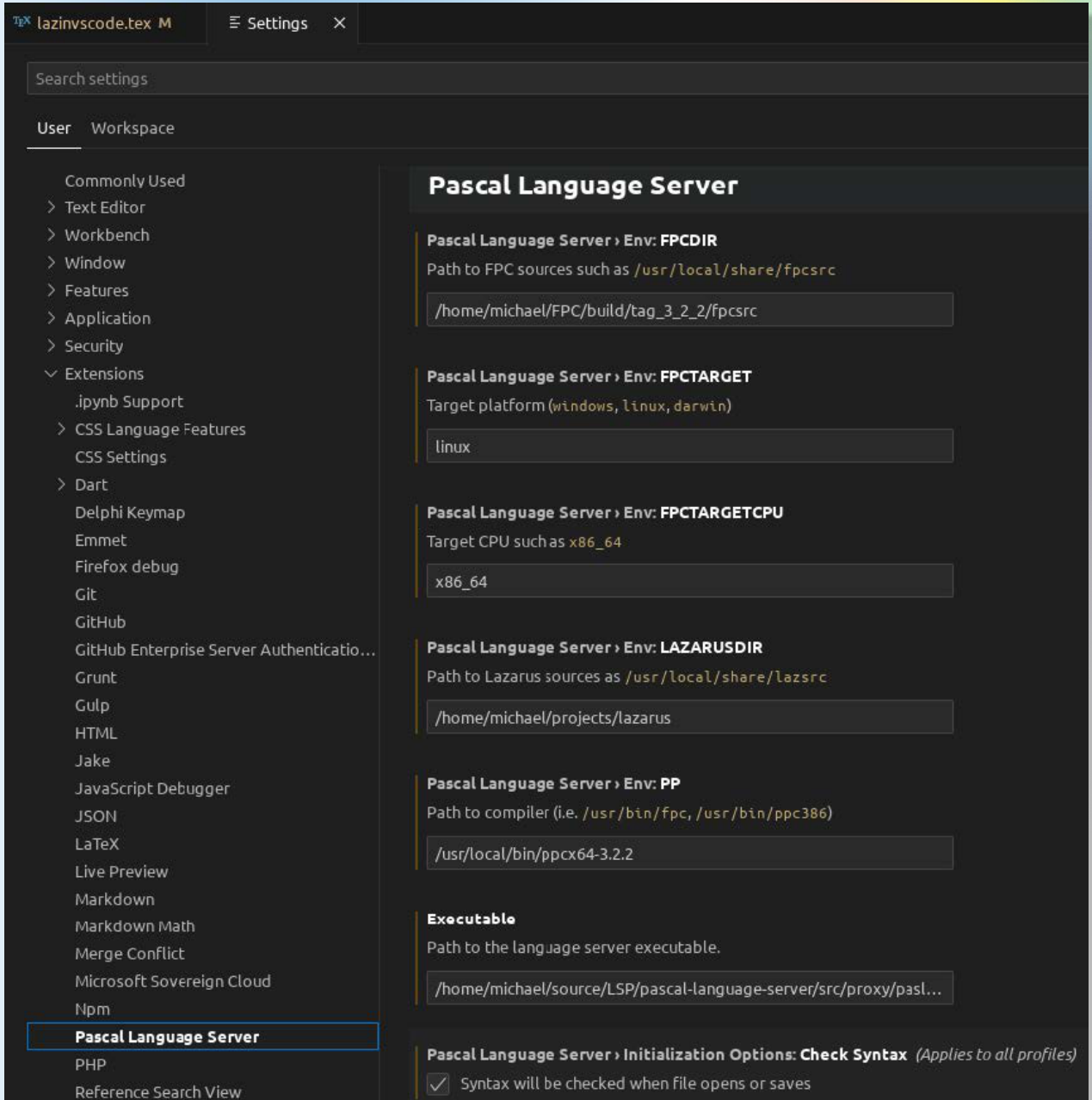
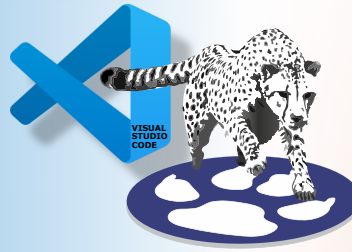
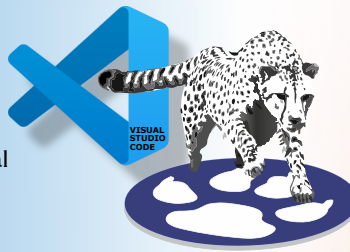


Abbildung 3: Die Einstellungen des Pascal Language Server





Env:FPCDIR

Das Verzeichnis, in dem sich die FreePascal Quellen gespeichert sind.

Env:FPCTARGET

Das Zielbetriebssystem.

Env:FPCTARGETCPU

Die Ziel-CPU.

Env:LAZARUSDIR

Das Verzeichnis, in dem sich die Lazarus-Quellen befinden (nur erforderlich, wenn Sie an Programmen arbeiten, die einige Lazarus-Pakete verwenden)

Env:PP

Der Pfad zum Free Pascal Compiler Binary: die Codetools verwenden dies, um einige Compilerinformationen zu erhalten.

EXECUTABLE

Der Pfad zu der von Ihnen kompilierten Pals-Binärdatei.

FORMAT Config

Der Pfad zu der Konfigurationsdatei für den Code-Formatierer. Diese Datei kann aus der Einstellungsdatei einer Lazarus-Installation kopiert werden (*die Datei heißt jcfsettings.cfg*). Darüber hinaus gibt es Optionen, die das Verhalten des Sprachservers selbst steuern, sie sind Teil der Initialisierungsoptionen:

SYNTAX PRÜFEN

Wenn diese Option aktiviert ist, prüft der Sprachserver die Syntax der aktuellen Datei, wenn Sie sie speichern.

DOKUMENTENSYMBOLLE

Wenn diese Option aktiviert ist, ist eine Abfrage nach Dokumentensymbolen möglich.

FPC-OPTIONEN

dies sind Optionen, die Sie normalerweise beim Kompilieren Ihres Projekts angeben würden: die Codetools analysieren diese Optionen, um Defines usw. zu bestimmen.

ARBEITSBEREICH-ORDNER als Include-Pfade einbeziehen

Wenn diese Option aktiviert ist, werden alle Unterverzeichnisse des VS Code-Arbeitsbereichsverzeichnisses als Include-Pfade verwendet (*die Befehlszeilenoption -Fu des Compilers*).

WORKSPACE-ORDNER als Unit-Pfade einbeziehen

Wenn diese Option aktiviert ist, werden alle Unterverzeichnisse des VS Code-Arbeitsbereichsverzeichnisses als Include-Pfade verwendet (*die Befehlszeilenoption -Fu des Compilers*).

VERVOLLSTÄNDIGUNGSPROZEDUR-Klammern einfügen

Wenn diese Option markiert ist und Sie einen Prozeduraufruf vervollständigen, werden an den Prozeduraufruf () Klammern angehängt (*auch wenn keine Parameter erwartet werden*).

VERVOLLSTÄNDIGUNGEN ALS SNIPPETS EINFÜGEN

Wenn diese Option aktiviert ist und Sie einen Prozeduraufruf vervollständigen, wird der Prozeduraufruf als Snippet eingefügt: Er enthält einen Cursor-Platzhalter für den Parameter (z.B. ' (\$0) ').

MAXIMALE VERVOLLSTÄNDIGUNGEN

die maximale Anzahl der möglichen Vervollständigungen, die angezeigt werden sollen.

MINIMALISTISCHE VERVOLLSTÄNDIGUNGEN

Minimalistische Vervollständigungen: Es wird nur der Name des Symbols angezeigt, nicht die Art des Symbols.

ÜBERLADUNGSRICHTLINIE

bestimmt, wie Überladungen in der Symbolliste behandelt werden.

Ein numerischer Wert mit den folgenden Bedeutungen:

- - Doppelte Funktionsnamen werden in der Liste angezeigt
- - Überladungen werden ignoriert, nur die erste wird verwendet.
- - Fügen Sie ein Suffix hinzu, das die Anzahl der Überladungen angibt





Programm Die Hauptprogrammdatei:

Diese Datei wird von den Codetools verwendet, um festzustellen, welche Einheiten Teil des Projekts sind, und um Referenzen zu finden.

Diagnose veröffentlichen

Wenn die Codetools einen Fehler im Sprachserver zurückgeben, wird dieser Fehler als Diagnose gemeldet.

Syntaxfehler anzeigen

Wenn während einer Syntaxfehlerprüfung Syntaxfehler auftreten, werden diese als kleine Fenster im Editor angezeigt.

Symbol-Datenbank

eine SQLite-Datenbank, die für Symbole verwendet werden soll: Diese Datenbank wird erstellt und mit Symbolen gefüllt. Sie wird dann als Cache verwendet: Anstatt die Dateien zu analysieren, wird stattdessen der Inhalt des Caches angezeigt.

Wenn Sie mit den Einstellungen fertig sind, können Sie loslegen.

Wenn Sie ein Pascal-Projekt in VS Code öffnen, können Sie im Ausgabefenster (*Abbildung 4 auf Seite 8 dieses Artikels*) sehen, dass der Pascal-Sprachserver korrekt gestartet wurde, wenn Sie das Werkzeug 'Pascal-Sprachserver' auswählen (siehe Bild, das rote Rechteck oben rechts).

Wenn der Pascal-Sprachserver korrekt initialisiert wurde, können Sie in VS Code mit der erweiterten Bearbeitung von Codes beginnen, wie in *Abbildung 5 auf Seite 9* zu sehen ist.

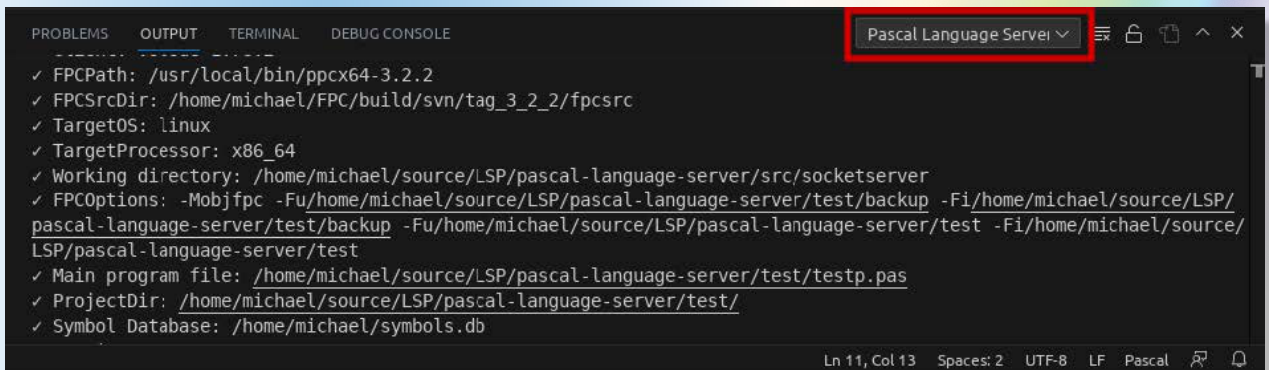


Abbildung 4: Die Ausgabe des Pascal Language Server beim Starten

4 AUSFÜHREN VON BENUTZERDEFINIERTEN BEFEHLEN

Um den Codeformatierer auszuführen, rufen Sie die übliche Codeformatierungsanfrage in VS Code auf: Die Erweiterung leitet diese Anfrage an den Sprachserver weiter. Die Standard-Tastenkombination hierfür ist **ctrl-shift-i**.

Der Code-Formatierer verwendet eine Konfigurationsdatei. Sie können den Speicherort der Konfigurationsdatei in den Einstellungen festlegen. Bei der Konfigurationsdatei handelt es sich um eine XML-Datei.

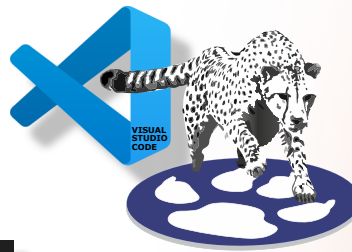
Eine Beispieldatei wurde in das Github-Repository des Pascal-Sprachservers aufgenommen. Die meisten Einstellungen sind selbsterklärend, so dass eine Bearbeitung der XML-Datei möglich ist, aber derzeit lässt sich die Konfigurationsdatei am einfachsten im Lazarus-Dialog '**Extra - Options**' bearbeiten.

Die Code-Vervollständigungsfunktion von Lazarus ist der Standard-Tastenkombination von Lazarus zugeordnet: **ctrl-shift-c**. Sie können aber auch einfach '**Codevervollständigung**' in die Befehlspalette eingeben und sie so aktivieren.

Um die anderen Befehle auszuführen (*leere Methoden entfernen oder Code-Formatierer*), rufen Sie die Befehlspalette auf (**ctrl-shift-;**) und geben die Beschreibung des Befehls ein.

So wird z.B. **entfernen** zu einer Liste führen, wie sie in *Abbildung 6 auf Seite 9* dieses Artikels zu sehen ist. In einer späteren Version des Sprachservers werden diese Befehle dem Refactoring-Menü hinzugefügt.





```
testp.pas > ...
10
11 Procedure Test(a : String);
12
13 var
14   C : Integer;
15
16 begin
17   Writeln(a);
18 end;
19
20 begin
21   Test('a');
22   Test('A');
23   Test('A');
24   Test(a: String)
25   Test()
26 end.
```

Abbildung 5: Ein hilfreicher Hinweis auf die Funktion, die Sie gleich aufrufen werden.

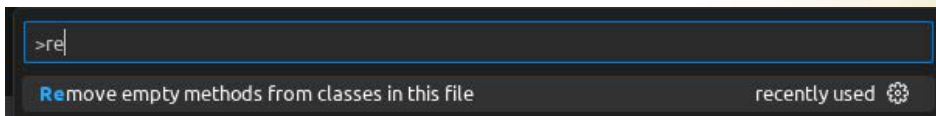


Abbildung 6: Aufrufen des Befehls 'Leere Methoden entfernen'.

6 SCHLUSSFOLGERUNG
Der PASCAL LANGUAGE SERVER ermöglicht es Ihnen, einen Teil der Werkzeuge, die die Lazarus IDE bietet, direkt in VS Code nutzen. Der Pascal Language Server ist nicht auf die Verwendung in VS Code beschränkt. Er kann mit allen Editoren verwendet werden, die den LSP-Prozess unterstützen: Einer der Autoren verwendet Sublime Text, um Pascal zu entwickeln. Der Pascal-Sprachserver befindet sich in aktiver Entwicklung, so dass in naher Zukunft weitere Leckerbissen aus den Lazarus Code Tools in naher Zukunft verfügbar gemacht werden.





Features Full-Featured Free Delphi IDE for Creating Native Cross-Platform Apps
<https://www.embarcadero.com/products/delphi/starter/free-download>

Delphi Community Edition (CE) ist eine voll funktionsfähige IDE für die Erstellung von iOS-, Android-, Windows- und macOS-Apps aus einer einzigen Delphi-Codebasis (begrenzte kommerzielle Nutzungslizenz). Delphi CE wird kostenlos an unsere Community aus freiberuflichen Entwicklern, Startups, Studenten und gemeinnützigen Organisationen weitergegeben. Delphi CE enthält einen Code-Editor, leistungsstarke Debugging-Tools, integrierten Zugriff auf beliebige lokale Datenbanken mit Live-Daten zur Entwurfszeit, Bluetooth-Funktionen und einen visuellen UI-Designer mit Unterstützung für pixelgenaues, plattformspezifisches Styling.

ERLÄUTERUNG:

Können Sie die Edition ohne Probleme erhalten?

Wenn Sie eine Einzelperson sind, können Sie Delphi CE verwenden, um Anwendungen für Ihren eigenen Gebrauch und Anwendungen zu erstellen, die Sie verkaufen können, bis Ihr Umsatz 5.000 US-Dollar pro Jahr erreicht. Wenn Sie ein kleines Unternehmen oder eine Organisation mit einem Jahresumsatz von bis zu 5.000 US-Dollar sind, können Sie Delphi CE ebenfalls verwenden. Sobald der Gesamtumsatz Ihres Unternehmens 5.000 US-Dollar erreicht oder Ihr Team auf mehr als fünf Entwickler angewachsen ist, können Sie auf eine uneingeschränkte kommerzielle Lizenz mit der Professional Edition umsteigen.

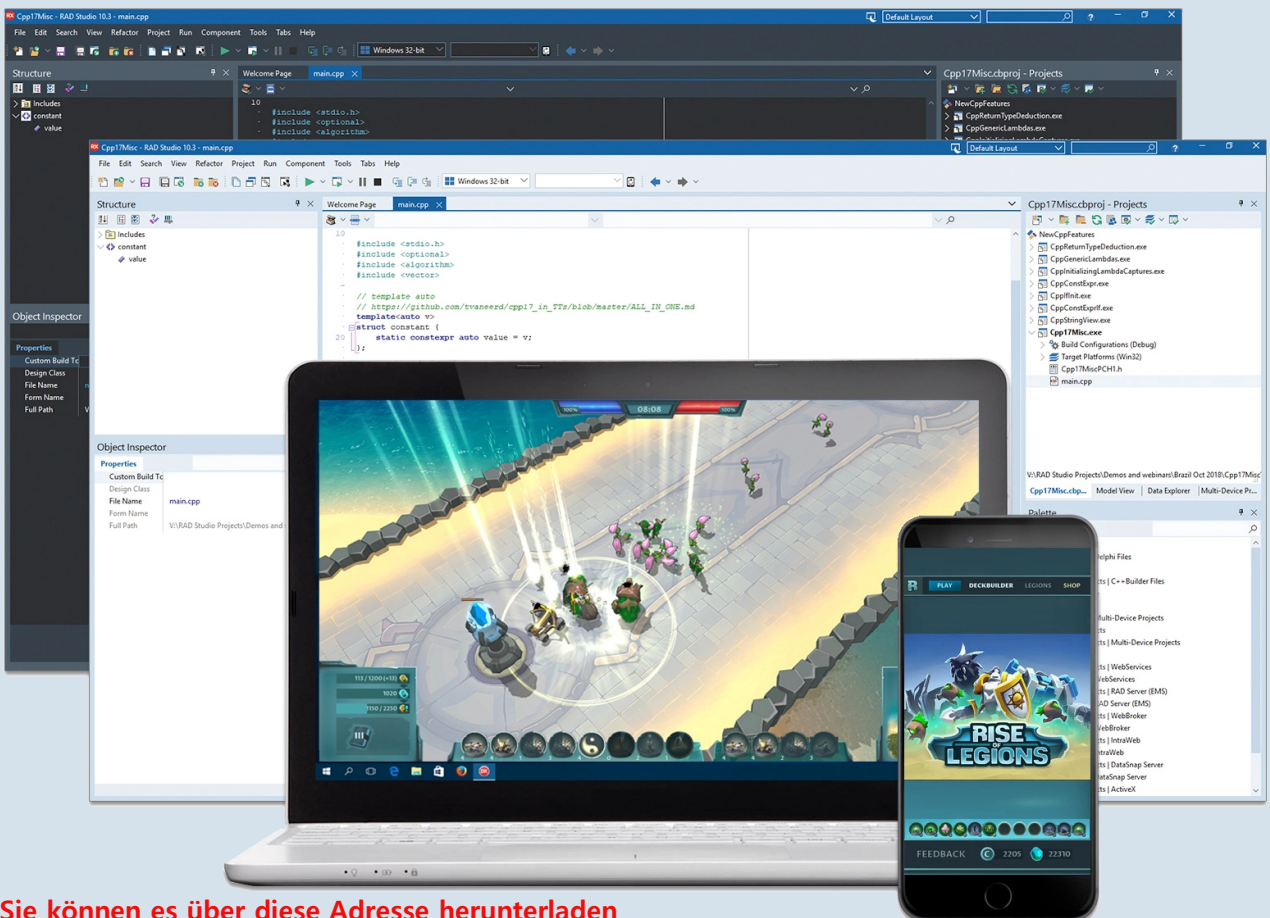
Delphi CE ist auch ideal für Startups in der Frühphase, die ihre Produktvision noch vor der Beschaffung von Kapital umsetzen wollen! Entwickeln Sie Ihre professionelle Anwendung mit der Community Edition in der Gewissheit, dass Sie die Lernkurve überspringen können, die Ihre Konkurrenz bei der Entwicklung für mehrere Plattformen durchlaufen muss.

Weitere Einzelheiten finden Sie in den FAQ's zur Community Edition.

Delphi ist in den Editionen Community, Academic, Professional, Enterprise und Architekt erhältlich.

Einzelheiten zu den Unterschieden zwischen den Editionen finden Sie auf der Seite Produktditionen und in der Feature-Matrix.

Wenn Sie auf die Professional Edition oder höher aufsteigen, erhalten Sie zusätzliche Funktionen wie Komponenten und Treiber für die Datenbankanbindung, eine vollständige kommerzielle Entwicklungslizenz und vieles mehr.



Sie können es über diese Adresse herunterladen
<https://www.embarcadero.com/products/delphi/starter/free-download>





Funktionen

Kostenlose Delphi IDE mit vollem Funktionsumfang für die Erstellung nativer, plattformübergreifender Anwendungen

Erstellen Sie native Windows-Anwendungen mit dem High-Performance UI Framework und Komponenten (VCL)

Visual Component Library (VCL) ist ein visuelles, komponentenbasiertes, objektorientiertes Framework für die Entwicklung von Benutzeroberflächen für Windows-Anwendungen. Es bietet eine Reihe von visuellen und nicht-visuellen Komponenten, um eine optimale Leistung und ein plattformspezifisches Benutzererlebnis auf dem Windows-Betriebssystem zu erreichen.



Erstellen Sie Mobile First, plattformübergreifende Apps mit Native Experience UI Framework (FMX) und Komponenten (iOS, Android, macOS, Windows)

FireMonkey (FMX) ist ein visuelles Komponenten-Framework, das intelligente Stile und Plattformdienste verwendet, um die Benutzeroberfläche einmal zu entwerfen und sie dann an jede Plattform anzupassen, so dass Sie mit demselben Code mehrere Plattformen ansprechen können, einschließlich der Anwendungslogik und der Benutzeroberfläche.



Quellcode der Runtime Library

Enthält den Quellcode für die VCL, FMX und die meisten anderen Bibliotheken, von denen Sie lernen oder die Sie mit Ihrem eigenen Code erweitern können

Eingeschränkte Nutzung

Volle Lizenz für kommerzielle Nutzung

In der EULA finden Sie die vollständigen Lizenzbedingungen für jede Edition.

Eingeschränkte Kommerzielle Nutzung

Stellen Sie mit FireDAC eine Verbindung zu lokalen Datenbanken her und erstellen Sie datengestützte Anwendungen mit Unterstützung für mehrere Datenquellen

FireDAC lokale/eingebettete Konnektivität zu bestimmten lokalen Datenbanken, einschließlich Microsoft Access Datenbank, SQLite Datenbank, InterBase ToGo / IBLite, InterBase auf localhost, MySQL Embedded, MySQL Server auf localhost, Advantage Database local engine, PostgreSQL auf localhost, Firebird Embedded und Firebird auf localhost.



InterBase Embedded Database

InterBase ist eine preisgekrönte, leistungsstarke SQL-Datenbank mit zahlreichen fortschrittlichen Funktionen, einschließlich Unternehmenssicherheit, Änderungsansichten, Warnmeldungen, Generatoren und mehr. Es gibt 2 eingebettete Versionen, IBLite und IBToGo, die Verschlüsselungsbeschränkung und zusätzliche Funktionen bieten.



IBLite Mobile Deployment

Nachfolgend finden Sie eine Online-Youtube-Videoadresse, die Ihnen bei der Installation von Delphi Win 11 CE hilft:

<https://www.youtube.com/watch?v=kjP680w1j-M>



Jim McKeeth has left Embarcadero



Jim McKeeth left and his successor Ian Barker Right

Wenn Sie sich Online-Inhalte von Embarcadero angesehen haben, an einem RAD Studio-Webinar teilgenommen haben, oder an einer der persönlichen Veranstaltungen teilgenommen haben, werden Sie höchstwahrscheinlich den wunderbaren Jim McKeeth kennen.

Jim ist seit dem 13. Juli 2013 Chief Developer Advocate und Engineer bei Embarcadero, also seit knapp zehn Jahren.

Heute jedoch kommt die große Nachricht, dass Jim Embarcadero verlassen hat und eine neue Aufgabe als Fürsprecher für Entwickler bei der EOS Network Foundation übernehmen wird.

Natürlich sind wir am Boden zerstört, dass Jims besondere Art von jovialem Code-Geekertum nicht mehr an der Spitze des Developer Relations Programms stehen wird, aber wir freuen uns auch, dass er zu neuen Horizonten aufbricht und sein aufkeimendes technisches Gehirn mit den Freuden von Dingen wie Blockchain beschäftigen kann.

Mit dieser Nachricht geht natürlich auch die Ankündigung einher, dass Ian Barker das Amt des Embarcadero Developer Advocate übernommen hat. Er wird sich um die meisten der Dinge kümmern, die Jim getan hat, die öffentlichen und die, die er so gekonnt hinter den Kulissen erledigt hat, wovon es viele gibt.

Eli Mapstead wird seine Rolle ebenfalls ausbauen und einige der Python-Projekte übernehmen, die Jim beaufsichtigt und vorangetrieben hat. Ja, Jim kann mit Fug und Recht behaupten: "Es brauchte zwei Leute, um mich zu ersetzen".

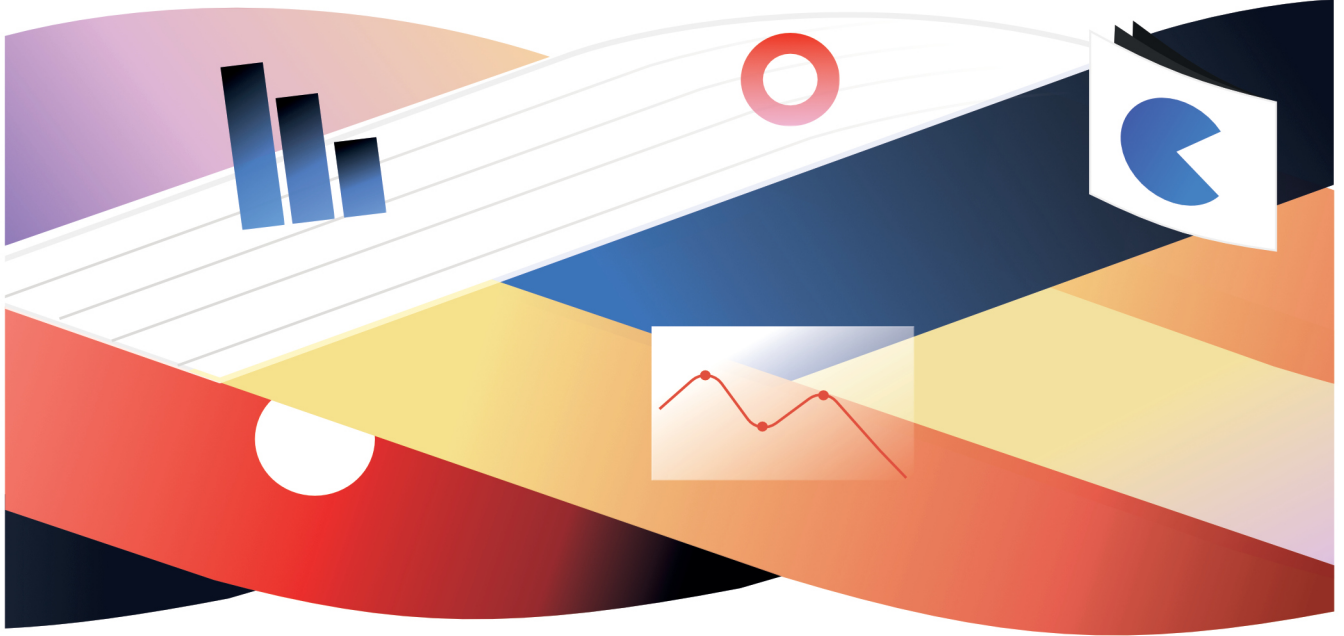
Ian Barker wird einen umfassenderen Blogbeitrag mit einem Rückblick auf einige der vielen großartigen Dinge schreiben, für die Jim verantwortlich war, und auch einige seiner verrückten Momente als Professor hervorheben.

Jim wurde von vielen Menschen wegen seines Humors und seiner Energie geliebt.

Neuestes von Jim:

Er hat sich selbstständig gemacht: Durch eine Meinungsverschiedenheit hat er seine neue Arbeitsumgebung sofort verlassen.





Update for FastReport

Creating a single ecosystem of report generators for Delphi in the FastReport 2023.2 release:

One installation system with online authorization — install and update all your products at once.

Common release system — major releases are published as a general.

Shared library for all products — fixes and new functionality can be available in multiple products at once.

Convenient to work with all products of the Delphi family already from the current release!

Also:

Improved compatibility with the current version of the IDE, HiDPI support for FastCube, and improved work with styles.

Upgrade to version 2023.2 or Download a free demo here:

www.fast-report.com



Installing FastReport in Lazarus for Linux

Es gibt eine neue FastReport Edition für Lazarus-Linux sowohl in einer Trial-Version als auch in einer Professional-Version.

Die Trial-Version hat ein Seitenlimit und einen Hinweis in der Ecke, dass es sich um eine Trial-Version handelt. In dieser Version fehlt die Rich View, aber die Professional Edition verfügt darüber und über Client/Server-Komponenten.

Dieser Artikel beschreibt, wie Sie FastReport in Lazarus für Linux installieren

Jeder Eintrag in dieser Liste besteht aus 4 Dateien (3 Installer-Pakete und eine Textdatei).

- Lazarus (Projekt) - Installationspaket;
- fpc-src Installationspaket;
- fpc(laz) Installationspaket;
- README.txt.

Es ist wichtig, dass Sie die Pakete in der richtigen Reihenfolge installieren.

Zuerst fpc(laz), dann fpc-src und schließlich Lazarus(project).

Lassen Sie uns das Problem mit den Schriftarten im Voraus lösen. Alle Betriebssysteme haben Standard-Schriftarten. Die Schriftart Arial ist zum Beispiel die Standardschriftart sowohl in Windows als auch in Ubuntu. Aber in Wirklichkeit ist die Standardschriftart Arial in Ubuntu nicht die Daher sehen Textberichte, die in Windows Lazarus erstellt wurden, in Linux Lazarus furchtbar aus (und umgekehrt). Um dies zu vermeiden, installieren wir die Schriftarten in Linux sofort wie in Windows.

Für Ubuntu können Sie den folgenden Befehl verwenden:

```
sudo apt-get install msttcorefonts
```

Bei anderen Linux-Distributionen kann der Befehl jedoch anders lauten.

Damit SQLite ordnungsgemäß funktioniert, müssen Sie außerdem die folgenden Pakete installieren: sqlite3, libsqlite3-dev.

Weitere Einzelheiten finden Sie unter:
<https://wiki.freepascal.org/SQLite>

Wir starten Lazarus und Sie werden aufgefordert, es zu konfigurieren.

Klicken Sie auf "OK", um die Standardeinstellungen zu übernehmen.

Lazarus
The professional Free Pascal RAD IDE

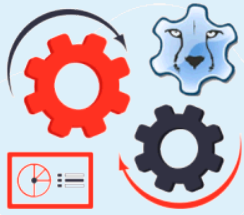
- Cross platform
- Drag & Drop Form Designer
- Open source (GPL/LGPL)
- Delphi converter

Download Now

Version 2.2.6 for Windows 64 bit | Other ▼

- Windows 32 Bits
- Windows 64 Bits
- Linux DEB 32 Bits
- Linux DEB 64 Bits
- Linux RPM 32 Bits
- Linux RPM 64 Bits
- Mac OS X 32 Bits
- macOS 64 Bits
- Other Downloads and mirrors



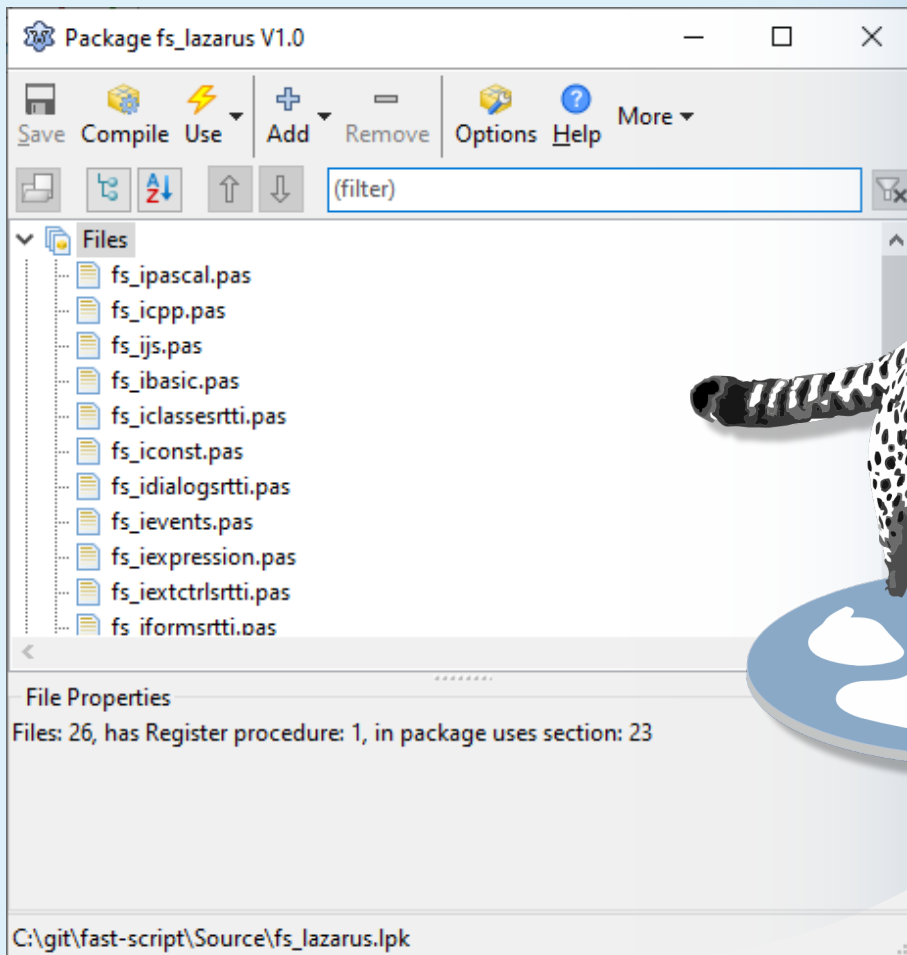


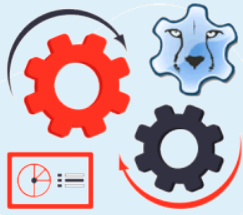
INSTALLIEREN VON FASTREPORT PACKAGES IN LAZARUS FÜR LINUX/WINDOWS

So, wir haben Lazarus bereits installiert, jetzt machen wir weiter mit der Installation der FastReport VCL Report Generator Pakete in Lazarus.

Dazu müssen wir zunächst die lizenzierte Version des Produkts von der offiziellen Website herunterladen und entpacken. Professional und höhere Versionen werden als .exe-Installationsprogramm geliefert, Trial und Academic - als Zip-Archive. Im Gegensatz zu Embarcadero Delphi, RAD Studio und C++ Builder, bei denen es ausreicht, die kompilierten Komponentenpakete einfach zu installieren, müssen sie in Lazarus kompiliert werden, mit Ausnahme von Trial und Academic, die mit geschlossenem (geschnittenem) Quellcode vorkompiliert sind.

Um Packages zu installieren, klicken Sie auf Package -> Open Package File *.lpk, wählen Sie das Package im Dateimanager aus und Sie werden das folgende Fenster sehen:





Für Professional und höher, klicken Sie auf Kompilieren, warten Sie, bis die Kompilierung abgeschlossen ist und klicken Sie auf Verwenden. Für Academic und Trial klicken Sie sofort auf Verwenden. Nach der Installation der einzelnen Pakete wird Lazarus neu gestartet.

Fahren wir mit der Reihenfolge der Installation der FastReport-Pakete fort:

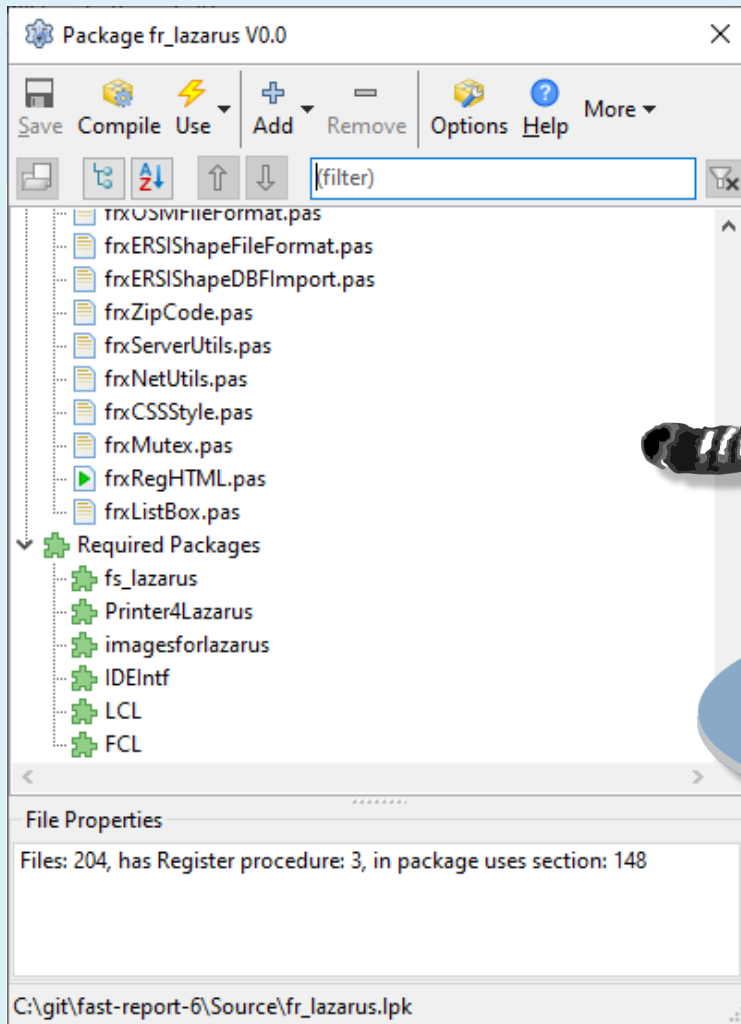
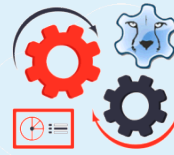
- ❶ `fast-script\Source\fs_lazarus.lpk` – library for executing scripts;
- ❷ `fast-report\Source\frN_lazarus.lpk` – package with all main components;
- ❸ In beliebiger Reihenfolge:
 - `fast-report\Source\ExportPack\frxeN_lazarus.lpk`
package with exports;
 - `fast-report\Source\frxchartlazarus.lpk`
package for charts (*diagrams*);
 - `fast-report\Source\lazdbf\frxlazdbf.lpk`
a package for working with a BDF format database;
 - `fast-report\Source\sqlite\frxlazsqlite.lpk`
a package for working with SQLite DBMS;
 - `fast-report\Source\PDFView\frxPDFlazarus.lpk`
a package for displaying PDF documents (Windows only);
 - `fast-report\Source\lazrich\frxrichlazarus.lpk`
a package for displaying Rich documents
(recommended only for Windows due to basic package restrictions);
- ❹ `fast-report\Source\ClientServer\frCS_lazarus.lpk`
a package with client-server components, you can read more about them here;

Wie im Editionsvergleich erwähnt, ist das frxRich-Paket nur für Professional und höher verfügbar, und die Client-Server-Komponenten sind nur in Enterprise und Ultimate verfügbar.

Vor Version 2.0.0 gab es einen sehr häufigen Kompilierungs- und/oder Installationsfehler. Zum Zeitpunkt der Erstellung dieses Dokuments ist die neueste Version 2.2.6 und dieser Fehler ist vielleicht noch nicht vollständig beseitigt worden. Lazarus noch nicht vollständig beseitigt, aber seine Wahrscheinlichkeit wurde deutlich unter Windows verringert.

Wenn sich eines der Packages nicht kompilieren/installieren lässt, müssen Sie die Package-Abhängigkeiten neu kompilieren/neu installieren die Package-Abhängigkeiten neu kompilieren/installieren.

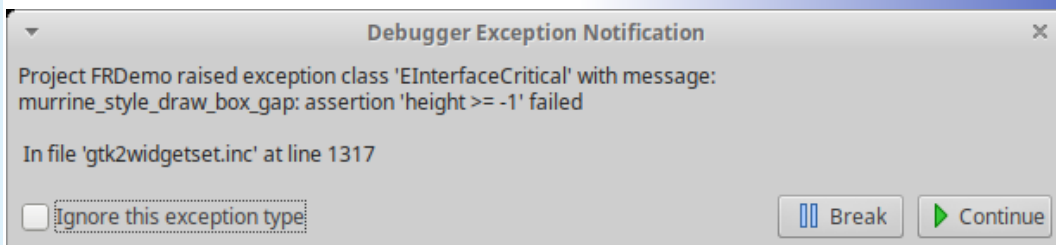


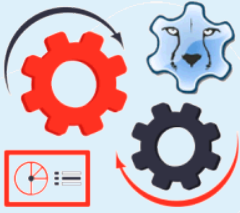


To do this, double-click on it and recompile, and then reinstall.

After successful installation of all the packages, click **Project -> Open Project** and open the `fast-report\LDemo\FRDemo.lpi` project and try to run it, then click the **Design** button.

If you get this negative height error on **Linux**:





Machen Sie sich keine Sorgen. Wir unterstützen sowohl GTK- als auch QT-Schnittstellen (aber bedenken Sie, dass die Entwicklung hauptsächlich mit GTK erfolgt). Sie können diesen Fehler also in einigen (eher seltenen) GTK-Schnittstellen finden. Führen Sie die Anwendung einfach ohne Debugging aus, oder aktivieren Sie die Option "Diese Art von Ausnahmen ignorieren".

*Oder wechseln Sie die grafische Shell. In unserem Team arbeiten zum Beispiel viele Mitarbeiter mit der GTK-Shell von KDE Plasma, wo dieser Fehler NICHT auftritt.

Die letzte Nuance, die Sie bei der Erstellung Ihrer Projekte beachten müssen, ist, dass unser Designer Multithreading verwendet, das in Linux Lazarus standardmäßig deaktiviert ist.

Um es zu aktivieren, öffnen Sie die Datei mit der Erweiterung ".lpr" im Projektinspektor (Projekt -> Projektinspektor) und fügen Sie die Unit cthreads im ersten Absatz im Abschnitt uses hinzu.





Donate for Ukraine and get a free license at:

<https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/>

If you are from Ukrainian origin you can get a free Subscription for Blaise Pascal Magazine, we will also give you a free pdf version of the Lazarus Handbook. You need to send us your Ukrainian Name and Ukrainian email address (*that still works for you*), so that it proofs you are real Ukrainian. please send it to editor@blaisepascal.eu and you will receive your book and subscription

BLAISE PASCAL MAGAZINE

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



Blaise Pascal



 **COMPONENTS
DEVELOPERS 4**

Donate for Ukraine and get a free license at:
<https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/>

 **COMPONENTS
DEVELOPERS 4**





Hello!



**10% off on FastReport VCL
Professional, Enterprise,
Ultimate editions**

From July 11 to July 25, get [Lazarus](#) support and source code with 10% off in FastReport VCL Professional and Enterprise editions, or full cross-platform with FastReport VCL Ultimate.

Save from **\$39,9** when purchasing a Single license, **\$89,9** when purchasing a Team and **\$599,9** when purchasing a Site!

Take advantage of getting much more functionality for less cost. You can see the differences between editions [here](#)

The offer is valid only for the license purchase and does not apply to an upgrade or renewal.

BUY WITH DISCOUNT

If you don't want to receive emails from Fast Reports - [manage your subscription](#)





Donate for Ukraine and get a free license at:
<https://components4developers.blog/2022/02/26/donate-to-ukraine-humanitarian-aid/>

NEW RELEASE

kbmMW Professional and Enterprise Edition v. 5.22.00 kbmMemTable v. 7.98.00 Standard and Professional Edition

5.22.00 is a release with containing new stuff, refinements and bugfixes. **OpenSSL v3 support**, WebSocket support, further improvements to SmartBind, new high performance hashing algorithms, improved RemoteDesktop sample and much more.

This release requires the use of **kbmMemTable** v. 7.97.00 or newer.

- RAD Alexandria supported
- Win32, Win64, Linux64, Android, IOS 32, IOS 64 and OS X client and server support
- Native high performance 100% developer defined application server
- Full support for centralised and distributed load balancing and fail-over
- Advanced ORM/OPF support including support of existing databases
- Advanced logging support
- Advanced configuration framework
- Advanced scheduling support for easy access to multi thread programming
- Advanced smart service and clients for very easy publication of functionality
- High quality random functions.
- High quality pronounceable password generators.
- High performance LZ4 and J peg compression
- Complete object notation framework including full support for YAML, BSON, Messagepack, J SON and XML
- Advanced object and value marshalling to and from YAML, BSON, Messagepack, JSON and XML
- High performance native TCP transport support
- High performance HTTPSys transport for Windows.
- CORS support in REST/HTML services.
- Native PHP, Java, OCX, ANSI C, C#, Apache Flex client support!

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

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

- New: full Web-socket support.
- The next release of kbmMW Enterprise Edition will include several new things and improvements.
- One of them is full Web-socket support.
- New I18N context sensitive internationalisation framework to make your applications multilingual.
- New ORM LINQ support for Delete and Update.
- Comments support in YAML.
- New StreamSec TLS v4 support (by StreamSec)
- Many other feature improvements and fixes.

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

NEW RELEASE

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

