입문서

Borland®
Kylix™ 2
Delphi™ for Linux®

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

Printed in the U.S.A.

HDE7020WW21000 2E2R1001
0001020304-9 8 7 6 5 4 3 2 1
D3

목 차

1 장

서문 1–1
Kylix란 무엇인가? . . . 1–1
시스템 요구 사양 . . . 1–2
설치 . . . 1–2
Kylix installer . . . 1–2
루트 설치와 루트가 아닌 설치 비교 . . . 1–2
루트 사용자로 설치할 때의 장점 . . . 1–2
표준 사용자로 설치할 때의 장점 . . . 1–3
Kylix 설치 . . . 1–3
Kylix 등록 . . . 1–4
Kylix 시작하기 . . . 1–4
정보 찾기 . . . 1–4
온라인 도움말 . . . 1–5
F1 도움말 . . . 1–5
인쇄된 설명서 . . . 1–6
개발자 지원 서비스 및 웹 사이트 . . . 1–6
Kylix 설치 제거 . . . 1–7
표기법 . . . 1–7

2 장

데스크탑 둘러 보기 2–1
IDE . . . 2–1
메뉴 및 툴바 . . . 2–2
컴포넌트 팔레트, 폼 디자이너 및 Object Inspector . . . 2–3
Object Repository . . . 2–4
코드 에디터 . . . 2–5
Code Insight 툴 . . . 2–6
Code Browsing . . . 2–6
Class Completion . . . 2–7
폼 코드 . . . 2–8
코드 탐색기 . . . 2–8
Project Manager . . . 2–9
Project Browser . . . 2–10
To-Do List . . . 2–10

3 장

Kylix 프로그래밍 3–1
프로젝트 생성 . . . 3–1
사용자 인터페이스 디자인 . . . 3–1
폼에 컴포넌트 두기 . . . 3–2
컴포넌트 속성 설정 . . . 3–2
코드 작성 . . . 3–4
이벤트 핸들러 작성 . . . 3–4
CLX 클래스 사용 . . . 3–4
데이터 모듈 추가 . . . 3–5
프로젝트 컴파일 및 디버깅 . . . 3–6
애플리케이션 배포 . . . 3–7
애플리케이션 국제화 . . . 3–7
프로젝트 유형 . . . 3–8
데이터베이스 애플리케이션 . . . 3–8
클라이언트/서버 애플리케이션 . . . 3–8
웹 서버 애플리케이션 . . . 3–9
웹 서비스 . . . 3–9
공유 객체 . . . 3–10
사용자 지정 컴포넌트 . . . 3–10

4 장

텍스트 에디터(자습서) 만들기 4–1
새 애플리케이션 시작 . . . 4–1
속성 값 설정 . . . 4–2
폼에 컴포넌트 추가 . . . 4–3
메뉴와 툴바에 대한 지원 추가 . . . 4–5
액션 리스트에 액션 추가 . . . 4–7
액션 리스트에 표준 액션 추가 . . . 4–9
이미지 목록에 이미지 추가 . . . 4–10
메뉴 추가 . . . 4–12
텍스트 영역 지우기 . . . 4–15
툴바 추가 . . . 4–15
이벤트 핸들러 작성 . . . 4–16
New 명령에 대한 이벤트 핸들러 만들기 . . . 4–17
Open 명령에 대한 이벤트 핸들러 만들기 . . . 4–19
Save 명령에 대한 이벤트 핸들러 만들기 . . . 4–20
Save As 명령에 대한 이벤트 핸들러 만들기 . . . 4–21
Exit 명령에 대한 이벤트 핸들러 만들기 . . . 4–22
About 상자 만들기 . . . 4–23
애플리케이션 완료 . . . 4–25

5 장

데이터베이스 애플리케이션 만들기–자습서 5–1
데이터베이스 아키텍처 개요 . . . 5–1
새 프로젝트 만들기 . . . 5–2
데이터 액세스 컴포넌트 설정 . . . 5–2
데이터베이스 연결 설정 . . . 5–3
단방향 데이터셋 설정 . . . 5–4
프로바이더, 클라이언트 데이터셋, 데이터 소스 설정 . . . 5–5
사용자 인터페이스 디자인 . . . 5–5

그리드와 탐색 막대 만들기 5-6
메뉴에 대한 지원 추가 5-7
메뉴 추가. 5-9
버튼 추가. 5-11
제목과 이미지 표시. 5-11
이벤트 핸들러 작성. 5-12
Update Now! 명령의 이벤트 핸들러 작성 . 5-12
Exit 명령의 이벤트 핸들러 작성 5-13
FormClose 이벤트 핸들러 작성 5-14

6 장

데스크탑 사용자 지정 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 지정 6-5
컴포넌트 팔레트 정렬. 6-5
컴포넌트 템플릿 생성. 6-6
컴포넌트 패키지 설치. 6-7
프레임 사용. 6-8
프로젝트 옵션 설정. 6-8
기본 프로젝트 옵션 설정 6-8
프로젝트와 폼 템플릿을 기본값으로 지정 . . . 6-9
Object Repository에 템플릿 추가. 6-9
툴 환경 설정. 6-10
폼 디자이너 사용자 지정 6-10
코드 에디터 사용자 지정 6-11
코드 탐색기 사용자 지정 6-11
도움말 항목 인쇄. 6-12
파일로 인쇄 6-12
프린터로 직접 인쇄. 6-12

색인 I-1

1

서문

*입문서*는 사용자들이 본 제품을 즉시 사용할 수 있도록 Kylix 개발 환경에 대한 개요를 제공합니다. 아울러 Kylix에서 사용할 수 있는 툴과 기능에 대한 자세한 내용을 어디서 찾을 수 있는지 알려 줍니다.

이 장에서는 Kylix를 설치, 등록, 시작하는 방법뿐만 아니라 정보를 찾는 방법에 대해서 설명합니다. 2장 "데스크탑 둘러 보기"에서는 데스크탑의 주요 툴과 통합 데스크탑 환경(IDE)에 대해 설명합니다. 3장 "Kylix 프로그래밍"에서는 이러한 툴을 사용하여 애플리케이션 또는 공유 객체를 만드는 방법에 대해 설명합니다. 4장 "텍스트 에디터(자습서) 만들기"에서는 텍스트 편집기 자습서를 통해 편집기를 만드는 방법을 단계별로 설명합니다. 5장 "데이터베이스 애플리케이션 만들기 – 자습서"에서는 데이터베이스 애플리케이션 자습서를 통해 데이터베이스 애플리케이션을 만드는 방법을 단계별로 설명합니다. 6장 "데스크탑 사용자 지정" 에서는 개발 목적에 따라 Kylix IDE를 사용자 지정하는 방법에 대해 설명합니다.

Kylix란 무엇인가?

Kylix는 신속한 애플리케이션 개발(RAD)을 위한 비주얼 환경을 제공하는 객체 지향 프로그램입니다. Kylix를 사용하면 코딩을 최소화하면서 효율이 높은 Intel 아키텍처용 32비트 Linux 애플리케이션을 만들 수 있습니다. Kylix는 애플리케이션을 개발, 테스트, 디버그, 배포하는 데 필요한 모든 툴, 즉 재사용할 수 있는 컴포넌트들로 이루어진 대형 라이브러리, 디자인 툴, 애플리케이션 템플릿, 프로그래밍 마법사 등을 제공합니다. 이러한 툴을 사용하여 프로토타입을 쉽게 만들 수 있을 뿐만 아니라 개발 기간도 단축시킬 수 있습니다.

Kylix는 기업용 에디션, 전문가용 에디션, 오픈 에디션의 세 가지 에디션으로 구성되어 있습니다. 사용할 수 있는 기능과 컴포넌트는 구입한 Kylix 에디션에 따라 달라집니다. 에디션에 따라 사용할 수 있는 툴을 보려면 http://www.borland.com/kylix에 있는 기능 목록을 참조하십시오.

시스템 요구 사양

Kylix를 실행하려면 다음과 같은 소프트웨어를 반드시 설치해야 합니다.

- Linux 커널 버전 2.2 이상
- libgtk.so 버전 1.2 이상(그래픽 설치에만 필요함)
- libjpeg 버전 6.2(libjpeg.so.62) 이상
- X11R6 호환 터미널 서버(예: XFree86)

설치

경고 Kylix를 설치하거나 제거할 때 RPM, kpackage, gnorpm과 같은 패키지 관리자(Package Manager)는 사용하지 마십시오. 언제나 Borland에서 제공하는 설치/해제 프로그램만 사용해야 합니다.

Linux의 ext2처럼 심볼릭 링크를 지원하는 파일 시스템에 Kylix를 설치해야 합니다. FAT, FAT32, Samba를 비롯한 SMB, Novell과 같은 파일 시스템에 Kylix를 설치하면 안 됩니다.

Kylix installer

Kylix는 RPM 시스템과 RPM이 아닌 시스템에 모두 잘 설치됩니다. Kylix installer는 설치된 소프트웨어를 반영하도록 RPM 패키지 데이터베이스를 업데이트합니다. 설치 프로그램은 RPM에 대해 독립적으로 작동할 수 있습니다.

설치 시스템에 RPM이 있고 Kylix가 루트 사용자로 설치되면 일부 Kylix 패키지가 설치 패키지 목록에 추가됩니다. Kylix 설치 제거는 이 패키지를 제거합니다. 패키지 이름은 언제나 "kylix_"로 시작합니다. 시스템에 RPM이 없거나 Kylix를 루트가 아닌 다른 사용자로 설치한다면 installer는 RPM 데이터베이스에 엔트리를 추가하지 않고 Kylix를 설치합니다.

루트 설치와 루트가 아닌 설치 비교

필요한 디스크 공간만 있다면 어떤 사용자라도 Kylix를 설치할 수 있습니다. 시스템의 루트 계정에 대한 암호를 모르는 경우 루트가 아닌 계정을 사용하여 Kylix를 설치합니다. 루트 암호를 알고 있다면 Kylix를 루트 사용자로 설치할지, 다른 사용자로 설치할지 결정해야 합니다.

루트 사용자로 설치할 때의 장점

- 설치 프로그램이 시스템에 대한 완벽한 액세스 권한을 갖고 있으면 라이브러리 링크의 누락 등과 같은 설치 시 발생하는 문제를 설치 프로그램에서 자동으로 해결할 수 있습니다.
- 모든 파일은 기준이 되는 가운데 위치에 설치되며 다른 어떤 사용자라도 Kylix를 실행할 수 있습니다. Kylix Object Repository를 사용하면 사용자 간에 CLX 객체를 공유할 수 있습니다.

- 시스템에서 RPM Package Manager를 사용하면 installer는 Kylix 항목 집합을 RPM 데이터베이스에 추가합니다. 이렇게 하면 Kylix와 다른 소프트웨어 간의 종속 관계가 자동으로 관리됩니다.

표준 사용자로 설치할 때의 장점

- 모든 Kylix 파일을 소유하게 되고 데모 프로젝트를 사용하는 추가 단계를 수행하거나 Object Repository를 수정할 필요가 없게 됩니다.
- 일부 설치 옵션을 생략하고 나중에 추가할 예정이라면 installer는 이 옵션들을 사용할 수 있도록 .borland/delphi60rc 파일을 자동으로 구성합니다. 루트로 설치하면 이 파일을 업데이트하는 유일한 방법은 파일을 지워서 Kylix가 다시 생성하도록 하는 것입니다. 그러나 이 파일을 지우면 IDE 옵션 변경 내용이 소실됩니다.

Kylix 설치

Kylix를 설치하려면 다음 단계를 따릅니다.

1 수퍼유저로 로그인합니다(일부 사용자만 Kylix에 액세스한다면 이 단계는 무시해도 됩니다).

   - X Windows 시스템을 사용하고 있으면 터미널 창을 엽니다.

2 드라이브에 CD-ROM을 넣고 CD-ROM 파일 시스템을 마운트합니다(일부 시스템에서는 루트 사용자만 CD-ROM을 마운트할 수 있습니다).

3 CD-ROM이 마운트된 위치로 작업 디렉토리를 바꿉니다.

4 Kylix installer인 setup.sh을 실행합니다. CD-ROM 파일 시스템이 실행 파일을 지원하도록 구성되지 않았으면 셸 명령인 "sh setup.sh"을 실행해야 합니다.

   Kylix의 기본 설치 위치는 언제나 설치 사용자의 홈 디렉토리 바로 아래입니다. 어떤 경우라도 /root(루트 사용자의 홈 디렉토리, 루트로 설치할 때 기본 위치임) 디렉토리에 설치하면 안 됩니다. 다른 사용자는 일반적으로 /root 디렉토리에 액세스할 수 없으며 Kylix를 루트 사용자로 실행하는 것은 바람직하지 않습니다. Kylix를 루트로 설치할 때 /usr/local/kylix2처럼 항상 다른 설치 위치를 지정하십시오.

5 installer 프롬프트 메시지를 따릅니다.

   예를 들어 CD-ROM 장치를 /mnt/cdrom에 마운트하도록 구성된 시스템에서 다음과 같은 셸 명령으로 CD-ROM 파일 시스템을 마운트하고 installer를 실행합니다.

```
mount /mnt/cdrom
cd /mnt/cdrom
./setup.sh
```

   마운트 위치가 /mnt/cdrom이 아닌 다른 곳이라면 예제에서 마운트 위치를 변경하십시오.

참고 X Windows에서 실행하고 올바른 libgtk.so 버전을 사용한다면 Kylix installer는 X Windows 대화 상자를 엽니다. 그렇지 않으면 Kylix installer는 일련의 텍스트 콘솔 프롬프트를 사용합니다. 둘 중 어떤 경우라도 Install Path와 Link Path를 지정하는 프롬프트 메시지가 나타납니다.

참고 루트로 Kylix를 설치해도 Kylix는 일반 사용자로 실행해야 합니다. 일반 사용자로 Kylix를 설치하면 루트를 제외하고는 설치한 사용자만 Kylix를 실행할 수 있습니다.

자세한 내용은...

Kylix 설치에 대한 전체 내용은 http://www.borland.com/techpubs/kylix를 참조하십시오.

Kylix 등록

이 제품은 일련 번호와 인증 키가 있어야 사용할 수 있습니다. Kylix를 설치한 다음에는 제품을 등록해야 합니다.

설치가 끝나면 나타나는 Registration 대화 상자에서는 온라인(안전한 연결로 직접 등록) 등록, 전화로 등록, 웹으로 등록의 3가지 등록 방법을 제공합니다.

각 등록 옵션을 선택하면 필요할 때마다 도움말이 제공되는 마법사가 실행됩니다.

자세한 내용은...

Kylix 등록에 관한 전체 내용은 http://www.borland.com/techpubs/kylix를 참조하십시오.

Kylix 시작하기

다음 방법으로 Kylix를 시작할 수 있습니다.

- 셸 창에서 startkylix를 입력합니다. 설치 시 선택한 링크 경로(link path)가 사용자의 경로에 있으면 이 명령이 잘 실행됩니다.
- {install path}/startkylix를 입력합니다. 예를 들어 설치 경로가 /usr/local/kylix2이면 /usr/local/kylix2/bin/startkylix를 입력합니다.
- Gnome 또는 KDE의 Application Starter 메뉴에서 Kylix를 실행합니다.

정보 찾기

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

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

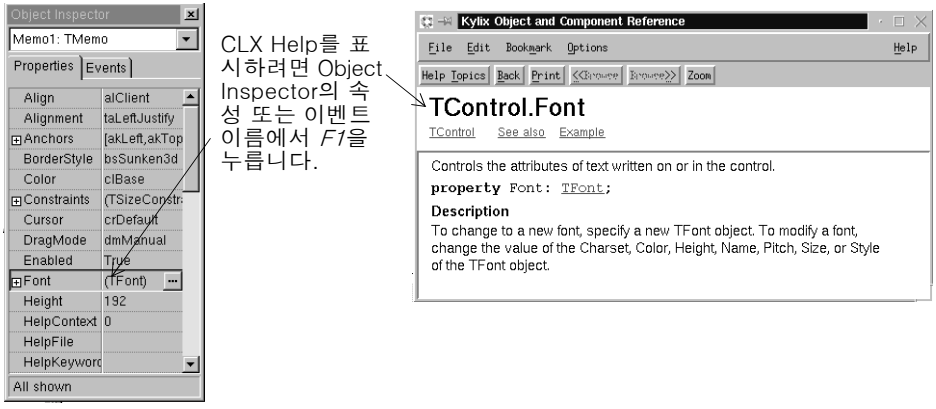
온라인 도움말

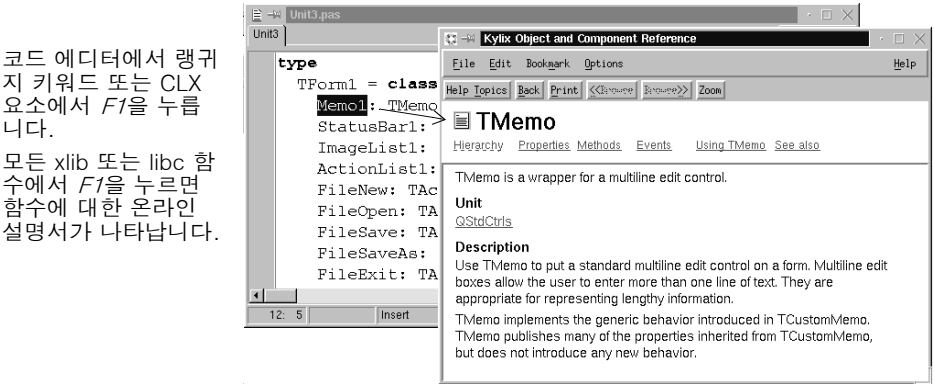
온라인 도움말 시스템은 Borland 크로스 플랫폼용 컴포넌트 라이브러리(CLX)의 컴포넌트, 사용자 인터페이스 기능, 랭귀지 구현, 프로그래밍 작업 등에 대한 자세한 정보를 제공합니다.

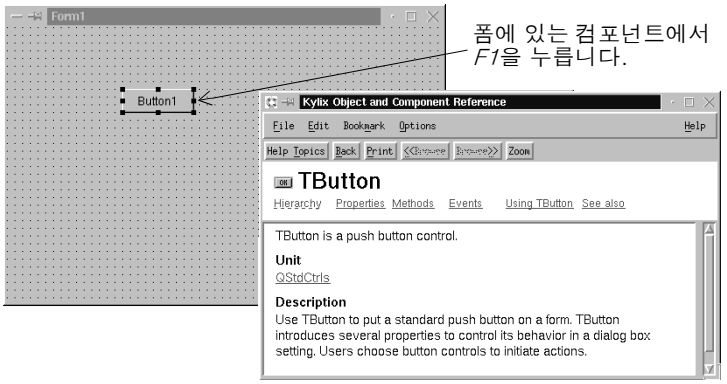
도움말 목차를 보려면 Help|Kylix Help를 선택하고 Contents 탭을 클릭합니다. CLX 객체에 대한 정보나 기타 다른 내용을 찾으려면 Index 또는 Find 탭을 클릭하고 찾을 내용을 입력합니다.

F1 도움말

도움말 시스템은 Kylix의 CLX 및 기타 다른 부분에 대한 광범위한 자료를 제공합니다. 항목을 선택하고 *F1*을 누르면 메뉴 항목, 대화 상자, 툴바, 컴포넌트 등을 비롯한 개발 환경의 모든 부분에서 문맥에 맞는 도움말을 볼 수 있습니다.







모든 대화 상자에서 Help 버튼을 누르면 문맥에 맞는 온라인 설명서가 나타납니다.

컴파일러와 링커에서 발생한 오류 메시지는 코드 에디터 아래에 별도의 창으로 나타납니다. 컴파일 오류에 대한 도움말이 필요하면 목록에서 메시지를 선택하고 *F1*을 누릅니다.

자세한 내용은...

도움말 시스템을 구성하여 Kylix Help 항목을 인쇄하는 방법은 6-12페이지의 "도움말 항목 인쇄"를 참조하십시오.

인쇄된 설명서

*입문서*는 Kylix를 소개하는 내용으로 구성되어 있습니다. *개발자 안내서*와 같은 인쇄된 설명서를 추가로 주문하려면 http://shop.borland.com을 참조하십시오.

개발자 지원 서비스 및 웹 사이트

Borland는 다양한 개발자 커뮤니티의 요구에 부응하기 위해 다양한 지원 옵션을 제공합니다. 지원에 대해 알아보려면 http://www.borland.com/devsupport/를 참조하십시오.

웹 사이트를 통해 Kylix 개발자들이 정보, 팁, 기술 등을 교환하는 여러 뉴스그룹에 액세스할 수 있습니다. 또한 이 사이트에는 Kylix에 관한 도서 목록이 있습니다.

Kylix 설치 제거

설치된 Kylix를 제거하려면 Kylix 설치 디렉토리에서 설치했을 때와 동일한 사용자로 설치 제거 프로그램을 실행합니다. 예를 들어 Kylix를 /usr/local에서 루트 사용자로 설치했다면 루트 사용자로 /usr/local/kylix2/uninstall을 실행합니다. 설치 제거를 실행하면 설치 시 하드 드라이브로 복사되었던 파일이 제거됩니다. 그러나 이전에 installer가 수정한 기존 구성 파일은 복구되지 않습니다.

표기법

이 설명서에서는 특별한 텍스트를 표시하기 위하여 다음과 같은 글꼴을 사용합니다.

표 1.1 표기법

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

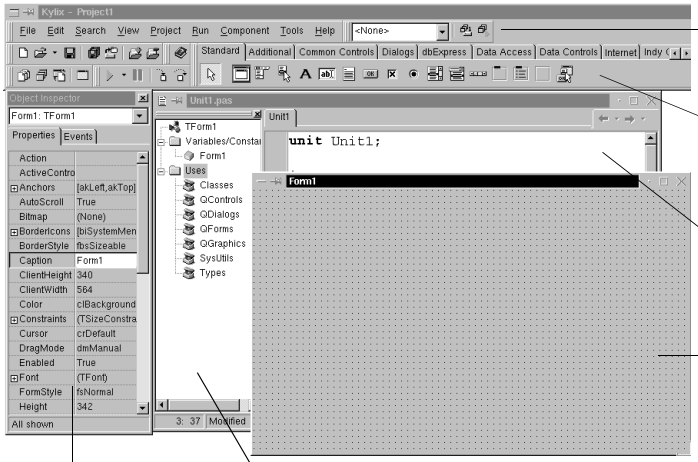
2

데스크탑 둘러 보기

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

IDE

Kylix를 처음 시작하면 IDE에 있는 주요 툴 몇 가지를 볼 수 있습니다. Kylix의 IDE에는 메뉴, 툴바, 컴포넌트 팔레트, Object Inspector, 코드 에디터, Code Explorer, Project Manager 및 기타 다른 툴이 많이 들어 있습니다. 사용할 수 있는 기능과 컴포넌트는 구입한 Kylix 에디션에 따라 다릅니다.



메뉴와 툴바의 다양한 툴에 액세스하여 애플리케이션을 쉽게 작성할 수 있습니다.

컴포넌트 팔레트는 프로젝트에 추가할 수 있는 미리 만들어진 컴포넌트를 포함합니다.

코드 에디터는 보거나 편집할 코드를 표시합니다.

폼 디자이너는 애플리케이션용 사용자 인터페이스의 디자인을 시작할 수 있는 빈 폼을 포함합니다. 애플리케이션은 폼을 여러 개 포함할 수 있습니다.

Object Inspector는 객체 속성을 변경하고 이벤트 핸들러를 선택하는 데 사용됩니다.

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

Kylix의 개발 모델은 *two-way* 툴을 기반으로 합니다. 이것은 비주얼 디자인 툴과 텍스트 기반의 코드 편집 사이를 자유롭게 이동할 수 있다는 것을 의미합니다. 예를 들어 폼 디자이너를 사용하여 그래픽 인터페이스에 있는 버튼 및 기타 요소를 정렬하는 즉시 폼에 대한 텍스트 코드가 있는 폼 파일을 볼 수 있습니다. 또한 비주얼 프로그래밍 환경에서 Kylix에 의해 생성된 모든 코드를 직접 편집할 수 있습니다.

IDE에서는 모든 프로그래밍 툴을 쉽게 찾아서 사용할 수 있습니다. IDE를 종료하지 않아도 그래픽 인터페이스 디자인, 클래스 라이브러리 찾기, 코드 작성, 프로젝트 컴파일, 테스트, 디버깅 및 관리 등을 수행할 수 있습니다.

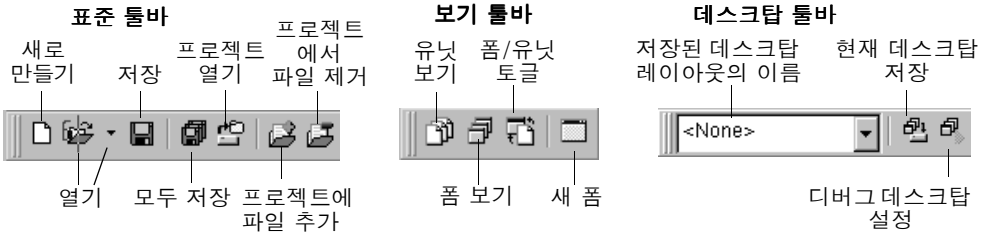
IDE를 구성하는 방법을 보려면 5장 "데이터베이스 애플리케이션 만들기 – 자습서"를 참조하십시오.

메뉴 및 툴바

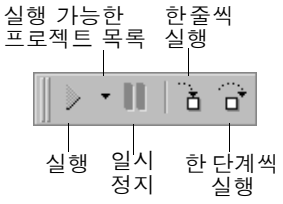
화면의 상단을 차지하는 메인 윈도우에는 메뉴, 툴바 및 컴포넌트 팔레트가 있습니다.

기본 정렬된 메인 윈도우

Kylix의 툴바를 사용하면 자주 사용하는 작업과 명령에 빠르게 액세스할 수 있습니다. 모든 툴바 작업은 드롭다운 메뉴에도 있습니다.



디버그 툴바



버튼이 수행하는 작업을 알아보려면 툴팁이 나타날 때까지 버튼에 마우스를 갖다 대십시오.

마우스 오른쪽 버튼 메뉴를 사용하여 툴바를 숨길 수 있습니다. 숨겨진 툴바를 표시하려면 View| Toolbars를 선택하고 표시할 툴바를 선택하십시오.

툴바 버튼뿐만 아니라 키보드 단축키도 많이 사용됩니다. 키보드 단축키는 항상 드롭다운 메뉴에 있는 명령 옆에 나타납니다.

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

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

자세한 내용은...

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

컴포넌트 팔레트, 폼 디자이너 및 Object Inspector

컴포넌트 팔레트, 폼 디자이너, Object Inspector를 함께 사용하여 애플리케이션의 사용자 인터페이스를 디자인할 수 있습니다.

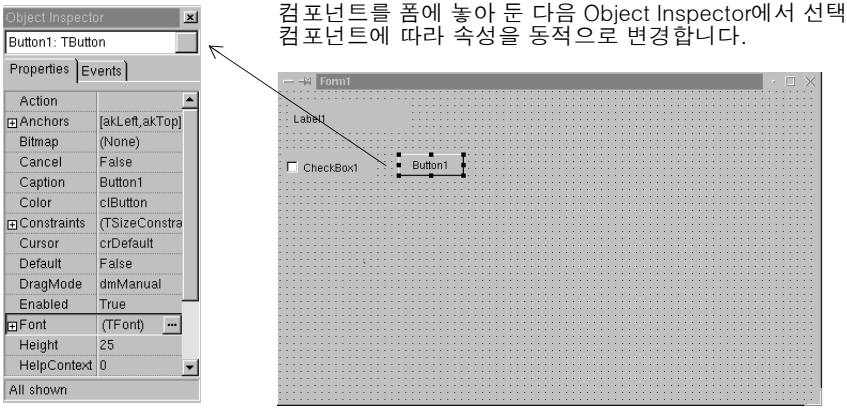
*컴포넌트 팔레트*에는 탭 모양의 페이지가 있는데 여기에는 인터페이스를 디자인하는 데 사용하는 비주얼 및 넌비주얼 CLX 컴포넌트를 나타내는 아이콘이 여러 개 있습니다. 각 페이지에는 컴포넌트들이 기능별로 그룹화되어 있습니다. 예를 들어 Dialogs 페이지에는 파일 열기 및 저장과 같은 파일 작업에 사용하는 일반 대화 상자가 있습니다.

기능별로 그룹화된 컴포넌트 팔레트 페이지

컴포넌트

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

폼이나 *폼 디자이너*에 컴포넌트를 놓은 다음 사용자 인터페이스에 나타낼 형태로 컴포넌트를 정렬할 수 있습니다. 폼에 놓아 둔 컴포넌트에 대해서는 *Object Inspector*를 사용하여 디자인 타임 속성을 설정하고 이벤트 핸들러를 만들며 보이는 속성 및 이벤트를 필터링하여 애플리케이션의 외관과 애플리케이션을 실행시키는 코드를 연결할 수 있습니다. 3-2페이지의 "폼에 컴포넌트 두기"를 참조하십시오.



컴포넌트를 폼에 놓아 둔 다음 Object Inspector에서 선택한 컴포넌트에 따라 속성을 동적으로 변경합니다.

자세한 내용은...

도움말 색인의 "Component palette" 및 "Object Inspector"를 참조하거나 3-1페이지의 "사용자 인터페이스 디자인"을 참조하십시오.

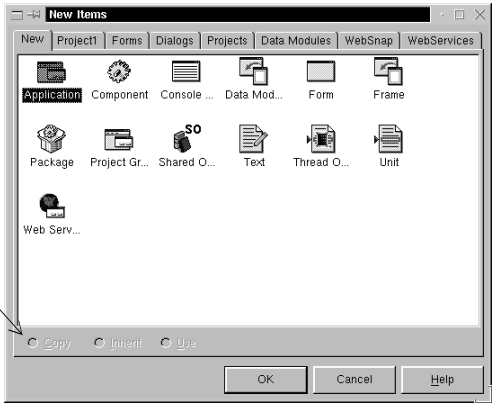
Object Repository

Object Repository를 New Items 대화 상자라고도 하며 New Items 대화 상자에는 폼, 대화 상자, 데이터 모듈, 마법사, 공유 객체 파일, 예제 애플리케이션 및 개발을 쉽게 하기 위한 기타 항목들이 포함되어 있습니다. 프로젝트를 시작할 때 New Items 대화 상자를 표시하려면 File|New를 선택합니다. 만들려는 것과 유사한 객체가 있는지 확인하려면 Repository를 선택하십시오.

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

Object Repository에 있는 항목을 복사, 상속, 사용하여 새로운 항목을 만들 수 있습니다.

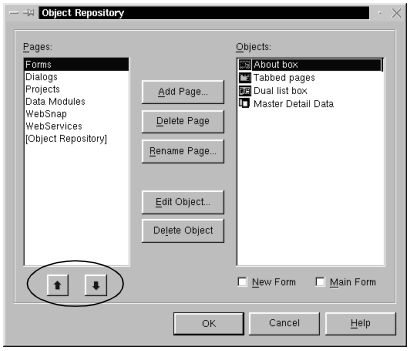
Copy(기본값)는 프로젝트에서 항목의 복사본을 만듭니다. *Inherit*는 Repository의 객체에 대한 변경 내용을 프로젝트의 객체가 상속받는다는 것을 의미합니다. *Use*는 프로젝트의 객체에 대한 변경 내용을 Repository의 객체가 상속받는다는 것을 의미합니다.



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

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

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



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

자세한 내용은...

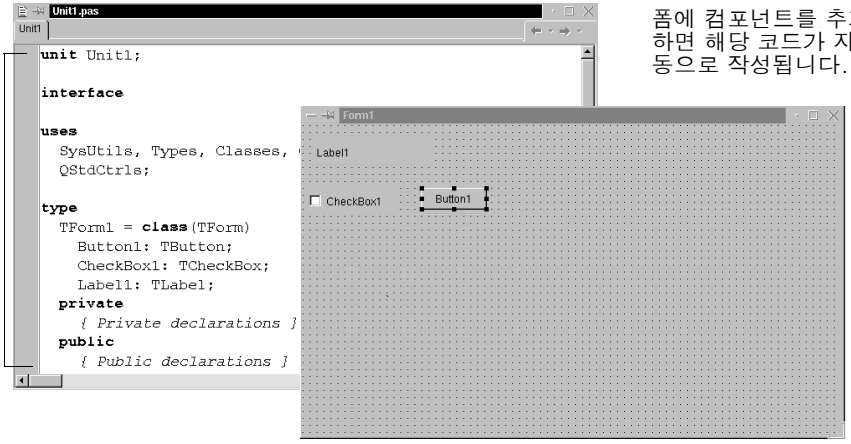
도움말 색인의 "Object Repository"를 참조하십시오. 사용할 수 있는 객체는 구입한 Kylix 에디션에 따라 다릅니다.

코드 에디터

애플리케이션에 대한 사용자 인터페이스를 디자인할 때 Kylix는 기본적인 오브젝트 파스칼 코드를 생성합니다. 폼과 컴포넌트의 속성을 선택하고 수정하면 변경 내용이 소스 파일에 자동으로 반영됩니다. ASCII 편집기 기능을 모두 갖춘 기본 제공 코드 에디터를 사용하면 소스 파일에 코드를 직접 추가할 수 있습니다.

생성된 코드

폼에 컴포넌트를 추가하면 해당 코드가 자동으로 작성됩니다.



코드 에디터에서는 Code Insight와 Class Completion과 같은 다양한 기능을 사용하여 코드를 보거나 작성할 수 있습니다.

자세한 내용은...

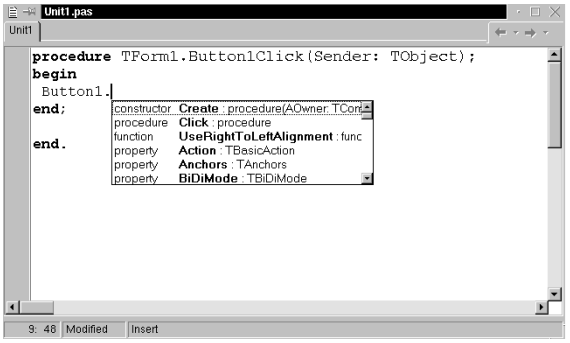
도움말 색인의 "Code editor"를 참조하십시오. 코드 편집 환경을 사용자 지정하려면 6-11페이지의 "코드 에디터 사용자 지정"을 참조하십시오.

Code Insight 툴

Code Insight 툴은 컨텍스트 팝업 창을 표시합니다.

표 2.1 Code Insight 툴

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



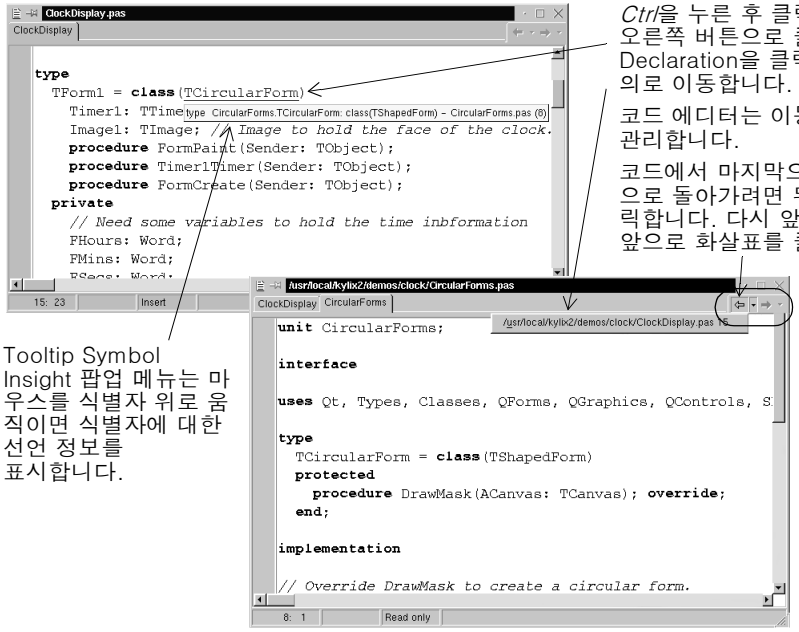
`Button1.`과 같이 점을 입력하면 Code Completion은 클래스에 대한 속성, 메소드, 이벤트 등의 목록을 표시합니다. 코드를 계속 입력하면 목록에서 해당 클래스에 속하는 선택 영역을 자동으로 필터링합니다. 목록에서 항목을 선택하고 *Enter* 키를 눌러 코드에 항목을 추가합니다.

이 툴들을 선택 또는 선택 해제하려면 Tools|Editor Options를 선택한 후 Code Insight 탭을 클릭합니다. Automatic features 섹션에서 툴을 선택하거나 선택을 해제합니다.

Code Browsing

클래스, 변수, 속성, 메소드 또는 다른 식별자에 마우스를 갖다 대면 Tooltip Symbol Insight라는 팝업 메뉴에서 식별자가 선언된 위치를 표시합니다. *Ctrl*을 누르면 커서가 손 모양으로 바뀌고 식별자는 파란색으로 변하면서 밑줄이 그어집니다. 이 식별자를 클릭하면 식별자 정의로 이동합니다.

코드 에디터에는 웹 브라우저처럼 앞으로 버튼과 뒤로 버튼이 있습니다. 코드 에디터는 사용자가 식별자 정의로 이동할 때 코드에서 이동한 위치를 추적합니다. 앞으로 버튼과 뒤로 버튼 옆에 있는 드롭다운 화살표를 클릭하면 이러한 참조 기록에서 앞뒤로 이동할 수 있습니다.



*Ctrl*을 누른 후 클릭하거나 마우스 오른쪽 버튼으로 클릭한 다음 Find Declaration을 클릭하고 식별자 정의로 이동합니다.

코드 에디터는 이동한 정의 목록을 관리합니다.

코드에서 마지막으로 작업하던 곳으로 돌아가려면 뒤로 화살표를 클릭합니다. 다시 앞으로 이동하려면 앞으로 화살표를 클릭합니다.

Tooltip Symbol Insight 팝업 메뉴는 마우스를 식별자 위로 움직이면 식별자에 대한 선언 정보를 표시합니다.

Ctrl+Shift+↑ 또는 *Ctrl+Shift+*↓를 눌러 프로시저 선언과 구현 사이를 자유롭게 이동할 수 있습니다.

Class Completion

Class Completion은 클래스의 뼈대 코드를 생성합니다. 클래스 선언 내에서 커서를 임의의 위치에 둔 다음 *Ctrl+Shift+C*를 클릭하거나 마우스 오른쪽 버튼을 클릭하고 Complete Class at Cursor를 선택합니다. Kylix는 private **read** 지정자와 private **write** 지정자가 필요한 모든 속성 선언에 이 지정자들을 추가한 다음 모든 클래스 메소드에 대한 뼈대 코드를 생성합니다. Class Completion을 사용하면 이미 구현한 메소드에 대한 클래스를 선언할 수도 있습니다.

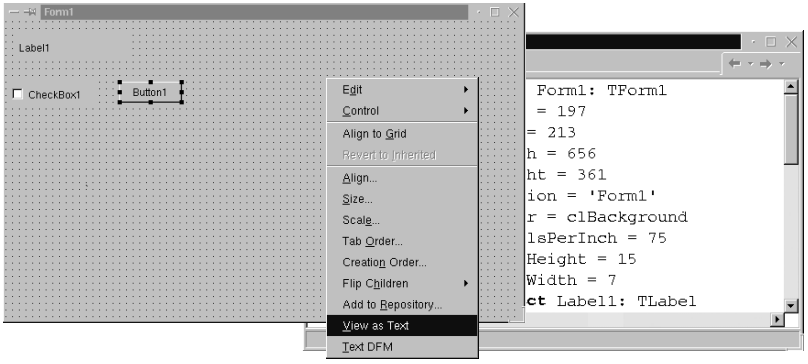
Class Completion을 구성하려면 Tools|Environment Options를 선택하고 Explorer 탭을 클릭합니다.

자세한 내용은...

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

폼 코드

폼은 대부분의 Kylix 프로젝트에서 매우 가시적인 부분으로서 폼에서 애플리케이션의 사용자 인터페이스를 디자인합니다. 일반적으로 Kylix의 비주얼 툴을 사용하여 폼을 디자인한 다음 폼 파일로 저장합니다. 폼 파일(.xfm)은 모든 영구적인 속성 값들을 비롯하여 폼의 각 컴포넌트에 대한 코드입니다. 코드 에디터에서 폼 파일을 보거나 편집하려면 폼을 마우스 오른쪽 버튼으로 클릭하고 View as Text를 선택합니다. 폼의 그래픽 보기로 돌아가려면 마우스 오른쪽 버튼을 클릭하고 View as Form을 선택합니다.



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

텍스트(기본값) 형식이나 바이너리 형식으로 폼 파일을 저장할 수 있습니다. Environment Options 대화 상자를 사용하여 새로 만든 폼에 사용할 형식을 지정할 수 있습니다.

자세한 내용은...

도움말 색인의 "form files"를 참조하십시오.

코드 탐색기

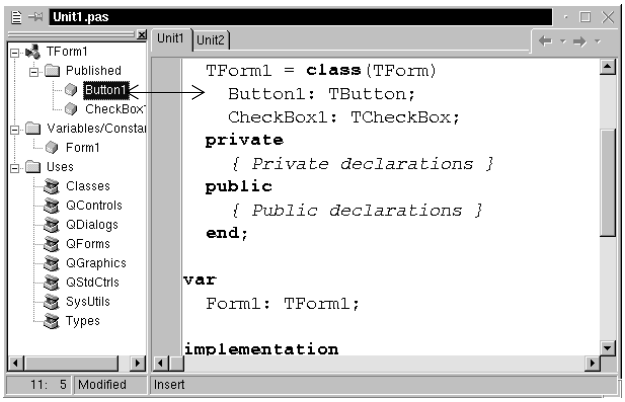
Kylix를 열면 사용자의 Kylix 에디션에서 코드 탐색기의 사용 가능 여부에 따라 코드 탐색기가 코드 에디터 창의 왼쪽에 도킹됩니다. 코드 탐색기는 유닛에 정의된 타입, 클래스, 속성, 메소드, 전역 변수 및 루틴을 트리 다이어그램으로 보여 줍니다. 또한 **uses** 절에 나열된 다른 유닛들도 보여 줍니다.

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

코드 탐색기에서 항목을 선택하면 커서는 코드 에디터의 해당 항목이 구현된 부분으로 이동합니다.

Ctrl+Shift+E를 누르면 코드 탐색기와 코드 에디터 사이의 마지막 위치로 커서가 앞과 뒤로 이동합니다.

클래스, 속성, 메소드, 변수 또는 루틴 등을 검색하려면 이름의 첫 문자만 입력합니다.



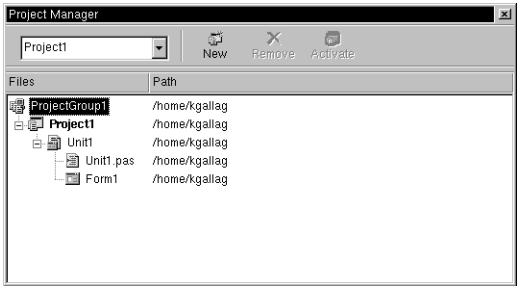
코드 탐색기에서 내용을 표시하는 방법을 구성하려면 Tools|Environment Options를 선택하고 Explorer 탭을 클릭합니다. 6-11페이지의 "코드 탐색기 사용자 지정"을 참조하십시오.

자세한 내용은...

도움말 색인의 "Code Explorer"를 참조하십시오.

Project Manager

Kylix를 처음 시작하면 2-1페이지의 "IDE"에 표시된 것처럼 새 프로젝트가 자동으로 열립니다. 프로젝트는 개발하려는 애플리케이션이나 공유 객체를 구성하는 파일을 여러 개 포함합니다. Project Manager라는 프로젝트 관리 도구에서 이러한 파일, 즉 폼, 유닛, 리소스, 객체, 라이브러리 파일을 보거나 구성할 수 있습니다. Project Manager를 표시하려면 View|Project Manager를 선택합니다.



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

자세한 내용은...

도움말 색인의 "Project Manager"를 참조하십시오.

Project Browser

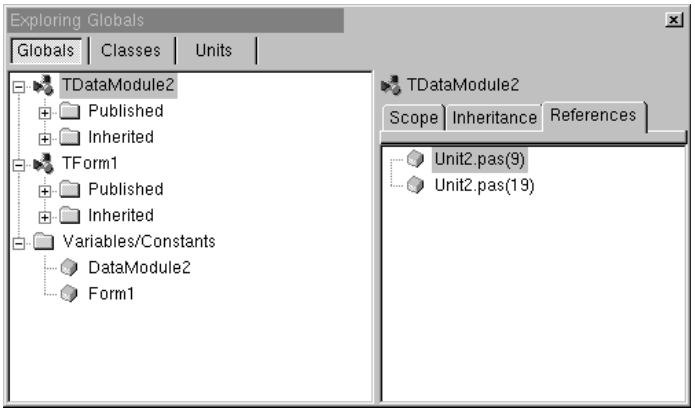
Project Browser는 프로젝트를 자세히 검사합니다. Browser는 프로젝트가 선언하거나 사용하는 클래스, 유닛 및 전역 기호(타입, 속성, 메소드, 변수, 루틴)를 트리 형태로 표시합니다. View|Browser를 선택하여 Project Browser를 표시합니다.

Project Browser에는 Inspector 창(왼쪽)과 Details 창이라는 크기를 조정할 수 있는 두 개의 창이 있습니다. Inspector 창에는 Globals, Classes, Units라는 3개의 탭이 있습니다.

Globals는 클래스, 타입, 속성, 메소드, 변수, 루틴 등을 표시합니다.

Classes는 클래스를 계층 다이어그램으로 표시합니다.

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



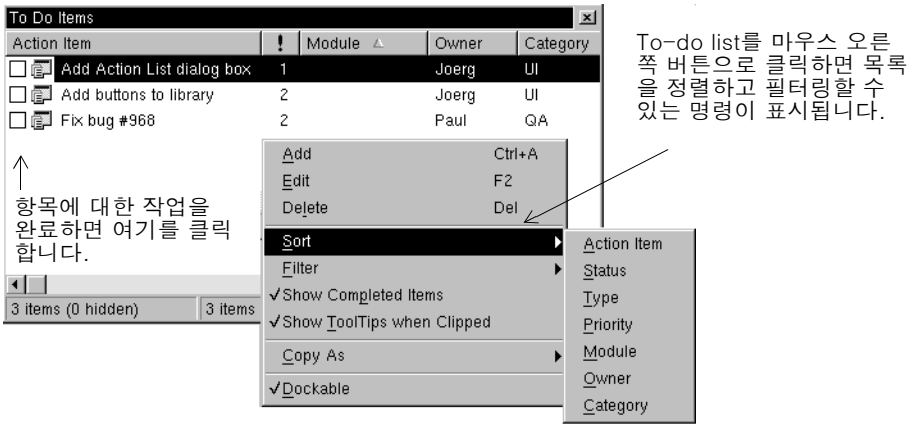
기본적으로 Project Browser는 현재 프로젝트에서만 사용되는 유닛 기호를 표시합니다. 유효 범위(scope)를 변경하면 Kylix에서 사용할 수 있는 모든 기호를 표시할 수 있습니다. Tools|Environment Options를 선택하고 Explorer 페이지에서 All symbols(CLX included)를 선택합니다.

자세한 내용은...

도움말 색인의 "Project Browser"를 참조하십시오.

To-Do List

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



자세한 내용은...

도움말 색인의 "To-do lists"를 참조하십시오.

3

Kylix 프로그래밍

이 장에서는 프로젝트 생성, 사용자 인터페이스 디자인, 코드 작성 및 프로그램 컴파일, 디버깅, 배포, 국제화를 비롯하여 Kylix를 사용한 소프트웨어 개발에 대한 개요를 다룹니다. 마지막 단원에는 Kylix에서 개발할 수 있는 여러 가지 유형의 프로젝트에 대해 소개합니다.

프로젝트 생성

프로젝트는 디자인 타임에 생성되거나 프로젝트 소스 코드를 컴파일할 때 생성된 파일 모음입니다. Kylix를 처음 시작하면 새 프로젝트가 열립니다. 여러 파일 중에서 프로젝트 파일(Project1.dpr), 유닛 파일(Unit1.pas), 리소스 파일(Unit1.xfm)은 자동으로 생성됩니다.

프로젝트가 이미 열려 있는 상태에서 새로운 프로젝트를 열려면 File|New Application 또는 File|New를 선택하고 Application 아이콘을 더블 클릭합니다. File|New를 누르면 대화 상자와 같이 미리 디자인된 템플릿뿐만 아니라 추가적인 폼, 프레임, 모듈을 제공하는 Object Repository가 열리고 이를 프로젝트에 추가할 수 있습니다. Object Repository에 대한 자세한 내용은 2-4페이지의 "Object Repository"를 참조하십시오.

프로젝트를 시작할 때에는 애플리케이션 또는 공유 객체와 같은 개발하고자 하는 대상에 대해 알아야 합니다. Kylix로 개발할 수 있는 프로젝트 유형에 대해 알아보려면 3-8페이지의 "프로젝트 유형"을 참조하십시오.

자세한 내용은...
도움말 색인의 "projects"를 참조하십시오.

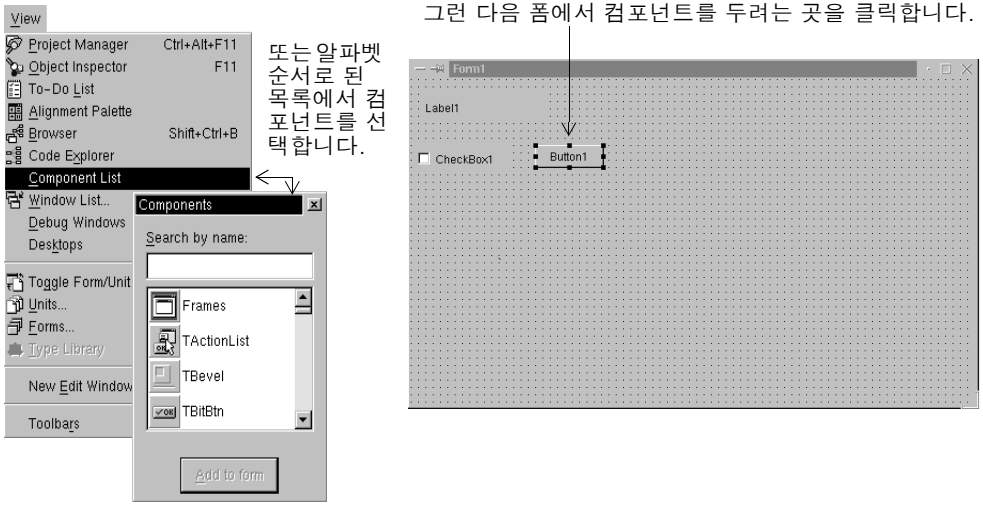
사용자 인터페이스 디자인

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

폼에 컴포넌트 두기

폼에 컴포넌트를 두려면 컴포넌트를 더블 클릭하거나 컴포넌트를 한 번 클릭한 다음 폼에서 컴포넌트가 나타나게 하려는 곳을 클릭합니다.

컴포넌트를 선택한 폼의 원하는 곳으로 끌어 놓습니다.



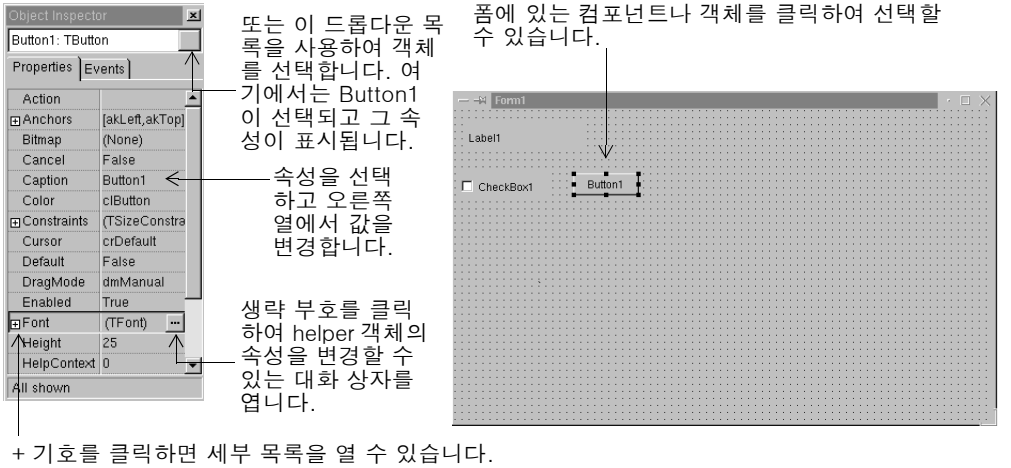
자세한 내용은...

도움말 색인의 "Component palette"를 참조하십시오.

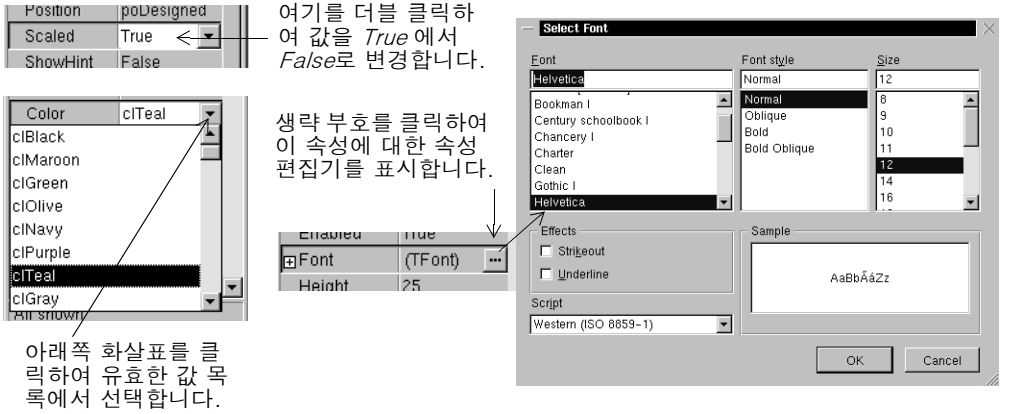
컴포넌트 속성 설정

컴포넌트를 폼에 둔 다음 컴포넌트의 속성을 설정하고 이벤트 핸들러를 코딩합니다. 컴포넌트 속성을 설정하면 애플리케이션에서 컴포넌트가 나타나고 동작하는 방식이 바뀝

니다. 폼에서 컴포넌트를 선택하면 선택한 컴포넌트의 속성과 이벤트가 Object Inspector에 표시됩니다.



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



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

자세한 내용은...

도움말 색인의 "Object Repository"를 참조하십시오.

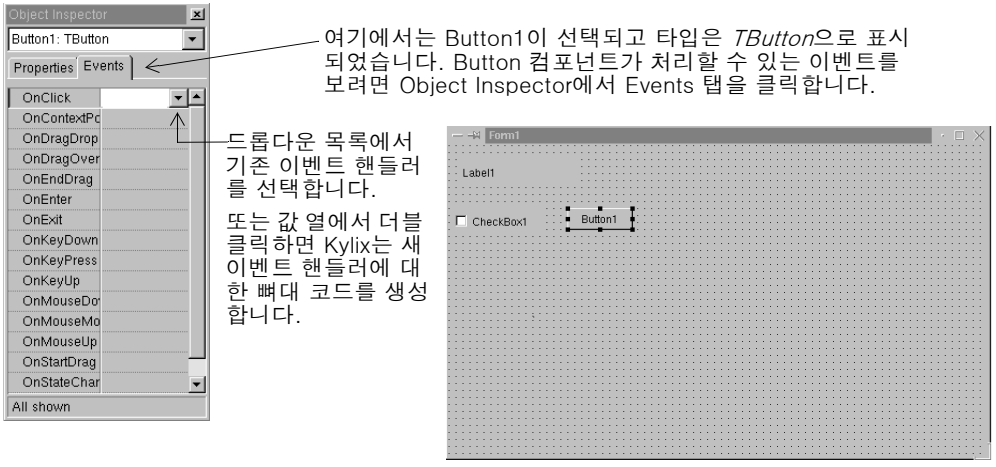
코드 작성

모든 애플리케이션의 핵심 부분은 각 컴포넌트의 코드 부분입니다. Kylix의 RAD 환경에서 미리 패키지된 비주얼 컴포넌트나 넌비주얼(nonvisual) 컴포넌트와 같은 빌딩 블록을 대부분 제공하지만 이벤트 핸들러와 일부 클래스는 개발자가 직접 만들어야 합니다. 개발자의 수고를 덜기 위해 Kylix의 CLX 클래스 라이브러리에서 약 750개의 객체를 제공합니다. 소스 코드를 보거나 편집하려면 2-5페이지의 "코드 에디터"를 참조하십시오.

이벤트 핸들러 작성

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

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



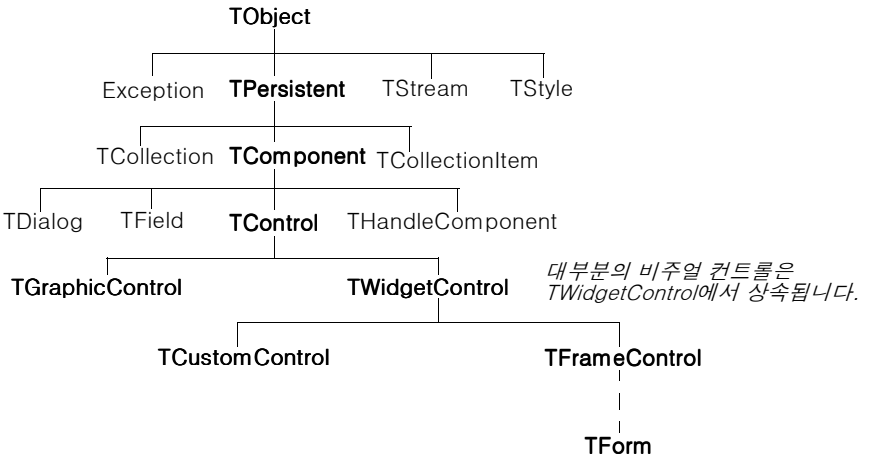
자세한 내용은...

도움말 색인의 "events"를 참조하십시오.

CLX 클래스 사용

Kylix에는 여러 개의 객체들로 구성된 클래스 라이브러리가 있는데 이 객체들 중에는 코드를 작성할 때 사용하는 컴포넌트나 컨트롤도 있습니다. Borland 크로스 플랫폼용 컴포넌트 라이브러리(CLX)라고 하는 클래스 계층에는 데이터셋과 타이머와 같은 넌비주얼 컨트롤뿐만 아니라 편집 컨트롤, 버튼 및 기타 사용자 인터페이스와 같은 런타임 시 보이는 객체도 포함되어 있습니다.

다음 다이어그램은 CLX를 구성하는 일부 주요 클래스를 보여 줍니다.



*TComponent*의 자손 객체에는 컴포넌트 팔레트에 설치할 수 있고 Kylix 폼에 추가할 수 있는 속성과 메소드가 있습니다. CLX 컴포넌트가 IDE로 혹되기 때문에 Form Designer와 같은 툴을 사용하여 애플리케이션을 신속하게 개발할 수 있습니다.

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

대부분의 Kylix 에디션에는 모든 CLX 소스 코드가 들어 있습니다. 온라인 설명서 외에 CLX 소스 코드는 오브젝트 파스칼 프로그래밍 기법에 유익한 예제들을 제공합니다.

자세한 내용은...

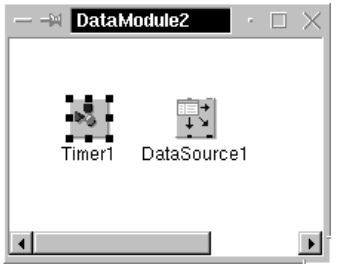
도움말 목차의 "Kylix Object and Component Reference"를 참조하십시오. CLX에 대한 오픈 소스와 라이센스 옵션에 대한 내용은 http://www.borland.com/kylix를 참조하십시오.

데이터 모듈 추가

데이터 모듈은 넌비주얼 컴포넌트만 포함하는 폼 형식입니다. 일반적인 폼에서는 넌비주얼 컴포넌트와 비주얼 컴포넌트를 함께 *둘 수 있습니다*. 데이터베이스와 시스템 객체 그룹을 재사용할 계획이 있거나 데이터베이스 연결과 비즈니스 룰을 처리하는 애플리케이션 부분을 분리하려는 경우라면 데이터 모듈을 간편한 구성 도구로 사용할 수 있습니다.

데이터 모듈을 만들려면 File|New를 선택하고 Object Repository에서 Data Module 아이콘을 더블 클릭합니다. Kylix는 코드 에디터에서 모듈에 대한 추가 유닛 파일을 표

시하는 빈 데이터 모듈을 열고 모듈을 현재 프로젝트에 새 유닛으로 추가합니다. 폼에서와 동일한 방식으로 넌비주얼(nonvisual) 컴포넌트를 데이터 모듈에 추가합니다.



컴포넌트 팔레트에서 넌비주얼 컴포넌트를 클릭하고 데이터 모듈에서 컴포넌트를 둘 위치를 클릭합니다.

기존 데이터 모듈을 다시 열면 Kylix는 추가된 컴포넌트를 표시합니다.

자세한 내용은...

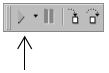
도움말 색인의 "data modules"를 참조하십시오.

프로젝트 컴파일 및 디버깅

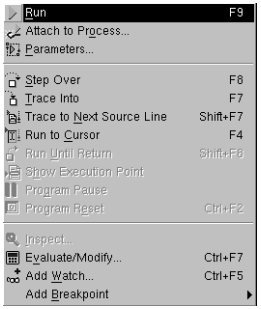
코드를 작성하고 나면 프로젝트를 컴파일하고 디버깅해야 합니다. Kylix를 사용하면 먼저 프로젝트를 컴파일한 다음 따로 디버깅하거나 통합 디버거를 사용하여 컴파일과 디버깅을 동시에 수행할 수 있습니다. 디버깅 정보로 프로그램을 컴파일하려면 Project|Options를 선택하고 Compiler 탭을 클릭한 다음 Debug informations가 선택되어 있는지 확인합니다.

Kylix는 통합 디버거를 사용하므로 프로그램 실행을 제어하고, 변수를 관찰하고, 데이터 값을 수정할 수 있습니다. 코드를 한 줄씩 작성해 나가면서 각 브레이크 포인트에서 프로그램 상태를 확인할 수 있습니다. 통합 디버거를 사용하려면 Tools|Debugger Options를 선택하고 General 탭을 클릭한 다음 Integrated debugging이 선택되어 있는지 확인합니다.

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



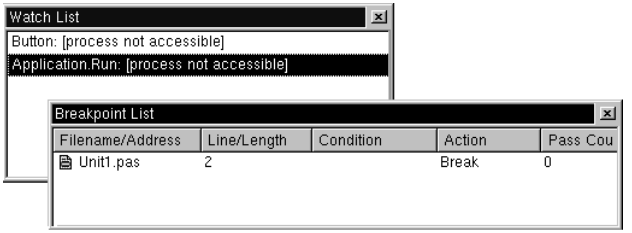
Run 버튼



Run 메뉴에서 디버깅 명령을 선택합니다. 일부 명령은 툴바에도 있습니다.

통합 디버거를 사용하면 Breakpoints, Call Stack, Watches, Local Variables, Threads, Modules, CPU, Event Log 등을 비롯한 여러 디버깅 창을 사용할 수 있습니

다. View|Debug Windows를 선택하여 디버깅 창을 표시합니다. 몇몇 디버거 뷰는 일부 Kylix 에디션에서 사용할 수 없습니다.



디버깅 창을 결합하여 디버깅을 더 쉽게 하는 방법은 6-2페이지의 "툴 윈도우 도킹"을 참조하십시오.

일단 디버깅용으로 데스크탑을 설정했으면 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...

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

애플리케이션 배포

다른 사람들이 설치하고 실행할 수 있도록 애플리케이션을 배포할 수 있습니다. 애플리케이션을 배포하려면 실행 파일과 같은 필수 파일뿐만 아니라 공유 객체 파일, 초기화 파일, 패키지 파일, helper 애플리케이션과 같은 모든 지원 파일을 포함하는 설치 패키지를 만들어야 합니다.

자세한 내용은...

도움말 색인의 "deploying"을 참조하십시오.

애플리케이션 국제화

Kylix는 로케일이 다른 애플리케이션의 국제화 및 지역화를 위한 여러 가지 기능을 제공합니다. IDE와 CLX는 IME와 확장 문자 집합을 지원합니다. 애플리케이션을 일단 국제화하면 배포하려는 다른 나라를 위한 지역화된 버전을 만들 수 있습니다.

Kylix는 애플리케이션에서 Borland 리소스를 추출한 다음 이 리소스를 포함하는 공유 객체 파일을 만드는 *resbind*라는 툴을 제공합니다. 그런 다음 런타임에 리소스를 동적으로 연결하거나 애플리케이션이 실행 중인 로컬 시스템에서 애플리케이션이 환경 변수를 확인하도록 할 수 있습니다. 이러한 기능들을 최대한 활용하려면 가능한 한 개발 과정 초기에 지역화 요구 사항을 고려하십시오.

자세한 내용은...

도움말 색인의 "international applications"를 참조하십시오.

프로젝트 유형

Kylix의 모든 에디션은 다양한 GUI 애플리케이션, 공유 객체, 패키지 및 기타 프로그램을 작성하는 범용 Linux 프로그래밍을 지원합니다. 일부 에디션은 분산 애플리케이션, 데이터베이스 애플리케이션, 웹 애플리케이션과 같은 서버 애플리케이션을 지원합니다. 사용 중인 에디션에서 지원하는 도구를 보려면 http://www.borland.com/kylix에서 Feature Matrix를 참조하십시오.

자세한 내용은...

*Kylix1 개발자 안내서*의 5장 "애플리케이션과 공유 객체 구축"을 참조하십시오.

데이터베이스 애플리케이션

Kylix는 데이터베이스 애플리케이션에 사용하기 위해 새로운 데이터 액세스 기술인 *dbExpress*를 사용합니다. dbExpress는 애플리케이션이 데이터베이스에 있는 데이터에 액세스하는 데 사용하는 드라이버 컬렉션입니다. Kylix는 DB2, Informix, InterBase, MySQL 및 Oracle과 같은 SQL 데이터베이스용 드라이버를 여러 개 가지고 있습니다.

데이터에 액세스하기 위해 dbExpress 컴포넌트를 데이터 모듈이나 폼에 추가할 수 있습니다. 이러한 컴포넌트에는 데이터에 연결하는 데 필요한 정보를 제어하는 연결 컴포넌트와 서버에서 fetch한 데이터를 나타내는 데이터셋 컴포넌트가 있습니다. dbExpress와 데이터베이스 컴포넌트를 사용하려면 컴포넌트 팔레트의 dbExpress 페이지와 Data Access 페이지를 클릭합니다. 몇몇 데이터베이스 연결 및 애플리케이션 툴은 일부 Kylix 에디션에서 사용할 수 없습니다.

자세한 내용은...

*Kylix1 개발자 안내서*의 2부 "데이터베이스 애플리케이션 개발"과 도움말 색인의 "database applications"를 참조하십시오.

클라이언트/서버 애플리케이션

다계층 클라이언트/서버 데이터베이스 애플리케이션을 구축할 때 DataSnap을 사용할 수 있습니다. DataSnap은 data-aware 애플리케이션 서버와 클라이언트 애플리케이션이 서로 통신하기 위한 프로토콜과 컴포넌트를 정의합니다. 애플리케이션 서버에 대해 데이터셋과 데이터셋 프로바이더를 SOAP 데이터 모듈에 추가합니다. SOAP 데이터 모듈은 웹 서비스 애플리케이션에서 웹 모듈로 작동하여 클라이언트 애플리케이션과 프로바이더 간에 메시지를 디스패치합니다.

애플리케이션 서버를 구축하려면 File|New를 선택한 다음 New Items 대화 상자에서 WebServices 페이지를 클릭합니다. SOAP Services Data Module 아이콘을 더블 클릭합니다. 서버와 연결을 설정하려면 컴포넌트 팔레트에서 WebServices 페이지를 클릭하고 *SoapConnection* 컴포넌트를 선택합니다.

자세한 내용은...

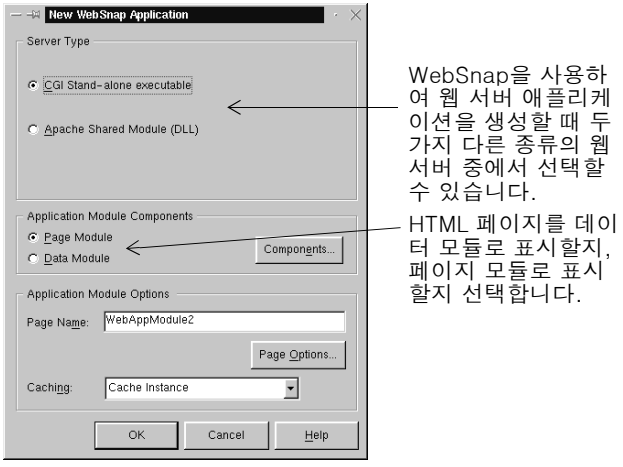
*Kylix1 개발자 안내서*의 14장 "다계층 아키텍처 사용" 단원을 참조하십시오.

웹 서버 애플리케이션

웹 서버 애플리케이션은 기존 웹 서버의 기능을 확장합니다. 웹 서버 애플리케이션은 웹 서버로부터 HTTP 요청 메시지를 받아 이 메시지에서 요청한 모든 작업을 수행하고 웹 서버로 다시 보낼 응답을 작성합니다. Kylix 애플리케이션으로 수행할 수 있는 여러 가지 작업을 하나의 웹 서버 애플리케이션으로 통합할 수 있습니다.

Kylix는 Kylix 에디션별로 두 가지 다른 기술을 포함합니다. WebBroker가 포함된 웹 서버 애플리케이션을 만들려면 File|New를 선택하고 Web Server Application 아이콘을 더블 클릭합니다. Internet 컴포넌트 팔레트 페이지의 웹 모듈에 WebBroker 컴포넌트를 추가할 수 있습니다.

WebSnap으로 웹 서버 애플리케이션을 개발하려면 File|New를 선택하고 WebSnap 페이지를 클릭한 다음 Web Server Application 아이콘을 더블 클릭합니다. WebSnap 컴포넌트 팔레트 페이지에서 WebSnap 컴포넌트를 추가할 수 있습니다. WebSnap은 어댑터, 추가 디스패처, 추가 페이지 프로듀서, 세션 지원 및 웹 페이지 모듈이 포함된 WebBroker 기능에 추가됩니다.



자세한 내용은...

*Kylix2 개발자 안내서 Supplement(보충판)*의 3부 "인터넷 애플리케이션 개발"과 도움말 색인의 "Web server applications"를 참조하십시오.

웹 서비스

재고 추적 프로그램과 같은 웹 서비스는 인터넷으로 액세스할 수 있는 애플리케이션입니다. 웹 서비스는 제공하는 서비스에 대한 인터페이스가 잘 정의되어 있습니다. 서버를 구현하면 클라이언트에서 하나 이상의 통신 방식과 인코딩 스키마를 사용할 수 있습니다. Kylix의 BizSnap 컴포넌트는 HTTP와 XML 방식의 SOAP(Simple Object Access

Protocol) 프로토콜을 사용하도록 디자인되었습니다. 이 컴포넌트들을 사용하여 웹 서비스를 구현하는 서버와 서비스를 호출하는 클라이언트를 모두 작성할 수 있습니다.

마법사를 사용하여 서버를 구축하려면 File|New를 선택하고 New Items 대화 상자에서 WebServices 페이지를 클릭한 다음 SOAP Server Application 아이콘을 더블 클릭합니다. 클라이언트에 대한 컴포넌트에 액세스하려면 컴포넌트 팔레트의 WebServices 페이지를 클릭합니다.

자세한 내용은...

*Kylix2 개발자 안내서 Supplement(보충판)*의 10장 "웹 서비스 사용"을 참조하십시오.

공유 객체

공유 객체는 애플리케이션과 다른 공유 객체에서 호출할 수 있는 루틴을 포함하고 있는 컴파일된 모듈입니다. 공유 객체에는 두 개 이상의 애플리케이션에서 사용하는 코드 또는 리소스가 포함되어 있습니다.

자세한 내용은...

도움말 색인의 "shared objects"를 참조하십시오.

사용자 지정 컴포넌트

Kylix에 있는 컴포넌트들은 컴포넌트 팔레트에 미리 설치되어 있으며 개발에 필요한 대부분의 기능을 제공합니다. 새 컴포넌트를 설치하지 않고도 Kylix로 프로그래밍할 수는 있지만 간혹 특별한 문제를 해결하거나 특정한 동작을 나타내기 위해 사용자 지정 컴포넌트가 필요한 경우도 있을 것입니다. 사용자 지정 컴포넌트는 애플리케이션 간에 코드를 재사용하거나 일관성을 유지하는 데 유용합니다.

협력 업체의 사용자 지정 컴포넌트를 설치하거나 사용자 지정 컴포넌트를 직접 만들 수 있습니다. 새 컴포넌트를 만들려면 Component|New Component를 선택하여 New Component 마법사를 표시합니다. 협력 업체에서 제공하는 컴포넌트를 설치하려면 6-7페이지의 "컴포넌트 패키지 설치"를 참조하십시오.

자세한 내용은...

*Kylix1 개발자 안내서*의 4부 "사용자 지정 컴포넌트 생성"과 도움말 색인의 "components, creating"을 참조하십시오.

4

텍스트 에디터(자습서) 만들기

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

참고 이 자습서는 Kylix의 모든 에디션에서 사용할 수 있습니다.

새 애플리케이션 시작

새 애플리케이션을 시작하기 전에 소스 파일을 저장할 디렉토리를 만듭니다.

1 홈 디렉토리에 TextEditor라는 디렉토리를 만듭니다.

2 File|New Application을 선택하거나 Kylix 시작 시 열린 기본 프로젝트를 사용하여 새 프로젝트를 시작합니다.

각 애플리케이션을 *프로젝트*라고 합니다. Kylix를 시작하면 Kylix는 기본적으로 빈 프로젝트를 만들고 다음과 같은 파일을 자동으로 생성합니다.

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

폼마다 자체 유닛 파일(*Unit1.pas*)과 폼 파일(*Unit1.xfm*)이 있습니다. 두 번째 폼을 만들면 두 번째 유닛 파일(*Unit2.pas*)과 폼 파일(*Unit2.xfm*) 이 만들어집니다.

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

- TextEditor 폴더를 탐색합니다.
- 기본 이름 Unit1.pas로 Unit1을 저장합니다.

- TextEditor.dpr이라는 이름으로 프로젝트를 저장합니다.(실행 파일은 확장자를 뺀 프로젝트 이름과 동일한 이름으로 지정됩니다.)

나중에 File|Save All을 선택하여 작업 내용을 다시 저장합니다.

프로젝트를 저장하면 Kylix는 프로젝트 디렉토리에 추가 파일을 만듭니다. 이러한 추가 파일에는 TextEditor.kof라는 Kylix Options 파일, TextEditor.conf라는 구성 파일, TextEditor.res라는 리소스 파일이 있습니다. 이러한 파일들에 대해 신경 쓸 필요는 없지만 삭제하면 안 됩니다.

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

F9 키를 누르면 언제든지 폼을 실행할 수 있습니다.

폼에 컴포넌트가 없어도 폼의 런타임 뷰는 최소화, 최대화, 닫기 버튼 및 컨트롤 메뉴 등을 완비한 디자인 타임 뷰처럼 보입니다.

폼에 컴포넌트가 없더라도 *F9* 키를 눌러 지금 폼을 실행해 보십시오.

Form1의 디자인 타임 뷰로 돌아가려면 다음 중 하나를 수행합니다.

- 폼의 런타임 뷰에서 애플리케이션의 제목 표시줄의 오른쪽 위 모서리에 있는 **X**를 클릭합니다.
- 제목 표시줄의 왼쪽 위 모서리에 있는 애플리케이션 종료 버튼 을 클릭합니다.
- Run|Program Reset을 선택하거나
- View|Forms를 누르고 Form1을 선택한 다음 OK를 클릭합니다.

속성 값 설정

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

지금 즉시 *Form1*의 캡션을 변경할 수 있습니다.

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

폼에 컴포넌트 추가

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

Kylix에는 애플리케이션의 부분을 나타내는 컴포넌트들이 많이 있습니다. 예를 들면 컴포넌트 팔레트에는 메뉴, 툴바, 대화 상자 및 그 밖의 비주얼(visual) 및 넌비주얼(nonvisual) 프로그램 요소들을 쉽게 프로그래밍할 수 있는 컴포넌트들(*객체*라고도 함)이 있습니다.

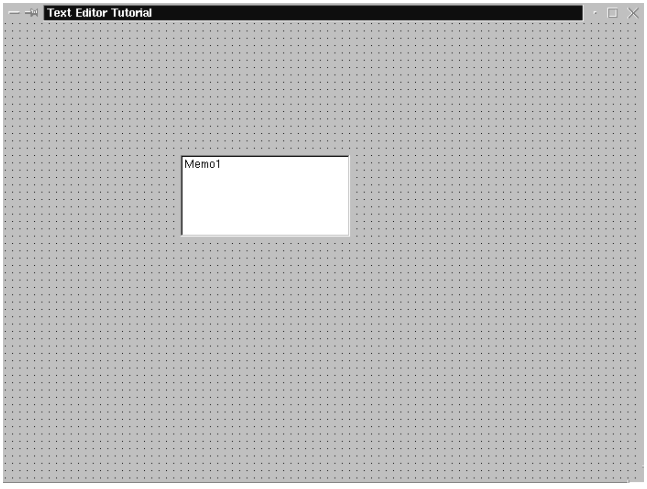
텍스트 에디터 애플리케이션에는 편집 영역, 편집 중인 파일 이름과 같은 정보를 표시하는 상태 표시줄, 메뉴, 명령에 쉽게 액세스하는 아이콘이 있는 툴바 등이 필요합니다. Kylix를 사용한 인터페이스 디자인의 장점은 다른 여러 가지 컴포넌트를 시도해 볼 수 있고 그 결과를 즉시 확인해 볼 수 있다는 것입니다. 이렇게 하면 애플리케이션 인터페이스의 프로토타입을 신속하게 만들 수 있습니다.

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

1 텍스트 영역을 만들려면 먼저 *Memo* 컴포넌트를 추가합니다. *Memo* 컴포넌트를 찾으려면 컴포넌트 팔레트의 Standard 탭에서 팔레트의 아이콘을 잠시 가리킵니다. 그러면 Kylix는 컴포넌트의 이름을 보여 주는 도움말 툴팁을 표시합니다.

Memo 컴포넌트를 찾았으면 다음 중 하나를 수행합니다.

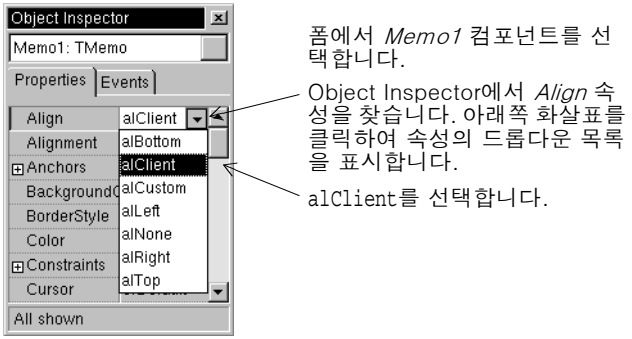
- 팔레트에서 컴포넌트를 선택하고 컴포넌트를 두려는 위치에서 폼을 클릭합니다.
- 더블 클릭하여 컴포넌트를 폼의 가운데에 둡니다.



각 Kylix 컴포넌트는 *클래스*이며 폼에 컴포넌트를 두는 것은 클래스의 *인스턴스*를 만드는 것입니다. 일단 컴포넌트가 폼에 있으면 Kylix는 애플리케이션이 실행 중일 때 객체의 인스턴스를 만드는 데 필요한 코드를 생성합니다.

2 *Memo1*의 *Align* 속성을 *alClient*로 설정합니다.

폼에서 *Memo1*을 클릭하여 선택한 다음 Object Inspector에서 *Align* 속성을 선택하면 됩니다. 드롭다운 목록에서 *alClient*를 선택합니다.



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

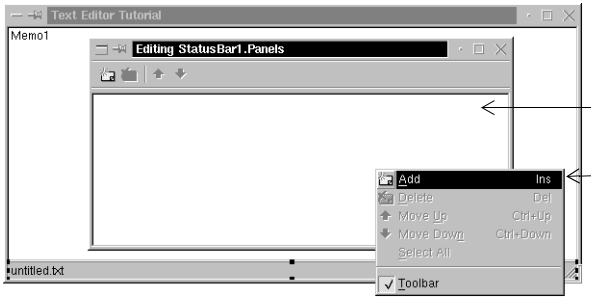
3 컴포넌트 팔레트의 Common Controls 탭에서 *StatusBar* 컴포넌트를 더블 클릭합니다. 이렇게 하면 폼의 하단에 상태 표시줄이 추가됩니다.

4 그런 다음 상태 표시줄에 공간을 만들어 텍스트 에디터로 편집 중인 파일의 이름과 경로를 표시할 수 있습니다. 가장 간단한 방법은 상태 패널을 하나만 두는 것입니다.

- *SimpleText* 속성을 `untitled.txt`로 바꿉니다. 편집 중인 파일을 아직 저장하지 않았다면 파일 이름은 untitled.txt가 됩니다.
- *Simple Panel*을 `True`로 설정합니다.

- Editing StatusBar1.Panels 대화 상자를 열려면 *Panels* 속성의 생략 버튼을 클릭합니다. 대화 상자를 늘려서 크게 만들 수도 있습니다.

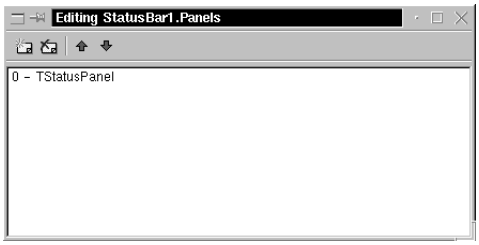
- 대화 상자를 마우스 오른쪽 버튼으로 클릭하고 Add를 클릭하여 상태 표시줄에 패널을 추가합니다.



마우스 오른쪽 버튼으로 여기를 클릭하면 컨텍스트 메뉴가 표시됩니다. 영구적인 텍스트를 유지할 수 있는 상태 표시줄에 패널을 만들려면 Add를 클릭합니다.

Panels 속성은 인덱스가 0부터 시작하는 배열(zero-based array)이므로 고유한 인덱스 값을 기반으로 만든 각 패널에 액세스할 수 있습니다. 기본적으로 첫 번째 패널은 0 값을 가집니다.

Add를 클릭할 때마다 상태 표시줄에 패널이 추가됩니다.



팁 상태 표시줄을 더블 클릭하여 Editing StatusBar1.Panels 대화 상자에 액세스해도 됩니다.

5 오른쪽 위 모서리에 있는 **X**를 클릭하여 Editing StatusBar1.Panels 대화 상자를 닫습니다.

이제 텍스트 에디터에 대한 사용자 인터페이스의 메인 편집 영역이 설정되었습니다.

메뉴와 툴바에 대한 지원 추가

애플리케이션으로 어떤 작업을 하려면 메뉴, 명령 및 편리하게 사용하기 위한 툴바가 필요합니다. 명령을 따로 코딩할 수 있지만 Kylix는 *액션 리스트(action list)*를 제공하여 코드를 한 가지로 통일하도록 지원합니다.

다음은 예제 텍스트 에디터 애플리케이션에 필요한 액션의 종류입니다.

표 4.1 Text Editor 명령

명령	메뉴	툴바에 포함	설명
New	File	예	새 파일을 만듭니다.
Open	File	예	기존 파일을 엽니다.
Save	File	예	현재 파일을 디스크에 저장합니다.
Save As	File	아니오	파일을 새 이름으로 저장하거나 새 파일을 지정된 이름으로 저장합니다.
Exit	File	예	에디터 프로그램을 종료합니다.
Cut	Edit	예	텍스트를 삭제하고 클립보드에 저장합니다.
Copy	Edit	예	텍스트를 복사하고 클립보드에 저장합니다.
Paste	Edit	예	클립보드에 있는 텍스트를 삽입합니다.
About	Help	아니오	애플리케이션에 대한 정보를 표시합니다.

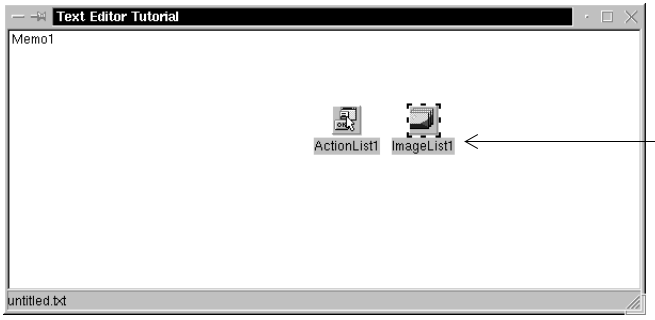
또한 이미지 목록의 툴바와 메뉴에 사용할 이미지를 중앙에 모을 수 있습니다.

다음과 같은 방법으로 *ActionList* 컴포넌트와 *ImageList* 컴포넌트를 폼에 추가합니다.

1 컴포넌트 팔레트의 Standard 탭에서 *ActionList* 컴포넌트를 더블 클릭해서 폼에 가져다 놓습니다.

2 Common Controls에서 *ImageList* 컴포넌트를 더블 클릭하여 폼에 가져다 놓습니다. 그러면 *ActionList* 컴포넌트 위쪽에 놓이므로 폼의 다른 위치로 가져다 놓습니다. *ActionList* 컴포넌트와 *ImageList* 컴포넌트는 모두 넌비주얼 컴포넌트이므로 폼의 어디에 두든지 상관없습니다. 넌비주얼 컴포넌트는 런타임 시 나타나지 않습니다.

이제 폼은 다음 그림과 비슷하게 보일 것입니다.



폼에 놓은 컴포넌트의 캡션을 표시하려면 Tools|Environment Options를 선택하고 Show component captions를 클릭합니다.

*ActionList*와 *ImageList* 컴포넌트는 넌비주얼 컴포넌트이므로 애플리케이션 실행 시 보이지 않습니다.

액션 리스트에 액션 추가

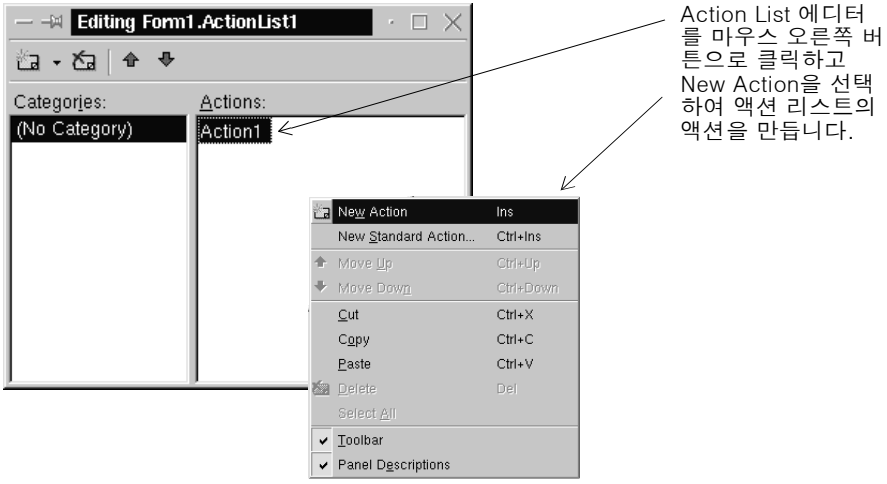
다음에서 액션 리스트에 액션을 추가합니다.

팁 규칙에 따라 메뉴 항목에 연결된 액션 이름을 상위 레벨 메뉴 이름과 항목 이름으로 지정합니다. 예를 들면 FileExit 액션은 File 메뉴의 Exit 명령을 의미합니다.

1 ActionList 아이콘을 더블 클릭합니다.

Editing Form1.ActionList1 대화 상자가 나타납니다. 이것을 Action List 에디터라고도 합니다.

2 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택합니다.



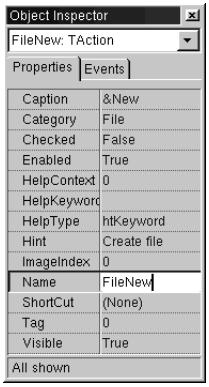
3 Object Inspector에서 다음과 같은 액션 속성들을 설정합니다.

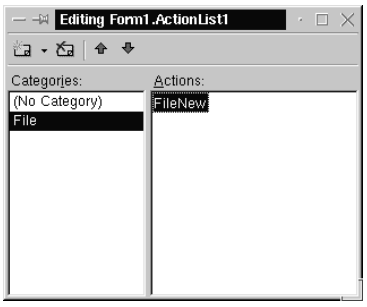
- *Caption*에 `&New`를 입력합니다. 문자 앞에 앰퍼샌드(&)를 입력하면 명령에 액세스하는 단축키가 만들어집니다.
- *Category*에 `File`을 입력하면 File 명령이 만들어집니다.
- *Hint*에 `Create file`을 입력하면 도움말 툴팁이 됩니다.
- *ImageIndex*에 `0`을 입력하면 ImageList에 있는 이미지 번호 0을 이 액션에 연결시킵니다.

- *Name*에 FileNew를 입력하고(File|New 명령) *Enter*를 눌러 변경 사항을 저장합니다.

Action List 에디터에서 선택한 새 액션을 사용하여 Object Inspector에 있는 속성을 변경합니다.

*Caption*은 메뉴에서 사용되고, *Category*는 액션의 종류이며, *Hint*는 도움말 툴팁이고, *ImageIndex* 를 사용하면 ImageList에 있는 그래픽을 참조할 수 있으며, *Name*은 코드에서 호출하는 액션 이름입니다.





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

5 Object Inspector에서 다음 속성을 설정합니다.

- *Caption*에 &Open을 입력합니다.
- *Category*가 File을 표시하는지 확인합니다.
- *Hint*에 Open file을 입력합니다.
- *ImageIndex*에 1을 입력합니다.
- *Name*에 FileOpen을 입력합니다(File|Open 명령).

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

7 Object Inspector에서 다음 속성을 설정합니다.

- *Caption*에 &Save를 입력합니다.
- *Category*가 File을 표시하는지 확인합니다.
- *Hint*에 Save file을 입력합니다.
- *ImageIndex*에 2를 입력합니다.
- *Name*에 FileSave를 입력합니다(File|Save 명령).

8 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택합니다.

9 Object Inspector에서 다음 속성을 설정합니다.

- *Caption*에 Save &As를 입력합니다.
- *Category*가 File을 표시하는지 확인합니다.
- *Hint*에 Save file as를 입력합니다.
- *ImageIndex*는 필요 없습니다. 기본값을 그대로 둡니다.
- *Name*에 FileSaveAs를 입력합니다(File|Save As 명령).

10 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택합니다.

11 Object Inspector에서 다음 속성을 설정합니다.

- *Caption*에 E&xit를 입력합니다.
- *Category*가 File을 표시하는지 확인합니다.
- *Hint*에 Exit application을 입력합니다.
- *ImageIndex*에 3을 입력합니다.
- *Name*에 FileExit를 입력합니다(File|Exit 명령).

12 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택하여 Help|About 명령을 만듭니다.

13 Object Inspector에서 다음 속성을 설정합니다.

- *Caption*에 `&About`를 입력합니다.
- *Category*에 `Help`를 입력합니다.
- *ImageIndex*는 필요 없습니다. 기본값을 그대로 둡니다.
- *Name*에 `HelpAbout`를 입력합니다(Help|About 명령).

화면에서 Action List 에디터를 닫지 않고 열어 둡니다.

액션 리스트에 표준 액션 추가

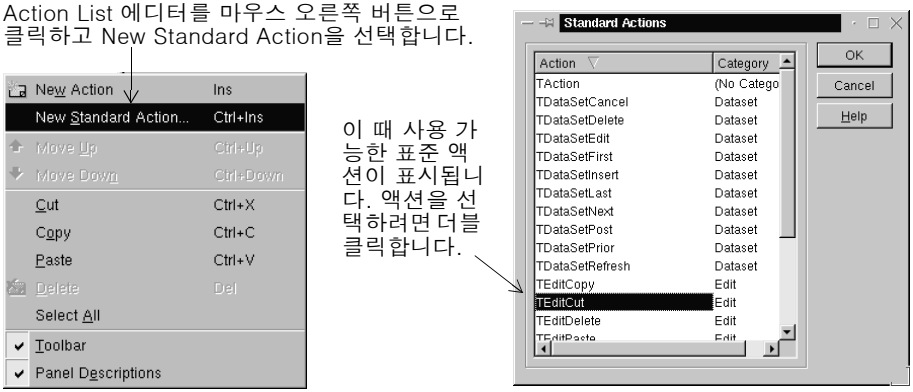
Kylix는 애플리케이션 개발 시 많이 사용되는 여러 개의 표준 액션을 제공합니다. 다음에서 액션 리스트에 표준 액션(잘라내기, 복사, 붙여넣기)을 추가합니다.

참고 Action List 에디터는 계속 열어 두어야 합니다. 에디터를 닫았을 경우 폼에서 ActionList 아이콘을 더블 클릭합니다.

다음과 같은 방법으로 액션 리스트에 표준 액션을 추가합니다.

1 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 선택합니다.

Standard Actions 대화 상자가 나타납니다.



- TEditCut을 더블 클릭합니다. 이 액션은 Edit라는 새 범주와 함께 Editing Form1.ActionList1 대화 상자에서 만들어집니다. EditCut1을 선택합니다.
- Object Inspector에서 *ImageIndex* 속성을 4로 설정합니다.

다른 속성들은 자동으로 설정됩니다.

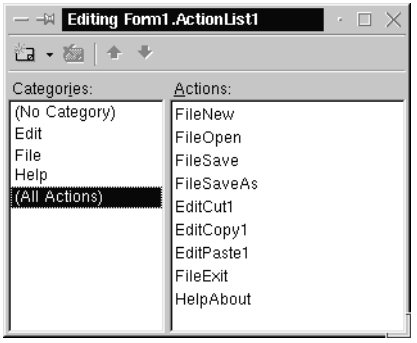
2 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 선택합니다.

- TeditCopy를 더블 클릭합니다.
- Object Inspector에서 *ImageIndex* 속성을 5로 설정합니다.

3 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 선택합니다.

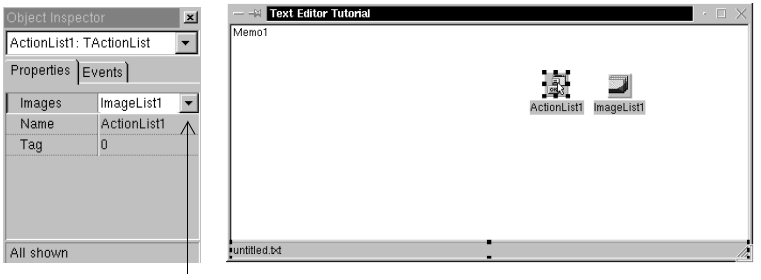
- TEditPaste를 더블 클릭합니다.
- Object Inspector에서 *ImageIndex* 속성을 6으로 설정합니다.

4 이제 메뉴와 툴바에서 필요한 모든 액션을 지정했습니다. All Actions 범주를 클릭하면 다음과 같이 목록에 있는 모든 액션을 볼 수 있습니다.



5 **X**를 클릭하여 Action List 에디터를 닫습니다.

6 폼에서 계속 선택된 상태인 *ActionList* 컴포넌트의 *Images* 속성을 ImageList1로 설정합니다.



Images 속성 옆에 있는 아래쪽 화살표를 클릭합니다. ImageList1을 선택합니다. 이렇게 하면 이미지 목록에 추가할 이미지가 액션 리스트에 있는 액션과 연결됩니다.

이미지 목록에 이미지 추가

앞에서는 ImageList 객체를 폼에 추가했습니다. 이 단원에서 툴바 및 메뉴에서 사용하기 위해 이미지를 목록에 추가합니다. 다음은 각 명령에 사용할 이미지입니다.

명령	아이콘 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

다음과 같은 방법으로 이미지를 이미지 목록에 추가합니다.

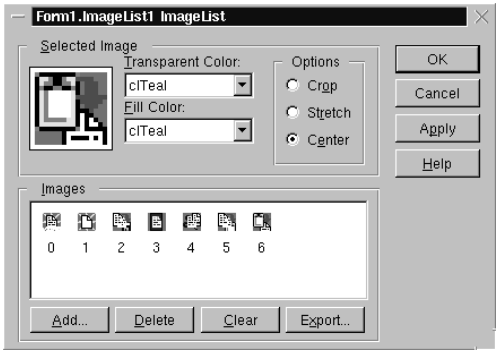
1 폼에서 ImageList 객체를 더블 클릭하여 Image List 에디터를 표시합니다.

2 Add 버튼을 클릭하고 제품이 설치되면서 생성된 Buttons 디렉토리를 탐색합니다. 기본 위치는 {install directory}/images/buttons입니다. 예를 들어 Kylix가 /usr/local/kylix2 디렉토리에 설치되었으면 /usr/local/kylix2/images/buttons를 찾아봅니다.

3 fileopen.bmp를 더블 클릭합니다.

4 메시지에서 비트맵을 두 개의 비트맵으로 분류할 것인지 물을 때마다 Yes를 클릭합니다. 각 아이콘은 이미지의 활성 버전과 비활성 버전을 포함합니다. 두 개의 이미지를 모두 볼 수 있습니다. 비활성(두 번째) 이미지를 삭제합니다.

   - Add를 클릭하고 filenew.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.

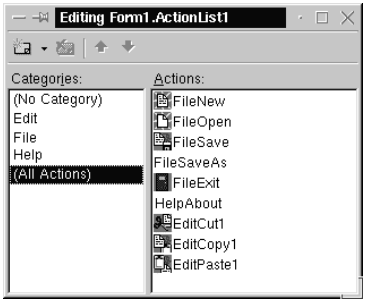


5 Image List 에디터를 닫으려면 OK를 클릭합니다.

이제 이미지 목록에 7 개의 이미지를 추가했고 추가한 이미지에는 각 액션의 ImageIndex 번호와 일치하도록 0에서 6까지 번호를 지정했습니다.

참고 순서가 맞지 않으면 Image List 에디터에서 올바른 위치로 드래그 앤 드롭합니다.

6 액션 리스트와 연결된 아이콘을 보려면 ActionList 객체를 더블 클릭한 다음 All Actions 범주를 선택합니다.



지금 Action List 에디터를 표시하면 이 액션과 연결된 아이콘을 볼 수 있습니다.

툴바에 아이콘을 만들지 않을 것이므로 명령에 대한 아이콘을 선택하지 않았습니다.

모두 끝났으면 Action List 에디터를 닫으십시오. 이제 메뉴와 툴바를 추가할 준비가 되었습니다.

메뉴 추가

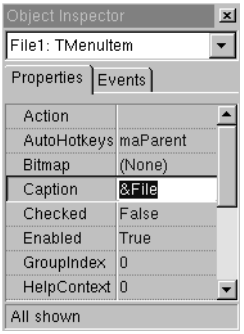
이 단원에서는 세 가지 드롭다운 메뉴(File, Edit, Help)가 있는 메인 메뉴 표시줄을 추가하고 액션 리스트의 액션을 사용하여 메뉴 항목을 각 메뉴에 추가할 수 있습니다.

1 컴포넌트 팔레트의 Standard 탭에서 *MainMenu* 컴포넌트를 폼에 가져다 놓습니다. 아무 곳에나 둘 수 있습니다.

2 메인 메뉴의 *Images* 속성을 `ImageList1`로 설정하면 비트맵을 메뉴 항목에 추가할 수 있습니다.

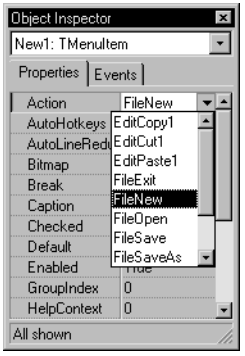
3 메인 메뉴 컴포넌트를 더블 클릭하여 메뉴 디자이너를 표시합니다.

4 Object Inspector에서 *Caption*에 `&File`을 입력하고 *Enter*를 눌러 첫 번째 상위 레벨 메뉴 항목을 설정합니다.

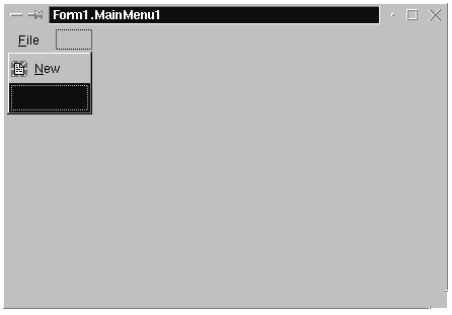


*&File*을 입력하고 Menu Designer에 포커스를 맞추면 상위 레벨 File 명령이 나타나 첫 번째 메뉴 항목을 추가할 수 있습니다.

5 메뉴 디자이너에서 방금 만든 File 항목을 선택합니다. File 항목 아래에 있는 빈 항목을 선택합니다. Object Inspector에서 *Action* 속성을 선택합니다. 액션 리스트의 액션이 모두 나열됩니다. `FileNew`를 선택합니다.



Action 속성 목록에서 `FileNew`를 선택하면 해당 Caption 및 ImageIndex와 함께 New 명령이 나타납니다.

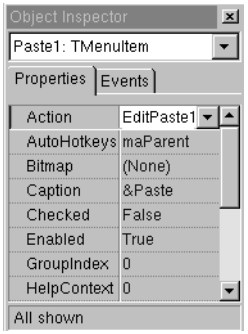


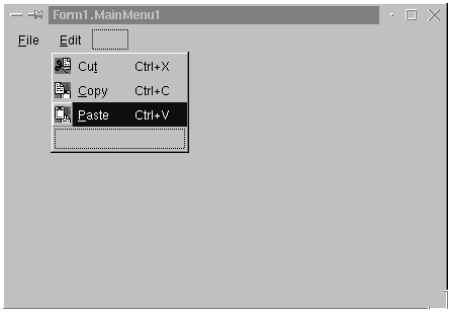
- New 아래에 있는 항목을 선택하고 *Action* 속성을 `FileOpen`으로 설정합니다.
- Open 아래에 있는 항목을 선택하고 *Action* 속성을 `FileSave`로 설정합니다.
- Save 아래에 있는 항목을 선택하고 *Action* 속성을 `FileSaveAs`로 설정합니다.
- Save As 아래에 있는 항목을 선택하고 *Caption* 속성 뒤에 하이픈을 입력하고 *Enter*를 누릅니다. 그러면 메뉴에 구분자 표시줄이 나타납니다.
- 구분자 표시줄 아래에 있는 항목을 선택하고 *Action* 속성을 `FileExit`로 설정합니다.

6 Edit 메뉴를 만듭니다.

- File 명령 오른쪽에 있는 항목을 선택하고 *Caption* 속성을 `&Edit`로 설정하고 *Enter*를 누릅니다.
- 이제 포커스가 Edit 항목 아래에 있습니다. *Action* 속성을 `EditCut1`로 설정합니다.
- Cut 아래에 있는 항목을 선택하고 *Action* 속성을 `EditCopy1`로 설정합니다.

- Copy 아래에 있는 항목을 선택하고 *Action* 속성을 EditPaste1로 설정합니다.





7 이제 Help 메뉴를 만듭니다.

- Edit 명령 오른쪽에 있는 항목을 선택하고 *Caption* 속성을 &Help로 설정하고 *Enter*를 누릅니다.
- Help 아래에 있는 항목을 선택하고 *Action* 속성을 HelpAbout로 설정합니다.

8 **X**를 클릭하여 메뉴 디자이너를 닫습니다.

9 File|Save를 선택하여 프로젝트에서 변경 사항을 저장합니다.

10 *F9* 키를 눌러 프로젝트를 컴파일하고 실행합니다.

참고 Debug 툴바의 Run 버튼을 클릭하거나 Run|Run을 선택하여 프로젝트를 실행할 수 있습니다.

F9 키를 눌러 프로젝트를 실행하면 애플리케이션 인터페이스가 나타납니다. 메뉴, 텍스트 영역, 상태 표시줄이 모두 폼에 나타납니다.

프로젝트를 실행할 때 Kylix는 런타임 폼에서 디자인한 것과 유사한 창에서 프로그램을 엽니다. 대부분의 명령이 비활성 상태라도 모든 메뉴는 작동합니다. 이미지는 아이콘과 연결한 메뉴 항목 옆에 나타납니다.

프로그램에 이미 기능들이 많이 들어 있지만 명령을 활성화하기 위해서는 몇 가지 작업을 더 해야 합니다. 툴바를 추가하면 명령에 쉽게 액세스할 수 있습니다.

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

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

텍스트 영역 지우기

프로그램을 실행하면 *Memo1*이라는 이름이 텍스트 영역에 나타납니다. Strings List editor를 사용하여 텍스트를 제거할 수 있습니다. 지금 텍스트를 지우지 않으면 마지막 단계에서 메인 폼을 초기화할 때 텍스트를 제거해야 합니다.

텍스트 영역을 지우려면 다음과 같이 합니다.

1 메인 폼에서 *Memo* 컴포넌트를 클릭합니다.

2 Object Inspector에서 *Lines* 속성 옆에 있는 값(TStrings)을 더블 클릭하여 String List editor를 표시합니다.

3 String List editor에서 제거하려는 텍스트를 선택하여 삭제하고 OK를 클릭합니다.

4 변경 사항을 저장하고 프로그램을 다시 실행합니다.

이제 메인 폼이 표시되면 텍스트 편집 영역은 지워져 있습니다.

툴바 추가

액션 리스트에서 액션을 설정했으므로 메뉴에서 사용했던 동일한 액션의 일부를 툴바에 추가할 수 있습니다.

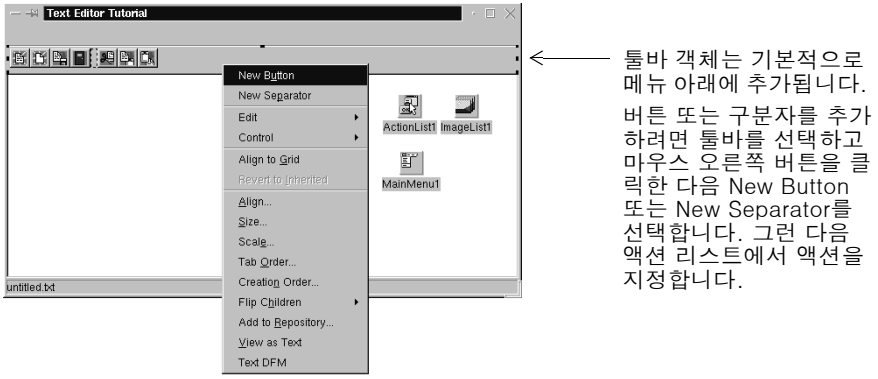
1 컴포넌트 팔레트의 Common Controls 탭에서 *ToolBar* 컴포넌트를 더블 클릭하여 폼에 추가합니다.

빈 툴바는 메인 메뉴 아래에 추가됩니다. 툴바를 계속 선택한 상태로 Object Inspector에서 다음 속성들을 변경합니다.

- 툴바의 *Indent* 속성을 4로 설정합니다.(이렇게 하면 툴바 왼쪽의 아이콘을 4픽셀 들여 씁니다.)
- 툴바의 *Images* 속성을 `ImageList1`로 설정합니다.
- *ShowHint*를 *True*로 설정합니다.(**팁:** *True*로 바꾸려면 *False*를 더블 클릭합니다)

2 다음과 같은 방법으로 툴바에 버튼과 구분자를 추가합니다.
- 툴바를 선택한 상태로 마우스 오른쪽 버튼을 클릭하고 New Button을 네 번 선택합니다.
- 마우스 오른쪽 버튼을 클릭하고 New Separator를 선택합니다.
- 마우스 오른쪽 버튼을 클릭하고 New Button을 세 번 이상 선택합니다.

참고 아이콘이 적합한지 걱정할 필요가 없습니다. 버튼에 액션을 지정하면 적합한 아이콘이 선택됩니다.



3 액션 리스트의 액션을 첫 번째 집합의 버튼으로 지정합니다.
   - 첫 번째 버튼을 선택하고 *Action* 속성을 `FileExit`로 설정합니다.
   - 두 번째 버튼을 선택하고 *Action* 속성을 `FileNew`로 설정합니다.
   - 세 번째 버튼을 선택하고 *Action* 속성을 `FileOpen`으로 설정합니다.
   - 네 번째 버튼을 선택하고 *Action* 속성을 `FileSave`로 설정합니다.

4 액션을 두 번째 집합의 버튼으로 지정합니다.
   - 첫 번째 버튼을 선택하고 *Action* 속성을 `EditCut1`로 설정합니다.
   - 두 번째 버튼을 선택하고 *Action* 속성을 `EditCopy1`로 설정합니다.
   - 세 번째 버튼을 선택하고 *Action* 속성을 `EditPaste1`로 설정합니다.

5 *F9* 키를 눌러 프로젝트를 컴파일하고 실행합니다.

   텍스트 에디터에는 이미 기능이 많이 들어 있습니다. 텍스트 영역 내에서 입력할 수 있습니다. 툴바를 확인합니다. 텍스트 영역에서 텍스트를 선택하면 Cut, Copy, Paste 버튼이 작동해야 합니다.

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

이벤트 핸들러 작성

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

코드가 없는 이벤트 핸들러나 뼈대 핸들러를 코드 에디터에서 생성할 수 있습니다. 뼈대 핸들러를 생성하려면 폼에 있는 컴포넌트를 더블 클릭하거나 Object Inspector에서 이벤트 오른쪽 공간을 클릭합니다. 뼈대 핸들러는 프로시저 이름과 **begin** 및 **end** 문이 포함된 빈 이벤트 핸들러입니다.

모든 메뉴 항목과 툴바 액션은 액션 리스트에 통합되므로 이벤트 핸들러를 액션 리스트에서 만들 수 있습니다.

이벤트와 이벤트 핸들러에 대한 자세한 내용은 *Kylix1 개발자 안내서*의 6장 "애플리케이션 사용자 인터페이스 개발" 또는 온라인 도움말을 참조하십시오.

New 명령에 대한 이벤트 핸들러 만들기

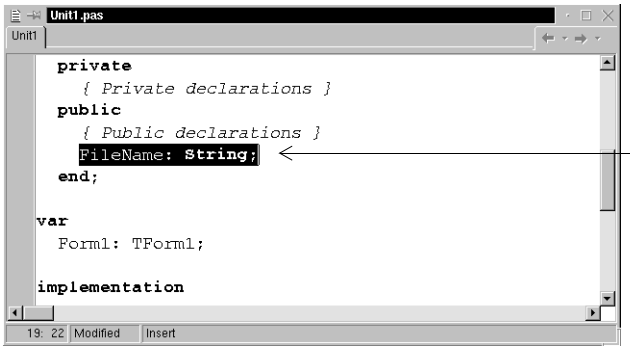
다음과 같은 방법으로 New 명령에 대한 이벤트 핸들러를 만듭니다.

1 View|Units를 선택한 다음 Unit1을 선택하여 Form1과 연결된 코드를 표시합니다.

2 이벤트 핸들러에서 사용할 파일 이름을 선언해야 합니다. 파일 이름에 대한 사용자 지정 속성을 추가하여 파일에 전역적으로 액세스할 수 있도록 합니다. Unit1.pas 파일 앞부분에 TForm1 클래스에 대한 공용 선언 부분을 두고 *{Public declarations}* 아래 줄에 다음과 같이 입력합니다.

```
FileName: String;
```

입력한 화면은 다음과 같이 보입니다.



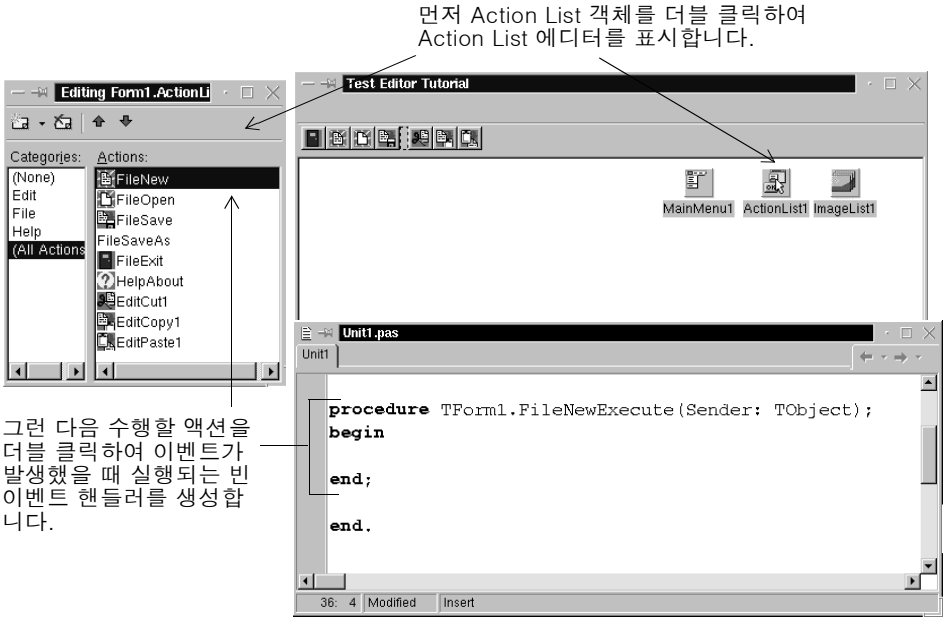
3 *F12* 키를 눌러 메인 폼으로 돌아갑니다.

팁 *F12* 키를 토글하면 폼과 관련 코드 사이를 전환할 수 있습니다.

4 폼에서 ActionList 아이콘을 더블 클릭하여 Action List 에디터를 표시합니다.

5 Action List 에디터에서 File 범주를 선택한 다음 FileNew 액션을 더블 클릭합니다.

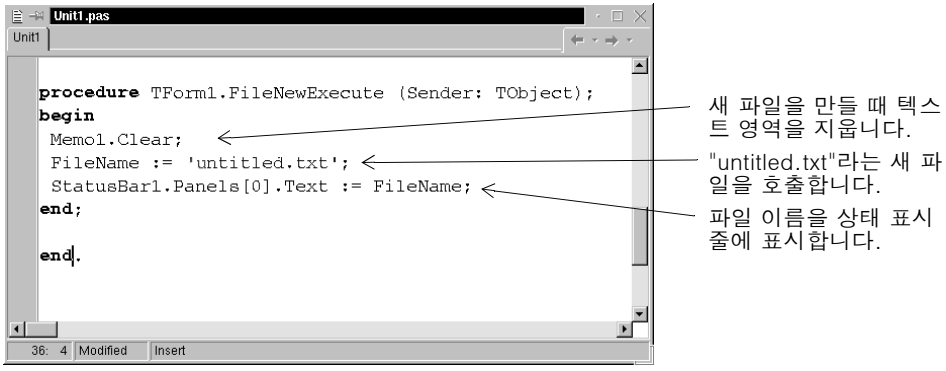
코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.



6 코드 에디터에서 커서가 있는 위치(**begin**과 **end** 사이)에 다음 줄을 입력합니다.

```
Memo1.Clear;
FileName := 'untitled.txt';
StatusBar1.Panels[0].Text := FileName;
```

입력한 이벤트 핸들러는 다음과 같이 나타납니다.



작업을 저장하면 File|New 명령에 대한 모든 작업이 끝납니다.

팁 창의 코드 부분의 크기를 조정하면 수평 스크롤을 줄일 수 있습니다.

Open 명령에 대한 이벤트 핸들러 만들기

파일을 열 때 File Open 대화 상자는 자동으로 표시됩니다. Open 명령에 이 대화 상자를 추가하려면 *TOpenDialog* 객체를 메인 에디터 폼에 가져다 놓습니다. 그러면 이 명령에 대한 이벤트 핸들러를 작성할 수 있습니다.

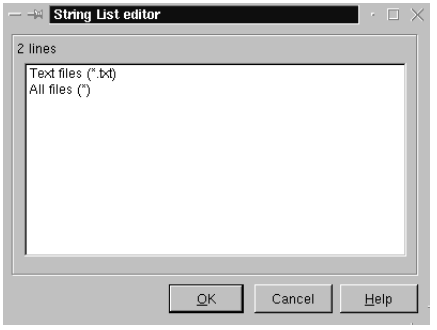
다음과 같은 방법으로 Open 대화 상자와 Open 명령에 대한 이벤트 핸들러를 만듭니다.

1 메인 폼을 찾습니다(빨리 찾으려면 View|Forms를 선택하고 Form1을 선택합니다).

2 컴포넌트 팔레트의 Dialogs 페이지에서 *OpenDialog* 컴포넌트를 더블 클릭하여 폼에 추가합니다. 이 컴포넌트는 넌비주얼 컴포넌트이므로 아무 곳에나 둘 수 있습니다. Kylix는 기본적으로 *OpenDialog1*이라는 이름을 지정합니다. (*OpenDialog1*의 *Execute* 메소드를 호출하면 파일을 열기 위한 표준 대화 상자를 호출합니다.)

3 Object Inspector에서 *OpenDialog1*의 속성을 다음과 같이 설정합니다.

- *DefaultExt*을 txt로 설정합니다.
- *Filter* 옆에 있는 텍스트 영역을 더블 클릭하여 String List editor를 표시합니다. 첫 번째 줄에 `Text files (*.txt)`를 입력합니다. "Text files"는 필터 이름이고 "(*.txt)"는 필터입니다. 두 번째 줄에서 `All files (*)`를 입력합니다. OK를 클릭합니다.



String List editor를 사용하여 *OpenDialog* 컴포넌트와 *SaveDialog* 컴포넌트에 대한 필터를 정의합니다.

- *Title*에 `Open file`을 입력합니다.

4 Action List 에디터는 계속 열어 두어야 합니다. 에디터를 닫았을 경우 폼에서 ActionList 아이콘을 더블 클릭합니다.

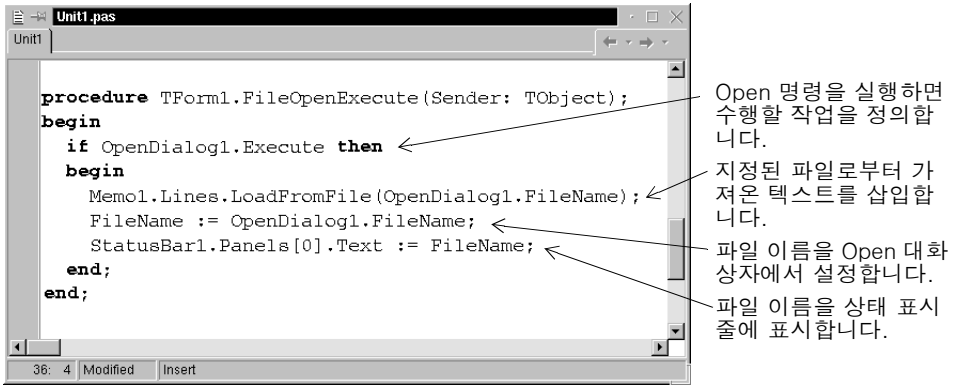
5 Action List 에디터에서 FileOpen 액션을 더블 클릭합니다.

코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

6 코드 에디터에서 커서가 있는 위치(**begin**과 **end** 사이)에 다음 줄을 입력합니다.

```
 if OpenDialog1.Execute then
begin
  Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
  FileName := OpenDialog1.FileName;
  StatusBar1.Panels[0].Text := FileName;
end;
```

입력한 FileOpen 이벤트 핸들러는 다음과 같이 나타납니다.



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

Save 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 Save 명령에 대한 이벤트 핸들러를 만듭니다.

1 Action List 에디터는 계속 열어 두어야 합니다. 에디터를 닫았을 경우 폼에서 ActionList 아이콘을 더블 클릭합니다.

2 Action List 에디터에서 FileSave 액션을 더블 클릭합니다.

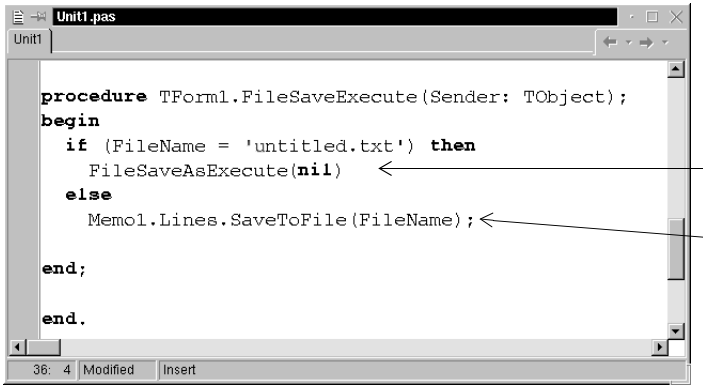
코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

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

```
if (FileName = 'untitled.txt') then
  FileSaveAsExecute(nil)
else
  Memo1.Lines.SaveToFile(FileName);
```

이 코드는 파일 이름이 지정되지 않았을 경우에 사용자가 이름을 지정할 수 있도록 SaveAs 대화 상자를 표시합니다. 이름이 지정되어 있으면 현재 이름으로 파일을 저장합니다. SaveAs 대화 상자는 4-21페이지에 있는 Save As 명령에 대한 이벤트 핸들러에서 정의됩니다. FileSaveAsExecute는 Save As 명령에 자동으로 부여된 이름입니다.

입력한 이벤트 핸들러는 다음과 같이 나타납니다.



이제 File|Save 명령에 대한 내용이 모두 끝났습니다.

Save As 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 Save As 명령에 대한 이벤트 핸들러를 만듭니다.

1 컴포넌트 팔레트의 Dialogs 탭에서 *SaveDialog* 컴포넌트를 폼에 가져다 놓습니다. 이 컴포넌트는 넌비주얼 컴포넌트이므로 아무 곳에나 둘 수 있습니다. Kylix는 기본적으로 *SaveDialog1*이라는 이름을 지정합니다. (*SaveDialog*의 *Execute* 메소드를 호출하면 파일 저장을 위한 표준 대화 상자를 호출합니다.)

2 Object Inspector에서 *SaveDialog1*의 속성을 다음과 같이 설정합니다.

- *DefaultExt*을 txt로 설정합니다.
- *Filter* 옆에 있는 텍스트 영역을 더블 클릭하여 String List editor를 표시합니다. Editor에서 파일 형식에 대한 필터를 Open 대화 상자에 있는 것으로 지정합니다. 첫 번째 줄에서 Text files (*.txt)를 입력하고 두 번째 줄에서 All files (*)를 입력합니다. OK를 클릭합니다.
- *Title*이 Save As로 설정되어 있는지 확인합니다.

참고 Action List 에디터는 계속 열어 두어야 합니다. 에디터를 닫았을 경우 폼에서 ActionList 아이콘을 더블 클릭합니다.

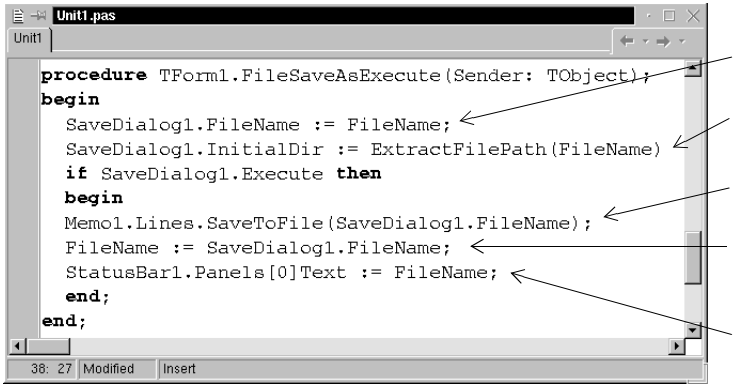
3 Action List 에디터에서 FileSaveAs 액션을 더블 클릭합니다.

코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

4 코드 에디터의 커서가 있는 위치에서 다음 줄을 입력합니다.

```
  SaveDialog1.FileName := FileName;
  SaveDialog1.InitialDir := ExtractFilePath(Filename);
if SaveDialog1.Execute then
begin
  Memo1.Lines.SaveToFile(SaveDialog1.FileName);
  FileName := SaveDialog1.FileName;
  StatusBar1.Panels[0].Text := FileName;
end;
```

입력한 FileSaveAs 이벤트 핸들러는 다음과 같이 나타납니다.



이제 File|SaveAs 명령에 대한 내용이 모두 끝났습니다.

Exit 명령에 대한 이벤트 핸들러 만들기

다음과 같은 방법으로 Exit 명령에 대한 이벤트 핸들러를 만듭니다.

1 Action List 에디터는 계속 열어 두어야 합니다. 에디터를 닫았을 경우 폼에서 ActionList 아이콘을 더블 클릭합니다.

2 Action List 에디터에서 FileExit 액션을 더블 클릭합니다.

코드 에디터가 열리면서 이벤트 핸들러 내부에 커서가 위치하게 됩니다.

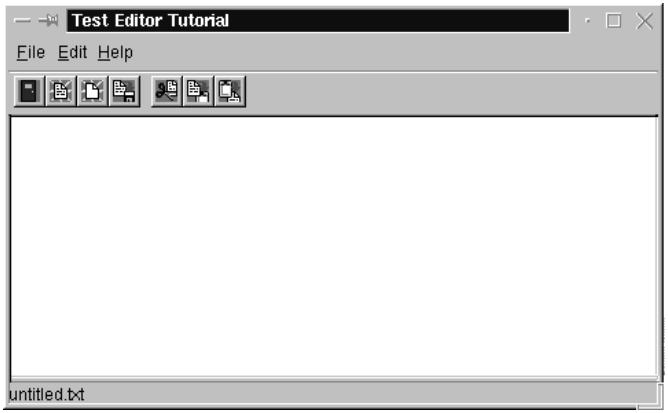
3 코드 에디터의 커서가 있는 위치에 다음 줄을 입력합니다.

```
Close;
```

이 코드는 메인 폼의 close 메소드를 호출합니다. 이제 File|Exit 명령에 필요한 작업이 모두 끝났습니다.

4 File|Save All을 선택하여 프로젝트를 저장합니다.

이제까지 작업한 내용이 어떻게 나타나는지 보려면 *F9* 키를 눌러 애플리케이션을 실행합니다.



대부분의 버튼과 툴바 버튼은 작동하지만 아직 다 끝난 것은 아닙니다.

5 디자인 모드로 돌아가려면 **X**를 클릭하여 텍스트 에디터 애플리케이션을 닫습니다.

오류 메시지가 발생했으면 메시지를 클릭하여 오류가 발생한 위치로 이동하십시오. 반드시 이 자습서에서 설명한 단계대로 따라야 합니다.

About 상자 만들기

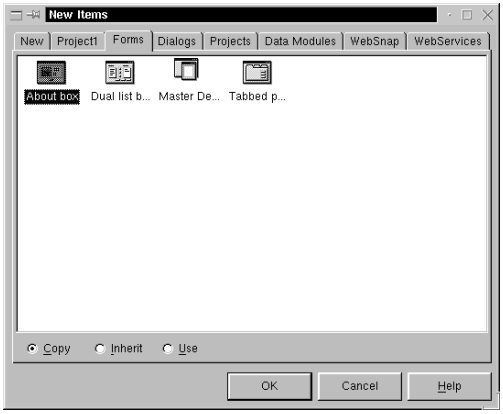
많은 애플리케이션에는 이름, 버전, 로고와 같은 제품에 대한 정보 및 저작권 정보를 비롯한 기타 법률 정보를 표시하는 About 상자가 있습니다.

액션 리스트에서 Help About 명령을 이미 설정했습니다.

다음과 같은 방법으로 About 상자를 만듭니다.

1 File|New를 선택하여 New Items 대화 상자를 표시하고 Forms 탭을 클릭합니다.

2 Forms 탭에서 About Box를 더블 클릭합니다.



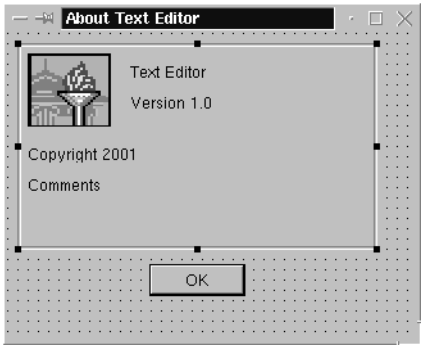
About 상자를 쉽게 만들기 위한 새 폼이 만들어집니다.

3 About 상자에서 다음과 같이 *TLabel* 항목을 선택하고 Object Inspector에서 *Caption* 속성을 변경합니다.

- `Product Name`을 `Text Editor`로 변경합니다.
- `Version` 옆에 `1.0`을 추가합니다.
- `Copyright` 옆에 년도를 추가합니다.

4 폼을 선택하고 Object Inspector에서 *Caption* 속성을 `About Text Editor`로 변경합니다.

팁 폼을 선택하는 가장 쉬운 방법은 그리드 부분을 클릭하는 것입니다.



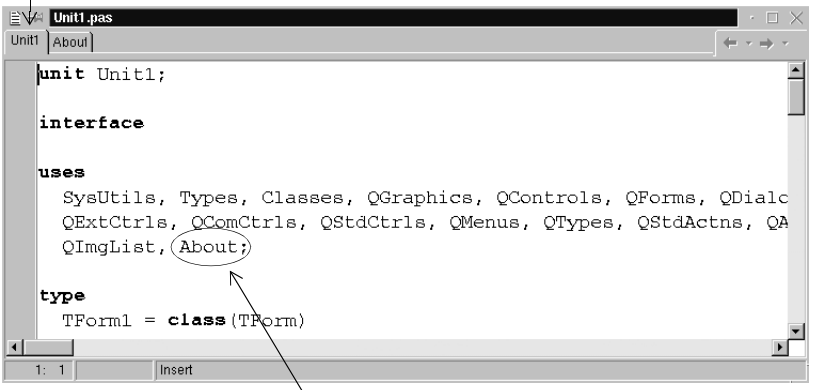
Object Repository에는 애플리케이션에 대한 설명을 추가할 수 있는 표준 About 상자가 들어 있습니다.

5 File|Save As를 선택하고 About.pas로 저장하여 About 상자 폼을 저장합니다.

6 Kylix 코드 에디터에 두 개의 파일, Unit1과 About가 나타나야 합니다. Unit1 탭을 클릭하여 Unit1.pas를 표시합니다.

7 **uses** 절에 포함된 유닛 목록의 끝에 About 단어를 입력하여 새 About 유닛을 Unit1의 **uses** 부분에 추가합니다.

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



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

8 액션 리스트에서 HelpAbout 액션을 더블 클릭하여 이벤트 핸들러를 만듭니다.

9 코드 에디터에서 커서가 있는 위치에 다음 행을 입력합니다.

```
AboutBox.ShowModal;
```

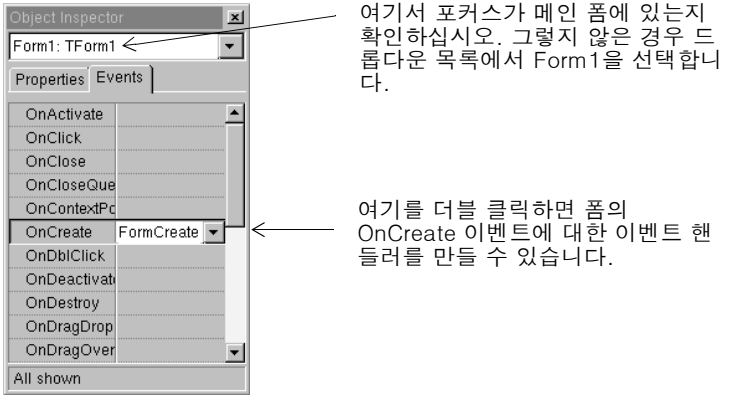
이 코드는 사용자가 Help|About를 클릭할 때 About 상자를 엽니다. ShowModal은 모드 상태, 즉 런타임 상태로 폼을 열기 때문에 폼이 닫힐 때까지는 아무것도 할 수 없습니다.

애플리케이션 완료

이제 애플리케이션이 거의 끝났습니다. 메인 폼에 지정할 몇 가지 항목만 남아 있습니다. 다음과 같은 방법으로 애플리케이션을 완료합니다.

1 메인 폼의 위치를 찾습니다(빨리 찾으려면 *F12* 키를 누릅니다).

2 포커스가 컴포넌트가 아닌 폼에 있는 것을 확인하십시오. Object Inspector의 상단 리스트 박스에 Form1: TForm1이 나타나야 합니다. (Form1: Tform1이 아니면 드롭다운 목록에서 Form1을 선택합니다.)



3 폼을 만들 때, 즉 애플리케이션을 열 때 발생하는 이벤트를 처리하는 이벤트 핸들러를 만들려면 Events 탭에서 OnCreate를 더블 클릭하고 드롭다운 목록에서 FormCreate를 선택합니다.

4 코드 에디터의 커서가 있는 위치에서 다음을 입력합니다.

```
FileName := 'untitled.txt';
StatusBar1.Panels[0].Text := FileName;
Memo1.Clear;
```

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

5 애플리케이션을 실행하려면 *F9* 키를 누릅니다.

이제 텍스트 에디터가 잘 작동하는지 테스트할 수 있습니다. 오류가 발생한 경우 오류 메시지를 더블 클릭하면 오류가 발생한 코드로 즉시 이동합니다.

이제 모두 완료했습니다.

5

데이터베이스 애플리케이션 만들기- 자습서

이 자습서에서는 직원 데이터베이스 샘플을 보거나 업데이트할 수 있는 데이터베이스 애플리케이션을 만듭니다.

참고 이 자습서는 데이터베이스 컴포넌트가 포함된 Kylix 기업용 에디션과 전문가용 에디션으로 작성되었습니다. 자습서를 성공적으로 끝내려면 InterBase도 설치되어 있어야 합니다.

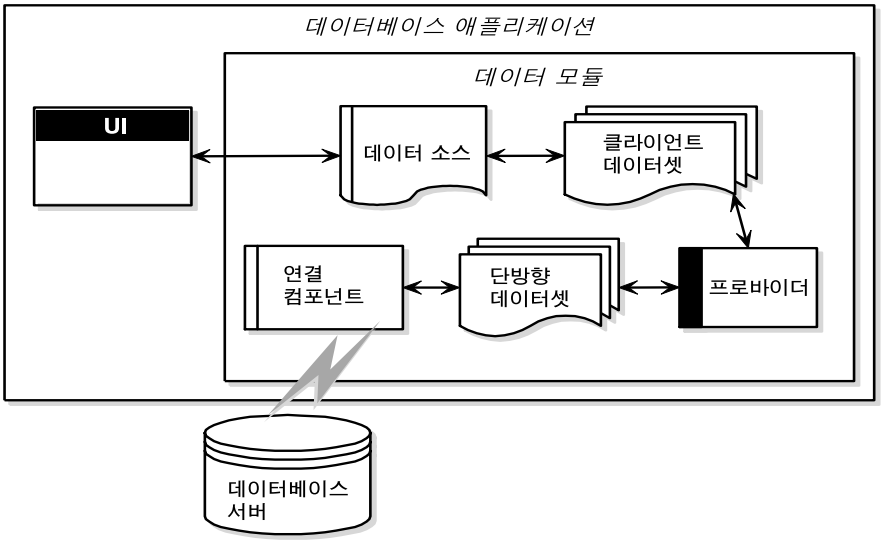
데이터베이스 아키텍처 개요

데이터베이스 애플리케이션의 아키텍처가 처음에는 복잡해 보일 수 있지만 여러 가지 컴포넌트를 사용함으로써 실제 데이터베이스 애플리케이션의 개발과 유지 보수가 간편해집니다.

데이터베이스 애플리케이션은 사용자 인터페이스, 데이터 액세스 컴포넌트의 집합, 데이터베이스의 3가지 중요한 부분으로 이루어집니다. 이 자습서에서는 dbExpress 데이터베이스 애플리케이션을 만듭니다. 다른 데이터베이스 애플리케이션도 이와 유사한 아키텍처로 되어 있습니다.

사용자 인터페이스는 그리드와 같이 사용자가 편집한 데이터를 데이터베이스에 포스트할 수 있는 data-aware 컨트롤을 포함합니다. 데이터 액세스 컴포넌트는 데이터 소스, 클라이언트 데이터셋, 데이터 프로바이더, 단방향 데이터셋, 연결 컴포넌트를 포함합니다. 데이터 소스는 사용자 인터페이스와 클라이언트 데이터셋 사이를 이어 주는 역할을 합니다. 클라이언트 데이터셋은 애플리케이션의 핵심 부분으로 메모리에 버퍼링되고 원본으로 사용하는 데이터베이스의 레코드 집합을 포함하고 있습니다. 프로바이더는 데이터베이스에서 데이터를 직접 fetch하는 단방향 데이터셋과 클라이언트 데이터셋

간에 데이터를 전송합니다. 연결 컴포넌트는 데이터베이스에 연결하는 컴포넌트입니다. 단방향 데이터셋의 각 타입은 연결 컴포넌트의 서로 다른 타입을 사용합니다.



참고 데이터베이스 개발에 대한 자세한 내용은 *Kylix1 개발자 안내서*의 14장 "데이터베이스 애플리케이션 디자인"이나 온라인 도움말을 참조하십시오.

새 프로젝트 만들기

자습서를 시작하기 전에 소스 파일을 저장할 폴더를 만듭니다. 그런 다음 새 프로젝트를 열고 저장합니다.

1 Tutorial이라는 폴더를 만들고 이 장에서 만드는 모든 프로젝트 파일을 이 디렉토리에 저장합니다.

2 Kylix 시작 시 만든 기본 프로젝트를 사용하거나 File|New Application을 선택하여 새 프로젝트를 시작합니다.

3 File|Save All을 선택하여 디스크에 파일을 저장합니다. Save As 대화 상자가 나타나면 Tutorial 폴더를 탐색하여 각 파일을 기본 이름으로 저장합니다.

   나중에 File|Save All을 선택하여 작업한 내용을 저장할 수 있습니다. File|Reopen을 선택하고 목록에서 자습서를 선택하면 나중에 계속할 수 있습니다.

데이터 액세스 컴포넌트 설정

데이터 액세스 컴포넌트는 데이터(데이터셋)를 의미하는 컴포넌트와 이 데이터셋을 애플리케이션의 다른 부분과 연결시키는 컴포넌트를 의미합니다. 각각의 데이터 액세스 컴포넌트는 다음에 오는 하위 컴포넌트를 가리킵니다. 예를 들어 데이터 소스는 클라이

언트 데이터셋을 가리키고, 클라이언트 데이터셋은 프로바이더를 가리킵니다. 그러므로 데이터 액세스 컴포넌트를 설정할 때 컴포넌트가 *가리키는* 순서대로 컴포넌트를 추가해야 합니다.

다음 단원에서는 데이터베이스 컴포넌트를 추가하여 데이터베이스 연결, 단방향 데이터셋, 프로바이더, 클라이언트 데이터셋, 데이터 소스를 만듭니다. 그런 다음 애플리케이션의 사용자 인터페이스를 만듭니다. 이 컴포넌트들은 컴포넌트 팔레트의 dbExpress, Data Access, Data Controls 페이지에 있습니다.

팁 사용자 인터페이스는 폼에 두고 데이터 액세스 컴포넌트는 데이터 모듈에 두어 사용자 인터페이스와 데이터 모듈을 분리하는 것이 좋습니다. 그러나 이 자습서에서는 편의를 위해 사용자 인터페이스와 모든 컴포넌트를 하나의 폼에 둡니다.

데이터베이스 연결 설정

dbExpress 페이지는 SQL 데이터베이스 서버에 빠르게 액세스할 수 있는 컴포넌트 집합을 포함합니다.

데이터베이스에 연결하려면 연결 컴포넌트를 추가해야 합니다. 사용할 연결 컴포넌트의 타입은 사용하는 데이터셋 컴포넌트의 타입에 따라 다릅니다. 이 자습서에서 *TSQLConnection* 및 *TSQLDataSet* 컴포넌트를 사용합니다.

dbExpress 연결 컴포넌트를 추가하려면 다음과 같이 합니다.

1. 컴포넌트 팔레트에서 dbExpress 페이지를 클릭하고 *TSQLConnection* 컴포넌트를 더블 클릭하여 폼에 이 컴포넌트를 놓습니다. *TSQLConnection* 컴포넌트를 찾으려면 팔레트에 있는 아이콘 위에 마우스를 잠시 갖다 대면 도움말 힌트에서 컴포넌트 이름을 보여 줍니다. 이 컴포넌트를 기본적으로 *SQLConnection1*이라고 합니다.

   *TSQLConnection*은 넌비주얼 컴포넌트이므로 어디에 두든지 상관 없습니다. 그러나 이 자습서에서는 모든 넌비주얼 컴포넌트를 폼의 상단에 둡니다.

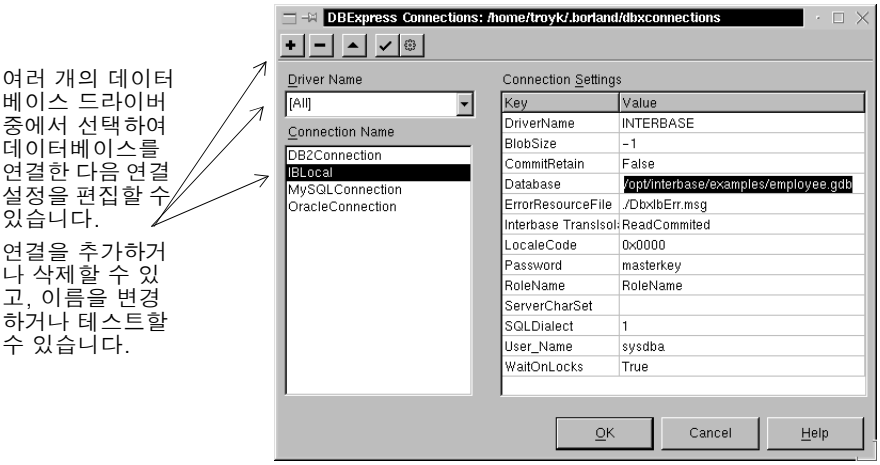
   팁 폼에 놓은 컴포넌트의 캡션을 표시하려면 Tools|Environment Options를 선택하고 Show component captions를 클릭합니다.

2. Object Inspector에서 *ConnectionName* 속성을 `IBLocal`(드롭다운 목록에 있음)로 설정합니다.
3. *LoginPrompt* 속성을 `False`로 설정합니다. (이 속성을 `False`로 설정하면 데이터베이스에 로그온할 때마다 나타나는 프롬프트 메시지가 나타나지 않습니다.)
4. *TSQLConnection* 컴포넌트를 더블 클릭하여 Connection Editor를 표시합니다.

   TSQLConnection 컴포넌트에 대한 연결 구성을 선택하거나 dbxconnections 파일에 저장된 연결을 편집하려면 Connection Editor를 사용합니다.

5. Connection Editor에서 시스템에 있는 employee.gdb라는 데이터베이스 파일의 경로 이름을 지정합니다. 이 자습서에서 Kylix에서 제공하는 예제 InterBase 데이

터베이스 employee.gdb에 연결할 것입니다. 기본적으로 InterBase 설치는 employee.gdb를 /opt/interbase/examples에 저장합니다.



6 User_Name 필드와 Password 필드의 값이 적절한지 확인합니다. 기본값을 변경하지 않았으면 필드를 변경할 필요가 없습니다. 다른 사용자가 데이터베이스 액세스를 관리하면 데이터베이스에 액세스하기 위해 자신의 사용자 이름과 암호를 알고 있는 것이 좋습니다.

7 필드 확인과 편집이 끝나면 OK를 클릭하여 Connection Editor를 닫고 변경 내용을 저장합니다.

변경 내용은 dbxconnections 파일에 기록되며 선택한 연결은 *SQL Connection* 컴포넌트의 *ConnectionName* 속성 값으로 할당됩니다.

8 File|Save All을 선택하여 프로젝트를 저장합니다.

단방향 데이터셋 설정

간단한 데이터베이스 애플리케이션에서는 데이터베이스의 정보에 액세스할 때 데이터셋을 사용합니다. dbExpress 애플리케이션에서는 단방향 데이터셋을 사용합니다. 단방향 데이터셋은 데이터베이스에서 데이터를 읽어 오지만 데이터를 업데이트하지는 않습니다.

다음과 같은 방법으로 단방향 데이터셋을 추가합니다.

1 dbExpress 페이지에서 *TSQLDataSet*을 폼의 맨 위에 가져다 놓습니다.

2 Object Inspector에서 이 컴포넌트의 *SQLConnection* 속성을 `SQLConnection1`로 설정합니다(데이터베이스 연결은 미리 만들어 두어야 합니다).

3 *CommandText* 속성을 `"Select * from sales"`로 설정하여 데이터셋이 실행할 명령을 지정합니다. Object Inspector에서 Select 구문을 직접 입력하거나 *CommandText* 오른쪽에 있는 생략 기호를 클릭하여 CommandText 에디터를 표시한 다음 쿼리 문을 직접 작성할 수 있습니다.

4 *Active*를 `True`로 설정하여 데이터셋을 엽니다.

5 File|Save All을 선택하여 프로젝트를 저장합니다.

프로바이더, 클라이언트 데이터셋, 데이터 소스 설정

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

프로바이더 컴포넌트를 사용하면 클라이언트 데이터셋이 다른 데이터셋으로부터 데이터를 얻을 수 있습니다. 프로바이더는 클라이언트 데이터셋으로부터 데이터 요청을 받으면 데이터를 fetch한 다음 패키지로 만들어서 클라이언트 데이터셋에게 데이터를 반환합니다. dbExpress에서 프로바이더는 클라이언트 데이터셋으로부터 업데이트 내용을 받아서 데이터베이스 서버에 업데이트 내용을 적용합니다.

다음과 같은 방법으로 프로바이더를 추가합니다.

1 Data Access 페이지에서 *TDataSetProvider* 컴포넌트를 폼의 맨 위에 가져다 놓습니다.

2 Object Inspector에서 프로바이더의 *DataSet* 속성을 `SQLDataSet1`로 설정합니다.

클라이언트 데이터셋은 메모리에 있는 데이터를 버퍼링합니다. 클라이언트 데이터셋은 업데이트 내용을 캐시하여 데이터베이스로 보냅니다. 데이터 소스 컴포넌트를 사용하는 사용자 인터페이스의 data-aware 컨트롤에 데이터를 전달하는 데 클라이언트 데이터셋을 사용할 수 있습니다.

다음과 같은 방법으로 클라이언트 데이터셋을 추가합니다.

1 Data Access 페이지에서 *TClientDataSet* 컴포넌트를 *TDataSetProvider* 컴포넌트의 오른쪽에 가져다 놓습니다.

2 *ProviderName* 속성을 `DataSetProvider1`로 설정합니다.

3 *Active* 속성을 `True`로 설정하여 데이터가 애플리케이션에 전달되도록 합니다.

데이터 소스는 클라이언트 데이터셋을 data-aware 컨트롤과 연결합니다. 데이터를 표시하고 처리하려면 각 data-aware 컨트롤을 데이터 소스 컴포넌트에 연결해야 합니다. 이와 비슷하게 데이터를 폼의 data-aware 컨트롤에 표시하고 처리하려면 모든 데이터셋을 데이터 소스 컴포넌트에 연결해야 합니다.

다음과 같은 방법으로 데이터 소스를 추가합니다.

1 Data Access 페이지에서 *TDataSource* 컴포넌트를 *TClientDataSet* 컴포넌트의 오른쪽에 가져다 놓습니다.

2 데이터 소스의 *DataSet* 속성을 `ClientDataSet1`로 설정합니다.

3 File|Save All을 선택하여 프로젝트를 저장합니다.

지금까지 넌비주얼 데이터베이스 인프라스트럭처를 애플리케이션에 추가했습니다. 이제 사용자 인터페이스를 디자인해야 합니다.

사용자 인터페이스 디자인

이제 애플리케이션에 비주얼 컨트롤을 추가하여 사용자가 데이터를 보거나 편집하고 저장할 수 있도록 해야 합니다. Data Controls 페이지에서는 data-aware 컨트롤 집합을 제공하여 데이터베이스의 데이터를 처리하고 사용자 인터페이스를 구축할 수 있도록 합니다. 데이터베이스를 그리드에 표시하고 몇 가지 명령과 탐색 막대를 추가할 것입니다.

그리드와 탐색 막대 만들기

다음과 같은 방법으로 애플리케이션의 인터페이스를 만듭니다.

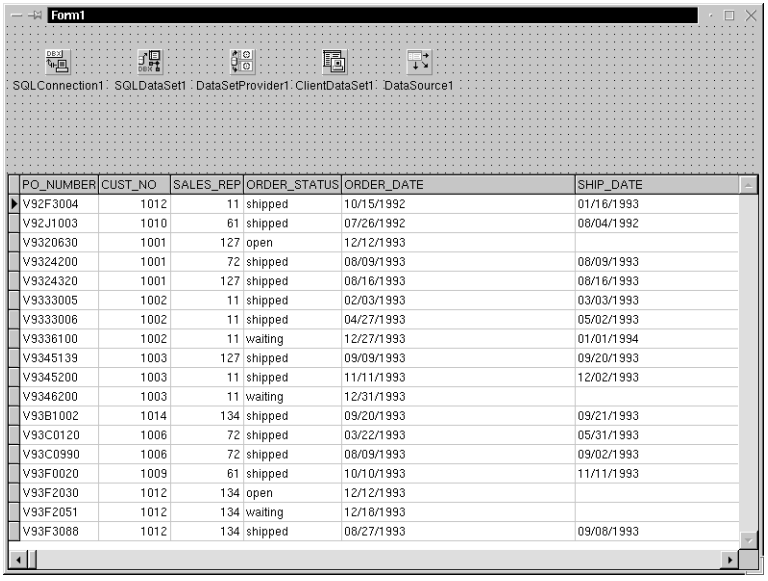
1 먼저 폼에 그리드를 추가합니다. Data Controls 페이지에서 *TDBGrid* 컴포넌트를 폼에 가져다 놓습니다.

2 *DBGrid*의 속성을 설정하여 그리드를 연결합니다. Object Inspector에서 *Anchors* 옆에 있는 **+**를 클릭하여 *akLeft*, *akTop*, *akRight*, *akBottom*을 표시하고 모두 `True`로 설정합니다. 가장 간단하게 값을 바꾸려면 Object Inspector에서 각각의 속성 옆에 있는 `False`를 더블 클릭하면 됩니다.

3 *Align* 속성을 `alBottom`으로 설정하여 그리드를 폼의 아래쪽에 정렬합니다. *Height* 속성을 `400`으로 설정하여 그리드 크기를 늘립니다.

4 그리드의 *DataSource* 속성을 `DataSource1`로 설정합니다. 이 값을 설정하면 그리드가 직원 데이터베이스의 데이터로 채워집니다. 그리드에 데이터가 표시되지 않으면 앞에서 설명한 지침대로 폼의 모든 객체 속성을 제대로 설정했는지 확인하십시오.

이제 애플리케이션이 다음과 같이 나타날 것입니다.



PO_NUMBER	CUST_NO	SALES_REP	ORDER_STATUS	ORDER_DATE	SHIP_DATE
V92F3004	1012	11	shipped	10/15/1992	01/16/1993
V92J1003	1010	61	shipped	07/26/1992	08/04/1992
V9320630	1001	127	open	12/12/1993	
V9324200	1001	72	shipped	08/09/1993	08/09/1993
V9324320	1001	127	shipped	08/16/1993	08/16/1993
V9333005	1002	11	shipped	02/03/1993	03/03/1993
V9333006	1002	11	shipped	04/27/1993	05/02/1993
V9336100	1002	11	waiting	12/27/1993	01/01/1994
V9345139	1003	127	shipped	09/09/1993	09/20/1993
V9345200	1003	11	shipped	11/11/1993	12/02/1993
V9346200	1003	11	waiting	12/31/1993	
V93B1002	1014	134	shipped	09/20/1993	09/21/1993
V93C0120	1006	72	shipped	03/22/1993	05/31/1993
V93C0990	1006	72	shipped	08/09/1993	09/02/1993
V93F0020	1009	61	shipped	10/10/1993	11/11/1993
V93F2030	1012	134	open	12/12/1993	
V93F2051	1012	134	waiting	12/18/1993	
V93F3088	1012	134	shipped	08/27/1993	09/08/1993

DBGrid 컨트롤은 IDE 환경에서 작업하면 디자인 타임 시 데이터를 표시하므로 애플리케이션이 데이터베이스에 제대로 연결되었는지 여부를 확인할 수 있습니다. 그러나 디자인 타임에 데이터를 편집할 수는 없습니다. 테이블의 데이터를 편집하려면 애플리케이션을 실행해야 합니다.

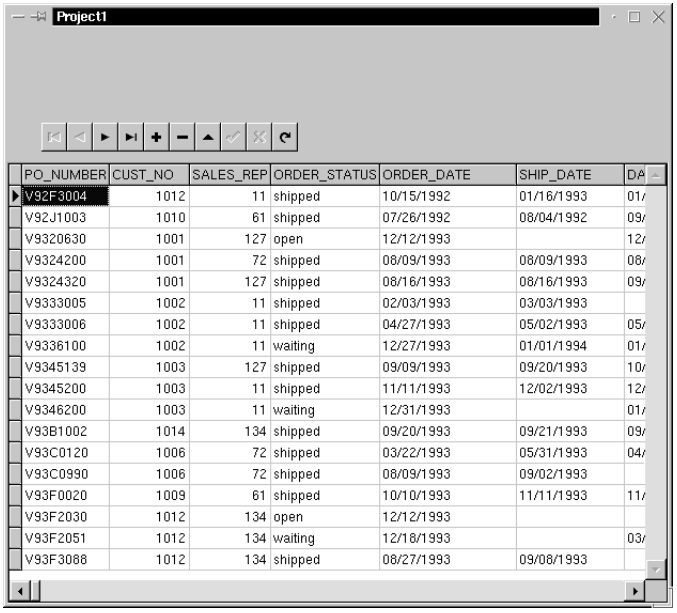
5 Data Controls 페이지에서 *TDBNavigator* 컨트롤을 폼에 가져다 놓습니다. 데이터베이스 탐색기는 다음 화살표와 이전 화살표를 사용하여 데이터셋 내에서 이동하고 데이터에 작업을 수행하기 위한 툴입니다.

6 탐색기 막대의 *DataSource* 속성을 DataSource1로 설정하여 탐색기로 클라이언트 데이터셋의 데이터를 찾아볼 수 있도록 합니다.

7 탐색기 막대의 *ShowHint* 속성을 True로 설정합니다. (*ShowHint*를 True로 설정하여 런타임 시 탐색기 막대에 있는 항목에 커서를 잠시 두면 도움말 힌트가 나타납니다.)

8 File|Save All을 선택하여 프로젝트를 저장합니다.

9 *F9* 키를 눌러 프로젝트를 컴파일하고 실행합니다. Run 메뉴의 Run을 선택하거나 Debug 툴바의 Run 버튼을 클릭해도 프로젝트를 실행할 수 있습니다.



PO_NUMBER	CUST_NO	SALES_REP	ORDER_STATUS	ORDER_DATE	SHIP_DATE	DA
V92F3004	1012	11	shipped	10/15/1992	01/16/1993	01/
V92J1003	1010	61	shipped	07/26/1992	08/04/1992	09/
V9320630	1001	127	open	12/12/1993		12/
V9324200	1001	72	shipped	08/09/1993	08/09/1993	08/
V9324320	1001	127	shipped	08/16/1993	08/16/1993	09/
V9333005	1002	11	shipped	02/03/1993	03/03/1993	
V9333006	1002	11	shipped	04/27/1993	05/02/1993	05/
V9336100	1002	11	waiting	12/27/1993	01/01/1994	01/
V9345139	1003	127	shipped	09/09/1993	09/20/1993	10/
V9345200	1003	11	shipped	11/11/1993	12/02/1993	12/
V9346200	1003	11	waiting	12/31/1993		01/
V93B1002	1014	134	shipped	09/20/1993	09/21/1993	09/
V93C0120	1006	72	shipped	03/22/1993	05/31/1993	04/
V93C0990	1006	72	shipped	08/09/1993	09/02/1993	
V93F0020	1009	61	shipped	10/10/1993	11/11/1993	11/
V93F2030	1012	134	open	12/12/1993		
V93F2051	1012	134	waiting	12/18/1993		03/
V93F3088	1012	134	shipped	08/27/1993	09/08/1993	

프로젝트를 실행할 때 Kylix는 폼에서 디자인한 것과 유사한 창에서 프로그램을 엽니다. 직원 데이터베이스로 탐색 막대를 테스트할 수 있습니다. 예를 들어 화살표 명령으로 레코드 간에 이동하거나 **+** 명령으로 레코드를 추가하거나 **-** 명령으로 레코드를 삭제할 수 있습니다.

팁 이전 버전의 애플리케이션을 테스트할 때 오류가 발생하면 Run|Program Reset을 선택하여 디자인 타임 뷰로 돌아가십시오.

메뉴에 대한 지원 추가

프로그램에 이미 기능이 많이 포함되었지만 GUI 애플리케이션으로 보기에는 아직도 부족합니다. 예를 들어 대개의 애플리케이션에는 메뉴뿐만 아니라 메뉴를 편리하게 실행하도록 버튼도 구현되어 있습니다.

이 단원에서는 *액션 리스트(action list)*를 추가할 것입니다. 액션 리스트를 사용하지 않아도 메뉴, 툴바, 버튼을 만들 수 있지만 액션 리스트를 사용하면 사용자 명령에 대한 응답을 집중된 방식으로 개발하고 유지 관리할 수 있습니다.

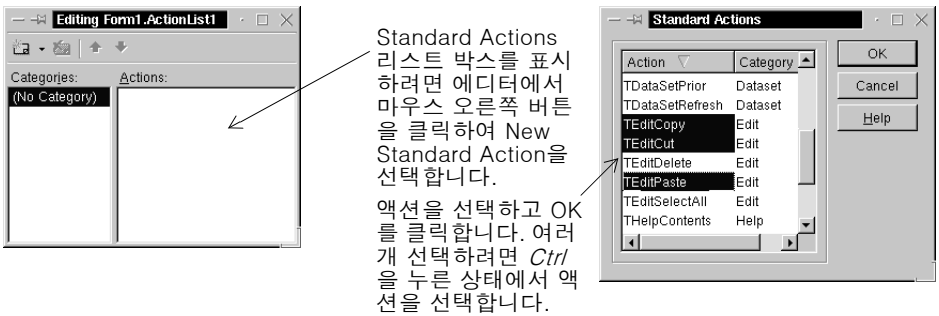
1 애플리케이션이 계속 실행 중이라면 오른쪽 위 모서리에 있는 **X**를 클릭하여 애플리케이션을 닫고 폼의 디자인 타임 뷰로 돌아옵니다.

2 컴포넌트 팔레트의 Common Controls 페이지에서 *ImageList* 컴포넌트를 폼에 가져다 놓습니다. 다른 넌비주얼 컴포넌트 옆에 둡니다. *ImageList*는 잘라내기나 붙여넣기와 같은 표준 액션 아이콘을 포함하고 있습니다.

3 컴포넌트 팔레트의 Standard 페이지에서 *ActionList* 컴포넌트를 폼에 가져다 놓습니다. 액션 리스트의 *Images* 속성을 `ImageList1`로 설정합니다.

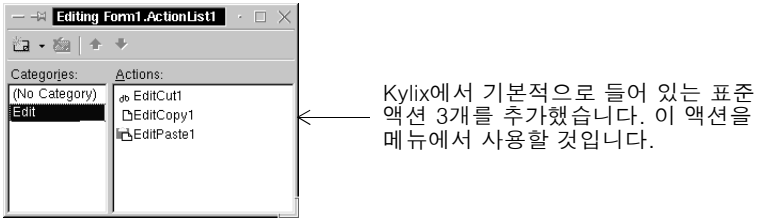
4 액션 리스트를 더블 클릭하여 Action List 에디터를 표시합니다.

5 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Standard Action을 선택합니다. Standard Actions 리스트 박스가 나타납니다.



6 *TEditCopy*, *TEditCut*, *TEditPaste*를 선택합니다. 여러 개 선택하려면 *Ctrl* 키를 누른 상태에서 액션을 선택합니다. 그런 다음 OK를 클릭합니다.

Action List 에디터에는 표준 액션과 기본적으로 연결되어 있는 이미지가 함께 나타납니다.



참고 연결되어 있는 이미지를 변경하려면 텍스트 에디터 자습서 4-10페이지의 "이미지 목록에 이미지 추가"를 참조하십시오.

7 Action List 에디터를 마우스 오른쪽 버튼으로 클릭하고 New Action을 선택하여 기본으로 제공된 액션이 아닌 다른 액션을 추가합니다. Action1이 기본적으로 추가됩니다. Object Inspector에서 *Caption* 속성을 `Update Now!`로 설정합니다.

이것과 동일한 액션을 메뉴와 버튼에 사용할 것입니다. 나중에 이벤트 핸들러를 추가하여 데이터베이스를 업데이트할 것입니다.

8 (No Category)를 클릭하고 마우스 오른쪽 버튼을 클릭하여 New Action을 선택하여 다른 액션을 추가합니다. Action2가 추가됩니다. *Caption* 속성을 `E&xit`로 설정합니다.

9 오른쪽 위 모서리에 있는 **X**를 눌러 Action List 에디터를 닫습니다.

3개의 표준 액션과 나중에 이벤트 핸들러를 연결할 다른 두 개의 액션을 추가했습니다.

10 File|Save All을 선택하여 프로젝트를 저장합니다.

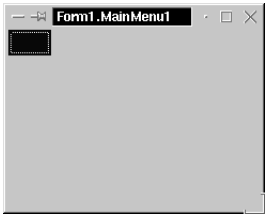
메뉴 추가

이 단원에서는 2개의 드롭다운 메뉴(File, Edit)가 있는 메인 메뉴 표시줄을 추가하고 액션 리스트의 액션을 사용하여 메뉴 항목을 각 메뉴에 추가합니다.

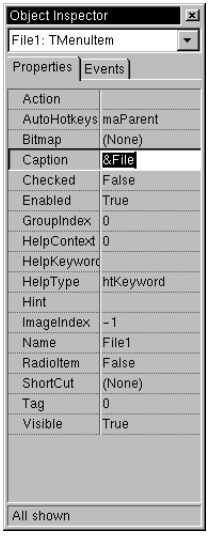
1 컴포넌트 팔레트의 Standard 페이지에서 *TMainMenu* 컴포넌트를 폼에 가져다 놓습니다. 이 컴포넌트를 다른 넌비주얼 컴포넌트 옆에 끌어다 놓습니다.

2 메인 메뉴의 *Images* 속성을 `ImageList1`로 설정하여 이미지 목록을 메뉴 항목과 연결합니다.

3 *TMainMenu* 컴포넌트를 더블 클릭하여 메뉴 디자이너를 표시합니다.

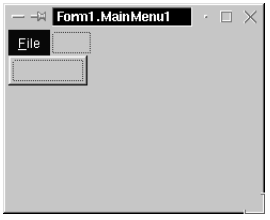


4 `&File`을 입력하여 첫 번째 최상위 메뉴 항목의 *Caption* 속성을 설정하고 *Enter*를 누릅니다.



`&File`을 입력하고 *Enter*를 누르면 최상위 File 명령이 나타나고 여기에 첫 번째 메뉴 항목을 추가할 수 있습니다.

앰퍼샌드(&) 바로 뒤에 있는 문자는 가속키가 됩니다.



5 File 메뉴 아래에 빈 메뉴 항목을 선택합니다. 빈 메뉴 항목의 *Action* 속성을 `Action2`로 설정합니다. Exit 메뉴 항목이 File 메뉴 아래에 나타납니다.

6 File 오른쪽에 두 번째 최상위 메뉴 항목을 클릭합니다. *Caption* 속성을 `&Edit`로 설정하고 *Enter*를 누릅니다. Edit 메뉴 항목 아래에 나타나는 빈 메뉴 항목을 선택합니다.

7 Object Inspector에서 *Action* 속성을 `EditCut1`로 설정하고 *Enter*를 누릅니다. 자동으로 항목의 캡션이 `Cut`으로 설정되고 잘라내기 기본 비트맵이 메뉴에 나타납니다.

8 *Cut* 메뉴 아래의 다음 빈 메뉴 항목을 선택하여 *Action* 속성을 `EditCopy1`로 설정합니다(복사 기본 비트맵이 메뉴에 나타납니다).

9 *Copy* 아래의 다음 빈 메뉴 항목을 선택하여 *Action* 속성을 `EditPaste1`로 설정합니다(붙여넣기 기본 비트맵이 메뉴에 나타납니다).

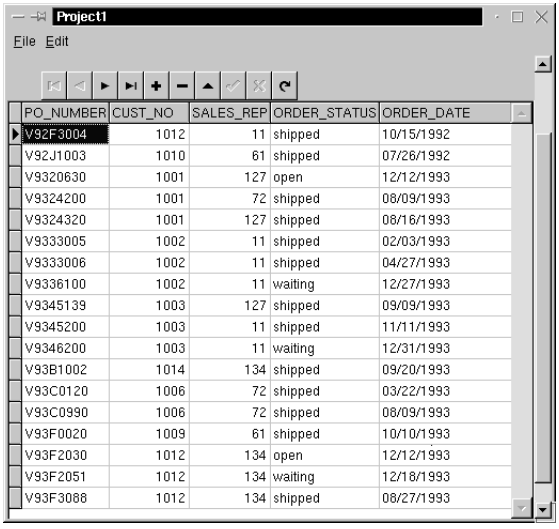
10 *Paste* 아래의 다음 빈 메뉴 항목을 선택하고 *Caption* 속성을 하이픈(-)으로 설정하여 메뉴에 구분자 표시줄을 만듭니다. *Enter* 키를 누릅니다.

11 구분자 표시줄 아래의 다음 빈 메뉴 항목을 선택하여 *Action* 속성을 `Action1`로 설정합니다. 메뉴 항목에 Update Now!가 표시됩니다.

12 **X**를 클릭하여 메뉴 디자이너를 닫습니다.

13 File|Save All을 선택하여 프로젝트를 저장합니다.

14 *F9*를 누르거나 툴바에서 Run을 클릭하여 애플리케이션이 실행된 외관을 살펴봅니다.



이제 Edit 메뉴와 탐색 막대에 있는 명령이 대부분 작동됩니다. Edit 메뉴의 Copy와 Cut은 데이터베이스에서 텍스트를 선택해야 활성화됩니다. 탐색 막대를 사용하여 데이터베이스 내의 레코드 간에 이동하거나 레코드를 삽입하고 지울 수 있습니다. Update 명령은 아직 작동하지 않습니다.

준비가 끝나면 애플리케이션을 닫습니다.

버튼 추가

이 단원에서는 Update Now 버튼을 애플리케이션에 추가하는 방법에 대해 설명합니다. 이 버튼을 클릭하면 레코드 편집, 새 레코드 추가, 레코드 삭제와 같은 사용자가 편집한 내용이 데이터베이스에 적용됩니다.

다음과 같은 방법으로 버튼을 추가합니다.

1 컴포넌트 팔레트의 Standard 페이지에서 *TButton* 컴포넌트를 폼에 가져다 놓습니다(컴포넌트를 클릭한 다음 폼의 탐색 막대 옆을 클릭하면 됩니다).

2 버튼의 *Action* 속성을 `Action1`로 설정합니다.

버튼의 캡션이 Update Now!로 변경됩니다. 애플리케이션을 실행하면 이벤트 핸들러를 추가해야 이 버튼이 활성화됩니다.

제목과 이미지 표시

회사 이름과 이미지를 애플리케이션 외관에 추가하면 좀더 전문적인 느낌을 줄 수 있습니다.

1 컴포넌트 팔레트의 Standard 페이지에서 *TLabel* 컴포넌트를 폼에 가져다 놓습니다. Kylix는 기본적으로 이 컴포넌트 이름을 *Label1*로 지정합니다.

2 Object Inspector에서 레이블의 *Caption* 속성을 `World Corp` 또는 다른 회사 이름으로 변경합니다.

3 *Font* 속성을 클릭하여 회사 이름에 적용된 글꼴을 변경합니다. 오른쪽에 있는 생략 기호를 클릭하고 Font 대화 상자에서 글꼴을 Helvetica Bold, 16pt로 변경합니다. OK를 클릭합니다.

4 레이블을 오른쪽 위 모서리에 놓아 둡니다.

5 컴포넌트 팔레트의 Additional 페이지에서 *TImage* 컴포넌트를 레이블 옆에 가져다 놓습니다. Kylix는 기본적으로 컴포넌트 이름을 *Image1*로 지정합니다.

6 *Image1* 컴포넌트에 이미지를 추가하기 위해 *Picture* 속성을 클릭합니다. 생략 기호를 클릭하여 Picture editor를 엽니다.

7 Picture editor에서 Load를 선택하여 기본 제공 아이콘 디렉토리를 탐색합니다. 기본 위치는 {install directory}/images/icons입니다. 예를 들어 Kylix가 /usr/local/kylix2 디렉토리에 설치되어 있으면 /usr/local/kylix2/images/icons를 찾아봅니다.

8 earth.ico을 더블 클릭합니다. OK를 클릭하여 그림을 로드하고 Picture editor를 닫습니다.

9 기본 이미지 영역을 그림 크기로 조정합니다. 이미지를 레이블 옆에 둡니다.

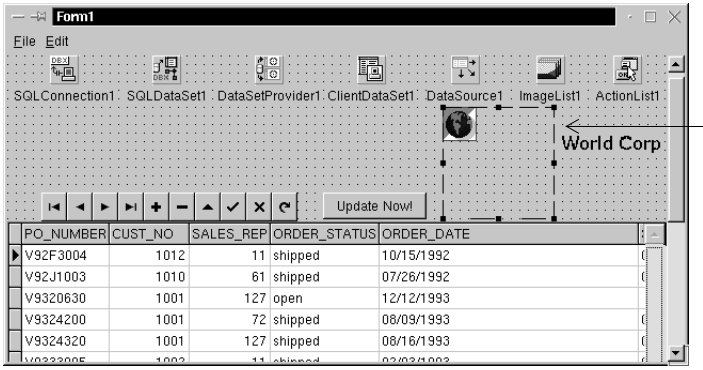


Image1 컴포넌트 크기가 그림과 일치하도록 크기를 변경하려면 *Image1*의 가장자리를 끌거나 Object Inspector의 *Width* 및 *Height* 속성을 변경합니다.

10 폼에서 텍스트와 이미지를 선택하고 마우스 오른쪽 버튼을 클릭한 다음 Align을 선택하여 두 객체를 정렬합니다. Alignment 대화 상자에서 Vertical 아래의 Bottoms를 클릭합니다.

11 File|Save All을 선택하여 프로젝트를 저장합니다.

12 *F9*를 눌러 애플리케이션을 컴파일하고 실행합니다.

13 준비가 끝나면 애플리케이션을 닫습니다.

이벤트 핸들러 작성

컴포넌트 팔레트에 있는 컴포넌트에는 대개 이벤트가 있으며 대부분 기본 이벤트를 포함합니다. 일반적인 기본 이벤트는 *OnClick*이며 *TButton*과 같은 컴포넌트를 클릭하면 실행됩니다. 폼에서 컴포넌트를 선택하여 Object Inspector의 Events 탭을 클릭하면 선택한 컴포넌트의 이벤트 목록이 나타납니다.

이벤트와 이벤트 핸들러에 대한 자세한 내용은 *Kylix1 개발자 안내서*의 6장 "애플리케이션 사용자 인터페이스 개발"이나 온라인 도움말을 참조하십시오.

Update Now! 명령의 이벤트 핸들러 작성

먼저 Update Now! 명령과 버튼에 대한 이벤트 핸들러를 작성합니다.

1 *ActionList* 컴포넌트를 더블 클릭하여 Action List 에디터를 표시합니다.

2 (No Category)를 선택하여 Action1과 Action2를 살펴봅니다.

3 Action1을 더블 클릭합니다. 코드 에디터에 다음과 같은 뼈대 이벤트 핸들러가 나타납니다.

```
procedure TForm1.Action1Execute(Sender: TObject);
begin

end;
```

begin과 **end** 사이의 커서가 위치한 곳에서 다음과 같은 코드를 입력합니다.

```
if ClientDataSet1.State in [dsEdit, dsInsert] then ClientDataSet1.Post;
ClientDataSet1.ApplyUpdates(-1);
```

여기서 이벤트 핸들러는 데이터베이스가 어떤 상태인지를 먼저 확인합니다. 사용자가 변경한 레코드를 떠나면 이 레코드는 자동으로 포스트됩니다. 그러나 변경한 레코드를 떠나지 않으면 데이터베이스는 편집이나 삽입 모드로 남아 있게 됩니다. **if** 문은 변경되었지만 클라이언트 데이터셋으로 전달되지 않은 모든 데이터를 포스트합니다. 다음 구문은 클라이언트 데이터셋에 있는 업데이트 내용을 데이터베이스에 적용합니다.

참고 dbExpress를 사용하면 데이터에 대한 변경된 내용은 자동으로 데이터베이스에 포스트되지 않습니다. 클라이언트 데이터셋으로부터 업데이트, 삽입, 삭제된 레코드를 데이터베이스에 모두 적용하려면 *ApplyUpdates* 메소드를 실행해야 합니다.

Exit 명령의 이벤트 핸들러 작성

이제 다음과 같은 방법으로 Exit 명령에 대한 이벤트 핸들러를 작성합니다.

1 *ActionList* 컴포넌트를 더블 클릭하여 Action List 에디터를 표시합니다(표시되지 않은 경우).

2 (No Category)를 클릭하여 Action2를 살펴봅니다.

3 Action2를 더블 클릭합니다. 코드 에디터에 다음과 같은 뼈대 이벤트 핸들러가 표시됩니다.

```
procedure TForm1.Action2Execute(Sender: TObject);
begin

end;
```

begin과 **end** 사이의 커서가 위치한 곳에 다음과 같은 코드를 입력합니다.

```
Close;
```

여기서 이벤트 핸들러는 메뉴에서 File|Exit 명령이 실행되면 애플리케이션을 닫습니다.

4 Action List 에디터를 닫습니다.

5 File|Save All을 선택하여 프로젝트를 저장합니다.

FormClose 이벤트 핸들러 작성

마지막으로 애플리케이션을 닫을 때 실행되는 이벤트 핸들러를 작성합니다. File|Exit를 누르거나 오른쪽 위 모서리의 **X**를 클릭하면 애플리케이션이 종료됩니다. 그러면 애플리케이션은 종료 방식에 상관없이 데이터베이스에 보류 중인 업데이트 내용이 있는지 확인하고 보류 중인 업데이트 내용이 있으면 사용자에게 어떻게 처리할지 묻는 메시지 창을 표시합니다.

이러한 코드를 Exit 이벤트 핸들러에 추가 할 수 있지만 **X**를 클릭하여 애플리케이션을 종료했다면 보류 중인 데이터베이스 변경 내용은 잃어버리게 됩니다.

1 폼에 있는 객체를 클릭하지 말고 메인 폼을 클릭하여 선택합니다.

2 Object Inspector에서 Events 탭을 선택하여 폼 이벤트를 살펴봅니다.

3 *OnClose*를 더블 클릭하거나 *OnClose* 이벤트 옆에 FormClose를 입력하고 이를 클릭합니다. 코드 에디터에서 다른 이벤트 핸들러 뒤에 다음과 같은 뼈대 FormClose 이벤트 핸들러가 작성되고 표시됩니다.

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin

end;
```

begin과 end 사이의 커서가 위치한 곳에 다음과 같은 코드를 입력합니다.

```
Action := caFree;
  if ClientDataSet1.State in [dsEdit, dsInsert] then
    ClientDataSet1.Post;
  if ClientDataSet1.ChangeCount> 0 then
  begin
    Option := Application.MessageBox('You have pending updates. Do you want to write them
     to the database?', 'Pending Updates',[smbYes, smbNo, smbCancel],
      smsWarning, smbYes);
    case Option of
      smbYes: ClientDataSet1.ApplyUpdates(-1);
      smbCancel: Action := caNone;
    end;
  end;
```

여기서 이벤트 핸들러는 데이터베이스의 상태를 확인합니다. 변경 내용이 보류 중이면 변경 내용이 클라이언트 데이터셋에 포스트되고 변경 카운트가 늘어납니다. 그런 다음 애플리케이션을 종료하기 전에 변경 내용을 어떻게 처리할지 묻는 메시지 상자를 표시합니다. 응답 옵션은 Yes, No, Cancel입니다. Yes를 선택하면 업데이트 내용을 데이터베이스에 적용합니다. No를 선택하면 데이터베이스를 변경하지 않고 애플리케이션을 종료합니다. Cancel을 선택하면 애플리케이션 종료는 취소하지만 변경 내용은 취소하지 않고 애플리케이션을 실행 상태로 남겨 둡니다.

4 프로시저에서 사용하는 변수를 선언해야 합니다. procedure와 begin 사이에 다음과 같은 코드를 입력합니다.

```
var
   Option: TMessageButton;
```

5 전체 프로시저가 다음과 유사한지 확인합니다.

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var
   Option: TMessageButton;
begin
Action := caFree;
  if ClientDataSet1.State in [dsEdit, dsInsert] then
    ClientDataSet1.Post;
  if ClientDataSet1.ChangeCount> 0 then
  begin
    Option := Application.MessageBox('You have pending updates. Do you want to write them
      to the database?', 'Pending Updates',[smbYes, smbNo, smbCancel],
      smsWarning, smbYes);
    case Option of
      smbYes: ClientDataSet1.ApplyUpdates(-1);
      smbCancel: Action := caNone;
    end;
  end;
end;
```

6 File|Save All을 선택하여 프로젝트를 저장합니다. 애플리케이션을 실행하려면 *F9* 키를 누릅니다.

팁 발생한 오류를 수정하려면 오류 메시지를 더블 클릭하여 문제가 발생한 코드로 이동하거나 F1을 눌러 메시지에 대한 도움말을 살펴보십시오.

이제 모두 끝났습니다. 애플리케이션을 실행하여 어떻게 구동하는지 확인해 볼 수 있습니다. 프로그램을 종료하려면 File|Exit 명령을 사용합니다.

6

데스크탑 사용자 지정

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

작업 영역 구성

IDE는 개발을 지원하는 툴을 많이 제공하므로 메뉴 및 툴바 재정렬, 툴 윈도우 도킹, 새로운 데스크탑의 저장 등을 비롯하여 작업 영역을 최대한 편리하게 재구성할 수 있습니다.

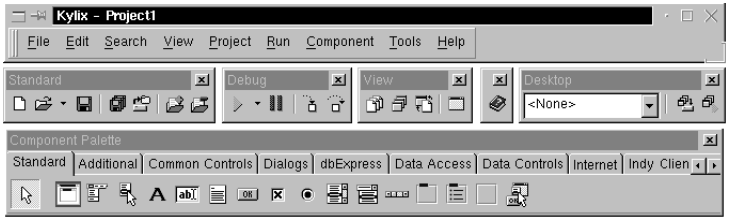
메뉴와 툴바 정렬

메인 윈도우에서는 메뉴, 툴바, 컴포넌트 팔레트 등의 왼쪽 그래버를 클릭한 다음 다른 위치로 끌어 놓아서 이를 재구성할 수 있습니다.

메인 윈도우 내에서 툴바와 메뉴를 이동시킬 수 있습니다. 옮기려는 툴바의 그래버 (왼쪽에 있는 두 줄 막대)를 끕니다.

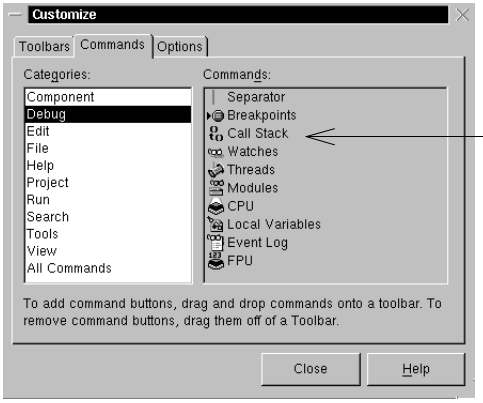
다르게 구성한 메인 윈도우

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



툴바를 옮기려면 각 툴바의 그래버를 끌어 놓습니다.

View|Toolbars|Customize를 선택하면 툴바에 툴을 추가하거나 삭제할 수 있습니다.



Commands 페이지에서 명령을 선택하고 툴바로 끌어 놓으십시오.

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

자세한 내용은...

도움말 색인에서 "toolbars, customizing"을 참조하십시오.

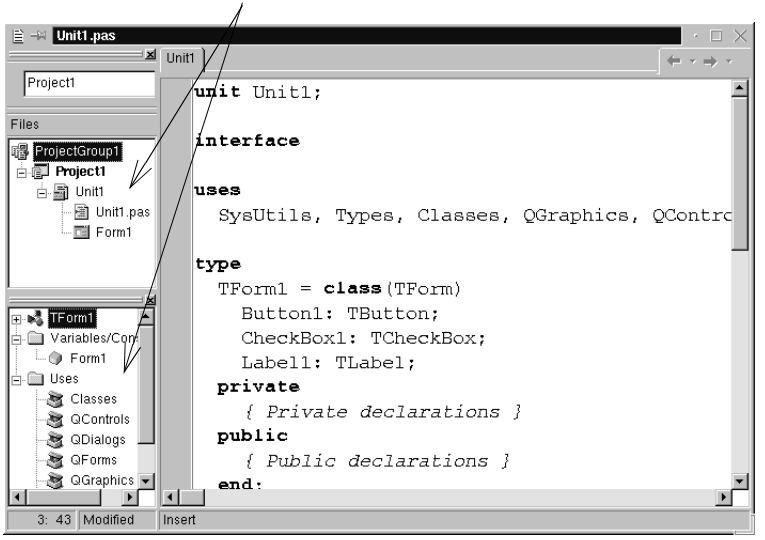
툴 윈도우 도킹

툴 윈도우는 개별적으로 열고 닫을 수 있으며 데스크탑에 보기 좋게 정렬할 수도 있습니다. 또한 관리를 용이하게 하기 위해 창들을 *도킹*할 수 있습니다. 도킹, 즉 창들을 서로 붙여서 함께 움직이도록 만들어 놓으면 툴에 빠르게 액세스할 수 있으며 화면을 효율적으로 사용할 수 있습니다.

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

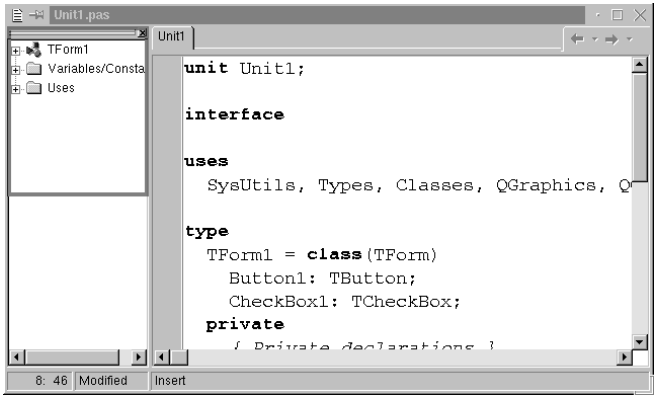
Project Manager 및 코드 탐색기가 이와 같이 코드 에디터에 도킹되어 있습니다.

창을 왼쪽 그래버나 6-4페이지의 탭에 결합하거나 "도킹"할 수 있습니다.

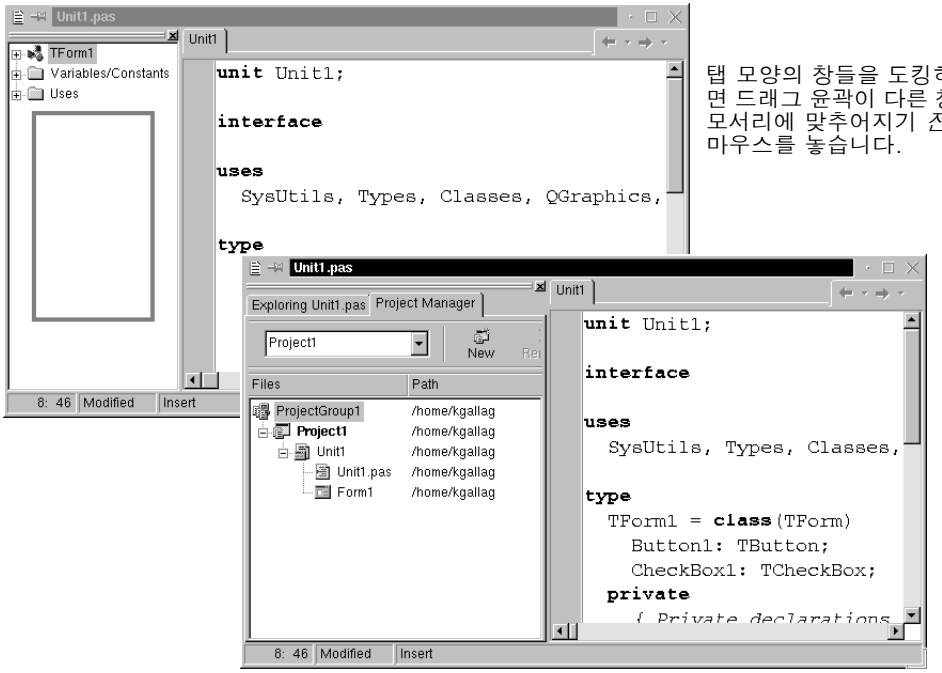


창을 도킹하려면 제목 표시줄을 클릭하여 다른 창으로 끌어 놓습니다. 드래그 윤곽이 직사각형으로 좁아지고 모서리와 맞추어지면 마우스를 놓습니다. 그러면 두 창이 함께 도킹됩니다.

그래버를사용하여 창을도킹시키려면 드래그 윤곽이 창의 모서리와 맞았을 때 마우스를 놓습니다.



도구들을 도킹시켜 탭 모양의 창을 만들 수도 있습니다.



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

창의 도킹을 해제하려면 그래버를 더블 클릭하거나 탭을 더블 클릭합니다.

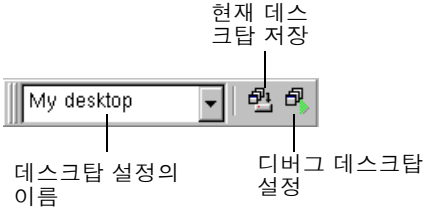
자동 도킹을 해제하려면 화면에 창을 옮길 때 *Ctrl* 키를 누릅니다.

자세한 내용은...

도움말 색인의 "docking"을 참조하십시오.

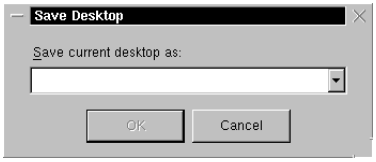
데스크탑 레이아웃 저장

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



특정한 창을 표시, 크기 조정, 도킹하거나 창을 화면에서 원하는 위치에 놓는 등 데스크탑을 보기 좋게 정렬합니다. Desktops 툴바에서 Save current desktop 아이콘을 클릭

하거나 View|Desktops|Save Desktop을 선택하고 새로운 레이아웃 이름을 입력합니다.



저장하려는 데스크탑 레이아웃 이름을 입력하고 OK를 클릭합니다.

자세한 내용은...

도움말 색인의 "desktop layout"을 참조하십시오.

컴포넌트 팔레트 사용자 지정

기본 구성에서 컴포넌트 팔레트는 기능별로 구성된 여러 개의 유용한 CLX 객체를 탭 모양의 페이지에 표시합니다. 다음과 같은 방법으로 컴포넌트 팔레트를 사용자 지정합니다.

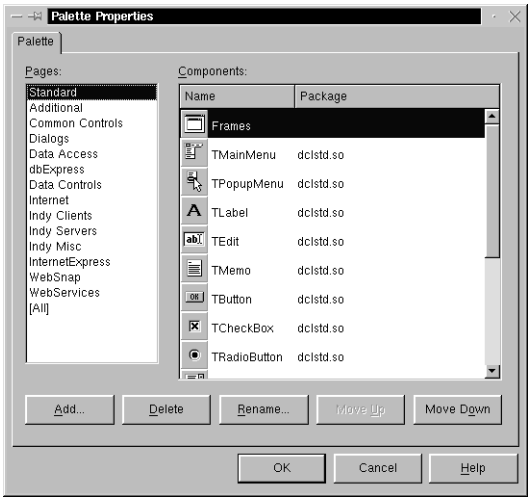
- 컴포넌트를 숨기거나 재정렬합니다.
- 페이지를 추가, 삭제, 재정렬하거나 이름을 변경합니다.
- 컴포넌트 템플릿을 작성한 다음 팔레트에 추가합니다.
- 새로운 컴포넌트를 설치합니다.

컴포넌트 팔레트 정렬

페이지를 추가, 삭제, 재정렬하거나 페이지 이름을 변경하거나 컴포넌트를 숨기거나 재정렬하려면 Palette Properties 대화 상자를 사용합니다. 이 대화 상자를 다음과 같은 여러 가지 방법으로 열 수 있습니다.

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

- 컴포넌트 팔레트를 마우스 오른쪽 버튼으로 클릭하고 Properties를 선택합니다.



팔레트를 재정렬하고 새로운 페이지를 추가합니다.

자세한 내용은...

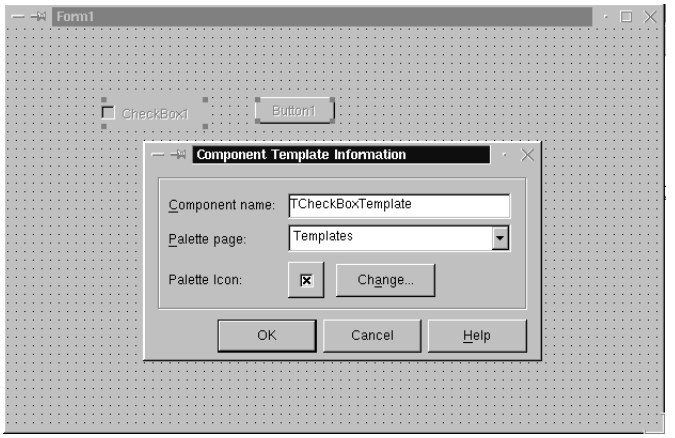
Palette Properties 대화 상자에서 Help 버튼을 클릭하십시오.

컴포넌트 템플릿 생성

컴포넌트 템플릿은 단 한 번의 작업으로 폼에 추가되는 컴포넌트 그룹입니다. 템플릿을 사용하면 하나의 폼에서 컴포넌트를 구성한 다음 컴포넌트 정렬, 기본 속성, 이벤트 핸들러를 컴포넌트 팔레트에 저장하면 다른 폼에서 재사용할 수 있습니다.

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

템플릿을 폼에 놓은 다음에는 컴포넌트들을 각기 따로 다시 배치하고, 컴포넌트의 속성을 재설정하고, 다른 작업에서 각 컴포넌트를 놓은 것처럼 컴포넌트의 이벤트 핸들러를 작성하거나 수정할 수 있습니다.



자세한 내용은...

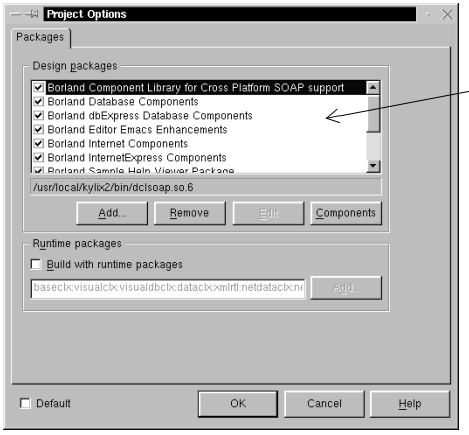
도움말 색인의 "templates, component"를 참조하십시오.

컴포넌트 패키지 설치

사용자 지정 컴포넌트를 직접 작성했거나 공급 업체로부터 구입했으면 컴포넌트 팔레트에 설치하기 전에 컴포넌트를 컴파일하여 *패키지*로 만듭니다.

패키지는 Kylix 애플리케이션과 IDE 사이에서 공유할 수 있는 코드가 들어 있는 특별한 공유 객체입니다. *런타임 패키지*는 사용자가 애플리케이션을 실행할 때 사용되는 기능을 제공합니다. *디자인 타임 패키지*는 IDE에 컴포넌트를 설치하는 데 사용합니다.

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



이러한 컴포넌트들은 Kylix에 이미 설치되어 있습니다. 협력 업체의 새 컴포넌트를 설치할 경우 패키지가 이 목록에 나타납니다.

패키지에 포함되어 있는 컴포넌트들을 알아보려면 Components를 클릭합니다.

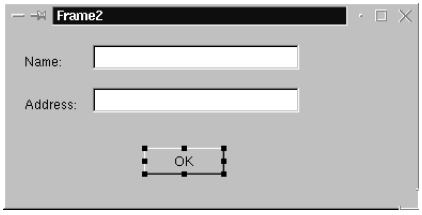
자세한 내용은...

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

프레임 사용

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

새 프레임을 열려면 File|New Frame을 선택합니다.



비주얼 컴포넌트나 넌비주얼 컴포넌트에 상관없이 프레임에 추가할 수 있습니다. 새로운 유닛이 코드 에디터에 자동으로 추가됩니다.

자세한 내용은...

도움말 색인의 "frames"와 "TFrame"을 참조하십시오.

프로젝트 옵션 설정

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

기본 프로젝트 옵션 설정

선택한 내용을 새 프로젝트에 대한 기본 설정으로 저장하려면 Project Options 대화 상자의 왼쪽 아래 모서리에 있는 Default에 선택 표시를 합니다. Default에 선택 표시를 하면 대화 상자의 현재 설정을 옵션 파일 defproj.kof에 기록합니다. Kylix 본래의 기본 설정을 복구하려면 {home directory}/.borland에 있는 defproj.kof 파일을 삭제하거나 이름을 변경합니다.

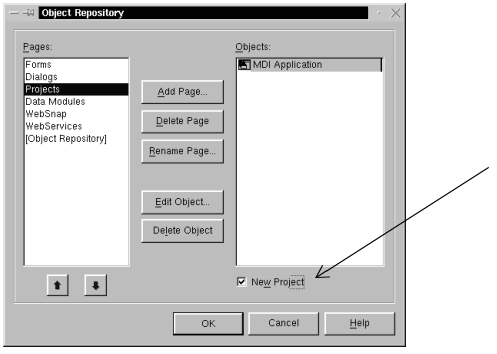
자세한 내용은...

도움말 색인의 "Project Options dialog box"를 참조하십시오.

프로젝트와 폼 템플릿을 기본값으로 지정

File|New Application을 선택하면 새 프로젝트가 IDE에서 열립니다. 프로젝트 *템플릿*을 *기본* 프로젝트로 지정하지 않으면 Kylix는 빈 폼을 가진 새로운 표준 애플리케이션을 생성합니다. Kylix에는 미리 디자인된 프로젝트 템플릿이 들어 있지 않지만 Project|Add to Repository를 선택함으로써 직접 작성한 프로젝트를 Projects 페이지에 있는 Object Repository에서 템플릿으로 저장할 수 있습니다. 6-9페이지를 참조하십시오.

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



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

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

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

일단 프로젝트 템플릿을 기본값으로 설정하면 Kylix는 File|New Application을 선택할 때마다 프로젝트 템플릿을 자동으로 엽니다.

기본 프로젝트를 지정하는 것과 같은 방법으로 Object Repository의 기존 폼 템플릿 목록에서 *기본 메인 폼*과 *기본 새 폼*을 지정할 수 있습니다. 기본 메인 폼은 새 애플리케이션을 열 때 생성되는 폼입니다. 기본 새 폼은 열린 프로젝트에 폼을 추가하기 위해 File|New Form을 선택하면 생성되는 폼입니다. 기본 폼을 지정하지 않을 경우 Kylix는 빈 폼을 사용합니다.

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

자세한 내용은...

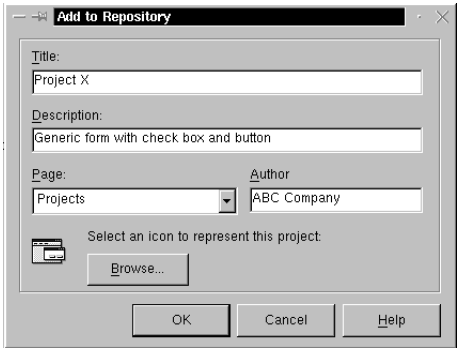
도움말 색인의 "templates, adding to Object Repository", "projects, specifying default", "forms, specifying default"를 참조하십시오.

Object Repository에 템플릿 추가

직접 만든 객체를 *템플릿* 형식으로 Object Repository에 추가하여 재사용하거나 네트워크 상에서 다른 개발자와 공유할 수 있습니다. 객체를 재사용하면 공통 사용자 인터페

이스와 기능이 포함된 애플리케이션 제품군을 만듦으로써 개발 시간을 단축하고 품질을 개선할 수 있습니다.

예를 들어 프로젝트를 템플릿 형식으로 Repository에 추가하려면 먼저 프로젝트를 저장하고 Project|Add To Repository를 선택한 다음 Add to Repository 대화 상자에 정보를 입력합니다.



제목, 설명, 작성자를 입력합니다. Page 리스트 박스에서 Projects를 선택하여 직접 만든 프로젝트가 Repository의 Projects 페이지에 나타나도록 합니다.

다음에 New Items 대화 상자를 열면 프로젝트 템플릿은 Projects 페이지나 템플릿을 저장한 페이지에 나타납니다. Kylix를 열 때마다 특정 템플릿을 기본으로 설정하려면 6-9페이지의 "프로젝트와 폼 템플릿을 기본값으로 지정"을 참조하십시오.

자세한 내용은...

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

툴 환경 설정

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

자세한 내용은...

도움말 색인의 "Environment Options dialog box"를 참조하거나 Environment Options 대화 상자에서 Help 버튼을 클릭합니다.

폼 디자이너 사용자 지정

Environment Options 대화 상자의 Preferences 페이지 설정은 폼 디자이너에 영향을 줍니다. 예를 들면 여러 개의 컴포넌트를 가장 가까운 그리드 라인(grid line)에 정렬하는 "snap to grid" 기능을 사용 가능 또는 사용 불가능으로 설정할 수 있습니다. 폼에 놓은 넌비주얼 컴포넌트의 이름이나 *캡션*을 표시하거나 숨길 수도 있습니다.

자세한 내용은...

Environment Options 대화 상자에서 Preferences 페이지의 Help 버튼을 클릭합니다.

코드 에디터 사용자 지정

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

또한 Editor Options의 Code Insight에 있는 에디터 안에서 사용할 수 있는 Code Insight 툴을 구성할 수 있습니다. 이러한 도구들에 대해 알아보려면 2-6페이지의 "Code Insight 툴"을 참조하십시오.

자세한 내용은...

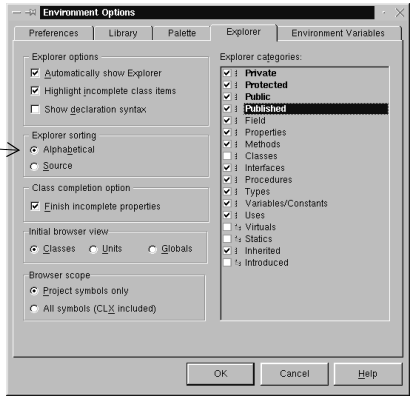
Editor Options 대화 상자에서 General, Display, Key Mappings, Color, Code Insight 페이지 등에 있는 Help 버튼을 클릭합니다.

코드 탐색기 사용자 지정

Kylix를 시작할 때 코드 탐색기(2-8페이지의 "코드 탐색기"에서 설명)는 구입한 Kylix 에디션에 코드 탐색기가 들어 있을 경우 자동으로 열립니다. 코드 탐색기가 자동으로 열리지 않게 하려면 Tools|Environment Options를 선택하고 Explorer 탭을 클릭한 다음 Automatically show Explorer를 선택 해제합니다.

코드 탐색기에서 마우스 오른쪽 버튼을 클릭하거나, Properties를 선택하거나, Explorer categories에서 체크 박스를 선택하여 코드 탐색기의 내용을 그룹화하는 방법을 바꿀 수 있습니다. 범주를 선택하면 해당 범주에 있는 요소가 단일 노드로 그룹화되며 범주를 선택 해제하면 범주의 각 요소들이 별도의 다이어그램에 표시됩니다. 예를 들어 Published 범주를 선택 해제하면 Published 범주 폴더는 없어지지만 그 안에 있는 항목은 없어지지 않습니다.

코드 탐색기에 있는 모든 소스 요소를 알파벳 순이나 소스 파일에 선언된 순서로 정렬할 수 있습니다.



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

자세한 내용은...

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

도움말 항목 인쇄

이 단원에서는 Kylix Help 항목을 인쇄하고 Kylix에서 프린터 설정을 구성하는 방법을 다룹니다. 두 가지 방법으로 Kylix Help를 인쇄할 수 있습니다. 항목을 PostScript 파일로 인쇄하거나 프린터로 직접 인쇄되도록 시스템을 구성할 수 있습니다.

참고 Linux 시스템은 직접 인쇄하도록 구성해야 합니다. 직접 인쇄하도록 Linux 시스템을 구성하려면 /etc/printcap 파일을 편집하십시오. 이 파일을 구성하려면 배포 관련(distribution-specific) 설명서인 Printing-HOWTO 파일을 확인하거나 http://www.linuxdoc.org에서 Printing-Usage-HOWTO 파일을 확인해 보십시오.

파일로 인쇄

Kylix Help 시스템은 기본적으로 PostScript 파일로 인쇄하도록 구성되어 있습니다. 도움말 항목을 PostScript 파일로 인쇄한 후 이 파일을 프린터에서 다음과 같은 방법으로 인쇄할 수 있습니다.

1 Help를 선택하고 Kylix Help에서 Help 메뉴 항목을 선택한 다음 Print를 클릭합니다. Topics 대화 상자가 나타납니다.

2 인쇄하려는 항목을 선택한 다음 OK를 클릭합니다.

Kylix Help 시스템은 사용자의 홈 디렉토리에서 PostScript 파일을 만듭니다. 항목이 파일로 인쇄되면 'Output is in /home/username/xprinter.out'이라는 메시지를 출력하는 메시지 박스가 나타납니다.

3 셸에서 이 파일을 PostScript가 지원되는 프린터로 인쇄하려면 홈 디렉토리로 이동한 다음 `lpr xprinter.out`을 입력합니다.

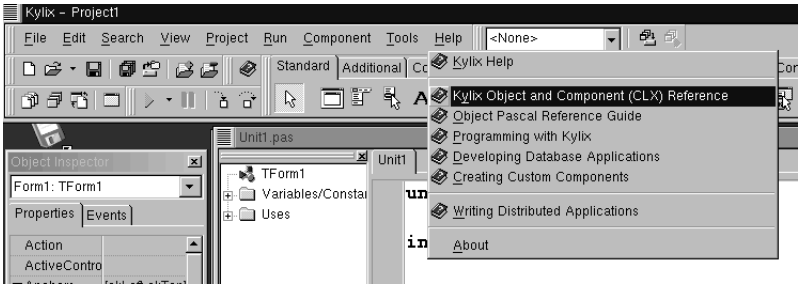
PostScript를 지원하지 않는 프린터에서는 Ghostview와 같은 유틸리티를 사용하여 xprinter.out 파일을 인쇄할 수 있습니다.

참고 파일로 인쇄할 때 Kylix는 기존 xprinter.out 파일을 겹쳐씁니다. 프린터로 인쇄하기 전에 여러 개의 항목을 파일로 인쇄하려면 항목을 파일로 인쇄할 때마다 xprinter.out 파일의 이름을 변경하십시오.

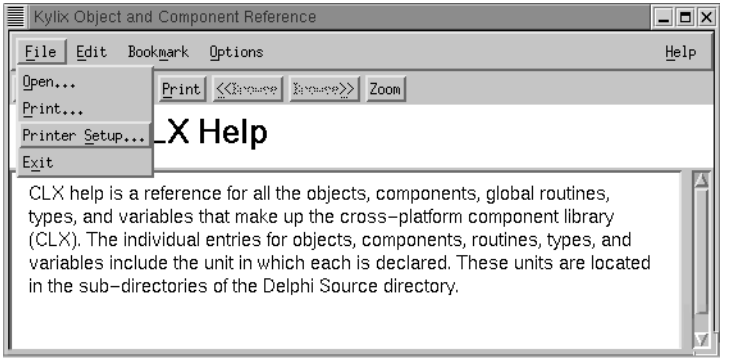
프린터로 직접 인쇄

파일로 인쇄하는 대신 프린터로 직접 인쇄하도록 시스템을 구성할 수 있습니다.

1 Help를 선택하고 Kylix Help에서 Help 메뉴 항목을 선택하여 도움말 시스템을 엽니다.



2 도움말 시스템에서 File|Printer Setup을 선택합니다.



Printer Setup 대화 상자가 나타납니다.

3 시스템에 특정 프린터를 추가하려면 Printer Specific 체크 상자를 선택하고 Install 버튼을 클릭합니다.

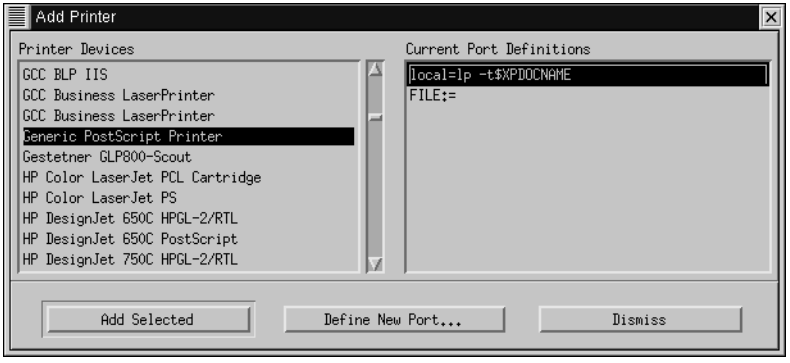
4 Printer Installation 대화 상자에서 Add Printer를 클릭합니다. Add Printer 대화 상자가 나타납니다.

5 Printer Devices 아래에 있는 목록에서 사용할 프린터를 선택합니다. 목록에는 없지만 PostScript 인쇄를 지원하는 프린터라면 Generic PostScript Printer를 선택합니다.

6 Current Port Definitions 아래에서 local=lp −t$XPDOCNAME을 선택합니다.

참고 Linux 배포판에 포함된 인쇄 서비스 소프트웨어에 따라 Current Port Definitions에 있는 명령 구문을 변경해야 할 수도 있습니다. 예를 들어 Red Hat 7.0 이상 버전은 LPRng(이 소프트웨어는 위의 구문을 지원함)를 사용하는 반면 이전 버전의 Red Hat은 포트 정의를 개인이 직접 local=lpr −T$XPDOCNAME으로 변경해야 하는 lpr 버전을 포함하고 있습니다.

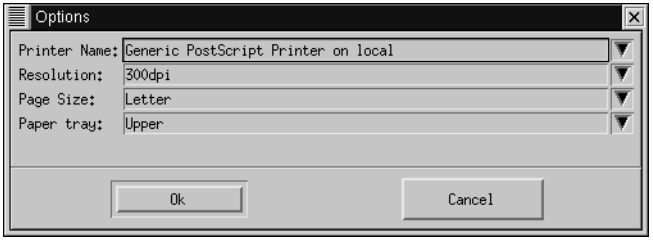
팁 정확한 구문을 알고 싶으면 사용 중인 시스템에서 lp 및 lpr에 대한 man 페이지를 확인해 보십시오.



7 정확한 프린터 장치를 선택했으면 Add Selected를 클릭한 다음 Dismiss를 클릭합니다. 이렇게 하면 인쇄 명령이 올바른 프린터로 연결됩니다.

8 Dismiss를 한 번 더 클릭합니다.

9 Printer Setup 대화 상자에서 Options를 클릭하여 Options 대화 상자를 엽니다.



10 Printer Name 필드에서 새로 구성한 로컬 프린터를 선택하고 OK를 클릭합니다.

11 Printer Setup 대화 상자에서 Apply를 클릭합니다.

이제 프린터로 직접 인쇄할 준비가 되었습니다.

12 Kylix Help에서 Help 메뉴 항목을 선택한 다음 Print를 클릭합니다. Topics 대화 상자가 나타납니다.

13 인쇄하려는 항목을 선택한 다음 OK를 클릭합니다.

색인

가

개발자 지원 1-6
객체
  정의 3-4, 4-3
  폼에 추가 4-4
공유 객체 3-10
기본
  프로젝트 옵션 6-8
  프로젝트와 폼 템플릿 6-9
기술 지원 1-6

나

뉴스그룹 1-6

다

다계층 애플리케이션, 생성 3-8
대화 상자, 템플릿 2-4
데스크탑
  구성 6-1 ~ 6-5
  레이아웃, 저장 6-4
데이터 모듈 2-4, 3-5
데이터베이스 드라이버 3-8
데이터베이스 애플리케이션
  개요 3-8, 5-1
  생성 5-1 ~ 5-15
  액세스 5-3 ~ 5-4
데이터베이스 자습서 5-1 ~ 5-15
도움말 시스템
  액세스 1-5 ~ 1-6
  인쇄 6-12
도움말 툴팁 4-3
디자인 타임 뷰 4-2

라

리소스 파일(.res) 4-2

마

마법사(Object Repository) 2-4
마우스 오른쪽 버튼 메뉴 2-2
메뉴
  구성 2-2 ~ 2-3, 6-1
  애플리케이션에 추가 4-12
  컨텍스트 2-2
  Kylix의 메뉴 2-2
메시지, 오류 4-23, 4-25
모듈, 데이터 3-5
문자 집합, 확장 3-7

바

분산 애플리케이션 3-8
비트맵, 애플리케이션에 추가 4-10

사

사용자 인터페이스, 디자인 3-1, 4-3
사용자 지정
  컴포넌트 팔레트 2-2
  코드 에디터 6-11
  코드 탐색기 6-11
  폼 디자이너 6-10
  IDE 6-1 ~ 6-11
설명서, 주문 1-6
설치 1-3
  사용자 지정 컴포넌트 6-7
  Kylix 1-2 ~ 1-4
소스 코드
  작성 시 도움말 2-6
  파일 4-1
  CLX 3-5
속성 설정 3-2, 4-2, 4-7
시스템 요구 사양 1-2
실행 파일 3-7

아

애플리케이션
  국제화 3-7
  데이터베이스 3-8, 5-1
  배포 3-7
  사용자 인터페이스, 디자인 3-1
  생성 3-1
  웹 서버 3-9
  웹 서비스 3-9
  컴파일 및 디버깅 3-6, 4-14
  클라이언트/서버 3-8
애플리케이션 실행 3-6, 4-14, 5-7
애플리케이션 지역화 3-7
액션, 애플리케이션에 추가 4-7, 4-9
오류 메시지 4-23, 4-25
온라인 도움말
  액세스 1-5 ~ 1-6
  인쇄 6-12
옵션
  기본 프로젝트 6-8
  코드 에디터 6-11
  코드 탐색기 6-11
요구 사양, 시스템 1-2
웹 사이트, Borland 1-6

웹 서버 애플리케이션 3-9
웹 서비스 3-9
유닛 파일 4-1
이미지, 애플리케이션에 추가 4-10
이벤트 4-16, 5-12
이벤트 핸들러
  생성 4-16 ~ 4-23, 5-12 ~ 5-15
  정의 3-4

자

자습서
  데이터베이스 5-1 ~ 5-15
  텍스트 에디터 4-1 ~ 4-25
저장
  데스크탑 레이아웃 6-4
  프로젝트 4-1
전역 기호 2-10
지원 서비스 1-6

차

창 도킹 6-2, 6-2 ~ 6-4
초기화 파일, 프로그램 배포 3-7
추가
  폼에 컴포넌트 추가 4-3, 4-12
  Object Repository에 항목 추가 2-4

카

컨텍스트 메뉴, 액세스 2-2
컴포넌트
  사용자 지정 3-1, 6-6
  생성 3-10
  설치 3-1, 6-7
  속성 설정 3-2, 4-2
  컴포넌트 팔레트 정렬 6-5
  컴포넌트 팔레트에 추가 6-5
  폼에 추가 3-2, 4-3
  CLX 클래스 라이브러리 3-5
컴포넌트 팔레트
  사용 3-2
  사용자 지정 6-5 ~ 6-7
  사용자 지정 컴포넌트 추가 3-10
  생성 6-6
  정의 2-3
  페이지 추가 6-5
코드
  보기 및 편집 2-5
  작성 3-4, 4-16 ~ 4-23, 5-12 ~ 5-15
  작성 시 도움말 2-6
  코드 에디터 사용 2-5
코드 에디터
  다른 창과 결합 6-2
  사용 2-5 ~ 2-6
  사용자 지정 6-11
코드 작성, 개요 3-4
코드 탐색기
  사용 2-8
  사용자 지정 6-11
크로스 플랫폼용 컴포넌트 라이브러리(CLX)
  다이어그램 3-5
  정의 3-4
클라이언트/서버 애플리케이션, 생성 3-8
클래스, CLX 사용 3-4
키스트로크 매핑 6-11

타

텍스트 에디터 자습서 4-1 ~ 4-25
템플릿
  기본값으로 지정 6-9
  Repository에 추가 2-5, 6-9
통합 개발 환경(IDE)
  둘러 보기 2-1
  사용자 지정 6-1 ~ 6-11
통합 디버거 3-6
툴바
  구성 6-1
  사용 2-2 ~ 2-3
  애플리케이션에 추가 4-15
  컴포넌트 추가 및 삭제 6-2
툴팁, 보기 4-3

파

파일
  구성 4-2
  유닛 4-1
  저장 4-1
  타입 4-1, 4-2
  폼 2-8, 4-1
  프로젝트 4-1
패키지
  정의 6-7
  컴포넌트 설치 6-7
폼
  기본값으로 지정 6-9
  닫기 4-2
  메인 폼 4-2, 6-9
  컴포넌트 추가 3-1, 4-3
  Object Repository의 폼 2-4
폼 디자이너
  사용 2-3
  사용자 지정 6-10
  정의 2-1
폼 파일
  정의 4-1
  코드 보기 2-8
표기법 1-7
표준 액션, 애플리케이션에 추가 4-9
프레임 6-8
프로그램
  국제화 3-7
  데이터 모듈 추가 3-5

데이터베이스 3-8
배포 3-7
웹 서버 애플리케이션 3-9
웹 서비스 3-9
컴파일 및 디버깅 3-6, 4-14
코드 작성 3-4
프로그램 디버깅 3-6
프로그램 배포 3-7
프로그램 컴파일 3-6, 4-14, 5-7
프로젝트
공유 객체 3-10
관리 2-9
기본 파일 4-1
기본값으로 지정 6-9
사용자 지정 컴포넌트 3-10
생성 3-1
옵션을 기본값으로 설정 6-8
유형 3-8 ~ 3-10
저장 4-1
템플릿 추가 6-9
항목 추가 2-4, 2-5 ~
프로젝트 그룹 2-9

A

About 상자, 추가 4-23

B

BizSnap 3-9

C

Class Completion 2-7
CLX
컴포넌트 2-3
클래스 라이브러리 3-4 ~ 3-5
Code Insight 툴 2-6

D

.dpr 파일 4-1
DataSnap 3-8
dbExpress 3-8, 5-1

E

Editor Options 대화 상자 2-6, 6-11
Environment Options 대화 상자 2-7, 6-10
Events 페이지(Object Inspector) 3-4, 5-12

G

GUI, 만들기 3-1

I

IDE
구성 6-1
둘러 보기 2-1
사용자 지정 6-1 ~ 6-11
정의 1-1
IME 3-7

K

Kylix
개요 1-1
도움말 항목 인쇄 6-12
등록 1-4
사용자 지정 6-1 ~ 6-11
설치 1-2
설치 제거 1-7
시작 1-4
프로그래밍 3-1
Kylix 등록 1-4
Kylix 설치 제거 1-7
Kylix 시작 1-4
Kylix 프로그래밍, 개요 3-1
Kylix Help 인쇄 6-12

N

New Items 대화 상자
사용 2-4, 4-23
템플릿 저장 6-9, 6-10

O

Object Inspector 3-4
사용 3-2, 3-4, 4-2
정의 2-3
Object Repository
사용 2-4 ~ 2-5, 3-1
정의 2-4
템플릿 추가 2-5, 6-9

P

.pas 파일 4-1
Project Browser 2-10
Project Manager 2-9
Project Options 대화 상자 6-8

R

resbind, 지역화 3-7
Run 버튼 5-7

T

to-do list **2-10**

W

WebSnap 3-9

X

.xfm 파일 2-8, 4-1