



백서

Delphi 2009 투어

작성자: Marco Cantù

2008년 11월

Corporate Headquarters

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

Asia-Pacific Headquarters

L7. 313 La Trobe Street
Melbourne VIC 3000
Australia

DEVGEAR

서울특별시 서초구
반포동 743-14
데브기어 4층

소개: DELPHI IDE

Delphi에는 Delphi 8 for .NET과 Delphi 2005 for Win32 이후로 지금까지와는 약간 다른 방식의 새 IDE가 채택되었으며, 이 새 IDE에서는 플로팅 에디터와 플로팅 디자이너가 아닌, 디자이너 창의 옆으로 도킹된 임베디드 에디터를 지원합니다. 이 “두번째 버전”의 Delphi IDE는 종종 내부 코드명인 Galileo로 불려집니다.

Delphi 2009는 Galileo IDE의 6번째 버전입니다. IDE 내의 모든 디자이너를 Unicode로 변환한 최초의 버전일 뿐만 아니라, 흥미로운 몇가지 새로운 기능들, 특히 프로젝트 관리 관련으로 새로운 기능들이 포함되어 있습니다.

설치와 실행

Delphi 2007과 마찬가지로, Delphi 2009의 설치는 InstallAware 기반입니다. 그러나 이번의 Delphi 2009에서는 설치 작업에서, 특히 속도 면에서 많이 개선되었습니다. Delphi 2009의 설치는 이전처럼 몇 시간이 아닌 20분 내에 완료할 수 있게 되었습니다.

설치 관련으로 놀라운 변화는, 헬프를 따로 설치하기 때문에 주 제품과는 별도로, 더 자주 업데이트할 수 있다는 점입니다. 따라서 헬프를 업데이트하려고 Delphi를 다시 설치하거나, IDE를 다시 설치할 때 헬프를 다시 설치할 필요가 없습니다. 헬프를 설치하는 데에는 실제 제품 설치보다 더 많은 시간이 걸릴 수 있으며 헬프 설치 이미지는 IDE 설치 이미지보다 크기가 큼니다.

Windows Vista에서 설치하는 경우 제품은 기본적으로 다음 폴더에 설치됩니다.

```
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 불필요

Delphi 8에서부터 Delphi 2007까지는 IDE를 설치하려면 Microsoft .NET SDK(앞 버전들에서는 버전 1.1, 뒷 버전들에서는 2.0)가 반드시 있어야 했습니다. Delphi 2009에서는 Microsoft .NET SDK가 필요 없습니다. Microsoft .NET의 런타임은 여전히 설치해야 하지만, 크기가 아주 작으며 운영 체제의 일부로 이미 설치되어 있을 수도 있습니다. 하지만 이보다 훨씬 크기가 크고 수백 MB의 공간이 필요한 Microsoft .NET SDK는 설치할 필요가 없습니다.

이번 버전의 Delphi에서 CodeGear는 Microsoft의 Document Explorer(*DExplorer*)를 사용합니다. 이전 버전에서는 SDK에 포함된 형태로만 설치가 가능했지만, 이제는 개별 설치로 배포할 수 있습니다.

Delphi 헬프는 CodeGear 문서와 Microsoft 플랫폼 문서 모두를 포함하므로 설치에 시간이 많이 걸리며 크기가 매우 큼니다. 그러나 이번 릴리스에서 개발팀은 일반 플랫폼 주제들이 나열되기 전에 항상 Delphi 관련 주제들이 먼저 나열되도록 “순서” 문제를 수정했습니다. 또한 Delphi 관련 콘텐츠도 많이 개선되었습니다.

WINDOWS 설치 클린업

경우에 따라 Delphi를 설치 제거한 후 업데이트된 버전으로 재설치할 때, 설치 프로그램이 중지되어 기대한 대로 동작하지 않을 수 있습니다. 이러한 경우, 애플리케이션 폴더(운영 체제에 종속되는 일부 숨겨진 폴더들 포함)를 모두 삭제하는 것이 좋습니다. 또는 다음 경로에 있는 Microsoft의 자체 Windows 설치 클린업 유틸리티를 사용할 수 있습니다.

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

이런 저수준 툴을 사용하면 시스템 작동에 문제가 생길 수도 있으므로, 사용법을 읽은 후 주의해서 사용하십시오.

-IDECAPTION 플래그

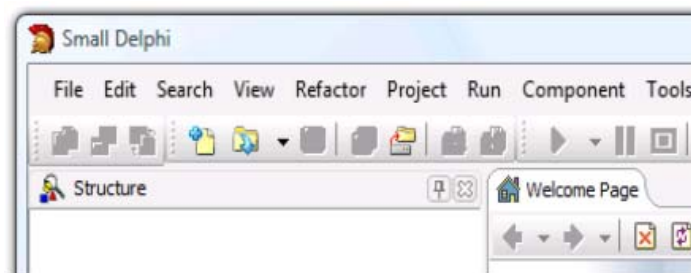
공식적으로 발표되지는 않았지만, 여러분은 -R 커맨드라인 플래그를 사용해 여러 레지스터리 설정으로 여러 IDE 인스턴스를 동시에 실행할 수 있다는 것을 알고 있을 수도 있을 것입니다.

문제는 서로 다른 버전의 두 IDE를 동시에 실행하는 경우, 어느 것이 어느 버전인지 알기가 어렵다는 것입니다. 이 문제를 해결하기 위한 IDE의 커맨드라인 파라미터가 -idecaption인데, 캡션을 이 파라미터의 값으로 넘깁니다. 두 플래그를 합하여 다음 링크로 IDE를 실행할 수 있습니다.

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

이 명령은 더 작은 레지스트리 설정만을 사용하여 Delphi Win32 퍼스널리티만을 사용하는 Delphi IDE를 실행하며, 아래에 표시된 대로 IDE 캡션을 "Small Delphi"로 변경합니다.

커맨드라인에서 IDE 캡션이 지정되지 않은 경우, 레지스트리의 퍼스널리티 섹션에서 캡션을 읽어내며, 이 섹션에는 IDE의 각 버전(또는 활성 퍼스널리티)에 대한 여러 문자열 값들이 있습니다.



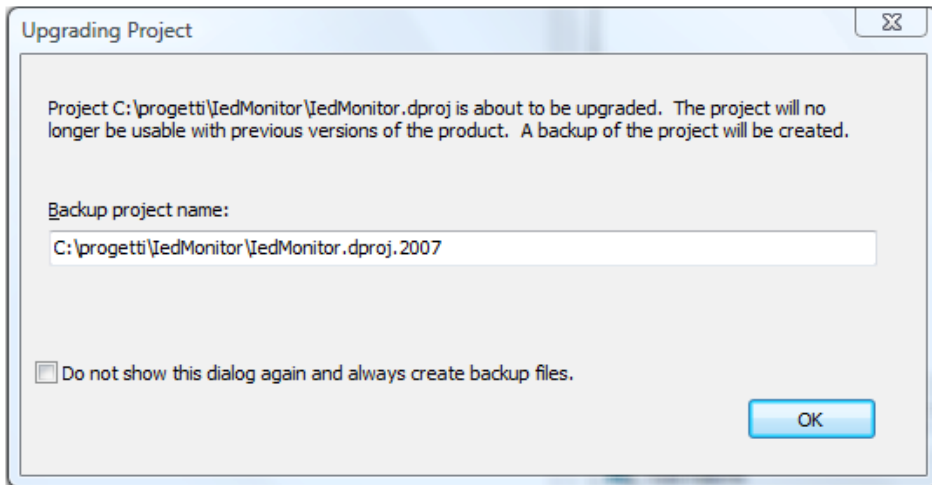
DELPHI 프로젝트 관리

프로젝트 관리는 매우 흔한 작업입니다. Delphi 2007에 MSBuild 지원, 타겟 빌드(Debug/Release), pre빌드/post빌드 이벤트 등 새로운 개념들이 추가되었다면, 이번의 새로운 버전에서는 프로젝트 매니저가 크게 개선되어 이러한 기능을 훨씬 쉽고 유연하게 사용할 수 있게 되었습니다. 그러나 프로젝트 매니저를 살펴보기 전에, 먼저 프로젝트 파일 업그레이드와 새로워진 프로젝트 옵션 다이얼로그를 살펴보겠습니다.

프로젝트 설정 파일 업그레이드

Delphi 초기 버전에서부터 .DPR 확장자를 가진 프로젝트 소스 코드 파일에는 Object Pascal 코드가 포함되어 있었으며, 기타 설정들을 저장하기 위해 하나 이상의 별도 설정 파일이 사용되었습니다. 프로젝트 설정 파일의 형식과 확장자는 최근 버전에서 몇 차례 변경을 거쳐 INI 파일에서 XML 파일로, 그리고 다시 MSBuild(.DPROJ 파일 형식)용 XML 파일로 변경되었습니다.

Delphi 2007에서 Delphi 2009로 업그레이드되면서 이 프로젝트 설정 파일의 전체 형식은 변하지 않았습니다. 그러나 사실상 그 내용은 매우 다르며, Delphi 2007에서는 새로운 버전의 IDE에서 추가된 옵션이 인식되지 않습니다. 기존의 Delphi 2007 프로젝트를 열면, Delphi 2009 IDE는 기존 버전의 프로젝트 설정 파일을 복사해둘 백업 파일 이름을 물어보게 됩니다.



프로젝트 설정 백업 파일의 기본 이름은 프로젝트 이름에 확장자는 .dproj.2007 입니다. 이번 예에서 저는 프로젝트 파일의 이름을 ledMonitor2007.dproj로 변경했습니다. 이 작업을 수행하고 나면 IDE는 다음 줄을 메시지 윈도우에 추가합니다.

```
Upgrading project. Backup  
C:\progetti\ledMonitor\ledMonitor2007.dproj created.
```

프로젝트 설정 파일의 업데이트된 Delphi 2009 버전은 실제로 저장할 때까지 생성되지 않습니다.

백업 버전을 사용하면 Delphi 2007에서 프로젝트를 다시 열 수 있습니다. 그러나 이전 버전과의 호환성이 필요하면 프로젝트의 Delphi 2009 버전을 다른 이름으로 저장하는 것이 좋습니다.

새 .DPROJ 파일에서 Delphi 2009는 새로운 프로젝트 버전 태그를 추가합니다.

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

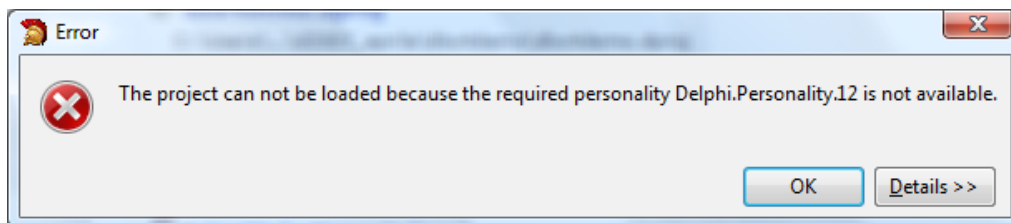
이런 프로젝트 파일 업그레이드는 나중에 설명할 빌드 컨피규레이션 및 리소스 관리에서의 변경 사항과 관련되어 있습니다. 다음은 신규 또는 많이 수정된 섹션들입니다.

```

<PropertyGroup Condition="'$(Configuration)' == 'Release' or
  '$(Cfg_Release)' != ''">
  <Cfg_Release>true</Cfg_Release>
  <CfgParent>Base</CfgParent>
  <Base>true</Base>
</PropertyGroup>
<PropertyGroup Condition="'$(Configuration)' == 'Debug' or
  '$(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>

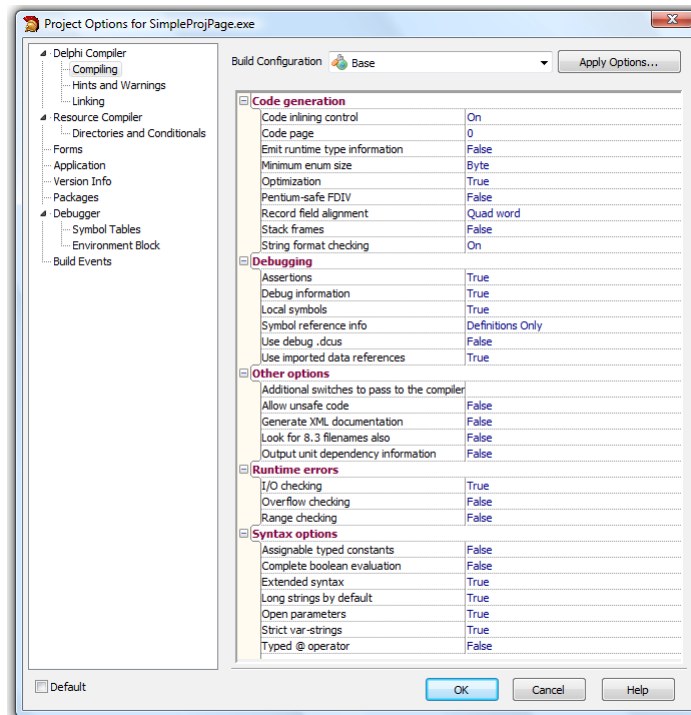
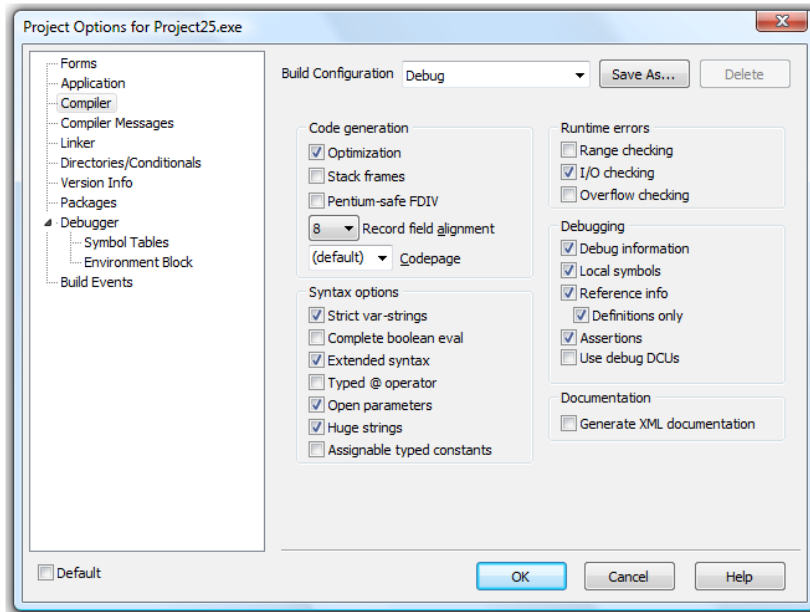
```

Delphi 2007(이런 형식을 인식하는 유일한 이전 버전)에서 이 프로젝트 파일을 다시 열려고 할 경우, 다음과 같은 에러가 표시됩니다.



재설계된 프로젝트 옵션 다이얼로그

프로젝트 옵션 다이얼로그는 개인적으로 더 자주 사용하는 Delphi 다이얼로그 중 하나이며, 저만 그럴지는 않을 것입니다. 그 때문에 Delphi 2009에서 광범위하게 재설계된 프로젝트 옵션 다이얼로그로 인해 때때로 당황하게 됩니다. 재설계는 빌드 컨피규레이션의 일부인 옵션이 포함된 페이지와 관련이 있으며 (“빌드 컨피규레이션 및 컨피규레이션 설정” 섹션에서 설명), 이런 다이얼로그 페이지는 프로젝트 설정 매니저 내에서도 사용됩니다. 예를 들어, Delphi 2007과 Delphi 2009 사이에서 Delphi Compiler Options 페이지의 차이를 살펴보겠습니다.



차이가 매우 큽니다. 체크박스는 “True/False”로 바뀌었으며, 라디오버튼은 여러 선택이 가능한 콤보박스로 바뀌었습니다. 또, 다이얼로그의 바닥에 있는 헬프 영역에는 다양한 선택에 대한 간단한 정보를 제공됩니다. (다이얼로그에 페이지의 모든 옵션들을 맞추기 위해 위 그림에서는 최소화됨) 흥미로운 점 한가지는, “설명” 영역에서 옵션에 대한 기본값을 보여준다는 것입니다.

화면상의 재설계는 익숙해지기까지 상당한 시간이 걸립니다. 게다가 각 그룹 내 항목들이 이제 알파벳 순으로 나열되기 때문에 이전과는 다른 순서로 나열됩니다. 또 디렉토리 옵션은 Delphi 컴파일러 메인 노드로 옮겨졌습니다. 하지만 구조적 변경 외에, 없어지거나 새로운 것이 있지는 않을까요?

컴파일러에 대한 새 프로젝트 옵션

예전에는 Compiler 페이지였던 Delphi Compiler/Compiling 페이지는 Code Generation 섹션에는 다음과 같은 새 옵션이 있습니다.

- *Code inlining control*은 **\$INLINE** compiler 지시어에 해당하며, 인라인 함수의 처리 방법을 결정합니다.
- *Emit runtime type information*은 커맨드라인 **-\$M** 플래그 또는 **\$M** 지시어에 해당하며, 지정된 클래스, 또는 프로젝트의 모든 클래스에 대한 런타임 시간 정보의 생성을 결정합니다.
- *Minimum enum size*는 **-\$Z** 플래그(혹은 **\$Z** 지시어)에 해당하며 열거 형식의 값에 사용될 최소 크기를 결정합니다. (Byte, Word, Double Word, Quad Word)
- *String format checking*은 기본적으로 ON으로 설정되며, 비활성화하면 **EnsureUnicodeString** 함수 및 **Ensure String** 계열의 기타 함수에 대한 호출과 같은 일부 문자열 형식 자동 확인 작업을 방지할 수 있습니다. 이 옵션은 **\$STRINGCHECKS** 지시어에 해당합니다. 이 컴파일러 옵션은 Delphi 2009의 새로운 특징으로, 이전에는 문서화되지 않고 사실상 숨겨져 있었습니다. 프로젝트 옵션 다이얼로그에 나타난 것은 꽤 놀랍군요.
- *Code page*는 이전 버전에도 이미 있었지만 이제 **AnsiString** 형식이 작동되는 방법과 관련성이 훨씬 많아졌습니다. (이는 이 문서의 시리즈 문서인 “Delphi와 Unicode”에 설명되어 있습니다.)

Debugging 섹션에는 새로운 옵션인 *Use imported data references*(**\$G**에 해당됨)가 추가되었으며, 이 옵션은 임포트 데이터 참조의 생성을 제어합니다. 이 옵션은 메모리 효율성을 높이지만 다른 런타임 패키지에서 정의된 글로벌 변수를 액세스할 수 없게 됩니다.

Runtime errors 및 Syntax options 섹션의 요소들은(그리고 기본값들도) 이전 버전의 Delphi와 동일합니다. Other options 섹션에는 다음과 같은 새로운 옵션들이 있습니다. (이전에도 있었던 *Generate XML documentation*은 제외)

- *Additional switches to pass to the compiler*는 IDE가 명확하게 지원하지 않는 추가 커맨드라인 컴파일러 옵션을 직접 삽입하는 데 사용됩니다. 이제 이 기능을 사용할 수 있다는 것은 기술적으로 볼 때 Delphi 2009가 모든 컴파일러 옵션을 지원한다는 것을 의미합니다.
- *Allow unsafe code* 옵션을 사용하면 .NET과 같은 매니지드 환경에서 안전하지 않다고 간주되는 코드를 컴파일할 수 있으며, Win32 컴파일러에서는 사용할 수 없습니다.
- *Look for 8.3 filenames*는 Windows의 이전 버전에서 작동하도록 컴파일러에 지시하며 **-P** 컴파일러 옵션에 해당합니다.
- *Output unit dependency information*은 **--depends** 컴파일러 플래그를 설정합니다. (현재로서는

명백하게 관리되지 않는 듯합니다)

기타 새 프로젝트 옵션

Hints 및 Warnings 페이지는 이전 **Compiler Messages** 페이지에 해당합니다. 당연히 **Unicode** 문자열 및 기타 새로운 컴파일러 기능과 관련된 몇몇 새로운 힌트가 있습니다.

예전에는 **Linker** 페이지였던 **Linking** 페이지는 보기에 많이 달라 보이지만(더 간결해 보이기도 합니다), 새로운 옵션은 *Set base address for relocatable images*뿐입니다.

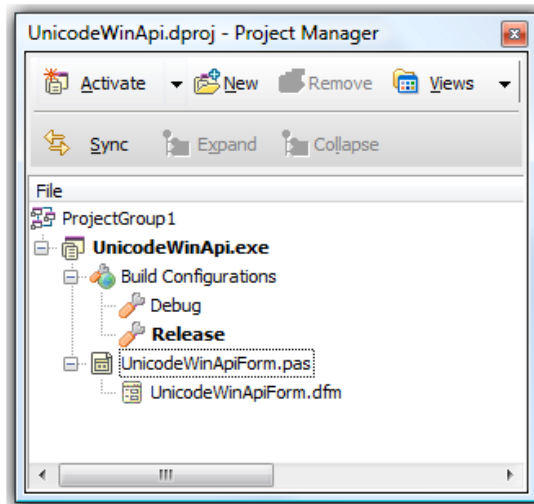
Delphi Compiler 페이지의 메인 노드에는 이전에 **Directories/Conditionals** 아래에서 볼 수 있었던 것과 동일한 옵션이 있습니다. 혼동을 일으킬 수 있는 부분은 리소스 컴파일러 구성의 일부로서 **Directories and Conditionals**라는 다른 페이지가 **Resource Compiler** 페이지의 메인 노드 아래에 있다는 것입니다. **Resource Compiler** 노드의 페이지들은 완전히 새로운 페이지들로, 리소스 컴파일러를 Delphi IDE에서 제어할 수 있게 해주는 설정들로서, 예전에는 불가능했던 것입니다. 이 문서 뒷부분의 “IDE에서의 리소스 관리”라는 특정 섹션에서 이 항목에 대해 다룹니다.

기본 프로젝트 위치

Delphi 2005부터 모든 새 프로젝트에 대한 기본 위치는 사용자의 내 문서 폴더 아래에 있습니다. 의외로, **Tools | Options** 다이얼로그의 **Environment Options** 페이지에 있는 **Default Project** 에디트에서 값을 설정하면 이 기본 프로젝트 위치를 변경할 수 있다는 것을 아는 Delphi 개발자가 매우 적습니다.

프로젝트 매니저

프로젝트 옵션 다이얼로그의 재설계와 함께, Delphi 2009에서는 가장 일반적으로 사용되는 IDE 윈도우들 중 하나인 **Project Manager** 윈도우에 대한 중요한 업데이트를 볼 수 있습니다. 이 윈도우를 간단히만 훑어보아도 다음과 같은 몇몇 새로운 기능을 볼 수 있습니다.



하위 노드가 포함된 새 **Build Configuration**(빌드 컨피규레이션) 노드가 있으며, 이 노드를 사용하면 **Delphi 2007**에서보다 훨씬 간단하게 특정 빌드 컨피규레이션을 활성화할 수 있습니다. 이는 "빌드 컨피규레이션 및 컨피규레이션 설정" 섹션에서 다룹니다.

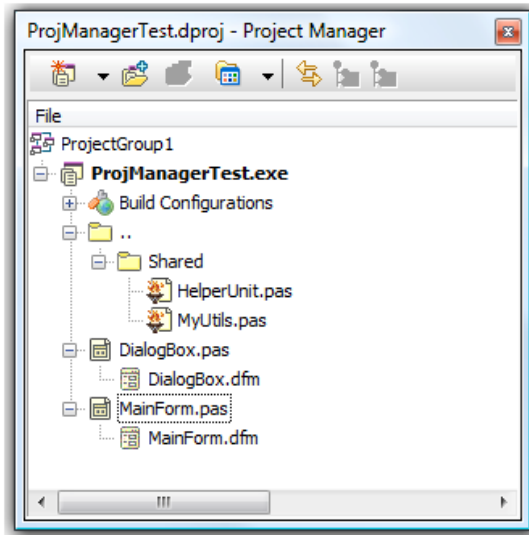
Project Manager 툴바에는 몇가지 새로운 버튼들이 있습니다. 새로 추가된 **Sync** 버튼을 누르면 코드 에디터의 현재 파일이 프로젝트 매니저에서 선택됩니다. 물론 파일이 프로젝트에 포함되어 있을 경우입니다. 반대로, 즉 프로젝트 매니저의 현재 선택된 항목을 코드 에디터에서 활성화하려면 더블 클릭을 하면 됩니다.

Expand 및 **Collapse** 버튼은 현재 노드 아래의 모든 노드를 축소 혹은 확장합니다. 프로젝트 그룹에서 **Expand** 버튼을 누르면 그룹 내의 모든 프로젝트의 모든 컨피규레이션 및 파일 노드가 있는 트리를 볼 수 있으며 이는 매우 유용합니다. 네 번째 새로운 버튼인 **Views**는 다음 섹션에서 다룹니다.

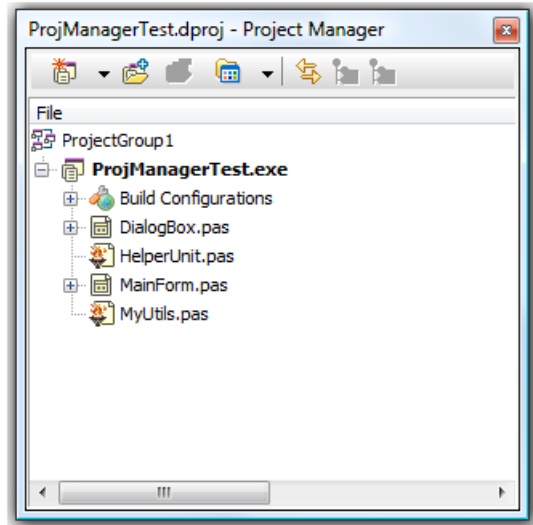
프로젝트 매니저 뷰

또 다른 새 기능은 프로젝트 매니저 뷰 설정입니다. 툴바의 오른쪽에는 새로운 **Views** 버튼이 있으며, 이 버튼을 사용하면 프로젝트 매니저가 다른 폴더에 위치한 파일들을 표시하는 방법을 변경할 수 있습니다. 여기에는 3가지 옵션이 있습니다. 이 세가지 보기 옵션을, 두 개의 품이 포함된 주 폴더와 두 개의 유닛이 포함된 **Shared**라는 보조 폴더가 디렉토리 구조에서 같은 레벨에 있는 샘플 프로그램 (**ProjManagerTest**)를 작성하여 이 세 가지 보기 옵션을 테스트했습니다.

- **Directory (Nested)**는 기본 설정(**Delphi 8**로부터 **Delphi 2007**까지의 유일한 옵션)으로, 파일을 디렉토리별로 그룹화하여 표시하며 디렉토리는 확장 가능한 개별 노드가 있는 실제 디스크 구조와 비슷합니다. (따라서 여러 단계의 하위 폴더로 가려면 여러 노드를 확장해야 합니다.)



- **Directory (Flat)**은 새로운 뷰로서, 파일들이 디렉토리별로 나누어지지만, 디렉토리상의 위치와는 관계 없이 각각의 디렉토리들이 나열됩니다. 다시 말하면, 이 뷰에는 폴더 리스트가 나타나며 각 폴더에는 다른 하위 폴더가 아닌 파일들이 포함됩니다.
- **List**는 새로운 뷰로서, 프로젝트 매니저에서 전통적인 **Delphi 7** 스타일의 파일 리스트의 형식으로 나타납니다. 디렉토리는 그냥 무시되며, 파일을 알파벳 순으로 나열할 수 있습니다.



빌드 컨피규레이션과 컨피규레이션 설정

이미 설명한 것처럼 (또한 이전 페이지의 이미지에서 볼 수 있듯이), 프로젝트 매니저에는 모든 프로젝트에 대해 빌드 컨피규레이션(Build Configuration) 노드가 있습니다. 이 노드는 **Delphi 2007**에서 빌드 컨피규레이션을 관리하는 데 사용했던 다소 번거로웠던 개별 윈도우를 대신합니다. 노드와 하부 노드를 사용하면, 단지 더블 클릭만으로 현재 빌드 컨피규레이션을 변경하고 지정된 노드에서 직접 실제 빌드를 할 수 있습니다.

빌드 컨피규레이션이나 메인 노드를 선택하여 새로 컨피규레이션을 추가할 수 있습니다. 작업할 때 선택한 항목에 따라 메인 컨피규레이션 또는 하위 컨피규레이션을 생성하게 됩니다. 더 정확히 얘기하자면, 개발자가 선택하는 노드에 따라 베이스 컨피규레이션(Base Configuration)이 결정됩니다. 미리 정의되어 있는 컨피규레이션들도 베이스 컨피규레이션의 핵심 설정을 상속하기 때문입니다.

제가 컨피규레이션에서 “설정을 상속한다”라고 한 말이 무슨 뜻일까요? **Delphi 2009**의 새 컨피규레이션 관리 시스템에서는, 설정을 디버그나 릴리스 등의 특정 컨피규레이션에만 적용하거나, 그 두 컨피규레이션이 베이스 컨피규레이션으로부터 상속하도록 옵션을 설정할 수 있습니다. 특정 컨피규레이션에서 특정한 값과 함께 베이스 컨피규레이션(Base Configuration)으로부터 상속된 값이 이어서 나오는 것을 볼 수 있으며, 두 값이 일치하는지 보고 두가지 중 하나의 값을 수정할 수 있습니다(상위 컨피규레이션에서 설정하면 특정 컨피규레이션에도 반영됩니다). 이렇게 하려면 왼쪽에 있는 플러스 기호를 선택하여 각 컨피규레이션 설정 줄을 확장하면 됩니다.

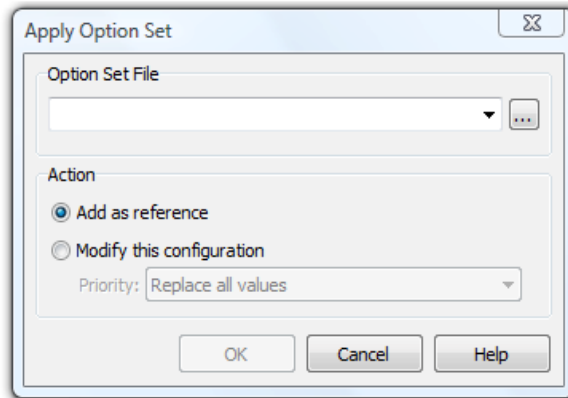
Delphi Compiler/Compiling 페이지에서 3개의 런타임 에러 줄을 확장하면 다음과 같은 결과가 나타납니다.

[-] Runtime errors	
[-] I/O checking	True
Value from "Base"	True
[-] Overflow checking	False
Value from "Base"	True
[-] Range checking	True
Value from "Base"	True

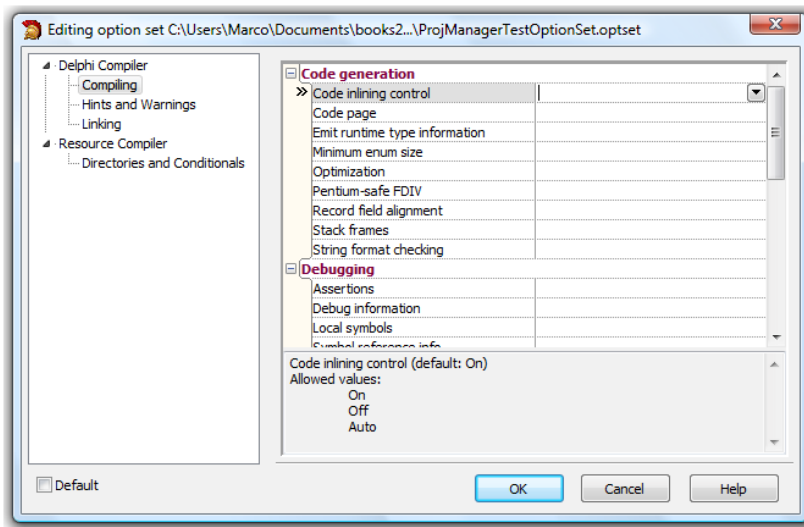
베이스 컨피규레이션에서 설정을 수정하면 해당 설정에서 상속된 다른 컨피규레이션에 영향을 줍니다.

프로젝트 매니저에서 빌드 컨피규레이션을 선택하고 해당 설정을 “option set” 파일로 내보낼 수 있습니다. 이것은 컨피규레이션의 템플릿 또는 기본 구조를 외부 파일에 저장하는 것과 비슷하며, 컨피규레이션은 그 파일에 링크됩니다.

이 기능 덕분에 프로젝트 매니저(빌드 컨피규레이션 항목에서 팝업 메뉴 항목 **Apply Options Set** 사용)나 프로젝트 옵션 다이얼로그(**Apply Options** 버튼 사용)를 사용할 때 새 프로젝트나 기존 프로젝트로 간단히 설정을 이동할 수 있습니다. 두 경우 모두 Delphi에서 **Apply Option Set** 다이얼로그가 나타나며, 이 다이얼로그에서 파일을 선택하고 외부 컨피규레이션 파일을 링크하거나(파일의 변경 사항이 해당 파일을 사용하는 프로젝트에 반영되도록), 어떤 우선순위(priority)에 따라 현재 설정에 통합되도록 할 수 있습니다.



일단 외부 옵션 세트 파일을 만든 후에는, 프로젝트 매니저의 **Edit** 팝업 메뉴를 사용하면 해당 옵션 파일을 링크하는 어떤 프로젝트에서든 옵션을 편집할 수 있습니다. 그러면 아래와 같은 프로젝트 옵션 다이얼로그의 일부 페이지들을 가지는 옵션 세트 에디터가 열립니다.



.OPTSET 파일은 .DPROJ의 포맷과 유사한 포맷의 XML 파일로서, 역시 MSBUILD XML 포맷을

기반으로 하며 **OptionSet** 프로젝트 타입입니다. 이 예제에서 **ProjManagerTestOptionsSet.optset** 파일의 내용은 다음과 같습니다.

```

<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>

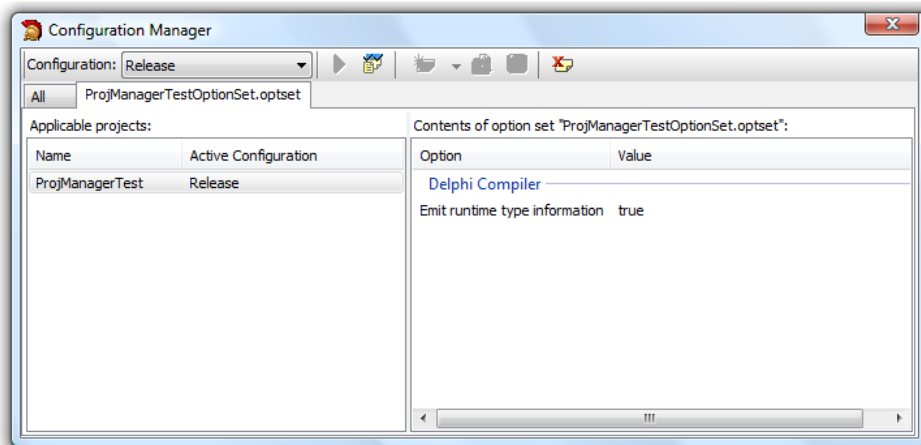
```

프로젝트 컨피규레이션 매니저

프로젝트 매니저에서 빌드 옵션을 직접 사용할 수 있으므로, 현재 빌드 컨피규레이션을 변경하기 위해 컨피규레이션 매니저를 사용할 필요가 없습니다. 하지만 **Configuration Manager** 다이얼로그를 사용하면 프로젝트 그룹 내의 여러 프로젝트에 대해 한번에 빌드 컨피규레이션을 변경할 수 있으므로 상당히 유용합니다. 사실 컨피규레이션 매니저는 **Delphi 2009**에서 계속 사용할 수 있으며 여러가지 면에서 개선되었습니다. 한번에 그룹의 모든 프로젝트에 대한 다양한 빌드 컨피규레이션 및 옵션 세트를 관리할 수 있게 되었습니다.

컨피규레이션 매니저를 불러내려면 **Delphi 2007**에서처럼 프로젝트 매니저의 팝업 메뉴를 사용할 수는 없으며, **Delphi** 메인 메뉴의 **Project** 메뉴에서 해당 항목을 선택해야 합니다.

이렇게 하면 다음과 같이 새롭게 설계된 사용자 인터페이스가 표시됩니다.



왼쪽에는 프로젝트와 각 프로젝트에 대한 활성 컨피규레이션 리스트가 표시됩니다. 오른쪽에는 기본 설정이 아닌 설정 등 선택된 컨피규레이션에 대한 몇몇 세부 정보들이 표시됩니다.

(위 이미지의 내용은 이전 섹션에서 설명했던 옵션 세트 파일의 요약입니다.) 탭 페이지를 통해 지정된 컨피규레이션 또는 활성 옵션 세트에 따라 왼쪽에 있는 프로젝트들을 필터링할 수도 있습니다.

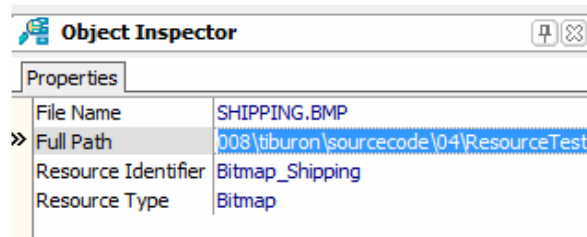
Delphi 2009의 컨피규레이션 매니저를 사용하면 각 빌드 컨피규레이션에 대해 프로젝트 옵션을 편집하거나, 새 컨피규레이션을 추가하거나, 옵션 세트를 만들고 수정하거나, 활성 컨피규레이션을 수정할 수 있으며, 만만한 작업은 아니기는 하지만, 관련 작업들 대부분을 한곳에서 할 수 있습니다.

프로젝트 그룹에서 여러 개의 프로젝트를 작업할 경우, 컨피규레이션 매니저는 프로젝트 매니저에서 찾아 일일이 빌드 컨피규레이션 작업을 하는 것보다 훨씬 이점이 많습니다. 단일 프로젝트의 경우 프로젝트 매니저만으로도 필요한 모든 기능을 사용할 수 있습니다.

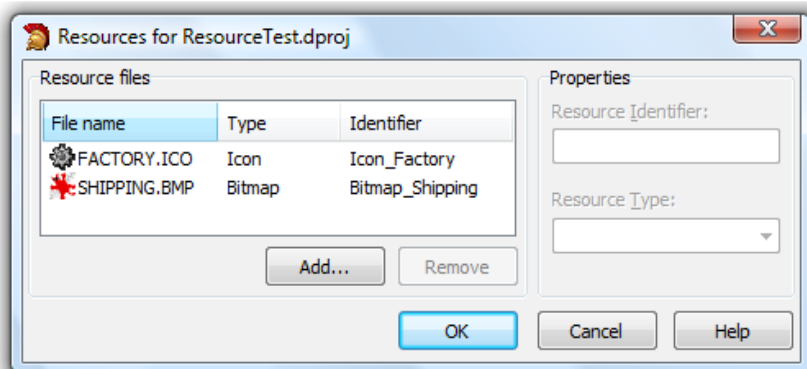
리소스 관리를 IDE에서

최신 버전의 Delphi에서는 리소스 스크립트(.RC 파일) 또는 컴파일된 리소스 파일(.RES 파일)을 프로젝트 매니저에 추가하여 프로젝트와 함께 실행 파일에 링크할 수 있습니다. Delphi 2009에서는 몇가지 도구들이 추가되어 리소스 관리가 더 간편해졌습니다.

먼저, 이제 아이콘, 비트맵 등의 리소스 파일을 프로젝트 매니저로 드래그하여 프로젝트의 리소스로 포함시킬 수 있게 되었습니다. Delphi는 이런 추가 프로젝트 리소스에 대한 리소스 스크립트 파일을 생성하고 해당 파일을 프로그램과 함께 직접 컴파일해서 실행 파일에 포함시킵니다. 다음과 같이 오브젝트 인스펙터에서 이런 리소스 파일의 속성을 변경할 수 있습니다. (내부 이름도 지정 가능)



다음으로, IDE 메인 메뉴의 **Projects** 메뉴에 새로운 **Resources** 메뉴 항목이 추가되었습니다. 이 항목을 선택하면 리소스 다이얼로그가 나타나며, 이 다이얼로그에서 프로그램의 모든 리소스를 수정하고, 새 리소스 파일을 추가하고, 리소스 이름을 변경하며, 포맷을 변경하는 등의 작업을 할 수 있습니다.



리소스들을 프로젝트에 추가하면 Delphi는 컴파일할 때 적절한 리소스 파일을 생성합니다. 위의 이미지에서 보이는 리소스들을 포함하는 프로젝트의 이름이 **ResourceTest**일 경우, Delphi 2009는 프로젝트 리소스들의 리스트를 가진 **ResourceTestResource.rc**라는 리소스 스크립트 파일을 생성합니다.

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

이 리소스 스크립트 파일은 프로젝트에 추가되지는 않지만(추가할 경우 duplicate resource 경고가 나타납니다) 프로젝트와 함께 컴파일됩니다. 예를 들어 비트맵을 아이콘으로 선언하는 등의 실수를 하게 되면 컴파일러는 다음과 같은 에러가 내고,

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

리소스 스크립트 파일이 열리고 잘못된 코드 라인을 보여줍니다. Delphi 2009는 컴파일을 할 때 리소스 스크립트 파일을 생성 혹은 수정하여 컴파일하고 실행 파일에 포함시킵니다. 이 과정에서 중간 파일로서 확장자가 **DRES**인 파일이 생기는데, 프로젝트 소스 코드 파일에 자동으로 추가되는 컴파일러 지시어에 의해 프로젝트에 포함됩니다. (프로젝트 아이콘과 문자열 리소스를 가진 표준 **RES** 파일도 포함됩니다)

```
program ResourceTest;

{$R *.dres}

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

{$R *.res}

begin
    Application.Initialize;
    ...
end;
```

Delphi 2007 버전부터 리소스 컴파일 단계에서 **MSBuild** 지원이 추가되었습니다. 다음의 출력 내용은 리소스 컴파일러 옵션의 **-Verbose** 플래그를 설정하면 확인할 수 있습니다.

```
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

```

Windows 리소스를 직접 사용해 본 경험이 없는 경우 여러분이 참고할 수 있도록, ResourceTest 프로그램에서 리소스의 아이콘을 애플리케이션 아이콘 및 메인 폼 아이콘으로 로드하고 리소스의 비트맵을 이미지 컴포넌트로 로드하는 코드를 보여드립니다.

```

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

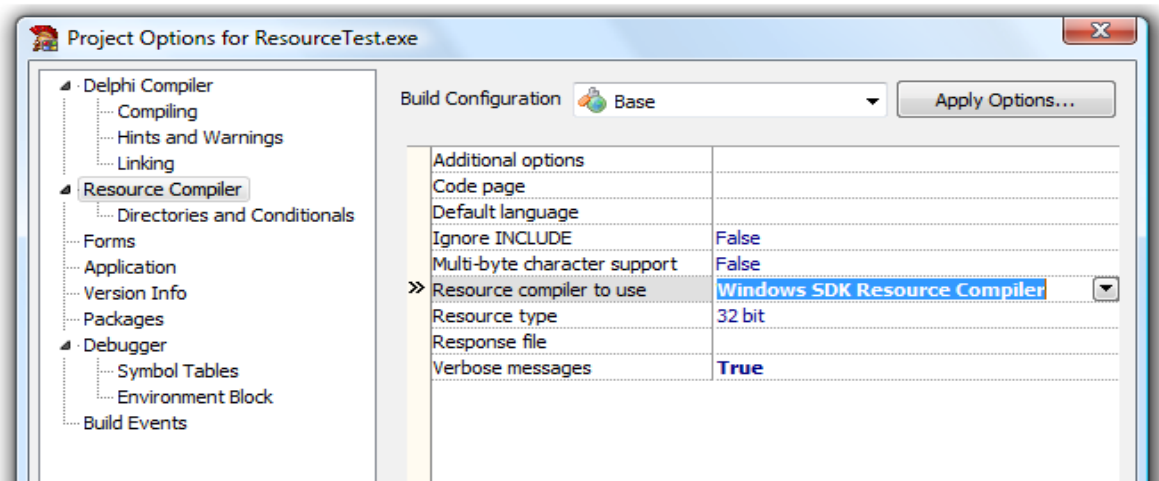
procedure TFormResourceTest.btnIconCIck(Sender: TObject);
begin
  Icon.LoadFromResourceName(hInstance, 'Icon_Factory');
  Application.Icon.LoadFromResourceName(
    hInstance, 'Icon_Factory');
end;

```

새 리소스 컴파일러

이전 버전의 Delphi에서는, 즉 Delphi 2007 버전까지는 Borland Resource Compiler(BRCC32.EXE)를 사용했습니다.

Delphi 2009에는 새 리소스 컴파일러, 더 정확히 말하자면 전과는 다른 리소스 컴파일러인 Microsoft Windows SDK의 컴파일러를 제공합니다. 이는 Windows가 다루는 모든 새로운 리소스 포맷들을 지원한다는 면에서는 장점이 많지만, Borland 리소스 컴파일러는 예전부터 Microsoft 리소스 컴파일러의 기능을 확장했다가 지금은 사용되지 않는 추가 기능들을 제공하는 것 때문에 몇 가지 문제가 발생합니다. 프로젝트 옵션 다이얼로그의 Resource Compiler 섹션에서 해당 옵션을 설정하면 사용할 리소스 컴파일러를 선택하여 지정할 수 있습니다. (프로젝트 옵션 다이얼로그에서는 리소스 컴파일러의 몇가지 파라미터들을 설정할 수도 있습니다.)



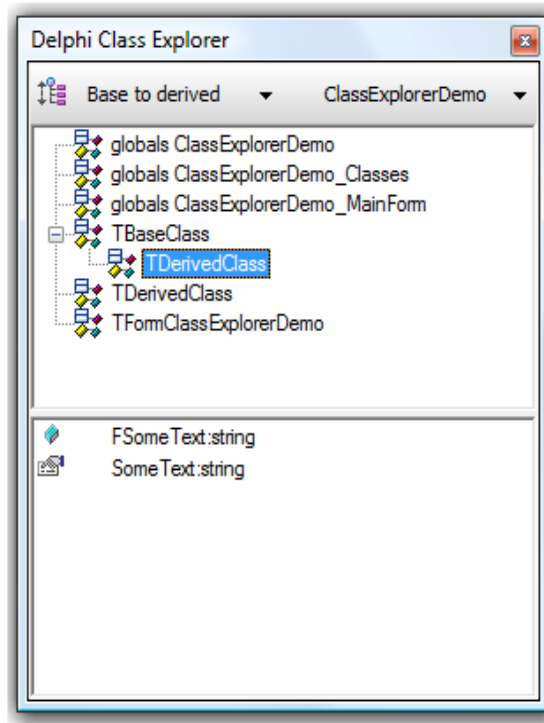
Windows SDK Resource Compiler는 SDK 컴파일러에 대한 프런트 엔드인 새 CodeGear Resource Compiler/Binder를 통해 호출됩니다. 그 외에 리소스 컴파일러 관련 기능의 새로운 점으로는, 이미지(바이너리) 데이터의 인라인 처리, 스트링리스트에서 문자열 끝의 쉼표에 대한 지원, 문자열 처리 방식의 변경(C-언어 문자열로 취급되므로 파일 이름 내의 \를 두개의 백슬래시로 써야 함), 파일 인클루드 관련으로 폴더를 관리 방법의 변경 등이 포함되었습니다.

리소스 파일을 직접 사용해본 적이 없다면 이러한 변경 사항을 무시할 수도 있습니다. DFM 파일을 리소스로 포함하는 것부터 **resourcestring** 선언의 사용에 이르기까지, Delphi 환경에서 직접 관리되는 모든 작업들이 이전 버전과 완벽히 호환됩니다. 리소스를 직접 사용하는 경우 주의를 기울여 리소스 파일을 수정해야 합니다.

DELPHI 클래스 익스플로러

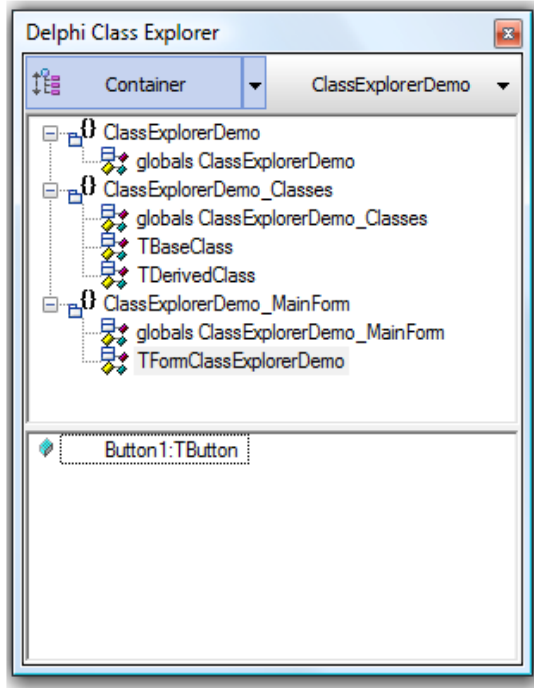
Delphi 2009에서 새롭게 등장한 윈도우로, Delphi 클래스 익스플로러 윈도우가 있습니다(View 메뉴의 Delphi Class Explorer 메뉴 항목). Delphi 클래스 익스플로러는 프로젝트 전반에 걸친 심볼들의 다이어그램을 보여주며, 이는 단일 유닛의 요소들을 비주얼하게 보여주는 Structure View와는 약간 비슷해 보이지만 다른 것입니다.

Delphi 클래스 익스플로러의 첫 수준에서는 각 유닛과 프로젝트 파일의 글로벌 정의인 노드 목록이 나타나며, 나머지 노드는 프로젝트에서 정의된 모든 클래스들을 보여 줍니다.



각 클래스에 대해 특정 멤버(상속된 멤버 아님)와 다른 클래스와의 관계를 볼 수 있습니다. 표시 방식은 툴바의 첫번째 버튼의 선택에 따라 달라집니다. **Base to derived**(위 그림과 같은 상태), **Derived to base** 또는 **Container** 세가지 중의 하나입니다. **Container**의 경우 클래스들과 글로벌들은 유닛 별로 나누어지며 상속 관계는 표시되지 않습니다.

팝업 메뉴를 이용하여 아래 이미지에서 보는 것처럼 클래스에 새 필드, 새 메소드(생성자 및 소멸자 포함) 또는 속성을 추가할 수 있습니다.



속성을 추가하면 (UML 기반 모델링을 사용하는 것보다 훨씬 더) Delphi에 적합한 방식으로 동작합니다. 이 도구는 기본적으로는 read 및 write 메소드로 연결되려는 경향이 있지만, 원하는 경우 Create Field 에디트를 통해 필드를 추가하고 해당 필드가 생성되도록 요청하고 직접 필드에 연결할 수 있습니다.



Delphi 클래스 익스플로러는 이전 스크린 샷의 내용에 따라 다음과 같이 클래스에 코드를 추가합니다.

```

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;

```

strict private var 블록을 사용하여 이상해 보이겠지만, 이는 형식상 정확하며, **var** 예약어를 추가함으로써 코드 생성도 쉬워지고 위험이 줄일 수 있을 것입니다. 저는 속성을 선언하는 것 외에도 작업할 것이 많아서 코드를 다시 포맷해야 해서, 속성만을 선언한 후 클래스 컴플리션을 사용하는 경우가 많습니다. 이 편이 더 깔끔한 표준 **Delphi** 코드를 생성하기도 합니다. 개인적으로 볼 때, **Delphi** 클래스 익스플로러는 프로젝트의 소스 코드를 탐색하는 데 효율적인 도구이며, **UML** 다이어그램을 생성하는 경우가 아니라면 모델 뷰보다는 **Delphi** 클래스 익스플로러를 사용하는 것을 선호합니다.

기타 새로운 기능들

업데이트된 프로젝트 옵션 다이얼로그, 프로젝트 매니저의 새로운 기능들과 확장된 빌드 컨피규레이션, 개선된 리소스 지원, 그리고 클래스 익스플로러 등이 **Delphi 2009**의 IDE에서 가장 눈에 띄는 새 기능일 것입니다. 물론, IDE 전반에 걸쳐 **Unicode**가 지원되게 되었다는 사실을 제외한다면 말입니다.

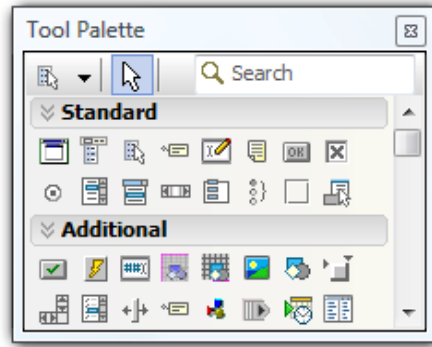
이 섹션에는 **Delphi** 개발 환경에서의 일상적인 작업에 도움이 될 수 있는 많은 기타 기능이 나열되어 있습니다. 주목할 만한 기타 IDE 개선 사항은 다음 사항과 관련이 있습니다.

- ITM(Integrated Translation Manager), **Unicode** 지원을 위해 수정되고 여러 영역에서 개선됨.
- COM 지원 및 타입 라이브러리 편집기에서의 대규모 변경 사항과 관련된 IDE의 변경 사항

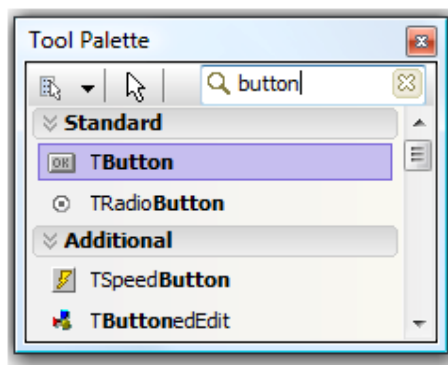
툴 팔레트 서치 박스

Delphi 2006에서는 툴 팔레트를 선택한 상태에서 키 타이핑을 하여 그 문자들로 시작하는 컴포넌트들을 필터링하여 보여주는 기능이 추가되었습니다(시작의 T는 제외). **Delphi 2007**에서는 그에 더해 컴포넌트 이름 안의 텍스트를 타이핑하여 필터링할 수 있도록 기능이 더 강화되었습니다. 예를 들면, **HTTP**를 타이핑하여 **IdHTTP**를 선택할 수 있었습니다.

Delphi 2009의 툴 팔레트는 **Delphi 2007**과 같은 동작을 하지만, 컴포넌트 리스트를 검색할 수 있다는 것이 더 명확하게 느껴지도록 유저 인터페이스에 변화가 있었습니다.



툴 팔레트를 선택하면(단축키 **Ctrl+Alt+P**) 서치 박스에 입력을 시작할 수 있습니다(윈도우의 캡션이 아니라). 툴 팔레트는 필터링된 컴포넌트들을 다음과 같이 보여줍니다.

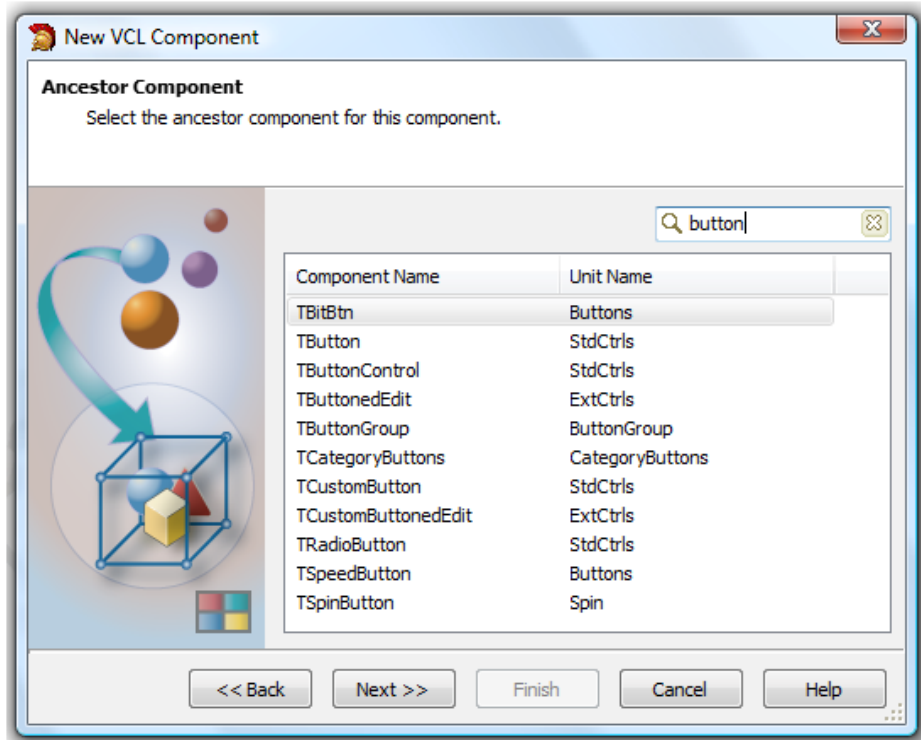


툴 팔레트에는 또 다른 변경 사항이 있습니다. 목록의 아래쪽의 카테고리에 도달하려면 지나치게 많이 스크롤해야 한다고 많은 사람들이 불평을 해왔습니다. 이제 카테고리 자동 축소가 기본 동작으로 변경되었습니다. 컴포넌트를 선택한 후 서치 박스의 선택이 유지될지의 여부도 사용자에게 따라 유용하게 설정할 수 있습니다.

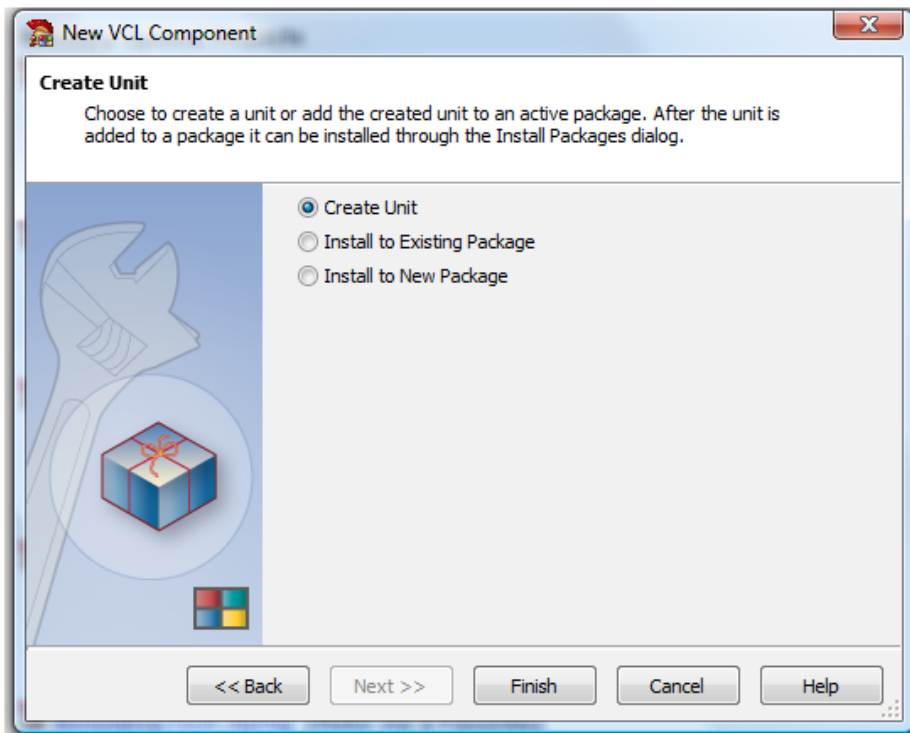
업데이트된 컴포넌트 위저드

새 VCL 컴포넌트를 만들거나 컴포넌트(COM 컨트롤처럼 사용하기 위한 **ActiveX** 컨트롤이나 **.NET** 어셈블리)를 가져오는 데 사용되는 다이얼로그가 개선되어 여러 단계를 가진 위저드로 변경되었습니다. 빈 컴포넌트 구조를 생성하거나 외부 컨트롤을 래핑하는 기능 자체는 크게 수정되지 않았습니다. 유일한 새 기능은 컴포넌트를 기존 패키지 또는 새 패키지에 설치할 수 있는 기능입니다(두 위저드 모두에 해당).

이와 관련하여, IDE의 컴포넌트 메뉴에서 활성화할 수 있는 이 위저드의 사용자 인터페이스도 변경되었습니다. 예를 들어 **New VCL Component Wizard**의 첫 페이지에는 상속받을 베이스 클래스 컴포넌트를 필터링하기 위한 서치 박스가 있습니다.



이 위저드를 계속 진행하여 클래스 이름과 기타 표준 세부 사항을 모두 채우면 마지막 페이지에 이르게 되고, 새 패키지 생성 혹은 새 컴포넌트를 기존 패키지에 추가 중에 선택할 수 있습니다. 활성 패키지 프로젝트가 있으면 새 컴포넌트를 그 프로젝트에 추가하기 위한 추가 옵션을 볼 수 있습니다.

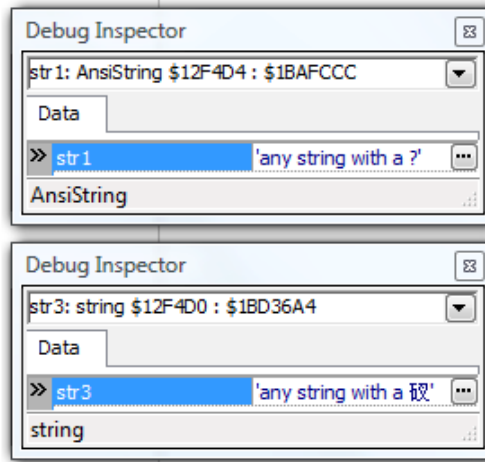


비슷한 기능이 Import Component Wizard에도 추가되었습니다.

디버거

IDE의 나머지 모든 부분들처럼 디버거도 **Unicode**를 완벽하게 지원합니다. 이전 버전들에서도 부분적으로 이런 **Unicode** 지원이 가능했었지만, **Delphi 2009**에서는 이를 확장했습니다.

예를 들어, **Run | Inspect**(또는 에디터 팝업 메뉴의 **Debug | Inspect**)로 문자열 변수를 검사할 경우, 정확한 **Unicode** 값을 볼 수 있을 뿐만 아니라 하단에서 변수의 실제 문자열 타입을 알려줍니다. 아래는 **AnsiString** 및 **UnicodeString**(간단히 **string**으로 보여줌)에 대한 **Inspect** 창을 비교한 것입니다.



두 윈도우는 동일한 문자열을 표시하지만 첫 번째 윈도우는 중국어 문자 때문에 적절히 변환될 수 없습니다.

디버거의 다른 새로운 기능들은 **CPU** 뷰에서의 **SSE3** 및 **SSE4** 명령 지원과 같이 **Unicode** 지원과 관련이 없습니다. (어셈블리 언어를 자주 사용하지 않는 개발자에게는 별로 중요한 문제가 아닙니다.)

역시 저수준의 문제이기도 하지만 훨씬 흥미로운 기능 하나는, **Vista** 및 **Windows Server 2008**의 **WCT**(Wait Chain Traversal) 기능에 대한 디버거 지원입니다. 자세한 내용은 운영 체제 수준의 **WCT**가 설명된 **MSDN** 기술 문서와 아래 **URL**에서 **Delphi** 디버거의 새로운 기능에 대한 **CodeGear**의 **Chris Hesik** 블로그 포스트를 참조하십시오.

<http://msdn.microsoft.com/en-us/library/ms681622.aspx>
<http://blogs.codegear.com/chrisshesik/2008/07/21/34833>

스레드 상태 윈도우에는 데드락을 일으키는 데에 관련된 다양한 스레드에 대한 정보를 가진 컬럼이 추가되어 멀티 스레드 애플리케이션의 동작을 이해하는 데 도움을 줍니다.

디버깅과 새 언어 기능

디버거가 이전 버전과 비슷하게 보이긴 하지만, 사용자가 제네릭 및 익명 메소드를 사용하는 애플리케이션을 디버그할 수 있도록 많은 노력을 기울였습니다. 백그라운드에서 복잡한 코드 생성이 이루어지기 때문에 디버깅하는 코드는 원래 작성한 코드와 매우 다릅니다. 일부 한정된 문제점이 있긴 하지만 새로운 언어 기능의 디버깅은 일반적으로 잘 실행되며 이는 대단한 성과라고 할 수 있습니다.

프로젝트 관리를 DELPHI에 포함

최근의 버전들에서 Delphi IDE는 눈에 띄게 확장되었으며 Delphi 2009도 역시 마찬가지입니다. 전반적으로 가장 확장된 기능은 IDE의 향상된 안정성일 것입니다. 그 다음으로는, 계층적 빌드 컨피규레이션 지원, 한 프로젝트에서 다른 프로젝트로 옮길 수 있는 옵션 설정, 통합된 리소스 관리, 사용자 기호에 맞게 적용할 수 있는 여러 프로젝트 매니저 뷰 등으로 무장한 프로젝트 관리 기능의 확장일 것입니다. 개선된 컨피규레이션 관리로 대규모 프로젝트들을 관리하는 것이 Delphi 7 등의 이전 버전들보다 훨씬 쉬워졌습니다. 이런 새로운 기능들이 Delphi 개발자의 일상 업무에 있어 미치는 영향은 매우 클 것이라고 생각합니다. 따라서 Delphi 2009 IDE는 업그레이드할 만한 충분한 가치가 있습니다.

필자에 대하여

이 문서는 베스트 셀러 시리즈인 Mastering Delphi의 저작자 Marco Cantù가 Embarcadero Technologies 를 위해 작성하였습니다. 이 문서의 내용은 그의 최근 저서인 "Delphi 2009 핸드북" (<http://www.marcocantu.com/dh2009>)에서 발췌한 것입니다. Marco Cantù에 대한 정보는 그의 개인 블로그(<http://blog.marcocantu.com>)에서 읽을 수 있으며 전자 메일(marco.cantu@gmail.com)을 통해 연락할 수 있습니다.



Embarcadero Technologies Inc.는 애플리케이션 개발자 및 데이터베이스 전문가가 자신이 선택한 환경에서 소프트웨어 애플리케이션을 설계, 빌드 및 실행하는 도구를 사용할 수 있도록 합니다. 전 세계 3백만 이상의 커뮤니티와 Fortune지 선정 100대 기업 중 90개 기업이 Embarcadero의 CodeGear™ 및 DatabaseGear™ 제품군을 기반으로 하여 생산성을 향상시키고 개방적인 협업 및 자유로운 혁신을 추구하고 있습니다. Embarcadero는 1993년에 설립되어 캘리포니아 샌프란시스코에 본사가 있으며 전 세계에 사무소를 두고 있습니다. Embarcadero의 온라인 주소는 www.embarcadero.com입니다. Embarcadero의 주요 제품인 DatabaseGear의 도구에는 ER/Studio®, DBArtisan®, Rapid SQL® 및 Embarcadero® Change Manager™가 있습니다.



데브기어는 미국 Embarcadero Technologies Inc.와 기존의 코드기어 한국 지사의 협력으로 전략적으로 설립된 엠바카데로 솔루션 전문 공급 기업입니다. 데브기어는 Delphi, C++Builder, JBuilder, Delphi Prism 등 개발툴 제품들과 ER/Studio, PowerSQL, DB Artisan, EA/Studio 등의 데이터베이스 툴 제품들에 대한 한국 시장에 공급은 물론 기술지원 및 교육을 제공합니다. 데브기어 웹 사이트는 <http://www.devgear.co.kr/> 이며 제품에 대한 문의는 ask@embarcadero.kr 로 하면 됩니다.