



# 입문서

---



Borland®  
**C++Builder™ 6**  
**Windows™ 용**

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

C++Builder 라이선스 조항과 하자 보증에 따라 배포할 수 있는 전체 파일 리스트를 보시려면 C++Builder 6 제품의 루트 디렉토리에 있는 DEPLOY 문서를 참조하십시오.

Borland는 이 문서에 포함된 내용에 대해서 특허권을 가지고 있거나 특허 출원 중에 있습니다. 이 문서를 공급한다고 해서 특허권에 대한 라이선스가 부여되는 것은 아닙니다. 적용 가능한 특허권 리스트는 제품 CD나 About 다이얼로그 박스를 참조하십시오.

COPYRIGHT © 1983–2002 Borland Software Corporation. All rights reserved. 모든 Borland 상표 및 제품 이름은 미국 및 기타 여러 나라에서 Borland Software Corporation의 상표 또는 등록 상표입니다. 기타 모든 표시는 해당 소유자의 자산입니다.

Printed in the U.S.A.

CPE1360WW21000 6E2R0102

0203040506-9 8 7 6 5 4 3 2 1

D3

# 목차

<b>1장</b>		
<b>서문</b>	<b>1-1</b>	
C++Builder란 무엇인가?	1-1	
C++Builder 등록	1-1	
정보 찾기	1-3	
온라인 도움말	1-3	
F1 도움말	1-4	
인쇄된 설명서	1-6	
개발자 지원 서비스와 웹 사이트	1-7	
표기법	1-7	
<b>2장</b>		
<b>환경 둘러 보기</b>	<b>2-1</b>	
C++Builder 시작	2-1	
IDE	2-1	
메뉴 및 툴바	2-3	
컴포넌트 팔레트, 폼 디자이너 및 Object Inspector	2-3	
Object TreeView	2-4	
Object Repository	2-5	
코드 에디터	2-6	
Diagram 페이지	2-7	
폼 코드 보기	2-8	
ClassExplorer	2-9	
Project Manager	2-9	
To-do list	2-10	
<b>3장</b>		
<b>C++Builder로 프로그래밍</b>	<b>3-1</b>	
프로젝트 만들기	3-1	
데이터 모듈 추가	3-2	
사용자 인터페이스 생성	3-2	
폼에 컴포넌트 놓기	3-2	
컴포넌트 속성 설정	3-4	
코드 작성	3-6	
이벤트 핸들러 작성	3-6	
VCL 및 CLX 라이브러리 사용	3-6	
프로젝트 컴파일 및 디버깅	3-7	
애플리케이션 배포	3-9	
애플리케이션 국제화	3-9	
프로젝트 타입	3-9	
CLX 애플리케이션	3-10	
웹 서버 애플리케이션	3-10	
데이터베이스 애플리케이션	3-11	
사용자 정의 컴포넌트	3-13	
DLL	3-13	
COM과 ActiveX	3-13	
<b>4장</b>		
<b>텍스트 에디터 만들기 - 자습서</b>	<b>4-1</b>	
새 애플리케이션 시작	4-1	
속성 값 설정	4-2	
폼에 컴포넌트 추가	4-3	
메뉴와 툴바에 대한 지원 추가	4-6	
Action Manager Editor와 Action List Editor의 차이점	4-6	
메뉴 및 툴바 이미지 추가 (기업용 및 전문가용)	4-7	
Action Manager에 액션 추가 (기업용 및 전문가용)	4-8	
표준 액션 추가(기업용 및 전문가용)	4-11	
메뉴 추가(기업용 및 전문가용)	4-12	
툴바 추가(기업용 및 전문가용)	4-13	
이미지 리스트 및 이미지 추가 (개인용 에디션)	4-13	
액션 리스트에 액션 추가 (개인용 에디션)	4-15	
액션 리스트에 표준 액션 추가 (개인용 에디션)	4-17	
메뉴 추가(개인용 에디션)	4-19	
툴바 추가(개인용 에디션)	4-21	
텍스트 영역 지우기(모든 에디션)	4-22	
이벤트 핸들러 작성	4-23	
New 명령에 대한 이벤트 핸들러 만들기	4-23	
Open 명령에 대한 이벤트 핸들러 만들기	4-26	
Save 명령에 대한 이벤트 핸들러 만들기	4-27	
Save As 명령에 대한 이벤트 핸들러 만들기	4-28	

도움말 파일 만들기 . . . . .	4-30
Help Contents 명령에 대한 이벤트 핸들러 만들기 . . . . .	4-30
Help Index 명령에 대한 이벤트 핸들러 만들기 . . . . .	4-31
About 박스 만들기 . . . . .	4-32
애플리케이션 완료 . . . . .	4-34

## 5장

### CLX 데이터베이스 애플리케이션 만들기 자습서 **5-1**

데이터베이스 아키텍처 개요 . . . . .	5-1
새 CLX 애플리케이션 만들기 . . . . .	5-2
데이터 액세스 컴포넌트 설정 . . . . .	5-3
데이터베이스 연결 설정 . . . . .	5-3
단방향 데이터셋 설정 . . . . .	5-5
프로바이더, 클라이언트 데이터셋 및 데이터 소스 설정 . . . . .	5-5
사용자 인터페이스 디자인 . . . . .	5-6
그리드 및 탐색 모음 만들기 . . . . .	5-6
메뉴에 대한 지원 추가 . . . . .	5-8
메뉴 추가 . . . . .	5-9
버튼 추가 . . . . .	5-11
이름과 이미지 표시 . . . . .	5-12

이벤트 핸들러 작성 . . . . .	5-13
Update Now! 명령 이벤트 핸들러 작성 . . . . .	5-13
Exit 명령 이벤트 핸들러 작성 . . . . .	5-14
FormClose 이벤트 핸들러 작성 . . . . .	5-14

## 6장

### 데스크탑 사용자 정의 **6-1**

작업 영역 구성 . . . . .	6-1
메뉴와 툴바 정렬 . . . . .	6-1
툴 윈도우 도킹 . . . . .	6-2
데스크탑 레이아웃 저장 . . . . .	6-4
컴포넌트 팔레트 사용자 정의 . . . . .	6-5
컴포넌트 팔레트 정렬 . . . . .	6-5
컴포넌트 템플릿 만들기 . . . . .	6-5
컴포넌트 패키지 설치 . . . . .	6-6
프로젝트 옵션 설정 . . . . .	6-8
디폴트 프로젝트 옵션 설정 . . . . .	6-8
프로젝트와 폼 템플릿을 기본값으로 지정 . . . . .	6-8
Object Repository에 템플릿 추가 . . . . .	6-9
도구 환경 설정 . . . . .	6-10
폼 디자이너 사용자 정의 . . . . .	6-10
코드 에디터 사용자 정의 . . . . .	6-11

### 색인 **1-1**



## 서문

이 입문서에서는 개발자가 본 제품의 사용 방법을 즉시 익힐 수 있도록 C++Builder 개발 환경에 대한 개요를 제공합니다. 아울러 C++Builder에서 사용할 수 있는 도구와 기능에 대한 자세한 내용을 어디서 찾을 수 있는지 알려 줍니다.

2장의 "환경 둘러 보기"에서는 C++Builder 데스크탑의 주요 도구와 통합 개발 환경(IDE)에 대해 설명합니다. 3장의 "C++Builder로 프로그래밍"에서는 이러한 도구를 사용하여 애플리케이션을 만드는 방법에 대해 설명합니다. 4장의 "텍스트 에디터 만들기 - 자습서"에서는 텍스트 에디터용 프로그램 작성 방법을 자습서를 통해 단계적으로 설명합니다. 5장의 "CLX 데이터베이스 애플리케이션 만들기 자습서"에서는 데이터베이스 애플리케이션을 만드는 과정을 보여 줍니다. 6장의 "데스크탑 사용자 정의"에서는 자신의 개발 요구에 맞게 C++Builder를 사용자 정의하는 방법에 대해 설명합니다.

## C++Builder란 무엇인가?

C++Builder는 신속한 애플리케이션 개발(RAD)을 위한 비주얼 객체 지향 프로그래밍 환경입니다. C++Builder를 사용하면 수동 코딩 작업을 최소화하는 동시에 성능이 뛰어난 Microsoft Windows XP, Microsoft Windows 2000 및 Microsoft Windows 98용 애플리케이션을 만들 수 있습니다. C++Builder는 애플리케이션을 개발, 테스트 및 배포하는 데 필요한 모든 도구, 즉 재사용할 수 있는 컴포넌트로 이루어진 대형 라이브러리, 일련의 디자인 도구, 애플리케이션 및 폼 템플릿, 프로그래밍 마법사 등을 제공합니다.

## C++Builder 등록

C++Builder는 여러 방법으로 등록할 수 있습니다. C++Builder를 설치 후 처음 시작하면 일련 번호와 인증 키를 입력하라는 메시지가 나타납니다. 이러한 정보를 입력하면 다음 세 가지 옵션을 제공하는 등록 다이얼로그 박스가 나타납니다.

- Register and activate the software online.

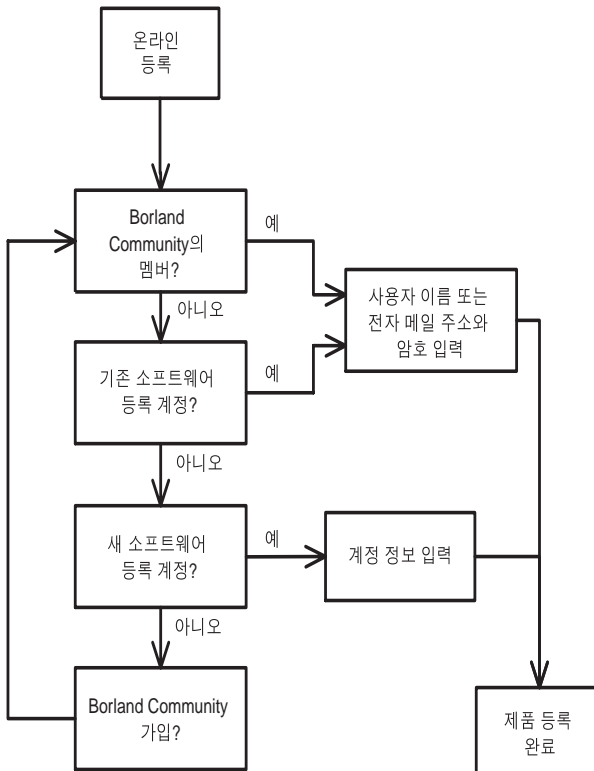
이 옵션을 선택하면 기존 인터넷 연결을 사용하여 온라인으로 등록할 수 있습니다.

- Register or activate by phone or web browser.

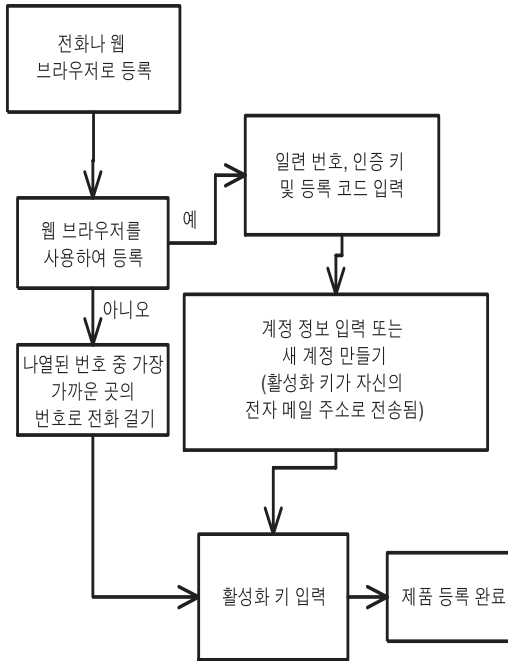
이 옵션을 선택하면 전화나 웹 브라우저를 통해 등록할 수 있습니다. 전자 메일을 통해 활성화 키를 받았다면 이 옵션을 사용하여 해당 키를 입력합니다.

- I will register at a later time.

온라인 등록이 C++Builder를 등록하는 가장 쉬운 방법이지만, 인터넷에 대한 활성 연결이 필요합니다. 현재 Borland Community의 멤버이거나 기존 소프트웨어 등록 계정이 있는 경우에는 단순히 관련 계정 정보만 입력하면 C++Builder가 자동으로 등록됩니다. 그렇지 않을 경우에는 등록 과정에서 제공하는 방법에 따라 이러한 계정을 만듭니다.



소프트웨어를 설치하는 컴퓨터가 인터넷에 연결되어 있지 않거나, 온라인 등록을 차단하는 방화벽을 사용 중이거나, 이전에 활성화 키를 받은 경우라면 전화나 웹 브라우저를 통해 등록하거나 활성화하는 두 번째 옵션이 유용합니다.



**참고** 특별한 경우가 아니면 가능한 한 온라인 등록 옵션을 사용하십시오.

## 정보 찾기

이 장에 설명되어 있는 다음과 같은 방법으로 C++Builder에 대한 정보를 찾을 수 있습니다.

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

이 릴리스의 새 기능에 대한 자세한 내용은 온라인 도움말 목차에 있는 What's New 항목과 [www.borland.com](http://www.borland.com) 웹 사이트를 참조하십시오.

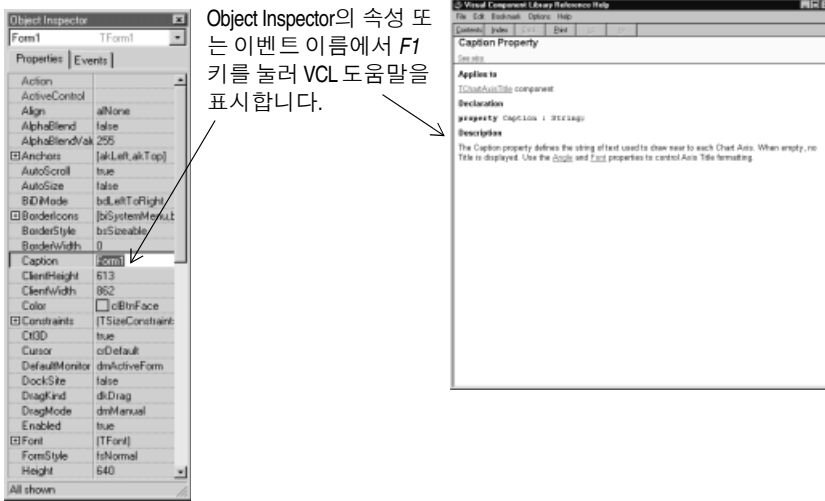
## 온라인 도움말

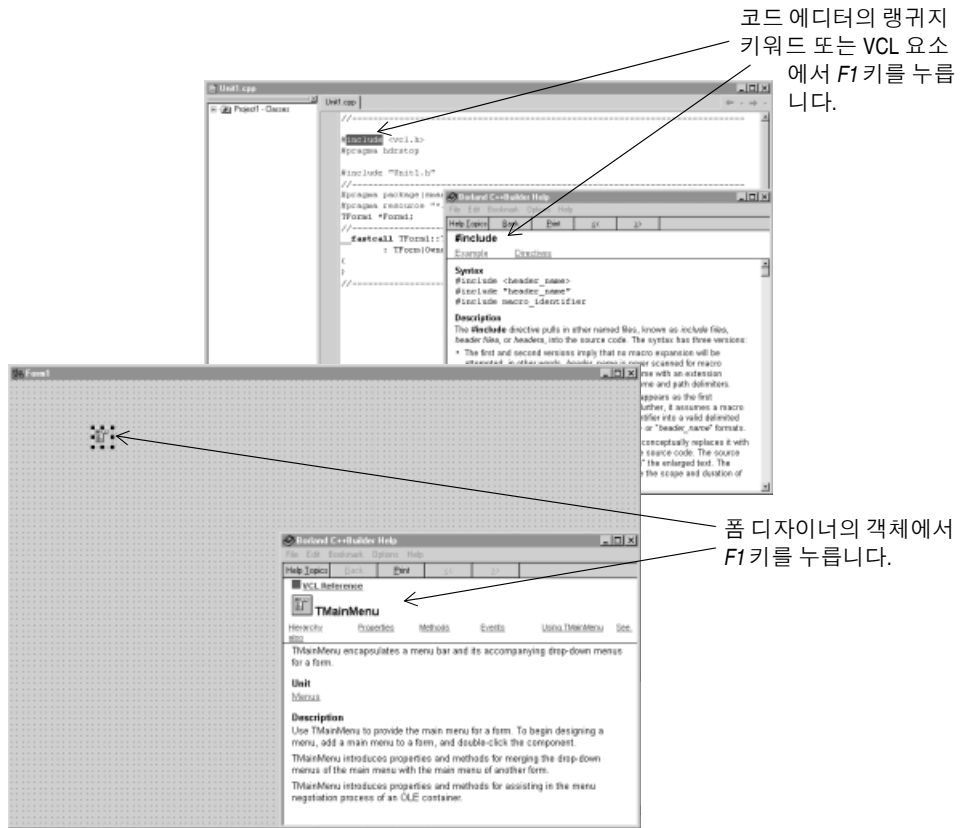
온라인 도움말 시스템은 사용자 인터페이스 기능, 랭귀지 구현, 프로그래밍 작업, 비주얼 컴포넌트 라이브러리(VCL) 참조 및 Borland 크로스 플랫폼용 컴포넌트 라이브러리(C LX)의 컴포넌트 등에 대한 자세한 정보를 제공합니다. 이 시스템에는 *개발자 안내서*의 모든 자료와 C++Builder에서 제공하는 다른 기능에 대한 일련의 도움말 파일이 포함되어 있습니다.

목차를 보려면 Help | C++Builder Help 및 Help | C++Builder Tools를 선택하고 Contents 탭을 클릭하십시오. VCL 또는 CLX 객체나 다른 항목을 찾으려면 Index 또는 Find 탭을 클릭하여 요청을 입력하십시오.

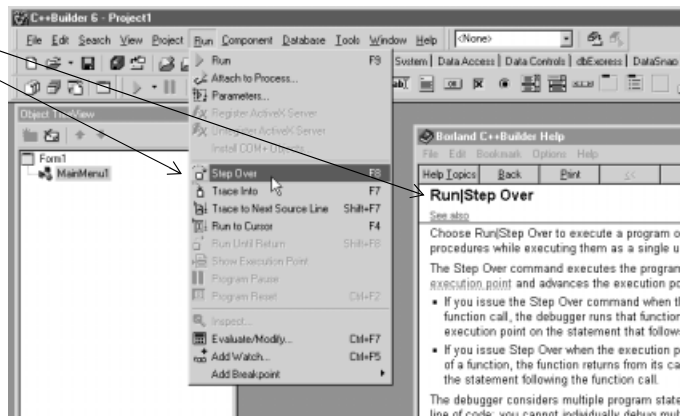
## F1 도움말

항목을 선택하고 **F1** 키를 누르면 메뉴 항목, 다이얼로그 박스, 툴바, 컴포넌트 등을 비롯하여 VCL, CLX 및 개발 환경의 모든 부분에 대한 상황에 맞는 도움말을 볼 수 있습니다.





메뉴 명령, 다이얼로그 박스 또는 윈도우에서 F1 키를 눌러 해당 항목에 대한 도움말을 표시합니다.



또한 모든 다이얼로그 박스에서 **Help** 버튼을 클릭하면 문맥에 따른 온라인 설명서가 나타납니다.

컴파일러와 링커에서 발생한 오류 메시지는 코드 에디터 아래에 별도의 윈도우로 나타납니다. 컴파일 오류에 대한 도움말을 보려면 리스트에서 메시지를 선택하고 **F1** 키를 누르십시오.

## 인쇄된 설명서

---

이 *입문서*는 C++Builder에 대한 소개입니다. *개발자 안내서*와 같은 인쇄된 설명서를 추가로 주문하려면 [shop.borland.com](http://shop.borland.com)을 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.

이 웹 사이트를 통해 C++Builder 개발자가 정보, 팁, 기술 등을 교환하는 여러 뉴스그룹에 액세스할 수 있습니다. 또한 이 웹 사이트에는 C++Builder에 대한 도서 리스트, 추가 C++Builder 기술 문서, FAQ(질문과 대답) 등이 포함되어 있습니다.

## 표기법

이 설명서에서는 특별한 의미를 지닌 텍스트에 다음과 같은 글꼴을 사용합니다.

글꼴	의미
Monospace type	고정 폭 글꼴은 텍스트를 화면이나 코드에 표시되는 모양 그대로 나타냅니다. 사용자가 입력해야 하는 내용 역시 이 글꼴로 나타냅니다.
<b>Boldface</b>	텍스트 또는 코드 리스트에서 굵게 쓰여진 글자는 예약어 또는 컴파일러 옵션을 나타냅니다.
<i>Italics</i>	텍스트에서 이탤릭체 단어는 변수 또는 타입 이름과 같은 C++Builder 식별자를 나타냅니다. 이탤릭체는 새로운 용어 같은 일부 단어를 강조하기 위해 사용하기도 합니다.
<i>Keycaps</i>	이 글꼴은 키보드에 있는 특정 키를 나타냅니다. 예를 들면, 다음과 같습니다. “Esc 키를 눌러 메뉴를 종료합니다.”





## 환경 둘러 보기

이 장에서는 C++Builder를 시작하는 방법에 대해 설명하고 통합 개발 환경(IDE)의 주요 부분과 도구에 대한 간략한 둘러 보기를 제공합니다.

### C++Builder 시작

---

다음 방법으로 C++Builder를 시작할 수 있습니다.

- 바로 가기를 만든 경우 C++Builder 아이콘을 더블 클릭합니다.
- Windows 시작 메뉴에서 프로그램 | Borland C++Builder를 선택합니다.
- Windows 시작 메뉴에서 실행을 선택한 다음 Bcb를 입력합니다.
- CBuilder6\Bin 디렉토리에서 Bcb.exe를 더블 클릭합니다.

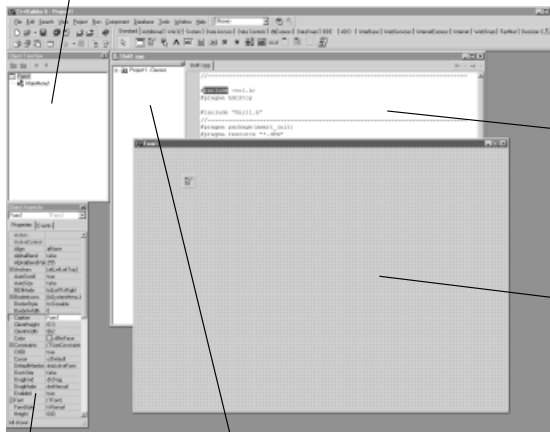
### IDE

---

C++Builder를 처음 시작하면 IDE에 있는 주요 도구 몇 가지를 볼 수 있습니다. C++Builder의 IDE에는 메뉴, 툴바, 컴포넌트 팔레트, Object Inspector, Object TreeView, 코드 에디터, ClassExplorer, Project Manager 및 기타 다른 도구들이 많이 들어 있습니다. 사용할 수 있는 기능과 컴포넌트는 구입한 C++Builder 에디션에 따라 다릅니다.

Object TreeView는 컴포넌트의 부모/자식 관계에 대한 계층 뷰를 표시합니다.

애플리케이션에서 사용하기 위해 미리 만들어진 컴포넌트로 구성된 팔레트입니다.



코드를 보고 편집하는데 사용할 코드 에디터입니다.

폼 디자이너는 애플리케이션의 UI 디자인을 시작할 비어 있는 폼을 포함합니다. 애플리케이션에는 많은 폼이 포함될 수 있습니다.

Object Inspector는 객체 속성을 변경하고 이벤트 핸들러를 선택하는 데 사용됩니다.

Class Explorer는 유닛에 있는 클래스, 변수 및 루틴을 보여 주고 신속하게 탐색할 수 있게 해 줍니다.

C++Builder의 개발 모델은 **two-way** 도구를 기반으로 합니다. 이것은 비주얼 디자인 도구와 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 C++Builder에 의해 생성된 모든 코드를 수동으로 편집할 수 있습니다.

IDE에서는 모든 프로그래밍 도구를 쉽게 찾고 사용할 수 있습니다. IDE를 종료하지 않고 그래픽 인터페이스 디자인, 클래스 라이브러리를 통해 찾기, 코드 작성 등과 프로젝트 컴파일, 테스트, 디버거 및 관리를 수행할 수 있습니다.

IDE 구성에 대해 알아보려면 6장의 "데스크탑 사용자 정의"를 참조하십시오.

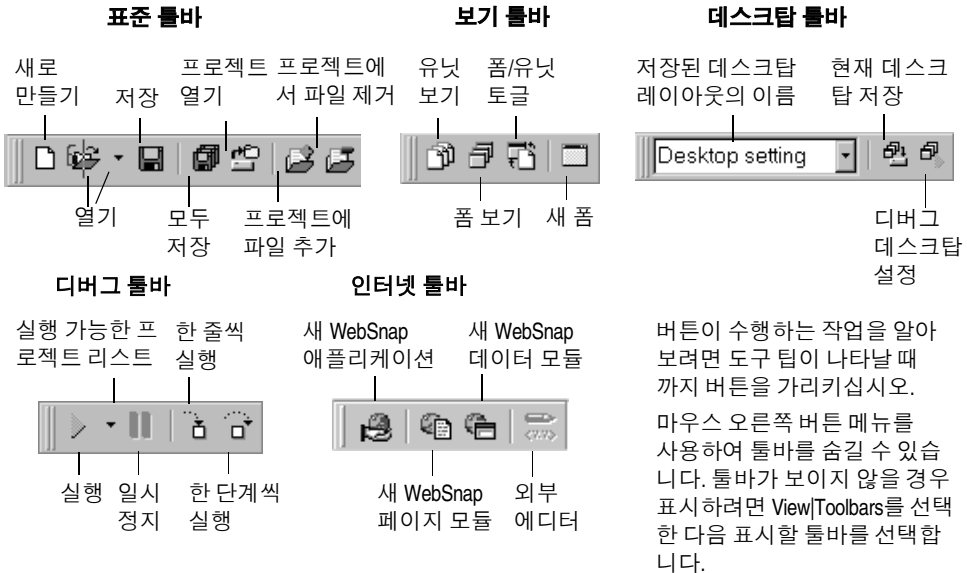
## 메뉴 및 툴바

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



디폴트 정렬  
상태의 메인  
윈도우

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



툴바 버튼 뿐만 아니라 단축키를 사용하여 많은 작업을 수행할 수 있습니다. 사용할 수 있는 단축키는 항상 드롭다운 메뉴에 있는 명령 옆에 나타납니다.

많은 도구와 아이콘을 마우스 오른쪽 버튼으로 클릭하면 작업 중인 객체와 관련된 명령 메뉴가 표시됩니다. 이 메뉴를 컨텍스트 메뉴라고 합니다.

툴바를 사용자 정의할 수도 있습니다. 원하는 명령을 추가하거나 툴바를 다른 위치로 이동할 수 있습니다. 자세한 내용은 6-1페이지의 "메뉴와 툴바 정렬" 및 6-4페이지의 "데스크탑 레이아웃 저장"을 참조하십시오.

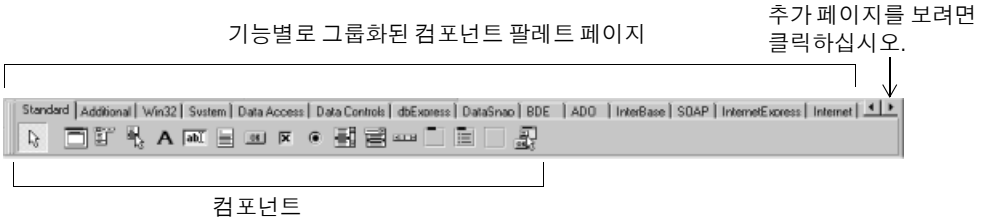
### 자세한 내용은...

메뉴 옵션에 대한 도움말을 보려면 해당 메뉴 옵션을 가리킨 다음 **F1** 키를 누르십시오.

## 컴포넌트 팔레트, 폼 디자이너 및 Object Inspector

컴포넌트 팔레트, 폼 디자이너, Object Inspector 및 Object TreeView는 함께 사용되어 애플리케이션의 사용자 인터페이스 생성을 도와 줍니다.

컴포넌트 팔레트에는 비주얼(visual) 또는 논비주얼(nonvisual) VCL 및 CLX 컴포넌트를 나타내는 아이콘 그룹이 있는 탭 모양의 페이지가 포함되어 있습니다. 각 페이지에 포함된 컴포넌트는 다양한 기능별로 그룹화되어 있습니다. 예를 들어, Standard, Additional 및 Win32 페이지에는 에디트 박스 및 위쪽/아래쪽 버튼과 같은 윈도우 컨트롤이 포함되어 있고 Dialogs 페이지에는 파일 열기 및 저장과 같은 파일 작업에 사용하는 일반적인 다이얼로그 박스가 포함되어 있습니다.



각 컴포넌트에는 애플리케이션을 제어할 수 있는 속성, 이벤트, 메소드 등의 특정 어트리뷰트(attribute)가 있습니다.

컴포넌트를 폼이나 폼 디자이너에 둔 다음 사용자 인터페이스에 나타나야 할 모양대로 정렬할 수 있습니다. 폼에 가져다 놓은 컴포넌트에 대해서는 *Object Inspector*를 사용하여 디자인 타임 속성을 설정하고 이벤트 핸들러를 만들며 비주얼 속성 및 이벤트를 필터링하여 애플리케이션의 비주얼한 모습과 애플리케이션을 실행시키는 코드를 연결할 수 있습니다. 3-2페이지의 "폼에 컴포넌트 놓기"를 참조하십시오.

#### 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 Inspector* 및 폼 디자이너와 동기화되므로 *Object TreeView*에서 포커스를 변경하면 *Object Inspector*와 폼에서도 포커스가 변경됩니다.

Object TreeView를 사용하면 관련 컴포넌트의 관계를 서로 변경할 수 있습니다. 예를 들어, 패널 및 체크 박스 컴포넌트를 폼에 추가할 경우 두 컴포넌트는 형제입니다. 그러나 *Object TreeView*에서 체크 박스를 패널 아이콘 위에 끌어 놓으면 체크 박스는 패널의 자식이 됩니다.

객체의 속성이 완성되지 않았다면 *Object TreeView*는 해당 객체 옆에 빨간색 물음표를 표시합니다. 트리 다이어그램에서 임의의 객체를 더블 클릭하여 이벤트 핸들러를 작성할 위치에서 코드 에디터를 열 수도 있습니다.

Object TreeView가 표시되지 않으면 View | Object TreeView를 선택합니다.

Object TreeView는 특히 데이터베이스 객체 간의 관계를 표시할 때 유용합니다.

#### 자세한 내용은...

온라인 도움말 색인의 "Object TreeView"를 참조하십시오.

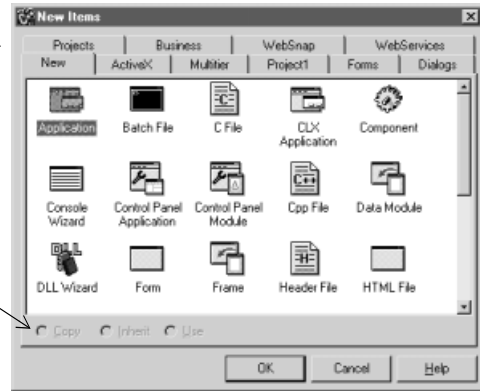
# Object Repository

Object Repository에는 개발 작업을 쉽게 수행할 수 있도록 도와 주는 폼, 다이얼로그 박스, 데이터 모듈, 마법사, DLL, 예제 애플리케이션 및 그 밖의 다른 항목이 포함되어 있습니다. 프로젝트를 시작할 때 File | New | Other를 선택하여 New Items 다이얼로그 박스를 표시합니다. New Items 다이얼로그 박스는 Object Repository와 같습니다. Repository를 선택하여 만들려는 것과 유사한 객체가 Repository에 있는지 확인합니다.

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

Object Repository에 들어 있는 한 항목을 기반으로 하여 항목을 하나 만들 경우, 항목을 복사, 상속 또는 사용할 수 있습니다.

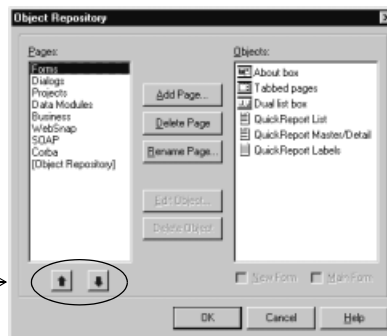
Copy(기본값)는 프로젝트에 있는 항목의 복사본을 만듭니다. Inherit는 Repository의 객체에 대한 변경 내용이 프로젝트의 객체에 의해 상속된다는 것을 의미합니다. Use는 프로젝트의 객체에 대한 변경 내용이 Repository의 객체에 의해 상속된다는 것을 의미합니다.



Object Repository에서 객체를 편집하거나 제거하려면, Tools | Repository를 선택하거나 New Items 다이얼로그 박스에서 마우스 오른쪽 버튼을 클릭하고 Properties를 선택하십시오.

Object Repository에서 탭 모양의 페이지를 추가 또는 제거하거나 이름을 바꿀 수 있습니다.

화살표를 클릭하여 New Items 다이얼로그 박스에서 탭 모양의 페이지가 나타나는 순서를 변경합니다.



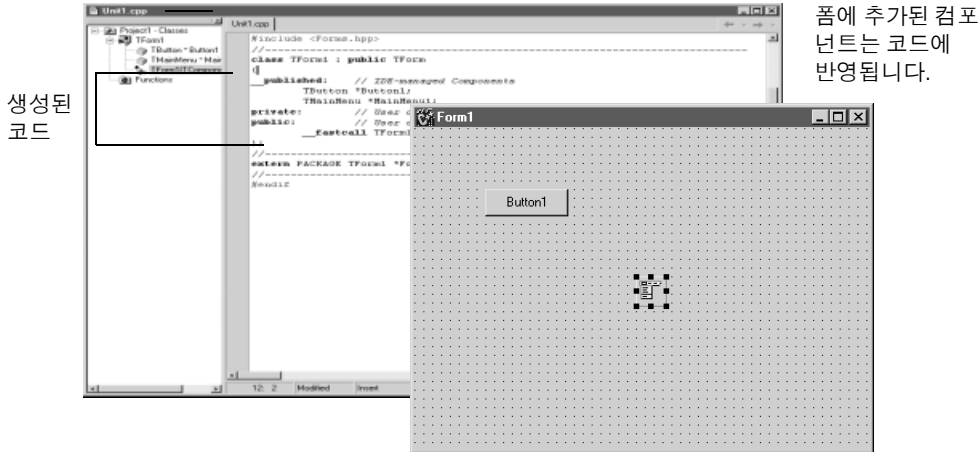
프로젝트 및 폼 템플릿을 Object Repository에 추가하려면 6-9페이지의 "Object Repository에 템플릿 추가"를 참조하십시오.

## 자세한 내용은...

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

## 코드 에디터

애플리케이션의 사용자 인터페이스를 디자인할 때, C++Builder는 원본으로 사용되는 객체 코드를 생성합니다. 폼과 객체의 속성을 선택하고 수정하면 변경 내용이 소스 파일에 자동으로 반영됩니다. 모든 기능을 갖춘 ASCII 에디터인 기본 코드 에디터를 직접 사용하면 소스 파일에 코드를 추가할 수 있습니다. C++Builder는 Code Insight 도구, Class Completion 및 Code Browsing를 비롯하여 코드 작성을 돕는 다양한 방식을 제공합니다.



### Code Insight

Code Insight 도구는 문맥에 따른 팝업 윈도우를 표시합니다.

도구	작동 방법
Code Completion	화살표(<=>)가 뒤에 오는 객체에 대한 포인터나 점이 뒤에 오는 VCL이 아닌 객체를 나타내는 변수의 이름을 입력합니다. 할당문의 시작 부분을 입력하고 <b>Ctrl+space</b> 를 눌러 변수에 대한 유효 값 리스트를 표시합니다. 프로시저, 함수 또는 메소드 이름을 입력하여 인수 리스트를 나타냅니다.
Code Parameters	메소드 이름과 빈 괄호를 입력하여 메소드 인수에 대한 구문을 표시합니다.
Tooltip Expression Evaluation	디버깅 중 프로그램 실행이 멈춘 상태에서 변수에 마우스를 갖다 대면 변수의 현재 값을 보여 줍니다.
Tooltip Symbol Insight	코드를 편집할 때 식별자에 마우스를 갖다 대면 타입 선언을 보여 줍니다.
Code Templates	<b>Ctrl+J</b> 를 눌러 코드에 삽입할 수 있는 일반적인 프로그래밍 코드 리스트를 확인합니다. C++Builder에서 제공되는 템플릿 외에도 자신의 고유한 템플릿을 만들 수 있습니다.

이러한 도구를 선택하거나 선택하지 않으려면 Tools | Editor Options를 선택하고 Code Insight 탭을 클릭하십시오. 그런 다음 Automatic features 섹션에서 도구를 선택하거나 선택 해제합니다.

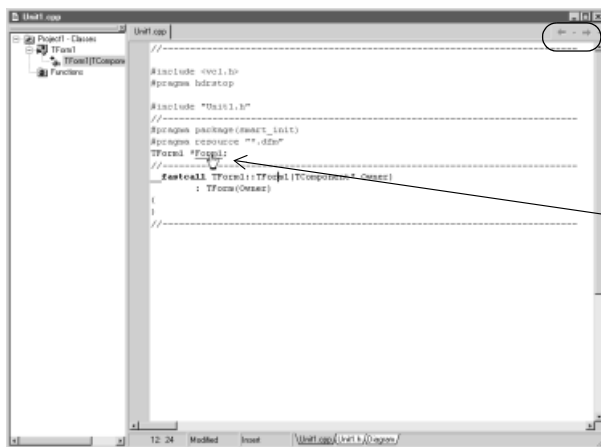
**자세한 내용은...**

온라인 도움말 색인의 "Code Insight"를 참조하십시오.

**Code Browsing**

클래스, 변수, 속성, 메소드 또는 다른 식별자 이름 위로 마우스를 가져가면 식별자가 선언된 위치에 **Tooltip Symbol Insight**라는 팝업 메뉴가 표시됩니다. **Ctrl** 키를 누르면 커서가 손 모양으로 바뀌고 식별자가 파란색으로 변하면서 밑줄이 그어집니다. 여기서 식별자를 클릭하면 해당 식별자의 정의로 이동할 수 있습니다.

코드 에디터에는 웹 브라우저에 있는 것과 비슷한 앞으로 버튼과 뒤로 버튼이 있습니다. 식별자 정의로 이동할 때 코드 에디터는 개발자가 코드에서 있었던 위치를 계속 기억합니다. 앞으로 버튼과 뒤로 버튼 옆에 있는 드롭다운 화살표를 클릭하면 이러한 참조의 기록을 통해 앞뒤로 이동할 수 있습니다.



← 에디터를 웹 브라우저처럼 사용합니다.

**Ctrl** 키를 누르고 식별자를 가리킵니다. 그러면 커서는 손 모양으로 바뀌고 식별자는 파란색으로 변하고 밑줄이 그어집니다.

이것을 클릭해서 식별자 정의로 갑니다.

탐색 후에는 뒤로 화살표를 클릭하여 이전 위치로 돌아 갑니다.

코드 편집 환경을 사용자 정의하려면 6-11 페이지의 "코드 에디터 사용자 정의"를 참조하십시오.

**자세한 내용은...**

온라인 도움말 색인의 "Code editor"를 참조하십시오.

**Diagram 페이지**

코드 에디터 하단에는 사용 중인 C++Builder의 에디션에 따라 하나 이상의 탭이 포함될 수 있습니다. 모든 코드를 작성하는 **Code** 페이지는 디폴트로 앞쪽에 표시됩니다. **Diagram** 페이지는 폼 또는 데이터 모듈에 둔 컴포넌트 간의 관계를 나타내는 아이콘과 연결선을 표시합니다. 이러한 관계에는 형제, 부모/자식 또는 컴포넌트/속성 간의 관계가 포함됩니다.

다이어그램을 만들려면 **Diagram** 페이지를 클릭하십시오. **Object TreeView**에서 간단하게 하나 이상의 아이콘을 **Diagram** 페이지로 끌어와 세로로 정렬합니다. 아이콘을 가로로 정렬하려면 **Shift** 키를 누른 상태에서 끌어 놓으십시오. 부모/자식 또는 컴포넌트/속성 종속 관계가 있는 아이콘을 페이지 위로 끌어 놓으면 종속 관계를 표시하는 선 또는 연결선이 자동으로 추가

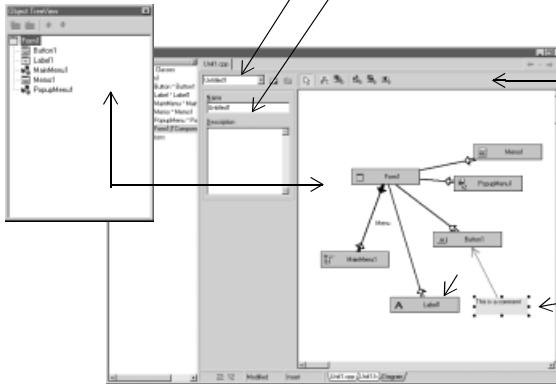
됩니다. 예를 들어, 데이터셋 컴포넌트를 데이터 모듈에 추가하고 데이터셋 아이콘을 그 속성 아이콘과 함께 **Diagram** 페이지에 끌어 놓으면 속성 연결선이 속성 아이콘과 데이터셋 아이콘을 자동으로 연결합니다.

종속 관계가 없지만 이를 표시하고자 하는 컴포넌트의 경우에는 **Diagram** 페이지 상단의 툴바 버튼을 사용하여 네 가지 연결선 타입, 즉 암시, 속성, 마스터/디테일 및 조화 중에서 하나를 추가합니다. 또한 서로 연결되거나 관련 아이콘에 연결되는 주석 블록을 추가할 수 있습니다.

Object TreeView에서 컴포넌트의 아이콘을 **Diagram** 페이지로 끌어 놓습니다.

현재 프로젝트에서 이름이 지정된 다른 다이어그램을 보려면 드롭다운 리스트 박스를 클릭합니다.

다이어그램의 이름과 설명을 입력합니다.



**Diagram** 페이지 툴바 버튼, 즉 **Property**, **Master/Detail** 및 **Lookup**을 사용하여 컴포넌트 간의 관계와 컴포넌트 및 그 속성 간의 관계를 지정합니다. 연결선 모양은 각 관계의 타입에 따라 달라집니다.

**Comment block** 버튼을 클릭하여 주석을 추가하고 **Allude connector** 버튼을 클릭하여 다른 주석 또는 아이콘과의 연결선을 그립니다.

다이어그램의 이름과 설명을 입력하고, 다이어그램을 저장하고, 작업이 끝나면 인쇄할 수 있습니다.

## 자세한 내용은...

온라인 도움말 색인의 "diagram page"를 참조하십시오.

## 폼 코드 보기

폼은 대부분의 C++Builder 프로젝트에서 매우 가시적인 부분으로써, 여기서 애플리케이션의 사용자 인터페이스를 디자인합니다. 보통 C++Builder의 비주얼 도구를 사용하여 폼을 디자인하며, C++Builder는 폼을 폼 파일에 저장합니다. 확장자가 .dfm(또는 CLX 애플리케이션의 경우에는 .xfrm)인 폼 파일은 모든 영구적 속성 값을 비롯하여 폼의 각 컴포넌트에 대해 설명합니다. 코드 에디터에서 폼 파일을 보고 편집하려면 폼을 마우스 오른쪽 버튼으로 클릭하고 **View as Text**를 선택하십시오. 폼의 그래픽 보기로 돌아가려면 마우스 오른쪽 버튼을 클릭하고 **View as Form**을 선택하십시오.

텍스트(기본값) 또는 바이너리 형식으로 폼 파일을 저장할 수 있습니다. **Tools | Environment Options**를 선택하고 **Designer** 페이지를 클릭한 다음 **New forms as text** 체크 박스를 선택하거나 선택 해제하여 새로 만든 폼에 사용할 형식을 지정합니다.



**자세한 내용은...**

온라인 도움말 색인의 "form files"를 참조하십시오.

## ClassExplorer

---

C++Builder를 열면 설치된 C++Builder의 에디션에서 Code Explorer를 사용할 수 있는지 여부에 따라 ClassExplorer가 코드 에디터 윈도우의 왼쪽에 도킹됩니다. ClassExplorer는 코드 에디터에 열려 있는 소스 코드의 목차를 트리 다이어그램으로 표시하여 유닛에 정의된 타입, 클래스, 속성, 메소드, 전역 변수, 루틴 등을 나열합니다.

ClassExplorer를 사용하면 코드 에디터에서 탐색할 수 있습니다. 예를 들어, ClassExplorer에서 메소드를 더블 클릭하면 코드 에디터의 유닛의 인터페이스 부분에 있는 클래스 선언의 정의로 커서가 이동합니다.

Code Explorer에서 내용이 표시되는 방법을 구성하려면 Tools | Environment Options를 선택하고 Explorer 탭을 클릭하십시오.

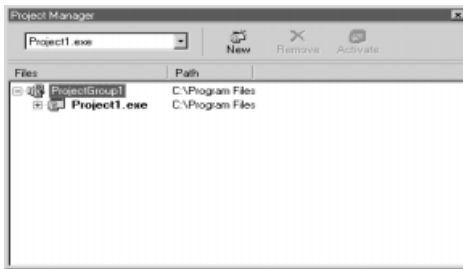
**자세한 내용은...**

온라인 도움말 색인의 "ClassExplorer"를 참조하십시오.

## Project Manager

---

C++Builder를 처음 시작하면 새 프로젝트가 자동으로 열립니다. 프로젝트는 개발하려는 애플리케이션 또는 DLL을 구성하는 여러 파일을 포함합니다. Project Manager라는 프로젝트 관리 도구에서 폼, 유닛, 리소스, 객체, 라이브러리 파일과 같은 파일들을 보면서 구성할 수 있습니다. Project Manager를 표시하려면 View | Project Manager를 선택하십시오.



Project Manager를 사용하면 관련 프로젝트에 대한 정보를 단일 프로젝트 그룹으로 결합하고 표시할 수 있습니다. 여러 실행 파일 같은 관련 프로젝트를 하나의 그룹으로 구성하여 동시에 컴파일할 수 있습니다. 프로젝트 컴파일과 같은 프로젝트 옵션을 변경하려면 6-8페이지의 "디폴트 프로젝트 옵션 설정"을 참조하십시오.

**자세한 내용은...**

온라인 도움말 색인의 "Project Manager"를 참조하십시오.

## To-do list

To-do list는 프로젝트와 관련하여 해야 할 일들을 모아 놓은 레코드 항목입니다. 리스트에 프로젝트 범용 항목을 직접 추가하거나 소스 코드에서 특정 항목을 직접 추가할 수 있습니다. View | To-Do list를 선택하여 프로젝트와 연관된 정보를 추가하거나 표시합니다.



To-do list를 마우스 오른쪽 버튼으로 클릭하여 리스트를 정렬하고 필터링할 수 있는 명령을 표시합니다.

항목에 대한 작업이 끝나면 체크 박스를 클릭합니다.

### 자세한 내용은...

온라인 도움말 색인의 "to-do lists"를 참조하십시오.

## C++Builder로 프로그래밍

다음 단원에서는 프로젝트 작성, 폼 작업, 코드 작성, 애플리케이션 컴파일, 디버깅, 배포 및 국제화 그리고 개발할 수 있는 프로젝트 타입을 비롯하여 C++Builder를 사용한 소프트웨어 개발에 대한 개요를 다룹니다.

### 프로젝트 만들기

프로젝트는 디자인 타임 시 만들어지거나 프로젝트 소스 코드를 컴파일할 때 생성되는 파일 모음입니다. C++Builder를 처음 시작하면 새 프로젝트가 열립니다. 여러 파일 중에서 프로젝트 파일(Project1.dpr), 유닛 파일(Unit1.pas), 리소스 파일(Unit1.dfm, 또는 CLX 애플리케이션의 경우에는 Unit1.xfm)은 자동으로 생성됩니다.

프로젝트가 이미 열려 있는 상태에서 새로운 프로젝트를 열려면 File|New|Application 또는 File|New|Other를 선택하고 Application 아이콘을 더블 클릭합니다. File|New|Other를 선택하면 프로젝트에 추가할 다이얼로그 박스와 같이 미리 디자인된 템플릿 뿐만 아니라 추가적인 폼, 프레임, 모듈을 제공하는 Object Repository가 열립니다. Object Repository에 대한 자세한 내용은 2-5페이지의 "Object Repository"를 참조하십시오.

프로젝트를 시작할 때에는 애플리케이션 또는 DLL과 같은 개발하고자 하는 대상에 대해 알아야 합니다. C++Builder로 개발할 수 있는 프로젝트 타입을 보려면 3-9페이지의 "프로젝트 타입"을 참조하십시오.

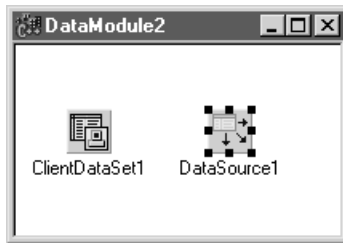
#### 자세한 내용은...

온라인 도움말 색인의 "projects"를 참조하십시오.

## 데이터 모듈 추가

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

데이터 모듈을 만들려면 File | New | Data Module을 선택하십시오. C++Builder는 코드 에디터에서 모듈에 대한 추가 유닛 파일을 표시하는 비어 있는 데이터 모듈을 열고, 그 모듈을 현재 프로젝트에 새 유닛으로 추가합니다. 폼에서와 동일한 방식으로 논비주얼(nonvisual) 컴포넌트를 데이터 모듈에 추가합니다.



데이터 모듈에 컴포넌트를 두려면 해당 논비주얼(nonvisual) 컴포넌트를 더블 클릭합니다.

기존 데이터 모듈을 다시 열면 C++Builder는 추가된 컴포넌트를 표시합니다.

### 자세한 내용은...

온라인 도움말 색인의 "data modules"를 참조하십시오.

## 사용자 인터페이스 생성

C++Builder를 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.

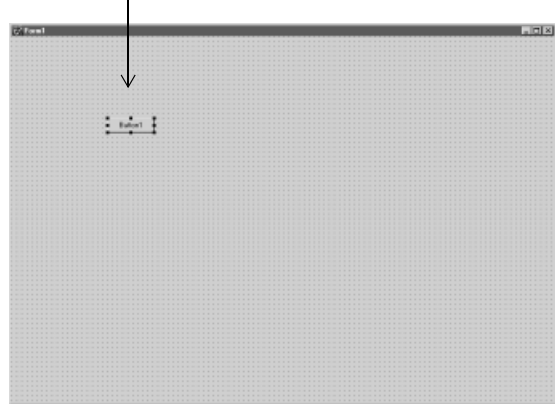
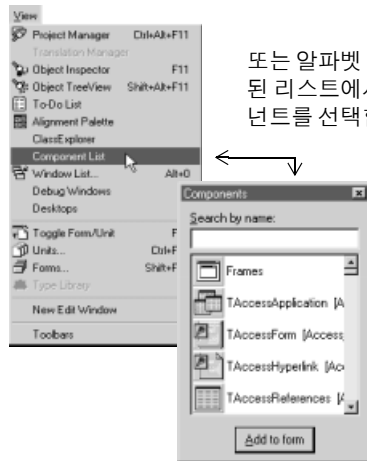
컴포넌트를 선택하고 폼의 원하는 곳으로 끌어 놓습니다.

컴포넌트 팔레트에서는 기능별로 그룹화된 많은 컴포넌트가 제공됩니다.

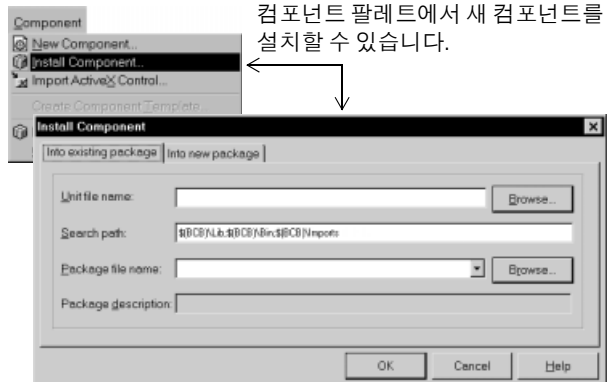
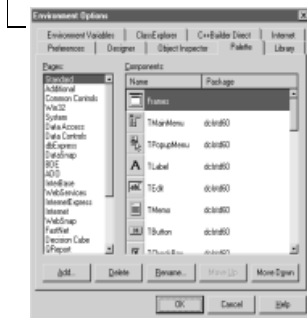


컴포넌트 팔레트에서 컴포넌트를 클릭합니다.

그런 다음 폼에서 컴포넌트를 두려는 곳을 클릭합니다.



또한 팔레트를 재정렬하고 새 페이지를 추가할 수 있습니다.  
Tools|Environment Options를 선택한 다음 Palette 페이지를 선택합니다.

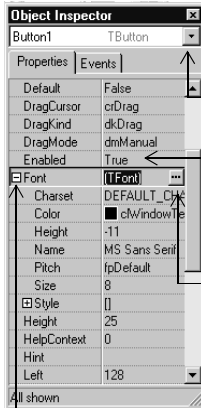


자세한 내용은...

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.

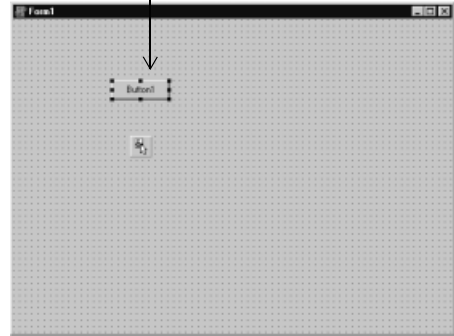


또는 이 드롭다운 리스트를 사용하여 객체를 선택합니다. 여기서는 Button1이 선택되어 있고 그 속성이 표시되어 있습니다.

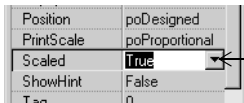
속성을 선택하고 오른쪽 열에서 값을 변경합니다. 생략 부호를 클릭하여 helper 객체의 속성을 변경할 수 있는 다이얼로그 박스를 엽니다.

+ 기호를 클릭하면 세부 리스트를 열 수 있습니다.

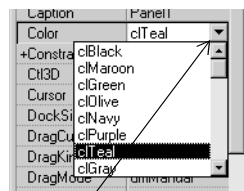
폼에 있는 컴포넌트나 객체를 클릭해서 선택할 수 있습니다.



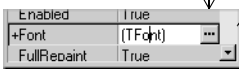
대부분의 속성은 색상 이름, *True* 또는 *False*, 정수와 같이 간단한 값을 가집니다. 부울 속성에서 *True*와 *False*를 토글하려면 더블 클릭합니다. 일부 속성은 연결된 속성 에디터를 가지고 있어서 더 복잡한 값을 설정할 수 있습니다. 이러한 속성 값을 클릭하면 생략 부호가 나타납니다. 크기와 같은 일부 속성에 대해서는 값을 입력합니다.



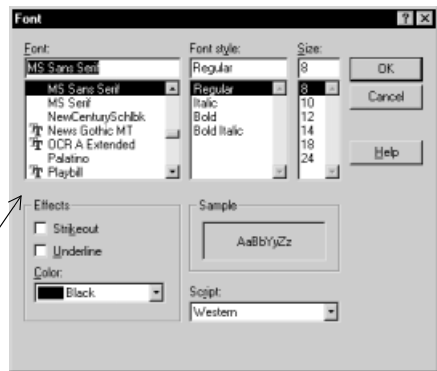
여기를 더블 클릭하여 *True*에서 *False*로 값을 변경합니다.



생략 부호를 클릭하여 이 속성에 대한 속성 에디터를 표시합니다.



아래쪽 화살표를 클릭하여 유효한 값 리스트에서 선택합니다.

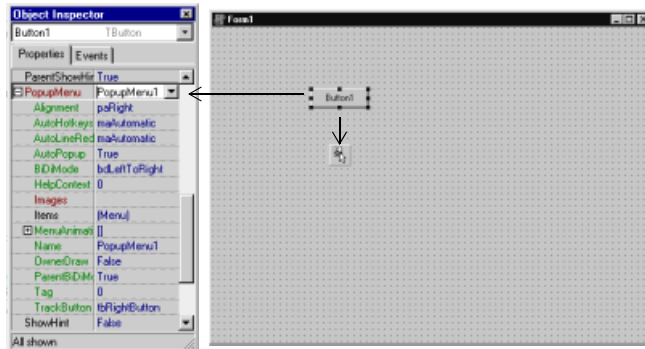


폼에서 둘 이상의 컴포넌트를 선택하면 Object Inspector는 선택한 컴포넌트가 공유하는 속성을 모두 표시합니다.

또한 **Object Inspector**는 확장된 인라인 컴포넌트 참조를 지원합니다. 따라서 참조된 컴포넌트 자체를 선택할 필요 없이 참조된 컴포넌트의 속성과 이벤트를 액세스할 수 있습니다. 예를 들어, 버튼 및 팝업 메뉴 컴포넌트를 폼에 추가할 경우에 **Button** 컴포넌트를 선택하면, **Object Inspector**에서 *PopupMenu* 속성을 팝업 메뉴의 모든 속성을 표시하는 *PopupMenu1*로 설정할 수 있습니다.

Button 컴포넌트의 *PopupMenu* 속성을 *PopupMenu1*로 설정합니다. 이제 + 기호를 클릭하면 팝업 메뉴의 모든 속성이 나타납니다.

인라인 컴포넌트 참조는 빨간색으로 표시되고 그 하위 속성은 녹색으로 표시됩니다.



### 자세한 내용은...

온라인 도움말 색인의 "Object Inspector"를 참조하십시오.

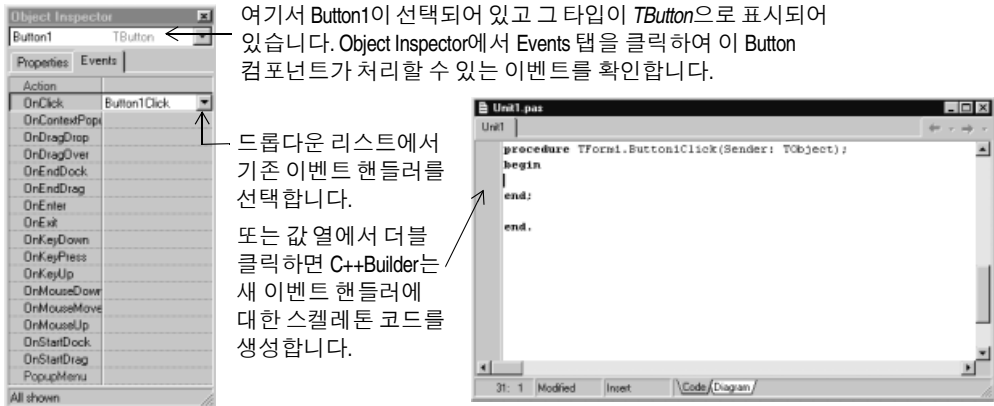
## 코드 작성

모든 애플리케이션의 핵심 부분은 각 컴포넌트의 코드 부분입니다. C++Builder의 RAD 환경에서는 미리 설치된 비주얼(visual) 및 논비주얼(nonvisual) 컴포넌트와 같은 빌딩 블록 대부분을 제공하지만 이벤트 핸들러, 메소드 및 일부 고유한 클래스는 개발자가 직접 작성해야 합니다. C++Builder의 VCL 및 CLX 클래스 라이브러리에 있는 수천 개의 객체에서 원하는 객체를 선택하여 이러한 작업을 쉽게 수행할 수 있습니다. 소스 코드를 사용하려면 2-6페이지의 "코드 에디터"를 참조하십시오.

### 이벤트 핸들러 작성

코드는 런타임 시 컴포넌트에 발생하는 이벤트에 응답해야 합니다. 이벤트는 버튼을 누르는 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 미리 정의된 이벤트 핸들러를 보려면, 컴포넌트를 선택하고 Object Inspector에서 Events 탭을 클릭하십시오.



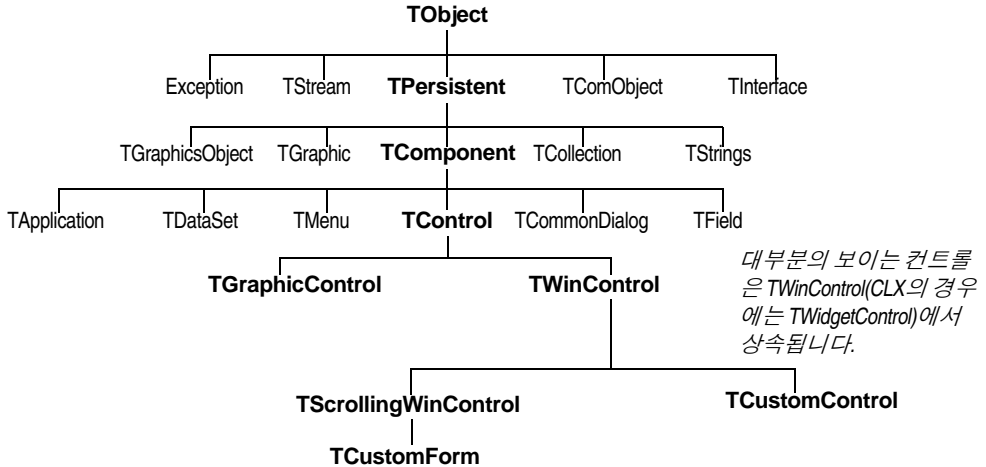
#### 자세한 내용은...

온라인 도움말 색인의 "events"를 참조하십시오.

### VCL 및 CLX 라이브러리 사용

C++Builder에는 여러 객체로 구성된 두 개의 클래스 라이브러리가 있으며, 이러한 객체 중에는 코드를 작성할 때 사용하는 컴포넌트나 컨트롤도 있습니다. Windows 애플리케이션에는 비주얼 컴포넌트 라이브러리(VCL)를 사용하고 Windows 및 Linux 애플리케이션에는 Borland 크로스 플랫폼용 컴포넌트 라이브러리(CLX)를 사용할 수 있습니다. 이러한 라이브러리에는 편집 컨트롤, 버튼 및 다른 사용자 인터페이스 요소와 같은 런타임에 보이는 객체 뿐만 아니라 데이터셋 및 타이머와 같은 보이지 않는 컨트롤도 포함되어 있습니다. 다음 다이어그램은 VCL을 구성하는 일부 기본 클래스를 보여 줍니다. CLX 계층도 이와 유사합니다.





**TComponent**의 자손 객체에는 컴포넌트 팔레트에 설치할 수 있고 C++Builder 폼과 데이터 모듈에 추가할 수 있는 속성과 메소드가 있습니다. VCL 및 CLX 컴포넌트가 IDE로 연결되기 때문에 폼 디자이너와 같은 도구를 사용하여 애플리케이션을 신속하게 개발할 수 있습니다.

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

대부분의 C++Builder 에디션에는 VCL 및 CLX 소스 코드가 들어 있습니다.

### 자세한 내용은...

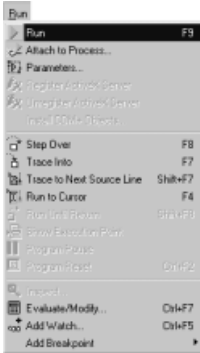
도움말 내용의 "Visual Component Library Reference" 및 "CLX Reference"의 온라인 도움말 색인의 "VCL"을 참조하십시오.

## 프로젝트 컴파일 및 디버깅

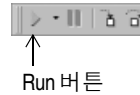
코드를 작성하고 나면 프로젝트를 컴파일하고 디버깅해야 합니다. C++Builder를 사용하면, 먼저 프로젝트를 컴파일한 다음 따로 디버깅하거나 아니면 통합 디버거를 사용하여 컴파일과 디버깅을 동시에 수행할 수 있습니다. 디버깅 정보로 프로그램을 컴파일하려면, **Project|Options**를 선택하고 **Compiler** 페이지를 클릭한 다음 **Debug information**이 선택되어 있는지 확인하십시오.

C++Builder는 통합 디버거를 사용하므로 프로그램 실행을 제어하고, 변수를 관찰하고, 데이터 값을 수정할 수 있습니다. 코드를 한 줄씩 작성해 나가면서 각 브레이크포인트에서 프로그램 상태를 확인할 수 있습니다. 통합 디버거를 사용하려면, **Tools|Debugger Options**를 선택하고 **General** 페이지를 클릭한 다음 **Integrated debugging**이 선택되어 있는지 확인하십시오.

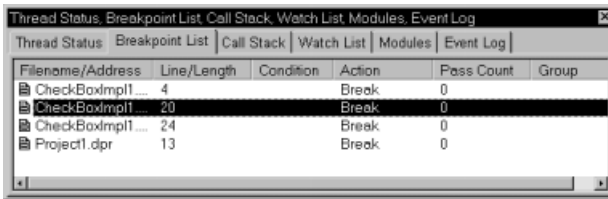
IDE에서는 **Debug** 툴바의 **Run** 버튼을 클릭하거나, **Run | Run**을 선택하거나, **F9** 키를 눌러 디버깅 세션을 시작할 수 있습니다.



Run 메뉴에서 디버깅 명령을 하나 선택합니다. 일부 명령은 툴바에도 있습니다.



통합 디버거를 사용하면, **Breakpoint**, **Call Stack**, **Watch**, **Local Variable**, **Thread**, **Module**, **CPU**, **Event Log** 등을 비롯한 많은 디버깅 윈도우를 사용할 수 있습니다. **View | Debug Windows**를 선택하여 디버깅 윈도우를 표시합니다. 몇몇 디버거 뷰는 일부 **C++Builder** 에디션에서 사용할 수 없습니다.



더 쉽게 사용하기 위하여 여러 디버깅 윈도우를 결합할 수 있습니다.

디버깅 윈도우를 결합하여 디버깅을 더 쉽게 하는 방법에 대한 자세한 내용은 6-2 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 6-4페이지의 "데스크탑 레이아웃 저장"을 참조하십시오.

### 자세한 내용은...

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

## 애플리케이션 배포

---

다른 사람들이 설치하고 실행할 수 있도록 애플리케이션을 배포할 수 있습니다. 애플리케이션을 배포할 때는 실행 파일, DLL, 패키지 파일 및 helper 애플리케이션과 같은 모든 필수 및 지원 파일이 필요합니다. C++Builder에는 이러한 파일이 포함된 설치 프로그램을 만들 수 있게 도와주는 InstallShield Express라는 설치 도구 키트가 들어 있습니다. InstallShield Express를 설치하려면 C++Builder 설치 화면에서 InstallShield Express Custom Edition for C++Builder를 선택하십시오.

### 자세한 내용은...

온라인 도움말 색인의 "deploying, applications"를 참조하십시오.

## 애플리케이션 국제화

---

C++Builder는 애플리케이션의 국제화 및 지역화를 위한 여러 기능을 제공합니다. IDE 및 VCL은 프로젝트를 국제화하는 데 필요한 IME와 확장 문자 집합을 지원합니다. C++Builder에는 다른 로케일에 대한 소프트웨어 지역화 및 동시 개발을 위한 일련의 번역 도구가 포함되어 있습니다. 모든 C++Builder 에디션에서 사용할 수 있는 것은 아니지만, 이러한 번역 도구를 사용하면 지역화된 여러 버전의 애플리케이션을 단일 프로젝트의 일부로 관리할 수 있습니다.

이 번역 도구에는 다음 세 가지 통합 도구가 포함되어 있습니다.

- 리소스 DLL을 생성 및 관리하는 DLL 마법사인 Resource DLL 마법사
- 번역된 리소스를 확인 및 편집하기 위한 테이블인 Translation Manager
- 번역된 리소스를 저장하는 공유 데이터베이스인 Translation Repository

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

### 자세한 내용은...

온라인 도움말 색인의 "international applications"를 참조하십시오.

## 프로젝트 타입

---

C++Builder의 모든 에디션은 범용 32비트 Windows 프로그래밍, DLL, 패키지, 사용자 정의 컴포넌트, 멀티 스레드, COM(Component Object Model) 및 Automation 컨트롤러, 멀티 프로세스 디버깅 등을 지원합니다. 일부 에디션은 웹 서버 애플리케이션, 데이터베이스 애플리케이션, COM 서버, 멀티 티어 애플리케이션, CORBA, 의사 결정 지원 시스템 등의 서버 애플리케이션을 지원합니다.

### 자세한 내용은...

현재 에디션에서 지원하는 도구를 보려면 [www.borland.com](http://www.borland.com)에서 기능 리스트를 참조하십시오.

## CLX 애플리케이션

---

C++Builder를 사용하면 Windows 및 Linux 운영 체제 모두에서 실행되는 크로스 플랫폼 32 비트 애플리케이션을 개발할 수 있습니다. Linux의 경우 Borland C++ 솔루션을 아직 사용할 수 없지만, 지금 C++Builder를 통해 애플리케이션을 개발하여 미리 준비할 수 있습니다. CLX 애플리케이션을 개발하려면 File|New|CLX Application을 선택하십시오. CLX 애플리케이션에서 사용할 수 있는 컴포넌트와 항목만 컴포넌트 팔레트와 Object Repository에 나타난다는 점을 제외하고 IDE는 일반적인 C++Builder 애플리케이션과 비슷합니다. C++Builder에서 지원되는 Windows 특정 기능은 Linux 환경으로 직접 이식되지 않습니다.

### 자세한 내용은...

크로스 플랫폼 애플리케이션을 개발하는 데 사용할 수 있는 컴포넌트를 보려면 온라인 도움말 내용의 "CLX Reference"를 참조하십시오.

## 웹 서버 애플리케이션

---

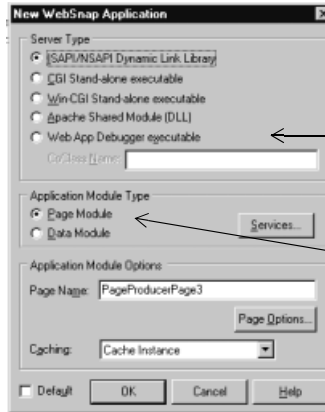
웹 서버 애플리케이션은 클라이언트의 요청을 처리하고 HTTP 메시지를 웹 페이지 형태로 반환하여 웹 서버에서 작동합니다. 사용 중인 C++Builder의 에디션에 따라 C++Builder는 데이터를 웹에 게시하기 위한 서로 다른 두 가지 기술을 제공합니다.

C++Builder의 가장 오래된 웹 서버 애플리케이션 기술을 Web Broker라고 합니다. Web Broker 애플리케이션은 요청을 디스패치하고, 동작을 수행하고, 웹 페이지를 사용자에게 반환할 수 있습니다. 애플리케이션의 비즈니스 로직은 대부분 애플리케이션 개발자가 작성한 이벤트 핸들러에 정의됩니다. WebBroker 웹 서버 애플리케이션을 만들려면 File|New|Other를 선택하고 Web Server Application 아이콘을 더블 클릭하십시오. Internet 및 InternetExpress 컴포넌트 팔레트 페이지에서 컴포넌트를 웹 모듈에 추가할 수 있습니다.

WebSnap은 어댑터, 추가 디스패처, 추가 페이지 프로듀서, 세션 지원, 웹 페이지 모듈 등의 추가 기능을 제공합니다. 이러한 추가 기능은 일반적인 웹 서버 애플리케이션 작업을 자동으로 처리하도록 디자인되었습니다. WebSnap 개발은 Web Broker 개발보다 좀더 비주얼하고 간단합니다. WebSnap 애플리케이션 개발자는 일반적인 페이지 전송 작업을 위한 이벤트 핸들러 작성에 드는 시간을 줄여 애플리케이션의 비즈니스 로직을 디자인하는 데 더 많은 시간을 투자할 수 있습니다. 새 WebSnap 서버 애플리케이션을 만들려면 File|New|Other를 선택하고 WebSnap 페이지를 클릭한 다음 Web Server Application 아이콘을 더블 클릭하십시오. WebSnap 컴포넌트 팔레트 페이지에서 WebSnap 컴포넌트를 추가할 수 있습니다.



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



웹 서버 애플리케이션을 디버깅하기 위한 테스트 서버를 비롯하여 다양한 웹 서버 애플리케이션 타입에서 실행할 애플리케이션을 만들 수 있습니다.

작업 시 HTML 페이지를 표시하기 위해 데이터 모듈을 사용할 것인지, 아니면 페이지 모듈을 사용할 것인지 선택합니다.

### 자세한 내용은...

온라인 도움말 색인의 "Web applications"를 참조하십시오.

## 데이터베이스 애플리케이션

C++Builder에서 제공하는 다양한 데이터베이스 및 연결 도구를 사용하면 데이터베이스 애플리케이션을 쉽게 개발할 수 있습니다.

데이터베이스 애플리케이션을 만들려면, 먼저 **Data Controls** 페이지 컴포넌트를 사용하여 폼에서 인터페이스를 디자인하십시오. 이어서 **Data Access** 페이지를 사용하여 데이터 소스를 데이터 모듈에 추가합니다. 그런 다음 다양한 데이터베이스 서버에 연결하기 위해 다음 연결 도구의 이전 또는 해당 페이지에서 데이터셋 및 데이터 연결 컴포넌트를 데이터 모듈에 추가합니다.

- **dbExpress**는 DB2, InterBase, MySQL 및 Oracle을 비롯한 SQL 데이터베이스 서버에 대한 빠른 액세스를 제공하는 크로스 플랫폼 애플리케이션용 데이터베이스 드라이버 모음입니다. dbExpress 드라이버를 사용하면 단방향 데이터셋을 사용하여 데이터베이스를 액세스할 수 있습니다.
- **BDE(Borland Database Engine)**는 dBASE, Paradox, FoxPro, Microsoft Access 및 모든 ODBC 데이터 소스를 비롯하여 널리 사용되는 데이터베이스 형식을 지원하는 드라이버 모음입니다. 일부 C++Builder 버전에서 사용할 수 있는 **SQL Links** 드라이버는 Oracle, Sybase, Informix, DB2, SQL Server 및 InterBase와 같은 서버를 지원합니다.
- **ADO(ActiveX Data Objects)**는 관계형 및 비관계형 데이터베이스, 전자 메일 및 파일 시스템, 텍스트 및 그래픽, 사용자 정의 비즈니스 객체 등을 비롯한 모든 데이터 소스에 대한 Microsoft의 고급 인터페이스입니다.
- **IBX(InterBase Express)** 컴포넌트는 사용자 정의 데이터 액세스 C++Builder 컴포넌트 아키텍처에 기초합니다. IBX 애플리케이션은 고급 InterBase 기능에 대한 액세스를 제공하고 InterBase 5.5 이후 버전의 최고 성능을 가진 컴포넌트 인터페이스를 제공합니다. IBX는 C++Builder의 데이터 인식 컴포넌트 라이브러리와 호환됩니다.

일부 데이터베이스 연결 도구는 몇몇 C++Builder 에디션에서 사용할 수 없습니다.

**자세한 내용은...**

온라인 도움말 색인의 "database applications"를 참조하십시오.

**BDE Administrator**

BDE Administrator(BDEAdmin.exe) 를 통해 BDE 드라이버를 구성하고 데이터 인식 VCL 컨트롤에서 사용하는 알리아스를 설정하여 데이터베이스에 연결합니다.

**자세한 내용은...**

Windows 시작 메뉴에서 프로그램 | Borland C++Builder | BDE Administrator를 선택한 다음 Help | Contents를 선택하십시오.

**SQL 탐색기(Database Explorer)**

SQL 탐색기(DBExplor.exe)는 데이터베이스를 탐색 및 편집할 수 있게 해 줍니다. 이 도구를 사용하면 데이터베이스 알리아스 작성, 스키마 정보 확인, SQL 쿼리 실행, 데이터 사전 및 어트리뷰트(attribute) 집합 유지 보수 등을 수행할 수 있습니다.

**자세한 내용은...**

C++Builder 메인 메뉴에서 Database | Explore를 선택한 다음 Help | Contents를 선택하십시오. 또는 온라인 도움말 색인의 "Database Explorer"를 참조하십시오.

**Database Desktop**

Database Desktop(DBD32.exe)을 사용하면 Paradox 및 dBase 데이터베이스 테이블을 다양한 형식으로 작성, 확인 및 편집할 수 있습니다.

**자세한 내용은...**

Windows 시작 메뉴에서 프로그램 | Borland C++Builder | Database Desktop을 선택한 다음 Help | User's Guide Contents를 선택하십시오.

**Data Dictionary**

BDE를 사용하면 Data Dictionary는 데이터 내용 및 모양을 설명하는 확장된 필드 어트리뷰트 집합을 만들 수 있는 사용자 정의 가능 저장소 영역을 애플리케이션에 상관 없이 제공합니다. Data Dictionary는 원격 서버에 상주하여 추가 정보를 공유할 수 있습니다.

**자세한 내용은...**

Help | C++Builder Tools를 선택하여 "Data Dictionary"를 참조하십시오.

## 사용자 정의 컴포넌트

---

C++Builder에 있는 컴포넌트는 컴포넌트 팔레트에 미리 설치되어 있으며 개발에 필요한 대부분의 기능을 제공합니다. 새 컴포넌트를 설치하지 않고도 C++Builder로 프로그래밍을 할 수는 있지만, 간혹 특별한 문제를 해결하거나 사용자 정의 컴포넌트를 필요로 하는 특정한 동작을 나타내고 싶은 경우도 있을 것입니다. 사용자 정의 컴포넌트는 애플리케이션 간에 코드를 재사용하거나 일관성을 유지하는 데 유용합니다.

서드파티의 사용자 정의 컴포넌트를 설치하거나 사용자 정의 컴포넌트를 직접 만들 수 있습니다. 새 컴포넌트를 만들려면 **Component|New Component**를 선택하여 **New Component** 마법사를 표시하십시오. 서드파티에서 제공하는 컴포넌트를 설치하려면 6-6페이지의 "컴포넌트 패키지 설치"를 참조하십시오.

### 자세한 내용은...

개발자 안내서의 4부 "사용자/지정 컴포넌트 만들기"와 온라인 도움말 색인의 "components, creating"를 참조하십시오.

## DLL

---

동적 연결 라이브러리(DLL)는 애플리케이션 및 다른 DLL에서 호출할 수 있는 루틴이 포함된 컴파일된 모듈입니다. DLL에는 대개 두 개 이상의 애플리케이션에서 사용하는 코드 또는 리소스가 포함되어 있습니다. **File|New|Other**를 선택하고 **DLL Wizard** 아이콘을 더블 클릭하여 DLL의 템플릿을 만듭니다.

### 자세한 내용은...

온라인 도움말 색인의 "DLLs"를 참조하십시오.

## COM과 ActiveX

---

C++Builder는 Microsoft의 COM 표준을 지원하고 ActiveX 컨트롤 작성을 위한 마법사를 제공합니다. **File|New|Other**를 선택하고 **ActiveX** 탭을 클릭하여 이러한 마법사를 액세스합니다. 예제 ActiveX 컨트롤은 컴포넌트 팔레트의 **ActiveX** 페이지에 설치됩니다. 컴포넌트 팔레트의 **Servers** 탭에는 많은 COM 서버 컴포넌트가 들어 있습니다. 이러한 컴포넌트를 VCL 컴포넌트인 것처럼 사용할 수 있습니다. 예를 들어, **Microsoft Word** 컴포넌트 중 하나를 폼에 두는 방법으로 애플리케이션 인터페이스 내에서 **Microsoft Word**의 인스턴스를 불러올 수 있습니다.

### 자세한 내용은...

온라인 도움말 색인의 "COM"과 "ActiveX"를 참조하십시오.

## 타입 라이브러리

타입 라이브러리는 ActiveX 컨트롤 또는 서버에서 노출하는 데이터 타입, 인터페이스, 멤버 함수 및 객체 클래스에 대한 정보를 포함하는 파일입니다. 타입 라이브러리를 COM 애플리케이션이나 ActiveX 라이브러리에 포함하여 이러한 엔티티에 대한 정보를 다른 애플리케이션이나 프로그래밍 도구에서 사용할 수 있게 만듭니다. C++Builder는 타입 라이브러리의 작성 및 유지 보수를 위한 Type Library Editor를 제공합니다.

### 자세한 내용은...

온라인 도움말 색인의 "type libraries"를 참조하십시오.



## 텍스트 에디터 만들기 - 자습서

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

**참고** 이 자습서는 C++Builder의 모든 에디션에 사용할 수 있습니다.

### 새 애플리케이션 시작

새 애플리케이션을 시작하기 전에 다음과 같이 소스 파일을 저장할 디렉토리를 만듭니다.

- 1 C:\Program Files\Borland\CBuilder6\Projects 디렉토리에 TextEditor라는 디렉토리를 만듭니다.
- 2 File|New|Application을 선택하여 새 프로젝트를 시작하거나 C++Builder를 시작할 때 이미 열려 있는 디폴트 프로젝트를 사용합니다.

각 애플리케이션을 *프로젝트*라고 합니다. C++Builder를 시작하면 디폴트로 비어 있는 프로젝트가 생성되고 다음 파일이 자동으로 만들어집니다.

- *Project1.cpp*: 프로젝트와 연결된 소스 코드 파일. *프로젝트 파일*이라고 합니다.
- *Unit1.cpp*: 메인 프로젝트 폼과 연결된 소스 코드 파일. *유닛 파일*이라고 합니다.
- *Unit1.h*: 메인 프로젝트 폼과 연결된 헤더 파일. *유닛 헤더 파일*이라고 합니다.
- *Unit1.dfm*: 메인 프로젝트 폼에 대한 정보를 저장하는 리소스 파일. *폼 파일*이라고 합니다.

각 폼은 자체 유닛(*Unit1.cpp*), 헤더(*Unit1.h*) 및 폼(*Unit1.dfm*) 파일을 가집니다. 두 번째 폼을 만들면 두 번째 유닛(*Unit2.cpp*), 헤더(*Unit2.h*) 및 폼(*Unit2.dfm*) 파일이 자동으로 만들어집니다.

3 File|Save All을 선택하여 파일을 디스크에 저장합니다. Save As 다이얼로그 박스가 나타나면 다음과 같이 합니다.

- TextEditor 폴더로 이동합니다.
- 디폴트 이름인 Unit1.cpp를 사용하여 Unit1을 저장합니다.
- TextEditor.dpr이란 이름으로 프로젝트를 저장합니다. 실행 파일은 확장자가 .exe인 프로젝트 이름과 동일한 이름으로 지정됩니다.

나중에 File|Save All을 선택하여 작업을 다시 저장할 수도 있습니다.

프로젝트를 저장하면, C++Builder는 프로젝트 디렉토리에 다양한 추가 파일을 만듭니다. 이러한 파일은 삭제하면 안됩니다.



디폴트 폼에는 Maximize, Minimize 및 Close 버튼과 Control 메뉴가 있습니다.

지금 F9키를 눌러 폼을 실행하면 이러한 버튼이 모두 작동하는 것을 볼 수 있습니다.

디자인 모드로 돌아가려면 X를 클릭하여 폼을 닫으십시오.

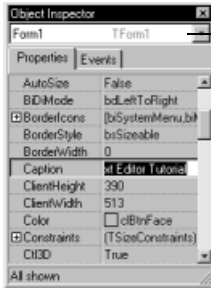
## 속성 값 설정

---

새 프로젝트를 열면 C++Builder는 디폴트로 *Form1*이라는 프로젝트의 메인 폼을 표시합니다. 이 폼에 컴포넌트를 두면 애플리케이션의 사용자 인터페이스와 다른 부분을 만들 수 있습니다.

폼 옆에는 폼과 폼 위에 둔 컴포넌트의 속성 값을 설정하는 데 사용할 수 있는 Object Inspector가 나타납니다. 속성을 설정할 때, C++Builder는 개발자를 대신해서 소스 코드를 유지 보수합니다. Object Inspector에서 설정하는 값은 *디자인 타임* 설정이라고 합니다.

- 1 Object Inspector에서 폼의 *Caption* 속성을 찾아서 디폴트 캡션인 *Form1* 대신에 *Text Editor Tutorial*을 입력합니다. 입력한 내용에 따라 폼의 헤더에 있는 캡션이 바뀌는 것에 주의하십시오.



Object Inspector의 상단에 있는 드롭다운 리스트는 현재 선택된 컴포넌트를 보여 줍니다. 이 경우, 컴포넌트는 *Form1*이고 그 타입은 *TForm1*입니다.

컴포넌트가 선택되면 Object Inspector는 그 속성을 표시합니다.

- 2 폼에 컴포넌트가 없더라도 F9 키를 눌러 지금 폼을 실행합니다.
- 3 *Form1*의 디자인 타임 뷰로 돌아가려면 다음 중 하나를 수행하십시오.
  - 애플리케이션의 제목 표시줄 오른쪽 위 모서리에 있는 **X**를 클릭합니다(폼의 런타임 뷰).
  - 제목 표시줄의 왼쪽 위 모서리에 있는 애플리케이션 종료 버튼을 클릭하고 **Close**를 클릭합니다.
  - **View | Forms**를 선택하고 *Form1*을 선택한 다음 **OK**를 클릭합니다.
  - 또는 **Run | Program Reset**을 선택합니다.

## 폼에 컴포넌트 추가

폼에 컴포넌트를 추가하기 전에, 애플리케이션에서 사용할 사용자 인터페이스(UI)를 만드는 최상의 방법에 대해 고려해야 합니다. UI는 사용자가 애플리케이션과 상호 작용할 수 있게 해 주는 것으로 쉽게 사용할 수 있도록 디자인해야 합니다.

C++Builder에는 애플리케이션의 부분을 나타내는 많은 컴포넌트가 포함되어 있습니다. 예를 들어, 컴포넌트 팔레트에는 메뉴, 툴바, 다이얼로그 박스, 그리고 그 밖의 여러 비주얼(visual) 및 논비주얼(nonvisual) 프로그램 요소를 쉽게 프로그램할 수 있는 컴포넌트(각각에서 파생됨)가 있습니다.

텍스트 에디터 애플리케이션에는 편집 영역, 편집 중인 파일 이름과 같은 정보를 표시하는 상태 표시줄, 메뉴, 명령에 쉽게 액세스할 수 있는 버튼이 있는 툴바 등이 필요합니다.

C++Builder를 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 텍스트 영역과 상태 표시줄을 폼에 추가하십시오.

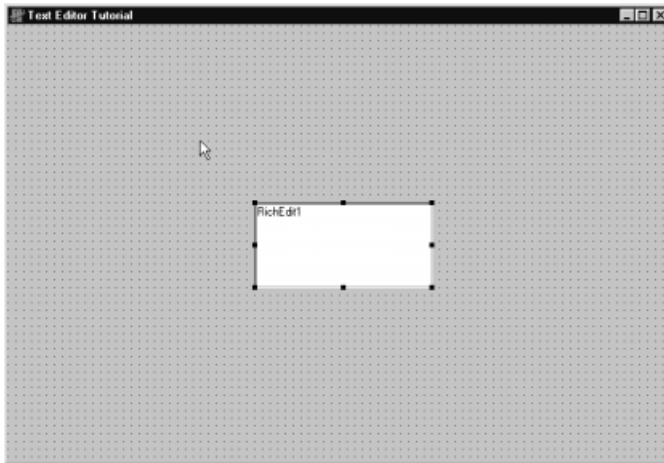


- 1 텍스트 영역을 만들려면 먼저 *RichEdit* 컴포넌트를 추가하십시오. *RichEdit* 컴포넌트를 찾으려면, 컴포넌트 팔레트의 **Win32** 페이지에서 팔레트의 아이콘을 잠시 가리키십시오. 그러면 C++Builder는 컴포넌트의 이름을 보여 주는 도움말 툴팁을 표시합니다.



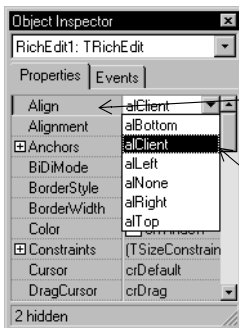
*RichEdit* 컴포넌트를 찾으면, 다음 중 하나를 수행합니다.

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



각 C++Builder 컴포넌트는 *클래스*이며, 폼에 컴포넌트를 두는 것은 그 클래스의 *인스턴스*를 만드는 것입니다. 일단 컴포넌트가 폼에 있으면, C++Builder는 애플리케이션이 실행 중일 때 객체의 인스턴스를 만드는 데 필요한 코드를 생성합니다.

- 2 *RichEdit* 컴포넌트를 선택하고 Object Inspector에서 *Align* 속성의 드롭다운 화살표를 클릭하여 *alClient*로 설정합니다.



폼에서 *RichEdit1* 컴포넌트가 선택되었는지 확인합니다.

Object Inspector에서 *Align* 속성을 찾습니다. 아래쪽 화살표를 클릭해서 속성의 드롭다운 리스트를 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 윈도우도 채울 수 있게 *RichEdit* 컨트롤의 크기가 변합니다.




- 3 컴포넌트 팔레트의 Win32 페이지에서 *StatusBar* 컴포넌트를 더블 클릭하여 상태 표시줄을 폼 하단에 추가합니다.

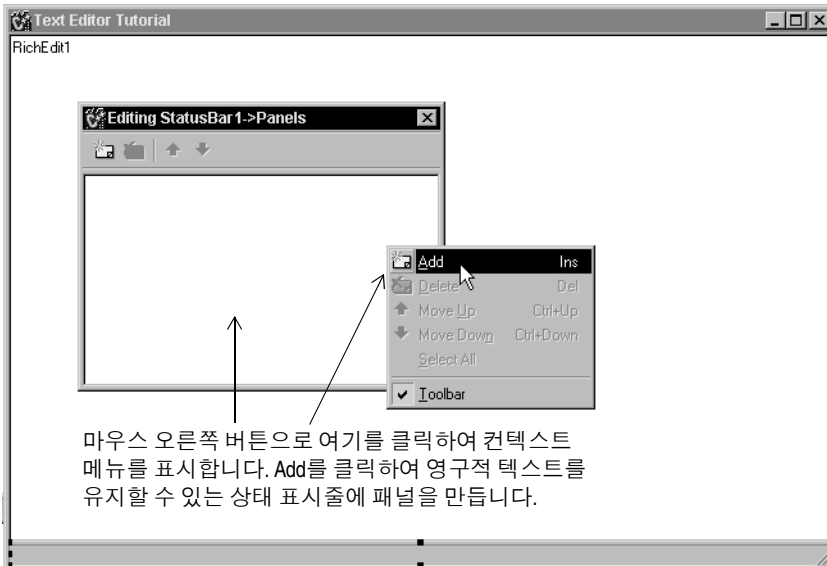


- 4 상태 표시줄을 더블 클릭하여 *Editing StatusBar1->Panels* 다이얼로그 박스를 엽니다.

팁

상태 표시줄의 *Panels* 속성에서 (*TStatusBar*) 생략 버튼을 클릭하여 *Editing StatusBar1->Panels* 다이얼로그 박스를 열 수도 있습니다.

- 5 이 다이얼로그 박스의 툴바에서 *Add New* 버튼  을 클릭하거나 마우스 오른쪽 버튼을 클릭하고 *Add*를 선택하여 패널을 상태 표시줄에 추가합니다. 패널은 텍스트 에디터에서 편집하는 파일의 경로와 이름을 표시합니다.



- 6 *Object Inspector*에서 *Text* 속성을 *untitled.txt*로 설정합니다. 텍스트 에디터를 사용할 때, 편집하는 파일을 아직 저장하지 않은 경우 파일 이름은 *untitled.txt*가 됩니다.

- 7 **X**를 클릭하여 *Editing StatusBar1->Panels* 다이얼로그 박스를 닫습니다.

이제 텍스트 에디터에 대한 사용자 인터페이스의 메인 편집 영역이 설정되었습니다.

## 메뉴와 툴바에 대한 지원 추가

애플리케이션으로 어떤 작업을 하려면 메뉴, 명령, 그리고 편리하게 사용하기 위한 툴바가 필요합니다. 명령을 별도로 코딩할 수 있지만, C++Builder는 메뉴 명령과 툴바 버튼의 액션, 이미지 및 코드를 한 곳에서 관리할 수 있는 *Action List* 또는 *Action Manager*를 제공합니다.

규칙에 따라 메뉴 명령에 연결된 액션에는 상위 레벨 메뉴 이름과 명령 이름이 있습니다. 예를 들어, FileExit 액션은 File 메뉴의 Exit 명령을 의미합니다. 다음 표는 텍스트 에디터 애플리케이션에 필요한 메뉴 명령의 종류와 액션에 연결된 툴바 버튼이 있는지 여부를 나열한 것입니다.

**표 4.1** 텍스트 에디터 명령 계획

메뉴	명령	툴바에 포함	설명
File	New	예	새 파일을 만듭니다.
File	Open	예	편집하기 위해 기존 파일을 엽니다.
File	Save	예	현재 파일을 디스크에 저장합니다.
File	Save As	아니오	파일을 새 이름 또는 지정된 이름으로 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 항목을 액세스할 수 있는 도움말 내용 화면을 표시합니다.
Help	Index	아니오	도움말 색인 화면을 표시합니다.
Help	About	아니오	애플리케이션에 대한 정보를 박스에 표시합니다.

## Action Manager Editor와 Action List Editor의 차이점

C++Builder의 에디션에 따라 메뉴와 툴바의 액션 및 이미지를 관리하는 두 가지 방법이 있습니다. 모든 C++Builder 에디션에는 사용자 명령에 대한 응답을 한 곳에서 관리할 수 있게 해 주는 *Action List Editor*가 포함되어 있습니다. *Action List Editor*는 Borland 크로스 플랫폼용 컴포넌트 라이브러리(CLT)의 일부이며, 나중에 Linux와 같은 다른 플랫폼으로 마이그레이션할 가능성이 있다면 *Action Manager Editor* 대신 이 에디터를 사용해야 합니다.

*Action Manager Editor*는 몇 가지 특수한 기능을 제공하지만, Windows 플랫폼에 한정된 비주얼 컴포넌트 라이브러리(VCL)의 일부로만 사용할 수 있습니다. *Action Manager Customize 다이얼로그 박스*를 사용하면 최종 사용자가 사용자 정의할 수 있고 일부 Microsoft Office 속성(예: 거의 사용하지 않는 메뉴 항목을 숨기는 기능)을 가진 메뉴 액션을 제공할 수 있습니다. 또한 *Action Manager Customize 다이얼로그 박스*에서 폼에 있는 메뉴 컴포넌트로 액션을 간단하게 끌어 놓을 수 있기 때문에 *Action Manager*는 보다 신속한 개발 과정을 제공합니다.

**경고** 이 자습서에서는 C++Builder의 기업용 및 전문가용 에디션에 쓰이는 *Action Manager Customize* 다이얼로그 박스가 사용됩니다. **기업용** 또는 **전문가용** 에디션이 있는 경우에는 아래의 "메뉴 및 툴바 이미지 추가(기업용 및 전문가용)"를 계속 진행합니다.

개인용 에디션이 있거나 Action List Editor를 사용하려면 4-13페이지의 "이미지 리스트 및 이미지 추가(개인용 에디션)"로 넘어가십시오.

## 메뉴 및 툴바 이미지 추가(기업용 및 전문가용)

이 단원에서는 액션 밴드에 사용할 이미지를 추가합니다.

일반적으로 *ImageList1* 컴포넌트를 폼에 추가하고 새로운 이미지를 임포트합니다. 이 자습서에서는 시간을 절약하기 위해 C++Builder IDE를 만드는 데 사용했던 이미지 리스트를 임포트합니다. 새로운 그래픽을 추가하지 않으면 컴포넌트 팔레트의 *ImageList1*은 표준 액션에 디폴트 이미지를 사용합니다.

다음과 같은 방법으로 기존의 이미지 리스트를 추가합니다.

- 1 C++Builder를 디폴트 디렉토리에 설치했다면 File|Open을 선택하고 C:\Program Files\Borland\CBuilder6\Source\vcl\actnres.pas를 선택합니다. 이 파일을 보려면 Open 다이얼로그 박스에서 Files of type:을 (Any file. \*.\*)로 설정하십시오.
- 2 *ImageList1* 컴포넌트를 선택 및 복사하여 폼에 붙여넣습니다. 이 컴포넌트는 논비주얼(nonvisual)이므로 어느 곳에도 붙여넣을 수 있습니다.
- 3 StandardActions 윈도우를 닫습니다.
- 4 *ImageList1* 컴포넌트를 더블 클릭하여 사용할 수 있는 모든 이미지를 표시합니다.

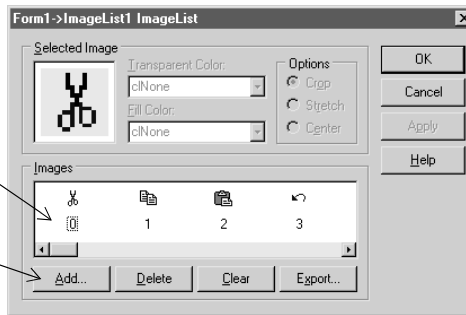


### 참고

*ImageList1*을 복사하려면 이 컴포넌트를 마우스 오른쪽 버튼으로 클릭하고 Edit|Copy를 클릭하십시오. 폼에서 마우스 오른쪽 버튼을 클릭하고 Edit|Paste를 선택합니다.

이미지 아래의 번호는 각액션에 대한 이미지 인덱스 속성에 해당합니다.

Add 버튼을 클릭하여 다른 소스의 이미지를 추가할 수 있습니다.



추가된 표준 액션 및 *ImageList1*에 대한 이미지 인덱스 속성은 다음을 포함합니다.

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 리스트에 있는 이미지를 추가할 수 있습니다. 4-13페이지의 "이미지 리스트 및 이미지 추가(개인용 에디션)"를 참조하십시오. **OK**를 클릭하여 *ImageList1* 다이얼로그 박스를 닫습니다.

## Action Manager에 액션 추가(기업용 및 전문가용)

기업용 및 전문가용 에디션은 메뉴와 툴바에 액션을 쉽게 추가할 수 있게 만드는 **Action Manager**를 제공합니다. 우선 **Action Manager**를 추가한 다음 액션을 추가합니다.

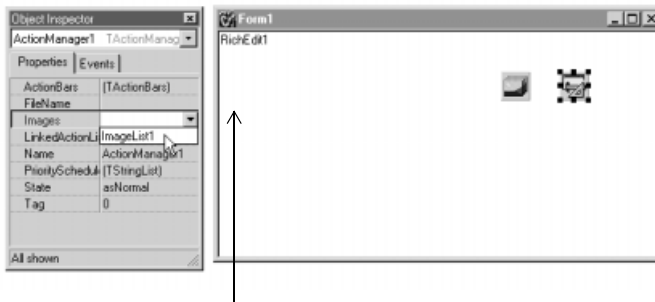


### 팁

1 컴포넌트 팔레트의 **Additional** 페이지에서 *ActionManager* 컴포넌트를 더블 클릭하여 폼에 놓습니다. 이 컴포넌트는 **nonvisual**(nonvisual)이므로 폼의 아무 곳이나 둘 수 있습니다.

폼에 놓은 **nonvisual** 컴포넌트의 캡션을 표시하려면 **Tools | Environment Options**를 선택하고 **Designer** 페이지를 클릭한 다음 **Show component captions**를 선택하고 **OK**를 클릭하십시오. 마우스로 컴포넌트를 가리켜도 컴포넌트의 이름이 표시됩니다.

2 폼에서 선택한 *ActionManager1*을 사용하여 **Object Inspector**에서 **Images** 속성을 *ImageList1*로 설정합니다.



**Images** 속성을 클릭한 다음 **Images** 옆의 아래쪽 화살표를 클릭합니다. *ImageList1*이 나열되면 이를 선택합니다. 이렇게 하면 이미지 리스트의 이미지가 액션 리스트에 있는 액션에 연결됩니다.



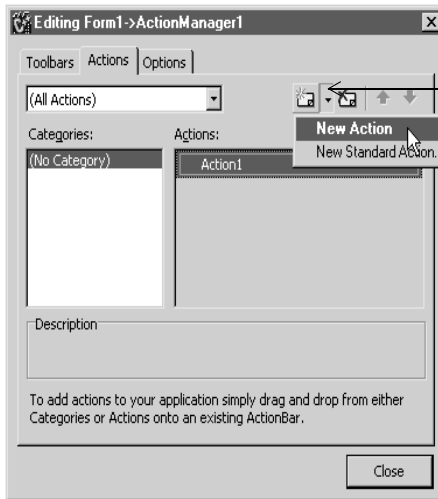
다음으로 액션을 **Action Manager**에 추가하고 그 속성을 설정합니다. 개발자가 모든 속성을 설정하는 비표준 액션과 자동으로 속성이 설정되는 표준 액션을 모두 추가합니다.

**3** *ActionManager* 컴포넌트를 더블 클릭하여 엽니다.

Editing Form1->ActionManager1 다이얼로그 박스 또는 Action Manager Editor 가 나타납니다.

**4** **Actions** 탭이 표시되는지 확인합니다. **New Action** 버튼 옆에 있는 드롭다운 화살표를 클릭하고 **New Action**을 클릭합니다.

**팁** Action Manager Editor 를 마우스 오른쪽 버튼으로 클릭하고 **New Action** 을 선택할 수도 있습니다.



New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하여 Action Manager의 새 액션을 만듭니다.

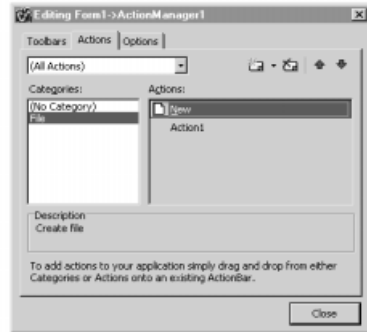
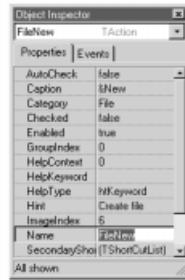
Delete 버튼이 활성화 되면 액션 리스트에서 기존 액션을 제거할 수 있습니다.

**5** 액션 리스트에서 *No Category*가 선택되었는지, 그리고 **Actions** 아래에 **Action1**이 선택되어 있는지 확인합니다. **Object Inspector**에서 다음 속성을 설정합니다.

- **Caption** 뒤에 **&New**를 입력합니다. 문자 앞에 앤퍼샌드(&)를 입력하면 명령에 액세스하는 바로 가기가 만들어집니다.
- **Category** 뒤에 **File**을 입력하면 **File** 명령이 만들어집니다.
- **Hint** 뒤에 **Create file**을 입력하면 도움말 툴팁이 됩니다.
- **ImageIndex**가 **6**으로 설정되어 있는지 확인합니다. 이것은 임포트한 이미지 리스트와 일치해야 합니다. 아래쪽 화살표를 클릭하여 적절한 이미지를 선택할 수도 있습니다.
- **Name** 뒤에 **FileNew**를 입력하고(**File|New** 명령) **Enter** 키를 눌러 변경 사항을 저장합니다.

Action ManagerEditor에서  
선택한 Action1을 사용  
하여 Object Inspector에서  
그 속성을 변경합니다.

*Caption*은 액션의 이름  
이고 *Category*는 액션의  
타입이며 *Hint*는 도움말  
툴팁입니다. *ImageIndex*  
를 사용하면 이미지 리  
스트에 있는 이미지를  
참조할 수 있습니다.  
*Name*은 코드에서 호출  
되는 액션입니다.



- 6 Editing Form1->ActionManager1 윈도우에서 File이 선택되었는지 확인합니다.  
New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하고 New Action을 선택합니다.
- 7 Object Inspector에서 다음 속성을 설정합니다.
  - *Caption* 뒤에 &Save를 입력합니다.
  - *Category*가 File로 설정되었는지 확인합니다.
  - *Hint* 뒤에 Save file을 입력합니다.
  - *ImageIndex* 뒤에서 image 8을 선택합니다.
  - *Name* 뒤에 FileSave를 입력합니다(File | Save 명령).
- 8 New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하고 New Action을 클릭합니다.
- 9 Object Inspector에서 다음 속성을 설정합니다.
  - *Caption* 뒤에 &Index를 입력합니다.
  - *Category* 뒤에 Help를 입력합니다.
  - *ImageIndex*는 필요하지 않습니다. 기본값을 그대로 둡니다.
  - *Name* 뒤에 HelpIndex를 입력합니다(Help | Index 명령).
- 10 New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하고 New Action을 클릭합니다.
- 11 Object Inspector에서 다음 속성을 설정합니다.
  - *Caption* 뒤에 &About을 입력합니다.
  - *Category*가 Help를 표시하는지 확인합니다.
  - *ImageIndex*는 필요하지 않습니다. 기본값을 그대로 둡니다.
  - *Name* 뒤에 HelpAbout을 입력합니다(Help | About 명령).
- 12 화면에서 Action Manager Customize 다이얼로그 박스를 닫지 않고 열어 둡니다.
- 13 File | Save All을 클릭하여 작업을 저장합니다.

## 표준 액션 추가(기업용 및 전문가용)

그런 다음 표준 액션(열기, 다른 이름으로 저장, 끝내기, 잘라내기, 복사, 붙여넣기 및 도움말 내용)을 Action Manager에 추가합니다.

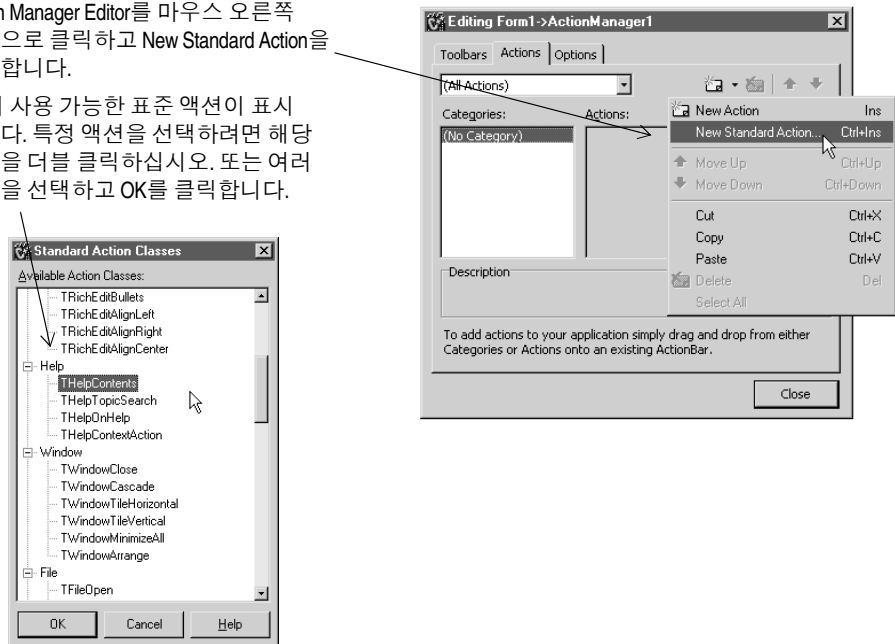
- 1 Action Manager Editor를 닫지 말고 계속 열어두어야 합니다. 이 에디터를 닫았을 경우 *ActionManager* 컴포넌트를 더블 클릭하여 에디터를 엽니다.
- 2 New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하고 New Standard Action을 클릭합니다.

Standard Action Classes 다이얼로그 박스가 나타납니다.

- 3 Edit 범주를 스크롤하면서 *Ctrl* 키를 사용하여 *TEditCut*, *TEditCopy* 및 *TEditPaste*를 선택합니다. OK를 클릭하여 이러한 액션을 *Editing Form1->ActionManager1* 다이얼로그 박스의 Categories 리스트에 있는 새 Edit 범주에 추가합니다.

Action Manager Editor를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 선택합니다.

이 때 사용 가능한 표준 액션이 표시됩니다. 특정 액션을 선택하려면 해당 액션을 더블 클릭하십시오. 또는 여러 액션을 선택하고 OK를 클릭합니다.



- 4 New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하고 New Standard Action을 클릭합니다.
- 5 File 범주를 스크롤하여 *TFileOpen*, *TFileSaveAs* 및 *TFileExit*를 선택합니다. OK를 클릭하여 이러한 액션을 File 범주에 추가합니다.
- 6 New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하고 New Standard Action을 클릭합니다.
- 7 Help 범주를 스크롤하여 *THelpContents*를 선택합니다. OK를 클릭하여 이 액션을 Help 범주에 추가합니다.

**참고** 사용자 정의 Help|Contents 명령은 항상 Help Contents 탭이 나타나는 도움말 파일을 표시합니다. 표준 Help|Contents 명령은 마지막으로 표시되었던 탭 모양의 Contents, Index 또는 Find 페이지를 불러옵니다.

이제 애플리케이션에 필요한 모든 표준 액션을 추가했습니다. 표준 액션은 이미지 인덱스를 비롯한 자체의 속성을 자동으로 설정합니다. 이미지 인덱스를 변경하여 다른 이미지를 표시할 수 있습니다.

- 8 원할 경우 표준 액션의 이미지를 변경할 수 있습니다. 예를 들어, Editing Form1->ActionManager1 다이얼로그 박스에서 File|Open 액션을 선택합니다. 이제 디폴트 이미지를 리스트에 있는 image 7로 변경합니다.
- 9 Close 버튼을 클릭하여 Action Manager Editor를 닫습니다.
- 10 File|Save All을 클릭하여 변경 사항을 저장합니다.

## 메뉴 추가(기업용 및 전문가용)

다음 두 단원에서는 사용자 정의 가능한 메뉴 표시줄과 도구 액션 밴드를 추가합니다. 텍스트 에디터 메뉴 표시줄에는 세 개의 드롭다운 메뉴, 즉 File, Edit 및 Help와 해당 메뉴 명령이 포함됩니다. Action Manager Customize 다이얼로그 박스를 사용하면 각 메뉴 범주와 그 명령을 동시에 메뉴 표시줄에 끌어 놓을 수 있습니다.



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

폼 상단에 비어 있는 메뉴 표시줄이 나타납니다.

- 2 아직 열지 않았다면 Action Manager Customize 다이얼로그 박스를 열고 Categories 리스트에서 File을 선택합니다. 하위 메뉴 명령이 원하는 순서로 되어 있지 않지만, Move Up 및 Move Down 버튼을 사용하거나 Ctrl+↑ 및 Ctrl+↓를 사용하여 이 순서를 쉽게 변경할 수 있습니다.
- 3 Open 액션을 선택하고 Action Manager Customize 다이얼로그 박스 톨바에서 Move Up 버튼을 클릭합니다. 그러면 File 명령이 New, Open, Save, Save As 및 Exit의 순서로 나열됩니다.
- 4 File을 메뉴 표시줄로 끌어 놓습니다. File 메뉴와 하위 메뉴 명령이 메뉴 표시줄에 나타납니다.

**팁** 메뉴 범주를 메뉴 표시줄에 끌어 놓은 후에 메뉴 명령을 재배치할 수도 있습니다. 예를 들어, 메뉴 표시줄에서 File을 클릭하여 하위 메뉴 명령을 표시하고 Open을 New 위로 끌어 놓은 다음 다시 원래 상태로 되돌릴 수 있습니다.

- 5 Action Manager Customize 다이얼로그 박스의 Categories 리스트에서 Edit를 메뉴 표시줄의 File 오른쪽에 끌어 놓습니다.
- 6 Action Manager Customize 다이얼로그 박스의 Categories 리스트에서 Help를 메뉴 표시줄의 Edit 오른쪽에 끌어 놓습니다.
- 7 Help 메뉴를 클릭하여 하위 메뉴 명령을 표시합니다. Contents 명령을 Index 명령 위로 끌어 놓습니다.
- 8 Esc 키를 누르거나 Help 메뉴를 다시 클릭하여 닫습니다.

9 File|Save All을 선택하여 변경 사항을 저장합니다.

이제 툴바를 추가하여 명령을 쉽게 액세스할 수 있습니다.

## 툴바 추가(기업용 및 전문가용)

Action Manager Customize 다이얼로그 박스에서 액션을 설정했으므로 메뉴에서 사용했던 것과 동일한 액션 중 일부를 액션 밴드 툴바에 추가할 수 있습니다. 작업이 끝나면 이 툴바는 Microsoft Office 2000 툴바와 비슷해집니다.



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

메뉴 표시줄 아래에 비어 있는 액션 밴드 툴바가 나타납니다.

### 팁

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

2 Action Manager Editor가 표시되지 않았다면 이 에디터를 열고 Categories 리스트에서 File 을 선택합니다. Actions 리스트에서 New, Open, Save 및 Exit를 선택하여 툴바로 끌어 놓습니다. 이러한 항목은 각각 할당된 이미지가 있는 버튼으로 자동 표시됩니다.

3 Action Manager Customize 다이얼로그 박스에서 Edit 범주를 툴바로 끌어 놓습니다. 모든 Edit 명령이 툴바에 나타나야 합니다.

### 참고

잘못된 명령을 툴바로 끌어 놓은 경우, 다시 제자리로 끌어 놓을 수 있습니다. 또는 Object TreeView에서 항목을 선택하고 Del 키를 누를 수도 있습니다. 버튼은 왼쪽이나 오른쪽으로 끌어 놓는 방법으로 간단하게 재배치할 수 있습니다.

4 File|Save All을 선택하여 변경 사항을 저장합니다.

5 F9 키를 눌러 프로젝트를 컴파일하고 실행합니다.

### 팁

또한 Debug 툴바의 Run 버튼을 클릭하거나 Run|Run을 선택하여 프로젝트를 실행할 수 있습니다. 프로젝트를 실행할 때, C++Builder 폼에서 디자인한 것과 비슷한 런타임 윈도우에서 프로그램을 엽니다.

텍스트 에디터에는 이미 많은 기능이 들어 있습니다. 텍스트 영역에서 텍스트를 선택하면 Cut, Copy 및 Paste 버튼이 작동해야 합니다. 일부 명령이 비활성 상태라도 메뉴와 툴바 버튼은 작동합니다. 일부 명령을 활성화하려면 이벤트 핸들러를 작성해야 합니다.

6 디자인 모드로 돌아가려면 애플리케이션의 오른쪽 위 모서리에 있는 X를 클릭하십시오.

자습서를 계속하려면 4-22페이지의 "텍스트 영역 지우기(모든 에디션)"로 넘어가십시오.

## 이미지 리스트 및 이미지 추가(개인용 에디션)

이 단원에서는 ImageList 컴포넌트를 폼에 추가하고 이미지를 이 리스트에 추가합니다. 컴포넌트 팔레트에서 ImageList 컴포넌트를 선택하고 폼을 클릭하여 ImageList1 컴포넌트를 추가합니다.

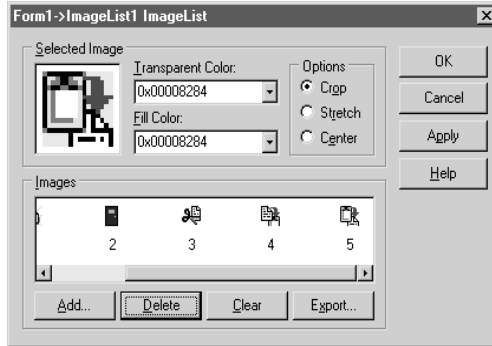
각 명령에 사용되는 이미지는 다음과 같습니다.

명령	이미지 이름	ImageIndex 속성
File New	Filenew.bmp	0
File Open	Fileopen.bmp	1
File Save	Filesave.bmp	2
File Exit	Doorshut.bmp	3
Edit Cut	Cut.bmp	4
Edit Copy	Copy.bmp	5
Edit Paste	Paste.bmp	6
Help Contents	Help.bmp	7

**참고** *ImageList*는 그래픽을 임포트하지 않고 디폴트 이미지를 표준 액션에 사용합니다.

다음과 같은 방법으로 이미지를 이미지 리스트에 추가합니다.

- 1 폼에서 *ImageList* 컴포넌트를 더블 클릭하여 Image List Editor를 표시합니다.
- 2 Add 버튼을 클릭합니다.
- 3 Add Images 다이얼로그 박스에서 제품이 설치되면서 만들어진 Buttons 디렉토리를 탐색합니다. 디폴트 위치는 C:\Program Files\Common Files\Borland Shared\Images\Buttons입니다.
- 4 filenew.bmp를 더블 클릭합니다.  
메시지에서 비트맵을 두 개의 비트맵으로 분류할 것인지를 물을 때마다 Yes를 클릭합니다. 각 아이콘은 이미지의 활성 버전과 비활성 버전을 포함합니다. 두 가지 이미지를 모두 볼 수 있습니다. 비활성(두 번째) 이미지를 삭제합니다.
- 5 다음과 같이 나머지 이미지를 추가합니다.
  - Add를 클릭하고 fileopen.bmp를 더블 클릭한 다음 비활성 이미지를 삭제합니다.
  - Add를 클릭하고 filesave.bmp를 더블 클릭한 다음 비활성 이미지를 삭제합니다.
  - Add를 클릭하고 doorshut.bmp를 더블 클릭한 다음 비활성 이미지를 삭제합니다.
  - Add를 클릭하고 cut.bmp를 더블 클릭한 다음 비활성 이미지를 삭제합니다.
  - Add를 클릭하고 copy.bmp를 더블 클릭한 다음 비활성 이미지를 삭제합니다.
  - Add를 클릭하고 paste.bmp를 더블 클릭한 다음 비활성 이미지를 삭제합니다.



**팁** 이미지를 클릭할 때 **Control** 키를 사용하면 여러 이미지를 선택할 수 있습니다. 그런 다음 비활성 이미지를 삭제합니다.

OK를 클릭하여 Image List Editor를 닫습니다.

이제 이미지 리스트에 7개의 이미지를 추가했고, 이러한 이미지에는 각 액션의 *ImageIndex* 속성과 일치하도록 0에서 6까지 번호를 매겼습니다.

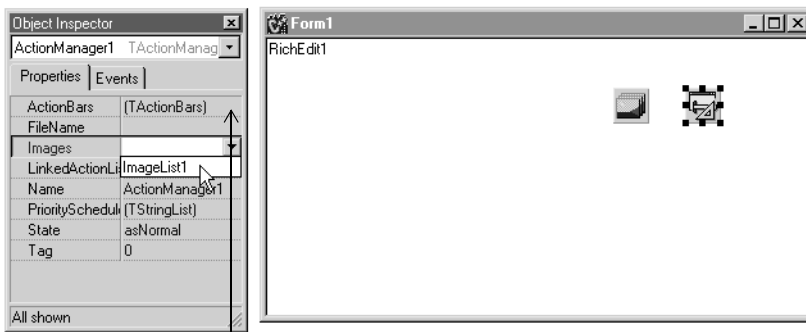
**참고** 순서가 맞지 않으면 Image List Editor에서 올바른 위치로 끌어 놓습니다.

이제 메뉴와 툴바를 추가할 준비가 되었습니다.

## 액션 리스트에 액션 추가(개인용 에디션)

*ImageList* 컴포넌트를 폼에 추가했고 이미지를 *ImageList*에 추가했습니다. 이 단원에서는 액션 리스트와 액션을 추가합니다.

- 1 컴포넌트 팔레트의 **Standard** 탭에서 *ActionList* 컴포넌트를 더블 클릭합니다. 폼에 추가되는 *ActionList1*은 논비주얼(nonvisual) 컴포넌트이므로 폼의 아무 곳이나 둘 수 있습니다.
- 2 *ActionList* 컴포넌트가 폼에서 선택된 상태로, 해당 *Images* 속성을 *ImageList1*로 설정합니다.

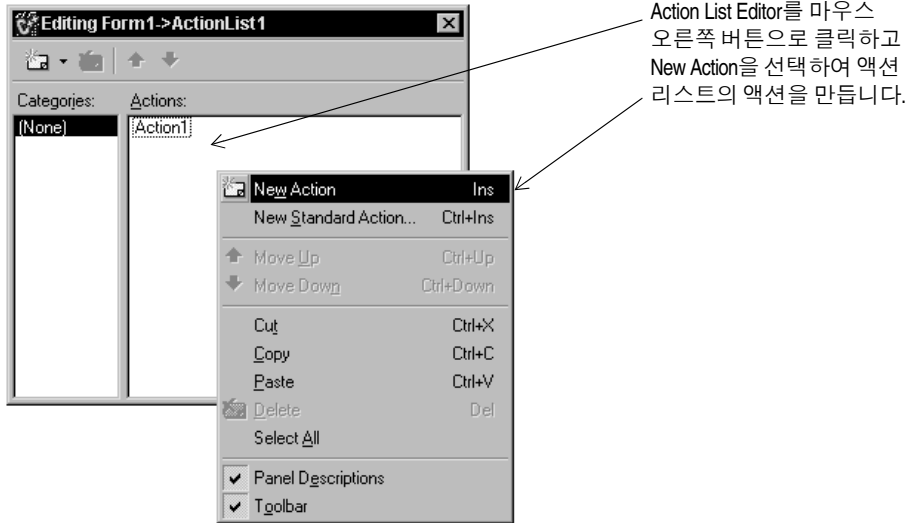


*Images* 속성을 클릭한 다음 *Images* 옆의 아래쪽 화살표를 클릭합니다. *ImageList1*이 나열되면 이를 선택합니다. 이렇게 하면 이미지 리스트에 추가할 이미지가 액션 리스트에 있는 액션에 연결됩니다.

3 *ActionList* 컴포넌트를 더블 클릭하여 엽니다.

Editing Form1->ActionList1 다이얼로그 박스가 나타납니다. 이것을 *Action List Editor*라고도 합니다.

4 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택합니다.



**팁** New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하고 New Action을 클릭할 수도 있습니다.

5 Object Inspector에서 Action1에 다음 속성을 설정합니다.

- *Caption* 뒤에 &New를 입력합니다. 문자 앞에 앰퍼샌드(&)를 입력하면 명령에 액세스하는 바로 가기가 만들어집니다.
- *Category* 뒤에 File을 입력하면 File 명령이 만들어집니다.
- *Hint* 뒤에 Create file을 입력하면 도움말 툴팁이 됩니다.
- *ImageIndex* 뒤에서 관련 이미지(위에서 설명한 순서대로 이미지 리스트를 추가한 경우에는 image 0)를 선택합니다.
- *Name* 뒤에 FileNew를 입력하고(File|New 명령) **Enter** 키를 눌러 변경 사항을 저장합니다.

6 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택합니다.

7 Object Inspector에서 Action1에 다음 속성을 설정합니다.

- *Caption* 뒤에 &Save를 입력합니다.
- *Category*가 File을 표시하는지 확인합니다.
- *Hint* 뒤에 Save file을 입력합니다.
- *ImageIndex* 뒤에서 관련 이미지(위에서 설명한 순서대로 이미지를 추가한 경우에는 image 2)를 선택합니다.
- *Name* 뒤에 FileSave를 입력합니다(File|Save 명령).



- 8 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택합니다.
  - 9 Object Inspector에서 Action1에 다음 속성을 설정합니다.
    - *Caption* 뒤에 &Index를 입력합니다.
    - *Category* 뒤에 Help를 입력합니다.
    - *ImageIndex*는 필요하지 않습니다. 기본값을 그대로 둡니다.
    - *Name* 뒤에 HelpIndex를 입력합니다(Help | Index 명령).
  - 10 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택합니다.
  - 11 Object Inspector에서 Action1에 다음 속성을 설정합니다.
    - *Caption* 뒤에 &About을 입력합니다.
    - *Category* 뒤에 Help를 입력합니다.
    - *ImageIndex*는 필요하지 않습니다. 기본값을 그대로 둡니다.
    - *Name* 뒤에 HelpAbout을 입력합니다(Help | About 명령).
- 화면에서 Action List Editor를 닫지 않고 열어 둡니다.

## 액션 리스트에 표준 액션 추가(개인용 에디션)

---

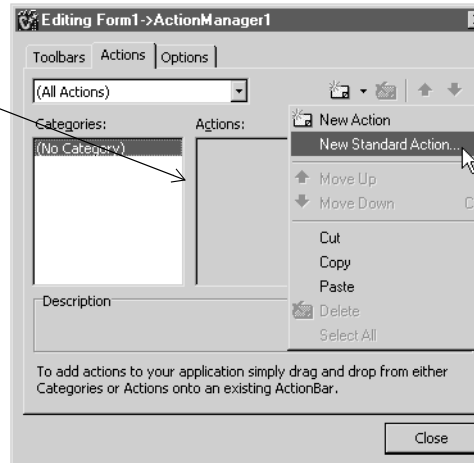
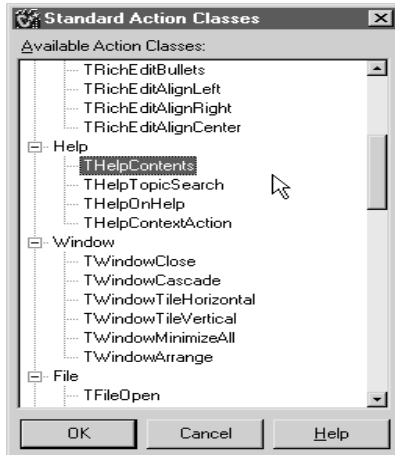
C++Builder는 애플리케이션 개발 시 자주 사용되는 여러 개의 표준 액션을 제공합니다. 이제 잘라내기, 복사 및 붙여넣기와 같은 표준 액션을 액션 리스트에 추가합니다.

- 1 Action List Editor를 닫지 말고 계속 열어두어야 합니다. 에디터를 닫았을 경우 폼에서 *ActionList* 컴포넌트를 더블 클릭합니다.
  - 2 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 클릭합니다.
- 팁** 또한 New Action 버튼 옆에 있는 드롭다운 화살표를 클릭하고 New Standard Action을 클릭할 수 있습니다.

- 3 Standard Action Classes 다이얼로그 박스에서 Edit 범주를 스크롤하면서 *Ctrl* 키를 사용하여 *TEditCut*, *TEditCopy* 및 *TEditPaste* 를 선택합니다. OK 를 클릭하여 이러한 액션을 Action List Editor의 새 Edit 범주에 추가합니다.

Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 선택합니다.

이 때 사용 가능한 표준 액션이 표시됩니다. 어떤 액션을 선택하려면 해당 액션을 더블 클릭하십시오.



- 4 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 클릭합니다.
- 5 File 범주를 스크롤하여 *TFileOpen*, *TFileSaveAs* 및 *TFileExit*를 선택합니다. OK를 클릭하여 이러한 액션을 File 범주에 추가합니다.
- 6 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 클릭합니다.
- 7 Help 범주를 스크롤하여 *THelpContents*를 선택합니다. OK를 클릭하여 이 액션을 Help 범주에 추가합니다.

#### 참고

사용자 정의 Help|Contents 명령은 항상 Help Contents 탭이 나타나는 도움말 파일을 표시합니다. 표준 Help|Contents 명령은 마지막으로 표시되었던 탭 모양의 Contents, Index 또는 Find 페이지를 불러옵니다.

표준 액션은 자체의 속성을 자동으로 설정합니다. 그러나 이미지 인덱스 속성을 변경하여 표준 액션을 개인용 에디션에서 제공하는 정확한 이미지에 연결해야 합니다.

- 8 Action List Editor의 Categories 리스트에서 (All Actions)를 선택합니다.
- 9 표준 액션에는 디폴트 이미지가 들어 있습니다. 이제 디폴트 이미지를 이전에 추가했던 이미지로 변경합니다. 액션 리스트에서 다음 액션을 한 번에 하나씩 선택하고 Object Inspector에서 해당 *ImageIndex* 속성을 변경합니다.
  - EditCut1을 선택하고 해당 *ImageIndex* 속성을 4로 설정합니다.

- EditCopy1을 선택하고 해당 *ImageIndex* 속성을 5로 설정합니다.
- EditPaste1을 선택하고 해당 *ImageIndex* 속성을 6으로 설정합니다.
- FileOpen1을 선택하고 해당 *ImageIndex* 속성을 1로 설정합니다.
- FileExit1을 선택하고 해당 *ImageIndex* 속성을 3으로 설정합니다.

10 X를 클릭하여 Action List Editor를 닫습니다.

11 File|Save All을 선택하여 변경 사항을 저장합니다.

## 메뉴 추가(개인용 에디션)

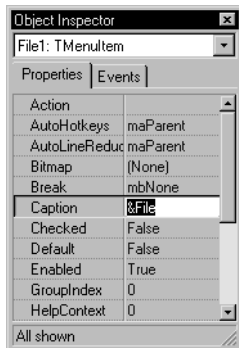
이 단원에서는, 세 가지 드롭다운 메뉴(File, Edit, Help)가 있는 메인 메뉴 표시줄을 추가하고 액션 리스트의 액션을 사용하여 메뉴 항목을 각 메뉴에 추가합니다.



- 1 컴포넌트 팔레트의 **Standard** 탭에서 **MainMenu** 컴포넌트를 폼에 가져다 놓습니다. 이 컴포넌트는 아무 곳이나 둘 수 있습니다.
- 2 메인 메뉴의 *Images* 속성을 *ImageList1*로 설정하면 이미지를 메뉴 명령에 추가할 수 있습니다.
- 3 **MainMenu** 컴포넌트를 더블 클릭하여 메뉴 디자인어를 엽니다.



- 4 메뉴 디자인어에서 첫 번째 상위 레벨 메뉴 항목을 설정하려면 **Object Inspector**에서 *Caption* 속성을 **&File**로 설정하고 **Enter** 키를 누르십시오.

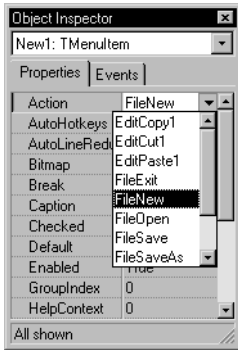


&File을 입력하고  
메뉴 디자인어에  
포커스를 맞추면,  
상위 레벨 File 명령  
이 나타나 첫 번째  
메뉴 항목을 추가  
할 수 있습니다.



- 5 방금 작성한 File 명령 아래에서 비어 있는 항목을 선택합니다.

- 6 Object Inspector에서 *Action* 속성을 FileNew로 설정합니다. 액션 리스트의 모든 액션이 나타납니다.

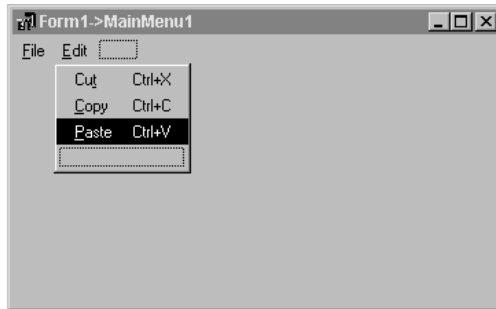
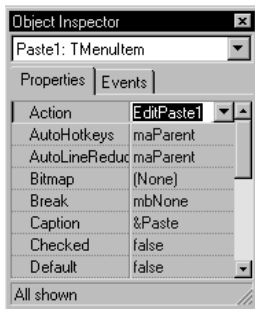


Action 속성 리스트에서 FileNew를 선택하면 올바른 Caption 및 ImageIndex와 함께 New 명령이 나타납니다.



- New 아래 항목을 선택하고 *Action* 속성을 FileOpen1로 설정합니다.
  - Open 아래 항목을 선택하고 *Action* 속성을 FileSave로 설정합니다.
  - Save 아래 항목을 선택하고 *Action* 속성을 FileSaveAs1로 설정합니다.
  - Save As 아래 항목을 선택하고 *Caption* 속성 뒤에 하이픈을 입력한 다음 *Enter* 키를 누릅니다. 그러면 메뉴에 구분자 표시줄이 만들어집니다.
  - 구분자 표시줄 아래 항목을 선택하고 *Action* 속성을 FileExit1로 설정합니다.
- 7 이어서 다음과 같이 Edit 메뉴를 만듭니다.

- File 명령 오른쪽에 있는 항목을 선택하고 *Caption* 속성을 &Edit로 설정한 다음 *Enter* 키를 누릅니다.
- 이제 포커스가 Edit 아래 항목에 있습니다. *Action* 속성을 EditCut1로 설정합니다.
- Cut 아래 항목을 선택하고 *Action* 속성을 EditCopy1로 설정합니다.
- Copy 아래 항목을 선택하고 *Action* 속성을 EditPaste1로 설정합니다.



- 8 다음과 같이 Help 메뉴를 만듭니다.

- Edit 명령 오른쪽에 있는 항목을 선택하고 *Caption* 속성을 &Help로 설정한 다음 *Enter* 키를 누릅니다.
- Help 아래 항목을 선택하고 *Action* 속성을 HelpContents로 설정합니다.
- Contents 아래 항목을 선택하고 *Action* 속성을 HelpIndex로 설정합니다.
- Index 아래 항목을 선택하고 *Caption* 속성 뒤에 하이픈을 입력한 다음 *Enter* 키를 눌러 Help 메뉴에 구분자 표시줄을 만듭니다.
- 구분자 표시줄 아래 항목을 선택하고 *Action* 속성을 HelpAbout로 설정합니다.

9 **X**를 클릭하여 메뉴 디자이너를 닫습니다.

10 File|Save All을 선택합니다.

11 디자인 모드로 돌아가려면 애플리케이션의 오른쪽 위 모서리에 있는 **X**를 클릭하십시오.

**참고** 폼이 사라져 버렸을 경우 View|Forms를 누르고 Form1을 선택한 다음 OK를 클릭합니다.

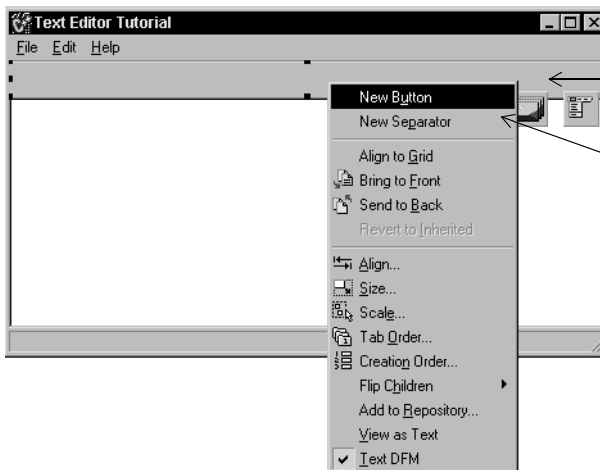
## 툴바 추가(개인용 에디션)

액션 리스트에서 액션을 설정했으므로, 메뉴에서 사용했던 것과 동일한 액션 중 일부를 툴바에 추가할 수 있습니다.



- 1 컴포넌트 팔레트의 Win32 탭에서 *ToolBar* 컴포넌트를 더블 클릭하여 폼에 추가합니다. 비어 있는 툴바는 메인 메뉴 아래에 추가됩니다.
- 2 툴바를 선택한 상태로 Object Inspector에서 다음 속성을 변경합니다.
  - *Images* 속성을 *ImageList1*로 설정합니다.
  - 툴바의 *Indent* 속성을 4로 설정합니다. 이렇게 하면 툴바 왼쪽의 아이콘을 4픽셀 들여 씁니다.
  - *ShowHint*를 *true*로 설정합니다. (팁: *false*를 더블 클릭하여 *true*로 변경합니다.)
- 3 버튼을 툴바에 추가하려면 마우스 오른쪽 버튼을 클릭하고 New Button을 네 번 선택하십시오.
- 4 구분자를 툴바에 추가하려면 마우스 오른쪽 버튼을 클릭하고 New Separator를 선택하십시오.
- 5 마우스 오른쪽 버튼을 클릭하고 New Button을 세 번 이상 선택합니다.

**참고** 이미지가 올바른지는 걱정할 필요가 없습니다. 액션을 버튼에 할당하면 올바른 이미지가 추가됩니다.



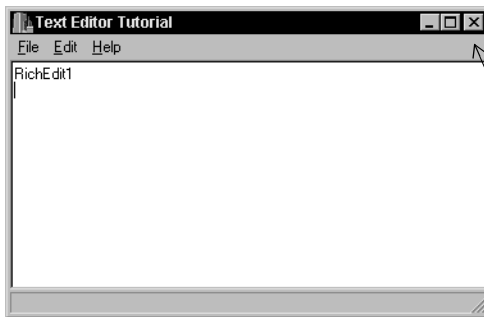
툴바 객체는 디폴트로 메뉴 아래에 추가됩니다.

버튼 또는 구분자를 추가하려면 툴바를 선택하고 마우스 오른쪽 버튼을 클릭한 다음 New Button 또는 New Separator를 선택하십시오. 그런 다음 액션 리스트에서 액션을 할당합니다.

- 6 액션 리스트의 액션을 첫 번째 버튼 집합에 할당합니다.
  - 첫 번째 버튼을 선택하고 *Action* 속성을 *FileNew*로 설정합니다.
  - 두 번째 버튼을 선택하고 *Action* 속성을 *FileOpen1*로 설정합니다.
  - 세 번째 버튼을 선택하고 *Action* 속성을 *FileSave*로 설정합니다.
  - 네 번째 버튼을 선택하고 *Action* 속성을 *FileExit1*로 설정합니다.
- 7 액션을 두 번째 버튼 집합에 할당합니다.
  - 다섯 번째 버튼을 선택하고 *Action* 속성을 *EditCut1*로 설정합니다.
  - 여섯 번째 버튼을 선택하고 *Action* 속성을 *EditCopy1*로 설정합니다.
  - 마지막 버튼을 선택하고 *Action* 속성을 *EditPaste1*로 설정합니다.
- 8 *File|Save All*을 선택합니다.
- 9 *F9* 키를 눌러 프로젝트를 컴파일하고 실행합니다.

#### 참고

또한 *Debug* 툴바의 *Run* 버튼을 클릭하거나 *Run|Run*을 선택하여 프로젝트를 실행할 수 있습니다.



프로젝트를 실행하기 위해 *F9* 키를 누르면 애플리케이션 인터페이스가 표시됩니다. 메뉴, 텍스트 영역, 상태 표시줄이 모두 폼에 나타납니다.

디자인 모드로 돌아가려면 **X**를 클릭하여 폼을 닫으십시오.

프로젝트를 실행할 때, *C++Builder*는 런타임 폼에서 디자인한 것과 비슷한 윈도우에서 프로그램을 엽니다. 대부분의 명령이 비활성 상태인 경우에도 모든 메뉴가 작동됩니다. 이미지는 이미지 인덱스와 연결한 메뉴 항목 옆에 나타납니다.

텍스트 에디터에는 이미 많은 기능이 들어 있습니다. 텍스트 영역 안에서 입력이 가능합니다. 텍스트 영역에서 텍스트를 선택하면 *Cut*, *Copy* 및 *Paste* 버튼이 작동해야 합니다.

- 10 오른쪽 위 모서리에 있는 **X**를 클릭하여 애플리케이션을 닫고 디자인 타임 뷰로 돌아갑니다.

## 텍스트 영역 지우기(모든 에디션)

**중요** 지금부터는 모든 에디션에 적용되는 내용입니다.

프로그램을 실행하면 *RichEdit1*이란 이름이 텍스트 영역에 나타납니다. *Strings List editor*를 사용하여 텍스트를 제거할 수 있습니다. 지금 텍스트를 지우지 않으면, 마지막 단계에서 메인 폼을 초기화할 때 텍스트를 제거해야 합니다.

다음과 같은 방법으로 텍스트 영역을 지웁니다.

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

이제 메인 폼이 표시되면 텍스트 편집 영역은 비워져 있습니다.

## 이벤트 핸들러 작성

지금까지는 코드를 작성하지 않고 애플리케이션을 개발했습니다. Object Inspector를 통해 디자인 타임에 속성 값을 설정하여 C++Builder RAD 환경의 장점을 최대한 살릴 수 있었습니다. 이 단원에서는 애플리케이션 실행 중에 사용자 입력에 응답하는 *이벤트 핸들러*라는 함수를 작성합니다. 메뉴와 툴바에 있는 항목에 이벤트 핸들러를 연결하므로, 항목이 선택되면 애플리케이션은 핸들러의 코드를 실행합니다.

비표준 액션의 경우, 이벤트 핸들러를 만들어야 합니다. File|Exit 및 Edit|Paste 명령과 같은 표준 액션의 경우에는 이벤트가 코드에 포함됩니다. 그러나 File|Save As 명령과 같은 일부 표준 액션에서는 고유한 이벤트 핸들러를 작성하여 명령을 사용자 정의할 수 있습니다.

모든 메뉴 항목과 툴바 액션이 Action Manager 또는 Action List Editor에 통합되므로, 이벤트 핸들러를 여기서 만들 수 있습니다.

**중요** C++Builder의 개인용 에디션이 설치되어 있다면 다음 단계에서 *ActionManager* 컴포넌트 대신 *ActionList* 컴포넌트를 사용합니다.

### New 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 New 명령에 대한 이벤트 핸들러를 만듭니다.

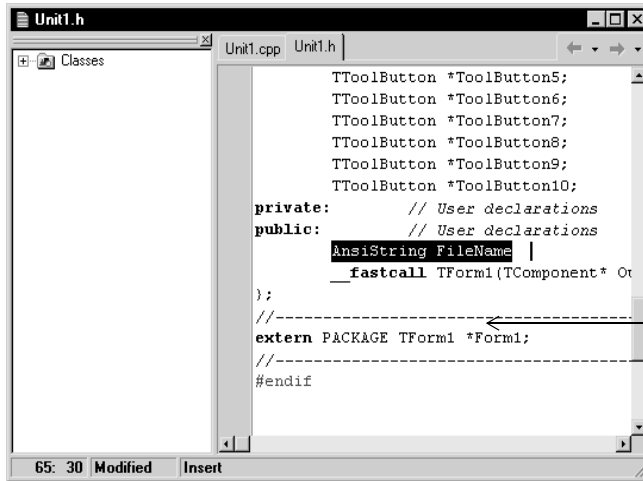
- 1 View|Units를 선택한 다음 Unit1을 선택하여 Form1과 연결된 코드를 표시합니다.
- 2 우선 이벤트 핸들러에서 사용할 파일 이름을 선언하고 이 파일 이름에 대한 사용자 정의 속성을 추가하여 다른 메소드에서 전역적으로 액세스할 수 있게 만들어야 합니다. 코드 에디터에서 마우스 오른쪽 버튼으로 Unit1.cpp 파일을 클릭하고 Open Source/Header File을 선택하거나 코드 에디터에서 Unit1.h 탭을 클릭하여 Unit1.h 파일을 엽니다. 헤더 파일에서 TForm1 클래스에 대한 공용 선언 부분을 두고

```
public:        // User declarations
```

아래 줄에 다음과 같이 입력합니다.

```
AnsiString FileName;
```

그러면 화면이 다음과 같이 나타납니다.



이 줄에서는 FileName을 다른 메소드에서 전역적으로 액세스할 수 있는 문자열로 정의합니다.

3 F12 키를 눌러 메인 폼으로 돌아갑니다.

**팁** F12 키를 토글하면 폼과 관련 코드 간에 전환할 수 있습니다. 또한 View | Forms를 선택하고 Form1을 선택할 수 있습니다.

4 ActionManager 또는 ActionList 컴포넌트를 더블 클릭하여 엽니다.

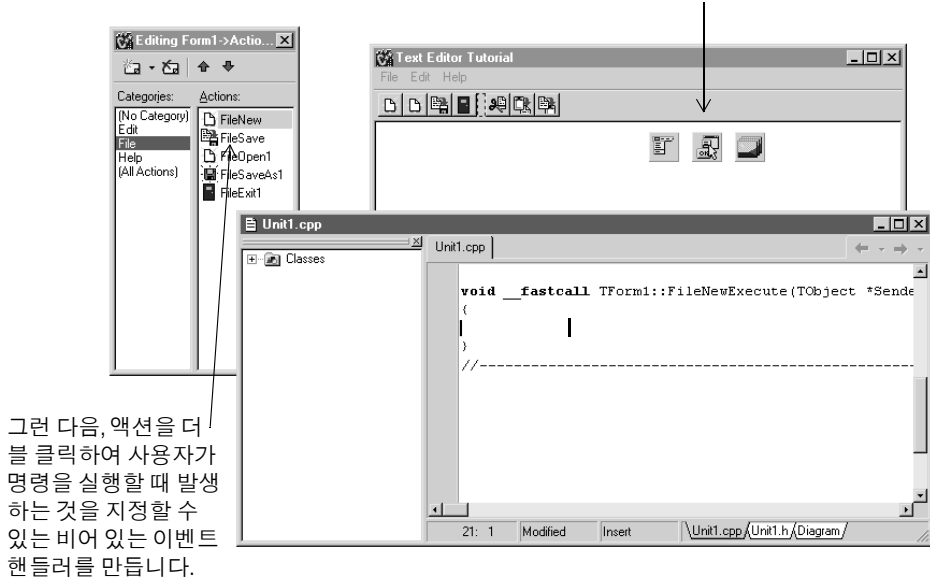
5 FileNew 액션을 더블 클릭합니다.

**팁** Object TreeView에서 FileNew 액션을 더블 클릭할 수도 있습니다.



코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

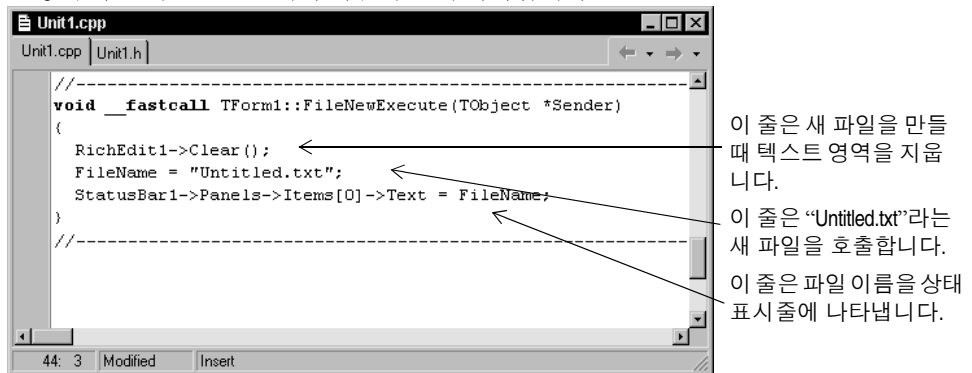
먼저 Action List 또는 Action Manager 객체를 더블 클릭하여 관련된 에디터를 표시합니다.



## 6 코드 에디터의 커서가 있는 위치({} 사이)에 다음 줄을 입력합니다.

```
RichEdit1->Clear();
FileName = "untitled.txt";
StatusBar1->Panels->Items[0]->Text = FileName;
```

그렇게 하면 이벤트 핸들러가 다음과 같이 나타납니다.



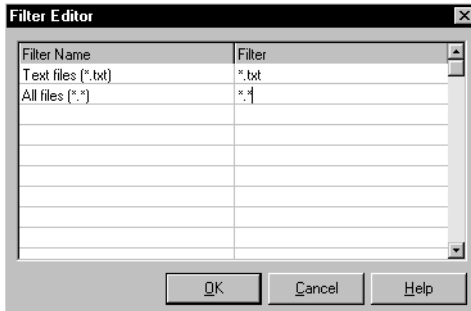
## 7 File|Save All을 선택합니다.

**참고** 윈도우의 코드 부분의 크기를 조정하면 수평 스크롤을 줄일 수 있습니다.

## Open 명령에 대한 이벤트 핸들러 만들기

텍스트 에디터에서 파일을 열기 위해 표준 Windows Open 다이얼로그 박스를 표시하려고 합니다. 이 다이얼로그 박스를 자동으로 포함하는 Action Manager 또는 Action List Editor에 표준 File|Open 명령을 이미 추가했습니다. 그러나 이 명령에 대한 이벤트 핸들러를 사용자 정의해야 합니다.

- 1 F12 키를 눌러 메인 폼을 찾고 ActionManager 또는 ActionList 컴포넌트를 더블 클릭하여 앞으로 가져옵니다.
- 2 FileOpen1 액션을 선택합니다.
- 3 Object Inspector에서 Dialog 왼쪽에 있는 + 기호를 클릭하여 속성을 확장합니다. Dialog는 Open 다이얼로그 박스를 만드는 참조된 컴포넌트입니다. C++Builder는 디폴트로 이 다이얼로그 박스의 이름을 FileOpen1->OpenDialog로 지정합니다. OpenDialog1의 Execute 메소드를 호출하면, 파일을 열기 위한 표준 다이얼로그 박스를 호출합니다.
- 4 DefaultExt 속성을 txt로 설정합니다.
- 5 Filter 옆에 있는 텍스트 영역을 더블 클릭하여 Filter Editor를 표시합니다.
  - Filter Name 열의 첫 번째 행에 Text files (\*.txt)를 입력하고 Filter 열에 \*.txt를 입력합니다.
  - Filter Name 열의 두 번째 행에 All files (\*.\*)를 입력하고 Filter 열에 \*.\*를 입력합니다.
  - OK를 클릭합니다.



Filter Editor를 사용하여  
FileOpen1.Dialog 및  
FileSaveAs1.Dialog 액션의  
필터를 정의합니다.

- 6 Title을 Open file로 설정합니다. 이 단어가 Open 다이얼로그 박스의 상단에 나타납니다.
- 7 Events 탭을 클릭합니다. OnAccept 이벤트 오른쪽의 공백을 더블 클릭하여 FileOpen1Accept를 표시합니다.

코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

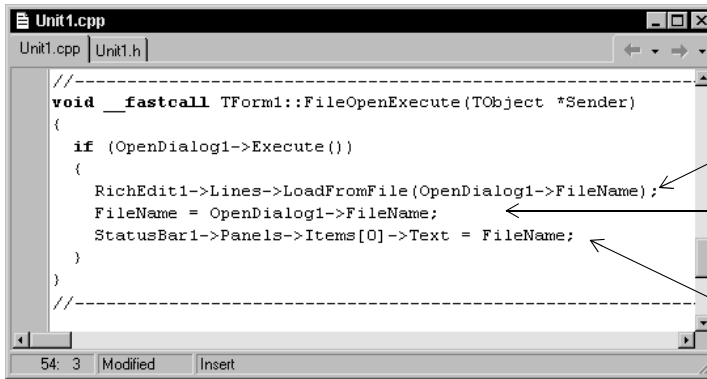
- 8 커서가 있는 위치({ } 사이)에 다음 줄을 입력합니다.

```
RichEdit1->Lines->LoadFromFile (FileOpen1->Dialog->FileName);
FileName = FileOpen1->Dialog->FileName;
StatusBar1->Panels->Items[0]->Text = FileName;
```

## 팁

2-6페이지에서 설명한대로 Code Insight 도구를 사용하여 코드를 더 빨리 작성할 수 있습니다. 예를 들어, RichEdit1 뒤에 화살표(->)를 입력하면 Code Completion 다이얼로그 박스가 나타납니다. "1"을 입력하여 Lines : TStrings;를 다이얼로그 박스 상단에 표시합니다. Enter 키를 누르거나 이 줄을 더블 클릭하여 코드에 추가합니다.

이렇게 하면 FileOpen 이벤트 핸들러가 다음과 같이 나타납니다.



이 줄은 지정된 파일로 부터 가져온 텍스트를 삽입합니다.

이 줄은 파일 이름을 Open 다이얼로그 박스에 있는 이름 중 하나로 설정합니다.

이 줄은 파일 이름을 상태 표시줄에 나타냅니다.

이제 File|Open 명령과 Open 다이얼로그 박스에 대한 내용이 모두 끝났습니다.

## Save 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 Save 명령에 대한 이벤트 핸들러를 만듭니다.

1 F12 키를 눌러 폼을 표시하고 ActionManager 또는 ActionList 컴포넌트를 더블 클릭합니다.

2 FileSave 액션을 더블 클릭합니다.

코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

## 팁

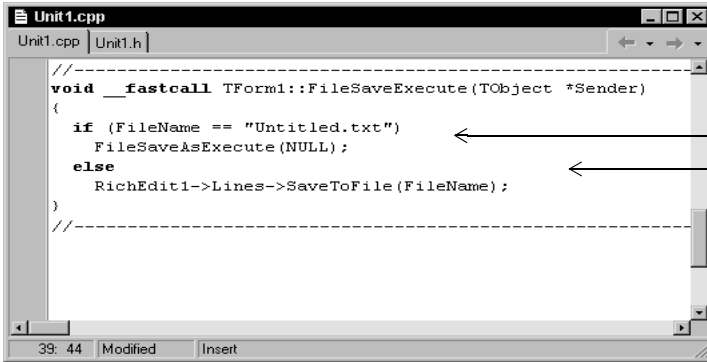
Object TreeView에서 FileSave 액션을 더블 클릭할 수도 있습니다.

3 커서가 있는 위치({ } 사이)에 다음 줄을 입력합니다.

```
if (FileName == "untitled.txt")  
    FileSaveAs1->Execute();  
else  
    RichEdit1->Lines->SaveToFile(FileName);
```

이 코드는 파일 이름이 지정되지 않았을 경우에 사용자가 이름을 지정할 수 있도록 Save As 다이얼로그 박스를 표시합니다. 이름이 지정된 경우에는 해당 이름으로 파일을 저장합니다. SaveAs 다이얼로그 박스는 Save As 명령에 대한 이벤트 핸들러에서 정의됩니다. FileSaveAs1BeforeExecute는 Save As 명령에 대해 자동으로 생성된 이름입니다.

그렇게 하면 이벤트 핸들러가 다음과 같이 나타납니다.



파일 이름이 지정되지 않은 경우, File Save As 다이얼로그 박스를 표시합니다.  
이름이 지정된 경우에는 명명된 파일로 저장합니다.

이제 File|Save 명령에 대한 내용이 모두 끝났습니다.

## Save As 명령에 대한 이벤트 핸들러 만들기

SaveDialog의 Execute 메소드가 호출되면 이 메소드는 파일 저장을 위한 표준 Windows Save As 다이얼로그 박스를 호출합니다. 다음과 같은 방법으로 Save As 명령에 대한 이벤트 핸들러를 만듭니다.

- 1 F12 키를 눌러 폼을 표시하고 ActionManager 또는 ActionList 컴포넌트를 더블 클릭합니다.
- 2 FileSaveAs1 액션을 선택합니다.
- 3 Object Inspector에서 Properties 탭을 클릭합니다. Dialog 왼쪽에 있는 + 기호를 클릭하여 속성을 확장합니다. Dialog는 Save As 다이얼로그 박스 컴포넌트를 참조하고 Save As 다이얼로그 박스의 속성을 표시합니다.
- 4 DefaultExt를 txt로 설정합니다.
- 5 Filter 옆에 있는 텍스트 영역을 더블 클릭하여 Filter Editor를 표시합니다. Filter Editor에서 파일 타입에 대한 필터를 Open 다이얼로그 박스에 있는 것처럼 지정합니다.
  - Filter Name 열의 첫 번째 행에 Text files (\*.txt)를 입력하고 Filter 옆에 \*.txt를 입력합니다.
  - Filter Name 열의 두 번째 행에 All files (\*.\*)를 입력하고 Filter 옆에 \*.\*를 입력합니다.
  - OK를 클릭합니다.
- 6 Title을 Save as로 설정합니다.
- 7 Events 탭을 클릭합니다. BeforeExecute 옆에 있는 텍스트 영역을 더블 클릭합니다. 이렇게 하면 코드 에디터가 열리면서 FileSaveAs1BeforeExecute 이벤트 핸들러 내부에 커서가 위치하게 됩니다.
- 8 코드 에디터의 커서가 있는 위치에서 다음 줄을 입력합니다.

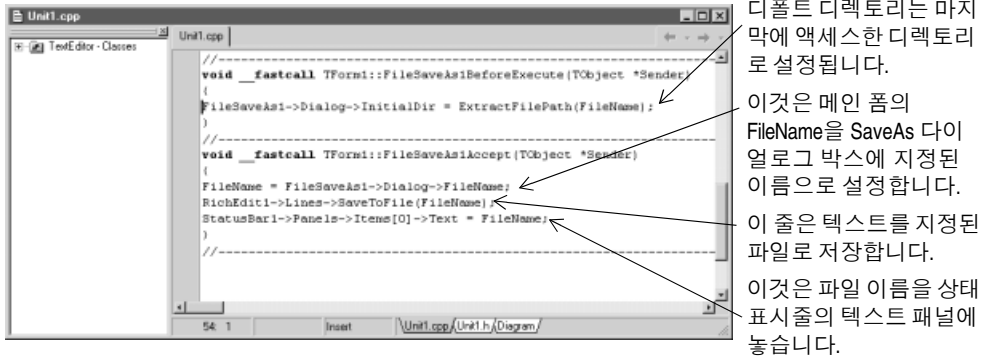
```
FileSaveAs1->Dialog->InitialDir = ExtractFilePath (Filename);
```

9 Events 탭은 닫지 말고 계속 열어두어야 합니다. *OnAccept* 이벤트 옆에 있는 텍스트 영역을 더블 클릭하여 *FileSaveAs1Accept*를 코드 에디터에 표시합니다.

10 커서가 있는 위치에 다음 줄을 입력합니다.

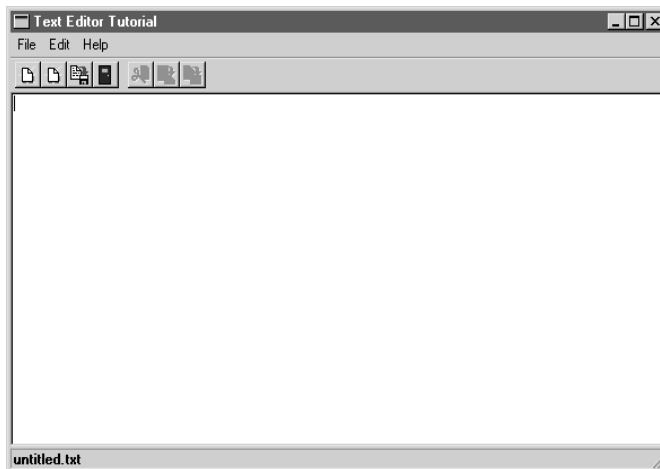
```
FileName = FileSaveAs1->Dialog->FileName;
RichEdit1->Lines->SaveToFile(FileName);
StatusBar1->Panels->Items[0]->Text = FileName;
```

이렇게 하면 *FileSaveAs* 이벤트 핸들러가 다음과 같이 나타납니다.



11 File|Save All을 선택하여 변경 사항을 저장합니다.

12 지금까지 작업한 애플리케이션이 어떻게 나타나는지 보려면 **F9** 키를 누르십시오.



실행 중인 애플리케이션은 디자인 모드의 메인 폼과 아주 유사합니다. 여기서는 논비주얼(nonvisual) 객체가 표시되지 않는다는 점에 주의하십시오.

애플리케이션은 다음 세 가지 방법으로 닫을 수 있습니다.

**X**를 클릭합니다.

File|Exit를 선택합니다.

툴바에서 Exit application 버튼을 클릭합니다.

코드 에디터 하단에 오류 메시지가 나타난 경우에는 해당 메시지를 더블 클릭하여 코드에서 오류가 발생한 위치로 이동합니다. 반드시 이 자습서에서 설명한 단계대로 따라야 합니다.

13 디자인 모드로 돌아가려면 애플리케이션의 오른쪽 위 모서리에 있는 **X**를 클릭하십시오.

## 도움말 파일 만들기

---

애플리케이션 사용법을 설명하는 도움말 파일을 만드는 것이 좋습니다. C++Builder는 C:\Project Files\Borland\CBuilder6\Help\Tools 디렉토리에서 Microsoft Help Workshop을 제공합니다. 여기에는 Windows 도움말 파일을 디자인 및 컴파일하는 것에 대한 정보가 포함되어 있습니다. 예제 텍스트 에디터 애플리케이션에서 사용자는 Help|Contents 또는 Help|Index를 선택하여 내용이나 색인이 표시되는 도움말 파일을 열 수 있습니다.

앞에서는 Action Manager 또는 Action List Editor에서 HelpContents와 HelpIndex 액션을 만들어 컴파일된 도움말 파일의 Contents 또는 Index 탭을 표시했습니다. 이제 상수 값을 Help 매개변수에 할당하고 원하는 바를 표시하는 이벤트 핸들러를 만들어야 합니다.

Help 명령을 사용하려면 Windows 도움말 파일을 작성 및 컴파일해야 합니다. 도움말 파일의 작성에 대한 내용은 이 자습서에서 다루지 않습니다. 그러나 다음과 같이 예제 rtf 파일(TextEditor.rtf), 도움말 파일(TextEditor.hlp) 및 내용 파일(TextEditor.cnt)을 다운로드할 수 있습니다.

- 1 Windows Explorer에서 C:\Program Files\Borland\CBuilder6\Help 디렉토리에 있는 B6X1.zip을 엽니다.
- 2 .hlp 및 .cnt 파일의 압축을 풀어 텍스트 에디터 디렉토리에 저장합니다. 이 디렉토리의 디폴트 위치는 C:\Program Files\Borland\CBuilder6\Projects\TextEditor입니다.

### 참고

C++Builder 도움말 파일 중 하나와 관련 .cnt 파일과 같이 프로젝트에 있는 임의의 .hlp 또는 .cnt 파일을 사용할 수도 있습니다. 이러한 파일은 프로젝트 디렉토리에 복사한 다음 애플리케이션에서 찾을 수 있도록 TextEditor.hlp 및 TextEditor.cnt로 이름을 바꿔야 합니다.

## Help Contents 명령에 대한 이벤트 핸들러 만들기

---

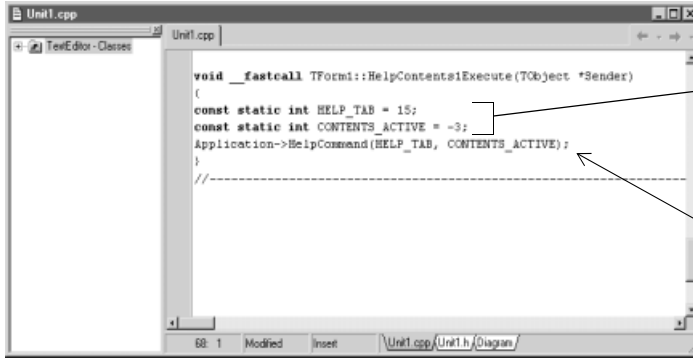
다음과 같은 방법으로 Help Contents 명령에 대한 이벤트 핸들러를 만듭니다.

- 1 *ActionManager* 또는 *ActionList* 컴포넌트를 더블 클릭합니다.
- 2 HelpContents1 액션을 더블 클릭합니다.  
코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.
- 3 커서가 있는 위치의 바로 뒤에서 다음 줄을 입력합니다.

```
const static int  HELP_TAB = 15;  
const static int  CONTENTS_ACTIVE = -3;  
Application->HelpCommand(HELP_TAB, CONTENTS_ACTIVE);
```

이 코드는 상수 값을 HelpCommand 매개변수에 할당합니다. HELP\_TAB을 15로 설정하면 Help 다이얼로그 박스가 표시되고 CONTENTS\_ACTIVE를 -3으로 설정하면 Contents 탭이 표시됩니다.

그렇게 하면 이벤트 핸들러가 다음과 같이 나타납니다.



이러한 줄은 TApplication의 HelpCommand 메소드에서 명령 및 데이터 매개변수를 정의합니다.  
이 정의는 Contents 탭이 나타나는 Help 다이얼로그 박스를 표시하도록 지시합니다.

**참고** HelpCommand 이벤트에 대한 도움말말을 얻으려면 에디터에서 HelpCommand 옆에 커서를 두고 F1 키를 누르십시오.

이제 Help|Contents 명령에 대한 내용이 모두 끝났습니다.

## Help Index 명령에 대한 이벤트 핸들러 만들기

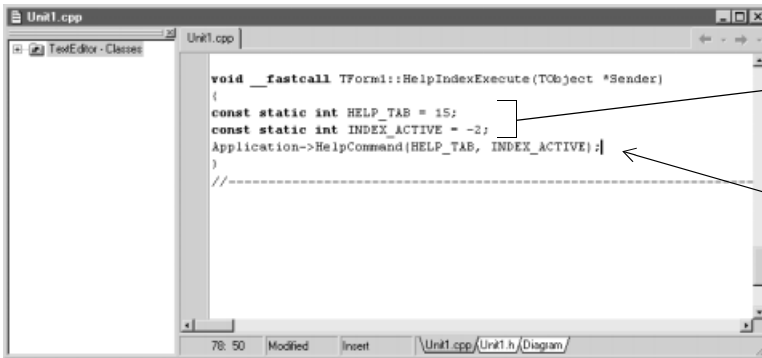
다음과 같은 방법으로 Help Index 명령에 대한 이벤트 핸들러를 만듭니다.

- 1 Action Manager 또는 Action List Editor를 닫지 말고 계속 열어두어야 합니다. 이미 닫은 경우에는 폼에서 ActionManager 또는 ActionList 컴포넌트를 더블 클릭합니다.
- 2 HelpIndex 액션을 더블 클릭합니다.  
코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.
- 3 텍스트 에디터의 커서가 있는 위치의 바로 뒤에서 다음 줄을 입력합니다.

```
const static int HELP_TAB = 15;
const static int INDEX_ACTIVE = -2;
Application->HelpCommand(HELP_TAB, INDEX_ACTIVE);
```

이 코드는 상수 값을 HelpCommand 매개변수에 할당합니다. HELP\_TAB을 다시 15로 설정하면 Help 다이얼로그 박스가 표시되고 INDEX\_ACTIVE를 -2로 설정하면 Index 탭이 표시됩니다.

그렇게 하면 이벤트 핸들러가 다음과 같이 나타납니다.



이러한 줄은 TApplication의 HelpCommand 메소드에서 명령 및 데이터 매개변수를 정의합니다.  
이 정의는 Index 탭이 나타나는 Help 다이얼로그 박스를 표시하도록 지시합니다.

이제 Help|Index 명령에 대한 내용이 모두 끝났습니다.

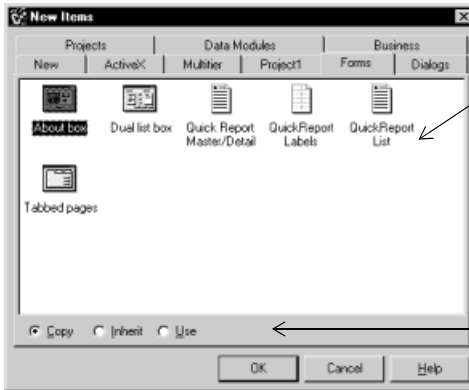
## About 박스 만들기

많은 애플리케이션에는 이름, 버전, 로고와 같은 제품에 대한 정보를 나타내는 About 박스가 있습니다. 이 박스에는 저작권 정보를 비롯한 기타 법률 정보가 포함될 수 있습니다.

Help About 명령은 이미 Action Manager 또는 Action List Editor에서 설정했습니다.

다음과 같은 방법으로 About 박스를 추가합니다.

- 1 File|New|Other를 선택하여 New Items 다이얼로그 박스를 표시합니다.
- 2 Forms 탭을 클릭하고 About Box 아이콘을 더블 클릭합니다.



New Items 다이얼로그 박스를 Object Repository라고도 부릅니다.

Object Repository에 들어 있는 한 항목을 기반으로 하여 항목을 만들 경우, 해당 항목을 복사, 상속 또는 사용할 수 있습니다.

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

About 박스의 미리 디자인된 폼이 나타납니다.



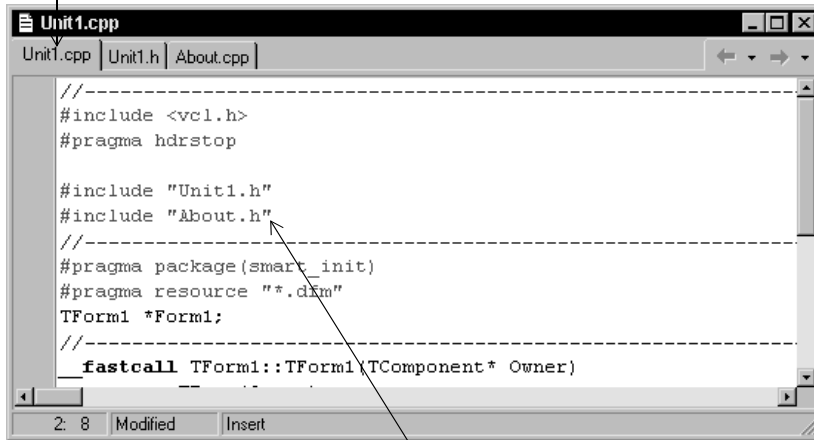
- 3 폼 자체를 선택(그리드 부분을 클릭)하고 Object Inspector에서 Properties 탭을 클릭한 다음 *Caption* 속성을 *About Text Editor*로 변경합니다.
- 4 다시 폼을 클릭합니다. 이 때 폼의 이름이 *About Text Editor*라는 점에 주의합니다. 폼에서 각 값을 변경하려면 해당 값을 클릭하여 선택한 다음 새 값을 입력하십시오.
  - Product Name을 *Text Editor*로 변경합니다.
  - Version을 *Version 1.0*으로 변경합니다.
  - Copyright를 Copyright 2002로 변경합니다.



Object Repository에는 애플리케이션을 설명하고자 하는대로 수정할 수 있는 표준 About 박스가 들어 있습니다.

- 5 File|Save As를 선택하고 About.cpp로 저장하여 About 박스 폼을 저장합니다.  
C++Builder 코드 에디터에는 여러 개의 파일, 즉 Unit1.cpp, Unit1.h, About.cpp, ActnRes (기업용 또는 전문가용 에디션이 설치되어 있거나 Action Manager Editor를 사용하는 경우) 등의 파일이 표시되어야 합니다. ActnRes 유닛이 필요하지 않더라도 그대로 둘 수 있습니다.
- 6 Unit1.cpp 탭을 클릭하고 코드 에디터의 맨 위로 스크롤합니다. About 유닛에 대한 include 문을 Unit1에 추가합니다. File|Include Unit Hdr을 선택한 다음 About을 선택하고 OK를 클릭합니다. #include About.h가 .cpp 파일 상단에 추가되었다는 것에 주의하십시오.

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



애플리케이션에 대한 새 폼을 만들 경우, 해당 폼을 메인 폼에 추가해야 합니다. File|Include Unit Hdr를 선택하고 추가할 헤더를 선택합니다.

- 7 F12 키를 눌러 디자인 모드로 돌아갑니다. *ActionManager* 또는 *ActionList* 컴포넌트를 더블 클릭하여 엽니다.
- 8 Help|About 액션을 더블 클릭하여 이벤트 핸들러를 만듭니다. 코드 에디터의 커서가 있는 위치에서 다음 줄을 입력합니다.

```
AboutBox->ShowModal();
```

이 코드는 사용자가 Help|About을 클릭할 때 About 박스를 엽니다. *ShowModal*은 모달 상태, 즉 런타임 상태로 폼을 열기 때문에, 사용자는 폼이 닫힐 때까지 아무 것도 할 수 없습니다.

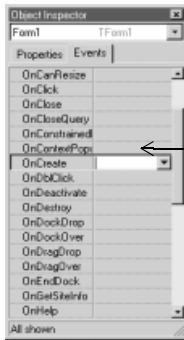
- 9 File|Save All을 선택합니다.

## 애플리케이션 완료

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

- 1 F12 키를 눌러 메인 폼을 찾습니다.
- 2 폼을 선택합니다. 포커스는 컴포넌트가 아닌 폼 자체에 있어야 합니다. 그렇지 않은 경우 Object Inspector의 상단에 있는 드롭다운 리스트 박스에서 `Form1: TForm1`을 선택합니다.

- 3 Events 탭을 클릭하고 OnCreate 옆에 있는 영역을 더블 클릭합니다.



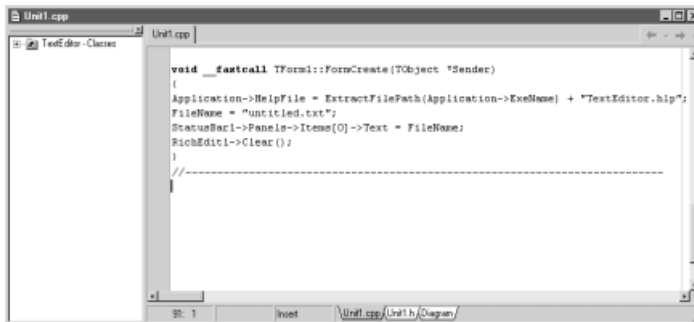
여기서 포커스가 메인 폼에 있는지 확인합니다. 그렇지 않은 경우 드롭다운 리스트에서 Form1을 선택합니다.

여기를 더블 클릭하여 폼의 OnCreate 이벤트에 대한 이벤트 핸들러를 만듭니다.

- 4 코드 에디터의 커서가 있는 위치에서 다음 줄을 입력합니다.

```
Application->HelpFile = ExtractFilePath(Application->ExeName) +  
"TextEditor.hlp";  
FileName = "untitled.txt";  
StatusBar1->Panels->Items[0]->Text = FileName;  
RichEdit1->Clear();
```

이 코드는 도움말 파일을 연결하고, *FileName* 값을 untitled.txt로 설정하고, 파일 이름을 상태 표시줄에 놓고, 텍스트 편집 영역을 지움으로써 애플리케이션을 초기화합니다.



- 5 File|SaveAll을 선택하여 변경 사항을 저장합니다.

- 6 F9키를 눌러 애플리케이션을 실행합니다.

축하합니다! 이제 모두 끝났습니다.



## CLX 데이터베이스 애플리케이션 만들기 자습서

이 자습서는 예제 직원 데이터베이스를 확인 및 업데이트할 수 있는 크로스 플랫폼 애플리케이션을 만드는 과정을 안내합니다. 크로스 플랫폼 애플리케이션은 Borland 크로스 플랫폼용 컴포넌트 라이브러리(CLX)를 사용합니다. 여러 종류의 플랫폼에서 컴파일하고 실행할 수 있도록 디자인된 CLX 애플리케이션을 사용할 경우, Windows 및 Linux 포트 간을 전환할 때 필요한 최소한의 변경 작업만 수행하면 됩니다. 크로스 플랫폼 컴파일러 지원에 대한 자세한 내용은 Borland의 최신 제품을 참조하십시오.

**참고** 이 자습서는 데이터베이스 컴포넌트를 포함하는 제품 에디션에 맞게 작성되었습니다. 이 자습서에서는 개인용 에디션에서 사용할 수 없는 기능이 필요한 데이터베이스 액세스를 설정합니다. 또한 이 자습서를 성공적으로 마치려면 InterBase를 설치해야 합니다.

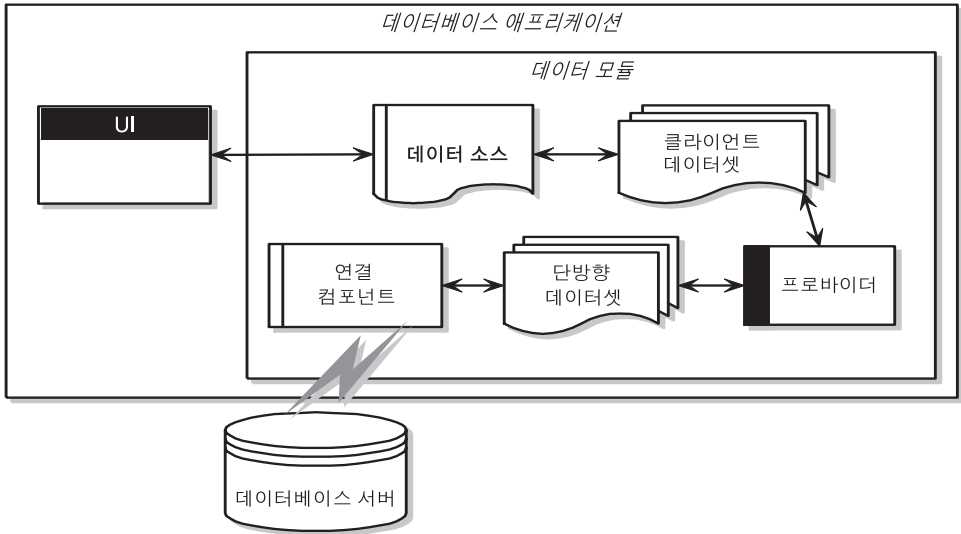
### 데이터베이스 아키텍처 개요

데이터베이스 애플리케이션의 아키텍처가 처음에는 복잡하게 보일 수 있지만, 여러 컴포넌트를 사용하면 실제로 데이터베이스 애플리케이션을 개발하고 유지 보수하는 작업이 간편해집니다.

데이터베이스 애플리케이션은 세 개의 주요 부분, 즉 사용자 인터페이스, 데이터 액세스 컴포넌트 집합 및 데이터베이스 자체를 포함합니다. 이 자습서에서는 dbExpress 데이터베이스 애플리케이션을 만듭니다. 그 밖의 데이터베이스 애플리케이션도 비슷한 아키텍처를 지닙니다.

사용자 인터페이스에는 그리드와 같은 데이터 인식 컨트롤이 포함되므로 사용자는 데이터를 편집하고 데이터베이스에 포스트할 수 있습니다. 데이터 액세스 컴포넌트는 데이터 소스, 클라이언트 데이터셋, 데이터 프로바이더, 단방향 데이터셋, 연결 컴포넌트 등을 포함합니다. 데이터 소스는 사용자 인터페이스와 클라이언트 데이터셋 사이에서 매개체로 동작합니다. 클라이언트 데이터셋은 메모리에서 버퍼링되는 기본 데이터베이스의 레코드 집합을 포함하기 때문에 애플리케이션의 핵심입니다. 프로바이더는 클라이언트 데이터셋과 데이터베이스에서 데이터를 직접 가져오는 단방향 데이터셋 간에 데이터를 전송합니다. 마지막으로 연결

컴포넌트는 데이터베이스에 대한 연결을 설정합니다. 각 타입의 단방향 데이터셋은 다른 타입의 연결 컴포넌트를 사용합니다.



데이터베이스 개발에 대한 자세한 내용은 *개발자 안내서* 또는 온라인 도움말에서 "데이터베이스 애플리케이션 디자인"을 참조하십시오.

## 새 CLX 애플리케이션 만들기

자습서를 시작하기 전에 소스 파일을 보유하는 폴더를 만듭니다. 그런 다음 새 프로젝트를 만들어 저장합니다.

- 1 이 자습서를 사용하는 도중에 만들게 될 프로젝트 파일을 보유하기 위해 **Tutorial**이라는 폴더를 만듭니다.
- 2 새 CLX 프로젝트를 시작합니다. **File | New | CLX Application**을 선택하여 새 크로스 플랫폼 프로젝트를 만듭니다.
- 3 **File | Save All**을 선택하여 파일을 디스크에 저장합니다. **Save** 다이얼로그 박스가 나타나면 **Tutorial** 폴더를 탐색하고 디폴트 이름을 사용하여 각 파일을 저장합니다.

나중에 **File | Save All**을 선택하여 작업을 언제든지 저장할 수 있습니다. 자습서를 한 번에 끝낼 계획이 아니라면, **File | Reopen**을 선택하고 리스트에서 이 자습서를 선택하여 저장된 버전을 열 수 있습니다.

## 데이터 액세스 컴포넌트 설정

데이터 액세스 컴포넌트는 데이터(데이터셋)와 데이터셋을 애플리케이션의 다른 부분에 연결하는 컴포넌트를 나타냅니다. 이러한 데이터 액세스 컴포넌트는 각각 바로 다음의 하위 컴포넌트를 가리킵니다. 예를 들어, 데이터 소스는 클라이언트 데이터셋을 가리키고 클라이언트 데이터셋은 프로바이더를 가리킵니다. 따라서 데이터 액세스 컴포넌트를 설정할 때는 컴포넌트를 올바른 순서로 추가합니다.

다음 단원에서는 데이터베이스 컴포넌트를 추가하여 데이터베이스 연결, 단방향 데이터셋, 프로바이더, 클라이언트 데이터셋 및 데이터 소스를 만듭니다. 그런 다음 애플리케이션의 사용자 인터페이스를 만듭니다. 이러한 컴포넌트는 컴포넌트 팔레트의 **dbExpress**, **Data Access** 및 **Data Controls** 페이지에 위치합니다.

**팁** 고유한 폼에서 사용자 인터페이스를 분리하고 데이터 액세스 컴포넌트를 데이터 모듈에 두는 것이 좋습니다. 그러나 이 자습서에서는 작업을 쉽게 하기 위해 사용자 인터페이스와 모든 컴포넌트를 동일한 폼에 둡니다.

### 데이터베이스 연결 설정

dbExpress 페이지에는 SQL 데이터베이스 서버에 대한 빠른 액세스를 제공하는 컴포넌트 집합이 포함되어 있습니다.

데이터베이스에 연결할 수 있도록 연결 컴포넌트를 추가해야 합니다. 사용되는 연결 컴포넌트의 타입은 사용되는 데이터셋 컴포넌트의 타입에 따라 달라집니다. 이 자습서에서는 *TSQLConnection* 및 *TSQLDataSet* 컴포넌트를 사용합니다.

다음과 같은 방법으로 dbExpress 연결 컴포넌트를 추가합니다.



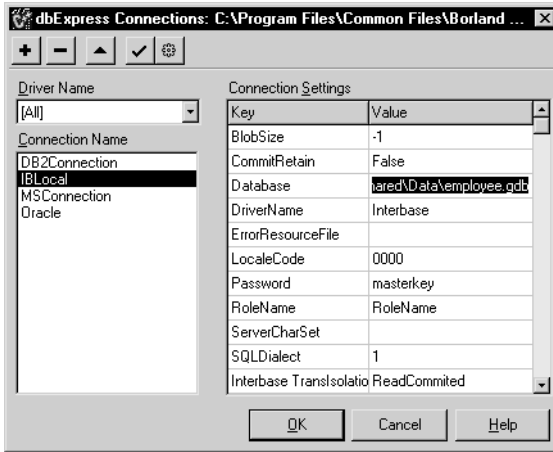
- 1 컴포넌트 팔레트에서 dbExpress 페이지를 클릭하고 *TSQLConnection* 컴포넌트를 더블 클릭하여 폼 위에 둡니다. *TSQLConnection* 컴포넌트를 찾으려면 팔레트에서 아이콘을 잠시 가리키십시오. 그러면 도움말 힌트가 컴포넌트의 이름을 나타냅니다. 디폴트로 이 컴포넌트를 *SQLConnection1*이라고 합니다.

*TSQLConnection*은 논비주얼(nonvisual) 컴포넌트이므로 아무 곳에도 둘 수 있습니다. 그러나 이 자습서에서는 모든 논비주얼(nonvisual) 컴포넌트를 폼 상단에 일렬로 맞춥니다.

**팁** 폼에 놓은 컴포넌트의 캡션을 표시하려면 Tools | Environment Options를 선택하고 Show component captions를 클릭하십시오.

- 2 Object Inspector에서 *ConnectionName* 속성을 IBLocal(드롭다운 리스트에 있음)로 설정합니다.
- 3 *LoginPrompt* 속성을 *false*로 설정합니다. 이 속성을 *false*로 설정하면 데이터베이스를 액세스할 때마다 로그인하라는 메시지가 나타나지 않습니다.

- 4 *TSQLConnection* 컴포넌트를 더블 클릭하여 Connection Editor를 표시합니다.



Connection Editor를 사용하여 *TSQLConnection* 컴포넌트에 대한 연결 구성을 선택하거나 dbxconnections.ini 파일에 저장된 연결을 편집합니다. 다이얼로그 박스에서의 모든 변경 사항은 OK를 클릭할 때 이 파일에 기록됩니다. 또한 OK를 클릭하면 선택한 연결이 *SQL Connection* 컴포넌트의 *ConnectionName* 속성에 대한 값으로 할당됩니다.

- 5 Connection Editor에서 시스템에 있는 employee.gdb라는 데이터베이스 파일의 경로 이름을 지정합니다. 이 자습서에서는 C++Builder와 함께 제공되는 예제 InterBase 데이터베이스, 즉 employee.gdb에 연결합니다. 디폴트로, InterBase는 설치될 때 employee.gdb를 C:\Program Files\Borland Shared\Data에 둡니다.
- 6 User\_Name 및 Password 필드에 허용되는 값이 있는지 확인합니다. 기본값을 변경하지 않았다면 이러한 필드를 변경할 필요가 없습니다. 데이터베이스 액세스를 다른 사람이 관리할 경우, 데이터베이스에 액세스하려면 사용자 이름과 암호를 얻어야 할 수 있습니다.
- 7 필드 확인과 편집이 끝나면 OK를 클릭하여 Connection Editor를 닫고 변경 사항을 저장합니다.

이러한 변경 사항은 dbxconnections.ini 파일에 기록되고 선택한 연결은 *SQL Connection* 컴포넌트의 *ConnectionName* 속성에 대한 값으로 할당됩니다.

**팁** Connection Editor를 사용하면서 추가 도움이 필요하면 Help 버튼을 클릭합니다.

- 8 File|Save All을 선택하여 프로젝트를 저장합니다.



## 단방향 데이터셋 설정

기본 데이터베이스 애플리케이션은 데이터셋을 사용하여 데이터베이스의 정보를 액세스합니다. dbExpress 애플리케이션에서는 단방향 데이터셋을 사용합니다. 단방향 데이터셋은 데이터베이스에서 데이터를 읽지만 데이터를 업데이트하지는 않습니다.

다음과 같은 방법으로 단방향 데이터셋을 추가합니다.



- 1 dbExpress 탭에서 *TSQLDataSet*을 폼 위에 가져다 놓습니다.
- 2 Object Inspector에서 *SQLConnection* 속성을 *SQLConnection1*(이전에 만든 데이터베이스 연결)로 설정합니다.
- 3 *CommandText* 속성을 "select \* from SALES"로 설정하여 데이터셋이 실행하는 명령을 지정합니다. Object Inspector에서 Select 문을 입력하거나 *CommandText* 오른쪽에 있는 생략 부호를 클릭하여 고유한 쿼리 문을 작성할 수 있는 *CommandText Editor*를 표시할 수 있습니다.

**팁** *CommandText Editor*를 사용하면서 추가 도움이 필요하면 *Help* 버튼을 클릭합니다.

- 4 *Active*를 *true*로 설정하여 데이터셋을 엽니다.
- 5 File|Save All을 선택하여 프로젝트를 저장합니다.

## 프로바이더, 클라이언트 데이터셋 및 데이터 소스 설정

Data Access 페이지에는 dbExpress 뿐만 아니라 모든 데이터 액세스 메커니즘에 사용할 수 있는 컴포넌트가 포함되어 있습니다.

프로바이더 컴포넌트는 클라이언트 데이터셋이 자신의 데이터를 다른 데이터셋에서 가져오는 방법입니다. 프로바이더는 클라이언트 데이터셋에서 데이터 요청을 받고 데이터를 가져와서 패키지한 다음 클라이언트 데이터셋에 반환합니다. dbExpress를 사용하는 중이면 프로바이더는 클라이언트 데이터셋에서 업데이트를 받아 데이터베이스 서버에 적용합니다.

다음과 같은 방법으로 프로바이더를 추가합니다.



- 1 Data Access 페이지에서 *TDataSetProvider* 컴포넌트를 폼 위에 가져다 놓습니다.
- 2 Object Inspector에서 프로바이더의 *DataSet* 속성을 *SQLDataSet1*로 설정합니다.

클라이언트 데이터셋은 자신의 데이터를 메모리에서 버퍼링합니다. 또한 클라이언트 데이터셋은 데이터베이스에 보낼 업데이트를 캐싱합니다. 클라이언트 데이터셋을 사용하면 데이터 소스 컴포넌트를 통해 사용자 인터페이스에 있는 데이터 인식 컨트롤에 데이터를 제공할 수 있습니다.

다음과 같은 방법으로 클라이언트 데이터셋을 추가합니다.



- 1 Data Access 페이지에서 *TClientDataSet* 컴포넌트를 *TDataSetProvider* 컴포넌트 오른쪽으로 가져다 놓습니다.
- 2 *ProviderName* 속성을 *DataSetProvider1*로 설정합니다.
- 3 *Active* 속성을 *true*로 설정하여 데이터를 애플리케이션에 전달할 수 있게 만듭니다.

데이터 소스는 클라이언트 데이터셋을 데이터 인식 컨트롤에 연결합니다. 각 데이터 인식 컨트롤은 데이터를 표시 및 처리하기 위하여 데이터 소스 컴포넌트에 연결되어야 합니다. 마찬가지로 모든 데이터셋은 데이터 소스 컴포넌트와 연결되어야만 폼의 데이터 인식 컨트롤에서 자체의 데이터를 표시 및 처리할 수 있습니다.

다음과 같은 방법으로 데이터 소스를 추가합니다.



- 1 Data Access 페이지에서 *TDataSource* 컴포넌트를 *TClientDataSet* 컴포넌트 오른쪽으로 가져다 놓습니다.
- 2 데이터 소스의 *DataSet* 속성을 *ClientDataSet1*로 설정합니다.
- 3 File|Save All을 선택하여 프로젝트를 저장합니다.

지금까지는 논비주얼(nonvisual) 데이터베이스 인프라를 애플리케이션에 추가했습니다. 다음에 할 일은 사용자 인터페이스를 디자인하는 것입니다.

## 사용자 인터페이스 디자인

이제 비주얼 컨트롤을 애플리케이션에 추가하여 사용자가 데이터를 확인, 편집 및 저장할 수 있게 해야 합니다. Data Controls 페이지는 데이터베이스의 데이터를 사용하고 사용자 인터페이스를 생성하는 데이터 인식 컨트롤 집합을 제공합니다. 여기서는 데이터베이스를 그리드에 표시하고 몇 가지 명령과 탐색 모음을 추가합니다.

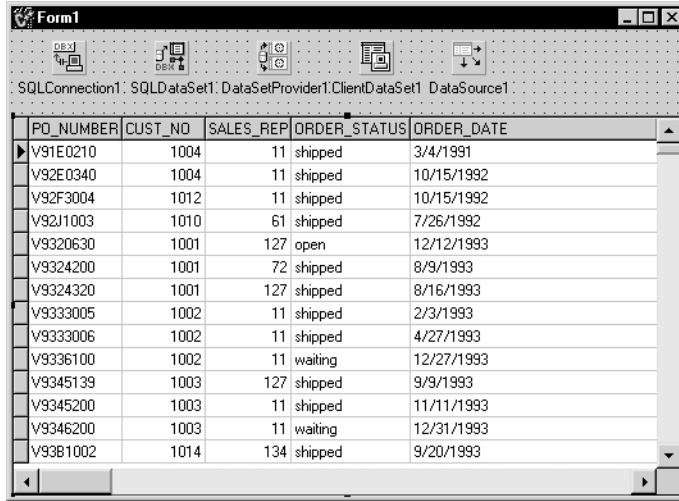
### 그리드 및 탐색 모음 만들기

다음과 같은 방법으로 애플리케이션의 인터페이스를 만듭니다.



- 1 우선 그리드를 폼에 추가할 수 있습니다. Data Controls 페이지에서 *TDBGrid* 컴포넌트를 폼 위에 가져다 놓습니다.
- 2 *DBGrid*의 속성을 설정하여 그리드를 연결합니다. Object Inspector에서 *Anchors* 옆에 있는 + 기호를 클릭하여 *akLeft*, *akTop*, *akRight* 및 *akBottom*을 표시하고 모두 *true*로 설정합니다. 가장 쉬운 방법은 Object Inspector에서 각 속성 옆에 있는 *false*를 더블 클릭하는 것입니다.
- 3 *Align* 속성을 *alBottom*으로 설정하여 폼 하단에 그리드를 정렬합니다. 그리드를 끌거나 *Height* 속성을 400으로 설정하여 그리드 크기를 늘릴 수도 있습니다.
- 4 그리드의 *DataSource* 속성을 *DataSource1*로 설정합니다. 이렇게 하면 그리드는 직원 데이터베이스의 데이터로 채워집니다. 그리드가 데이터를 표시하지 않으면 이전 지침에서 설명한대로 폼에서 모든 객체의 속성을 제대로 설정했는지 확인합니다.

지금까지 작업한 애플리케이션은 다음과 같이 나타나야 합니다.



PO_NUMBER	CUST_NO	SALES_REP	ORDER_STATUS	ORDER_DATE
V91E0210	1004	11	shipped	3/4/1991
V92E0340	1004	11	shipped	10/15/1992
V92F3004	1012	11	shipped	10/15/1992
V92J1003	1010	61	shipped	7/26/1992
V9320630	1001	127	open	12/12/1993
V9324200	1001	72	shipped	8/9/1993
V9324320	1001	127	shipped	8/16/1993
V9333005	1002	11	shipped	2/3/1993
V9333006	1002	11	shipped	4/27/1993
V9336100	1002	11	waiting	12/27/1993
V9345139	1003	127	shipped	9/9/1993
V9345200	1003	11	shipped	11/11/1993
V9346200	1003	11	waiting	12/31/1993
V93B1002	1014	134	shipped	9/20/1993

*DBGrid* 컨트롤은 IDE에서 작업하는 동안 디자인 타임에 데이터를 표시합니다. 따라서 데이터베이스에 제대로 연결되었는지 확인할 수 있습니다. 그러나 디자인 타임에는 데이터를 편집할 수 없습니다. 테이블에서 데이터를 편집하려면 애플리케이션을 실행해야 합니다.



- 5 **Data Controls** 페이지에서 *TDBNavigator* 컨트롤을 폼 위에 가져다 놓습니다. 데이터베이스 탐색기는 다음 및 이전 화살표 등을 사용하여 데이터셋의 데이터 사이를 이동하고 데이터에 대한 작업을 수행하기 위한 도구입니다.
- 6 탐색기 표시줄의 *DataSource* 속성을 *DataSource1*로 설정하여 탐색기가 클라이언트 데이터셋에서 데이터를 찾게 합니다.
- 7 탐색기 표시줄의 *ShowHint* 속성을 *true*로 설정합니다. *ShowHint*를 *true*로 설정하면 런타임시 탐색기 표시줄에 있는 각 항목 위로 커서를 이동했을 때 도움말 힌트가 나타납니다.
- 8 **File|Save All**을 선택하여 프로젝트를 저장합니다.



- 9 F9 키를 눌러 프로젝트를 컴파일하고 실행합니다. Debug 툴바에서 Run 버튼을 클릭하거나 Run 메뉴에서 Run을 선택하여 프로젝트를 실행할 수도 있습니다.

PO_NUMBER	CUST_NO	SALES_REP	ORDER_STATUS	ORDER_DATE
V91E0210	1004	11	shipped	3/4/1991
V92E0340	1004	11	shipped	10/15/1992
V92F3004	1012	11	shipped	10/15/1992
V92I1003	1010	61	shipped	7/26/1992
V9320630	1001	127	open	12/12/1993
V9324200	1001	72	shipped	8/9/1993
V9324320	1001	127	shipped	8/16/1993
V9333005	1002	11	shipped	2/3/1993
V9333006	1002	11	shipped	4/27/1993
V9336100	1002	11	waiting	12/27/1993
V9345139	1003	127	shipped	9/9/1993
V9345200	1003	11	shipped	11/11/1993
V9346200	1003	11	waiting	12/31/1993
V9381002	1014	134	shipped	9/20/1993

프로젝트를 실행할 때, 프로그램은 폼에서 디자인한 것과 비슷한 윈도우에서 열립니다. 탐색 모음을 직원 데이터베이스를 통해 테스트할 수 있습니다. 예를 들어, 화살표 명령을 사용하여 레코드 사이를 이동하고, + 명령을 사용하여 레코드를 추가하고, - 명령을 사용하여 레코드를 삭제할 수 있습니다.

- 팁** 애플리케이션의 초기 버전을 테스트하는 동안 오류가 발생하면 Run | Program Reset을 선택하여 디자인 타임 뷰로 돌아갑니다.

## 메뉴에 대한 지원 추가

프로그램에 많은 기능이 포함되었지만, GUI 애플리케이션에서 일반적으로 제공하는 여러 기능이 여전히 부족합니다. 예를 들어, 대부분의 애플리케이션은 사용을 쉽게 하기 위한 메뉴와 버튼을 구현합니다.

이 단원에서는 *액션 리스트*를 추가합니다. 액션 리스트를 사용하지 않고 메뉴, 툴바 및 버튼을 만들 수 있지만, 액션 리스트는 사용자 명령에 대한 응답을 한 곳에서 처리하여 개발 및 유지 보수를 쉽게 만듭니다. Windows 전용 개발의 경우, 액션 밴드를 통해서도 툴바와 메뉴의 개발을 쉽게 만들 수 있습니다.

- 1 애플리케이션이 계속 실행 중이면 오른쪽 위 모서리에 있는 **X**를 클릭하여 애플리케이션을 닫고 폼의 디자인 타임 뷰로 돌아갑니다.

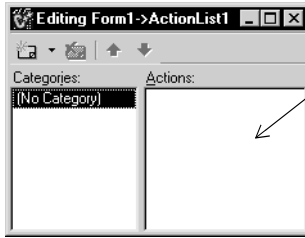


- 2 컴포넌트 팔레트의 Common Controls 페이지에서 *ImageList* 컴포넌트를 폼 위에 가져다 놓은 다음 다른 논비주얼(nonvisual) 컴포넌트 옆에 일렬로 맞춥니다. *ImageList*에는 잘라내기 및 붙여넣기와 같은 표준 액션을 나타내는 아이콘이 포함됩니다.

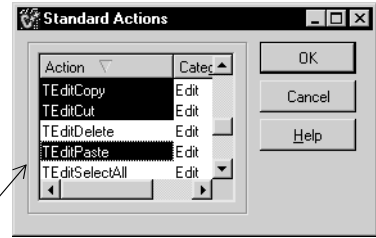


- 3 컴포넌트 팔레트의 Standard 페이지에서 *ActionList* 컴포넌트를 폼 위에 가져다 놓습니다. 액션 리스트의 *Images* 속성을 *ImageList1*로 설정합니다.

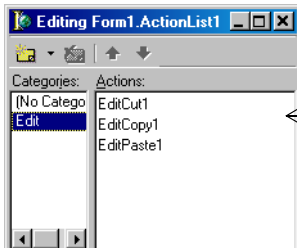
- 4 액션 리스트를 더블 클릭하여 Action List Editor를 표시합니다.



이 에디터에서 마우스 오른쪽 버튼을 클릭하고 New Standard Action을 선택하여 Standard Actions 리스트 박스를 표시합니다. 원하는 액션을 선택하고 OK를 클릭합니다. *Ctrl* 키를 사용하면 여러 액션을 선택할 수 있습니다.



- 5 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 선택합니다. Standard Actions 리스트 박스가 나타납니다.
- 6 표준 액션, 즉 *TEditCopy*, *TEditCut* 및 *TEditPaste*를 선택합니다. *Ctrl* 키를 사용하면 여러 항목을 선택할 수 있습니다. 그런 다음 OK를 클릭합니다.



제품과 함께 제공되는 표준 액션을 추가했습니다. 이러한 액션을 메뉴에서 사용합니다.

- 7 Action List Editor를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택하여 디폴트로 제공되지 않는 다른 액션을 추가합니다. Action1은 디폴트로 추가됩니다. Object Inspector에서 *Caption* 속성을 Update Now!로 설정합니다.
- 이 액션은 메뉴와 버튼에서 사용됩니다. 나중에 이 액션이 데이터베이스를 업데이트하도록 이벤트 핸들러를 추가합니다.
- 8 (No Category)를 클릭하고 마우스 오른쪽 버튼을 클릭한 다음 New Action을 선택하여 다른 액션을 추가합니다. Action2가 추가되면 *Caption* 속성을 *E&xit*로 설정합니다.
- 9 오른쪽 위 모서리에 있는 **X**를 클릭하여 Action List Editor를 닫습니다.
- 이제 세 개의 표준 액션과 나중에 이벤트 핸들러에 연결할 다른 두 개의 액션을 추가했습니다.
- 10 File|Save All을 선택하여 프로젝트를 저장합니다.

## 메뉴 추가

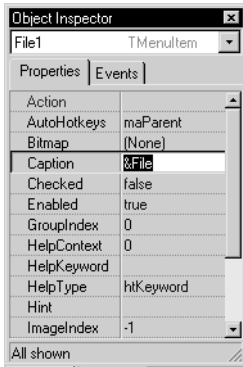
이 단원에서는 두 가지 드롭다운 메뉴(File과 Edit)가 있는 메인 메뉴 표시줄을 추가하고 액션 리스트의 액션을 사용하여 메뉴 항목을 각 메뉴에 추가합니다.



- 1 컴포넌트 팔레트의 **Standard** 페이지에서 *TMainMenu* 컴포넌트를 폼 위에 가져다 놓은 다음 다른 난비주얼(nonvisual) 컴포넌트 옆으로 끌어 놓습니다.
- 2 메인 메뉴의 *Images* 속성을 *ImageList1*로 설정하여 이미지 리스트를 메뉴 항목에 연결합니다.
- 3 메뉴 컴포넌트를 더블 클릭하여 메뉴 디자이너를 표시합니다.



- 4 **&File**을 입력하여 첫 번째 상위 레벨 메뉴 항목의 *Caption* 속성을 설정하고 **Enter** 키를 누릅니다.



**&File**을 입력하고 **Enter**키를 누르면 상위 레벨 **File** 명령이 나타나 첫 번째 메뉴 항목을 추가할 수 있습니다. 문자 앞의 앰퍼샌드는 가속키를 활성화합니다.



- 5 **File** 메뉴 아래에서 비어 있는 메뉴 항목을 선택합니다. 비어 있는 메뉴 항목의 *Action* 속성을 *Action2*로 설정합니다. **Exit** 메뉴가 **File** 메뉴 아래 나타납니다.
- 6 **File** 오른쪽에 있는 두 번째 상위 레벨 메뉴 항목을 클릭합니다. *Caption* 속성을 **&Edit**로 설정하고 **Enter** 키를 누릅니다. **Edit** 메뉴 아래에서 비어 있는 메뉴 항목을 선택합니다.
- 7 **Object Inspector**에서 *Action* 속성을 *EditCut1*로 설정하고 **Enter** 키를 누릅니다. 항목의 캡션이 **Cut**으로 자동 설정되고 디폴트 잘라내기 비트맵이 메뉴에 나타납니다.
- 8 **Cut** 아래에서 비어 있는 다음 메뉴 항목을 선택하고 *Action* 속성을 *EditCopy1*로 설정합니다. 그러면 디폴트 복사 비트맵이 메뉴에 나타납니다.
- 9 **Copy** 아래에서 비어 있는 다음 메뉴 항목을 선택하고 *Action* 속성을 *EditPaste1*로 설정합니다. 그러면 디폴트 붙여넣기 비트맵이 메뉴에 나타납니다.
- 10 **Paste** 아래에서 비어 있는 다음 메뉴 항목을 선택하고 *Caption* 속성을 하이픈(-)으로 설정하여 메뉴에 분할선을 만듭니다. **Enter** 키를 누릅니다.
- 11 분할선 아래에서 비어 있는 다음 메뉴 항목을 선택하고 *Action* 속성을 *Action1*로 설정합니다. 그러면 이 메뉴 항목에 **Update Now!**가 표시됩니다.

12 X를 클릭하여 메뉴 디자이너를 닫습니다.

13 File|Save All을 선택하여 프로젝트를 저장합니다.



14 F9 키를 누르거나 톨바에서 Run을 클릭하여 프로그램을 실행하고 어떻게 나타나는지 확인합니다.

PO_NUMBER	CUST_NO	SALES_REF	ORDER_STATUS	ORDER_DATE
V91E0210	1004	11	shipped	3/4/1991
V92E0340	1004	11	shipped	10/15/1992
V92F3004	1012	11	shipped	10/15/1992
V92J1003	1010	61	shipped	7/26/1992
V9320630	1001	127	open	12/12/1993
V9324200	1001	72	shipped	8/9/1993
V9324320	1001	127	shipped	8/16/1993
V9333005	1002	11	shipped	2/3/1993
V9333006	1002	11	shipped	4/27/1993
V9336100	1002	11	waiting	12/27/1993
V9345139	1003	127	shipped	9/9/1993
V9345200	1003	11	shipped	11/11/1993
V9346200	1003	11	waiting	12/31/1993
V93B1002	1014	134	shipped	9/20/1992

Edit 메뉴와 탐색 모음에 여러 명령이 작동하는 것을 볼 수 있습니다. Copy와 Cut은 데이터베이스에서 텍스트를 선택할 때까지 Edit 메뉴에서 비활성 상태입니다. 탐색 모음을 사용하면 데이터베이스의 레코드 사이를 이동하거나, 레코드를 삽입하거나, 레코드를 삭제할 수 있습니다. Update 명령은 아직 작동하지 않습니다.

다음 작업을 계속할 준비가 되면 애플리케이션을 닫습니다.

## 버튼 추가

이 단원에서는 Update Now 버튼을 애플리케이션에 추가하는 방법에 대해 설명합니다. 이 버튼은 레코드 편집, 새 레코드 추가, 레코드 삭제 등과 같이 사용자가 편집한 사항을 데이터베이스에 적용하는 데 사용됩니다.

다음과 같은 방법으로 버튼을 추가합니다.



1 컴포넌트 팔레트의 Standard 페이지에서 TButton을 폼 위에 가져다 놓습니다. 이렇게 하면 컴포넌트를 선택한 다음 탐색 모음 옆에 있는 폼을 클릭합니다.

2 버튼의 Action 속성을 Action1로 설정합니다.

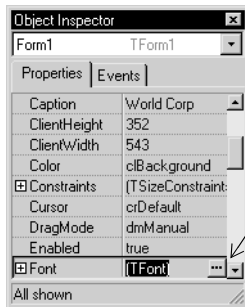
버튼의 캡션이 Update Now!로 변경됩니다. 애플리케이션을 실행하면 이 버튼은 작동을 위한 이벤트 핸들러를 추가할 때까지 비활성 상태로 남습니다.

## 이름과 이미지 표시

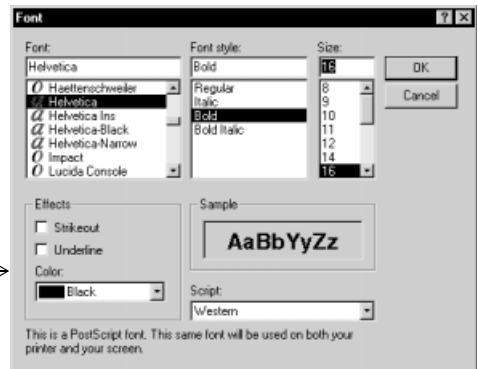
다음과 같이 회사 이름과 이미지를 추가하여 애플리케이션을 더 전문적으로 보이게 할 수 있습니다.



- 1 컴포넌트 팔레트의 **Standard** 페이지에서 *TLabel* 컴포넌트를 폼 위에 가져다 놓습니다. 이 컴포넌트의 이름은 디폴트로 *Label1*로 지정됩니다.
- 2 **Object Inspector**에서 레이블의 *Caption* 속성을 *World Corp*나 다른 회사 이름으로 변경합니다.
- 3 *Font* 속성을 클릭하여 회사 이름의 글꼴을 변경합니다. *Font* 다이얼로그 박스의 오른쪽에 나타나는 생략 부호를 클릭하고 글꼴을 *Helvetica Bold*, 16포인트 타입으로 변경한 다음 **OK**를 클릭합니다.



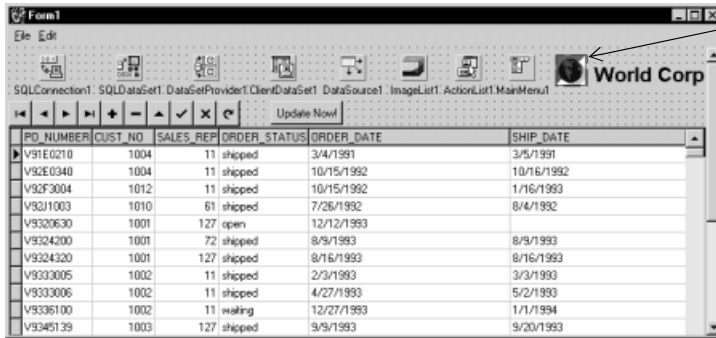
Object Inspector에서 *Font* 속성을 사용하여 레이블의 글꼴을 변경할 수 있습니다. 생략 부호를 클릭하여 표준 글꼴 다이얼로그 박스를 표시합니다.



- 4 레이블을 오른쪽 위 모서리에 놓습니다.
- 5 **Additional** 컴포넌트 팔레트 페이지에서 *TImage* 컴포넌트를 레이블 옆에 가져다 놓습니다. 이 컴포넌트의 이름은 디폴트로 *Image1*로 지정됩니다.
- 6 이미지를 *Image1* 컴포넌트에 추가하려면 *Picture* 속성을 클릭하십시오. 생략 부호를 클릭하여 **Picture Editor**를 표시합니다.
- 7 **Picture Editor**에서 **Load**를 선택하고 *earth.ico*를 찾습니다. C++Builder에서 이 파일의 경로는 *Program Files\Common Files\Borland Shared\images\icons\earth.ico*입니다.
- 8 *Earth.ico*를 더블 클릭하고 **OK**를 클릭하여 그림을 로드하고 **Picture Editor**를 닫습니다.
- 9 디폴트 이미지 영역의 크기를 그림 크기로 조정하고, 이미지를 레이블 옆에 둡니다.







가장자리를 끌어  
Image의 너비를 설정  
하거나 Object Inspector  
에서 Width 속성을  
설정할 수 있습니다.

- 10 텍스트와 이미지를 정렬하려면, 폼에서 두 객체 모두를 선택하고 마우스 오른쪽 버튼을 클릭한 다음 **Align**을 선택하십시오. **Alignment** 다이얼로그 박스의 **Vertical** 아래에서 **Bottoms**를 클릭합니다.
  - 11 **File|Save All**을 선택하여 프로젝트를 저장합니다.
  - 12 **F9** 키를 눌러 애플리케이션을 컴파일하고 실행합니다.
- 다음 작업을 계속할 준비가 되면 애플리케이션을 닫습니다.

## 이벤트 핸들러 작성

컴포넌트 팔레트에 있는 대부분의 컴포넌트에는 이벤트가 있으며, 대부분의 컴포넌트는 디폴트 이벤트를 가집니다. *TButton*과 같은 컴포넌트를 클릭할 때마다 호출되는 *OnClick*이 일반적인 디폴트 이벤트 중 하나입니다. 폼에서 컴포넌트를 선택하고 **Object Inspector**의 **Events** 탭을 클릭하면 컴포넌트의 이벤트 리스트가 나타납니다.

이벤트와 이벤트 핸들러에 대한 자세한 내용은 *개발자 안내서* 또는 온라인 도움말에서 "애플리케이션 사용자 인터페이스 개발"을 참조하십시오.

### Update Now! 명령 이벤트 핸들러 작성

우선 다음과 같이 **Update Now!** 명령 및 버튼에 대한 이벤트 핸들러를 작성합니다.

- 1 *ActionList* 컴포넌트를 더블 클릭하여 **Action List Editor**를 표시합니다.
- 2 (No Category)를 선택하여 **Action1**과 **Action2**를 표시합니다.
- 3 **Action1**을 더블 클릭합니다. 코드 에디터에 다음 스켈레톤 이벤트 핸들러가 나타납니다.

```
void __fastcall TForm1::Action1Execute(TObject *Sender)
{
}

```

커서가 있는 위치(중괄호 사이)에서 다음을 입력합니다.

```
if(ClientDataSet1->State == dsEdit || ClientDataSet1->State ==
dsInsert)
    ClientDataSet1->Post();

ClientDataSet1->ApplyUpdates(-1);
```

이 이벤트 핸들러는 먼저 데이터베이스 상태를 확인합니다. 변경된 레코드에서 빠져 나오면 해당 레코드는 자동으로 포스트됩니다. 그러나 변경된 레코드에서 빠져 나오지 않으면 데이터베이스는 편집 또는 삽입 모드로 남아 있습니다. **if** 문은 변경되었지만 클라이언트 데이터셋에 전달되지 않은 모든 데이터를 포스트합니다. 그 다음에 나오는 문은 클라이언트 데이터셋에 있는 업데이트를 데이터베이스에 적용합니다.

**참고** dbExpress를 사용하는 중이면 변경된 데이터가 자동으로 데이터베이스에 포스트되지 않습니다. 업데이트, 삽입 및 삭제된 모든 레코드를 클라이언트 데이터셋에서 데이터베이스에 기록하려면 *ApplyUpdates* 메소드를 호출해야 합니다.

## Exit 명령 이벤트 핸들러 작성

---

이제 다음과 같이 Exit 명령에 대한 이벤트 핸들러를 작성합니다.

- 1 Action List Editor를 아직 표시하지 않았다면 *ActionList* 컴포넌트를 더블 클릭하여 표시합니다.
- 2 (No Category)를 클릭하여 Action2를 표시합니다.
- 3 Action2를 더블 클릭합니다. 코드 에디터에 다음 스켈레톤 이벤트 핸들러가 표시됩니다.

```
void __fastcall TForm1::Action2Execute(TObject *Sender)
{

}
```

커서가 있는 위치(중괄호 사이)에서 다음을 입력합니다.

```
Close();
```

메뉴의 File|Exit 명령이 사용되면 이 이벤트 핸들러는 애플리케이션을 닫습니다.

- 4 Action List Editor를 닫습니다.
- 5 File|Save All을 선택하여 프로젝트를 저장합니다.

## FormClose 이벤트 핸들러 작성

---

마지막으로 애플리케이션을 닫을 때 호출되는 다른 이벤트 핸들러를 작성합니다. 애플리케이션은 File|Exit를 사용하거나 오른쪽 위 모서리에 있는 **X**를 클릭하여 닫을 수 있습니다. 모든 경우에 프로그램은 데이터베이스에 대한 보류된 업데이트가 없는지 확인하고 변경 사항이 보류된 경우 어떻게 할 것인지 묻는 메시지 윈도우를 표시합니다.

이 코드를 Exit 이벤트 핸들러에 둘 수 있지만, 사용자가 **X**를 사용하여 애플리케이션을 종료하면 보류 중인 모든 데이터베이스 변경 사항이 손실됩니다.

- 1 메인 폼 위에 있는 특정 객체가 아니라 메인 폼 자체를 클릭하여 선택합니다.
- 2 Object Inspector에서 Events 탭을 선택하여 폼 이벤트를 확인합니다.
- 3 *OnClose*를 더블 클릭하거나 *OnClose* 이벤트 옆에 *FormClose*를 입력하고 클릭합니다. 다음과 같이 코드 에디터에서 다른 이벤트 핸들러 뒤에 스켈레톤 *FormClose* 이벤트 핸들러가 작성 및 표시됩니다.

```
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction
&Action)
{
}
}
```

커서가 있는 위치(중괄호 사이)에서 다음을 입력합니다.

```
TMessageButton Option;
TMessageButtons msgButtons;

msgButtons << smbYes << smbNo << smbCancel;

Action = caFree;
if(ClientDataSet1->State == dsEdit || ClientDataSet1->State ==
dsInsert)
    ClientDataSet1->Post();
if(ClientDataSet1->ChangeCount > 0) {

    Option = Application->MessageBox("You have pending updates.
Do you want to write them to the database?", "Pending Updates",
msgButtons, smsWarning, smbYes, smbNo);
    if(Option == smbYes)
        ClientDataSet1->ApplyUpdates(-1);
    else
        if(Option == smbCancel)
            Action = caNone;
}
```

이 이벤트 핸들러는 데이터베이스의 상태를 확인합니다. 보류된 변경 사항은 변경 카운트가 증가하는 클라이언트 데이터셋에 포스트됩니다. 그런 다음 애플리케이션을 닫기 전에 변경 사항의 처리 방법을 묻는 메시지 박스가 표시되며 **Yes**, **No** 또는 **Cancel**로 응답할 수 있습니다. **Yes**는 데이터베이스에 업데이트를 적용하고, **No**는 데이터베이스를 변경하지 않은 상태로 애플리케이션을 닫고, **Cancel**은 애플리케이션 종료를 취소하지만 데이터베이스에 대한 변경 사항을 취소하지 않고 애플리케이션을 실행된 채로 그대로 둡니다.

**4** 전체 절차가 다음과 같이 나타나는지 확인합니다.

```
void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction
&Action)
{
    TMessageButton Option;
    TMessageButtons msgButtons;

    msgButtons << smbYes << smbNo << smbCancel;

    Action = caFree;
    if(ClientDataSet1->State == dsEdit || ClientDataSet1->State ==
dsInsert)
        ClientDataSet1->Post();
    if(ClientDataSet1->ChangeCount > 0) {

        Option = Application->MessageBox("You have pending updates. Do
you want to write them to the database?", "Pending Updates",
msgButtons, smsWarning, smbYes, smbNo);
        if(Option == smbYes)
            ClientDataSet1->ApplyUpdates(-1);
        else
            if(Option == smbCancel)
                Action = caNone;
    }
}
```

**5** 작업을 마치려면 File|Save All을 선택하여 프로젝트를 저장하십시오. 그런 다음 *F9* 키를 눌러 애플리케이션을 실행합니다.

**팁** 오류가 발생하면 오류 메시지를 더블 클릭하여 문제의 코드로 이동하거나, 오류 메시지에 서 *F1* 키를 눌러 도움말을 표시하여 오류를 수정합니다.

이제 모든 내용이 끝났습니다. 애플리케이션을 실행하여 제대로 작동하는지 확인할 수 있습니다. 완전한 기능을 갖춘 File|Exit 명령을 사용하면 프로그램을 종료할 수 있습니다.

## 데스크탑 사용자 정의

이 장에서는 C++Builder IDE에서 도구를 사용자 정의하는 방법에 대해 설명합니다.

### 작업 영역 구성

IDE는 개발을 지원하는 많은 도구를 제공하므로 메뉴 및 툴바 재정렬, 툴 윈도우 결합, 새로운 데스크탑 모양 저장 등을 비롯하여 작업 영역을 최대한 편리하게 재구성할 수 있습니다.

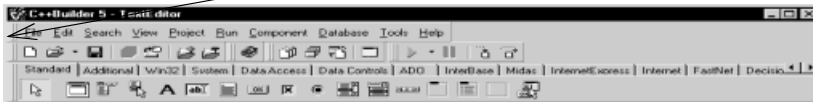
### 메뉴와 툴바 정렬

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



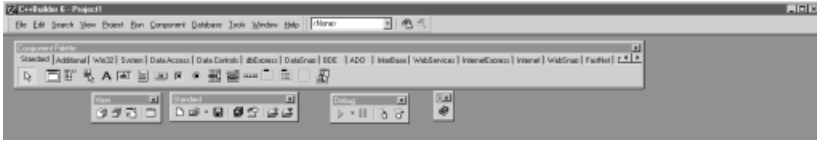
디폴트 정렬 상태  
의 메인 윈도우

메인 윈도우 안에서 툴바와 메뉴를 이동할 수 있습니다. 그래버(왼쪽에 있는 이중 막대)를 클릭하여 원하는 곳으로 끌어 놓습니다.



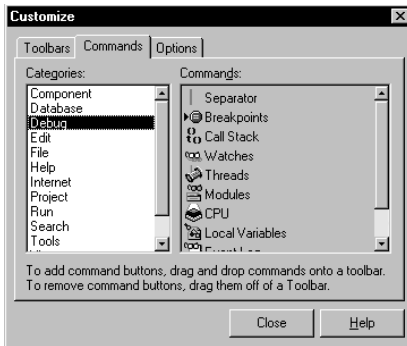
다르게 구성된  
메인 윈도우

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 윈도우입  
니다.

View | Toolbars | Customize를 선택하면 툴바에 도구를 추가하거나 삭제할 수 있습니다. Commands 페이지를 클릭하고 범주를 선택한 다음 명령을 선택하여 원하는 툴바로 끌어 놓습니다.



Commands 페이지에서  
명령을 선택하고 툴바  
로 끌어 놓습니다.

Options 페이지에서  
Show tooltips를 클릭하여  
컴포넌트 및 툴바 아이  
콘에 대한 힌트가 표시  
되는지 확인합니다.

### 자세한 내용은...

온라인 도움말 색인의 "toolbars, customizing"을 참조하십시오.

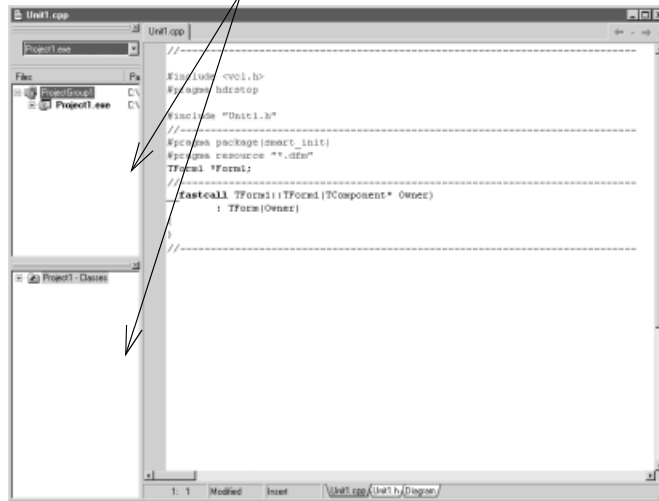
## 툴 윈도우 도킹

툴 윈도우는 개별적으로 열고 닫을 수 있으며 원하는 대로 데스크탑에 정렬할 수도 있습니다. 또한 쉽게 관리할 수 있도록 여러 윈도우를 서로 도킹할 수 있습니다. 여러 윈도우를 함께 움직이도록 연결하는 도킹은 도구를 신속하게 액세스하면서 화면 공간을 효율적으로 사용할 수 있게 도와줍니다.

View 메뉴에서 임의의 툴 윈도우를 연 다음 다른 윈도우와 직접 도킹할 수 있습니다. 예를 들어, C++Builder를 디폴트 구성으로 처음 열 때 ClassExplorer는 코드 에디터의 왼쪽에 도킹됩니다. Project Manager를 처음 두 개 윈도우에 추가하면 세 개의 도킹 윈도우를 만들 수 있습니다.

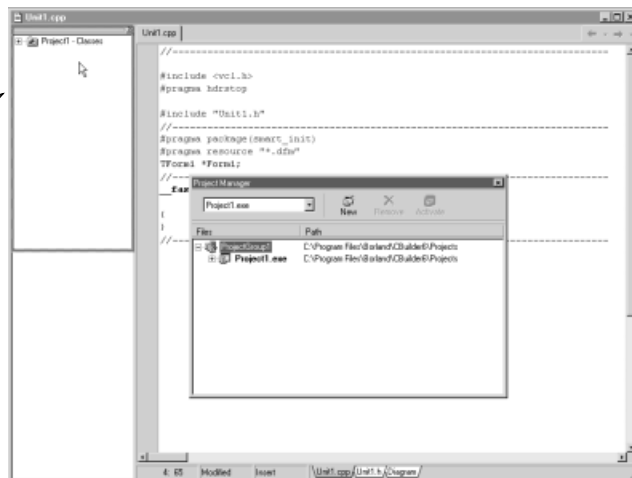
Project Manager 및 Class Explorer가 이와 같이 코드 에디터에 도킹되어 있습니다.

윈도우를 오른쪽에 있는 그래버 또는 탭에 결합하거나 "도킹" 할 수 있습니다.

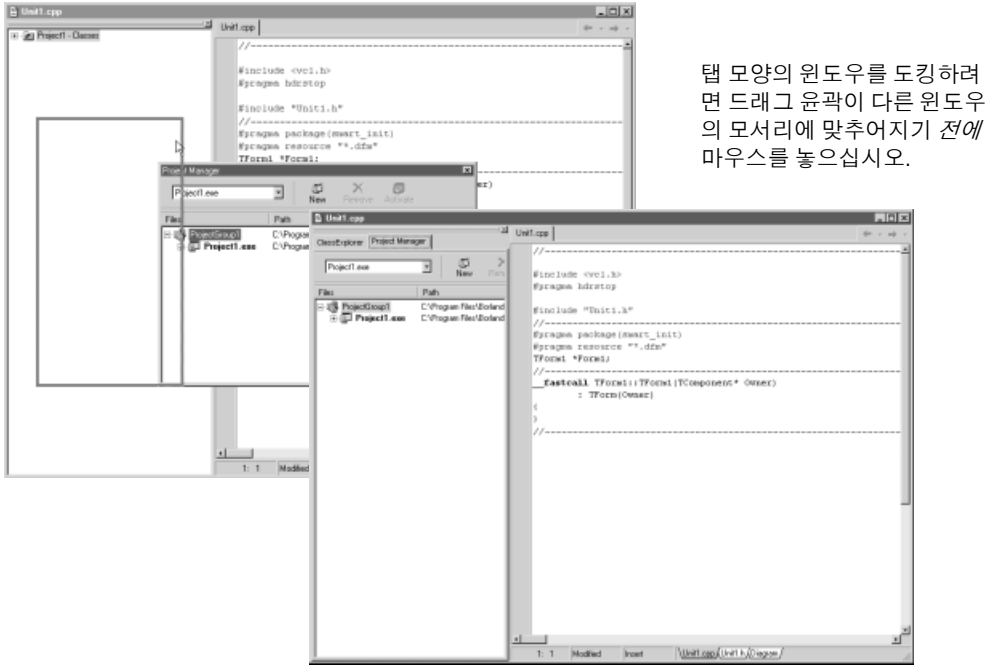


윈도우를 도킹하려면 제목 표시줄을 클릭하여 해당 윈도우를 다른 윈도우로 끌어 놓으십시오. 드래그 윤곽이 직사각형으로 좁아지고 모서리와 맞추어지면 마우스를 놓습니다. 그러면 두 윈도우가 하나로 맞추어집니다.

그래버를 사용하여 윈도우를 도킹시키려면 드래그 윤곽이 윈도우의 모서리와 맞았을 때 마우스를 놓으십시오.



또한 도구를 도킹시켜 탭 모양의 윈도우를 만들 수도 있습니다.



윈도우의 도킹을 해제하려면 그래버 또는 탭을 더블 클릭하거나 탭을 클릭하여 도킹 영역 밖으로 끌어 놓으십시오.

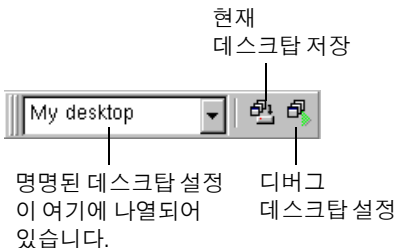
자동 도킹을 사용하지 않으려면 화면에서 윈도우를 이동할 때 **Ctrl** 키를 누르거나, **Tools | Environment Options**를 선택하고 **Preferences** 페이지를 클릭한 다음 **Auto drag docking** 체크 박스를 선택 해제하십시오.

### 자세한 내용은...

온라인 도움말 색인의 "docking"을 참조하십시오.

## 데스크탑 레이아웃 저장

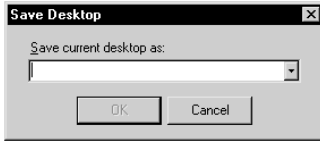
데스크탑 레이아웃을 사용자 정의하고 저장할 수 있습니다. IDE의 **Desktops** 툴바에는 사용 가능한 데스크탑 레이아웃의 선택 리스트와 아이콘 두 개가 있어서 데스크탑을 사용자 정의하기가 쉽습니다.





특정한 윈도우를 표시, 크기 조정 및 도킹하는 것을 비롯하여 데스크탑을 원하는 대로 정렬합니다.

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



저장하려는 데스크탑 레이아웃의 이름을 입력하고 OK를 클릭합니다.

### 자세한 내용은...

온라인 도움말 색인의 "desktop layout"을 참조하십시오.

## 컴포넌트 팔레트 사용자 정의

디폴트 구성에서 컴포넌트 팔레트는 기능적으로 구성된 여러 유용한 VCL 또는 CLX 객체를 탭 모양의 페이지에 표시합니다. 다음을 수행하여 컴포넌트 팔레트를 사용자 정의할 수 있습니다.

- 컴포넌트 숨기기 또는 재정렬
- 페이지 추가, 제거, 재정렬 또는 이름 바꾸기
- 컴포넌트 템플릿 작성 및 팔레트에 추가
- 새 컴포넌트 설치

### 컴포넌트 팔레트 정렬

페이지를 추가, 삭제, 재정렬하고 이름을 바꾸거나 컴포넌트를 숨기거나 재정렬하려면 Palette Properties 다이얼로그 박스를 사용하십시오. 이 다이얼로그 박스를 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.

### 자세한 내용은...

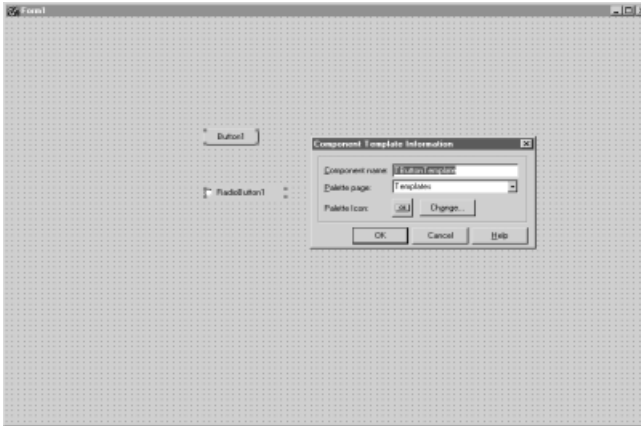
Palette Properties 다이얼로그 박스에서 Help 버튼을 클릭하십시오.

### 컴포넌트 템플릿 만들기

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

컴포넌트 템플릿을 만들려면 하나 이상의 컴포넌트를 폼에 정렬하고 그 속성들을 Object Inspector에서 설정한 다음 그 위로 마우스를 끌어서 모든 컴포넌트를 선택하십시오. 그런 다음 Component | Create Component Template을 선택합니다. Component Template Information 다이얼로그 박스가 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 그것의 이벤트 핸들러를 만들거나 수정할 수 있습니다.



### 자세한 내용은...

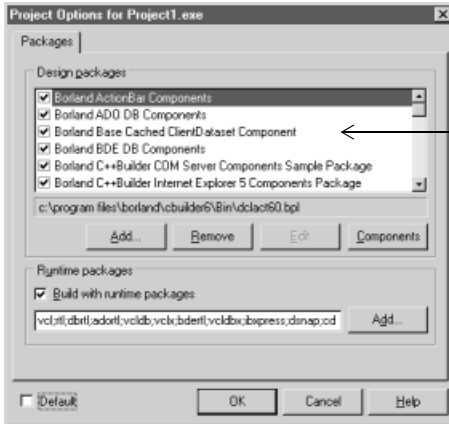
온라인 도움말 색인의 "templates, component"를 참조하십시오.

## 컴포넌트 패키지 설치

사용자 정의 컴포넌트를 작성했거나 업체로부터 구입했다면, 컴포넌트를 컴포넌트 팔레트에 설치하기 전에 *패키지*에 컴파일해야 합니다.

패키지는 C++Builder 애플리케이션과 IDE 사이에서 공유할 수 있는 코드가 들어 있는 특별한 DLL입니다. *런타임 패키지*는 사용자가 애플리케이션을 실행할 때 사용되는 기능을 제공합니다. *디자인 타임 패키지*는 IDE에 컴포넌트를 설치하는 데 사용됩니다. C++Builder 패키지는 .bpl 확장자를 가집니다.

서드파티의 컴포넌트가 이미 패키지에 컴파일되어 있는 경우에는 업체의 지침을 따르거나 Component | Install Packages를 선택합니다.



이러한 컴포넌트들은 C++Builder에 미리 설치되어 있습니다. 서드파티의 새 컴포넌트를 설치할 경우 패키지가 이 리스트에 나타납니다.

패키지에 들어 있는 컴포넌트를 보려면 Components를 클릭합니다.

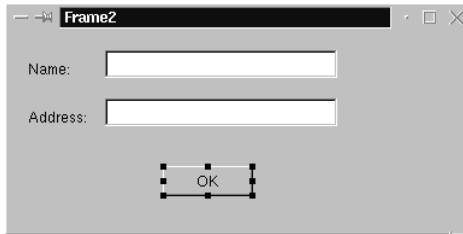
### 자세한 내용은...

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

## 프레임 사용

프레임(TFrame)은 폼과 마찬가지로 재사용할 컴포넌트의 컨테이너입니다. 프레임은 폼보다는 사용자 정의 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 저장하면 프레임은 계속적으로 유닛 기능을 수행하고 컴포넌트(다른 프레임 포함)의 변경 내용을 계속 상속합니다. 프레임이 다른 프레임이나 폼에 포함되면 프레임은 이전 프레임의 변경 내용을 계속 상속합니다.

새 프레임을 열려면 File | New | Frame을 선택하십시오.



프레임에 필요한 비주얼(visual) 컴포넌트나 논비주얼(nonvisual) 컴포넌트 모듈을 추가할 수 있습니다. 새 유닛은 코드 에디터에 자동으로 추가됩니다.

### 자세한 내용은...

도움말 색인의 "frames"와 "TFrame"을 참조하십시오.

## ActiveX 컨트롤 추가

ActiveX 컨트롤을 컴포넌트 팔레트에 추가하여 C++Builder 프로젝트에서 사용할 수 있습니다. Component|Import ActiveX Control을 선택하여 Import ActiveX 다이얼로그 박스를 엽니다. 이 다이얼로그 박스에서 IDE에 설치하기 위하여 새 ActiveX 컨트롤을 등록하거나 이미 등록되어 있는 컨트롤을 선택할 수 있습니다. ActiveX 컨트롤을 설치하면 C++Builder는 해당 컨트롤에 대한 "랩퍼"유닛 파일을 작성 및 컴파일합니다.

### 자세한 내용은...

Component|Import ActiveX Control을 선택하고 Help 버튼을 클릭하십시오.

## 프로젝트 옵션 설정

---

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

## 디폴트 프로젝트 옵션 설정

---

선택한 내용을 모든 새 프로젝트에 대한 디폴트 설정으로 저장하려면 Project Options 다이얼로그 박스의 왼쪽 아래 모서리에 있는 Default에 선택 표시를 하십시오. Default에 선택 표시를 하면 다이얼로그 박스의 현재 설정을 Cbuilder6\Bin 디렉토리에 있는 옵션 파일 default.bpr에 기록합니다. C++Builder의 원래 디폴트 설정을 복원하려면 default.bpr 파일을 삭제하거나 이름을 바꾸십시오.

### 자세한 내용은...

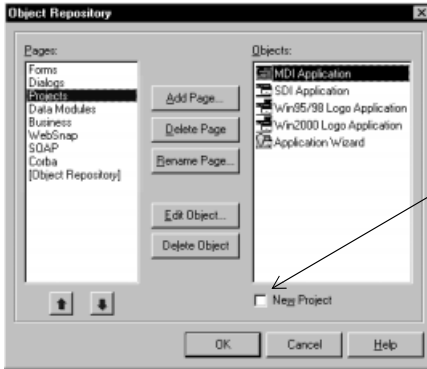
온라인 도움말 색인의 "Project Options dialog box"를 참조하십시오.

## 프로젝트와 폼 템플릿을 기본값으로 지정

---

프로젝트 템플릿을 기본 프로젝트로 지정하지 않은 경우 File|New|Application을 선택하면 C++Builder는 비어 있는 폼을 가진 새로운 표준 애플리케이션을 만듭니다. Project|Add to Repository를 선택하면 고유한 프로젝트를 Projects 페이지에 있는 Object Repository에 템플릿으로 저장할 수 있습니다(6-9 페이지의 "Object Repository에 템플릿 추가" 참조). 또는 Object Repository에서 C++Builder의 기존 프로젝트 템플릿 중 하나를 선택할 수 있습니다(2-5 페이지의 "Object Repository" 참조).

프로젝트 템플릿을 기본값으로 지정하려면 Tools|Repository를 선택하십시오. Object Repository 다이얼로그 박스에서 Pages 아래의 Projects를 선택합니다. Projects 페이지에서 프로젝트를 템플릿으로 저장하면 Objects 리스트에 나타납니다. 템플릿 이름을 선택하고 New Project를 선택한 다음 OK를 클릭합니다.



Object Repository의 페이지에는 프로젝트 템플릿이나 폼 템플릿만 포함되거나 두 템플릿이 모두 포함될 수 있습니다.

프로젝트 템플릿을 기본값으로 설정하려면 Objects 리스트에서 항목을 선택하고 New Project를 선택합니다.

폼 템플릿을 기본값으로 설정하려면 Objects 리스트에서 항목을 선택하고 New Form 또는 Main Form을 선택합니다.

프로젝트 템플릿을 기본값으로 지정하면 File|New|Application을 선택할 때마다 C++Builder는 이 템플릿을 자동으로 엽니다.

디폴트 프로젝트를 지정하는 것과 같은 방법으로 Object Repository에 있는 기존의 폼 템플릿 리스트에서 *디폴트 새 폼*과 *디폴트 메인 폼*을 지정할 수 있습니다. 디폴트 새 폼은 열린 프로젝트에 폼을 추가하기 위해 File|New|Form을 선택할 때 만들어지는 폼입니다. 디폴트 메인 폼은 새 애플리케이션을 열 때 만들어지는 폼입니다. 디폴트 폼을 지정하지 않을 경우, C++Builder는 비어 있는 폼을 사용합니다.

File|New|Other를 선택하고 New Items 다이얼로그 박스에서 다른 템플릿을 선택하면 디폴트 프로젝트 또는 디폴트 폼을 일시적으로 무시할 수 있습니다.

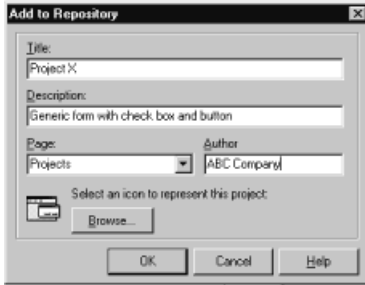
### 자세한 내용은...

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

## Object Repository에 템플릿 추가

자신이 만든 객체를 *템플릿*으로 Object Repository에 추가하여 다시 사용하거나 네트워크에서 다른 개발자와 공유할 수 있습니다. 객체를 재사용하면 일반적인 사용자 인터페이스 및 기능을 갖춘 애플리케이션 제품군을 생성하여 개발 시간을 단축하고 품질을 개선할 수 있습니다.

예를 들어, 프로젝트를 Repository에 템플릿으로 추가하려면 먼저 프로젝트를 저장하고 Project|Add To Repository를 선택하십시오. 그런 다음 Add to Repository 다이얼로그 박스를 완료합니다.



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

다음에 New Items 다이얼로그 박스를 열면 프로젝트 템플릿은 Projects 페이지나 템플릿을 저장한 페이지에 나타납니다. C++Builder를 열 때마다 특정 템플릿을 기본값으로 설정하려면 6-8 페이지의 "프로젝트와 폼 템플릿을 기본값으로 지정"을 참조하십시오.

#### 자세한 내용은...

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

## 도구 환경 설정

---

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

#### 자세한 내용은...

온라인 도움말 색인의 "Environment Options dialog box"를 참조하거나 Environment Options 다이얼로그 박스의 모든 페이지에 있는 Help 버튼을 클릭하십시오.

## 폼 디자이너 사용자 정의

---

Tools | Environment Options 다이얼로그 박스의 Designer 페이지 설정은 폼 디자이너에 영향을 줍니다. 예를 들어, 컴포넌트를 가장 가까운 그리드 라인에 정렬하는 "snap to grid" 기능을 활성화 또는 비활성화할 수 있습니다. 또한 폼 위에 둔 논비주얼 (nonvisual) 컴포넌트의 이름 또는 캡션을 표시하거나 숨길 수 있습니다.

#### 자세한 내용은...

Environment Options 다이얼로그 박스에서 Designer 페이지를 클릭하고 Help 버튼을 클릭하십시오.

## 코드 에디터 사용자 정의

---

바로 사용자 정의할 수 있는 도구는 코드 에디터입니다. Tools | Editor Options 다이얼로그 박스의 여러 페이지에는 코드 편집 방법에 대한 설정이 있습니다. 예를 들어, 키스트로크 매핑, 글꼴, 여백 너비, 색상, 구문 강조, 탭, 들여쓰기 스타일 등을 선택할 수 있습니다.

또한 Editor Options의 Code Insight 페이지에 있는 에디터 안에서 사용할 수 있는 Code Insight 도구를 구성할 수 있습니다. 이러한 도구에 대해 알아보려면 2-6 페이지의 "Code Insight"를 참조하십시오.

### 자세한 내용은...

Editor Options 다이얼로그 박스에서 General, Display, Key Mappings, Color, Code Insight 페이지 등에 있는 Help 버튼을 클릭하십시오.





# 색인

## A

---

About 박스, 추가 4-32  
Action Manager Editor 4-9 ~ 4-12  
ActiveX  
    컨트롤 설치 6-8  
    컴포넌트 팔레트 페이지 3-13  
ADO 3-11

## B

---

BDE 3-11  
BDE Administrator 3-12  
Borland 크로스 플랫폼용 컴포넌트 라이브러리  
(CLX) 3-6  
.BPR 파일 4-2

## C

---

C++Builder  
    사용자 정의 6-1 ~ 6-11  
    프로그래밍 3-1  
C++Builder 버전 3-9  
C++Builder 시작 2-1  
C++Builder 프로그래밍 3-1  
ClassExplorer 2-9  
CLX  
    애플리케이션, 만들기 3-10  
    정의 3-6  
    컴포넌트 추가 2-4  
Code Completion 2-6  
Code Explorer  
    사용 2-9  
Code Parameters 2-6  
Code Templates 2-6

## D

---

Data Dictionary 3-12  
Database Desktop 3-12  
Database Explorer 3-12  
dbExpress 3-11  
.dfm 파일 2-8, 4-1  
Diagram 페이지 2-7  
DLL  
    배포 3-9  
    생성 2-5  
    정의 3-13

## E

---

Editor Options 다이얼로그 박스 2-6, 6-11  
Environment Options 다이얼로그 박스 6-10

## G

---

GUI, 작성 4-2

## I

---

IDE  
    구성 6-1  
    둘러 보기 2-1  
    사용자 정의 6-1 ~ 6-11  
    정의 1-1  
IME 3-9  
InterBase 3-11

## M

---

MainMenu 컴포넌트 5-9  
makefile 4-2

## N

---

New Items 다이얼로그 박스  
    사용 2-5, 4-32  
    템플릿 저장 6-8, 6-10

## O

---

Object Inspector  
    사용 3-4, 4-2  
    인라인 컴포넌트 참조 3-5  
    정의 2-4  
Object Repository  
    사용 2-5  
    정의 2-5, 3-1  
    템플릿 추가 6-8, 6-9  
Object Repository에 항목 추가 2-5  
Object TreeView 2-4  
ODBC 3-11

## P

---

Panel 컴포넌트 5-12  
Paradox 3-11  
.pas 파일 4-1  
Project Manager 2-9  
Project Options 다이얼로그 박스 6-8

## R

---

Resource DLL 마법사 3-9

Run 버튼 5-8

## S

---

SQL Links 3-11

SQL 데이터베이스 서버 3-11

SQL 서버 3-11

SQL 탐색기 3-12

StatusBar1.Panels 다이얼로그 박스 편집 4-5

## T

---

to-do list 2-10

Tooltip Expression Evaluation 2-6

Tooltip Symbol Insight 2-6

## W

---

WebSnap, 소개 3-10

What's New 1-3

## X

---

.xfrm 파일 2-8

## ㄱ

---

개발자 지원 1-7

객체, 정의 3-6

그래픽, 표시 5-12

그림, 표시 5-12

기본

프로젝트와 폼 템플릿 6-8

기술 지원 1-7

## ㄴ

---

뉴스그룹 1-7

## ㄷ

---

다이얼로그 박스, Object Repository 2-5

데스크탑

구성 6-1 ~ 6-5

레이아웃 저장 6-4

데이터 모듈

만들기 2-5

추가 3-2

데이터베이스 애플리케이션

액세스 5-3 ~ 5-4

데이터베이스 애플리케이션, 만들기 3-11

데이터베이스 예제 5-1 ~ 5-16

도움말 툴팁 4-3

도움말 파일, 애플리케이션에 추가 4-30

도움말, F1 1-3

디자인 타임 뷰, 폼 단기 4-3

디폴트

프로젝트 옵션 6-8

## ㅁ

---

마법사, 찾기 2-5

마우스 오른쪽 버튼 메뉴 2-3

메뉴

C++Builder 2-3

구성 2-3, 6-1

애플리케이션에 추가 4-19

컨텍스트 2-3

메시지, 오류 4-29

메인 폼, 정의된 6-9

문자 집합, 확장 3-9

## ㅂ

---

번역 도구 3-9

부모/자식 관계 2-4

비주얼 컴포넌트 라이브러리(VCL)

사용 3-6

컴포넌트 추가 2-4

비트맵, 애플리케이션에 추가 4-7, 4-13

## ㅅ

---

사용자 인터페이스, 작성 3-2, 4-2, 4-3

사용자 정의

IDE 6-1 ~ 6-11

컴포넌트 팔레트 2-3

코드 에디터 6-11

폼 디자이너 6-10

사용자 정의 컴포넌트 설치 6-6

새 폼, 정의된 6-9

설명서, 주문 1-6

소스 코드

작성 시 도움말 2-6

속성 설정 3-4, 4-2, 4-9, 4-11, 4-16

속성, 설정 3-4, 4-2, 4-9, 4-11, 4-16

실행 파일, 배포 3-9

## 0

---

### 애플리케이션

국제화 3-9

데이터베이스 3-11

만들기 3-1, 3-10

배포 3-9

웹 서버 3-10

컴파일 및 디버깅 3-7, 4-13, 4-22

### 애플리케이션 국제화 3-9

### 애플리케이션 배포 3-9

### 애플리케이션 실행 3-7, 4-13, 4-22, 5-8

### 애플리케이션 지역화 3-9

### 애플리케이션 컴파일 3-7

### 액션, 애플리케이션에 추가 4-9, 4-11, 4-17

### 예제 프로그램 4-1 ~ 4-35, 5-1 ~ 5-16

### 오류 메시지 4-29

### 온라인 도움말 파일 1-3

### 옵션, 프로젝트에 대한 설정 6-8

### 웹 사이트, Borland 1-7

### 웹 서버 애플리케이션, 만들기 3-10

### 윈도우 도킹 6-2 ~ 6-4

### 윈도우, 결합 6-2

### 유닛 파일 4-1

### 유닛 헤더 파일 4-1

### 이미지

표시 5-12

### 이미지, 애플리케이션에 추가 4-7, 4-13

### 이벤트 5-13 ~ 5-16

### 이벤트 핸들러 5-13 ~ 5-16

만들기 4-23 ~ 4-29

정의 3-6

## ㅈ

---

### 자습서 4-1 ~ 4-35, 5-1 ~ 5-16

### 저장

데스크탑 레이아웃 6-4

프로젝트 4-2

### 정보, 찾기 1-3

### 지원 서비스 1-7

## ㅊ

---

### 추가

폼에 컴포넌트 4-19

## ㅋ

---

### 컨텍스트 메뉴, 액세스 2-3

### 컨트롤, 폼에 추가 3-2, 4-3

### 컴포넌트

사용자 정의 3-13, 6-5

사용자 정의 만들기 3-13

설치 3-13, 6-6

속성 설정 3-4, 4-2

정의 4-3

컴포넌트 팔레트에 추가 6-5

컴포넌트 팔레트에서 정렬 6-5

폼에 추가 3-2, 4-3

### 컴포넌트 팔레트

사용 3-2

사용자 정의 6-5 ~ 6-7

사용자 정의 컴포넌트 추가 3-13

정의 2-4

페이지 추가 6-5

### 컴포넌트 팔레트, 만들기 6-5

### 코드

보기 및 편집 2-6 ~ 2-9

이벤트 핸들러 3-6

작성 3-6

작성 시 도움말 2-6

### 코드 에디터

다른 윈도우와 결합 6-2

사용 2-6 ~ 2-7

사용자 정의 6-11

### 크로스 플랫폼

애플리케이션 개발 3-10

### 클래스 라이브러리 3-6

### 클래스, 정의 4-4

### 키스트로크 매핑 6-11

## ㅌ

---

### 타입 라이브러리, 정의 3-14

### 탭 모양의 윈도우, 도킹 6-3

### 텍스트 에디터 자습서 4-1 ~ 4-35

### 템플릿

Object Repository에 추가 6-9

기본값으로 지정 6-8

### 통합 개발 환경(IDE)

둘러 보기 2-1

사용자 정의 6-1 ~ 6-11

- 통합 디버거 3-7
- 툴 윈도우, 결합 6-2
- 툴바 2-3
  - 구성 6-1
  - 애플리케이션에 추가 4-13, 4-21
  - 컴포넌트 추가 및 삭제 6-2
- 툴팁 4-3

## 표

---

- 파일
  - 유닛 4-1
  - 저장 4-2
  - 폼 2-8, 4-1
  - 프로젝트 4-1
- 패키지 6-6
- 폼
  - 기본값으로 지정 6-9
  - 닫기 4-3
  - 메인 4-2, 6-9
  - 찾기 2-5
  - 컴포넌트 추가 3-2, 4-3
- 폼 닫기 4-3
- 폼 디자이너
  - 사용자 정의 6-10
  - 정의 2-4
- 폼 파일
  - 정의 4-1
  - 코드 보기 2-8

- 폼에 컴포넌트 추가 4-3
- 표기법 1-7
- 표준 액션, 애플리케이션에 추가 4-17
- 프레임 6-7
- 프로그램
  - CLX 애플리케이션 3-10
  - 국제화 3-9
  - 배포 3-9
  - 웹 서버 애플리케이션 3-10
  - 컴파일 및 디버깅 3-7, 4-13, 4-22
- 프로그램 디버깅 3-7 ~ 3-8, 4-13
- 프로그램 컴파일 4-22, 5-8
- 프로젝트
  - 관리 2-9
  - 기본값으로 지정 6-8
  - 만들기 3-1
  - 옵션을 기본값으로 설정 6-8
  - 저장 4-2
  - 타입 3-9
  - 항목 추가 2-5
  - 형식 3-13
- 프로젝트 그룹 2-9
- 프로젝트 템플릿 6-9
- 프로젝트 파일 4-2
- 프로젝트 파일, 디폴트 이름 4-1

## 응

---

- 헤더 파일 4-1