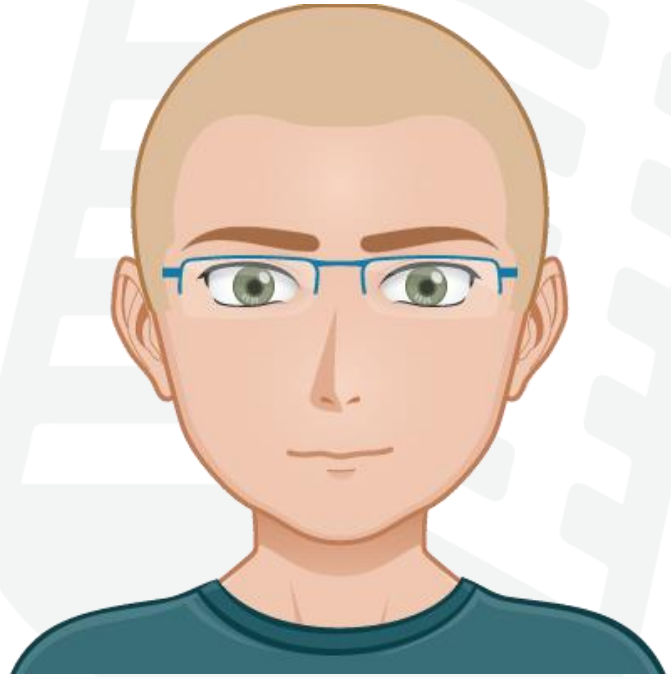
Does the Delphi IDE narrow you? Extend it!

Marco Breveglieri

Marco Breveglieri

ABLS TEAM *Software and Web* (Reggio Emilia)



featuring «Leo»

Homepage: **www.breveglieri.it**
Blog: **www.compilaquindiva.com**
Delphi Podcast: **www.delphipodcast.com**

What do we talk about?

What do we talk about?

Plugins!
Extensions!
Experts!
Add-ons!

Plugins, plugins, plugins!

Plugins have made the success of many apps.

- Browsers
  - Mozilla Firefox
  - Google Chrome
- CMS
  - Wordpress
  - Drupal
- Developer tools
  - Atom
  - Eclipse
  - Visual Studio Code

Sometimes, plugins are the piece of software that really matters.

What about Delphi?

Delphi has some really good add-ons or tools, freeware or opensource.

- General purpose and coding
  - *GExperts*
  - *ModelMaker Code Explorer (MMX)*
  - *CnPack Wizards*
  - *DDevExtensions*
  - *Documentation Insight*
  - *Castalia*
- Error handling
  - *Eureka Log*
  - *madExcept*
- Testing and quality
  - *Automated QA*
  - *Test Insight*
- Hot fixes
  - *Delphi SpeedUp*
  - *IDE Fix Pack*

…and many others… 👌

But Delphi often lacks specific, targeted and «feature focused» plugin. 🤔

What do I mean?

Open Command Line

Add New File

Trailing Whitespace Visualizer

Dummy Text Generator

Show Selection Length

XML, XPath & XSLT Tools

Comment Remover

Console Launcher

Solution Cleaner

PowerShell Prompt

Encourage

Farticus

What can I do?

The only limit is the sky!

Many scenarios deals with writing code, but this is not always the case.

You can create

- Productivity tools that add file templates and projects, refactoring and cleaning code, add new dialogs or tool windows.
- Support for new syntax and languages directly in the editor.
- Domain specific designers that allow you to «draw» a model and optionally produce the code (or not).
- Extend debugging features
- Create replacements, integrations and work arounds of limited or disliked IDE features

Benefits

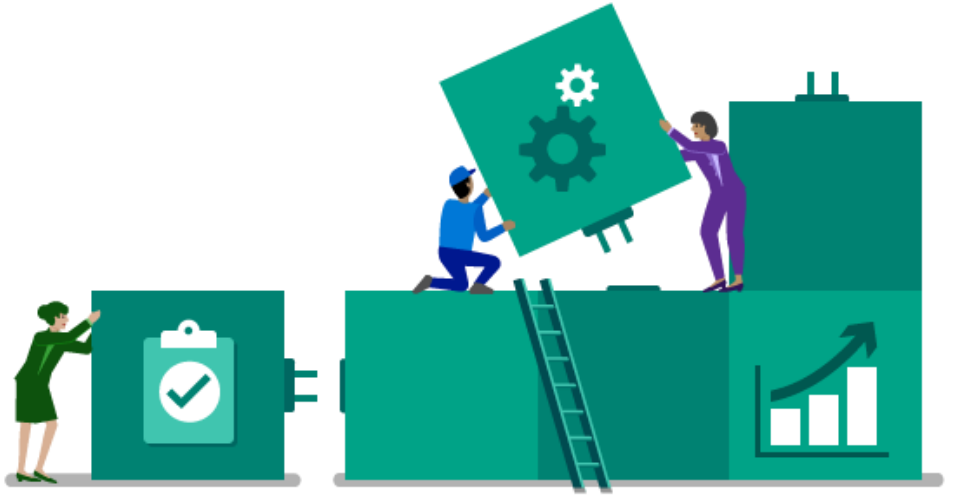
- Do things faster!
- Invest more time in automation than documentation
- More speed, less errors, high reliability, soft learning curve
- Share your tools with colleagues or an entire community to get feedback and improvements, and also new tools
- Decorate your framework with templates for projects and modules
- Fix Delphi bugs or implement workarounds waiting for fixes
- Last but not least… have fun!

So what?

Nobody is perfect.
Not even Delphi.

What can we do?

Let’s build some **Experts**!



Tools API

What is Tools API

- Suite of over 100 interfaces!

- Let's you interact with and control the IDE
  - *Main action lists, image lists and menus*
  - *Tool bars*
  - *Modules and source editors*
  - *Keyboard macros and bindings*
  - *Forms (and their components)*
  - *Debugger and Breakpoints*
  - *Code Completion*
  - *Message View*
  - *To-Do List*
  - *Build process*
  - *Project and Item Repository*

How to use ToolsAPI

- All the interfaces are inside a unique code file:
  `C:\Program Files (x86)\Embarcadero\Studio\20.0\source\ToolsAPI.pas`

- You can implement some interfaces to extend the IDE

- ToolsAPI provides some main kinds of extensions
  - Wizard

    a sort of setup class to easily build simple features (or host complex ones)
  - Notifier

    a class that contains logic that Delphi calls back when something interesting happens
  - Creator

    a class that supports the creation of new project, modules, units or virtually any kind of file

Sample Wizard

```
interface

uses
  ToolsAPI;

type

  TFirstWizard = class(TInterfacedObject, IOTAWizard)
  public
    procedure Execute;
    function GetIDString: string;
    function GetName: string;
    function GetState: TWizardState;
    procedure AfterSave;
    procedure BeforeSave;
    procedure Destroyed;
    procedure Modified;
  end;
```

Where to put the code

Package
(.dpk)

Library
(.dll)

Library (.dll)

- Quite «old school» solution
- They must be installed via registry
- Runtime packages are required!
- Registration, initialization and finalization code is more complex
  - You have to set the Application variable
  - What about the Memory Manager?
- Delphi loads them early (some menus to anchor to may not be available)

Where to put the code

- Natural container for all the elements in Delphi (types, records, classes, interfaces, routines, forms, frames, data modules, …)
- Use of runtime packages is embedded
- Register classes (and hence experts) is easy and straightforward
- They can be installed/uninstalled «hot» and «live» directly from the IDE

Demo

Warning!

- **There is little… no documentation** 😢
  - You should refer to the comments inside «ToolsAPI.pas»
- **Pay attention! Your code runs inside the IDE**
  - Delphi can crash or hang if your code misbehave or has errors
- **Testing is difficult, debugging even more!**
  - If you are unsure, install the tool using a separate profile
  - Launch the Delphi IDE (bds.exe) as the Host Application
- **An effective «Code DOM» and a working parser would be appreciated**
  - We hope that LSP support will resolve this issue and fill the gap

IDE Services

Native/Open Interfaces

NTA (Native Tools API)

- Grants direct access to actual IDE objects, such as the TMainMenu object of the IDE.
- The wizard must use RAD Studio packages (*rtl*, *vcl*, …).
- The wizard is tied to a specific version of the IDE.

OTA (Open Tools API)

- Interfaces do not grant full access to the IDE.
- All the functionality is available through OTA interfaces.
- Interacting and extending the IDE occurs with the OTA layer as a mediator.

Tools API Services

- Services give access to specific features through interfaces.
- Interfaces can be obtained from one single global variable: **`BorlandIDEServices`**.
- To get a service reference, you can call the ***Supports()*** method or cast **`BorlandIDEServices`** to the required type.
- Some interfaces have numbers (are versioned when changed).

Get a service /1

```
procedure SetKeystrokeDebugging(Debugging: Boolean);
var
  Service: IOTAKeyboardDiagnostics;
begin
  if Supports(BorlandIDEServices,
       IOTAKeyboardDiagnostics, Service) then
    Dialog.KeyTracing := Debugging;
end;
```

Get a service /2

```
var
  LServices: IOTAWizardServices;

begin
  LServices := BorlandIDEServices as IOTAWizardServices;
  WizardIndex := LServices.AddWizard(TFirstWizard.Create);
end;

begin
  with BorlandIDEServices as IOTAWizardServices do
    WizardIndex := AddWizard(TFirstWizard.Create);
end;
```

Demo(s)

Recap

Resources

- Delphi (RAD Studio) Documentation

  http://docwiki.embarcadero.com/RADStudio/Rio/en/Extending_the_IDE_Using_the_Tools_API

- Dave's (David Hoyle) Development Blog

  https://www.davidghoyle.co.uk/WordPress/?page_id=667

- Stack Overflow

  https://stackoverflow.com/search?q=delphi+ota

Q & A


Thanks.
gif-finder.com