EMBARCADERO
TECHNOLOGIES®

Tech Notes

A Tour of Delphi 2009

Marco Cantù

December 2008

Corporate Headquarters
100 California Street, 12th Floor
San Francisco, California 94111

EMEA Headquarters
York House
18 York Road
Maidenhead, Berkshire
SL6 1SF, United Kingdom

Asia-Pacific Headquarters
L7. 313 La Trobe Street
Melbourne VIC 3000
Australia

INTRODUCTION: THE DELPHI IDE

Since Delphi 8 for .NET and Delphi 2005 for Win32, Delphi has embraced a new IDE that is based on a slightly different metaphor, which favors embedded editors with designer panes docked to their side, rather than floating editors and floating designers. This "second edition" of the Delphi IDE is often indicated by its internal codename, Galileo.

Delphi 2009 features the 6th incarnation of the Galileo IDE. Not only is this the first version of the IDE that has every designer converted to Unicode, but there are also some very interesting new features, particularly in the area of project management.

INSTALLING AND RUNNING

Like Delphi 2007, the installation of Delphi 2009 is based on InstallAware. This time around, though, the installation experience has been considerably improved, particularly in speed. Delphi 2009 installation can be completed in 20 minutes rather than several hours.

A noticeable change in this respect is the fact that the help is now a separate installation, so it can be updated more frequently and separately from the main product (so you don't have to reinstall Delphi to get updated help, or to reinstall help should you want to reinstall the IDE). Installing help can take much more time than installing the actual product and the help install image is bigger than the IDE one.

When installing on Windows Vista, you'll have (by default) the product installed in the following folders:

```
C:\Program Files\CodeGear\RAD Studio\6.0
C:\Users\Public\Documents\RAD Studio\6.0\Demos\
C:\Program Files\Common Files\CodeGear Shared
```

.NET SDK NOT NEEDED

Previously, since Delphi 8 up to and including Delphi 2007, one of the prerequisites for installing the IDE was the presence of the Microsoft .NET SDK (version 1.1 earlier, version 2.0 later). It is not needed for Delphi 2009. You still have to install the considerably smaller Microsoft .NET runtime, which you might already have as part of the operating system, but don't need the development kit, which is much bigger and requires hundreds of MB.

In this version of Delphi, CodeGear is using Microsoft's Document Explorer, or *DExplorer*. This was previously available only in the SDK, but now can be deployed as a separate install.

Delphi help is very large (which is why it takes so much time to install), as it includes both CodeGear documentation and the Microsoft Platform documentation. In this release, however, the team fixed some "ranking" issues so that Delphi-specific topics should always be listed before the generic platform ones. Delphi-specific content was also much improved.

Windows Install Clean Up

At times, when uninstalling Delphi to replace it with an updated version, the installer complains, stops and won't work as expected. In these cases, CodeGear recommends cleaning all of the application folders (including some hidden ones that depend on the operating system). Alternatively, you can use Microsoft's own Windows Install Clean Up utility that can be found at:

```
http://support.microsoft.com/default.aspx?
  scid=kb;en-us;290301
```

Beware that using such a low-level tool can hamper your system, so proceed with caution (only after reading the instructions and at your own risk).

The -IDECAPTION Flag

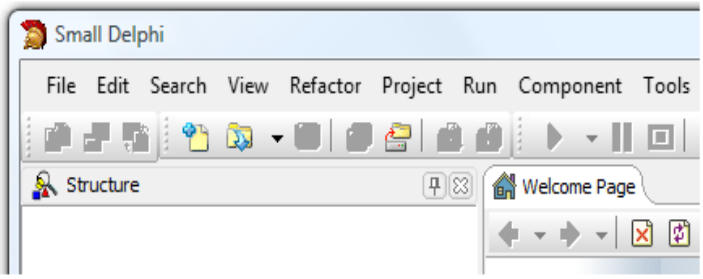
You probably know (although this was a well-kept secret for many years) that you can run multiple instances of the IDE, possibly at the same time, with different registry settings using the `-R` command line flag.

The problem with running run two different versions of the IDE at the same time is that it is hard to tell which is which. Another companion command line parameter for the IDE is `-idecaption`, that takes a caption as value. Summing the two flags you could run the IDE with the following link:

```
"C:\Program Files\CodeGear\RAD Studio\6.0\bin\bds.exe" -pDelphi -
rSmall -idecaption="Small Delphi"
```

This command runs the Delphi IDE with the Delphi Win32 personality only, activated the "Small" registry settings, and changes the IDE caption to "Small Delphi", as shown below:

If not specified from the command line, the IDE caption is retrieved from the Registry, in the Personalities section, in which there is a different string for each version (or active personality) of the IDE.



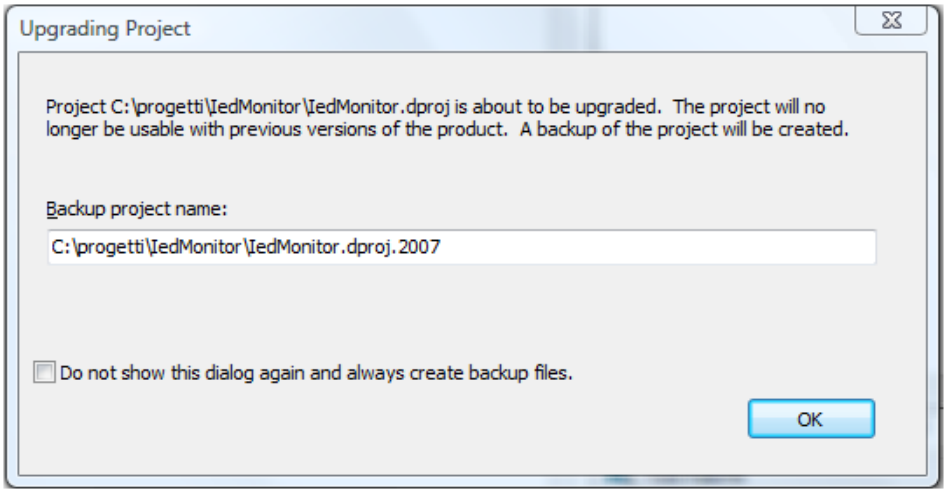
Managing Delphi Projects

Managing projects is a very common operation. If Delphi 2007 added some brand new concepts, like the MSBuild support, the target builds (Debug and Release) and the pre-build and post-build events, the new version makes these features more flexible and much easier to use, starting with a significant revamp of the Project Manager itself. Before we look at the Project Manager, though, we have to look to upgrading project files and the renewed Project Options dialog box.

UPGRADING PROJECT CONFIGURATION FILES

Since the early days of Delphi, the project source code file (with the .DPR extension) has contained Object Pascal code and uses one or more separate project configuration files for storing other settings. The format and extension of the project configuration file has changed a few times in recent versions, moving from an INI file to an XML file and then to an XML file for MSBuild (the .DPROJ file format).

From Delphi 2007 to Delphi 2009, the overall format of this project configuration file doesn't change. But its content is indeed different, and Delphi 2007 doesn't recognize further options added by the newer version of the IDE. When you open an existing Delphi 2007 project, the Delphi 2009 IDE will ask you for the name of a backup file into which it can copy the existing version of the project configuration file:



The default name for the project configuration backup file is the project name with the extension of `.dproj.2007`. In this specific case, I renamed the project file as `IedMonitor2007.dproj`. After you perform this operation, the IDE adds the following line to the message pane:

```
Upgrading project.  Backup
C:\progetti\IedMonitor\IedMonitor2007.dproj created.
```

Note that an updated Delphi 2009 version of the project configuration file is not created until you actually save it.

You can use the backup version to re-open the project in Delphi 2007. However, if you need backwards compatibility, a better idea might be to save the Delphi 2009 version of the project with a different name.

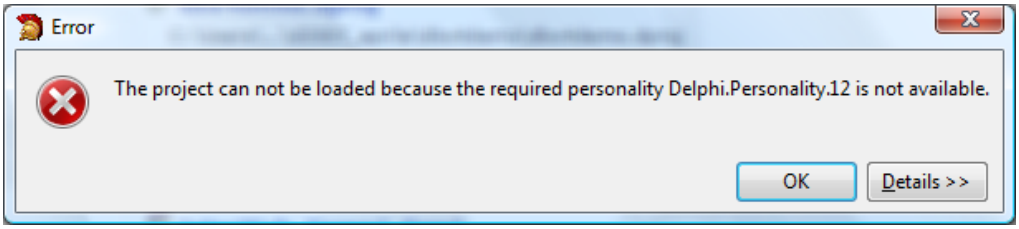
In the new .DPROJ file, Delphi 2009 adds a new project version tag:

```
<ProjectVersion>11.1</ProjectVersion>
```

The upgrade involves changes in the build configuration (as explained later), and in the resource management. The following sections are new or heavily modified:

```
<PropertyGroup Condition="'$(Config)'=='Release' or
          '$(Cfg_Release)'!=''">
     <Cfg_Release>true</Cfg_Release>
     <CfgParent>Base</CfgParent>
     <Base>true</Base>
</PropertyGroup>
<PropertyGroup Condition="'$(Config)'=='Debug' or
          '$(Cfg_Debug)'!=''">
     <Cfg_Debug>true</Cfg_Debug>
     <CfgParent>Base</CfgParent>
     <Base>true</Base>
</PropertyGroup>
<PropertyGroup Condition="'$(Base)'!=''">
     <DCC_DependencyCheckOutputName>SimpleApp.exe
     </DCC_DependencyCheckOutputName>
</PropertyGroup>
<ItemGroup>
     <DelphiCompile Include="SimpleApp.dpr">
          <MainSource>MainSource</MainSource>
     </DelphiCompile>
     <DCCReference Include="SimpleAppMainForm.pas">
          <Form>Form30</Form>
     </DCCReference>
     <BuildConfiguration Include="Base">
          <Key>Base</Key>
     </BuildConfiguration>
     <BuildConfiguration Include="Release">
          <Key>Cfg_Release</Key>
          <CfgParent>Base</CfgParent>
     </BuildConfiguration>
     <BuildConfiguration Include="Debug">
          <Key>Cfg_Debug</Key>
          <CfgParent>Base</CfgParent>
     </BuildConfiguration>
</ItemGroup>
```

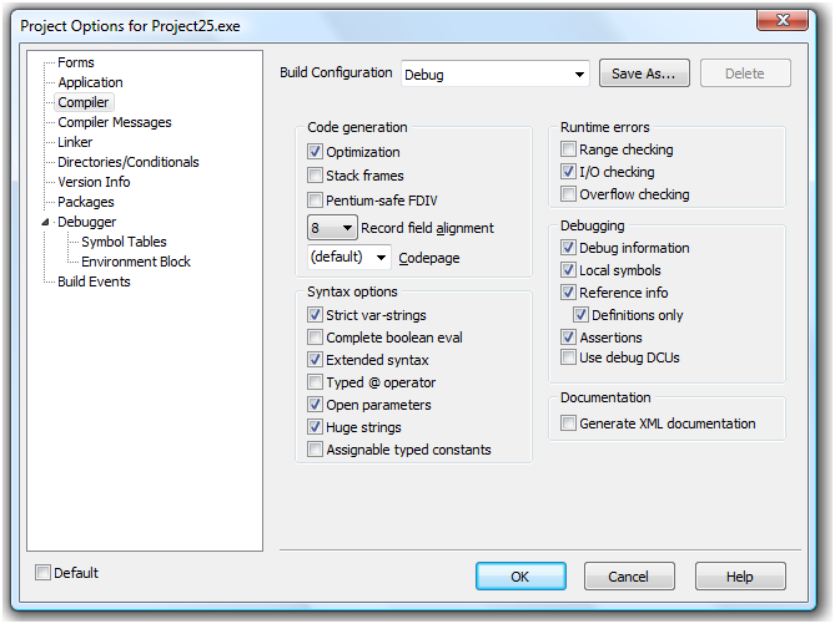
If you try re-opening this project file in Delphi 2007 (the only past version recognizing this format), you'll see the following error:

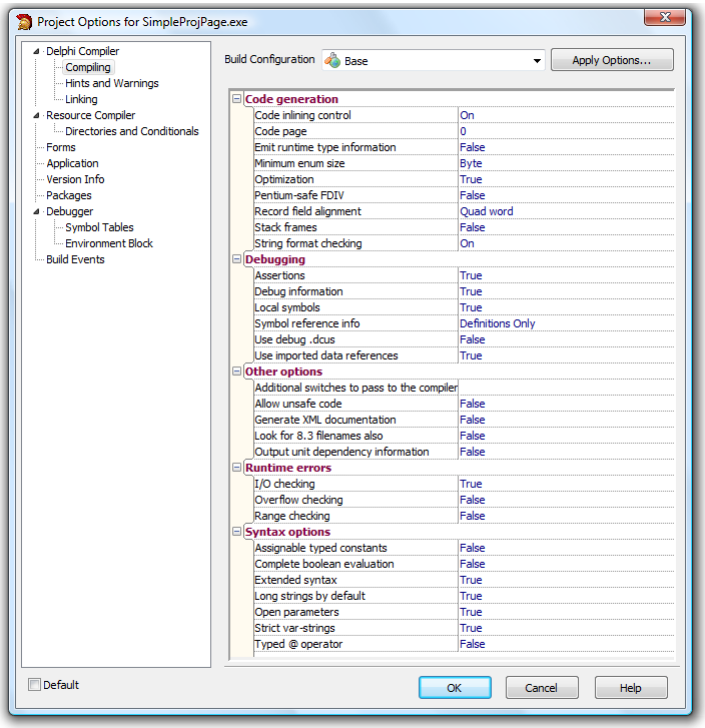


PROJECT OPTIONS DIALOG REDESIGNED

The Project Options dialog box is one of the Delphi dialogs I tend to use more often, and I guess I'm not alone. That's why its extensive redesign in Delphi 2009 at times leaves me puzzled. The redesign involves the pages with options that are part of the build configuration, and (as we'll see later in the section "Build Configurations and Configuration Settings") those pages of the dialog box are also

used inside the Project Configuration Manager. For example, look at the differences in the Delphi Compiler Options page between Delphi 2007 and Delphi 2009:





The difference is very significant. The check boxes are replaced by "True/False" and radio buttons by combo boxes with various options. There is also a help area at the bottom (minimized in the picture above, as I wanted to fit all options of the page in the dialog), that provides limited information about the various alternatives. One interesting element the "description" area provides the default value for the option.

The graphical redesign takes a while to get used to; but in addition, items within each group are now listed alphabetically, so they are in a different order than before. The directory options now under the main Delphi Compiler node. But beside organizational changes, is there anything missing or new?

NEW PROJECT OPTIONS FOR THE COMPILER

In the Delphi Compiler/Compiling page, which used to be called Compiler, the Code Generation section has the following new options:

- *Code inlining control* corresponds to the `$INLINE` compiler directive and controls how inlining works.
- *Emit runtime type information* corresponds to `-$M` flag on the command line or the `$M` directive, and determines the generation of runtime time information for a given class (or all of the classes of a project).
- *Minimum enum size* corresponds to `-$Z` flag (or the `$Z` directive) and determines the minimum size used for values of enumerated types (a Byte, a Word, a Double Word, or a Quad Word).
- *String format checking*, which is on by default, can be disabled to avoid some automatic string format checks (like the calls to `EnsureUnicodeString` function and other functions of the *Ensure String* family) and corresponds to the `$STRINGCHECKS` directive. This compiler option is new to Delphi 2009 and was supposed to remain undocumented and fairly hidden... so it is quite a surprise to see it prominently in Project Options dialog box.
- *Code page* was already in past versions but is now much more relevant in relationship with how the AnsiString type works (as explained in the white paper of this series covering Unicode in Delphi 2009).

The Debugging section has the new option *Use imported data references* (mapped to `$G`), which controls the creation of imported data references (increasing memory efficiency but preventing the access of global variables defined in other runtime packages).

The Runtime errors and Syntax options sections have the same elements (and the same defaults) as past versions of Delphi. The Other options section sports new options, except for the *Generate XML documentation* which was already available:

- *Additional switches to pass to the compiler* can be used to insert directly further command line compiler options not specifically supported by the IDE, although having this feature available now technically means that Delphi 2009 now supports each and every compiler option.
- *Allow unsafe code* will let you compile code deemed unsafe for a managed environment like .NET and makes little (or no) sense with the Win32 compiler.
- *Look for 8.3 filenames also* instructs the compiler to work on very old versions of Windows and corresponds to the -P compiler option.
- *Output unit dependency information* will turn on the `--depends` compiler flag, which is apparently not maintained for now.

OTHER NEW PROJECT OPTIONS

The Hints and Warnings page corresponds to the old Compiler Messages page. There are, of course, several new hints related with Unicode strings and other new compiler features.

The Linking page, which used to be called Linker, is visually quite different (and much more compact) but the only new option is *Set base address for relocatable images*.

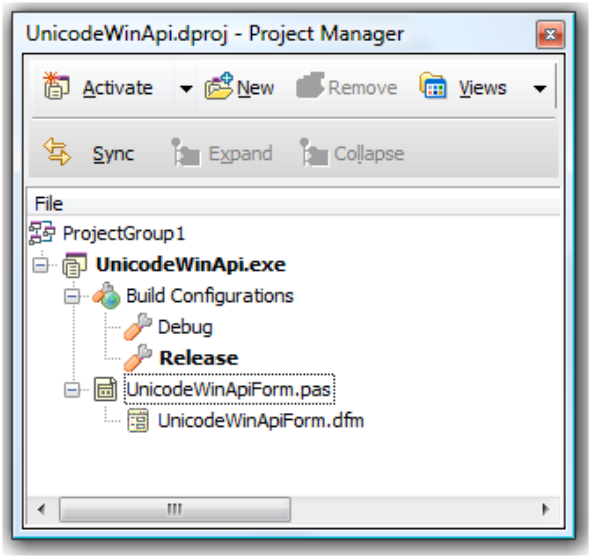
The main level Delphi compiler page has exactly the same options previously found under Directories/Conditionals. What can be quite confusing is that there is another page named Directories and Conditionals which is part of the resource compiler configuration under the Resource Compiler main level page. These pages are brand new and let you control the resource compiler from the Delphi IDE in ways never experienced in the past. There is a specific section later in this white paper, "Managing Resources in the IDE", covering this topic.

Default Projects Location

Since Delphi 2005, the default location for all new projects has been under the user documents folder. Few Delphi developers know this can be modified by setting a value for the Default project edit box in the Environment Options page of the Tools | Options dialog box.

The Project Manager

Along with a redesign of the Project Options dialog box, Delphi 2009 sees a significant update of the Project Manager pane, one of the most commonly used panes of the IDE. Even a cursory glance of its window will reveal some of its new features:



You can see there is a new Build Configurations node, with sub nodes, used to activate a build configuration in a much simpler way than in Delphi 2007. This is the topic covered in the later section "Build Configurations and Configuration Settings".

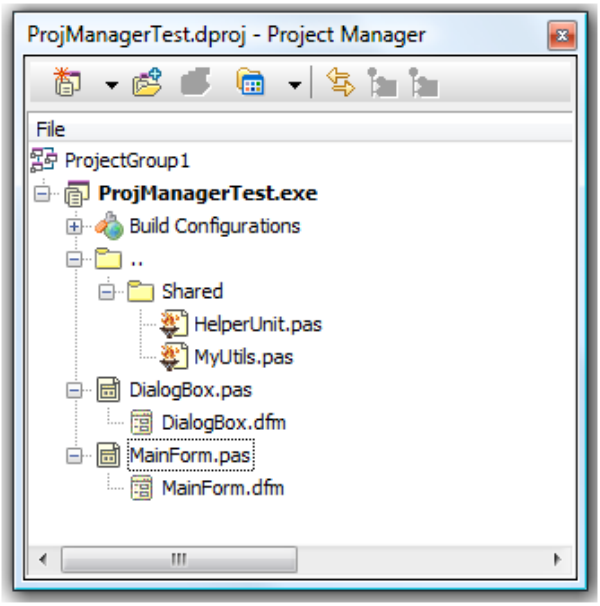
The Project Manager toolbar has several new buttons. The new Sync button selects the current file in the editor in the Project Manager, only if the file is part of the project, of course. The opposite operation (that is, activate the current selection of the Project Manager in the editor) can be done with a double-click.

The Expand and Collapse buttons will recursively expand and collapse all nodes under the current node. Apply Expand to a project group and you'll see a tree with all the configuration and file nodes of all projects in the group. Very handy, I have to say. The fourth new button, Views, is covered in the next section.

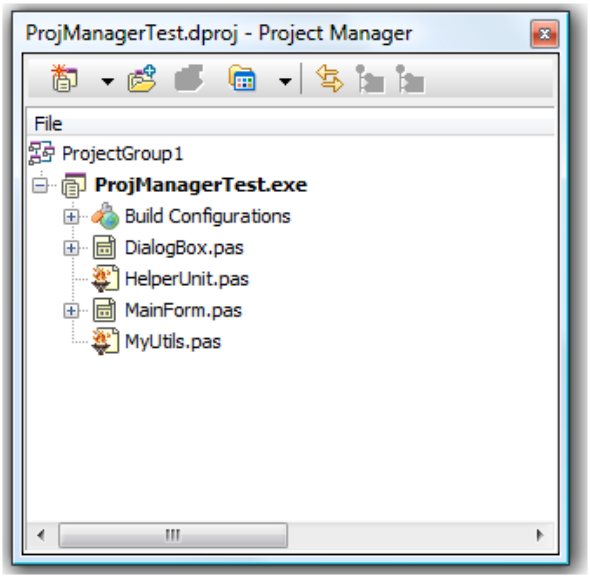
PROJECT MANAGER VIEWS

Another brand new feature is the Project Manager Views configuration. On the right side of the toolbar, you can see a new Views button that lets you change how the Project Manager shows files that have been placed in different folders. There are three options. I tested them by creating a sample program (called ProjManagerTest) with two forms in the main folder and two units in a secondary folder called Shared and placed at the same level in the file system hierarchy:

- Directory (Nested) is the default setting (and the only one available in Delphi 8 to Delphi 2007) that shows the files grouped by directory and the directories mimic the actual disk structure with separate nodes you can expand (so you might have to expand multiple nodes to move down a couple of sub-folders):



- Directory (Flat) is a new view in which the files are still divided by directory but each different directory is part of a list regardless of its position on the file system. In other words, you get a list of folders, each containing files, rather than (possibly) other nested folders:



- List is a new view corresponding to the traditional Delphi 7 list of files in the project manager. Directories are simply ignored and you can an alphabetic list of files:

Build Configurations and Configuration Settings

As I mentioned earlier (and you can see from the images on the previous pages), the Project Manager has a new Build Configurations node for every project (that is, in cases where you are working with a project group with multiple projects active). This node replaces the rather cumbersome separate window that used to manage the build configuration in Delphi 2007. Using the node and its sub-nodes you can change the current build configuration with a double-click, and execute an actual build directly on the given node.

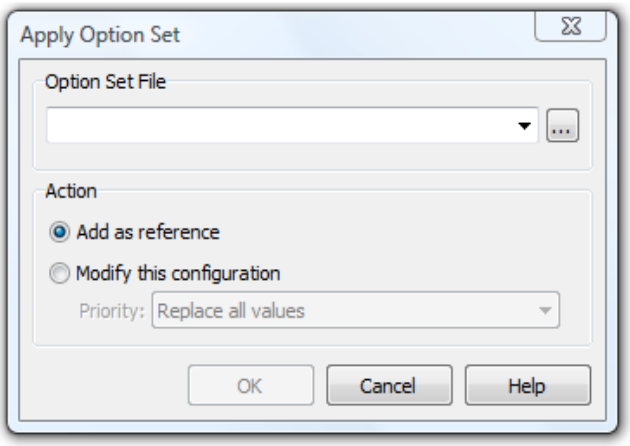
You can add a new configuration by selecting either a specific build configuration or the main node. Depending on the item selected when you do the operation, you'll create a main configuration or a sub-configuration. To be more precise, the node you pick determines the base configuration, since even the predefined configurations inherit their core setting from the Base configuration (which is the core configuration from which Debug and Release inherit).

What do I mean by "inherit settings" from a configuration? Delphi 2009 has a new configuration management system, in which you can apply a setting to a specific configuration (like Debug or Release) or set an option that the two configurations *inherit* from the Base configuration. In a specific configuration you can see the specific value and the one inherited from a base configuration in two consecutive lines, see if they match and change either one or the other (affecting also the specific configuration). This is archived by expanding each configuration setting line by selecting the plus

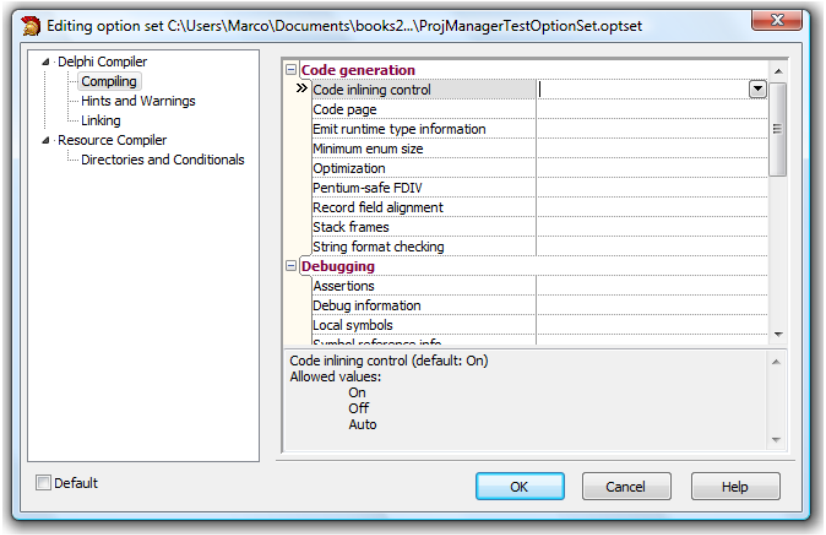
sign on the left. This is what you can get by expanding the three Runtime errors lines in the Delphi Compiler/Compiling page:
Modifying the setting in the Base configuration also affects any other configuration which inherits from that setting.

In the Project Manager, you can select a build configuration and export its settings to an "option set" file. This is like saving a configuration *template* or skeleton to an external file and the configuration will be linked to the file.

This makes it easy to move settings to a new or another existing project, as you can use the Project Manager (using the Apply Options Set local menu item while on a build configuration) or the Project Options dialog box (using the Apply Options button) to import a set of configuration options. In both cases Delphi opens up the Apply Option Set dialog box, where you can pick a file and choose whether to keep the external configuration file linked (so that a change in the file will be reflected in the projects using it) or simply merge the current settings using some *priority* rules:



Once you have created an external option set on a file, you can edit it from any project that refers to it, using the Edit local menu of the Project Manager pane. This opens up an editor containing a subset of the pages of the Project Options dialog box, as shown below:
The .OPTSET file is an XML file with a format similar to the .DPROJ format, again based on the



MSBUILD XML format, and an `OptionSet` project type. In this specific example the ProjManagerTestOptionsSet.optset file has the following content:

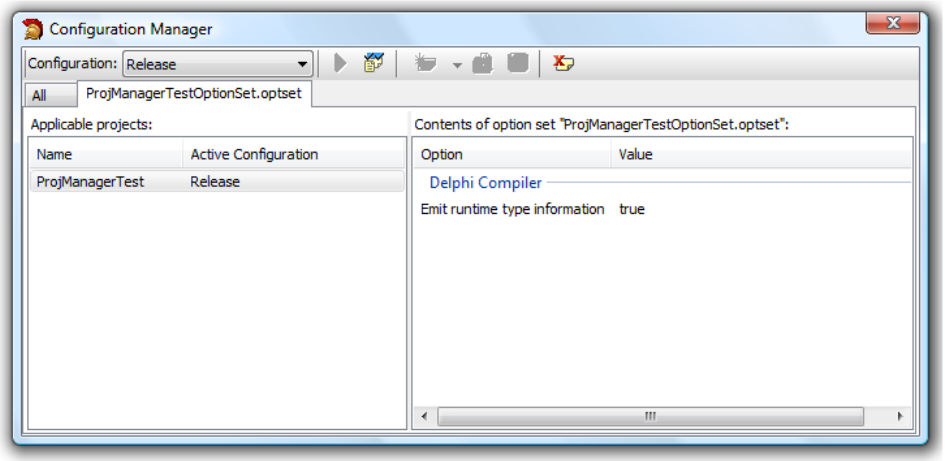
```
<Project xmlns="http://.../msbuild/2003">
      <PropertyGroup>
            <DCC_RunTimeTypeInfo>true</DCC_RunTimeTypeInfo>
      </PropertyGroup>
      <ProjectExtensions>
            <Borland.Personality>
                  Delphi.Personality
            </Borland.Personality>
            <Borland.ProjectType>
                  OptionSet
            </Borland.ProjectType>
            <BorlandProject>
                  <Delphi.Personality/>
            </BorlandProject>
      </ProjectExtensions>
</Project>
```

PROJECT CONFIGURATION MANAGER

Because the build options are available directly in the Project Manager pane, you don't have to use the Configuration Manager to change the current build configuration. Still, the Configuration Manager dialog box was quite handy also because it could let you change the build configuration for many projects in a project group at the same time. In fact, the Configuration Manager is still available in Delphi 2009, and in a much improved way: it lets you manage the various build configuration and option sets for all of the projects of a group at once.

To invoke the Configuration Manager you cannot use the local menu of the Project Manager, as in Delphi 2007, but have to select the corresponding item in the Project menu of the Delphi main menu.

When you do, you'll get this redesigned user interface:



The left side displays a list of projects and the active configuration for each. On the right, you see some details for the selected configuration, like the list of the non-default settings (the one in the

image is the summary of the option set file listed in the previous section). Using the tab you can also filter the projects on the left side according to the given configuration or option set active.

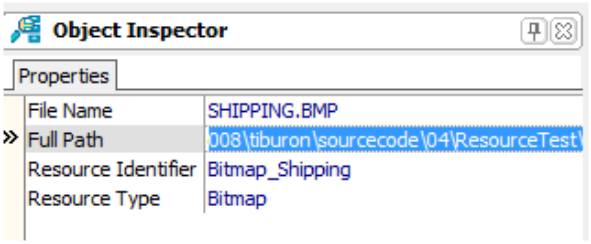
In Delphi 2009, the Configuration Manager lets you edit the project options for each build configuration, add new configurations, create or edit option sets, modify the active configuration... and perform most of the related operations in a single location, even if it's not trivial to use.

When you are working on multiple projects within a project group, the Configuration Manager has a distinct advantage over browsing in the Project Manager to work on the build configurations. For single projects, the Project Manager now has all you need.

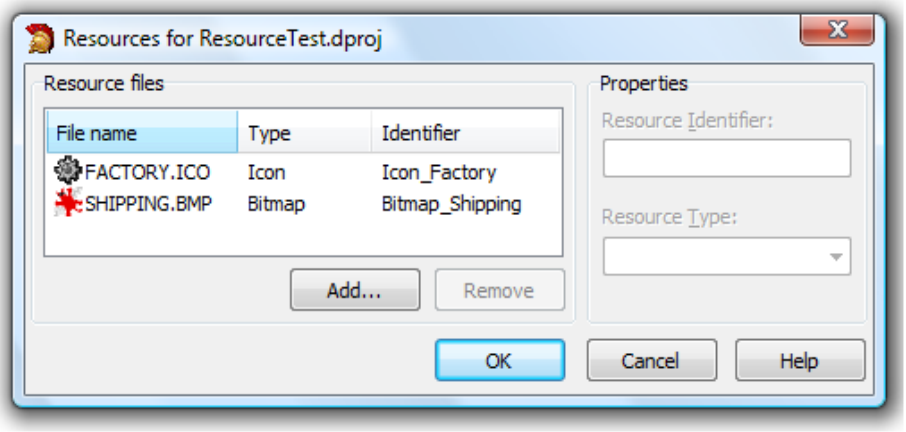
MANAGING RESOURCES IN THE IDE

In the most recent versions of Delphi, you could add resource scripts (.RC files) or compiled resource files (.RES files) to the Project Manager to let it compile them along with the project linking them to the executable. In Delphi 2009 managing resources has been simplified by the inclusion of a few more tools.

First, you can now drag individual resource files to the Project Manager to get them included as resources in a project. You can drag icons, bitmaps, and more. Delphi will generate a resource script files for these extra project resources, and compile it directly along with your program, embedding these resources in the executable. You can change any attribute of these resource files (including their internal name) in the Object Inspector:



Secondly, under the Project pull-down of the main menu of the IDE there is a new Resources menu item. Selecting this item brings up the Resources dialog box, which you can use to revise all of the resources of the program, add new resource files, rename them, change the format, and so on:



By adding a few resources to a project, Delphi will generate a proper resource file for you at compile time. For a program called ResourceTest (with the resources depicted above), Delphi 2009 generates a resource script file listing the project resources called `ResourceTestResource.rc`:

```
Icon_Factory Icon "FACTORY.ICO"
Bitmap_Shipping Bitmap "SHIPPING.BMP"
```

This resource script file is not added to the project (if you do so, you'll see duplicate resource warnings), but it is compiled along with it. In fact, if you make an error, like declaring your bitmap as an icon, the compiler will stop with the error:

```
[BRCC32 Error] ResourceTestResource.rc(2): resource file
SHIPPING.BMP is not in 3.00 format
```

and open the resource script file at the offending line. At compile time, Delphi 2009 generates (or updates) the resource script file, compiles it, and binds it to the executable. The intermediary file is a file with extension DRES that's included in the project with a directive automatically added to the project source code file (with the standard RES file including the project icon and string resources):

```
program ResourceTest;

{$R *.dres}

uses
  Forms,
  ResourceTest_MainForm in
    'ResourceTest_MainForm.pas' {FormResourceTest};

{$R *.res}

begin
  Application.Initialize;
  ...
```

MSBuild support has been present in the resource compilation steps in the since Delphi 2007. Here is the related output you'll see if you keep the -Verbose flag of the resource compiler options on:

```
c:\program files\codegear\rad studio\6.0\bin\cgrc.exe -c65001 -v
ResourceTestResource.rc -foResourceTest.dres

CodeGear Resource Compiler/Binder
Version 1.00 Copyright (c) 2008 Embarcadero Technologies Inc.

Microsoft (R) Windows (R) Resource Compiler Version 6.0.5724.0
Copyright (C) Microsoft Corporation.  All rights reserved.

Creating ResourceTest.dres
Using codepage 65001 as default

ResourceTestResource.rc.

Writing ICON:1,  lang:0x409,      size 744
Writing GROUP_ICON:ICON_FACTORY, lang:0x409,     size 20.
```

```
Writing BITMAP:BITMAP_SHIPPING,  lang:0x409,      size 44264
```

In case you've never used Windows resources directly, the ResourceTest program has a few lines of code to load the icon as the application and main form icon and to load the bitmap in an Image component:

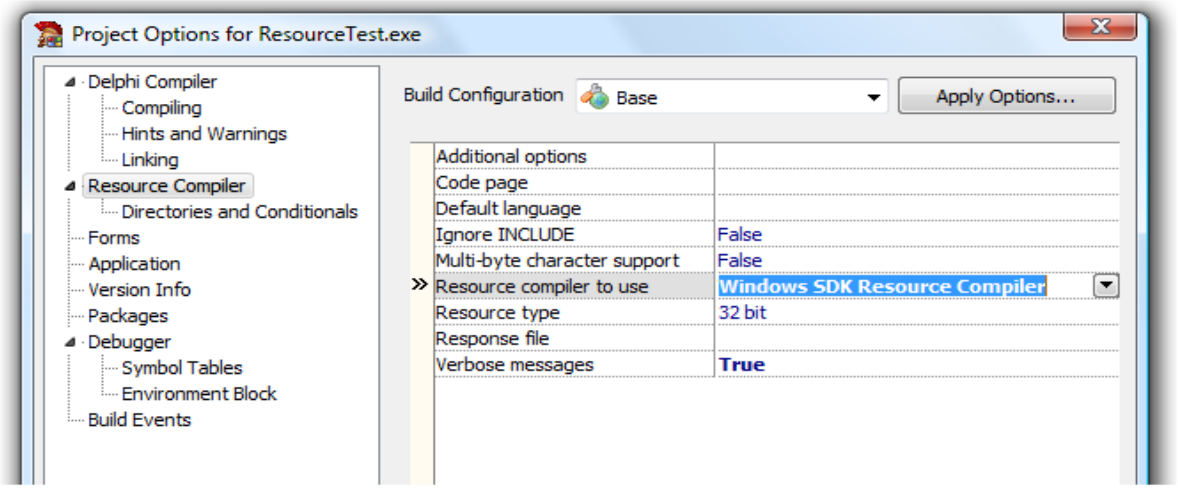
```
procedure TFormResourceTest.btnGifClick(
  Sender: TObject);
begin
  Image1.Picture.Bitmap.LoadFromResourceName(
    hInstance, 'Bitmap_Shipping');
end;

procedure TFormResourceTest.btnIconClick(
  Sender: TObject);
begin
  Icon.LoadFromResourceName(hInstance, 'Icon_Factory');
  Application.Icon.LoadFromResourceName(
    hInstance, 'Icon_Factory');
end;
```

A "New" Resource Compiler

Previous versions of Delphi, up to and including Delphi 2007, used the Borland Resource Compiler (BRCC32.EXE).

Delphi 2009 ships with a new resource compiler, or (to be more precise) a different resource compiler: the one from the Microsoft Windows SDK. This is certainly beneficial in terms of support for all of new resource formats Windows handles, but causes a few problems due to the fact the Borland resource compiler from the early days extended the Microsoft one, providing extra features that are now lost. You can set which resource compiler to use by setting the corresponding option in the resource compiler section of the Project Options dialog box (which lets you edit several other parameters of the resource compiler):



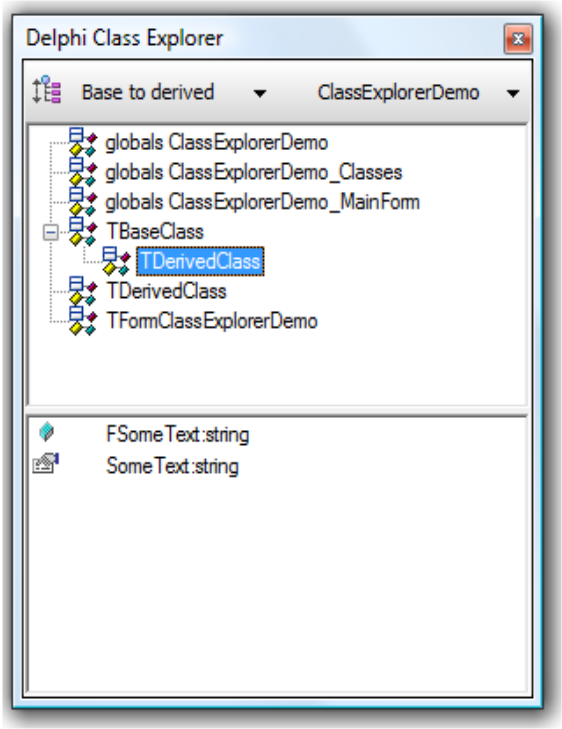
The Windows SDK Resource Compiler is invoked through the new CodeGear Resource Compiler/Binder, which is simply a front end to the SDK compiler. Changes in the resource compiler include the added ability to handle image (binary) data inline, to support trailing commas after strings in a string list, the different way to handle strings (now treated as C-language strings, forcing you to escape any \ in a file name with a double backslash), the different way to manage the folders for includes, and the like...

Again, if you've never used resource files directly, you can probably ignore any of these changes. Anything managed directly by the Delphi environment, from the embedding of DFM files as resources to the use of the `resourcestring` declaration, is fully backwards-compatible. If you did use resources directly, you should revise your resource files with some care.

The Delphi Class Explorer

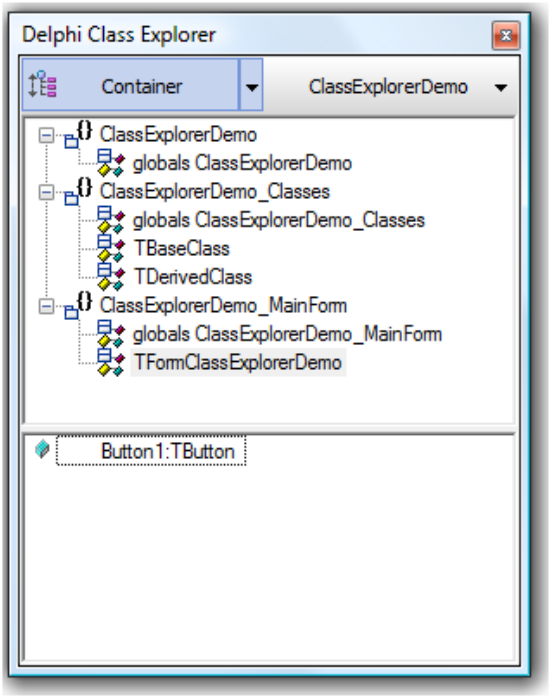
A brand new pane in Delphi 2009 is the Delphi Class Explorer pane (available from the Delphi Class Explorer item from the View menu). The Delphi Class Explorer offers a project wide representation of the symbols, which is different from the Structure View which shows a (somewhat similar) graphical representation of the elements of a single unit.

At the first level of the Delphi Class Explorer, you'll see a list of nodes hosting the global definition of each unit (plus the project file), while the remaining nodes show all of the classes defined in the project:

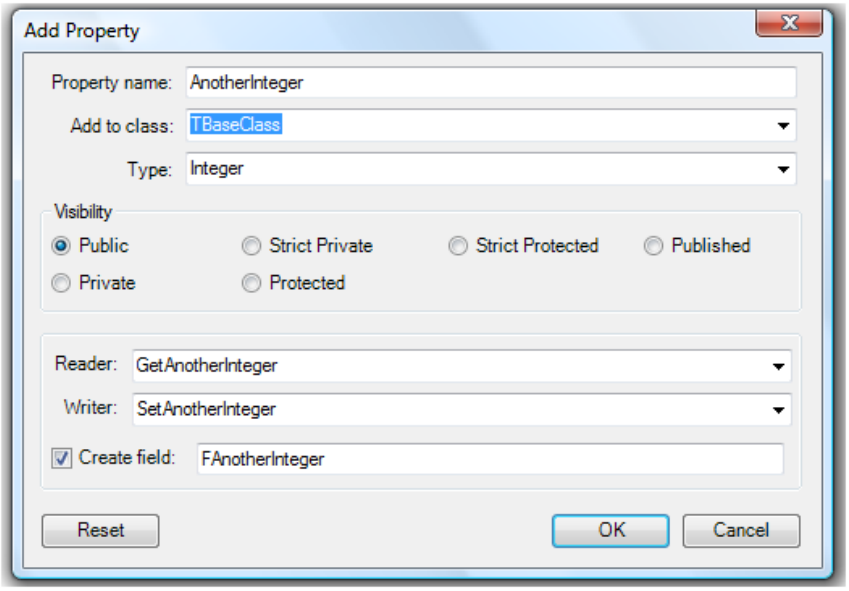


For each class you can see the specific members (not the inherited ones) and the relationship with other classes. This is depicted according to the selection in the first toolbar button: Base to derived (shown above), Derived to base, or Container. In this last case, classes (and globals) are divided by unit and no inheritance relationship is displayed:

The local menu lets you add a new field to a class, a new operation (or a method, including constructors and destructors), or a property, as in the image below:



Adding a property works in a proper Delphi way (much more than using UML-based modeling). The tools tend to map to setter and getter methods though you can go for a direct field mapping if you prefer, by adding a property and asking for a corresponding field to be created.



The Delphi Class Explorer will add the following lines to the class, as in the previous screen shot:

```
type
  TBaseClass = class
  strict private
    function GetAnotherInteger : Integer;
    procedure SetAnotherInteger(val : Integer);
  public
    property AnotherInteger : Integer
      read GetAnotherInteger write SetAnotherInteger;
  strict private
  var
    FAnotherInteger:Integer;
  end;
```

I find the use of a `strict private var` block quite odd, but it is formally correct and probably adding the extra `var` keyword makes code generation easier and less risky. I were to reformat the code to my liking, I'd do more than just declaring the property and I'd use Class Completion, which produces cleaner and more standard Delphi code. For me, the Delphi Class Explorer is an effective tool for navigating the source code of a project, and I'd rather use it than the Model View unless I needed to generate UML diagrams.

Other New Features

The updated Project Options dialog, the new features of the Project Manager and the extended build configurations, the improved support for resources, and the Class Explorer are probably the most significant new features of the IDE in Delphi 2009, if you don't consider the fact that the entire IDE has been Unicode enabled.

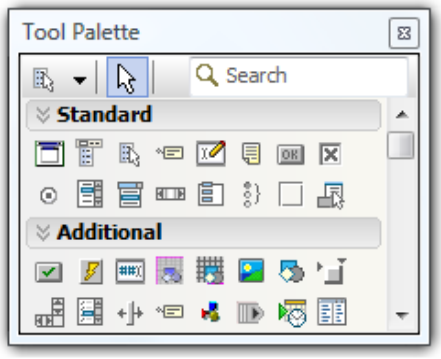
This section lists some of the many other minor features that can help you in the day-to-day work with the Delphi development environment. Other noticeable IDE improvements relate to:

- The Integrated Translation Manager (ITM), which has been revised for Unicode support and improved in various areas.
- The changes in the IDE related with the large changes in COM support and the Type Library editor.

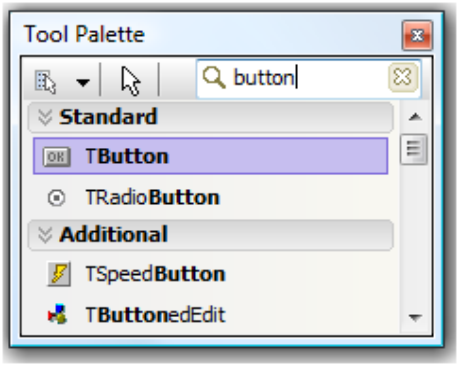
Tool Palette Search Box

In Delphi 2006, you could type while the Tool Palette was selected to filter components starting with those letters (with the exception of the initial T). In Delphi 2007 you could do the same, but also by selecting text inside the component name, so you could pick, say, IdHTTP by typing the more obvious HTTP.

In Delphi 2009, the Tool Palette has the same behavior as Delphi 2007, but with a different user interface that makes it more obvious to all users that you can search the components list:



As you select the palette (the handy shortcut is Ctrl+Alt+P), you can start typing in the search box (rather than in the caption of the pane) and the Tool Palette will filter the components being displayed:

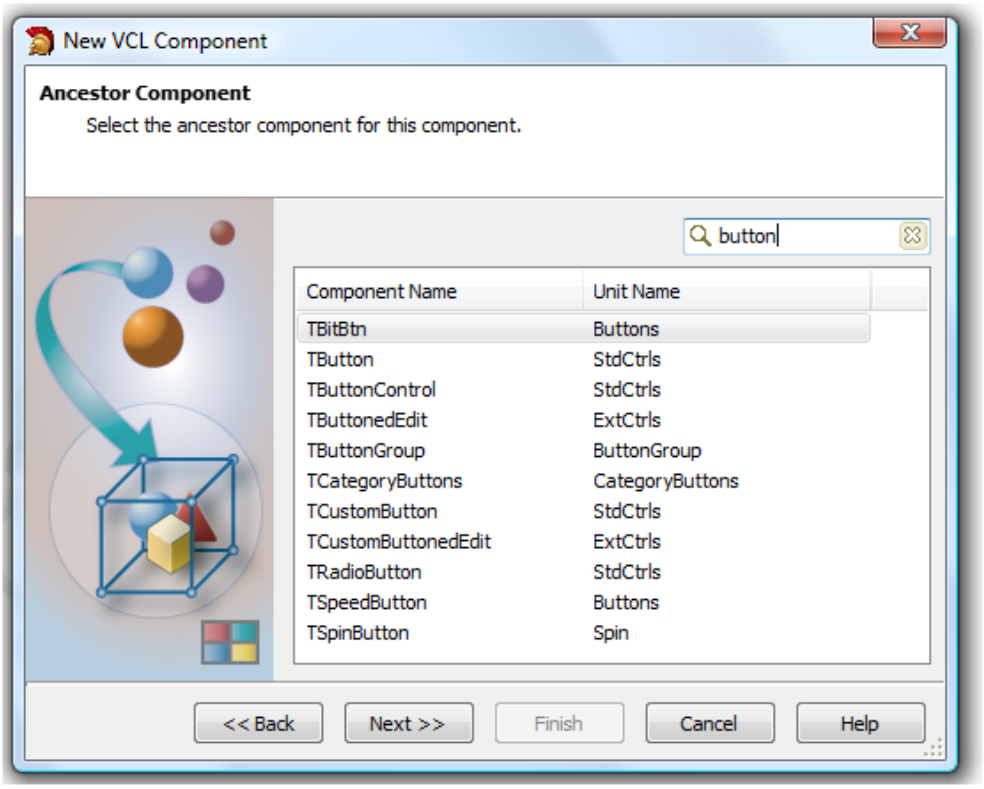


There is another change in the Tool Palette. As many people complained because of the excessive scrolling needed to reach the categories towards the bottom of the list, the auto-collapse of categories is now the default behavior. Another behavior you can fine-tune is whether the current selection of the Search box is kept after selecting a component or not.

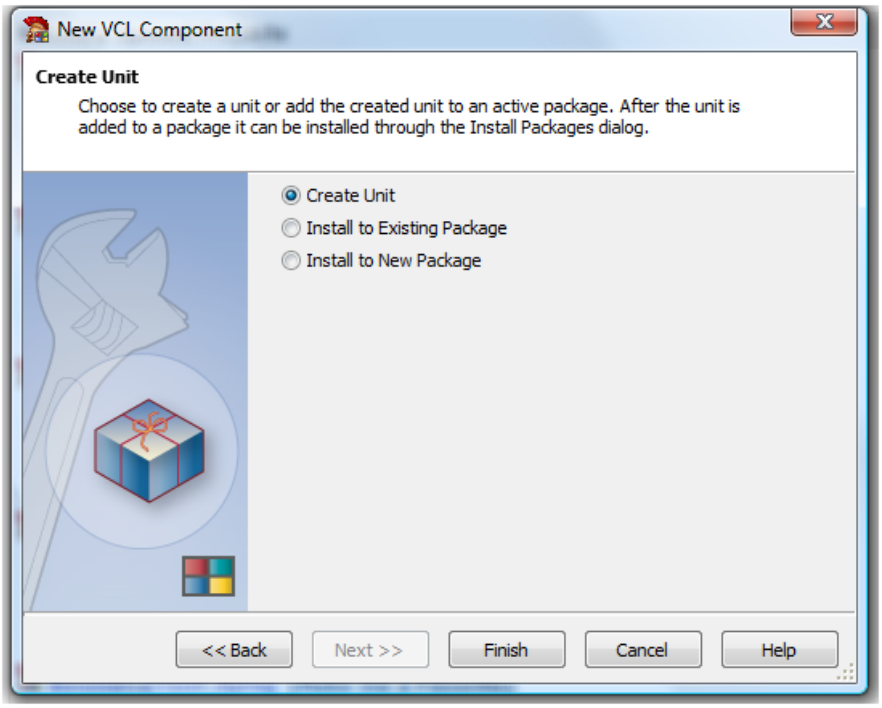
Updated Component Wizards

The dialog boxes used to create a new VCL component or import a component (an ActiveX control or a .NET assembly, to be used like a COM control) have been improved and turned into multi-step wizards. The actual capability to create an empty component skeleton or one wrapping an external control has not been modified significantly. The only new feature is the ability (from both wizards) to install the component into an existing package or into a new one which you have to name.

The relevant change is in the user interface of this wizard where you can activate from the Components menu of the IDE. For example, the initial page of the New VCL Component wizard has a search box used to filter the base class component to inherit from:



As you proceed, filling in the class name and other standard details, you'll get to the final page, which lets you create a new package or add the new component to an existing one:
If you have an active package project, you'll see an extra option to add the new component to it.

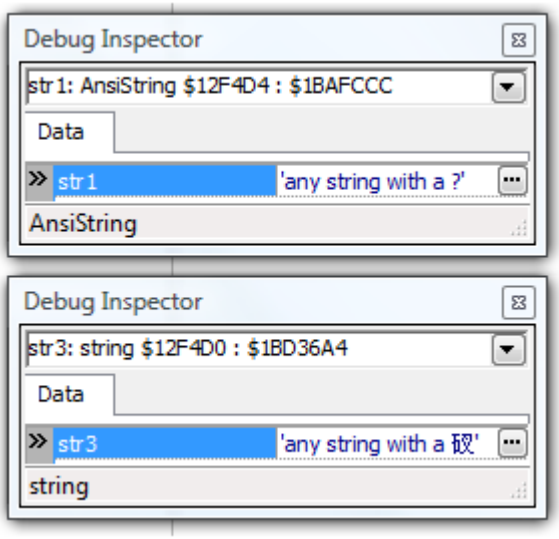


Similar capabilities have been added to the Import Component wizard.

DEBUGGER

Like the rest of the IDE, the debugger fully supports Unicode, too. This support was partially available in past versions, but Delphi 2009 extends it.

For example, if you inspect a string variable with Run | Inspect (or Debug | Inspect in the editor local menu) not only will you get the proper Unicode value but an indication at the bottom will inform you of the actual string type of the variable. Below you can see a comparison between the Inspect pane for an AnsiString and a UnicodeString (reported simply as *string*):



The two windows show the same string, but the first couldn't be converted properly due to the Chinese characters.

Other features of the debugger don't relate to Unicode support such as the CPU view supports the SSE3 and SSE4 instructions (minor for someone like me who infrequently uses assembly language).

A much more interesting, even if still quite low-level, feature is the support of the debugger for the Wait Chain Traversal feature of Vista (and Windows Server 2008). For more information, see an MSDN technical article that describes Wait Chain Traversal at the operating system level and a blog post by Chris Hesik of CodeGear about this new feature of the Delphi debugger at these URLs:

```
http://msdn.microsoft.com/en-us/library/ms681622.aspx
http://blogs.codegear.com/chrishesik/2008/07/21/34833
```

The Threads Status pane has an extra column with information about various threads that are contributing to a deadlock that can help you understand your multi-threaded applications.

DEBUGGING AND NEW LANGUAGE FEATURES

Even if the debugger looks quite similar to the previous version, a lot of effort was devoted to allow users to debug applications that use generics and anonymous methods. Because of the sophisticated code generation done in the background, the code you are debugging is quite different from that which you originally wrote. Even with some limited glitches, debugging the new language features generally works well enough, and that was far from an easy task to achieve.

Project Management Comes to Delphi

The Delphi IDE has been significantly extended over the last few editions, and Delphi 2009 continues in this trend. The most relevant feature overall is probably the increased stability of the IDE. Next come the extensions to the project management features, with support for hierarchical build configuration, options settings you can move from a project to another, integrated resource management, and multiple Project Manager views to adapt this pane to the user preferences. Improved configuration management makes it a lot easier to work with larger projects, even compared to the classic editions like Delphi 7. I'm pretty sure the impact of these new features on the everyday work of Delphi developers will be significant The Delphi 2009 IDE is well worth the upgrade.

About the Author

This paper has been written for Embarcadero Technologies by Marco Cantù, author of the best-selling series Mastering Delphi. The content has been extracted from his latest book "*Delphi 2009 Handbook*", http://www.marcocantu.com/dh2009. You can read about Marco on his blog (http://blog.marcocantu.com) and reach him at his email address marco.cantu@gmail.com.

Embarcadero Technologies, Inc. is a leading provider of award-winning tools for application developers and database professionals so they can design systems right, build them faster and run them better, regardless of their platform or programming language. Ninety of the Fortune 100 and an active community of more than three million users worldwide rely on Embarcadero products to increase productivity, reduce costs, simplify change management and compliance and accelerate innovation. The company's flagship tools include: Embarcadero® Change Manager™, CodeGear™ RAD Studio, DBArtisan®, Delphi®, ER/Studio®, JBuilder® and Rapid SQL®. Founded in 1993, Embarcadero is headquartered in San Francisco, with offices located around the world. Embarcadero is online at www.embarcadero.com.