

입문서



Borland®
Delphi™ 6
for Windows

볼랜드 코리아 주식회사
서울특별시 강남구 삼성동 159-1 ASEM 타워 30 층
연락처 : (02) 6001-3162
www.borlandkorea.co.kr



볼랜드와 볼랜드 코리아는 이 문서에 포함된 모든 내용에 대한 특허를 가지고 있습니다.
이 문서를 공급함에 있어 사용자에게 특허권에 대한 어떠한 라이선스도 주어지지 않습니다 .

COPYRIGHT © 1997, 2001 Borland Software Corporation. All rights reserved. All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries. Other product names are trademarks or registered trademarks of their respective holders.

제품 구입 문의: Tel:02-6001-3194, 02-6001-3191

Fax:02-6001-3199

볼랜드 코리아

서울시 강남구 삼성동 159-1 ASEM 타워 30 층

목 차

1장

서문	1-1
Delphi란 무엇인가?	1-1
정보 찾기	1-1
온라인 도움말	1-2
F1 도움말	1-2
인쇄된 설명서	1-3
개발자 지원 서비스와 웹 사이트	1-4
표기법	1-4

2장

데스크탑 둘러 보기	2-1
Delphi 시작	2-1
IDE	2-1
메뉴 및 툴바	2-3
컴포넌트 팔레트, 폼 디자이너 및 Object Inspector	2-4
Object TreeView	2-5
Object Repository	2-6
코드 에디터	2-7
Code Insight	2-7
Class Completion	2-8
Code Browsing	2-8
Diagram 페이지	2-9
폼 코드 보기	2-11
코드 탐색기	2-11
Project Manager	2-12
Project Browser	2-13
To-Do List	2-14

3장

Delphi 프로그래밍	3-1
프로젝트 생성	3-1
데이터 모듈 추가	3-2
사용자 인터페이스 구축	3-2
폼에 컴포넌트 놓기	3-2
컴포넌트 속성 설정	3-3
코드 작성	3-5
이벤트 핸들러 작성	3-5
VCL 및 CLX 라이브러리 사용	3-5
프로젝트 컴파일 및 디버깅	3-6
애플리케이션 배포	3-8
애플리케이션 국제화	3-8
프로젝트 타입	3-8
CLX 애플리케이션	3-9
웹 서버 애플리케이션	3-9
데이터베이스 애플리케이션	3-10

BDE Administrator	3-10
SQL Explorer(데이터베이스 탐색기)	3-11
Database Desktop	3-11
Data Dictionary	3-11
사용자 지정 컴포넌트	3-11
DLL	3-12
COM 및 ActiveX	3-12
타입 라이브러리	3-12

4장

텍스트 에디터(자습서) 만들기	4-1
새 애플리케이션 시작	4-1
속성 값 설정	4-2
폼에 컴포넌트 추가	4-3
메뉴와 툴바에 대한 지원 추가	4-6
Action Manager 컴포넌트에 Action 추가	4-7
Action Manager 컴포넌트에 표준 Action 추가	4-9
Image List 컴포넌트에 이미지 추가	4-10
메뉴 추가	4-12
툴바 추가	4-14
텍스트 영역 지우기(옵션)	4-15
이벤트 핸들러 작성	4-16
New 명령에 대한 이벤트 핸들러 만들기	4-16
Open 명령에 대한 이벤트 핸들러 만들기	4-18
Save 명령에 대한 이벤트 핸들러 만들기	4-19
Save As 명령에 대한 이벤트 핸들러 만들기	4-20
도움말 파일 생성	4-22
Help Contents 명령에 대한 이벤트 핸들러 만들기	4-22
Help Index 명령에 대한 이벤트 핸들러 만들기	4-23
About 상자 만들기	4-24
애플리케이션 완료	4-26

5장

데스크탑 사용자 지정	5-1
작업 영역 구성	5-1
메뉴와 툴바 배열	5-1
툴 윈도우 도킹	5-2
데스크탑 레이아웃 저장	5-4
컴포넌트 팔레트 사용자 지정	5-5

컴포넌트 팔레트 배열	5-5
컴포넌트 템플릿 만들기	5-6
컴포넌트 패키지 설치	5-7
프레임 사용	5-8
ActiveX 컨트롤 추가	5-9
프로젝트 옵션 설정	5-9
기본 프로젝트 옵션 설정	5-9
프로젝트와 폼 템플릿을 기본값으로 지정	5-9
Object Repository에 템플릿 추가	5-10
툴 환경 설정	5-11
폼 디자이너 사용자 지정	5-11
코드 에디터 사용자 지정	5-12
코드 탐색기 사용자 지정	5-12

색인

I-1

1

서문

입문서는 Delphi 개발 환경에 대한 개요를 제공하므로 사용자들이 본 제품을 즉시 사용할 수 있습니다. 아울러 Delphi에서 사용할 수 있는 툴과 기능에 대한 자세한 내용을 어디서 찾을 수 있는지 알려 줍니다.

2장 "데스크탑 둘러보기"에서는 Delphi 데스크탑의 주요 툴과 통합 데스크탑 환경(IDE)에 대해 설명합니다. 3장 "Delphi 프로그래밍"에서는 이러한 툴을 사용하여 애플리케이션을 만드는 방법에 대해 설명합니다. 4장 "텍스트 에디터(자습서) 작성"에서는 텍스트 에디터용 프로그램 작성 방법을 자습서를 통해 단계적으로 설명합니다. 5장 "데스크탑 사용자 지정"에서는 개발 목적에 따라 Delphi IDE를 사용자 지정할 수 있는 방법에 대해 설명합니다.

Delphi란 무엇인가?

Delphi는 신속한 애플리케이션 개발(RAD)을 위한 비주얼 객체 지향 프로그램입니다. Delphi를 사용하면 코딩을 최소화하면서 효율이 높은 Microsoft Windows 2000, Windows 98 및 Windows NT용 애플리케이션을 만들 수 있습니다. 또한 Delphi는 Kylix 및 Linux용 Borland RAD 툴과 함께 사용하는 경우 완전한 크로스 플랫폼 솔루션을 제공합니다. Delphi는 애플리케이션을 개발, 테스트, 배포하는 데 필요한 모든 툴 즉, 재사용할 수 있는 컴포넌트들로 이루어진 대형 라이브러리, 디자인 툴, 애플리케이션 및 폼 템플릿, 프로그래밍 마법사 등을 제공합니다.

정보 찾기

Delphi에 대한 정보는 이 장에 설명되어 있는 다음 내용에서 찾을 수 있습니다.

- 온라인 도움말
- 인쇄된 설명서
- Borland 개발자 지원 서비스 및 웹 사이트

이 릴리스의 새로운 기능에 대한 내용은 온라인 도움말 목차의 What's New 및 www.borland.com 웹 사이트를 참조하십시오.

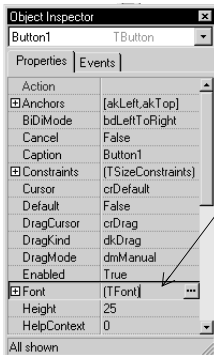
온라인 도움말

온라인 도움말 시스템은 사용자 인터페이스 기능, 랭귀지 구현, 프로그래밍 작업, VCL (Visual Component Library Reference) 및 Borland 크로스 플랫폼용 컴포넌트 라이브러리 (CLX) 의 컴포넌트에 대한 자세한 내용을 제공합니다. 온라인 도움말 시스템에는 Delphi 개발자 안내서, 오브젝트 파스칼 랭귀지 안내서 및 Delphi와 함께 제공되는 다른 기능에 대한 많은 도움말 파일 자료가 모두 들어 있습니다.

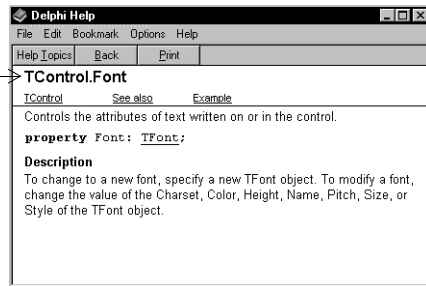
목차를 보려면 Help|Delphi Help 및 Help|Delphi Tools를 선택하고 Contents 탭을 클릭합니다. VCL 또는 CLX 객체 및 다른 항목을 조회하려면 Index 또는 Find 탭을 클릭하고 조회할 내용을 입력합니다.

F1 도움말

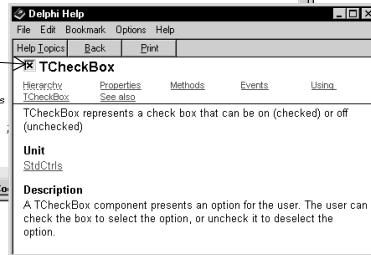
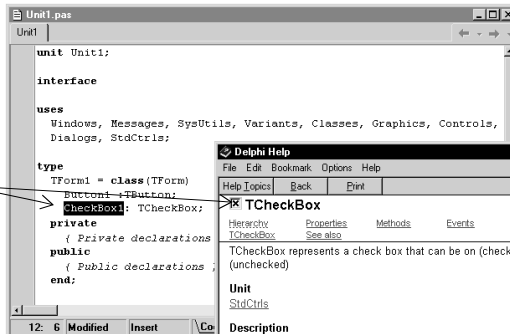
항목을 선택하고 F1을 누르면 메뉴 항목, 대화 상자, 툴바, 컴포넌트 등을 포함한 VCL, CLX 및 개발 환경의 모든 부분에서 문맥에 따른 도움말을 얻을 수 있습니다.



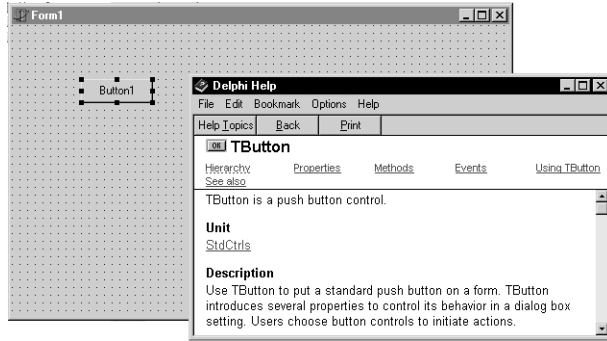
VCL Help를 표시하려면 Object Inspector의 속성 또는 이벤트 이름에서 F1을 누릅니다.



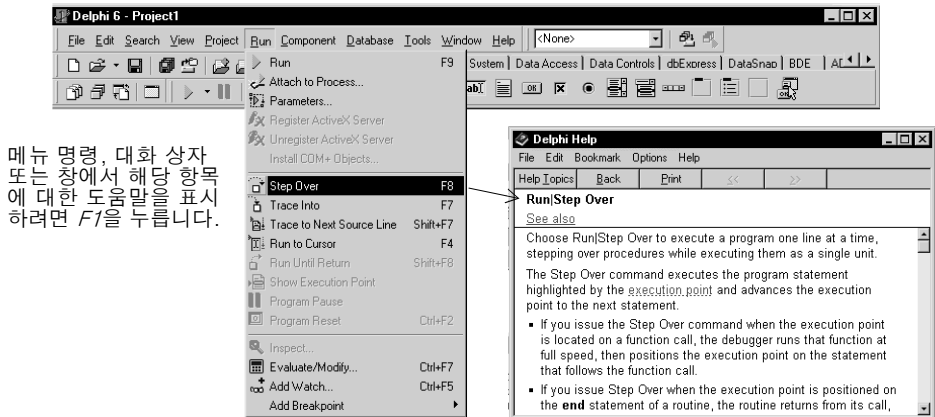
코드 에디터에서 랭귀지 키워드, VCL 또는 CLX 요소에서 F1을 누릅니다.



폼에 있는 컴포넌트에서 **F7**을 누릅니다.



또한 모든 대화 상자에서 Help 버튼을 누르면 문맥에 따른 온라인 설명서가 나타납니다.



메뉴 명령, 대화 상자 또는 창에서 해당 항목에 대한 도움말을 표시하려면 **F7**을 누릅니다.

컴파일러와 링커에서 발생한 오류 메시지는 코드 에디터 아래에 별도의 창으로 나타납니다. 컴파일 오류에 대한 도움말이 필요하면 목록에서 메시지를 선택하고 **F7**을 누릅니다.

인쇄된 설명서

이 입문서는 Delphi에 대한 소개입니다. 개발자 안내서와 같은 인쇄된 설명서를 추가로 주문하려면 shop.borland.com을 참조하십시오.

개발자 지원 서비스와 웹 사이트

Borland는 다양한 개발자 커뮤니티의 요구에 맞출 수 있는 다양한 지원 옵션을 제공합니다. 지원에 대해 알아보려면 <http://www.borland.com/devsupport/> 를 참조하십시오.

웹 사이트를 통해 Delphi 개발자들이 정보, 팁, 기술 등을 교환하는 여러 뉴스그룹에 액세스할 수 있습니다. 또한 이 사이트에는 Delphi 도서 목록, 그 외의 추가적인 Delphi 기술 문서, FAQ가 있습니다.

표기법

이 설명서에서는 특별한 텍스트를 표시하기 위하여 다음과 같은 글꼴을 사용합니다.

표 1.1 표기법

글꼴	의미
Monospace type	고정 폭 형식은 텍스트를 화면이나 코드에 표시되는 모양 그대로 나타냅니다. 사용자가 입력해야 하는 내용 역시 이 글꼴로 나타냅니다.
Boldface	텍스트 또는 코드 목록에서 굵게 쓰여진 글자는 예약어 또는 컴파일러 옵션을 나타냅니다.
<i>Italics</i>	텍스트에서 이탤릭체 단어는 변수 또는 타입 이름과 같은 Delphi 식별자를 나타냅니다. 이탤릭체는 새로운 용어 같은 일부 글자를 강조하기 위해 사용하기도 합니다.
<i>Keycaps</i>	이 글꼴은 키보드에 있는 특정 키를 나타냅니다. 예를 들면, 다음과 같습니다. "메뉴를 종료하려면 <i>Esc</i> 를 누릅니다."

2

데스크탑 둘러 보기

이 장에서는 Delphi를 시작하는 방법에 대해 설명하고 데스크탑 또는 통합 데스크탑 환경(IDE)의 주요 부분과 툴에 대한 빠른 둘러 보기를 제공합니다.

Delphi 시작

다음과 같은 방법으로 Delphi를 시작할 수 있습니다.

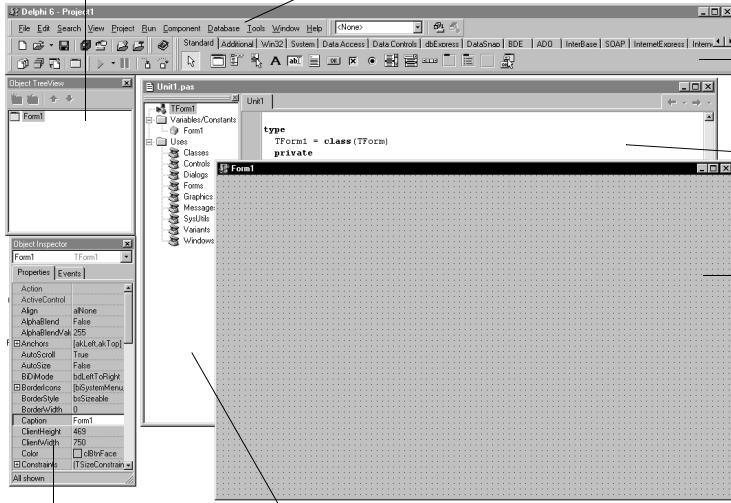
- 바로 가기를 만든 경우 Delphi 아이콘을 더블 클릭합니다.
- Windows 시작 메뉴에서 프로그램 | Borland Delphi 6 | Delphi 6을 선택합니다.
- Windows 시작 메뉴에서 [실행]을 선택한 다음 Delphi32를 입력합니다.
- Delphi\Bin 디렉토리의 Delphi32.exe를 더블 클릭합니다.

IDE

Delphi를 처음 시작하면 IDE에 있는 주요 툴 몇 가지를 볼 수 있습니다. Delphi의 IDE에는 메뉴, 툴바, 컴포넌트 팔레트, Object Inspector, Object TreeView, 코드 에디터, 코드 탐색기, Project Manager 및 기타 다른 도구들이 많이 들어 있습니다. 사용할 수 있는 기능과 컴포넌트는 구입한 Delphi 에디션에 따라 다릅니다.

Object TreeView는 컴포넌트의 부모 자식 관계에 대한 계층적 뷰를 표시합니다.

메뉴와 툴바에서 다양한 기능과 툴에 액세스하여 애플리케이션을 작성할 수 있습니다.



컴포넌트 팔레트는 프로젝트에 추가할 수 있는 미리 만들어진 컴포넌트를 포함합니다.

코드 에디터는 보거나 편집할 코드를 표시합니다.

폼 디자이너는 애플리케이션용 사용자 인터페이스의 디자인을 시작할 빈 폼을 포함합니다. 애플리케이션에는 많은 폼을 포함할 수 있습니다.

Object Inspector는 객체 속성을 변경하고 이벤트 핸들러를 선택하는데 사용됩니다.

코드 탐색기는 유닛에 있는 클래스, 변수 및 루틴을 보여 주고 신속하게 탐색할 수 있게 해줍니다.

Delphi의 개발 모델은 *two-way* 툴을 기반으로 합니다. 이것은 비주얼 디자인 툴과 텍스트 기반의 코드 편집 사이에서 자유롭게 이동할 수 있다는 것을 의미합니다. 예를 들어, 폼 디자이너를 사용하여 그래픽 인터페이스에 있는 버튼 및 기타 요소를 정렬한 다음 즉시 폼에 대한 텍스트 설명이 포함된 폼 파일을 볼 수 있습니다. 또한 비주얼 프로그래밍 환경에서 Delphi에 의해 생성된 모든 코드를 수동으로 편집할 수 있습니다.

IDE에서는 모든 프로그래밍 툴을 쉽게 찾고 사용할 수 있습니다. IDE를 종료하지 않고 그래픽 인터페이스 디자인, 클래스 라이브러리를 통해 찾기, 코드 작성, 프로젝트 컴파일, 테스트, 디버깅 및 관리 등을 수행할 수 있습니다.

IDE의 구성에 대해 알아보려면 5장 "데스크탑 사용자 지정"을 참조하십시오.

메뉴 및 툴바

화면의 상단을 차지하는 메인 윈도우에는 메뉴, 툴바 및 컴포넌트 팔레트가 있습니다.



기본 정렬된 메인 윈도우입니다.

Delphi의 툴바를 사용하면 자주 사용하는 작업과 명령에 빠르게 액세스할 수 있습니다. 대부분의 툴바 작업은 드롭다운 메뉴에도 있습니다.

표준 툴바

보기 툴바

데스크탑 툴바

디버그 툴바

인터넷 툴바

버튼이 수행하는 작업을 알아 보려면 도구 팁이 나타날 때까지 버튼을 가리키십시오.

마우스 오른쪽 버튼 메뉴를 사용하여 툴바를 숨길 수 있습니다. 툴바가 보이지 않을 경우 표시하려면 View | Toolbars를 선택하고 원하는 툴바를 선택하십시오.

많은 작업에서 툴바 버튼뿐만 아니라 키보드 단축키를 사용합니다. 사용할 수 있는 키보드 단축키는 항상 드롭다운 메뉴에 있는 명령 옆에 나타납니다.

많은 도구와 아이콘을 마우스 오른쪽 버튼으로 클릭하여 작업 중인 객체에 적당한 명령 메뉴를 표시할 수 있습니다. 이러한 메뉴를 *컨텍스트 메뉴*라고 합니다.

툴바를 사용자 지정할 수도 있습니다. 툴바에 원하는 명령을 추가하거나 다른 위치로 툴바를 이동할 수 있습니다. 자세한 내용은 5-1 페이지의 "메뉴와 툴바 배열" 및 5-4 페이지에 있는 "데스크탑 레이아웃 저장"을 참조하십시오.

자세한 내용은...

메뉴 옵션에 대한 도움말이 필요하면 해당 메뉴 옵션을 가리킨 후 *F1* 키를 누르십시오.

컴포넌트 팔레트, 폼 디자이너 및 Object Inspector

컴포넌트 팔레트, 폼 디자이너, Object Inspector와 Object TreeView를 함께 사용하여 애플리케이션에 사용자 인터페이스를 빌드할 수 있습니다.

컴포넌트 팔레트에는 탭 모양의 페이지가 있는데 여기에는 비주얼 또는 논비주얼 (nonvisual) VCL 및 CLX 컴포넌트를 나타내는 아이콘이 여러 개 들어 있습니다. 각 페이지는 다양한 기능 그룹으로 컴포넌트가 나뉘어져 있습니다. 예를 들어 Standard, Additional 및 Win32 페이지에는 편집 상자와 위/아래 버튼 등의 윈도우 컨트롤이 들어 있으며 Dialogs 페이지에는 파일 열기 및 파일 저장과 같은 파일 작업에 사용하기 위한 일반 대화 상자가 들어 있습니다.

기능별로 그룹화된 컴포넌트 팔레트 페이지

페이지를 더 보려면 클릭합니다.

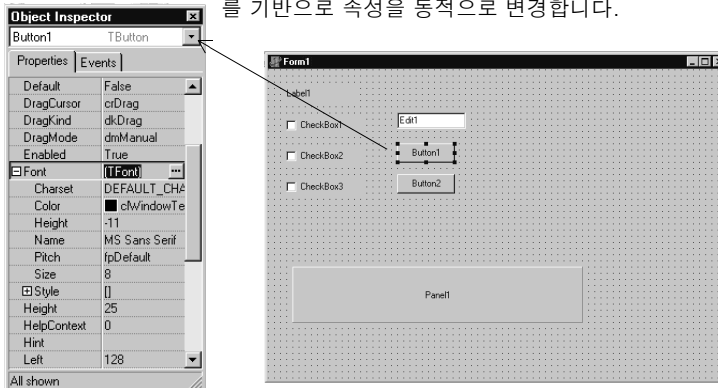


컴포넌트

각 컴포넌트에는 애플리케이션을 제어할 수 있는 속성, 이벤트, 메소드 등의 특정 속성이 있습니다.

폼이나 폼 디자이너에 컴포넌트를 놓은 다음 사용자 인터페이스에서 보여야 하는 형태로 컴포넌트를 정렬할 수 있습니다. 폼에 가져다 놓은 컴포넌트에 대해서는 *Object Inspector*를 사용하여 디자인 타임 속성을 설정하고 이벤트 핸들러를 만들며 보이는 속성 및 이벤트를 필터링하여 애플리케이션의 시각적인 모습과 애플리케이션을 실행시키는 코드를 연결할 수 있습니다. 3-2 페이지의 "폼에 컴포넌트 놓기"를 참조하십시오.

컴포넌트를 폼에 둔 다음, Object Inspector에서 선택한 컴포넌트를 기반으로 속성을 동적으로 변경합니다.



자세한 내용은...

온라인 도움말 색인의 "Component palette"를 참조하십시오.

Object TreeView

Object TreeView는 컴포넌트의 형제 및 부모 자식 관계를 계층 또는 트리 다이어그램으로 표시합니다. 트리 다이어그램은 Object Inspector와 폼 디자이너에서 동기화되므로 Object TreeView에서 포커스를 변경하면 Object Inspector와 폼의 포커스가 모두 변경됩니다.

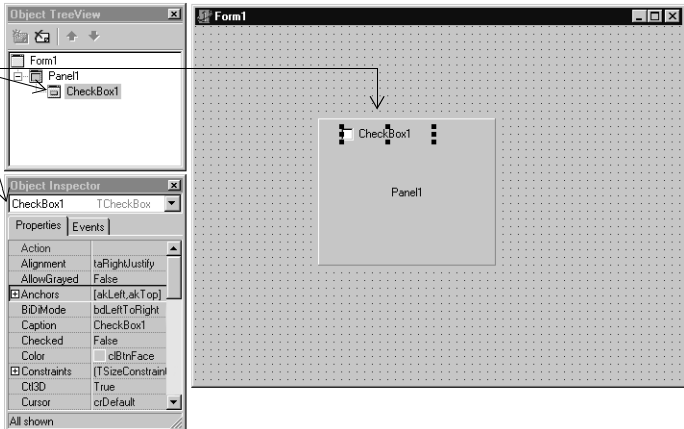
Object TreeView를 사용하여 서로 연관된 컴포넌트 관계를 변경할 수 있습니다. 예를 들어, 폼에 패널과 체크 박스 컴포넌트를 추가하면 두 컴포넌트는 형제가 됩니다. 그러나 Object TreeView에서 패널 아이콘 상단에 체크 박스를 끌어다 놓으면 그 체크 박스는 패널의 자식이 됩니다.

객체 속성이 완료되지 않은 경우 Object TreeView는 객체 속성 옆에 빨간색의 물음표를 표시합니다. 트리 다이어그램의 어떤 객체든지 더블 클릭하면 이벤트 핸들러를 작성할 수 있는 곳에 코드 에디터를 열 수 있습니다.

Object TreeView가 보이지 않는 경우 View|Object TreeView를 선택합니다.

Object TreeView, Object Inspector와 폼 디자이너는 함께 작동됩니다. 폼에서 객체를 클릭하면 Object TreeView와 Object Inspector 모두 포커스가 자동으로 변경되며 그 반대의 경우도 마찬가지입니다.

*Alt-Shift-F11*을 눌러 Object TreeView에 포커스를 맞춥니다.



Object TreeView는 데이터베이스 객체 간의 관계를 표시하는 데 특히 유용합니다.

자세한 내용은...

온라인 도움말 색인의 "Object TreeView"를 참조하십시오.

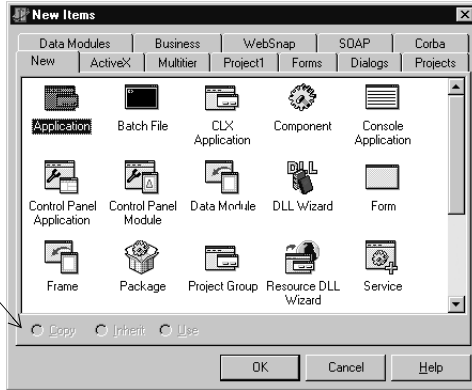
Object Repository

Object Repository에는 개발을 쉽게 하기 위한 폼, 대화 상자, 데이터 모듈, 마법사, DLL, 예제 애플리케이션 및 그 밖의 다른 항목 등이 있습니다. 프로젝트를 시작할 때 New Items 대화 상자를 표시하려면 File|New|Other를 선택합니다. New Items 대화 상자는 Object Repository와 같습니다. Repository에 만들려는 것과 유사한 객체가 있는지 확인하려면 Repository를 선택하십시오.

Repository의 탭 모양 페이지에는 폼, 프레임, 유닛 등과 같은 객체와 특수 항목을 만드는 마법사가 있습니다.

Object Repository에 들어 있는 항목을 기반으로 하여 새로운 항목을 만들 경우에는 항목을 복사, 상속 또는 사용할 수 있습니다.

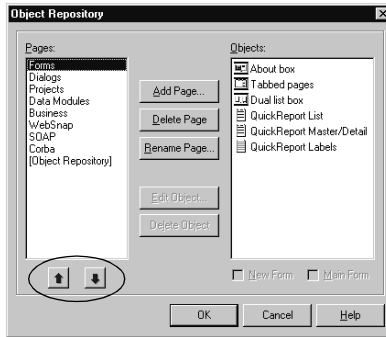
Copy(기본값)는 프로젝트에서 항목의 복사본을 만듭니다. *Inherit*는 Repository의 객체에 대한 변경 내용이 프로젝트의 객체에 의해 상속된다는 것을 의미합니다. *Use*는 프로젝트의 객체에 대한 변경 내용이 Repository의 객체에 의해 상속된다는 것을 의미합니다.



Object Repository에서 객체를 편집하거나 제거하려면 Tools|Repository를 선택하거나 New Items 대화 상자에서 마우스 오른쪽 버튼을 클릭하고 Properties를 선택합니다.

Object Repository에서 탭 모양의 페이지를 추가, 제거 또는 이름을 다시 지정할 수 있습니다.

New Items 대화 상자에서 탭 모양의 페이지가 나타나는 순서를 변경하려면 화살표를 클릭합니다.



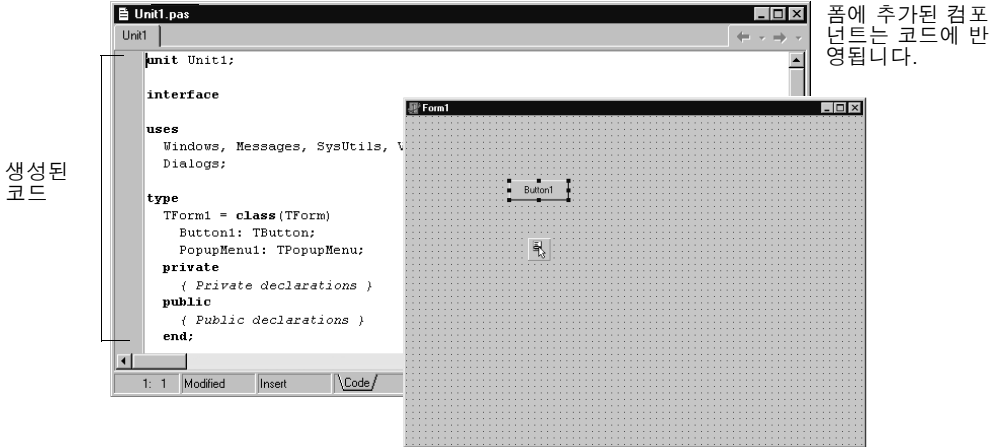
Object Repository에 프로젝트와 폼 템플릿을 추가하려면 5-9 페이지의 "Object Repository에 템플릿 추가"를 참조하십시오.

자세한 내용은...

온라인 도움말 색인의 "Object Repository"를 참조하십시오. 사용할 수 있는 객체는 구입한 Delphi 에디션에 따라 다릅니다.

코드 에디터

애플리케이션용 사용자 인터페이스를 디자인할 때 Delphi는 원본으로 사용하는 오브젝트 파스칼 코드를 생성합니다. 폼과 객체의 속성을 선택하고 수정하면 변경 내용이 소스 파일에 자동으로 반영됩니다. 모든 기능을 갖춘 아스키 에디터인 기본 제공 코드 에디터를 직접 사용하면 소스 파일에 코드를 추가할 수 있습니다.



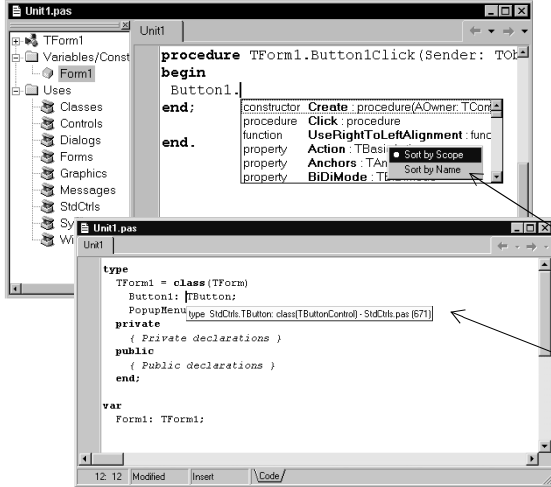
Delphi는 Code Insight 툴, class completion 및 code browsing 등 코드 작성에 도움을 주는 다양한 도움 기능을 제공합니다.

Code Insight

Code Insight 툴은 문맥에 맞는 팝업 창을 표시합니다.

표 2.1 Code Insight 툴

툴	작동 방법
Code Completion	클래스 이름 뒤에 점(.)을 찍고 잠시 기다리면 클래스에 적합한 속성, 메소드 및 이벤트 목록이 나타나고 목록에서 항목을 선택한 후 <i>Enter</i> 를 누릅니다. 코드의 interface 섹션에서는 두 개 이상의 항목을 선택할 수 있습니다. 변수에 유효한 값들의 목록을 표시하려면 할당 문의 시작 부분을 입력하고 <i>Ctrl+space</i> 를 누릅니다. 인수 목록을 나타내려면 프로시저, 함수, 메소드 이름을 입력합니다.
Code Parameters	메소드 인수에 대한 구문을 표시하려면 메소드 이름과 여는 괄호를 입력합니다.
Tooltip Expression Evaluation	디버깅 중 프로그램 실행이 멈춘 상태에서 변수에 마우스를 갖다 대면 그 변수의 현재 값을 보여 줍니다.
Tooltip Symbol Insight	코드를 편집할 때 식별자에 마우스를 갖다 대면 타입 선언을 보여 줍니다.
Code Templates	코드에 삽입할 수 있는 일반 프로그래밍 문의 목록을 보려면 <i>Ctrl+J</i> 를 누릅니다. Delphi에서 기본 제공되는 것 외에도 사용자가 직접 템플릿을 만들 수 있습니다.



Code Completion은 Button1에 점(.)을 입력하며 Delphi는 클래스의 속성, 메소드 및 이벤트 목록을 표시합니다. 입력하면 목록에서 해당 클래스에 속하는 선택 영역을 자동으로 필터링합니다. 목록에서 항목을 선택하고 Enter 키를 눌러 코드에 항목을 추가합니다.

프로시저와 속성은 청록색으로 나타나고 함수는 파란색으로 나타납니다. 마우스 오른쪽 버튼을 클릭한 후 Sort by Name을 선택하면 이 목록을 알파벳 순으로 정렬할 수 있습니다.

식별자에 마우스를 갖다 대면 Tooltip Symbol Insight는 해당 식별자에 대한 선언 정보를 표시합니다.

이러한 툴을 선택 또는 선택 해제하려면 Tools | Editor Options를 선택한 후 Code Insight 탭을 클릭합니다. Automatic features 섹션에서 툴을 선택 또는 선택 해제합니다.

Class Completion

Class Completion은 클래스에 대한 뼈대 코드를 생성합니다. 커서를 유닛에 있는 **interface** 섹션의 클래스 선언 내 아무 위치에 놓은 다음 **Ctrl+Shift+C**를 누르거나 마우스 오른쪽 버튼을 클릭한 후 Complete Class at Cursor를 선택합니다. Delphi는 private **read** 지정자와 **write** 지정자를 필요로 하는 모든 속성 선언에 이 지정자들을 추가한 다음 모든 클래스 메소드에 대한 뼈대 코드를 생성합니다. Class Completion을 사용하면 이미 구현한 메소드에 대한 클래스를 선언할 수도 있습니다.

Class Completion을 선택하려면 Tools | Environment Options를 선택하고 Explorer 탭을 클릭하여 Finish incomplete 속성이 선택되어 있는지 확인합니다.

자세한 내용은...

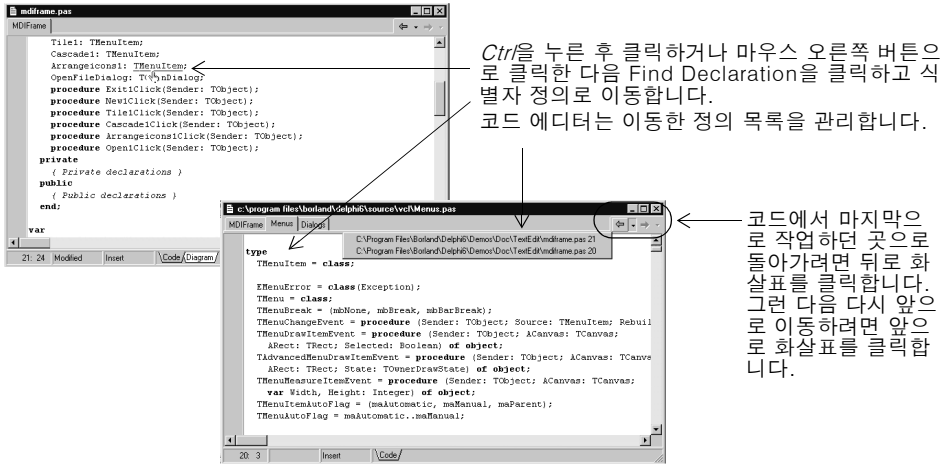
온라인 도움말 색인의 "Code Insight" 및 "class completion"을 참조하십시오.

Code Browsing

클래스, 변수, 속성, 메소드 또는 다른 식별자에 마우스를 갖다 대면 Tooltip Symbol Insight라는 팝업 메뉴가 식별자를 선언한 위치를 나타냅니다. **Ctrl**을 누르면 커서가 손 모양으로 바뀌고 식별자는 파란색으로 변하고 밑줄이 그어집니다. 이것을 클릭하면 식별자 정의로 이동합니다.

코드 에디터에는 웹 브라우저처럼 앞으로 버튼과 뒤로 버튼이 있습니다. 이러한 정의로 이동하는 동안 코드 에디터는 코드에서 이동한 위치를 추적합니다. 앞으로 버튼과 뒤로

버튼 옆에 있는 드롭다운 화살표를 클릭하면 이러한 참조 기록에서 앞뒤로 이동할 수 있습니다.



Ctrl+Shift+↑ 또는 Ctrl+Shift+↓을 누르면 프로시저의 선언과 프로시저 구현 사이에서 이동할 수도 있습니다.

코드 편집 환경을 사용자 지정하려면 5-12 페이지의 "코드 에디터 사용자 지정"을 참조하십시오.

자세한 내용은...

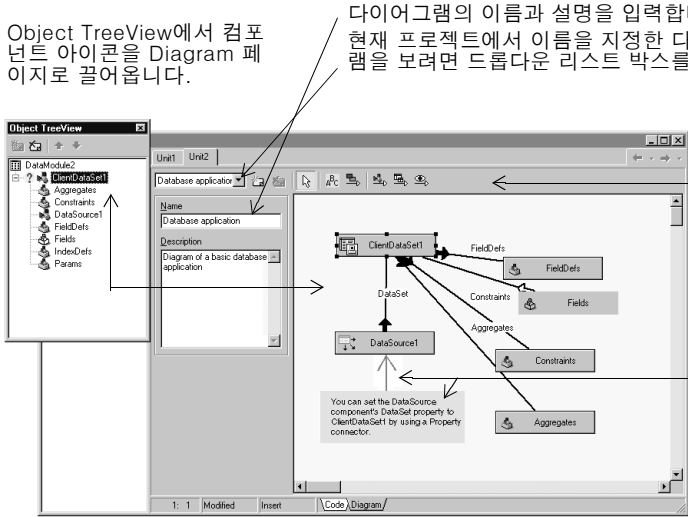
온라인 도움말 색인의 "Code editor"를 참조하십시오.

Diagram 페이지

코드 에디터의 하단에는 한 개 이상의 탭이 있습니다. 모든 코드를 작성하는 Code 페이지는 기본적으로 전경에 나타납니다. 사용자의 Delphi 에디션에 따라 Diagram 페이지에는 폼 또는 데이터 모듈에 놓은 컴포넌트 간의 관계를 나타내는 아이콘과 연결선이 표시됩니다. 이러한 관계에는 형제 간의 관계, 부모-자식 관계, 컴포넌트-속성 관계 등이 있습니다.

다이어그램을 만들려면 Diagram 페이지를 클릭합니다. Object TreeView에서 Diagram 페이지로 한 개 또는 여러 개의 아이콘을 끌어온 다음 세로 방향으로 배열합니다. 아이콘을 가로 방향으로 배열하려면 아이콘을 끌면서 Shift를 누릅니다. 부모 자식 또는 컴포넌트 속성 종속 관계를 갖는 아이콘을 페이지로 끌어오면 종속 관계를 표시하는 선 또는 연결선이 자동으로 추가됩니다. 예를 들어, 데이터셋 컴포넌트를 데이터 모듈에 추가하고 데이터셋 아이콘과 데이터셋의 속성 아이콘을 Diagram 페이지로 끌어오면 데이터셋의 속성 아이콘들이 데이터셋 아이콘에 자동으로 연결됩니다.

컴포넌트에 종속 관계가 없지만 종속 관계를 보여 주고자 하면 Diagram 페이지 상단의 툴바 버튼을 사용하여 Allude, Property, Master/Detail 및 Lookup 등의 네 가지 연결선 타입 중 하나를 추가합니다. 주석 블록을 추가하여 블록을 서로 연결하거나 관련된 아이콘에 연결할 수도 있습니다.



Object TreeView에서 컴포넌트 아이콘을 Diagram 페이지로 끌어옵니다.

다이어그램의 이름과 설명을 입력합니다. 현재 프로젝트에서 이름을 지정한 다른 다이어그램을 보려면 드롭다운 리스트 박스를 클릭합니다.

Diagram 페이지 툴바 버튼(Property, Master/Detail 및 Lookup)을 사용하여 컴포넌트 간에 또는 컴포넌트와 그 속성 사이의 관계를 지정합니다. 연결선의 모양은 관계의 유형에 따라 다릅니다. Comment block 버튼을 클릭하여 주석 블록을 추가하고 Allude connector 버튼을 클릭하여 다른 주석 블록이나 아이콘에 연결선을 그립니다.

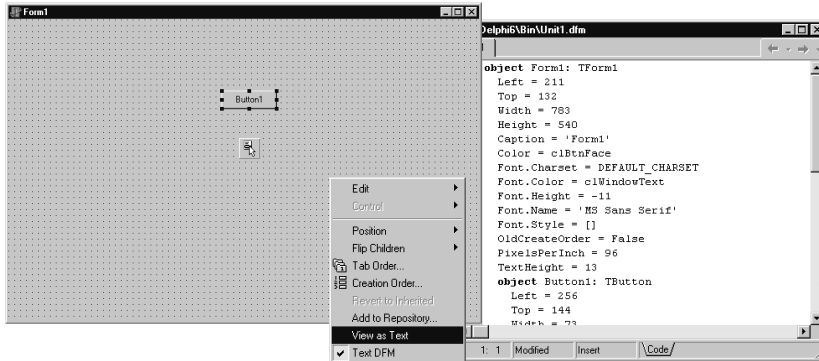
다이어그램의 이름과 설명을 입력하고 다이어그램을 저장하고 난 다음 다이어그램을 인쇄할 수 있습니다.

자세한 내용은...

온라인 도움말 색인에서 "Diagram page"를 참조하십시오.

폼 코드 보기

폼은 대부분의 Delphi 프로젝트에서 매우 가지적인 부분으로서 여기서 애플리케이션의 사용자 인터페이스를 디자인합니다. 보통 Delphi의 비주얼한 도구를 사용하여 폼을 디자인한 다음 폼 파일로 저장합니다. 폼 파일(.dfm 또는 CLX 애플리케이션용 .xfm)은 모든 지속적인 속성의 값들을 비롯하여 폼의 각 컴포넌트에 대해 기술한 것입니다. 코드 에디터에서 폼 파일을 보고 편집하려면 폼을 마우스 오른쪽 버튼으로 클릭하고 View as Text를 선택합니다. 폼의 그래픽 보기로 돌아가려면 마우스 오른쪽 버튼을 클릭하고 View as Form을 선택합니다.



View As Text를 사용하면 코드 에디터에서 폼의 속성에 대한 텍스트 설명을 볼 수 있습니다.

텍스트(기본) 형식이나 마이너리 형식으로 폼 파일을 저장할 수 있습니다. Tools | Environment Options를 선택하고 Designer 페이지를 선택한 후 New forms as text 체크 박스를 선택 또는 선택 해제하여 새로 만든 폼에 사용할 형식을 지정합니다.

자세한 내용은...

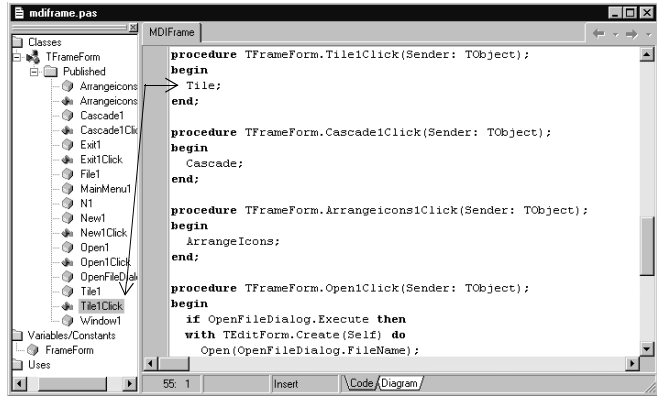
온라인 도움말 색인의 "form files"를 참조하십시오.

코드 탐색기

Delphi를 열면 사용자의 Delphi 에디션에서 코드 탐색기의 사용 가능 여부에 따라 코드 탐색기가 코드 에디터 창의 왼쪽에 도킹됩니다. 코드 탐색기에는 유닛에 정의된 타입, 클래스, 속성, 메소드, 전역 변수 및 루틴 등을 보여 주는 트리 다이어그램이 있습니다. 또한 **uses** 절에 나열된 다른 유닛들도 보여 줍니다.

코드 탐색기를 사용하여 코드 에디터를 탐색할 수 있습니다. 예를 들면, 코드 탐색기의 메소드를 더블 클릭하면 커서가 코드 에디터 내의 유닛에서 인터페이스 부분의 클래스 선언에 있는 정의로 이동합니다.

코드 탐색기에서 항목을 더블 클릭하면 코드 에디터의 해당 항목이 구현된 부분으로 커서가 이동합니다. *Ctrl+Shift+E*를 눌러 코드 탐색기와 코드 에디터에 있었던 마지막 위치에서 앞으로 뒤로 커서를 이동합니다. 코드 탐색기의 각 항목에는 그 타입을 지정하는 아이콘이 있습니다.



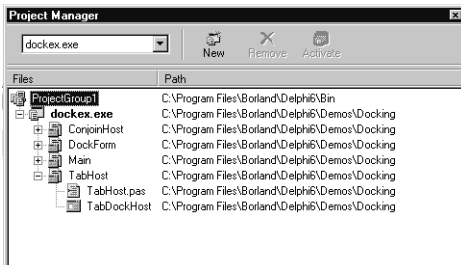
코드 탐색기에서 그 콘텐츠를 표시하는 방법을 구성하려면 Tools|Environment Options를 선택하고 Explorer 탭을 클릭합니다. 5-12 페이지의 "코드 탐색기 사용자 지정"을 참조하십시오.

자세한 내용은...

온라인 도움말 색인의 "Code Explorer"를 참조하십시오.

Project Manager

Delphi를 처음 시작하면 2-2 페이지에 표시된 것처럼 새 프로젝트가 자동으로 열립니다. 프로젝트에는 개발하려는 애플리케이션 또는 DLL을 구성하는 파일이 여러 개 들어 있습니다. Project Manager라는 프로젝트 관리 도구에서 이러한 파일, 즉 폼, 유닛, 리소스, 객체, 라이브러리 파일을 보면서 구성할 수 있습니다. Project Manager를 표시하려면 View|Project Manager를 선택합니다.



Project Manager를 사용하면 관련 프로젝트에 대한 정보를 단일 프로젝트 그룹으로 결합하고 표시할 수 있습니다. 여러 실행 파일과 같은 관련 프로젝트를 하나의 그룹으로 구성하여 동시에 컴파일할 수 있습니다. 프로젝트 컴파일과 같은 프로젝트 옵션을 변경하려면 5-9 페이지의 "프로젝트 옵션 설정"을 참조하십시오.

자세한 내용은...

온라인 도움말 색인의 "Project Manager"를 참조하십시오.

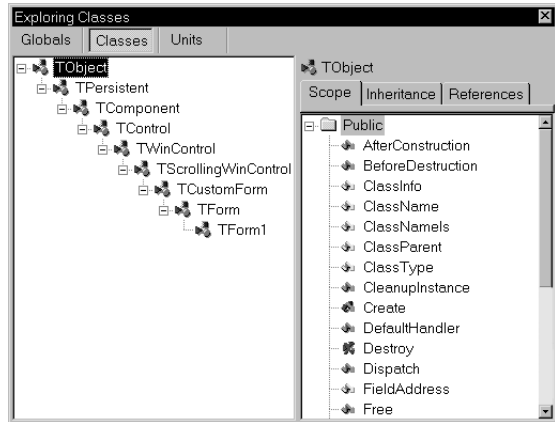
Project Browser

Project Browser는 프로젝트를 자세히 검사합니다. Browser는 프로젝트가 선언하거나 사용하는 클래스, 유닛 및 전역 기호(타입, 속성, 메소드, 변수, 루틴)를 트리 형태로 표시합니다. View|Browser를 선택하여 Project Browser를 표시합니다.

Project Browser에는 Inspector 창(왼쪽)과 Details 창이라는 크기를 조정할 수 있는 두 개의 창이 있습니다. Inspector 창에는 전역, 클래스, 유닛에 대한 세 가지 탭이 있습니다.

전역은 클래스, 타입, 속성, 메소드, 변수, 루틴 등을 표시합니다. 클래스는 계층 다이어그램에서 클래스를 표시합니다.

유닛은 각 유닛에서 선언된 유닛과 식별자, 사용하고 있거나 각 유닛에서 사용한 다른 유닛 등을 표시합니다.



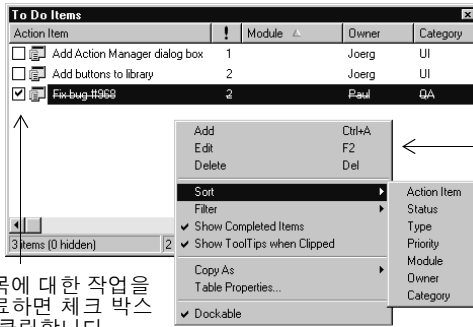
기본적으로 Project Browser는 현재 프로젝트에서만 사용되는 유닛의 기호를 표시합니다. 범위를 변경하면 Delphi에서 사용할 수 있는 모든 기호를 표시할 수 있습니다. Tools|Environment Options를 선택하고 Explorer 페이지에서 All symbols (VCL included)를 선택합니다.

자세한 내용은...

온라인 도움말 색인의 "Project Browser"를 참조하십시오.

To-Do List

To-do list는 프로젝트를 완료하기 위해 필요한 항목을 기록합니다. 목록에 프로젝트 범용 항목을 직접 추가하거나 소스 코드에서 특정 항목을 직접 추가할 수 있습니다. 프로젝트와 연관된 정보를 추가하거나 보려면 View | To-Do List를 선택합니다.



목록을 정렬하고 필터링할 수 있는 명령을 표시하려면 To-do list를 마우스 오른쪽 버튼으로 클릭합니다.

항목에 대한 작업을 완료하면 체크 박스를 클릭합니다.

자세한 내용은...

온라인 도움말 색인의 "To-do lists"를 참조하십시오.

3

Delphi 프로그래밍

이 장에서는 프로젝트 생성, 폼 작업, 코드 작성, 컴파일, 디버깅, 배포, 애플리케이션 국제화 및 개발 가능한 프로젝트 타입 등을 비롯한 Delphi를 사용한 소프트웨어 개발에 대한 개요를 다룹니다.

프로젝트 생성

프로젝트는 디자인 타임 시 만들어지거나 프로젝트 소스 코드를 컴파일할 때 생성된 파일 모음입니다. Delphi를 처음 시작하면 새 프로젝트가 열립니다. 여러 파일 중에서 프로젝트 파일(Project1.dpr), 유닛 파일(Unit1.pas), 리소스 파일(Unit1.dfm 또는 CLX 애플리케이션용 Unit1.xfm)은 자동적으로 생성됩니다.

프로젝트가 이미 열려 있는 상태에서 새로운 프로젝트를 열려면 File|New|Application 또는 File|New|Other를 선택하고 Application 아이콘을 더블 클릭합니다. File|New|Other를 클릭하면 대화 상자와 같이 미리 디자인된 템플릿뿐만 아니라 추가적인 폼, 프레임, 모듈을 제공하는 Object Repository가 열리고 이를 프로젝트에 추가할 수 있습니다. Object Repository에 대한 자세한 내용은 2-6 페이지의 "Object Repository"를 참조하십시오.

프로젝트를 시작할 때에는 애플리케이션 또는 DLL과 같은 개발하고자 하는 대상에 대해 알아야 합니다. Delphi를 사용하여 개발할 수 있는 프로젝트 타입에 대해 알아보려면 3-8 페이지의 "프로젝트 타입"을 참조하십시오.

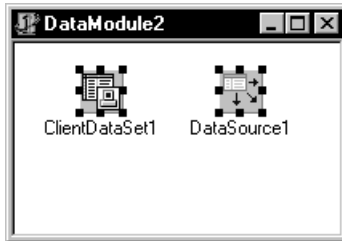
자세한 내용은...

온라인 도움말 색인의 "projects"를 참조하십시오.

데이터 모듈 추가

데이터 모듈은 논비주얼(nonvisual) 컴포넌트만 포함하는 폼 타입입니다. 일반적인 폼에서는 논비주얼(nonvisual) 컴포넌트와 비주얼 컴포넌트를 함께 놓을 수 있습니다. 데이터베이스와 시스템 객체 그룹을 재사용할 계획이 있거나 데이터베이스 연결과 비즈니스 룰을 처리하는 애플리케이션 부분을 분리하려는 경우라면 데이터 모듈을 간편한 조직 도구로 사용할 수 있습니다.

데이터 모듈을 만들려면 File|New|Data Module을 선택합니다. Delphi는 코드 에디터에서 모듈에 대한 추가 유닛 파일을 표시하는 빈 데이터 모듈을 열고 그 모듈을 현재 프로젝트에 새 유닛으로 추가합니다. 폼에서와 동일한 방식으로 논비주얼(nonvisual) 컴포넌트를 데이터 모듈에 추가합니다.



컴포넌트 팔레트에서 논비주얼 컴포넌트를 더블 클릭해서 데이터 모듈에 놓습니다.

기존 데이터 모듈을 다시 열면 Delphi는 추가된 컴포넌트를 표시합니다.

자세한 내용은...

온라인 도움말 색인의 "data modules"를 참조하십시오.

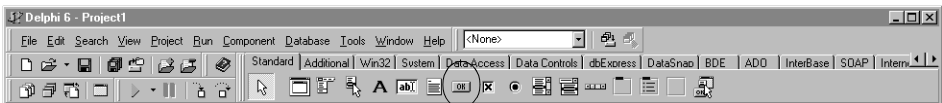
사용자 인터페이스 구축

Delphi를 사용하여 먼저 컴포넌트 팔레트에서 컴포넌트를 선택한 다음 메인 폼 위에 놓아 사용자 인터페이스(UI)를 만듭니다.

폼에 컴포넌트 놓기

폼에 컴포넌트를 두려면 다음 중 하나를 수행합니다.

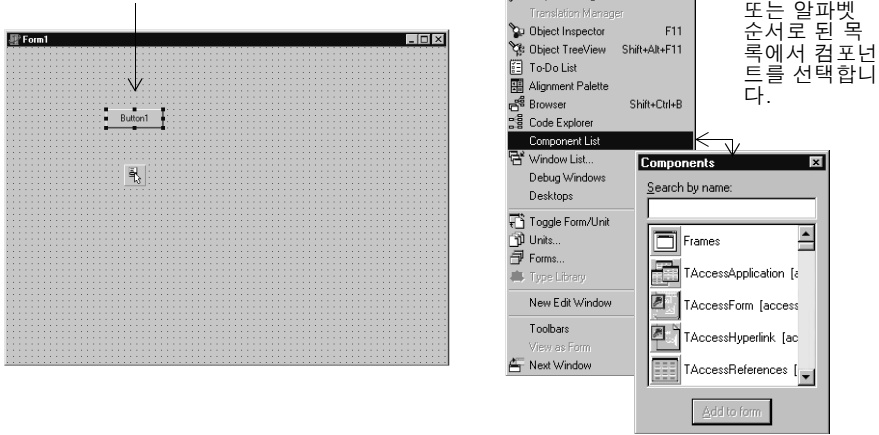
- 1 컴포넌트를 더블 클릭합니다.
- 2 컴포넌트를 한 번 클릭한 다음 컴포넌트를 표시할 폼을 클릭합니다.



컴포넌트 팔레트에서 컴포넌트를 클릭합니다.

컴포넌트를 선택한 폼의 원하는 곳으로 끌어 놓습니다.

그런 다음 폼에서 컴포넌트를 두려는 곳을 클릭합니다.

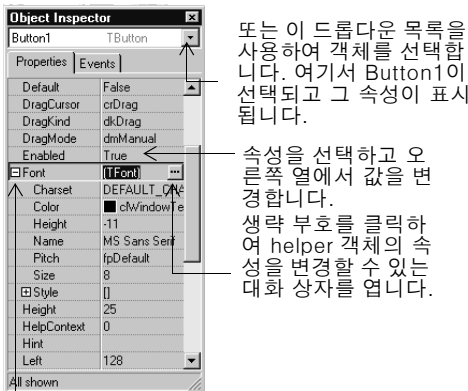


자세한 내용은...

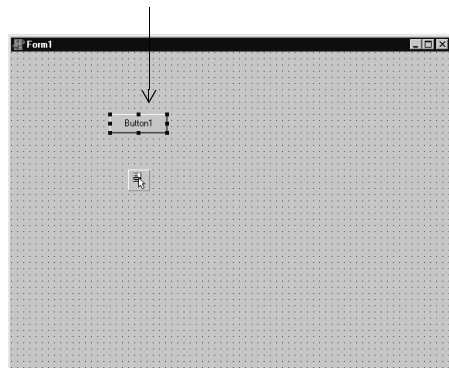
온라인 도움말 색인의 "Component palette"를 참조하십시오.

컴포넌트 속성 설정

컴포넌트를 폼에 둔 다음 컴포넌트의 속성을 설정하고 이벤트 핸들러를 코딩합니다. 컴포넌트 속성을 설정하면 애플리케이션에서 컴포넌트가 나타나고 동작하는 방식이 바뀝니다. 폼에서 컴포넌트를 선택하면 선택한 컴포넌트의 속성과 이벤트가 Object Inspector에 표시됩니다.



폼에 있는 컴포넌트나 객체를 클릭해서 선택할 수 있습니다.



+ 기호를 클릭하면 세부 목록을 열 수 있습니다.

여러 속성들이 색상 이름, *True* 또는 *False*, 정수와 같이 간단한 값을 가집니다. 부울 속성에서 *True*와 *False*를 토글하려면 더블 클릭합니다. 일부 속성은 연결된 속성 편집기를 가지고 있어서 더 복잡한 값을 설정할 수 있습니다. 이러한 속성 값을 클릭하면 생략 부호가 나타납니다. 크기와 같은 일부 속성에 대해서는 값을 입력합니다.

여기를 더블 클릭하여 값을 *True*에서 *False*로 변경합니다.

생략 부호를 클릭하여 이 속성에 대한 속성 편집기를 표시합니다.

아래쪽 화살표를 클릭하여 유효한 값 목록에서 선택합니다.

폼에서 둘 이상의 컴포넌트를 선택하면 Object Inspector는 선택한 컴포넌트들이 공유하는 속성들을 모두 표시합니다.

또한 Object Inspector는 확장된 인라인 컴포넌트 참조를 지원합니다. 이를 통해 참조된 컴포넌트를 선택하지 않고도 참조된 컴포넌트의 속성과 이벤트에 액세스할 수 있습니다. 예를 들어 버튼과 팝업 메뉴 컴포넌트를 폼에 추가하는 경우, 버튼 컴포넌트 선택 시 Object Inspector에서 *PopupMenu* 속성을 *PopupMenu1*로 설정하여 팝업 메뉴의 속성을 모두 표시할 수 있습니다.

Button 컴포넌트의 *PopupMenu* 속성을 *PopupMenu1*로 설정하면 + 기호 클릭 시 팝업 메뉴의 속성이 모두 나타납니다.

인라인 컴포넌트 참조는 빨간색으로 표시되고 하위 속성은 초록색으로 표시됩니다.

자세한 내용은...

온라인 도움말 색인의 "Object Inspector"를 참조하십시오.

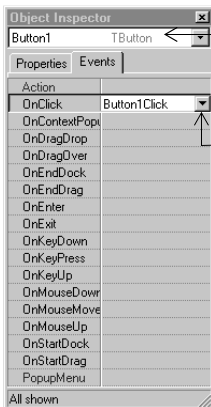
코드 작성

모든 애플리케이션의 핵심 부분은 각 컴포넌트의 코드 부분입니다. Delphi의 RAD 환경은 미리 패키징된 비주얼 컴포넌트나 논비주얼 컴포넌트와 같은 빌딩 블록 대부분을 제공하지만 이벤트 핸들러, 메소드 및 몇몇 클래스는 개발자가 직접 만들어야 합니다. 이 작업을 돕기 위해 만들어진 Delphi의 VCL 및 CLX 클래스 라이브러리에서 수천 개의 객체를 이용할 수 있습니다. 소스 코드를 편집하려면 2-7 페이지의 "코드 에디터"를 참조하십시오.

이벤트 핸들러 작성

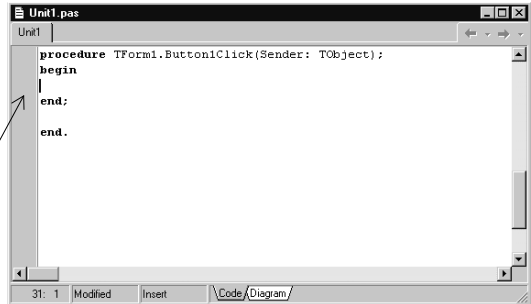
코드는 런타임 시 컴포넌트에 발생하는 이벤트에 응답해야 합니다. 이벤트는 버튼을 클릭하는 것처럼 시스템에서 발생하는 사건과 이 사건에 응답하는 코드 부분 간의 연결입니다. 이 응답 코드가 이벤트 핸들러입니다. 이 코드는 속성 값을 수정하고 메소드를 호출합니다.

폼에서 컴포넌트에 대해 이미 정의된 이벤트 핸들러를 보려면 컴포넌트를 선택하고 Object Inspector에서 Events 탭을 클릭합니다.



여기서 Button1이 선택되고 그 타입은 TButton으로 표시됩니다. Button 컴포넌트가 처리할 수 있는 이벤트를 보려면 Object Inspector에서 Events 탭을 클릭합니다.

드롭다운 목록에서 기존 이벤트 핸들러를 선택합니다.
또는 값 열에서 더블 클릭하면 Delphi는 새 이벤트 핸들러에 대한 뼈대 코드를 생성합니다.



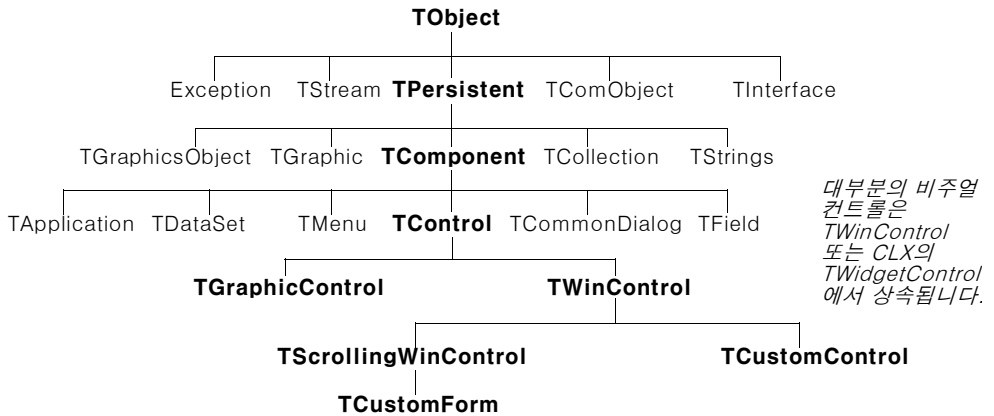
자세한 내용은...

온라인 도움말 색인의 "events"를 참조하십시오.

VCL 및 CLX 라이브러리 사용

Delphi에는 여러 객체들로 구성된 클래스 라이브러리가 두 개 있는데 이 객체들 중에는 코드를 작성할 때 사용할 수 있는 컴포넌트나 컨트롤도 있습니다. 사용자는 Windows 애플리케이션에 대한 Visual Component Library (VCL)와 Linux 애플리케이션에 대한 크로스 플랫폼용 Borland 컴포넌트 라이브러리 (CLX)를 사용할 수 있습니다. 이러한 라이브러리에는 데이터셋과 타이머와 같은 논비주얼 (nonvisual) 컨트롤뿐만 아니라 편집 컨트롤, 버튼 및 다른 사용자 인터페이스 요소와 같은 런타임 시 볼 수 있는 객체도

포함되어 있습니다. 다음 다이어그램은 VCL을 구성하는 일부 주요 클래스를 보여 줍니다. CLX 계층 구조도 이와 유사합니다.



TComponent의 자손 객체에는 컴포넌트 팔레트에 설치할 수 있고 Delphi 폼과 데이터 모듈에 추가할 수 있는 속성과 메소드가 있습니다. VCL 및 CLX 컴포넌트가 IDE로 훅(hook)되기 때문에 폼 디자이너와 같은 툴을 사용하여 애플리케이션을 빠르게 개발할 수 있습니다.

컴포넌트는 고도로 캡슐화되어 있습니다. 예를 들면, 버튼은 OnClick 이벤트를 발생시켜서 마우스 클릭에 응답하도록 미리 프로그래밍되어 있습니다. VCL 또는 CLX 버튼 컨트롤을 사용하면 버튼을 클릭할 때 발생하는 이벤트를 처리하기 위해 사용자가 코드를 작성할 필요가 없으므로 클릭 그 자체에 응답하여 실행되는 애플리케이션 로직만 관리하면 됩니다.

대부분의 Delphi 에디션에는 VCL 및 CLX 소스 코드와 오브젝트 파스칼 프로그래밍 기법의 예제들이 함께 제공됩니다.

자세한 내용은...

도움말 목차에서 "Visual Component Library Reference"와 "CLX Reference" 및 온라인 도움말 색인의 "VCL"을 참조하십시오. CLX에 대한 오픈 소스와 라이선스 옵션에 대한 내용은 <http://www.borland.com/delphi>를 참조하십시오.

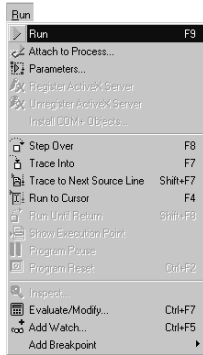
프로젝트 컴파일 및 디버깅

코드를 작성하고 나면 프로젝트를 컴파일하고 디버깅해야 합니다. Delphi를 사용하면 먼저 프로젝트를 컴파일한 다음 따로 디버깅하거나 아니면 통합 디버거를 사용하여 컴파일과 디버깅을 동시에 수행할 수 있습니다. 디버깅 정보로 프로그램을 컴파일하려면 Project|Options를 선택하고 Compiler 페이지를 클릭한 다음 Debug informations가 선택되어 있는지 확인합니다.

Delphi는 통합 디버거를 사용하므로 프로그램 실행을 제어하고, 변수를 관찰하고, 데이터 값을 수정할 수 있습니다. 코드를 한 줄씩 작성해 나가면서 각 브레이크 포인트에서

프로그램 상태를 확인할 수 있습니다. 통합 디버거를 사용하려면 Tools|Debugger Options를 선택하고 General 페이지를 클릭한 다음, Integrated debugging이 선택되어 있는지 확인합니다.

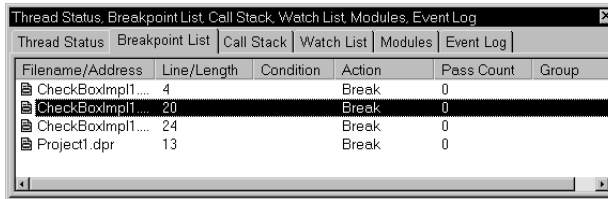
Debug 툴바의 Run 버튼을 클릭하거나, Run|Run을 선택하거나, F9를 눌러 IDE에서 디버깅 세션을 시작할 수 있습니다.



Run 메뉴에서 디버깅 명령을 선택합니다. 일부 명령은 툴바에도 있습니다.



통합 디버거를 사용하면 Breakpoints, Call Stack, Watches, Local Variables, Threads, Modules, CPU, Event Log 등을 비롯한 여러 디버깅 창을 사용할 수 있습니다. View|Debug Windows를 선택하여 디버깅 창을 표시합니다. 몇몇 디버거 뷰는 일부 Delphi 에디션에서 사용할 수 없습니다.



여러 디버깅 창을 결합하여 더 쉽게 사용할 수 있습니다.

디버깅 창을 결합하여 디버깅을 더 쉽게 하는 방법에 대한 자세한 내용은 5-2 페이지의 "툴 윈도우 도킹"을 참조하십시오.

일단 디버깅용으로 데스크탑을 설정했다면 이 설정을 디버깅 또는 런타임 데스크탑으로 저장할 수 있습니다. 이 데스크탑 레이아웃은 모든 애플리케이션을 디버깅할 때마다 사용됩니다. 자세한 내용은 5-4 페이지의 "데스크탑 레이아웃 저장"을 참조하십시오.

자세한 내용은...

온라인 도움말 색인에서 "debugging"과 "integrated debugger"를 참조하십시오.

애플리케이션 배포

다른 사람들이 설치하고 실행할 수 있도록 애플리케이션을 배포할 수 있습니다. 애플리케이션을 배포하려면 실행 파일, DLL, 패키지 파일 및 helper 애플리케이션과 같은 필수 파일과 지원 파일이 모두 필요합니다. Delphi는 이러한 파일들로 설치 프로그램을 만드는 InstallShield Express라는 설치 툴킷을 번들로 제공합니다. InstallShield Express를 설치하려면 Delphi 설치 화면에서 Delphi용 InstallShield Express Custom Edition을 선택합니다.

자세한 내용은...

온라인 도움말 색인의 "deploying, applications"를 참조하십시오.

애플리케이션 국제화

Delphi는 애플리케이션 국제화 및 지역화를 위한 여러 기능들을 제공합니다. IDE와 VCL은 프로젝트를 국제화할 수 있도록 IME와 확장 문자 집합을 지원합니다. Delphi에는 소프트웨어 지역화 및 다른 로케일로 동시 개발을 위한 번역 도구가 포함되어 있지만 일부 Delphi 에디션에서는 이러한 변환 도구를 사용할 수 없습니다. 번역 도구를 사용하면 애플리케이션의 여러 언어로 지역화된 버전을 단일 프로젝트로 관리할 수 있습니다.

번역 도구의 세 가지 통합 도구는 다음과 같습니다.

- Resource DLL 마법사는 리소스 DLL을 생성하고 관리합니다.
- Translation Manager는 번역된 리소스를 보고 편집하기 위한 테이블입니다.
- Translation Repository는 번역을 저장하기 위한 공유 데이터베이스입니다.

Resource DLL 마법사를 열려면 File|New|Other를 선택하고 Resource DLL Wizard 아이콘을 더블 클릭합니다. 번역 도구를 구성하려면 Tools|Translation Tools Options를 선택합니다.

자세한 내용은...

온라인 도움말 색인의 "international applications"를 참조하십시오.

프로젝트 타입

Delphi의 모든 에디션은 범용 32비트 Windows 프로그래밍, DLL, 패키지, 사용자 지정 컴포넌트, 다중 스레드, COM(Component Object Model) 및 Automation 컨트롤러, 다중 프로세스 디버깅을 지원합니다. 일부 에디션은 웹 서버 애플리케이션, 데이터베이스 애플리케이션, COM 서버, 다계층 애플리케이션, CORBA 및 의사 결정 지원 시스템 등의 서버 애플리케이션을 지원합니다.

자세한 내용은...

사용하는 에디션에서 지원하는 도구를 보려면 www.borland.com/delphi에서 기능 목록을 참조하십시오.

CLX 애플리케이션

Delphi를 사용하여 Linux에서 실행하도록 프로젝트를 컴파일, 디버깅 및 배포하는 Kylix로 포팅할 수 있는 크로스 플랫폼 애플리케이션을 개발할 수 있습니다. CLX 애플리케이션을 개발하려면 File|New|CLX Application을 선택합니다. CLX 애플리케이션에서만 사용할 수 있는 컴포넌트 및 항목이 컴포넌트 팔레트와 Object Repository에 나타난다는 것을 제외하면 IDE는 일반적인 Delphi 애플리케이션의 IDE와 유사합니다. Delphi에서 지원되는 Windows 특정 기능은 Linux 환경으로 직접 포팅되지 않습니다.

자세한 내용은...

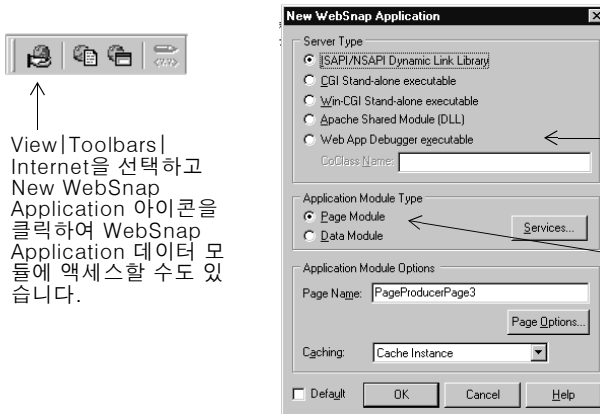
크로스 플랫폼 애플리케이션 개발을 위해 사용할 수 있는 컴포넌트를 보려면 온라인 도움말 목차의 "CLX Reference"를 참조하십시오.

웹 서버 애플리케이션

웹 서버 애플리케이션은 클라이언트의 요청을 처리하고 웹 페이지의 폼으로 HTTP 메시지를 반환하여 웹 서버와 함께 실행됩니다. Delphi는 Delphi 에디션에 따라 웹에 데이터를 게시하기 위한 두 가지 기술을 제공합니다.

기본적인 웹 서버 애플리케이션을 개발하려면 Windows 애플리케이션과 Linux 애플리케이션에서 요청을 디스패치하고, 액션을 정의하고, HTML 페이지를 생성하고, 이벤트 핸들러를 작성할 수 있는 웹 모듈을 만듭니다. WebBroker 웹 서버 애플리케이션을 만들려면 File|New|Other를 선택하고 Web Server Application 아이콘을 더블 클릭합니다. Internet 및 InternetExpress 컴포넌트 팔레트 페이지의 웹 모듈에 컴포넌트를 추가할 수 있습니다.

WebSnap은 어댑터, 추가 디스패처, 추가 페이지 프로듀서, 세션 지원 및 웹 페이지 모듈로 이 기능을 추가합니다. WebSnap 서버 애플리케이션을 새로 만들려면 File|New|Other를 선택하고 WebSnap 페이지를 클릭한 다음 Web Server Application 아이콘을 더블 클릭합니다. WebSnap 컴포넌트 팔레트 페이지에서 WebSnap 컴포넌트를 추가할 수 있습니다.



View|Toolbars|Internet을 선택하고 New WebSnap Application 아이콘을 클릭하여 WebSnap Application 데이터 모듈에 액세스할 수도 있습니다.

웹 서버 애플리케이션 디버깅에 사용하는 테스트 서버를 비롯한 다양한 웹 서버 애플리케이션 타입에서 실행되는 애플리케이션을 만들 수 있습니다. 작업 중 HTML 페이지를 데이터 모듈 또는 페이지 모듈로 표시할지 여부를 선택합니다.

자세한 내용은...

온라인 도움말 색인의 "Web applications"를 참조하십시오.

데이터베이스 애플리케이션

Delphi는 데이터베이스 애플리케이션 개발 작업을 단순화시키는 다양한 데이터베이스와 연결 도구를 제공합니다.

데이터베이스 애플리케이션을 만들려면 첫째, Data Controls 페이지 컴포넌트를 사용하여 폼에서 인터페이스를 디자인합니다. 둘째, Data Access 페이지를 사용하여 데이터 모듈에 데이터 소스를 추가합니다. 세째, 다양한 데이터베이스 서버에 연결하기 위해 데이터셋과 데이터 연결 컴포넌트를 다음 연결 도구의 이전 또는 해당 페이지에 있는 데이터 모듈에 추가합니다.

- dbExpress는 DB2, InterBase, MySQL 및 Oracle과 같은 SQL 데이터베이스 서버에 빠른 액세스를 제공하는 크로스 플랫폼 애플리케이션용 데이터베이스 드라이버 컬렉션입니다. dbExpress 드라이버를 통해 단방향 데이터셋을 사용하여 데이터베이스에 액세스할 수 있습니다.
- BDE(Borland Database Engine)는 dBASE, Paradox, FoxPro, Microsoft Access 및 ODBC 데이터 소스 등을 비롯한 여러 데이터베이스 형식을 지원하는 드라이버 컬렉션입니다. Delphi의 일부 버전에서 사용할 수 있는 SQL Links 드라이버는 Oracle, Sybase, Informix, DB2, SQL Server 및 InterBase 등의 서버를 지원합니다.
- ADO(ActiveX Data Objects)는 관계형 데이터베이스와 관계형이 아닌 데이터베이스, 전자 메일 및 파일 시스템, 텍스트와 그래픽, 사용자 지정 비즈니스 객체 등 데이터 소스에 대한 Microsoft의 고급 인터페이스입니다.
- InterBase Express (IBX) 컴포넌트는 사용자 지정 데이터 액세스 Delphi 컴포넌트 아키텍처를 기반으로 합니다. IBX 애플리케이션은 고급 InterBase 기능에 대한 액세스를 제공하고 InterBase 5.5 이상에 대한 최고 성능의 컴포넌트 인터페이스를 제공합니다. IBX는 data-aware 컴포넌트의 Delphi 라이브러리와 호환됩니다.

어떤 데이터베이스 연결 도구는 일부 Delphi 에디션에서 사용할 수 없습니다.

자세한 내용은...

온라인 도움말 색인의 "database applications"를 참조하십시오.

BDE Administrator

BDE Administrator (BDEAdmin.exe) 를 사용하여 BDE 드라이버를 구성하고 data-aware VCL 컨트롤에 의해 사용되는 별칭을 설정하여 데이터베이스에 연결합니다.

자세한 내용은...

Windows 시작 메뉴에서 프로그램 | Borland Delphi 6 | BDE Administrator 를 선택합니다. 그런 다음 Help | Contents 를 선택합니다.

SQL Explorer(데이터베이스 탐색기)

SQL Explorer(DBExplor.exe)를 사용하면 데이터베이스를 검색하고 편집할 수 있습니다. SQL Explorer를 사용하여 데이터베이스 별칭을 만들고, 스키마 정보를 보고, SQL 쿼리를 실행하고, 데이터 사전과 속성 설정을 관리할 수 있습니다.

자세한 내용은...

Delphi 메인 메뉴에서 Database|Explore를 선택합니다. 그런 다음 Help|Contents를 선택합니다. 또는 온라인 도움말 색인의 "Database Explorer"를 참조하십시오.

Database Desktop

Database Desktop(DBD32.exe)을 사용하면 다양한 형식으로 Paradox 및 dBase 데이터베이스 테이블을 생성하고 보고 편집할 수 있습니다.

자세한 내용은...

Windows 시작 메뉴에서 프로그램|Borland Delphi 6|Database Desktop을 선택합니다. 그런 다음 Help|User's Guide Contents를 선택합니다.

Data Dictionary

BDE를 사용할 때 Data Dictionary는 애플리케이션과 독립적으로 사용자 지정 저장 영역을 제공하므로 이 영역에서 데이터의 내용과 외관을 설명하는 확장 필드 속성 설정 집합을 만들 수 있습니다. Data Dictionary를 원격 서버에 상주시켜 추가 정보를 공유할 수 있습니다.

자세한 내용은...

Help|Delphi Tools를 선택하여 "Data Dictionary"를 참조하십시오.

사용자 지정 컴포넌트

Delphi에 있는 컴포넌트들은 컴포넌트 팔레트에 이미 설치되어 있으며, 개발에 필요한 대부분의 기능을 제공합니다. 새 컴포넌트를 설치하지 않고도 Delphi로 프로그래밍할 수는 있지만, 간혹 특별한 문제를 해결하거나 사용자 지정 컴포넌트를 필요로 하는 특정한 동작을 나타내고 싶은 경우도 있을 것입니다. 사용자 지정 컴포넌트는 애플리케이션 간에 코드를 재사용하거나 일관성을 유지하는 데 유용합니다.

협력 업체의 사용자 지정 컴포넌트를 설치하거나 아니면 사용자 지정 컴포넌트를 직접 만들 수 있습니다. 새 컴포넌트를 만들려면 Component|New Component를 선택하여 New Component 마법사를 표시합니다. 협력 업체에서 제공하는 컴포넌트를 설치하려면 5-7 페이지의 "컴포넌트 패키지 설치"를 참조하십시오.

자세한 내용은...

개발자 안내서의 5부 "사용자 지정 컴포넌트 생성"과 온라인 도움말 색인의 "components, creating"을 참조하십시오.

DLL

DLL(동적 연결 라이브러리)은 애플리케이션과 다른 DLL에서 호출할 수 있는 루틴을 포함하는 컴파일된 모듈입니다. DLL에는 두 개 이상의 애플리케이션에서 사용하는 코드나 리소스가 포함되어 있습니다. File|New|Other를 선택하고 DLL Wizard 아이콘을 더블 클릭하여 DLL에 대한 템플릿을 만듭니다.

자세한 내용은...

온라인 도움말 색인의 "DLLs"를 참조하십시오.

COM 및 ActiveX

Delphi는 Microsoft의 COM 표준을 지원하고 ActiveX 컨트롤을 생성하는 마법사를 제공합니다. File|New|Other를 선택하고 ActiveX 탭을 클릭하여 마법사에 액세스합니다. 예제 ActiveX 컨트롤은 컴포넌트 팔레트의 ActiveX 페이지에 설치되어 있습니다. 많은 COM 서버 컴포넌트가 컴포넌트 팔레트의 Servers 탭에 들어 있습니다. VCL 컴포넌트처럼 이 컴포넌트를 사용할 수 있습니다. 예를 들어, Microsoft Word 컴포넌트 중 하나를 폼에 놓아 애플리케이션 인터페이스 내에 Microsoft Word의 인스턴스를 가져올 수 있습니다.

자세한 내용은...

온라인 도움말 색인의 "COM"과 "ActiveX"를 참조하십시오.

타입 라이브러리

타입 라이브러리는 데이터 타입, 인터페이스, 멤버 함수, ActiveX 컨트롤이나 서버에 의해 노출된 객체 클래스에 대한 정보가 들어 있는 파일입니다. COM 애플리케이션 또는 ActiveX 라이브러리에 타입 라이브러리를 포함하여 다른 애플리케이션과 프로그래밍 도구에서 사용할 수 있는 항목에 대한 정보를 만듭니다. Delphi는 타입 라이브러리를 만들고 유지 관리하기 위한 타입 라이브러리 에디터를 제공합니다.

자세한 내용은...

온라인 도움말 색인의 "type libraries"를 참조하십시오.

4

텍스트 에디터(자습서) 만들기

이 자습서는 메뉴, 툴바, 상태 표시줄 등을 완비한 텍스트 에디터를 만드는 과정을 보여줍니다.

참고 이 자습서는 Delphi의 모든 에디션과 Windows 플랫폼에서만 사용할 수 있습니다.

새 애플리케이션 시작

새 애플리케이션을 시작하기 전에 소스 파일을 저장할 디렉토리를 만듭니다.

- 1 C:\Program Files\Borland\Delphi6\Projects 디렉토리에 TextEditor라는 디렉토리를 만듭니다.
- 2 새 프로젝트를 엽니다.

각 애플리케이션을 *프로젝트*라고 합니다. Delphi를 시작하면 기본적으로 빈 프로젝트가 만들어집니다. 다른 프로젝트가 이미 열려 있을 경우 File|New Application을 선택하여 새 프로젝트를 만듭니다.

새 프로젝트를 열면 Delphi는 자동으로 다음 파일들을 만듭니다.

- *Project1.dpr*: 프로젝트와 연결된 소스 코드 파일. *프로젝트 파일*이라고 합니다.
- *Unit1.pas*: 메인 프로젝트 폼과 연결된 소스 코드 파일. *유닛 파일*이라고 합니다.
- *Unit1.xfm*: 메인 프로젝트 폼에 대한 정보를 저장하는 리소스 파일. *폼 파일*이라고 합니다.

각 폼은 자체 유닛 파일 (*Unit1.pas*)과 폼 파일 (*Unit1.xfm*)을 가집니다. 두 번째 폼을 만들면 두 번째 유닛 파일 (*Unit2.pas*)과 폼 파일 (*Unit2.dfm*)이 자동으로 만들어집니다.

3 파일을 디스크에 저장하려면 File|Save All을 선택합니다. Save 대화 상자가 나타나면 다음과 같이 합니다.

- TextEditor 폴더를 탐색합니다.
- Unit1을 기본 이름 Unit1.pas로 저장합니다.
- TextEditor.dpr이라는 이름으로 프로젝트를 저장합니다. (실행 파일은 확장자가 exe이며 프로젝트 이름과 동일한 이름으로 지정됩니다.)

나중에 File|Save All을 선택하여 작업을 다시 저장합니다.

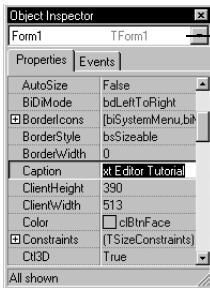
프로젝트를 저장하면 Delphi는 프로젝트 디렉토리에 추가 파일을 만듭니다. 추가 파일에는 TextEditor.dof라는 Delphi Options 파일, TextEditor.cfg라는 구성 파일, TextEditor.res라는 Windows 리소스 파일 등이 있습니다. 이러한 파일들에 대해 신경 쓸 필요는 없지만 삭제해서는 안됩니다.

속성 값 설정

새 프로젝트를 열 때 Delphi는 기본적으로 *Form1*이라는 프로젝트의 메인 폼을 표시합니다. 이 폼에 컴포넌트를 두면 애플리케이션의 사용자 인터페이스와 기타 부분을 만들 수 있습니다.

폼 옆에는 컴포넌트와 폼의 속성 값을 설정하는 데 사용하는 Object Inspector가 나타납니다. 속성을 설정할 때 Delphi는 사용자를 대신해서 소스 코드를 관리합니다. Object Inspector에서 설정하는 값은 *디자인 타임* 설정이라고 합니다.

1 Object Inspector에서 폼의 *Caption* 속성을 찾아서 기본 캡션 Form1 대신에 Text Editor Tutorial을 입력합니다. 입력한 내용에 따라 폼의 헤더에 있는 캡션이 바뀌는 것에 유의하십시오.



Object Inspector의 상단에 있는 드롭다운 목록에 현재 선택되어 있는 컴포넌트가 표시됩니다. 이 그림에서 컴포넌트는 *Form1*이고 타입은 *TForm1*입니다.

컴포넌트를 선택하면 Object Inspector는 컴포넌트의 속성을 표시합니다.

- 2 폼에 컴포넌트가 없어도 *F9* 키를 눌러서 지금 폼을 실행해 보십시오.



폼에 컴포넌트가 없어도 폼의 런타임 뷰는 최소화, 최대화, 닫기 버튼을完비한 디자인 타임 뷰와 유사하게 보입니다.

- 3 Form1의 디자인 타임 뷰로 돌아가려면 다음 중 하나를 수행합니다.



- 폼의 런타임 뷰에서 애플리케이션의 제목 표시줄의 오른쪽 위 모서리에 있는 **X**를 클릭합니다.(폼의 런타임 뷰)
- 제목 표시줄의 왼쪽 위 모서리에 있는 애플리케이션 종료 버튼을 클릭하고 Close를 클릭합니다.
- View|Forms를 선택하고 Form1을 선택한 다음 OK를 클릭합니다.
- Run|Program Reset을 선택합니다.

폼에 컴포넌트 추가

폼에 컴포넌트를 추가하기 전에 애플리케이션에서 사용할 사용자 인터페이스(UI)를 만드는 최상의 방법에 대해 고려해야 합니다. UI는 사용자와 애플리케이션 사이를 인터페이스하는 것이므로 사용하기 쉽게 디자인해야 합니다.

Delphi에는 애플리케이션의 부분을 나타내는 컴포넌트들이 많이 있습니다. 예를 들면, 컴포넌트 팔레트에는 메뉴, 툴바, 대화 상자, 그리고 그 밖의 여러 비주얼한, 논비주얼한 프로그램 요소들을 쉽게 프로그래밍할 수 있는 컴포넌트들(객체라고도 함)이 있습니다.

텍스트 에디터 애플리케이션에는 편집 영역, 편집 중인 파일 이름과 같은 정보를 표시하는 상태 표시줄, 메뉴, 명령에 쉽게 액세스하는 아이콘이 있는 툴바 등이 필요합니다. Delphi를 사용한 인터페이스 디자인의 장점은 다른 여러 가지 컴포넌트를 시도해 볼 수 있고 그 결과를 즉시 볼 수 있다는 것입니다. 이렇게 하면 애플리케이션 인터페이스의 프로토타입을 신속하게 만들 수 있습니다.

텍스트 에디터 디자인을 시작하려면, 다음과 같은 방법으로 폼에 *RichEdit*와 *StatusBar* 컴포넌트를 추가합니다.

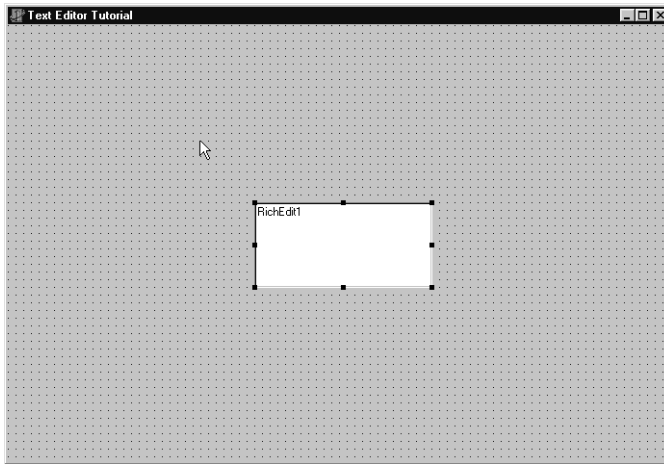


- 1 텍스트 영역을 만들려면 먼저 *RichEdit* 컴포넌트를 추가합니다. *RichEdit* 컴포넌트를 찾으려면 컴포넌트 팔레트의 Win32 페이지에서 팔레트의 아이콘을 잠시 가리킵니다. 그러면 Delphi는 컴포넌트의 이름을 보여 주는 도움말 툴팁을 표시합니다.



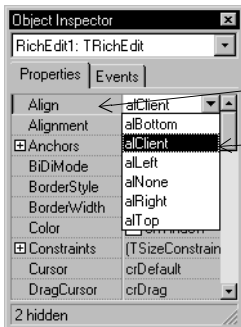
RichEdit 컴포넌트를 찾으면 다음 중 하나를 수행합니다.

- 팔레트에서 컴포넌트를 선택하고 컴포넌트를 두려는 위치에서 폼을 클릭합니다.
- 더블 클릭하여 컴포넌트를 폼의 가운데에 둡니다.



각 Delphi 컴포넌트는 클래스이며 폼에 컴포넌트를 두는 것은 그 클래스의 인스턴스를 만드는 것입니다. 일단 컴포넌트가 폼에 있으면 Delphi는 애플리케이션이 실행 중일 때 객체의 인스턴스를 만드는 데 필요한 코드를 생성합니다.

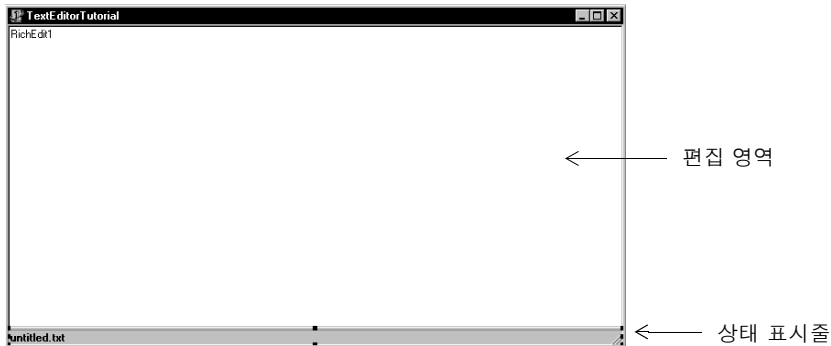
- 2 *RichEdit* 컴포넌트가 선택되어 있으면 Object Inspector에서 *Align* 속성의 드롭다운 화살표를 클릭하고 속성을 *alClient*로 설정합니다.




폼에서 *RichEdit1* 컴포넌트가 선택되어 있는지 확인합니다.
Object Inspector에서 *Align* 속성을 찾습니다. 아래쪽 화살표를 클릭하여 속성의 드롭다운 목록을 표시합니다. *alClient*를 선택합니다.

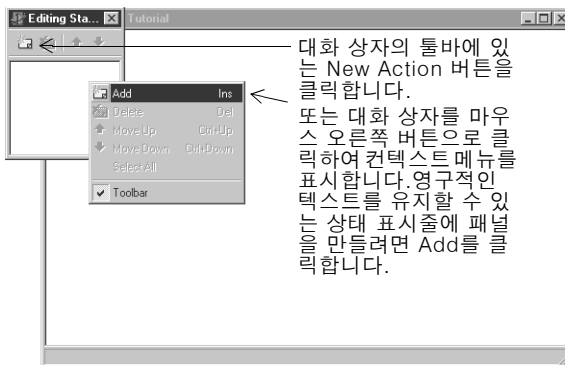
이제 *RichEdit* 컴포넌트가 폼 전체를 채워서 텍스트 편집 영역이 넓어졌습니다. *Align* 속성에 대한 *alClient* 값을 선택하면 폼의 크기를 조정하더라도 어떤 크기의 창도 채울 수 있게 *RichEdit* 컨트롤의 크기가 변합니다.

- 3 컴포넌트 팔레트의 Win32 페이지에서 *StatusBar* 컴포넌트를 더블 클릭합니다. 이렇게 하면 폼의 하단에 상태 표시줄이 추가됩니다.
- 4 다음과 같은 방법으로 상태 표시줄에 하나의 패널을 만들어 텍스트 에디터로 편집 중인 파일의 이름과 경로를 표시합니다.
 - 상태 표시줄이 선택되어 있는지 확인합니다.
 - *SimpleText* 속성 다음에 *untitled.txt* 를 입력합니다. 텍스트 에디터 사용 시 편집 중인 파일을 아직 저장하지 않았다면 파일 이름은 *untitled.txt*가 됩니다.



- Editing *StatusBar1.Panels* 대화 상자를 열려면 *Panels* 속성의 (*TStatusBarPanel*) 생략 버튼을 클릭합니다.
- 대화 상자의 툴바에서 *New Action* 버튼  을 클릭하여 상태 표시줄에 패널을 추가합니다.

팁 또한 폼에서 상태 표시줄을 더블 클릭해서 Editing *StatusBar1.Panels* 대화 상자에 액세스할 수 있습니다.



Panels 속성은 인덱스가 0부터 시작하는 배열 (zero-based array)이므로 고유한 인덱스 값을 기반으로 만든 각 패널에 액세스할 수 있습니다. 기본적으로 첫 번째 패널은 0 값을 가집니다. *Add*를 클릭할 때마다 상태 표시줄에 패널이 추가됩니다.

- 5 X를 클릭하여 `Editing StatusBar1.Panels` 대화 상자를 닫습니다.
이제 텍스트 에디터에 대한 사용자 인터페이스의 메인 편집 영역이 설정되었습니다.

메뉴와 툴바에 대한 지원 추가

애플리케이션으로 어떤 작업을 하려면 메뉴, 명령, 편리하게 사용하기 위한 툴바가 필요합니다. 명령을 따로 코딩할 수 있지만 Delphi는 코드를 한 가지로 통일하는 *액션 매니저 (Action Manager)*와 이미지를 한 가지로 통일하는 *이미지 목록 (Image List)*을 제공하므로 통일된 방법으로 메뉴와 툴바에 명령을 추가할 수 있습니다.

규칙에 따라 메뉴 명령에 연결된 액션의 이름은 상위 레벨 메뉴 이름과 명령 이름으로 지정합니다. 예를 들면, `FileExit` 액션은 File 메뉴의 Exit 명령을 의미합니다.

다음은 예제 텍스트 에디터 애플리케이션에 필요한 액션의 종류입니다.

표 4.1 Text Editor 명령 계획

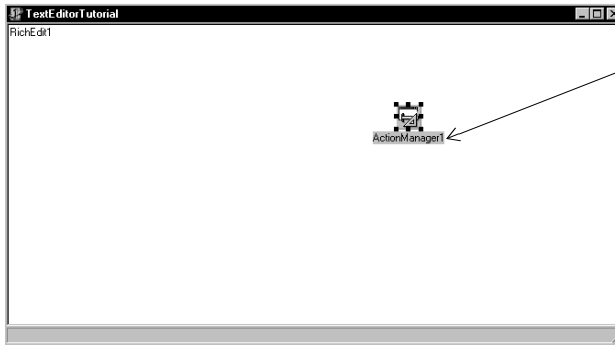
메뉴	명령	툴바에 포함	설명
File	New	예	새 파일을 만듭니다.
File	Open	예	편집용 기존 파일을 엽니다.
File	Save	예	현재 파일을 디스크에 저장합니다.
File	Save As	아니오	파일을 새 이름 또는 지정된 이름으로 새 파일을 저장할 수 있습니다.
File	Exit	예	에디터 프로그램을 종료합니다.
Edit	Cut	예	텍스트를 삭제하고 클립보드에 저장합니다.
Edit	Copy	예	텍스트를 복사하고 클립보드에 저장합니다.
Edit	Paste	예	클립보드에서 가져온 텍스트를 삽입합니다.
Help	Contents	아니오	도움말 항목에 액세스할 수 있는 도움말 목차 화면을 표시합니다.
Help	Index	아니오	도움말 색인 화면을 표시합니다.
Help	About	아니오	애플리케이션에 대한 정보를 상자에 표시합니다.

액션 매니저의 코드와 이미지를 모두 통일시키려면 Action Manager 에디터를 프로젝트에 추가해야 합니다.



- 1 컴포넌트 팔레트의 Additional 페이지에서 *ActionManager* 컴포넌트를 더블 클릭하여 폼에 놓습니다. 이 컴포넌트는 논비주얼(nonvisual) 컴포넌트이므로 폼의 어느 곳에 두든지 상관 없습니다.

- 2 폼에 놓은 년비주얼 컴포넌트의 캡션을 표시하려면 Tools | Environment Options를 선택하고 Designer 페이지를 클릭한 다음 Show component captions를 선택하고 OK를 클릭합니다.



폼에 놓은 컴포넌트의 캡션을 표시하려면 Tools | Environment Options | Designer를 선택하고 Show component captions를 클릭합니다.
 ActionManager 컴포넌트는 년비주얼 컴포넌트이므로 애플리케이션이 실행 중일 때는 보이지 않습니다.

Action Manager 컴포넌트에 Action 추가

먼저 Action Manager 컴포넌트에 Action을 추가하고 속성을 설정합니다. 규칙에 따라 메뉴 명령에 연결된 액션의 이름을 상위 레벨 메뉴 이름과 명령 이름으로 지정합니다. 예를 들면, FileExit 액션은 File 메뉴의 Exit 명령을 의미합니다.

속성을 설정하는 액션과 속성을 자동으로 설정하는 표준 액션을 모두 추가합니다.

- 1 ActionManager 컴포넌트를 더블 클릭하여 엽니다.

Editing Form1.ActionManager1 대화 상자 또는 Action Manager 에디터가 나타납니다.

- 2 Actions 탭이 표시되어 있는지 확인합니다. New Action 버튼 옆의 드롭다운 화살표를 클릭하고 New Action을 클릭합니다.

팁

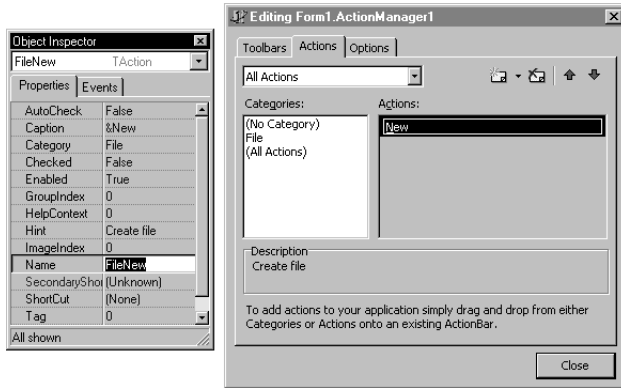
Action Manager 에디터를 마우스 오른쪽 버튼으로 클릭한 후 New Action을 선택할 수도 있습니다.



New Action 버튼 옆의 드롭다운 화살표를 클릭하여 액션 매니저에 새로운 액션을 만듭니다.
 Delete 버튼이 활성화되어 있으면 액션 리스트에서 기존 액션을 제거할 수 있습니다.

- 3 No Category가 선택되어 있으면 액션 리스트에서 Action1을 클릭합니다. Object Inspector에서 다음 속성을 설정합니다.
 - *Caption* 뒤에 &New를 입력합니다. 문자 앞에 앤퍼샌드(&)를 입력하면 명령에 액세스하는 단축키가 만들어집니다.
 - *Category* 뒤에 File을 입력하면 File 명령이 만들어집니다.
 - *Hint* 뒤에 Create file을 입력하면 도움말 툴팁이 됩니다.
 - *ImageIndex* 뒤에 6을 입력하면 ImageList에 있는 이미지 번호 6을 이 액션에 연결시킵니다.
 - *Name* 뒤에 FileNew를 입력하고 (File|New 명령) *Enter*를 눌러 변경 사항을 저장합니다.

Action Manager 에디터에 Action1이 선택되어 있으면 Object Inspector에서 속성을 변경합니다.
*Caption*은 액션의 이름이고, *Category*는 액션의 타입이며, *Hint*는 도움말 툴팁입니다. *ImageIndex*를 통해 이미지 목록의 이미지를 참조할 수 있으며 *Name*은 코드에서의 액션 이름입니다.



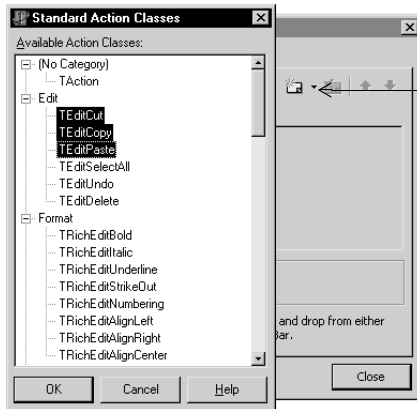
- 4 New Action 버튼 옆의 드롭다운 화살표를 클릭하고 New Action을 클릭합니다.
- 5 No Category를 선택한 상태에서 Action1을 클릭합니다. Object Inspector에서 다음 속성을 설정합니다.
 - *Caption* 뒤에 &Save를 입력합니다.
 - *Category* 뒤에 드롭다운 화살표를 클릭하고 File을 클릭합니다.
 - *Hint* 뒤에 Save file을 입력합니다.
 - *ImageIndex* 뒤에 8을 입력합니다.
 - *Name* 뒤에 FileSave를 입력합니다 (File|Save 명령).
- 6 New Action 버튼 옆의 드롭다운 화살표를 클릭하고 New Action을 클릭합니다.
- 7 No Category를 선택한 상태에서 Action1을 클릭합니다. Object Inspector에서 다음 속성을 설정합니다.
 - *Caption* 뒤에 &Index를 입력합니다.
 - *Category* 뒤에 Help를 입력합니다.
 - *Name* 뒤에 HelpIndex를 입력합니다 (Help|Index 명령).

- 8 New Action 버튼 옆의 드롭다운 화살표를 클릭하고 New Action을 클릭합니다.
- 9 No Category 옆의 Action1을 선택합니다. Object Inspector에서 다음 속성을 설정합니다.
 - *Caption* 뒤에 &About을 입력합니다.
 - *Category*가 Help를 표시하는지 확인합니다.
 - *Name* 뒤에 HelpAbout을 입력합니다(Help|About 명령).
- 10 화면에서 Action Manager 에디터를 닫지 않고 열어 둡니다.

Action Manager 컴포넌트에 표준 Action 추가

이제, Action Manager 컴포넌트에 표준 Action(open, save as, exit, cut, copy, paste 및 help 등)을 추가합니다.

- 1 Action Manager 에디터를 계속 열어 두어야 합니다. Action Manager 에디터를 닫았을 경우 *ActionManager* 컴포넌트를 더블 클릭하여 Action Manager 에디터를 엽니다.
- 2 New Action 버튼 옆의 드롭다운 화살표를 클릭하고 New Standard Action을 클릭합니다.
Standard Actions Classes 대화 상자가 나타납니다.
- 3 Standard Actions Classes 대화 상자에서 Edit 범주를 스크롤하여 *TEditCut*, *TEditCopy*, *TEditPaste*를 선택합니다. OK를 클릭하여 이 액션들을 *Editing Form1.Action Manager1* 대화 상자의 Categories 목록에 있는 새 Edit 범주에 추가합니다.



New Action 버튼을 클릭하고 New Standard Action을 선택합니다.

그러면 사용 가능한 표준 액션이 표시됩니다. 액션을 선택하려면 해당 액션을 더블 클릭합니다.

- 4 New Action 버튼 옆의 드롭다운 화살표를 클릭하고 New Standard Action을 클릭합니다.
- 5 File 범주를 스크롤하여 *TFileOpen*, *TFileSaveAs*, *TFileExit* 액션을 선택합니다. OK를 클릭하여 File 범주에 이 액션들을 추가합니다.

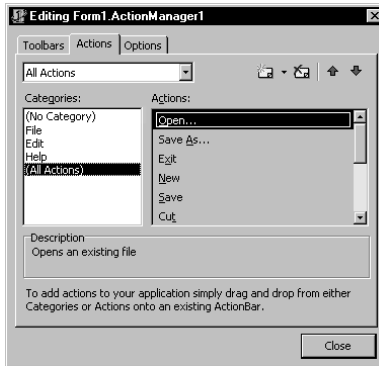
- 6 New Action 버튼 옆의 드롭다운 화살표를 클릭하고 New Standard Action을 클릭합니다.
- 7 Standard Actions Classes 대화 상자에서 Help 범주를 스크롤하여 *THelpContents*를 선택합니다. OK를 클릭하여 Help 범주에 이 액션을 추가합니다.

참고

사용자 지정 Help|Contents 명령을 추가하면 Help Contents 탭과 함께 도움말 파일이 표시됩니다. 표준 Help|Contents 명령은 Contents, Index 또는 Find가 표시된 최근의 탭 모양 페이지를 불러옵니다.

이제 애플리케이션에 필요한 모든 표준 액션이 추가되었습니다. 표준 액션은 이미지 인덱스를 포함하여 해당 속성을 자동으로 설정합니다.

- 8 All Actions를 클릭하여 방금 추가한 비표준 액션과 표준 액션을 모두 표시합니다.



All Actions를 클릭하면 모든 범주에 방금 추가한 액션이 표시됩니다.

- 9 Close 버튼을 클릭하여 Action Manager 에디터를 닫습니다.

- 10 File|Save All을 클릭하여 변경 사항을 저장합니다.

Image List 컴포넌트에 이미지 추가

이 단원에서는 메뉴와 툴바에서 사용할 이미지를 액션 매니저에 추가합니다.

표준 액션은 Delphi와 함께 제공된 기본 제공 Image List 컴포넌트의 사전 지정된 이미지와 연결됩니다. 예를 들어, Edit|Cut 액션의 이미지 인덱스는 0입니다. 텍스트 에디터 명령에 사용할 모든 이미지가 이 파일에 들어 있습니다.

다음과 같은 방법으로 이미지 목록을 추가합니다.

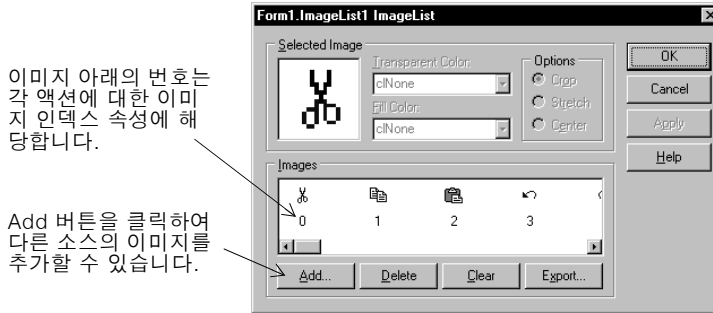
- 1 기본 디렉토리에 Delphi를 설치한 경우 C:\Program Files\Borland\Delphi6\Source\Vcl\ActnRes.pas를 엽니다. StandardsActions 창이 열립니다.



- 2 *ImageList1* 컴포넌트를 선택하고 선택한 컴포넌트를 잘라내어 폼에 붙여넣습니다. 이 컴포넌트는 닌비주얼 (nonvisual) 컴포넌트이므로 아무 위치에나 둘 수 있습니다. 코드 에디터에 ActnRes.pas 유닛이 추가됩니다.

- *ImageList1*을 복사하려면 컴포넌트를 마우스 오른쪽 버튼으로 클릭하고 Edit|Copy를 선택합니다. 폼에서 마우스 오른쪽 버튼을 클릭한 후 Edit|Paste를 선택합니다.

- 3 StandardActions 창을 닫습니다.
- 4 *ImageList1*을 더블 클릭하여 사용 가능한 모든 이미지를 표시합니다.



다음은 각 명령에 사용하는 이미지 인덱스 번호입니다.

명령	ImageIndex 속성
Edit Cut	0
Edit Copy	1
Edit Paste	2
File New	6
File Open	7
File Save	8
File SaveAs	30
File Exit	43
Help Contents	40

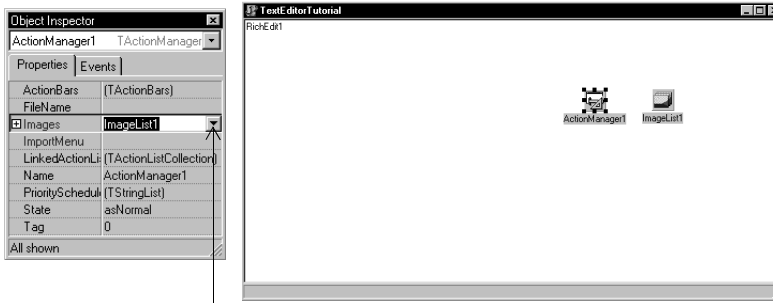
참고

완전히 다른 목록의 이미지를 추가할 수 있습니다. *Form1.ImageList* 대화 상자에서 Add 버튼을 클릭하여 제품의 Buttons 디렉토리로 이동합니다. 기본 위치는 C:\Program Files\Common Files\Borland Shared\Images\Buttons입니다.

예를 들어, File | Open 명령의 경우 fileopen.bmp를 더블 클릭합니다. 비트맵을 두 개로 따로 구분할지 여부를 묻는 메시지가 나타나면 Yes를 선택합니다. 각 이미지는 이미지의 활성 버전과 비활성 버전을 포함합니다. 두 가지 이미지를 모두 볼 수 있습니다. 비활성(두 번째) 이미지를 삭제합니다. 그런 다음 Object Inspector의 이미지 인덱스가 이미지 목록의 해당 이미지에 할당된 새 번호와 일치하는지 확인합니다.

- 5 OK를 클릭하여 *ImageList* 대화 상자를 닫습니다.

6 *ActionManager* 컴포넌트를 선택하고 *Images* 속성을 *ImageList1*로 설정합니다.



Images 속성 옆에 있는 아래쪽 화살표를 클릭합니다. *ImageList1*을 선택합니다. 이렇게 하면 이미지 목록에 추가할 이미지가 액션 매니저에 있는 액션에 연결됩니다.

모든 액션에 대해서 이미지 인덱스를 이미 설정했으므로 자동으로 이미지가 해당 액션에 추가됩니다. 액션에 8개의 이미지가 연결됐습니다.

7 액션 매니저의 연결된 이미지를 보려면 *ActionManager* 컴포넌트를 열고 Actions 탭이 선택되어 있는지 확인한 후 All Actions 범주를 클릭합니다.



이제 Action Manager 에디터를 표시하면 이 액션과 연결된 이미지를 볼 수 있습니다.

8 File|Save All을 선택하여 변경 사항을 저장합니다.

9 Action Manager 에디터를 단지 열고 열어 둡니다.

이제 메뉴와 툴바를 추가할 준비가 되었습니다.

메뉴 추가

다음 두 단원에서는 *액션 밴드 (action bands)*라는 사용자 지정 메뉴 표시줄과 툴바를 추가합니다.

메인 메뉴 표시줄에는 세 개의 드롭다운 메뉴, 즉 File, Edit, Help와 해당 메뉴 명령이 들어 있습니다. Action Manager 에디터를 사용하여 각 메뉴 범주와 해당 명령을 메뉴 표시줄로 동시에 끌어올 수 있습니다.



- 1 컴포넌트 팔레트의 Additional 페이지에서 *ActionMainMenuBar* 컴포넌트를 더블 클릭하여 컴포넌트를 폼에 추가합니다.

폼의 상단에 빈 메뉴 표시줄이 나타납니다.

- 2 빈 메뉴 표시줄이 나타나지 않는 경우 Action Manager 에디터를 열고 Categories 목록에서 File 을 선택합니다. 하위 메뉴 명령이 원하는 순서로 나타나지 않아도 Move Up 버튼과 Move Down 버튼 또는 *Shift+↑* 및 *Shift+↓*를 사용하여 이 순서를 쉽게 변경할 수 있습니다.

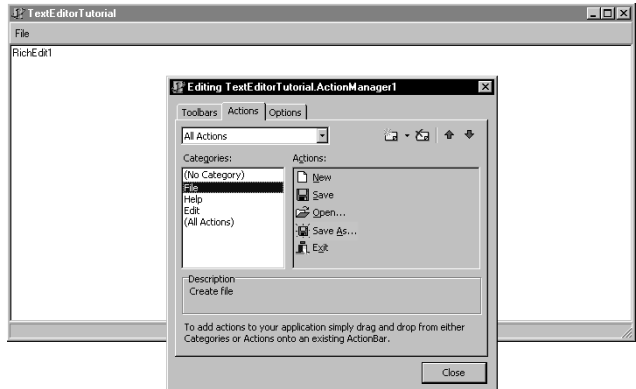
- Open 액션을 선택하고 Action Manager 에디터 툴바의 Move Up 버튼을 클릭하면 File 명령이 New, Open, Save, Save As, Exit 순서로 나열됩니다.

- 3 File 을 메뉴 표시줄로 끌어 놓습니다. File 메뉴와 하위 메뉴 명령이 메뉴 표시줄에 나타납니다.

팁

메뉴 범주를 메뉴 표시줄에 끌어 놓은 다음 메뉴 명령을 배치할 수도 있습니다. 예를 들어, 메뉴 표시줄의 File을 클릭하면 하위 메뉴 명령이 나타나고 Open을 New 위로 끌어 놓은 다음 다시 돌아올 수 있습니다.

Action Manager 에디터에서 File 범주를 선택하여 메뉴 표시줄로 끌어 놓을 때 File 범주와 함께 하위 메뉴 명령을 모두 끌어 놓습니다.



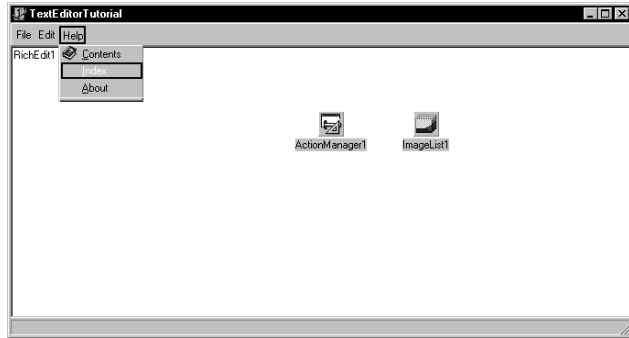
- 4 Action Manager 에디터의 Categories 목록에서 Edit를 메뉴 표시줄의 File 오른쪽으로 끌어 놓습니다.

- 5 Action Manager 에디터의 Categories 목록에서 Help를 메뉴 표시줄의 Edit 오른쪽으로 끌어 놓습니다.

- 6 Help 메뉴를 클릭하여 하위 메뉴 명령을 봅니다. Contents 명령을 Index 명령 위로 끌어 놓습니다.

다음 두 가지 방법으로 하위 메뉴 명령의 위치를 변경할 수 있습니다.

Action Manager 에디터의 액션을 선택하고 Move Up 또는 Move Down 버튼을 클릭합니다. 또는 Help 범주 메뉴 표시줄에 끌어 놓은 후 Contents를 About 위로 끌어 놓습니다.



- 7 Esc를 누르거나 Help 메뉴를 다시 클릭하여 Help 메뉴를 닫습니다.
 - 8 File|Save All을 선택하여 변경 사항을 저장합니다.
- 이제 툴바를 추가하면 명령에 쉽게 액세스할 수 있습니다.

툴바 추가

액션 매니저에 액션을 설정했으므로 메뉴에 사용된 동일한 액션 중 일부를 액션 밴드 툴바에 추가할 수 있습니다. 이 액션 밴드 툴바를 완료하면 Microsoft Office 2000 툴바와 유사하게 보입니다.



- 1 컴포넌트 팔레트의 Additional 페이지에서 *ActionToolBar* 컴포넌트를 더블 클릭하여 폼에 추가합니다.

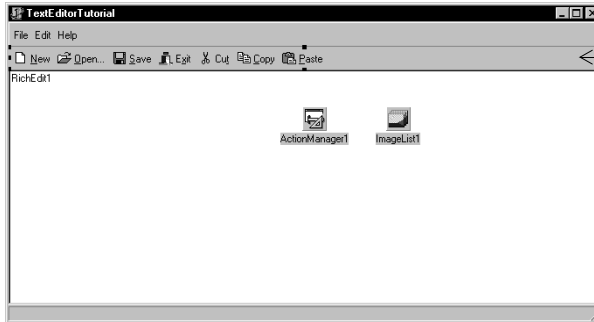
메뉴 표시줄 아래에 빈 툴바가 나타납니다.

팁

Action Manager 에디터를 열고 Toolbars 탭을 클릭한 다음 New 버튼을 클릭하여 액션 밴드 툴바를 추가할 수도 있습니다.

- 2 Action Manager 에디터가 표시되지 않는 경우 Action Manager 에디터를 열고 Categories 목록에서 File을 선택합니다.
- 3 Actions 목록에서 New, Open, Save, Exit를 선택한 다음 이 항목들을 툴바에 끌어 놓습니다. 이 항목들은 각각 자동으로 할당된 이미지를 갖는 버튼으로 나타납니다.
- 4 Action Manager 에디터의 Categories 목록에서 Edit를 선택합니다.

- Actions 목록에서 Cut, Copy, Paste를 선택하고 이 항목들을 투바에 끌어 놓습니다.



ActionToolBar 컴포넌트는 기본적으로 메뉴 표시줄 아래에 추가됩니다. 메뉴 투바를 끌어서 메뉴 표시줄 위나 아래로 이동할 수 있습니다. 버튼을 투바로 끌어오거나 끌어낼 수 있습니다.

잘못된 명령을 투바로 끌어온 경우 이 명령을 제자리로 다시 돌려 놓을 수 있습니다. 또는 Object TreeView에서 항목을 선택하고 delete 키를 클릭할 수 있습니다. 버튼을 다른 버튼의 오른쪽이나 왼쪽으로 끌어 놓아서 버튼들을 배치할 수 있습니다.

- 5 File|Save All을 선택하여 변경 사항을 저장합니다.
- 6 F9 키를 눌러 프로젝트를 컴파일하고 실행합니다.

팁

Debug 투바의 Run 버튼을 클릭하거나 Run|Run을 선택하여 프로젝트를 실행할 수도 있습니다.

프로젝트를 실행할 때 Delphi는 사용자가 디자인한 대로 런타임 창에서 프로그램을 엽니다. 메뉴 및 투바 버튼은 명령 일부가 비활성이어도 작동합니다.

사용자의 텍스트 에디터에는 이미 많은 기능이 들어 있습니다. 텍스트 영역 안에서 입력이 가능합니다. 텍스트 영역에서 텍스트를 선택하면 Cut, Copy, Paste 버튼이 작동해야 합니다. 하지만 명령을 활성화하기 위해서는 해야 할 일이 더 있습니다.

- 7 디자인 모드로 돌아가려면 오른쪽 위 모서리에 있는 X를 클릭합니다.

텍스트 영역 지우기(옵션)

프로그램을 실행하면 RichEdit1이라는 이름이 텍스트 영역에 나타납니다. Strings List Editor를 사용하여 텍스트를 제거할 수 있습니다. 지금 텍스트를 지우지 않으면 마지막 단계에서 메인 폼을 초기화할 때 텍스트를 제거해야 합니다.

텍스트 영역을 지우려면 다음과 같이 합니다.

- 1 메인 폼에서 RichEdit1 컴포넌트를 클릭합니다.
- 2 Object Inspector에서 Lines 속성 옆에 있는 값(TStrings)을 더블 클릭하여 Filter editor를 표시합니다.
- 3 Filter editor에서 제거하려는 텍스트(RichEdit1)를 선택하여 삭제하고 OK를 클릭합니다.
- 4 변경 사항을 저장하고 프로그램을 다시 실행합니다.

이제 메인 폼이 표시되면 텍스트 편집 영역은 지워져 있습니다.

이벤트 핸들러 작성

이제까지 단 한 줄의 코드도 작성하지 않고 애플리케이션을 만들었습니다. Object Inspector를 사용하여 디자인 타임에 속성 값을 설정하면 Delphi의 RAD 환경의 장점을 최대한 살릴 수 있습니다. 이 단원에서는 애플리케이션 실행 중에 사용자 입력에 응답하는 *이벤트 핸들러*라는 프로시저를 작성합니다. 메뉴와 툴바에 있는 항목에 이벤트 핸들러를 연결하므로 항목이 선택되면 애플리케이션은 핸들러의 코드를 실행합니다.

비표준 액션의 경우 이벤트 핸들러를 작성해야 합니다. File|Exit 및 Edit|Paste 명령과 같은 표준 액션에서는 코드에 이벤트가 포함되어 있습니다. 하지만 File|SaveAs 명령과 같은 표준 액션 일부에서는 사용자 고유의 이벤트 핸들러를 작성하여 명령을 사용자 지정할 필요가 있을 것입니다.

모든 메뉴 항목과 툴바 액션은 Action Manager 에디터에서 통합되므로 이벤트 핸들러를 액션 매니저에서 만들 수 있습니다.

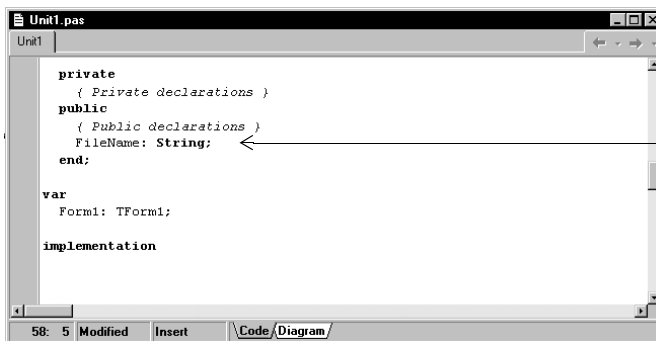
New 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 New 명령에 대한 이벤트 핸들러를 만듭니다.

- 1 View|Units를 선택한 다음 Unit1을 선택하여 Form1과 연결된 코드를 표시합니다.
- 2 이벤트 핸들러에서 사용할 파일 이름을 선언함으로써 파일 이름에 대한 사용자 지정 속성을 추가하여 파일에 전역적으로 액세스할 수 있도록 합니다. Unit1.pas 파일의 앞부분에 TForm1 클래스에 대한 공용 선언 부분을 두고 *{Public declarations}* 아래 줄에 다음과 같이 입력합니다.

```
FileName:String;
```

그러면 화면은 다음과 같이 보입니다.

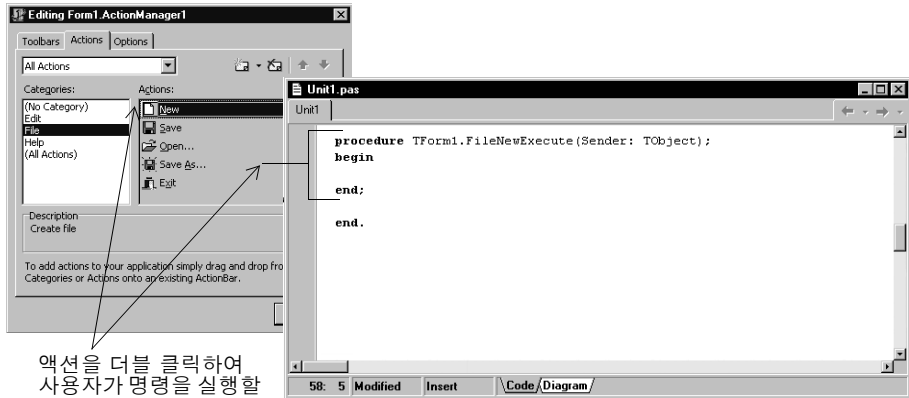


이 줄에서 FileName을 다른 어느 메소드에서도 전역적으로 액세스할 수 있는 문자열로 정의합니다.

- 3 F12 키를 눌러 메인 폼으로 돌아갑니다.

팁 F12 키를 토글하면 폼과 관련 코드 간에 전환할 수 있습니다.

- 4 *ActionManager*를 더블 클릭하여 엽니다.
 - 5 Action Manager 에디터에서 File 범주를 선택한 다음 New 액션을 더블 클릭합니다.
- 팁** Object TreeView에서 File|FileNew 액션을 더블 클릭할 수도 있습니다.
코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

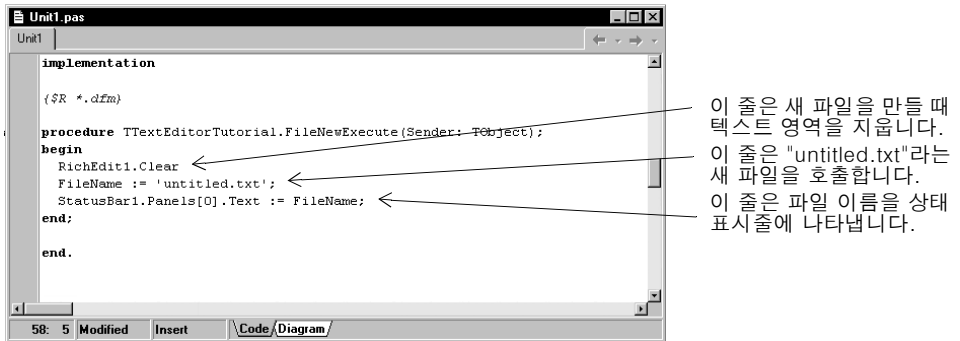


액션을 더블 클릭하여 사용자가 명령을 실행할 때 수행할 액션을 지정할 수 있는 빈 이벤트 핸들러를 만듭니다.

- 6 코드 에디터에서 커서가 있는 위치(begin과 end 사이)에 다음 행을 입력합니다.

```
RichEdit1.Clear;
FileName := 'untitled.txt';
StatusBar1.Panels[0].Text := FileName;
```

그렇게 하면 이벤트 핸들러가 다음과 같이 나타납니다.



이 줄은 새 파일을 만들 때 텍스트 영역을 지웁니다.
이 줄은 "untitled.txt"라는 새 파일을 호출합니다.
이 줄은 파일 이름을 상태 표시줄에 나타냅니다.

작업을 저장하면 File|New 명령에 대한 모든 작업이 끝납니다.

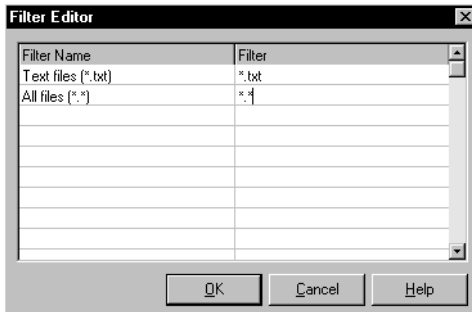
참고

창의 코드 부분의 크기를 조정하면 수평 스크롤을 줄일 수 있습니다.

Open 명령에 대한 이벤트 핸들러 만들기

텍스트 에디터에서 파일을 열려면 표준 Windows Open 대화 상자가 나타나야 합니다. 대화 상자가 자동으로 포함된 Action Manager 에디터에 표준 File|Open 명령이 이미 추가되어 있습니다. 하지만 Open 명령에 대한 이벤트 핸들러를 사용자 지정해야 합니다.

- 1 *F12*를 눌러 메인 폼으로 돌아갑니다(또는 View|Forms를 선택하고 Form1을 선택합니다).
- 2 Action Manager 에디터를 더블 클릭하여 엽니다. File|Open 액션을 선택합니다.
- 3 Object Inspector에서 *Dialog* 속성의 왼쪽에 있는 + 기호를 클릭하여 속성을 확장합니다. Delphi는 기본적으로 대화 상자 이름을 *FileOpen1.OpenDialog*라고 지정합니다. *OpenDialog1*의 *Execute* 메소드를 호출하면 파일을 열기 위한 표준 대화 상자를 호출합니다.
- 4 *FileOpen1.Dialog*의 다음 속성을 설정합니다.
 - *DefaultExt*를 txt로 설정합니다.
 - *Filter* 옆에 있는 텍스트 영역을 더블 클릭하여 Filter editor를 표시합니다. Filter Name 열 아래 첫 번째 행에 text files(*.txt)를 입력합니다. Filter 열에 *.txt를 입력합니다. Filter Name 열 아래의 두 번째 행에 All files(*.*)를 입력하고 Filter 열에는 *.*를 입력합니다. OK를 클릭합니다.

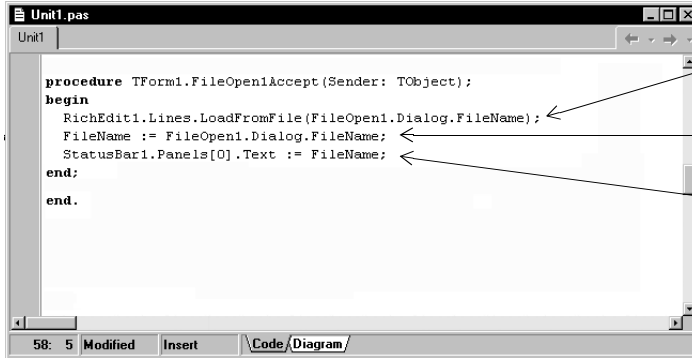


Filter editor를 사용하여 *FileOpen1.Dialog* 및 *FileSaveAs1.Dialog* 액션에 대한 필터를 정의합니다.

- *Title* 뒤에 Open file을 입력합니다. 이러한 단어들은 Open 대화 상자의 상단에 나타납니다.
- 5 Events 탭을 클릭합니다. FileOpen1Accept가 나타나도록 *OnAccept* 이벤트를 더블 클릭합니다.
- 6 코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.
- 7 코드 에디터에서 커서가 있는 위치(**begin**과 **end** 사이)에 다음 행을 입력합니다.

```
RichEdit1.Lines.LoadFromFile(FileOpen1.Dialog.FileName);
FileName := FileOpen1.Dialog.FileName;
StatusBar1.Panels[0].Text := FileName;
```

그러면 FileOpen 이벤트 핸들러가 다음과 같이 나타납니다.



이 줄은 지정된 파일로부터 가져온 텍스트를 삽입합니다.
이 줄은 파일 이름을 Open 대화 상자에서 설정합니다.
이 줄은 파일 이름을 상태 표시줄에 나타냅니다.

이제 File|Open 명령과 Open 대화 상자에 대한 내용이 모두 끝났습니다.

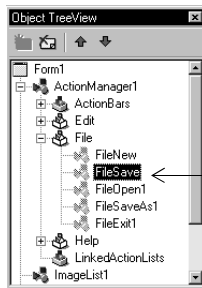
Save 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 Save 명령에 대한 이벤트 핸들러를 만듭니다.

- 1 F12를 눌러 폼을 표시합니다. *ActionManager* 컴포넌트를 더블 클릭하여 엽니다.
- 2 File|Save 액션을 더블 클릭합니다.

코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

팁 Object TreeView에서 File|FileSave 액션을 더블 클릭할 수도 있습니다.



Object TreeView를 사용하여 각 액션의 이벤트 핸들러에 액세스할 수 있습니다.

액션을 더블 클릭하여 코드 에디터를 열고 새 이벤트 핸들러를 작성합니다.

- 3 코드 에디터에서 커서가 있는 위치(**begin**과 **end** 사이)에 다음 행을 입력합니다.

```
if (FileName = 'untitled.txt') then
  FileSaveAs1.Execute
else
  RichEdit1.Lines.SaveToFile(FileName);
```

이 코드는 파일 이름이 지정되지 않았을 경우에 사용자가 이름을 지정할 수 있도록 Save As 대화 상자를 표시합니다. 이름이 있는 파일이면 현재 이름으로 파일을 저장합니다. SaveAs 대화 상자는 4-20 페이지에 있는 Save As 명령에 대한 이벤트 핸들러에서 정의됩니다. *FileSaveAs1BeforeExecute*는 Save As 명령에 대해 자동으로 생성된 이름입니다.

그렇게 하면 이벤트 핸들러가 다음과 같이 나타납니다.

```

Unit1.pas
Unit1
procedure TForm1.FileSaveExecute(Sender: TObject);
begin
  if (FileName = 'untitled.txt') then
    FileSaveAs1.Execute
  else
    RichEdit1.Lines.SaveToFile(FileName);
end;
end.
58: 5 Modified Insert \Code\Diagram/
    
```

이 줄은 파일 이름이 지정되지 않은 경우에 File Save As 대화 상자가 나타난다는 것을 표시합니다.
 그렇지 않은 경우에 파일은 현재 파일 이름으로 저장됩니다.

이제 File|Save 명령에 대한 내용이 모두 끝났습니다.

Save As 명령에 대한 이벤트 핸들러 만들기

SaveDialog의 Execute 메소드를 호출하면 파일 저장을 위한 표준 Windows Save As 대화 상자를 호출합니다. 다음과 같은 방법으로 Save As 명령에 대한 이벤트 핸들러를 만듭니다.

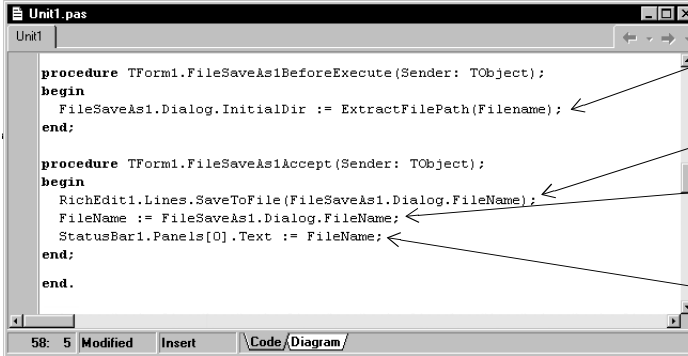
- 1 F12를 눌러 폼을 표시합니다. ActionManager 컴포넌트를 더블 클릭하여 엽니다.
- 2 File|SaveAs 액션을 선택합니다.
- 3 Object Inspector에서 Properties 탭을 클릭하고 FileSaveAs1 대화 상자에 대한 다음 속성을 설정합니다. Delphi는 기본적으로 대화 상자 이름을 FileSaveAs1.Dialog라고 지정합니다.
- 4 Dialog 속성의 왼쪽에 있는 + 기호를 클릭하여 다음 속성을 설정합니다.
 - DefaultExt를 txt로 설정합니다.
 - Filter 옆에 있는 텍스트 영역을 더블 클릭하여 Filter 에디터를 표시합니다. Filter editor에서 파일 형식에 대한 필터를 Open 대화 상자과 같게 지정합니다. Filter Name 열 아래 첫 번째 행에 text files(*.txt)를 입력합니다. Filter 열에 *.txt를 입력합니다. Filter Name 열 아래의 두 번째 행에 All files(*.*)를 입력하고 Filter 열에는 *.*를 입력합니다. OK를 클릭합니다.
 - Title을 Save As로 설정합니다.
- 5 Object Inspector에서 Events 탭을 클릭합니다. BeforeExecute 옆에 있는 텍스트 영역을 더블 클릭하여 FileSaveAs1BeforeExecute를 표시합니다. 코드 에디터가 열리면서 코드 에디터 내부에 커서가 위치하게 됩니다.
- 6 코드 에디터에서 커서가 있는 위치에 다음 행을 입력합니다.

```
FileSaveAs1.Dialog.InitialDir := ExtractFilePath(FileName);
```

- Object Inspector에서 Events 탭이 여전히 표시됩니다. *OnAccept* 이벤트 옆에 있는 텍스트 영역을 더블 클릭하여 *FileSaveAs1Accept*를 표시합니다.
- 코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다. 다음 줄을 입력합니다.

```
RichEdit1.Lines.SaveToFile(FileSaveAs1.Dialog.FileName);
FileName := FileSaveAs1.Dialog.FileName;
StatusBar1.Panels[0].Text := FileName;
```

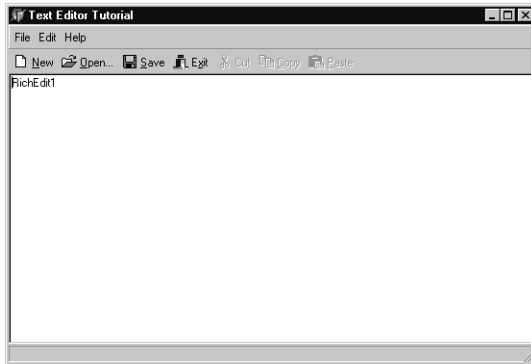
이렇게 하면 FileSaveAs 이벤트 핸들러가 다음과 같이 나타납니다.



이 줄은 기본 디렉토리를 마지막에 액세스한 디렉토리로 설정합니다.
 이 줄은 텍스트를 지정된 파일로 저장합니다.
 이것은 메인 폼의 FileName을 SaveAs 대화 상자에 지정된 이름으로 설정합니다.
 이것은 파일 이름을 상 태 표시줄에 놓습니다.

이제 File|SaveAs 명령에 대한 내용이 모두 끝났습니다.

- File|Save All을 선택하여 프로젝트를 저장합니다.
- 이제까지 작업한 내용이 어떻게 나타나는지 보려면 *F9* 키를 눌러 애플리케이션을 실행합니다.



사용자의 애플리케이션에 모든 기능이 완벽하게 갖 추어졌습니다. 이미지는 이미지 인덱스와 연결된 명령 옆에 나타납니다.
 논비주얼(nonvisual) 컴 폰트는 나타나지 않는다는 것에 유의하십시오.
 메뉴, 툴바, 텍스트 영역 및 상태 표시줄은 모두 폼에 나타납니다.

대부분의 버튼 및 툴바 버튼은 작동하지만 아직 다 끝난 것은 아닙니다.

코드 에디터의 하단에 오류 메시지가 나타나면 오류 메시지를 클릭하여 오류가 발생한 코드의 위치로 이동합니다. 반드시 이 자습서에서 설명한 단계대로 따라야 합니다.

- 디자인 모드로 돌아가려면 오른쪽 위 모서리에 있는 **X**를 클릭합니다.

도움말 파일 생성

애플리케이션 사용 방법에 대해 설명하는 도움말 파일을 생성하는 것이 좋습니다. Delphi는 Windows 도움말 파일 디자인 및 컴파일에 대한 정보가 들어 있는 C:\Project Files\Borland\Delphi6\Help\Tools 디렉토리의 Microsoft Help Workshop을 제공합니다. 예제 텍스트 에디터 애플리케이션에서 Help|Contents 또는 Help|Index를 선택하여 콘텐츠 또는 색인을 갖는 도움말 파일에 액세스할 수 있습니다.

앞에서 컴파일된 도움말 파일의 Contents 탭 또는 Index 탭을 표시하기 위해 액션 매니저에서 HelpContents 및 HelpIndex 액션을 생성했습니다. Help 매개변수에 상수 값을 할당하고 원하는 내용을 표시하는 이벤트 핸들러를 작성해야 합니다.

Help 명령을 사용하려면 Windows 도움말 파일을 생성하고 컴파일해야 합니다. 도움말 파일 생성은 이 자습서에 나오지 않습니다. 하지만 예제 rtf 파일(TextEditor.rtf), 도움말 파일(TextEditor.hlp) 및 콘텐츠 파일(TextEditor.cnt)을 다운로드할 수 있습니다.

- 1 C:\Project Files\Borland\Delphi6\Help 디렉토리에서 D6X1.zip을 엽니다.
- 2 Text Editor 디렉토리에서 .hlp와 .cnt 파일의 압축을 풀고 저장합니다. 기본 디렉토리는 C:\Project Files\Borland\Delphi6\Projects\TextEditor입니다.

참고 프로젝트에서 HLP 또는 CNT 파일(Delphi 도움말 파일 및 연결된 CNT 파일 등)을 사용할 수 있습니다. 애플리케이션에서 이 파일들을 찾으려면 파일 이름을 TextEditor.hlp와 TextEditor.cnt로 지정해야 합니다.

Help Contents 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 Help Contents 명령에 대한 이벤트 핸들러를 만듭니다.

- 1 *ActionManager* 컴포넌트를 더블 클릭하여 엽니다.
- 2 Action Manager 에디터에서 Help 범주를 선택한 다음 HelpContents 액션을 더블 클릭합니다.
코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.
- 3 텍스트 에디터에 커서가 있는 위치 바로 앞, 즉 **begin** 바로 앞에 다음 줄을 입력합니다.

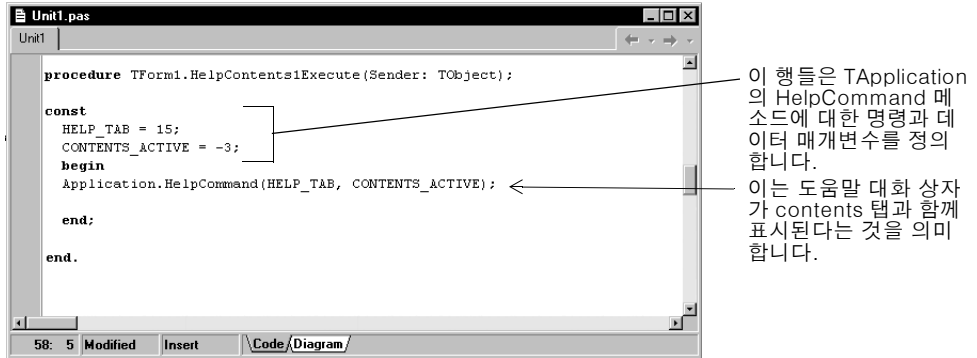
```
const
    HELP_TAB = 15;
    CONTENTS_ACTIVE = -3;
```

begin 바로 뒤에 다음을 입력합니다.

```
Application.HelpCommand(HELP_TAB, CONTENTS_ACTIVE);
```

이 코드는 HelpCommand 매개변수에 상수 값을 할당합니다. HELP_TAB을 15로 설정하면 도움말 대화 상자가 표시되고 CONTENTS_ACTIVE를 -3으로 설정하면 Contents 탭이 표시됩니다.

그렇게 하면 이벤트 핸들러가 다음과 같이 나타납니다.



참고 HelpCommand 이벤트에 대한 도움말을 얻으려면 에디터의 HelpCommand 옆에 커서를 놓고 *F1*을 누릅니다.

이제 Help|Contents 명령에 대한 내용이 모두 끝났습니다.

Help Index 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 Help Index 명령에 대한 이벤트 핸들러를 만듭니다.

- 1 Action Manager 에디터를 닫지 말고 계속 열어두어야 합니다. Action Manager 에디터를 닫았을 경우 폼에서 *ActionManager* 컴포넌트를 더블 클릭합니다.
- 2 Action Manager 에디터에서 Help 범주를 선택한 다음 HelpIndex 액션을 더블 클릭합니다.

코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

- 3 텍스트 에디터에 커서가 있는 위치 바로 앞, 즉 **begin** 바로 앞에 다음과 같이 입력합니다.

```

const
    HELP_TAB = 15;
    INDEX_ACTIVE = -2;
    
```

begin 바로 뒤에 다음을 입력합니다.

```

Application.HelpCommand(HELP_TAB, INDEX_ACTIVE);
    
```

이 코드는 HelpCommand 매개변수에 상수 값을 할당합니다. HELP_TAB을 15로 설정하면 도움말 대화 상자가 표시되고 INDEX_ACTIVE를 -2로 설정하면 Index 탭이 표시됩니다.

그렇게 하면 이벤트 핸들러는 다음과 같습니다.

```

Unit1.pas
Unit
procedure TForm1.HelpIndexExecute(Sender: TObject);
const
  HELP_TAB = 15;
  INDEX_ACTIVE = -2;
begin
  Application.HelpCommand(HELP_TAB, INDEX_ACTIVE);
end;
end.

```

이 행들은 TApplication의 HelpCommand 메소드에 대한 명령과 데이터 매개변수를 정의합니다.

← 이는 도움말 대화 상자가 index 탭과 함께 표시된다는 것을 의미합니다.

이제 Help|Index 명령에 대한 내용이 모두 끝났습니다.

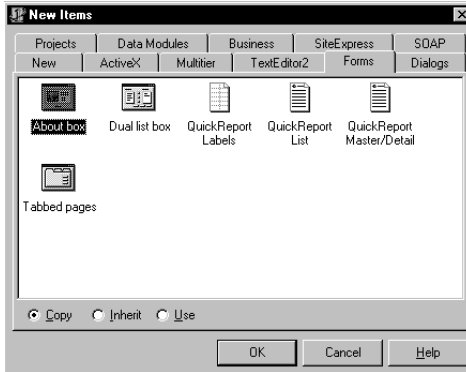
About 상자 만들기

많은 애플리케이션에는 이름, 버전, 로고와 같은 제품에 대한 정보를 나타내는 About 상자가 있고 저작권 정보를 비롯한 기타 다른 법률에 관한 정보가 있습니다.

액션 매니저에서 Help About 명령을 이미 설정했습니다.

다음과 같은 방법으로 About 상자를 추가합니다.

- 1 File|New|Other를 선택하여 New Items 대화 상자를 선택하고 Forms 탭을 클릭합니다.
- 2 Forms 탭에서 About Box를 더블 클릭합니다.



About 상자는 Delphi에 미리 디자인된 여러 개의 품 중 하나입니다.

Copy가 기본적으로 선택되어 있으면 About 상자의 복사본이 프로젝트에 추가됩니다.

About 상자를 쉽게 만들기 위한 새 품이 만들어집니다.

- 3 품 자체를 선택(그리드 부분 클릭)하고 Object Inspector에서 *Caption* 속성을 About Text Editor로 변경합니다.

4 Object Inspector에서 Properties 탭을 클릭하고 다음 *TLabel* 항목에 대한 *Caption* 속성을 변경합니다.

- Product Name을 Text Editor로 변경합니다.
- Version 뒤에 1.0을 추가합니다.
- Copyright 뒤에 연도를 추가합니다.



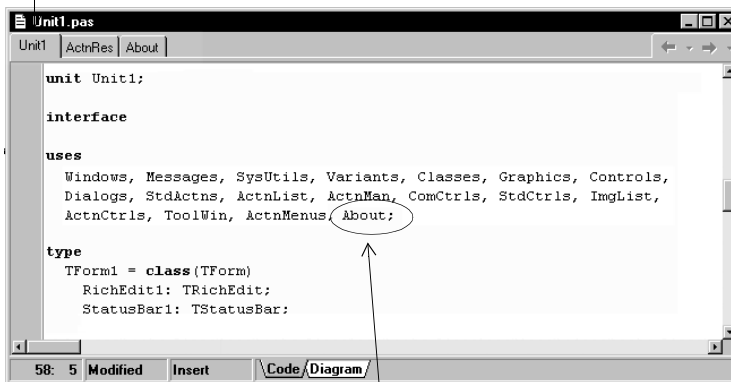
Object Repository에는 애플리케이션을 설명하고자 하는 대로 수정할 수 있는 표준 About 상자가 들어 있습니다.

5 File|Save As를 선택하여 About 상자 폼을 About.pas로 저장합니다.

6 Delphi 코드 에디터에 세 개의 유닛 파일, 즉 Unit1, ActnRes와 About이 나타나야 합니다. Unit1 탭을 클릭하여 Unit1.pas를 표시합니다. ActnRes 유닛이 필요하지는 않아도 그곳에 남겨둘 수 있습니다.

7 Unit1 탭을 클릭하고 **uses** 절에 포함된 유닛 목록에 About을 입력하여 새로운 About 유닛을 추가합니다.

탭을 클릭하여 유닛과 관련된 파일을 표시합니다. 프로젝트에 대한 액션을 하는 동안 다른 파일을 열면 코드 에디터에 추가 탭이 나타납니다.



애플리케이션에서 새 폼을 만들 때 메인 폼의 **uses** 절에 새 폼을 추가해야 합니다. 여기서 About 상자를 추가하는 것입니다.

8 *F12*를 눌러 디자인 모드로 돌아갑니다. *ActionManager* 컴포넌트를 더블 클릭하여 엽니다.

- 9 HelpAbout 액션을 더블 클릭하여 이벤트 핸들러를 생성합니다. 코드 에디터에서 커서가 있는 위치에 다음 행을 입력합니다.

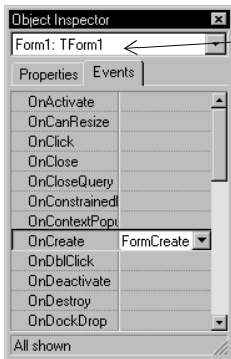
```
AboutBox.ShowModal;
```

이 코드는 사용자가 Help|About을 클릭할 때 About 상자를 엽니다. ShowModal은 모드 상태, 즉 런타임 상태로 폼을 열기 때문에 사용자는 폼이 닫힐 때까지 아무 것도 할 수 없습니다.

애플리케이션 완료

이제 애플리케이션이 거의 끝났습니다. 그러나 메인 폼의 일부 항목을 지정해야 합니다. 다음과 같은 방법으로 애플리케이션을 완료합니다.

- 1 F12를 눌러 메인 폼으로 돌아갑니다.
- 2 포커스가 컴포넌트가 아닌 폼 자체에 있는 것을 확인하십시오. Object Inspector의 상단에 있는 리스트 박스에 Form1:TForm1이라고 표시되어야 합니다.(그렇지 않은 경우 드롭다운 목록에서 Form1을 선택합니다.)
- 3 Events 탭을 클릭하고 OnCreate 이벤트 옆의 드롭다운 목록에서 FormCreate를 선택하여 폼이 생성될 때 (즉, 애플리케이션을 열 때) 발생하는 내용을 나타내는 이벤트 핸들러를 생성합니다.



여기서 포커스가 메인 폼에 있는지 확인하십시오. 그렇지 않은 경우 드롭다운 목록에서 Form1을 선택합니다.

여기를 더블 클릭하면 폼의 OnCreate 이벤트에 대한 이벤트 핸들러를 만듭니다.

- 4 코드 에디터의 커서가 있는 위치에서 다음을 입력합니다.

```
Application.HelpFile := ExtractFilePath(Application.ExeName) + 'TextEditor.hlp';  
FileNew.Execute
```

이 코드는 *FileName*의 값을 *untitled.txt*로 설정하고, 파일 이름을 상태 표시줄에 놓으며, 텍스트 편집 영역을 지움으로써 도움말 파일에 연결된 애플리케이션을 초기화합니다.

```

Unit1.pas
Unit1
-----
procedure TForm1.FormCreate(Sender: TObject);
begin
  Application.HelpFile := ExtractFilePath(Application.ExeName) + 'TextEditor.hlp';
  FileNew.Execute
end;

end.
-----
58: 5 Modified Insert \Code/Diagram/
    
```

← 이 줄은 애플리케이션을 초기화합니다.
 ← 이 줄은 4-17 페이지의 File|New 액션에 대해 처음 작성했던 FileNew.Execute 프로시저를 호출합니다.

- 5 File|SaveAll을 선택하여 변경 내용을 저장합니다.
- 6 애플리케이션을 실행하려면 *F9* 키를 누릅니다.
 축하합니다! 이제 모두 끝났습니다.

애플리케이션 완료

5

데스크탑 사용자 지정

이 장에서는 Delphi의 IDE 환경에서 툴을 사용자 지정하는 방법에 대해 설명합니다.

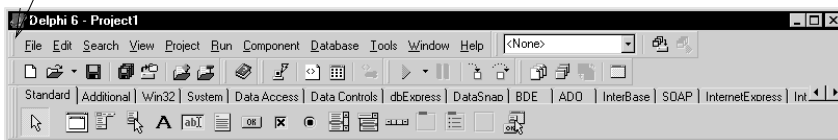
작업 영역 구성

IDE는 개발을 지원하는 많은 도구들을 제공하므로 메뉴 및 툴바 재정렬, 툴 윈도우 조합, 새로운 데스크탑의 저장 등을 포함하여 작업 영역을 최대한 편리하게 재구성할 수 있습니다.

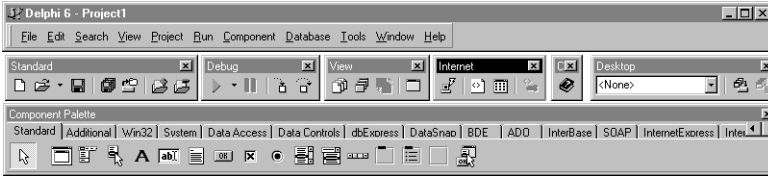
메뉴와 툴바 배열

메인 윈도우에서는 메뉴, 툴바, 컴포넌트 팔레트 등을 각 왼쪽 그래버를 클릭하고 다른 위치로 끌어 놓아서 재구성할 수 있습니다.

메인 윈도우 내에서 툴바와 메뉴를 이동할 수 있습니다. 각 툴바의 그래버(왼쪽에 있는 이중 막대)를 끌어서 이동하십시오.

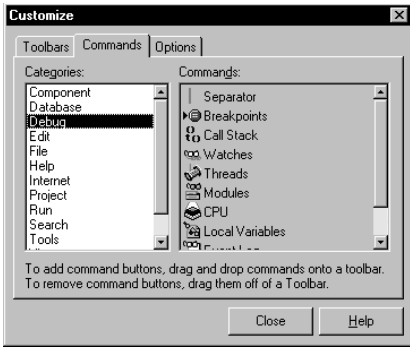


메인 윈도우의 일부를 분리하여 화면의 다른 곳에 놓거나 데스크탑에서 보이지 않게 할 수 있습니다. 이것은 듀얼 모니터를 설치한 경우 유용하게 사용할 수 있습니다.



다르게 구성된 메인 윈도우입니다.

View|Toolbars|Customize를 선택하면 툴바에 툴을 추가하거나 삭제할 수 있습니다. Commands 페이지를 클릭하고 범주를 선택한 후 명령을 선택하고 툴을 두려는 위치의 툴바로 끌어 놓으십시오.



Commands 페이지에서 명령을 선택하고 툴바로 끌어 놓으십시오.

Options 페이지에서 Show tooltips를 클릭하여 컴포넌트와 툴바 아이콘의 힌트가 나타나도록 합니다.

자세한 내용은...

온라인 도움말 색인에서 "toolbars, customizing"을 참조하십시오.

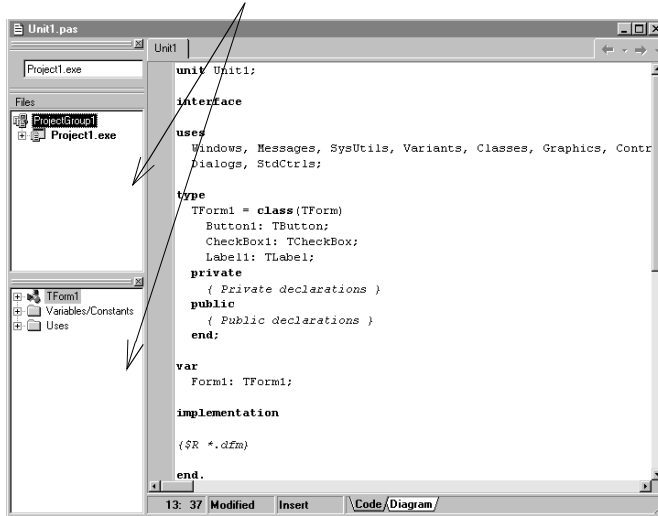
틀 윈도우 도킹

틀 윈도우는 개별적으로 열고 닫을 수 있으며 원하는 대로 데스크탑에 배열할 수도 있습니다. 또한 쉽게 관리할 수 있도록 여러 창들을 서로 도킹할 수 있습니다. 도킹, 즉 창들을 서로 붙여서 함께 움직이는 방식을 통해 틀에 빠르게 액세스할 수 있으며 화면을 효율적으로 사용할 수 있습니다.

View 메뉴에서 툴 윈도우를 가져온 다음 다른 창에 직접 도킹할 수 있습니다. 예를 들어, Delphi를 기본 구성으로 처음 열 때 코드 탐색기는 코드 에디터의 왼쪽에 도킹됩니다. Project Manager를 처음 두 개 창에 추가하면 세 개의 도킹 창을 만들 수 있습니다.

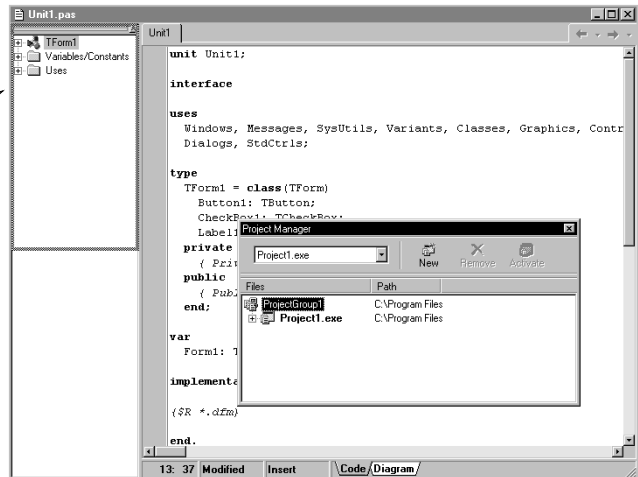
Project Manager 및 코드 탐색기가 이와 같이 코드 에디터에 도킹되어 있습니다.

오른쪽에서와 같이 그래버를 사용하여 5-4 페이지처럼 탭을 결합하여 창들을 결합하거나 "도킹"할 수 있습니다.

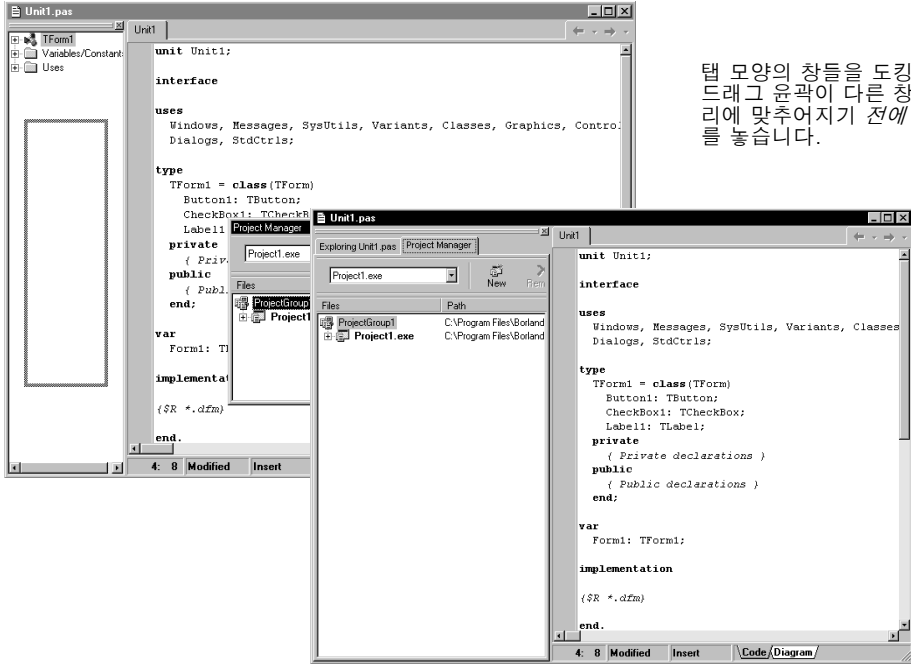


창을 도킹하려면 제목 표시줄을 클릭하여 그 창을 다른 창으로 끌어 놓습니다. 드래그 윤곽이 직사각형으로 좁아지고 모서리와 맞추어지면 마우스를 놓습니다. 그러면 두 창이 함께 도킹됩니다.

그래버를 사용하여 창을 도킹시키려면 드래그 윤곽이 창의 모서리와 맞았을 때 마우스를 놓습니다.



도구들을 도킹시켜 탭 모양의 창을 만들 수도 있습니다.



탭 모양의 창들을 도킹하려면 드래그 윗쪽이 다른 창의 모서리에 맞추어지기 전에 마우스를 놓습니다.

창의 도킹을 해제하려면 그래버나 탭을 더블 클릭합니다.

자동 도킹을 선택 해제하려면 창을 화면 주위로 이동시키면서 *Ctrl*/키를 누르거나 *Tools|Environment Options*를 선택하고 *Designer* 페이지를 클릭한 후 *Auto drag docking* 체크 박스의 선택을 해제합니다.

자세한 내용은...

온라인 도움말 색인의 "docking"을 참조하십시오.

데스크탑 레이아웃 저장

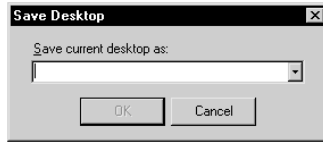
데스크탑 레이아웃을 사용자 지정하고 저장할 수 있습니다. IDE의 데스크탑 툴바에는 사용 가능한 데스크탑 레이아웃의 선택 목록과 아이콘 두 개가 있어서 데스크탑을 사용자 지정하기가 쉽습니다.



명명된 데스크탑 설정이 여기에 나열되어 있습니다.

특정한 창들을 표시, 크기 조정, 도킹하고 그 창들을 화면에서 원하는 위치에 놓는 등 데스크탑을 원하는 대로 조정합니다.

Desktops 툴바에서 Save current desktop 아이콘을 클릭하거나 View|Desktops|Save Desktop을 선택하고 새로운 레이아웃 이름을 입력합니다.



저장하려는 데스크탑 레이아웃을 입력하고 OK를 클릭합니다.

자세한 내용은...

온라인 도움말 색인의 "desktop layout"을 참조하십시오.

컴포넌트 팔레트 사용자 지정

기본 구성에서 컴포넌트 팔레트는 기능적으로 구성된 여러 유용한 VCL 또는 CLX 객체를 탭 모양의 페이지에 표시합니다. 컴포넌트 팔레트를 다음과 같이 사용자 지정할 수 있습니다.

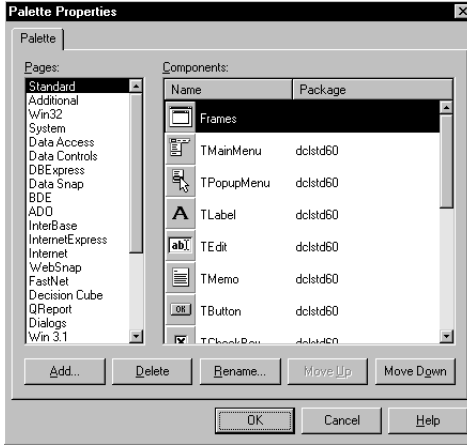
- 컴포넌트 숨기기 또는 재정렬을 할 수 있습니다.
- 페이지 추가, 삭제, 재정렬 또는 이름 재지정을 할 수 있습니다.
- 컴포넌트 템플릿을 작성한 다음 팔레트에 추가할 수 있습니다.
- 새로운 컴포넌트를 설치할 수 있습니다.

컴포넌트 팔레트 배열

페이지 추가, 삭제, 재정렬 및 페이지 이름 재지정 또는 컴포넌트를 숨기거나 재정렬하려면 Palette Properties 대화 상자를 사용합니다. 이 대화 상자를 다음과 같은 여러 가지 방법으로 열 수 있습니다.

- Component|Configure Palette를 선택합니다.
- Tools|Environment Options를 선택하고 Palette 탭을 클릭합니다.

- 컴포넌트 팔레트를 마우스 오른쪽 버튼으로 클릭하고 Properties를 선택합니다.



팔레트를 재정렬하고 새로운 페이지를 추가합니다.

자세한 내용은...

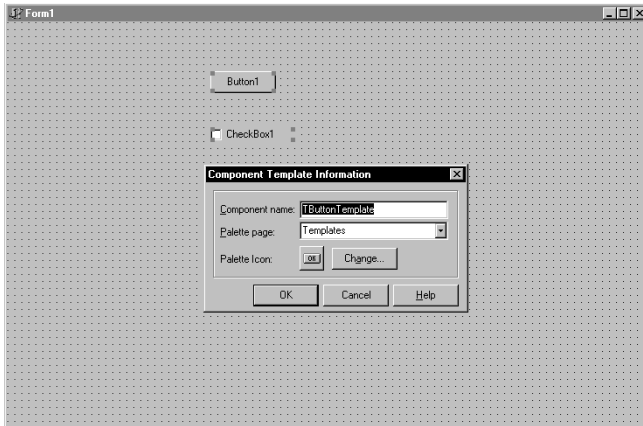
Palette Properties 대화 상자에서 Help 버튼을 클릭하십시오.

컴포넌트 템플릿 만들기

컴포넌트 템플릿은 단일 작업의 폼에 추가하는 컴포넌트 그룹입니다. 템플릿을 사용하면 하나의 폼에서 컴포넌트를 구성한 다음 컴포넌트 배열, 기본 속성, 이벤트 핸들러를 컴포넌트 팔레트에 저장하여 다른 폼에서 재사용할 수 있습니다.

컴포넌트 템플릿을 만들려면 하나 이상의 컴포넌트를 폼에 배열하고 그 속성들을 Object Inspector에서 설정한 다음 그 위로 마우스를 끌어서 모든 컴포넌트를 선택합니다. 그리고 Component|Create Component Template을 선택합니다. Component Template Information 대화 상자가 열리면 템플릿 이름, 템플릿을 표시하려는 팔레트 페이지, 팔레트에서 템플릿을 나타낼 아이콘 등을 선택합니다.

템플릿을 폼에 놓은 다음에는 컴포넌트들을 각기 따로 다시 배치하고, 그 속성들을 재설정하고, 마치 다른 작업에서 각 컴포넌트를 놓은 것처럼 컴포넌트들의 이벤트 핸들러를 만들거나 수정할 수 있습니다.



자세한 내용은...

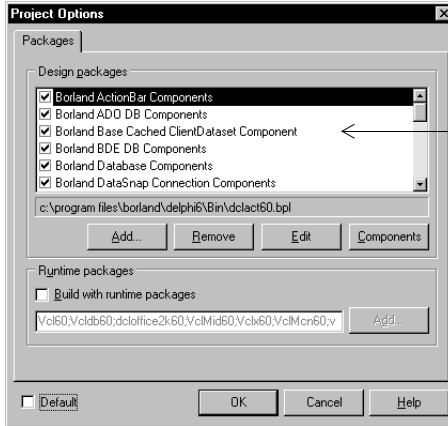
온라인 도움말 색인에서 "template, component"를 참조하십시오.

컴포넌트 패키지 설치

사용자 지정 컴포넌트를 만들었거나 공급 업체로부터 구입했다면 컴포넌트를 컴포넌트 팔레트에 설치하기 전에 *패키지*에 컴파일해야 합니다.

패키지는 Delphi 애플리케이션들, IDE 또는 양쪽 모두에서 공유할 수 있는 코드를 포함하는 특수 DLL입니다. *런타임 패키지*는 사용자가 애플리케이션을 실행할 때 사용되는 기능을 제공합니다. *디자인 타임 패키지*는 IDE에 컴포넌트를 설치하는 데 사용됩니다. Delphi 패키지는 .bpl 확장자를 갖습니다.

협력 업체의 컴포넌트가 이미 패키지에 컴파일된 경우에는 해당 협력 업체의 지침을 따르거나 Component|Install Packages를 선택합니다.



이러한 컴포넌트들은 Delphi에 이미 설치되어 있습니다. 협력 업체의 새 컴포넌트를 설치할 경우 패키지가 이 목록에 나타납니다. 패키지에 들어 있는 컴포넌트들을 알아보려면 Components를 클릭합니다.

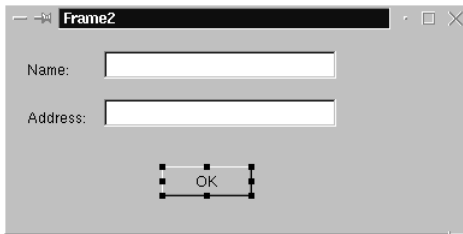
자세한 내용은...

온라인 도움말 색인의 "installing components"와 "packages"를 참조하십시오.

프레임 사용

프레임 (TFrame)은 폼과 마찬가지로 재사용할 컴포넌트의 컨테이너입니다. 프레임은 폼보다는 사용자 지정 컴포넌트와 더 유사합니다. 프레임은 쉽게 다시 사용하기 위해 컴포넌트 팔레트에 저장할 수 있으며 폼, 다른 프레임 또는 다른 컨테이너 객체 내에 중첩시킬 수 있습니다. 프레임을 만들고 저장한 후 프레임은 계속 유닛으로 기능하고 프레임에 들어 있는 컴포넌트(다른 프레임 포함)의 변경 내용을 상속합니다. 다른 프레임이나 폼에 포함된 프레임은 자신이 파생된 프레임의 변경 내용을 계속 상속합니다.

새 프레임을 열려면 File|New|Frame을 선택합니다.



프레임에 필요한 것이라면 비주얼 컴포넌트나 년비주얼 컴포넌트 어떤 것도 추가할 수 있습니다. 새로운 유닛이 코드 에디터에 자동으로 추가됩니다.

자세한 내용은...

도움말 색인의 "frames"와 "TFrame"을 참조하십시오.

ActiveX 컨트롤 추가

ActiveX 컨트롤을 컴포넌트 팔레트에 추가하여 Delphi 프로젝트에서 사용할 수 있습니다. Component|Import ActiveX Control을 선택하여 Import ActiveX 대화 상자를 엽니다. 여기에서 새로운 ActiveX 컨트롤을 등록하거나 이미 등록된 컨트롤을 선택하여 IDE에 설치할 수 있습니다. ActiveX 컨트롤을 설치하면 Delphi는 "랩퍼(wrapper)" 유닛 파일을 만들고 컴파일합니다.

자세한 내용은...

Component|Import ActiveX Control을 선택하고 Help 버튼을 클릭합니다.

프로젝트 옵션 설정

프로젝트 디렉토리를 관리하고 프로젝트에 대한 폼, 애플리케이션, 컴파일러, 링커 등을 지정해야 할 경우에는 Project|Options를 선택합니다. Project Options 대화 상자를 변경하면 변경된 내용이 현재 프로젝트에만 영향을 미치지만 선택한 것을 새 프로젝트에 대한 기본 설정으로 저장할 수도 있습니다.

기본 프로젝트 옵션 설정

선택한 내용을 새 프로젝트에 대한 기본 설정으로 저장하려면 Project Options 대화 상자의 왼쪽 아래 모서리에 있는 Default에 선택 표시를 합니다. Default에 선택 표시를 하면 대화 상자의 현재 설정을 Delphi6\Bin 디렉토리에 있는 Defproj.dof 옵션 파일에 기록합니다. Delphi 본래의 기본 설정을 복구하려면 Defproj.dof 파일을 삭제하거나 이름을 재지정합니다.

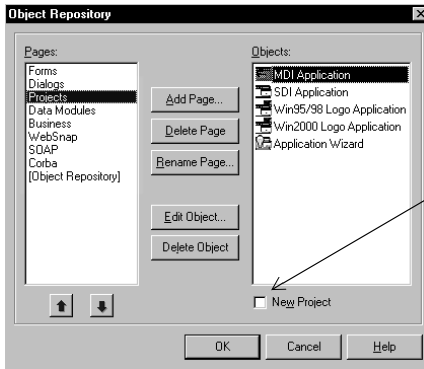
자세한 내용은...

온라인 도움말 색인의 "Project Options dialog box"를 참조하십시오.

프로젝트와 폼 템플릿을 기본값으로 지정

File|New|Application을 선택하면 Delphi에서는 프로젝트 **템플릿**을 기본 프로젝트로 지정하지 않는 한, 빈 폼을 가진 새로운 표준 애플리케이션을 생성합니다. Project|Add to Repository를 선택함으로써 자신의 프로젝트를 Projects 페이지에 있는 Object Repository에서 템플릿으로 저장할 수 있습니다(5-10 페이지의 "Object Repository에 템플릿 추가" 참조). 또는 Object Repository에서 Delphi의 기존 프로젝트 템플릿 중 하나를 선택할 수 있습니다(2-6 페이지의 "Object Repository" 참조).

프로젝트 템플릿을 기본으로 지정하려면 Tools|Repository 를 선택합니다. Object Repository 대화 상자에서 Pages 아래의 Projects를 선택합니다. Projects 페이지에서 프로젝트를 템플릿으로 저장하면 Objects 목록에 나타납니다. 템플릿 이름을 선택하고 New Project를 선택한 후 OK를 클릭합니다.



Object Repository 페이지는 프로젝트 템플릿만 들어 있는 페이지 또는 폼 템플릿만 들어 있는 페이지 또는 두 템플릿이 조합된 페이지가 들어 있습니다.

프로젝트 템플릿을 기본으로 설정하려면 Objects 목록에서 항목을 선택한 후 New Project를 선택합니다.

폼 템플릿을 기본으로 설정하려면 Objects 목록에서 항목을 선택한 후 New Form 또는 Main Form을 선택합니다.

일단 프로젝트 템플릿을 기본으로 지정하면 Delphi는 File|New|Application을 선택할 때마다 프로젝트 템플릿을 자동으로 엽니다.

기본 프로젝트를 지정하는 것과 같은 방법으로 Object Repository의 기존 폼 템플릿 목록에서 기본 메인 폼과 기본 새 폼을 지정할 수 있습니다. 기본 새 폼은 열린 프로젝트에 폼을 추가하기 위해 File|New Form을 선택할 때 만들어지는 폼입니다. 기본 메인 폼은 새 애플리케이션을 열 때 만들어지는 폼입니다. 기본 폼을 지정하지 않을 경우 Delphi는 빈 폼을 사용합니다.

File|New|Other를 선택하고 New Items 대화 상자에서 다른 템플릿을 선택하면 기본 프로젝트 또는 기본 폼을 일시적으로 무시할 수 있습니다.

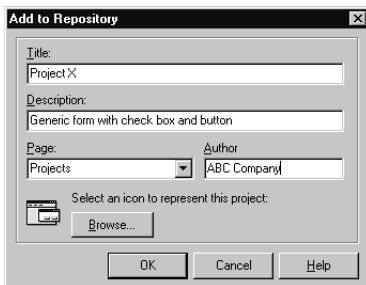
자세한 내용은...

온라인 도움말 색인의 "templates, adding to Object Repository", "projects, specifying default", "forms, specifying default"를 참조하십시오.

Object Repository에 템플릿 추가

자신이 만든 객체를 템플릿 형식으로 Object Repository에 추가하여 다시 사용하거나 네트워크에서 다른 개발자와 공유할 수 있습니다. 객체를 재사용하면 일반 사용자 인터페이스 및 기능을 가진 애플리케이션 제품군을 만들어 개발 시간을 단축하고 품질을 개선할 수 있습니다.

예를 들어, 프로젝트를 템플릿 형식으로 Repository에 추가하려면 먼저 프로젝트를 저장하고 Project|Add To Repository를 선택한 다음 Add to Repository 대화 상자를 완료합니다.



제목, 설명, 작성자를 입력합니다. Page 리스트 박스에서 Projects를 선택하면 프로젝트는 Repository의 Projects 탭 모양의 페이지에 나타납니다.

다음에 New Items 대화 상자를 열면 프로젝트 템플릿은 Projects 페이지나 템플릿을 저장한 페이지에 나타납니다. Delphi를 열 때마다 특정 템플릿을 기본으로 설정하려면 5-9 페이지의 "프로젝트와 폼 템플릿을 기본값으로 지정"을 참조하십시오.

자세한 내용은...

온라인 도움말 색인의 "templates, adding to Object Repository"를 참조하십시오.

툴 환경 설정

폼 디자이너, Object Inspector, 코드 탐색기와 같은 IDE의 외관 및 동작을 제어할 수 있습니다. 이러한 설정은 현재 프로젝트뿐만 아니라 나중에 열고 컴파일할 프로젝트에도 영향을 미칩니다. 모든 프로젝트에 대한 전체 IDE 설정을 변경하려면 Tools|Environment Options를 선택합니다.

자세한 내용은...

온라인 도움말 색인의 "Environment Options dialog box"를 참조하거나 Environment Options 대화 상자의 모든 페이지에 있는 Help 버튼을 클릭합니다.

폼 디자이너 사용자 지정

Tools|Environment Options 대화 상자의 Designer 페이지 설정은 폼 디자이너에 영향을 줍니다. 예를 들면, 컴포넌트들을 가장 가까운 그리드 라인(grid line)에 정렬하는 "snap to grid" 기능을 사용 가능 또는 사용 불가능으로 설정할 수 있습니다. 또한 폼에 놓은 논비주얼(nonvisual) 컴포넌트의 이름이나 캡션을 표시하거나 숨길 수 있습니다.

자세한 내용은...

Environment Options 대화 상자에서 Designer 페이지를 클릭한 후 Help 버튼을 클릭합니다.

코드 에디터 사용자 지정

곧바로 사용자 지정할 수 있는 툴은 코드 에디터입니다. Tools|Editor Options 대화 상자의 여러 페이지에는 코드 편집 방법에 대한 설정이 있습니다. 예를 들면, 키스트로크 매핑, 글꼴, 여백 너비, 색상, 구문 강조, 탭, 들여쓰기 스타일 등을 선택할 수 있습니다.

또한 Editor Options의 Code Insight에 있는 편집기 안에서 사용할 수 있는 Code Insight 툴을 구성할 수 있습니다. 이러한 도구들에 대해 알아보려면 2-7 페이지의 "Code Insight"를 참조하십시오.

자세한 내용은...

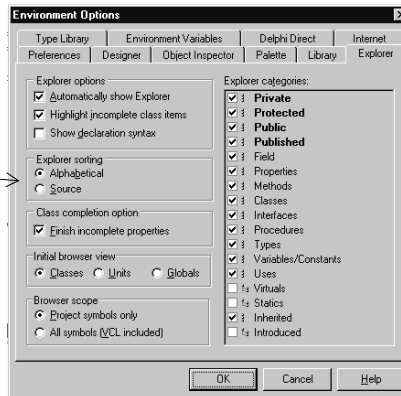
Editor Options 대화 상자에서 General, Display, Key Mappings, Color, Code Insight 페이지 등에 있는 Help 버튼을 클릭합니다.

코드 탐색기 사용자 지정

Delphi를 시작할 때 코드 탐색기(2-11 페이지의 "코드 탐색기"에서 설명)가 자동으로 열립니다. 코드 탐색기가 자동으로 열리지 않게 하려면 Tools|Environment Options를 선택하고 Explorer 탭을 클릭한 다음 Automatically show Explorer 선택을 해제합니다.

코드 탐색기에서 마우스 오른쪽 버튼을 클릭한 후 Properties를 선택하고 Explorer 범주에서 체크 박스를 선택 또는 선택 해제하여 코드 탐색기의 콘텐츠가 코드 탐색기 내에서 그룹화되는 방법을 변경할 수 있습니다. 범주를 선택하면 해당 범주에 있는 요소가 단일 노드에서 그룹화되며 범주의 선택을 해제하면 범주의 각 요소들이 별도의 다이어그램에 표시됩니다. 예를 들어, Published 범주를 선택 해제하면 Published 폴더는 없었지만 그 안에 있는 항목은 없어지지 않습니다.

코드 탐색기에 있는 모든 소스 요소를 알파벳순으로 또는 소스 파일순으로 선언된 순으로 정렬할 수 있습니다.



코드 탐색기에 있는 각 유형의 소스 요소에 대해 폴더를 표시하려면 Explorer 범주를 선택합니다.

자세한 내용은...

온라인 도움말 색인의 "Code Explorer, Environment options"를 참조하십시오.

색인

가

개발자 지원 1-4
객체, 정의 3-5
기본
 프로젝트 옵션 5-9
 프로젝트와 폼 템플릿 5-9
기술 지원 1-4

나

뉴스그룹 1-4

다

대화 상자, Object Repository 2-6
데스크탑
 구성 5-1 ~ 5-5
 레이아웃 저장 5-4
데이터 모듈
 생성 2-6
 추가 3-2
데이터 사전 3-11
데이터베이스 데스크탑 3-11
데이터베이스 애플리케이션, 생성 3-10
데이터베이스 탐색기 3-11
도움말 툴팁 4-4
도움말 파일, 애플리케이션에 추가 4-22
도움말, F1 1-2
디자인 타임 뷰, 폼 닫기 4-3

라

리소스 파일(.res) 4-2

마

마법사, 찾기 2-6
마우스 오른쪽 버튼 메뉴 2-3
메뉴
 구성 2-3, 5-1
 애플리케이션에 추가 4-12
 컨텍스트 2-3
 Delphi 2-3
메뉴 컴포넌트 MainMenu 컴포넌트 참조
메시지, 오류 4-21
메인 폼, 정의 5-10
문자 집합, 확장 3-8

바

번역 도구 3-8
부모 자식 관계 2-5
비트맵, 애플리케이션에 추가 4-10

사

사용자 인터페이스, 만들기 3-2, 4-2, 4-3
사용자 지정
 컴포넌트 팔레트 2-3
 코드 에디터 5-12
 코드 탐색기 5-12
 폼 디자이너 5-11
 IDE 5-1 ~ 5-12
사용자 지정 컴포넌트 설치 5-7
새 폼, 정의 5-10
새로운 기능 1-2
설명서 도움말 시스템
설명서, 주문 1-3
소스 코드
 작성 시 도움말 2-7 ~ 2-8
 파일 4-1
속성 설정 3-3, 4-2, 4-8, 4-9, 4-10
속성, 설정 3-3, 4-2, 4-8, 4-9
실행 파일, 배포 3-8

아

애플리케이션
 국제화 3-8
 데이터베이스 3-10
 배포 3-8
 생성 3-1, 3-9
 웹 서버 3-9
 컴파일 및 디버깅 3-6, 4-15
애플리케이션 국제화 3-8
애플리케이션 배포 3-8
애플리케이션 실행 3-6, 4-15
애플리케이션 지역화 3-8
애플리케이션 컴파일 3-6
액션 밴드(action bands) 4-12
에디터 코드 에디터 참조
예제 프로그램 4-1 ~ 4-27
오류 메시지 4-21
온라인 도움말 파일 1-2
옵션, 프로젝트 설정 5-9
웹 사이트, Borland 1-4
웹 서버 애플리케이션, 생성 3-9
유닛 파일 4-1
이미지, 애플리케이션에 추가 4-10
이벤트 핸들러
 생성 4-21
 작성 4-16
 정의 3-5

자

자습서 4-1 ~ 4-27

작업, 애플리케이션에 추가 4-7, 4-9
저장
 데스크탑 레이아웃 5-4
 프로젝트 4-2
전역 기호 2-13
정보, 찾기 1-1
지원 서비스 1-4

차

창 도킹 5-2 ~ 5-4
창, 결합 5-2

카

컨텍스트 메뉴, 액세스 2-3
컨트롤, 폼에 추가 3-2, 4-3
코드
 보기 및 편집 2-7 ~ 2-12
 이벤트 핸들러 3-5
 작성 3-5
 작성 시 도움말 2-7 ~ 2-8
코드 에디터
 다른 창과 결합 5-2
 사용 2-7 ~ 2-9
 사용자 지정 5-12
코드 탐색기
 사용 2-11
 사용자 지정 5-12
컴포넌트
 사용자 지정 3-11, 5-6
 사용자 지정 컴포넌트 생성 3-11
 설치 3-11, 5-7
 속성 설정 3-3, 4-2
 정의 4-3
 컴포넌트 팔레트에 추가 5-5
 컴포넌트 팔레트에서 배열 5-5
 폼에 추가 3-2, 4-4
컴포넌트 팔레트
 사용 3-2
 사용자 지정 5-5 ~ 5-8
 사용자 지정 컴포넌트 추가 3-11
 정의 2-4
 페이지 추가 5-5
컴포넌트 팔레트, 만들기 5-6
크로스 플랫폼용 Borland 컴포넌트 라이브러리
(CLX) 3-5
클래스 라이브러리 3-5
클래스, 정의 4-4
키스트로크 매핑 5-12

타

타입 라이브러리, 정의 3-12
탭 모양의 창, 도킹 5-4
텍스트 에디터 자습서 4-1 ~ 4-27
템플릿

기본값으로 지정 5-9
Object Repository에 추가 5-10
통합 개발 환경 (IDE)
 둘러 보기 2-1
 사용자 지정 5-1 ~ 5-12
통합 디버거 3-6
툴 윈도우, 도킹 5-2
툴바 2-3
 구성 5-1
 애플리케이션에 추가 4-14
 컴포넌트 추가 및 삭제 5-2
툴팁 4-4

파

파일
 리소스 4-2
 유닛 4-1
 저장 4-2
 폼 2-11, 4-1
 프로젝트 4-1
패키지 5-7
폼
 기본값으로 지정 5-10
 닫기 4-3
 메인 4-2, 5-10
 찾기 2-6
 컴포넌트 추가 3-2, 4-3
폼 닫기 4-3
폼 디자이너
 사용자 지정 5-11
정의 2-4
폼 파일
 정의 4-1
 코드 보기 2-11
폼에 컴포넌트 추가 4-3, 4-12
표기법 1-4
프레임 5-8
프로그램
 국제화 3-8
 배포 3-8
 웹 서버 애플리케이션 3-9
 컴파일 및 디버깅 3-6, 4-15
 CLX 애플리케이션 3-9
프로그램 디버깅 3-6 ~ 3-7, 4-15
프로젝트
 관리 2-12 ~ 2-13
 기본값으로 옵션 설정 5-9
 기본값으로 지정 5-9
 생성 3-1
 저장 4-2
 타입 3-8 ~ 3-11
 항목 추가 2-6
프로젝트 그룹 2-13
프로젝트 템플릿 5-10
프로젝트 파일, 기본 이름 4-1

A

About 상자, 추가 4-24
Action Manager 에디터 4-7 ~ 4-10
ActiveX
 컨트롤 설치 5-9
 컴포넌트 팔레트 페이지 3-12
ADO 3-10

B

BDE 3-10
BDE 관리자 3-10
Browser 2-13

C

Class Completion 2-8
CLX
 애플리케이션, 생성 3-9
 정의 3-5
 컴포넌트 추가 2-4
Code Completion 2-7
Code Insight 툴 2-7
Code Parameters 2-7
Code Templates 2-7

D

dbExpress 3-10
Delphi
 사용자 지정 5-1 ~ 5-12
 소개 1-1
 시작 2-1
 프로그래밍 3-1
Delphi 시작 2-1
Delphi 프로그래밍 3-1
Diagram 페이지 2-9
DLL
 배포 3-8
 생성 2-6
 정의 3-12
.dfm 파일 2-11, 4-1
.dpr 파일 참조 4-1

E

Editor Options 대화 상자 2-8, 5-12
Environment Options 대화 상자 2-8, 5-11

G

GUI, 만들기 4-2

I

IDE
 구성 5-1
 둘러 보기 2-1
 사용자 지정 5-1 ~ 5-12

정의 1-1
IDE 통합 개발 환경 참조
IME 3-8
InterBase 3-10

K

Kylix
 애플리케이션 개발 3-9
 정의 1-1

N

New Items 대화 상자
 사용 2-6, 4-24
 템플릿 저장 5-9, 5-11

O

Object Inspector
 사용 3-3 ~ 3-4, 4-2
 인라인 컴포넌트 참조 3-4
 정의 2-4
Object Repository
 사용 2-6
 정의 2-6, 3-1
 템플릿 추가 5-9, 5-10
Object Repository에 항목 추가 2-6
Object TreeView 2-5
ODBC 3-10

P

Paradox 3-10
Project Browser 2-13
Project Manager 2-12 ~ 2-13
Project Options 대화 상자 5-9
.pas 파일 4-1

R

Resource DLL 마법사 3-8

S

SQL 데이터베이스 서버 3-10
SQL 탐색기 3-11
SQL Links 3-10
SQL Server 3-10
StatusBar1.Panels 대화 상자 편집 4-5

T

To-do List 2-14
Tooltip Expression Evaluation 2-7
Tooltip Symbol Insight 2-7

V

Visual Component Library (VCL)

사용 3-5

컴포넌트 추가 2-4

W

WebSnap, 소개 3-9

X

.xflm 파일 2-11

