Chromium Embedded Framework (CEF)

Fikret Hasovic, October 2018

Overview

The Chromium Embedded Framework (CEF) is an open source framework for embedding a web browser engine based on the Chromium core. It allows developers to add web browser control and implement an HTML5-based layout GUI in a desktop application or to provide web browser capabilities to a software application or game, and provides the infrastructure for developers to add HTML rendering and JavaScript to a C++ project. It also comes with bindings for C, C++, Delphi, Go, Java, .NET / Mono, Visual Basic 6.0, and Python and runs on Linux, Mac OS X and Windows.

There are two versions of Chromium Embedded Framework: CEF 1 and CEF 3. Development of CEF 2 was abandoned after the appearance of the Chromium Content API.

CEF 1 is a single-process implementation based on the Chromium WebKit API. It is no longer actively developed or supported.

CEF 3 is a multi-process implementation based on the Chromium Content API and has performance similar to Google Chrome. It uses asynchronous messaging to communicate between the main application process and one or more render processes (Blink + V8 JavaScript engine). It supports PPAPI plugins and extensions, both internal (PDF viewer) or externally loadable. The single-process run mode is not supported, but still present; currently it is being used for debugging purposes only.

I will therefore focus on the Lazarus version. In fact, there are two different versions of components, called fpCEF3 and CEF4Delphi.

CEF4Delphi is an open source project created by Salvador Díaz Fau to embed Chromium-based browsers in applications made with Delphi or Lazarus/FPC and is based on DCEF3, made by Henri Gourvest.

The current version of CEF4Delphi uses CEF 3.3497.1829.g004ef91 which includes Chromium 69.0.3497.81.

CEF4Delphi is being more developed than fpCEF3 lately, and is up-to-date with the chromium engine. Both projects are hosted on github, respectively: https://github.com/dliw/fpCEF3, https://github.com/salvadordf/CEF4Delphi.

Installation

Besides the Lazarus package, the CEF framework itself is needed. It is important to use the correct version of the CEF framework for the fpCEF release used. Further specific installation instructions can be found on Github: https://github.com/dliw/fpCEF3.

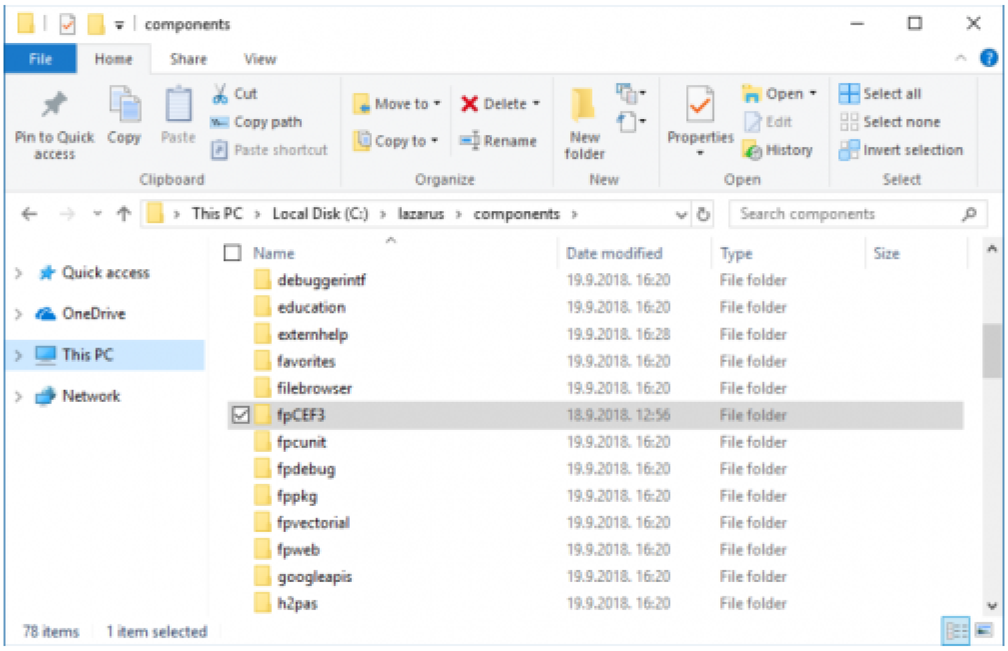
Warning: You can install fpCEF3 or CEF4Delphi, not both, since the component is named Chromium, the same in both packages. However, you can modify it and make a non-standard install.

Package Installation: FpCEF3

Download & Install fpCEF3

1. First of all, we need to download fpCEF3. You can download the latest version from https://github.com/dliw/fpCEF3.

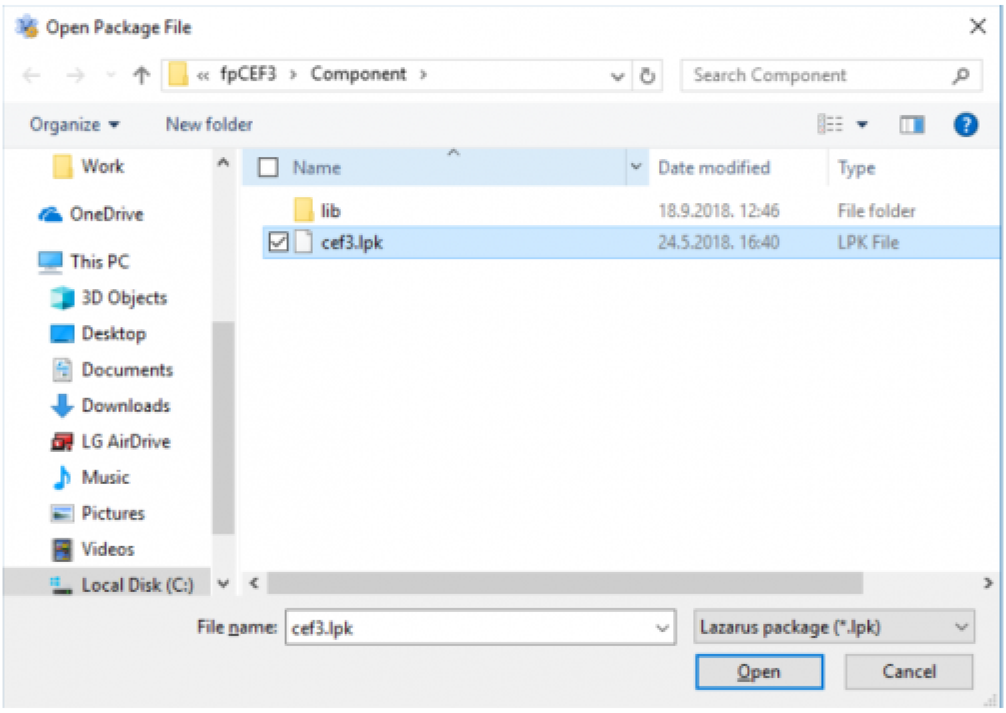
2. Extract & Copy the folder to the Components directory inside your Lazarus installation (this is usually C:\Lazarus\Components).



Name	Date modified	Type	Size
debuggerintf	19.9.2018. 16:20	File folder	
education	19.9.2018. 16:20	File folder	
externhelp	19.9.2018. 16:28	File folder	
favorites	19.9.2018. 16:20	File folder	
filebrowser	19.9.2018. 16:20	File folder	
fpCEF3	18.9.2018. 12:56	File folder	
fpcunit	19.9.2018. 16:20	File folder	
fpdebug	19.9.2018. 16:20	File folder	
fppkg	19.9.2018. 16:20	File folder	
fpvectorial	19.9.2018. 16:20	File folder	
fpweb	19.9.2018. 16:20	File folder	
googleapis	19.9.2018. 16:20	File folder	
h2pas	19.9.2018. 16:20	File folder	

After downloading the fpCEF3 package, it can be installed in Lazarus as follows:

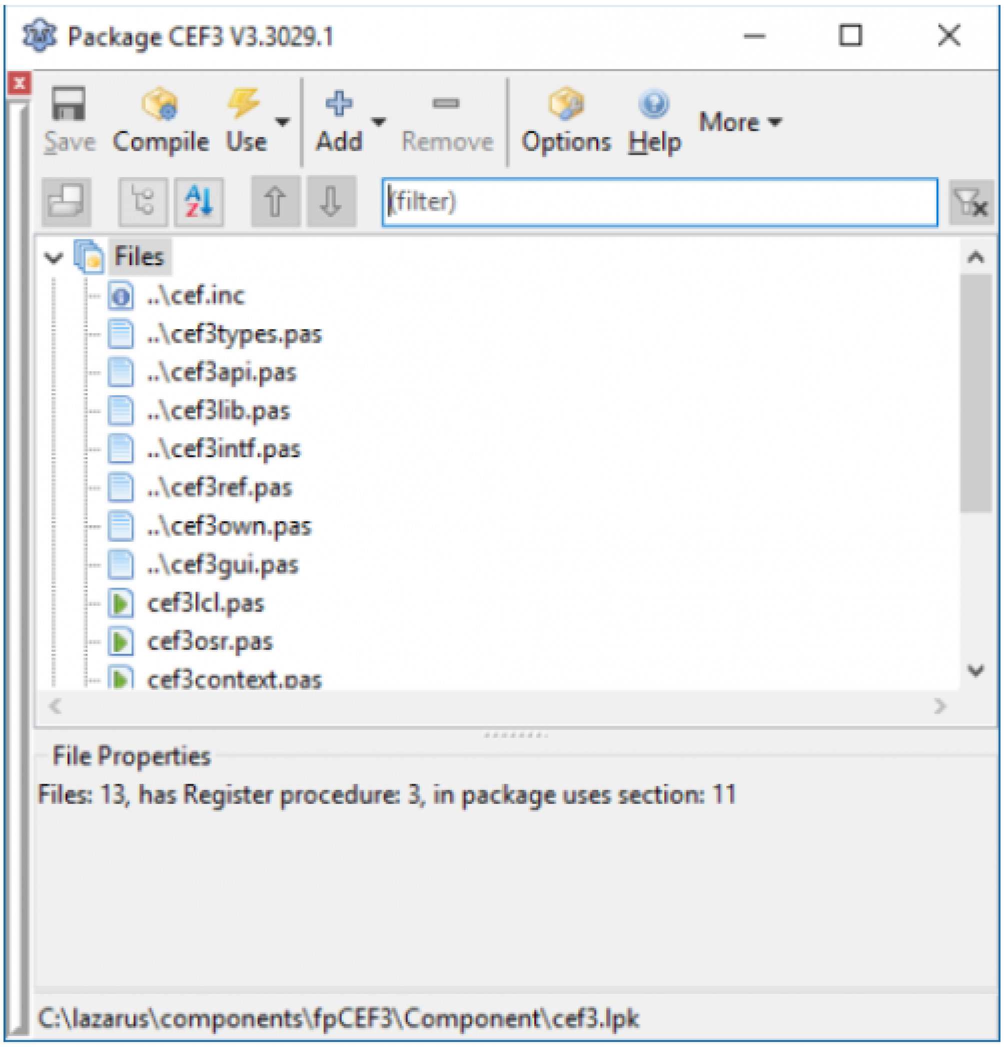
- Start Lazarus and open the main menu Package → Open package file (.lpk)…
- Select the file cef3.lpk in the fpCEF3 subdirectory Component



Name	Date modified	Type
lib	18.9.2018. 12:46	File folder
cef3.lpk	24.5.2018. 16:40	LPK File

File name: cef3.lpk
Lazarus package (*.lpk)

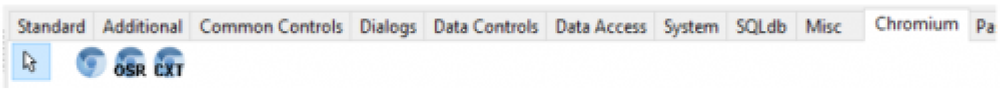
- The package editing window will open



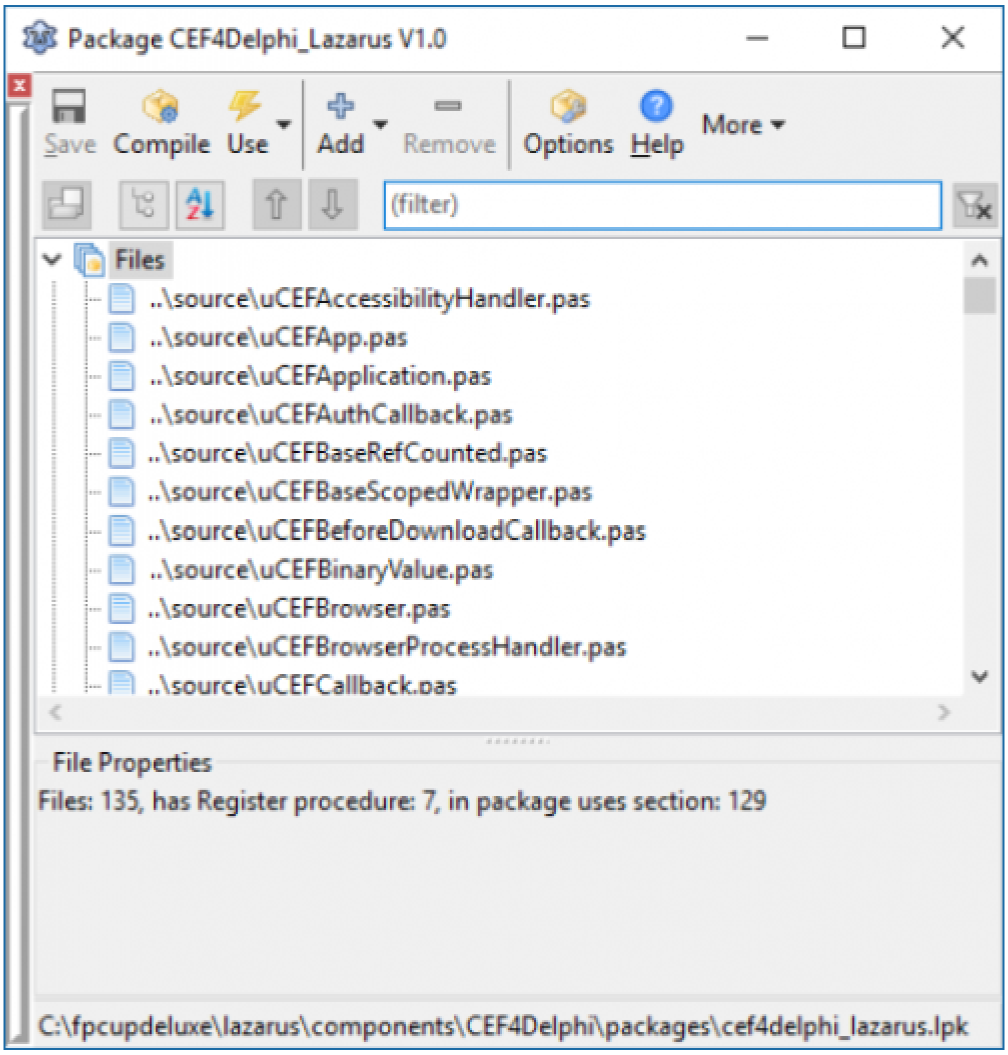
Files: 13, has Register procedure: 3, in package uses section: 11

C:\lazarus\components\fpCEF3\Component\cef3.lpk

- Click on Compile, the package will be compiled
- Click on Use…→ Install and confirm the dialog window with Yes
- Lazarus is now compiling and create a new tab Chromium in the component palette



CEF4Delphi



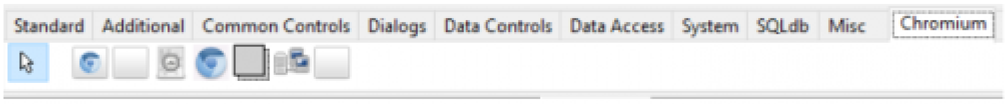
```
..\source\uCEFAccessibilityHandler.pas
..\source\uCEFApp.pas
..\source\uCEFApplication.pas
..\source\uCEFAuthCallback.pas
..\source\uCEFBaseRefCounted.pas
..\source\uCEFBaseScopedWrapper.pas
..\source\uCEFBeforeDownloadCallback.pas
..\source\uCEFBinaryValue.pas
..\source\uCEFBrowser.pas
..\source\uCEFBrowserProcessHandler.pas
..\source\uCEFCallback.pas
```

File Properties

Files: 135, has Register procedure: 7, in package uses section: 129

C:\fpcupdeluxe\lazarus\components\CEF4Delphi\packages\cef4delphi_lazarus.lpk

Follow the same steps as in fpCEF3 install, only you will install different package name, downloaded from https://github.com/salvadordf/CEF4Delphi (see image), and installed component pallete will look just a bit different:



Download Framework CEF3

Download Framework CEF3

You can find CEF3 precompiled binaries for different platforms in following location:

https://opensource.spotify.com/cefbuilds/index.html

Be sure to check the version number or commit messages to look for the appropriate version of CEF to be downloaded for it to work.

1. Download CEF. 2. Download & Install 7-zip. You'll need 7-zip to extract the file above. 3. Now extract the file and keep it that way. We'll only need the *Release*“ & ”*Resources* folders from this archive.

Usage

Now let’s create a web browser even quicker than you can install one! All using the people's favorite chromium engine!

FpCEF3 is a great project that will let us use the Chromium engine right in our Lazarus project. It is possible to use it in Windows, Linux and Mac OS platforms. But now, we are using Windows. (You can adopt this tutorial for the other platforms quite easily if you know your way around.)

So, let's make this browser …

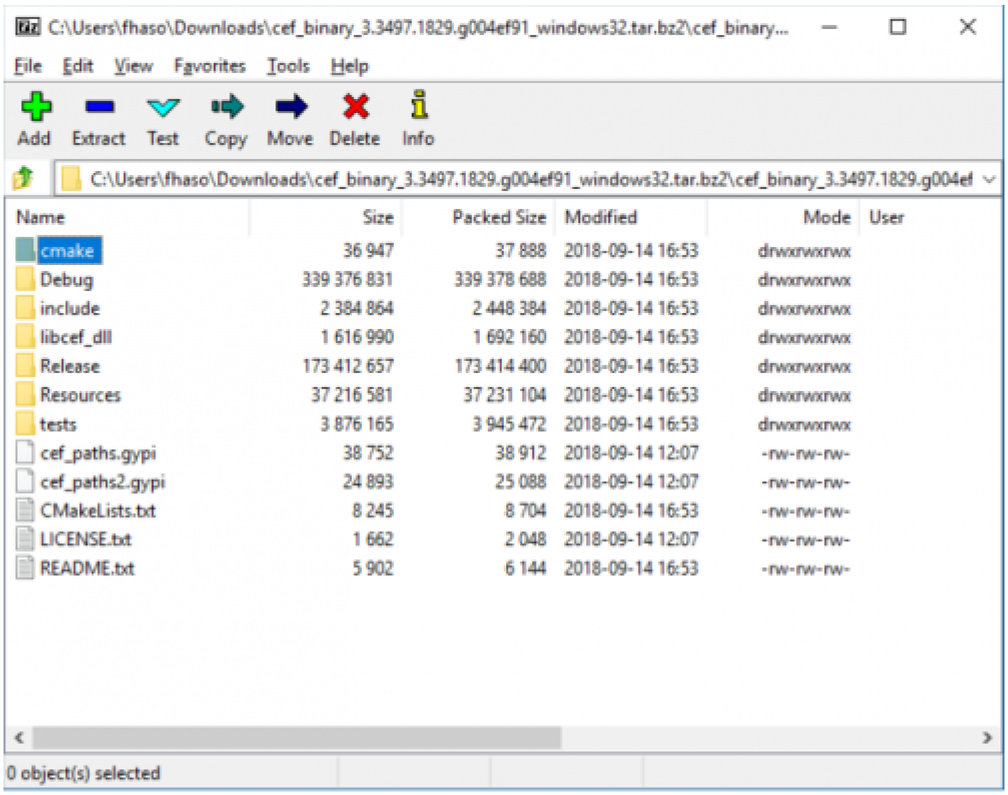
The previous steps were just preparations – which you'll only have to do once.

Start Lazarus.

Create a new Application project (Project→New Project→Application→OK).

Save the Project (File → Save All) in a directory.

And then copy all the files inside the Release and Resources folders from CEF package, and paste it directly into the project directory.



C:\Users\fhaso\Downloads\cef_binary_3.3497.1829.g004ef91_windows32.tar.bz2\cef_binary...

C:\Users\fhaso\Downloads\cef_binary_3.3497.1829.g004ef91_windows32.tar.bz2\cef_binary_3.3497.1829.g004ef

Name	Size	Packed Size	Modified	Mode	User
cmake	36 947	37 888	2018-09-14 16:53	drwxrwxrwx	
Debug	339 376 831	339 378 688	2018-09-14 16:53	drwxrwxrwx	
include	2 384 864	2 448 384	2018-09-14 16:53	drwxrwxrwx	
libcef_dll	1 616 990	1 692 160	2018-09-14 16:53	drwxrwxrwx	
Release	173 412 657	173 414 400	2018-09-14 16:53	drwxrwxrwx	
Resources	37 216 581	37 231 104	2018-09-14 16:53	drwxrwxrwx	
tests	3 876 165	3 945 472	2018-09-14 16:53	drwxrwxrwx	
cef_paths.gypi	38 752	38 912	2018-09-14 12:07	-rw-rw-rw-	
cef_paths2.gypi	24 893	25 088	2018-09-14 12:07	-rw-rw-rw-	
CMakeLists.txt	8 245	8 704	2018-09-14 16:53	-rw-rw-rw-	
LICENSE.txt	1 662	2 048	2018-09-14 12:07	-rw-rw-rw-	
README.txt	5 902	6 144	2018-09-14 16:53	-rw-rw-rw-	

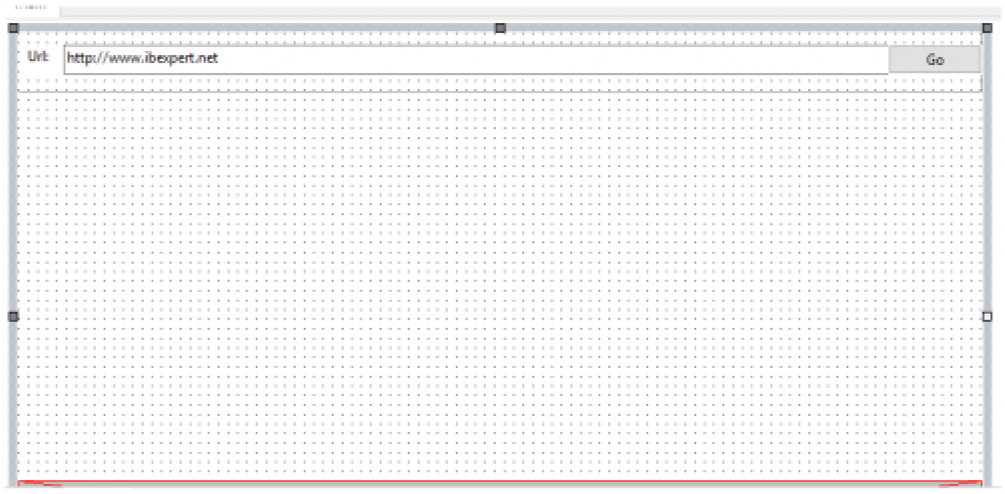
0 object(s) selected

Drop a new TPanel in to the form and remove it’s Caption. On TPanel drop TEdit and TButton. Set its properties like this:

```
TEdit.Name = edtURL
TPanel.Align = alTop
TButton.Caption = Go
```

The result is as follows:

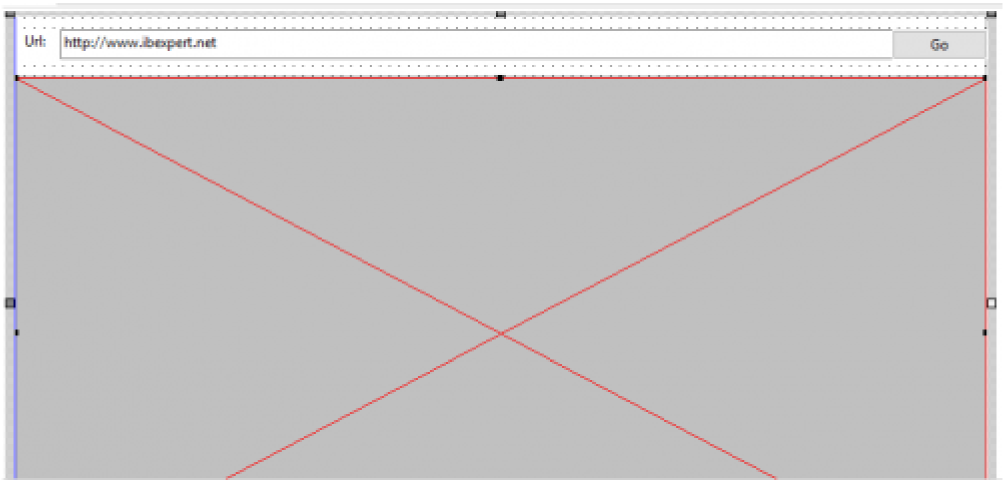
Now we'll have a nice Location bar/Address bar at the top.



Now Drop a new TChromium (from Chromium tab) in the form. Set its properties:

```
Name = Chromium
Align = alClient
```

It will fit the Chromium component nicely in the form's blank area.



Now double click the Form1 item in the Object Inspector and enter the code:

```


procedure TForm1.FormCreate(Sender: TObject);
begin
  edtURL.Text:='https://ibexpert.ne]';
end;

```

Now that you are in the Code view, add the necessary unit in the uses clause:

```
 uses
 ..., ..., ...
```

3 , cef3lib, cef3intf;

Now add the following code on edtURL's OnButtonClick event (select the edtURL component, go to Event tab, click the […] button beside OnButtonClick item):

```
procedure TForm1.edtURLButtonClick(Sender: TObject);
begin
   Chromium.Load(UTF8Decode(edtURL.Text));
end;
```

Now add the following code on Chromium's OnLoadEnd event (select the Chromium component, go to Event tab, click the […] button beside OnLoadEnd item):

```
 procedure TForm1.ChromiumLoadEnd(Sender: TObject; const Browser:
ICefBrowser;
    const Frame: ICefFrame; httpStatusCode: Integer);
begin
    edtURL.Text:=UTF8Encode(Browser.MainFrame.Url);
end;
```

All done! In about 2 minutes!

If you want to use CEF4Delphi, you need a little bit more time (I will leave it simple here):

In the project lpr file, add uCEFApplication to uses list. Also, add the following

```

  {$IFDEF MSWINDOWS}
       // CEF3 needs to set the LARGEADDRESSAWARE flag which allows 32-bit
          processes to use up to 3GB of RAM.
      {$SetPEFlags $20}

```

And make the body of main lpr file look like

```
  GlobalCEFApp := TCefApplication.Create;

  if GlobalCEFApp.StartMainProcess then
     begin
           RequireDerivedFormResource:=True;
           Application.Scaled:=True;
           Application.Initialize;
           Application.CreateForm(TForm1, Form1);
           Application.Run;
     end;

   GlobalCEFApp.Free;
   GlobalCEFApp := nil;
```

In the main unit, add Windows, uCEFChromium, uCEFWindowParent, uCEFInterfaces, uCEFConstants to uses list.

Also, add the following to the implementation section

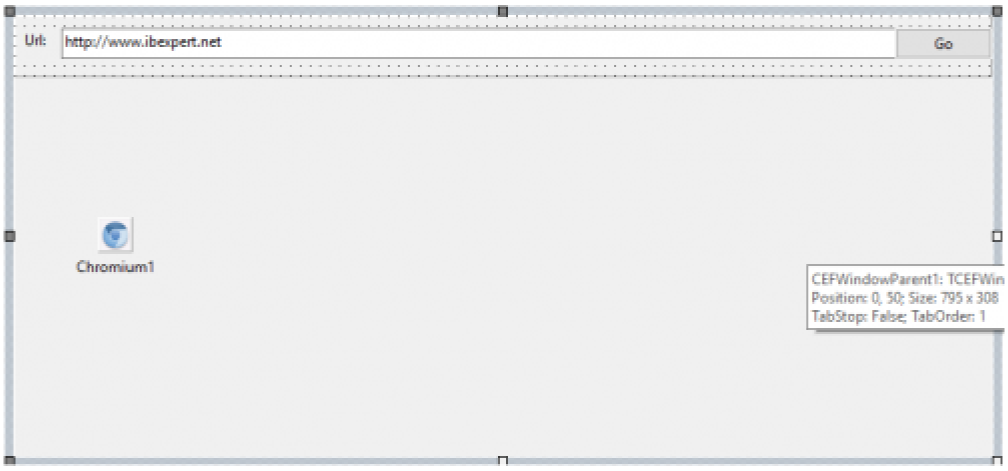
```
uses
   uCEFApplication;
```

In the protected section of your main form class, define:

```
 protected
     // You have to handle this two messages to call NotifyMoveOrResizeStarted or some page elements will be misaligned.
     procedure WMMove(var aMessage : TWMMove); message WM_MOVE;
     procedure WMMoving(var aMessage : TMessage); message WM_MOVING;
     // You also have to handle these two messages to set GlobalCEFApp.OsmodalLoop
     procedure WMEnterMenuLoop(var aMessage: TMessage); message WM_ENTERMENULOOP;
     procedure WMExitMenuLoop(var aMessage: TMessage); message WM_EXITMENULOOP;

     procedure BrowserCreatedMsg(var aMessage : TMessage); message CEF_AFTERCREATED;
     procedure BrowserDestroyMsg(var aMessage : TMessage); message CEF_DESTROY;
```

This time we need to use two components, a TChromium and a TCEFWindowParent


Url:
http://www.ibexpert.net
Go
Chromium1
CEFWindowParent1: TCEFWin
Position: 0, 50; Size: 795 x 308
TabStop: False; TabOrder: 1

And the rest of the code is just pasted here:

```
  procedure TForm1.BGoClick(Sender: TObject);
 begin
   Chromium1.LoadURL(UTF8Decode(EUrl.Text));
 end;

 procedure TForm1.Chromium1AfterCreated(Sender: TObject;
   const browser: ICefBrowser);
 begin
   // Now the browser is fully initialized we can send a message to the main
```

```
form to load the initial web page.
   PostMessage(Handle, CEF_AFTERCREATED, 0, 0);
 end;

 procedure TForm1.FormShow(Sender: TObject);
 begin
   // You *MUST* call CreateBrowser to create and initialize the browser.
   // This will trigger the AfterCreated event when the browser is fully
   // initialized and ready to receive commands.

   // GlobalCEFApp.GlobalContextInitialized has to be TRUE before creating
any browser
   // If it's not initialized yet, we use a simple timer to create the
browser later.
    //if not(Chromium1.CreateBrowser(CEFWindowParent1)) then Timer1.Enabled
:= True;
   Chromium1.CreateBrowser(CEFWindowParent1);
 end;

 procedure TForm1.WMMove(var aMessage: TWMMove);
 begin
   inherited;

   if (Chromium1 <> nil) then Chromium1.NotifyMoveOrResizeStarted;
 end;

 procedure TForm1.WMMoving(var aMessage: TMessage);
 begin
   inherited;

   if (Chromium1 <> nil) then Chromium1.NotifyMoveOrResizeStarted;
 end;

 procedure TForm1.WMEnterMenuLoop(var aMessage: TMessage);
 begin
   inherited;

   if (aMessage.wParam = 0) and (GlobalCEFApp <> nil) then
GlobalCEFApp.OsmodalLoop := True;
 end;

 procedure TForm1.WMExitMenuLoop(var aMessage: TMessage);
 begin
   inherited;

   if (aMessage.wParam = 0) and (GlobalCEFApp <> nil) then
GlobalCEFApp.OsmodalLoop := False;
 end;

 procedure TForm1.BrowserCreatedMsg(var aMessage: TMessage);
 begin
```

```
  BGo.Click;
end;

procedure TForm1.BrowserDestroyMsg(var aMessage: TMessage);
begin
  CEFWindowParent1.Free;
end;
```

Now Run the project (F9 or Run → Run). Click the "Go" button.