Inside Mac OS X

Cocoa Objective-C 응용 프로그램 개발하기: 튜토리얼

1 장. 머릿말

이 튜토리얼에서는 Mac OS X 의 Cocoa 응용 프로그램 프레임웍을 소개하고, Apple 의 개발 툴과 Objective-C 언어로 강력한 객체 지향 응용 프로그램을 개발하는 방법에 대해 설명합니다. Cocoa 는 현대적이고, 풍부한 멀티미디어의 객체 지향 응용 프로그램을 개발하기 위한 최상의 방법을 제공합니다. 이 튜토리얼은 C 프로그래밍에 대한 지식을 기반으로 하지만, Cocoa, Project Builder 또는 Interface Builder 에 대한 사용자의 경험을 필요로 하지는 않습니다.

이 튜토리얼에서 배울 내용

이 튜토리얼에서는 달러 값을 환율에 따라 다른 통화값으로 변환해주는 간단한 응용 프로그램을 개발하는 방법을 보여줍니다. 완성된 프로그램은 다음과 같습니다.

그림 1-1 완성된 통화 변환기 프로그램

통화 변환기는 간단한 프로그램이지만 Cocoa 소프트웨어 개발에 관한 다양한 예시를 보여줍니다. 차츰 알게 되겠지만, 통화 변환기는 Cocoa 응용 프로그램으로 놀랍도록 손쉽게 제작할 수 있으며, 곁들여서 Cocoa 의 많은 기능을 습득할 수 있습니다.

텍스트 필드를 클릭하여 값을 입력하고, 탭 키를 사용하여 필드를 이동할 수 있습니다. Convert 버튼을 누르거나 리턴 키를 누르면 변환된 값이 호출됩니다. Cocoa 응용 프로그램 환경에서 상속된 특징 때문에, 변환된 값을 선택하고 이것을 복사(Edit 메뉴의 Copy 명령어로)하여 텍스트를 입력할 수 있는 다른 프로그램에 옮겨 붙일 수 있습니다.

이 튜토리얼은 Cocoa 응용 프로그램을 제작하기 위한 기본 절차를 안내할 것입니다. 다음과 같은 내용을 배우게 됩니다.

- Project Builder 프로젝트를 제작하는 방법
- Interface Builder 를 사용하여 그래픽 사용자 인터페이스를 제작하는 방법
- Cocoa 프레임웍 클래스의 커스텀 서브클래스를 제작하는 방법
- 커스텀 서브클래스를 인터페이스에 연결하는 방법

다음 절차를 따라서 하다 보면 응용 프로그램 개발에 사용되는 가장 중요한 두 가지 Cocoa 응용 프로그램인 Interface Builder, Project Builder 와 친숙해질 것입니다. 또한 Cocoa 응용 프로그램 개발의 일반적인 작업 흐름에 대해서도 배울 수 있을 것입니다.

1. 응용 프로그램 디자인
2. 프로젝트 제작 (Project Builder)
3. 인터페이스 제작 (Interface Builder)
4. 클래스 정의 (Interface Builder)
5. 클래스 구현 (Project Builder)
6. 프로젝트 구성 (Project Builder)
7. 응용 프로그램 실행 및 테스트

이와 같은 과정을 거쳐, 객체 지향 패러다임으로 프로그램을 설계하는 방법을 배울 수 있습니다. 객체 지향 프로그래밍과 Objective-C 에 대해 더 알고 싶으시면, *Inside Mac OS X: The Objective-C Programming Language* 를 참조하십시오.

이 튜토리얼은 순서대로 완성해야 하는 두 개의 장으로 구성되어 있습니다.

1. “통화 변환기 응용 프로그램 디자인”
2. “통화 변환기 응용 프로그램 개발”

마지막 장인 “Cocoa 리소스”에서는 Cocoa 및 Mac OS X 프로그래밍에 관한 지식을 한층 향상시킬 수 있습니다.

2 장. 통화 변환기 응용 프로그램 디자인

이 장에서는 Cocoa 에서 가장 일반적으로 사용되는 객체 지향 디자인 패러다임인 모델-뷰-컨트롤러에 대해 설명하고, 통화 변환기 응용 프로그램의 디자인을 통해 이를 설명합니다.

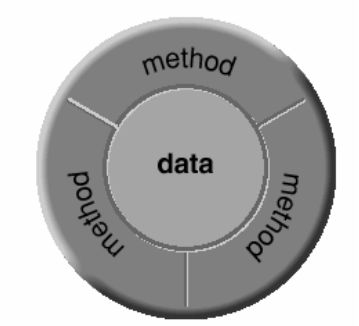
통화 변환기 응용 프로그램 디자인

객체 지향 프로그램은 프로그램의 객체를 식별하고 그 객체들의 기능과 책임을 명확히 정의하는 설계에 기반을 두어야 합니다. 대개 코드를 작성하기 전에 설계 작업부터 합니다. 프로그램 설계에 많은 도구가 필요하지는 않습니다. 노트와 연필만 있으면 충분합니다.

객체 표시법

객체 지향 프로그램을 디자인할 때에는 객체들간의 관계를 그림으로 나타내어 보는 것이 많은 도움이 됩니다. 이 튜토리얼에서는 객체를 그림 2-1 과 같이 나타냅니다.

그림 2-1 도넛 모양의 객체



도넛이나 구명 튜브를 연상시키는 이 그림은 객체의 근본적인 특징인 데이터 캡슐화를 보여줍니다. 하나의 객체는 데이터와 그 데이터를 조작하는

과정으로 구성됩니다. 다른 객체나 외부 코드가 그 데이터를 직접적으로 액세스할 수는 없으며, 객체에 데이터를 요청하는 메시지를 보내게 됩니다.

객체의 절차들은 메소드라고 불리우며, 메시지에 반응하고 요청한 객체에 대한 데이터를 반환합니다. 그림이 제시하는 것처럼 객체의 메소드는 데이터와 그 데이터와 관련된 메소드를 하나로 묶는 캡슐화를 통해 객체의 데이터에 효과적으로 접근하도록 해줍니다. 또한 객체의 메소드는 객체가 외부와 명확하게 소통하도록 해주는 인터페이스입니다.

또한 도넛형의 그림은 객체의 모듈화된 특성을 이해할 수 있게 해줍니다. 객체는 정의된 데이터와 로직을 캡슐화하기 때문에 프로그램의 특정 임무에 쉽게 객체를 배정할 수 있습니다. 개념적으로 하나의 기능을 가진 단위(예를 들면 “고객 기록”)처럼 설계도에서 어디든 가져다 붙일 수 있으며, 인터페이스 기반에서 다른 객체와 정보를 주고 받는 소통 경로를 그릴 수 있습니다.

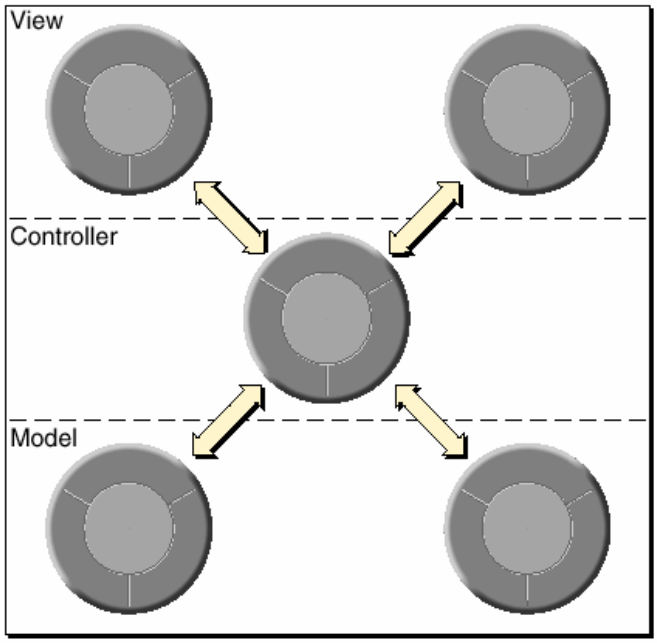
“객체 지향 프로그래밍” 과 관련된 데이터 캡슐화, 메시지, 메소드 등 자세한 내용은 *Inside Mac OS X: The Objective-C Language* 를 참조하십시오.

모델-뷰-컨트롤러(MVC) 패러다임

통화 변환기는 지극히 간단한 응용 프로그램이지만, 설계는 필요합니다. 설계는 모델-뷰-컨트롤러(MVC) 패러다임에 기반을 둔 것으로, 객체 지향 프로그램용으로 자주 쓰이는 모델입니다. 이 디자인 패러다임은 유지, 확장, 그리고 이해가 가능한 시스템을 개발할 수 있도록 도와줍니다.

모델-뷰-컨트롤러(MVC)는 Smalltalk-80 에서 유래하였습니다. MVC 는 같은 응용 프로그램 내에 있는 객체들을 모델, 뷰, 컨트롤러라는 추상적인 경계로 나누어 서로 소통하는 구조로 되어 있습니다.

그림 2-2 모델-뷰-컨트롤러 패러다임의 객체 관계



모델 객체

이 객체는 전문적인 지식과 기술을 말합니다. 모델 객체는 데이터를 포함하고 있으며, 그 데이터를 조작하는 로직을 정의합니다. 예를 들자면, '고객'이란 객체는 비즈니스용 프로그램에서 자주 사용되는 모델 객체입니다. 모델 객체는 고객에 관한 대표적인 정보를 나타내는 데이터를 가지고 있으며, 그 정보로부터 새로운 데이터를 계산하고 액세스 해주는 알고리즘에 접근합니다. 더욱 전문화된 모델 클래스로는 기상 관측 시스템의 'Front'라는 클래스를 들 수 있는데,

이 클래스의 객체는 날씨 전망을 나타내는 데이터와 정보를 가지고 있습니다. 모델 객체는 직접적으로 보여지지 않습니다. 이들은 다양한 플랫폼에서 이동, 지속, 분배, 재사용할 수 있습니다.

뷰 객체

패러다임의 뷰 객체는 사용자 인터페이스에서 볼 수 있는 것들(윈도우, 버튼 등)을 말합니다. 뷰 객체는 디스플레이하는 데이터에 대해 알지 못합니다. 응용 프로그램 키트는 윈도우, 텍스트 필드, 스크롤 뷰, 브라우저 등 필요한 모든 뷰 객체를 제공합니다. 새로운 방법(그래프 등)으로 자신만의 뷰 객체를 만들어 데이터를 나타낼 수 있으며 프로그램에 적합한 새로운 방법으로 윈도우 내에서 뷰 객체들을 그룹화할 수도 있습니다. 특히 키트

내의 뷰 객체는 재사용성이 매우 높아 프로그램 간에 일관성을 유지할 수 있습니다.

컨트롤러 객체

컨트롤러 객체는 응용 프로그램내에서 모델 객체와 뷰 객체 사이의 매개체로 작용합니다. 일반적으로 하나의 프로그램 또는 하나의 윈도우에는 한 개의 컨트롤러 객체가 있습니다. 컨트롤러 객체는 모델 객체와 뷰 객체 사이에서 데이터를 전달해 줍니다. 또한 nib 파일 로드, 윈도우와 프로그램의 대리자 역할 등 프로그램 특유의 잡무들을 수행합니다. 프로그램에 대해 컨트롤러가 하는 일은 매우 구체적이기 때문에 프로그램 코드의 많은 부분을 차지하면서도 일반적으로 재사용이 불가능합니다. (컨트롤러 객체가 모두 재사용될 수 없다는 것은 아닙니다. 설계를 잘 한다면 재사용이 가능합니다.)

컨트롤러의 핵심적인 역할인 중개자로서의 역할 때문에 모델 객체는 유저 인터페이스의 상태와 이벤트에 대해 알 필요가 없으며, 뷰 객체는 모델 객체의 프로그램 인터페이스를 알 필요가 없습니다. 직접 작성한 뷰 객체와 모델 객체는 인터페이스 빌더에 있는 팔레트에서 다른 사람이 사용할 수 있게 할 수 있습니다.

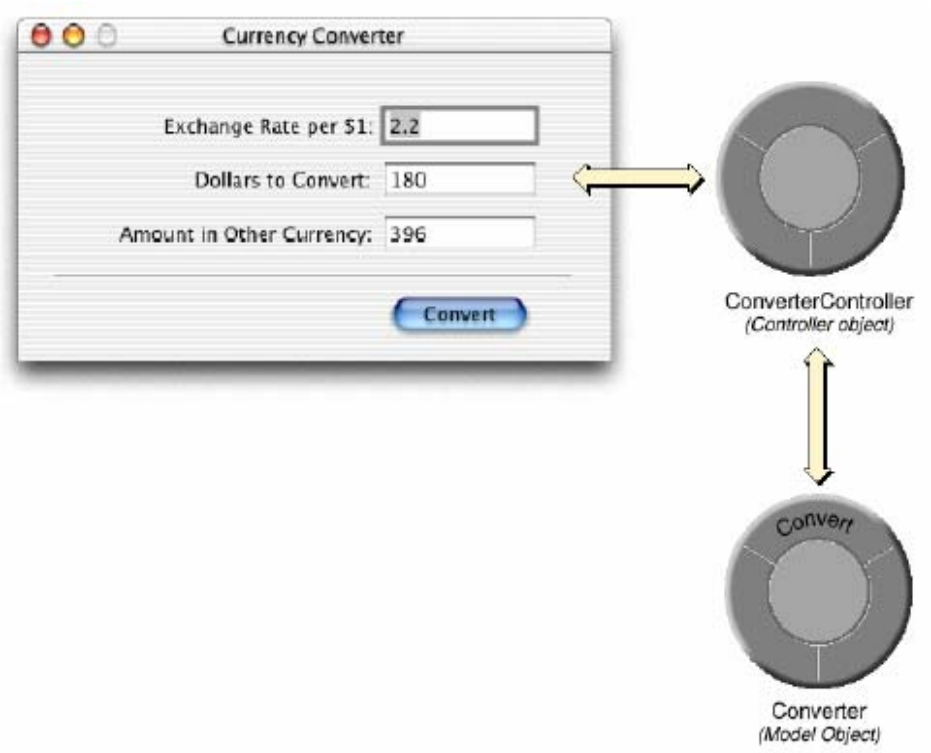
하이브리드 모델 (혼성 모델)

엄격히 말해서, MVC 는 모든 환경에서 권장할 만한 것은 아닙니다. 종종 각 객체의 역할을 결합시키는 것이 좋습니다. 예를 들어 아케이드 게임과 같은 그래픽이 중요한 응용 프로그램에서는 뷰와 모델의 역할을 합친 여러 개의 뷰 객체를 만들 수 있습니다. 몇몇 프로그램, 특히 간단한 프로그램에서는 컨트롤러와 모델을 결합할 수 있습니다. 이들 객체는 컨트롤러의 후크를 사용하여 특수한 데이터 구조와 모델 객체의 로직을 인터페이스에 결합합니다.

통화 변환기 설계 MVC

통화 변환기는 미리 준비된 프로그램 키트 객체의 컬렉션을 사용하여 실행되는 커스텀 객체(모델과 컨트롤러)와 사용자 인터페이스(뷰)로 이루어져 있습니다. Converter 객체는 통화량을 계산하여 그 값을 반환해주는 역할을 합니다. 사용자 인터페이스와 Converter 객체 사이의 컨트롤러 객체는 ConverterController 입니다. ConverterController 는 Converter 객체와 UI 객체 사이에서의 활동을 조정합니다.

그림 2-3 통화 변환기의 모델-뷰-컨트롤러 관계



ConverterController 클래스는 중심 역할을 합니다. 모든 컨트롤러 객체처럼 인터페이스 객체, 모델 객체와 서로 정보를 주고 받으며, 프로그램과 관련된 특수한 작업을 처리합니다. ConverterController 는 사용자가 필드에 입력한 값을 받아서 Converter 객체에 넘겨주며, Converter 로부터 결과를 받아서 인터페이스에 있는 필드에 이 결과를 넣습니다.

Converter 클래스는 단지 두 인자가 넘겨 준 값을 계산하여 결과값을 반환합니다. 계산 뿐 아니라 다른 모델 객체처럼 데이터를 가질 수도 있습니다. 따라서 (예를 들어) 고객의 레코드를 나타내는 객체는 Converter 와 유사합니다. Converter 클래스를 그 프로그램의 구체적인 세부사항과 분리하면 통화 변환기 설계의 재사용성이 높아집니다.

이 통화 변환기 디자인은 몇 가지의 예시를 보여 주기 위해 지나치게 단순화하여 설계된 부분도 있습니다. 프로그램의 컨트롤러 클래스인 ConverterController 가 Converter 클래스 없이 계산하게 할 수도 있습니다.

3 장. 통화 변환기 프로젝트 제작

이 장에서는 통화 변환기를 개발하는 방법을 설명하고, 그 과정에서 Objective-C 를 사용하여 Cocoa 응용 프로그램을 개발하는데 필수적인 단계에 대해 논합니다.

이 장에서는 다음과 같은 내용을 제공합니다.

1. “통화 변환기 프로젝트 제작”

2. “통화 변환기 인터페이스 제작”

3. “통화 변환기 클래스 정의”

4. “ConverterController 를 인터페이스에 연결”

5. “통화 변환기 클래스 구현”

6. “통화 변환기 개발, 디버그 및 실행”

통화 변환기 프로젝트 제작

모든 Cocoa 응용 프로그램은 프로젝트에서 시작됩니다. 프로젝트는 소스 코드 파일, 프레임웍, 라이브러리, 유저 인터페이스, 사운드, 이미지 등 프로그램에 들어가는 모든 요소들의 저장소입니다. Project Builder 프로그램을 이용하여 프로젝트를 제작하고 관리할 수 있습니다.

Cocoa 프로젝트를 만드는 세가지 기본 단계입니다.

“Project Builder” 열기” (7 페이지)

“New Project 명령 선택” (8 페이지)

“Project 타입 선택” (8 페이지)

Project Builder 열기

Project Builder 를 열기 위해서는,

1. /Developer/Applications/에서 Project Builder 를 찾으십시오.

2. 아이콘을 이중 클릭 하십시오.

그림 3-1 Project Builder 응용 프로그램 아이콘

Project Builder 를 처음 실행하면 몇 가지 설정에 관한 질문이 나타날 것입니다. 대다수의 사용자는 기본값을 사용하여 작업할 수 있습니다.

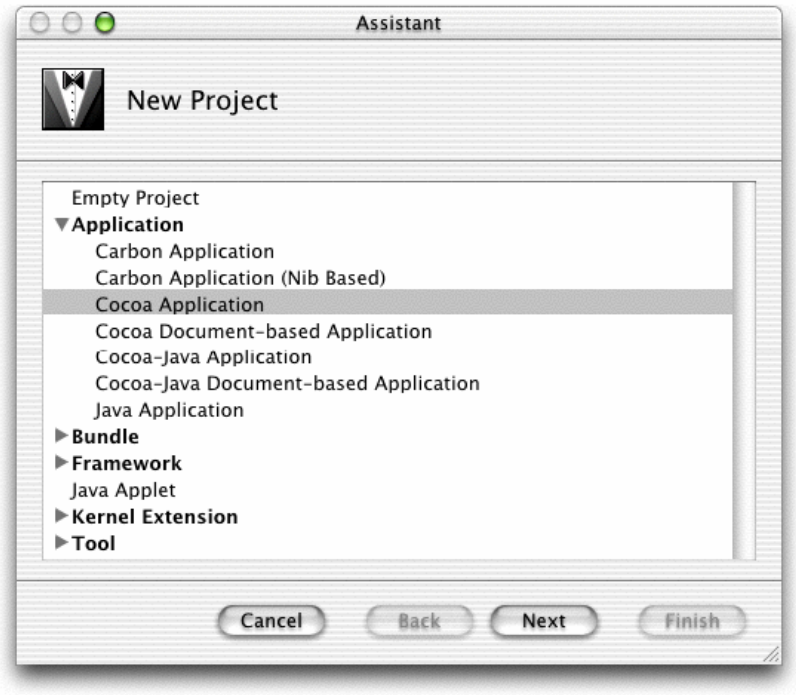
New Project 명령 선택

Project Builder 를 실행하면 메뉴만 나타납니다. 프로젝트를 만들기 위해서는 File 메뉴에서 New Project 를 선택합니다. 그러면 Project Builder 에 New Project 패널이 나타날 것입니다.

Project Type 선택

Project Builder 를 사용하면 Carbon 과 Cocoa 프로그램에서부터 Mac OS X 커널 확장, Mac OS X 프레임웍에 이르기 까지 매우 다양한 종류의 프로그램을 제작할 수 있습니다. 이 튜토리얼의 작업을 위해서 그림 3-2 와 같이 Cocoa Application 을 선택하고 Next 를 클릭하십시오.

그림 3-2 Project Builder 의 New Project Assistant

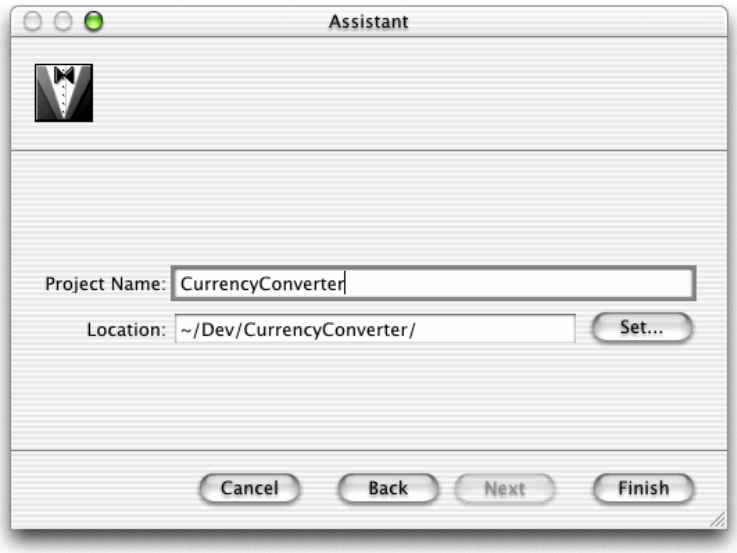


1. 파일 시스템 브라우저를 사용하여 원하는 프로젝트를 만들고자 하는 디렉토리로 이동하십시오.

2. Name 필드에 프로젝트명을 입력하십시오. 이 프로젝트에서는 “CurrencyConverter”를 입력하면 됩니다.

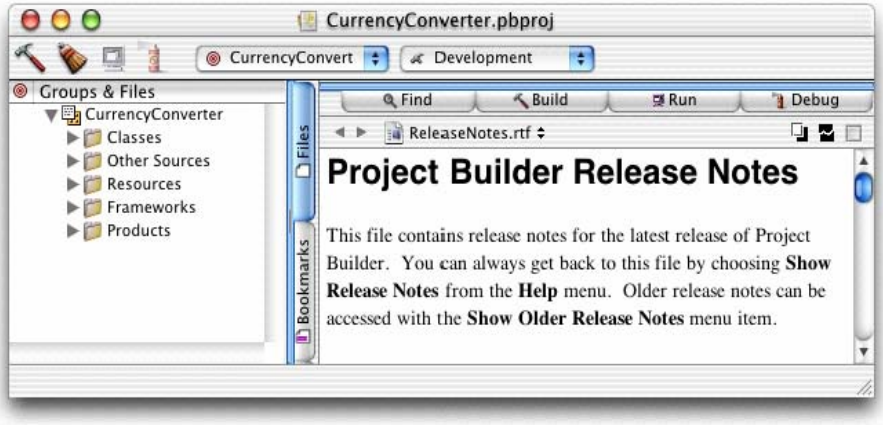
3. Finish 를 클릭하십시오.

그림 3-3 Project Builder 의 New Project Assistant



Finish 를 클릭하면 Project Builder 에 그림 3-4 와 같은 프로젝트 윈도우가 생성되어 나타납니다.

그림 3-4 Project Builder 의 새로운 통화 변환기



Project Builder 는 계층적인 그룹을 사용하여 프로젝트를 구성합니다. 이들 그룹은 매우 융통적이므로 디스크상의 프로젝트 레이아웃이나 파일을 다루는 빌드 시스템을 반드시 나타낼 필요가 없습니다. 이 그룹들은 전적으로 프로젝트를 구성하는 데에만 사용됩니다. 템플릿으로 생성된 기본 그룹들은 그대로, 혹은 사용자가 원하는 대로 재구성하여 사용할 수 있습니다.

이제 프로젝트 브라우저의 왼쪽 열에 있는 아이템을 클릭하여 기본 그룹을 살펴 보겠습니다.

Classes 이 그룹은 처음에는 비어 있으나 프로젝트를 위한 실행 파일과 헤더 파일들을 담게 됩니다.

Other Sources 이 그룹은 초기 리소스들을 로드하여 프로그램을 실행하는 main.m, main() 루틴을 포함하고 있습니다. (이 파일은 수정할 필요가 없습니다.)

Resources 이 그룹은 nib 파일(확장자가 .nib 인 파일)과 프로그램의 유저 인터페이스를 지정해 주는 기타 리소스들을 가지고 있습니다. nib 파일에 대한 자세한 내용은 뒤에서 설명하도록 하겠습니다.

Frameworks 이 그룹은 프로그램이 임포트할 프레임웍(라이브러리와 유사)을 가지고 있습니다.

Products 이 그룹은 프로젝트 빌드의 결과물을 가지고 있으며 프로젝트에 있는 각각의 타겟에 의해 생성된 프로덕트의 레퍼런스가 자동적으로 위치합니다.

이제 프로젝트 디렉토리에 어떤 종류의 파일이 있는지 확인해 보겠습니다.

English.lproj

사용자의 언어로 로컬라이즈된 리소스를 포함하고 있는 디렉토리입니다. nib 파일은 이 디렉토리에서 자동으로 생성됩니다. nib 파일에 관한 더 자세한 정보는 Interface Builder 에 관한 개발자 문서를 참고하시기 바랍니다.

main.m

각 프로젝트를 위해 생성되는 하나의 파일로서 프로그램을 위한 시작 지점 코드를 포함하고 있습니다.

CurrencyConverter.pbproj

이 파일은 프로젝트를 정의하는 정보를 가지고 있습니다. 이 파일도 수정해서는 안됩니다. Finder 에서 이 파일을 더블 클릭하면 프로젝트를 열 수 있습니다.

통화 변환기 인터페이스 제작

이 섹션에서는 통화 변환기의 기능적인 그래픽 인터페이스 제작 시 코딩 없이 12 단계로 작업하는 방법에 대해 소개하고, 이러한 과정을 거쳐 Cocoa 프로그래밍의 흥미롭고, 중요한 방식에 대해 설명합니다.

12 단계는 다음과 같습니다.

“메인 Nib 파일 열기”

“윈도우 사이즈 조정”

“윈도우의 타이틀과 속성 설정”

“텍스트 필드 배치, 크기 조정 및 초기화”

“객체 복사”

“텍스트 필드의 속성 변경”

“필드에 레이블 부여”

“인터페이스에 버튼 추가 및 초기화”

“수평선 추가”

“윈도우 레이아웃 완성”

“텍스트 필드 간 탭 이동”

Nib 파일

그래픽 유저 인터페이스를 가지고 있는 모든 프로그램은 최소한 한 개 이상의 nib 파일을 가지고 있습니다. 메인 nib 파일은 응용 프로그램을 실행시킬 때 자동적으로 로딩되며 일반적으로 여러 가지 객체와 한 개 이상의 윈도우를 가지고 있습니다. 물론 프로그램은 여러 개의 nib 파일을 가질 수 있습니다. 각각의 nib 파일에는 다음과 같은 항목이 포함되어 있습니다.

아카이브 객체 객체 지향 용어에서는 “flattened” 또는 “serialized” 객체라고도 알려져 있습니다. 이것은 디스크에 저장되고(또는 네트워크로 전송되고) 나중에 메모리에 저장되는 것과 같은 방법으로 인코드된 객체란 뜻입니다. 아카이브 객체는 크기, 위치, 객체 계층상의 지위 등에 관한 정보를 가지고 있습니다. 아카이브 객체에서 최상위 계층은 File 의 Owner 객체로서, nib 파일(일반적으로 디스크에서 nib 파일을 로드하는 파일)을 가지고 있는 실제 객체를 가리키는 프록시 객체입니다.

이미지 nib 파일 윈도우 또는 이미지 파일을 수용할 수 있는 객체(단추 또는 이미지) 위로 드래그앤드롭하는 이미지 파일입니다.

Class References Interface Builder 는 Cocoa 객체와 팔레트(고정 팔레트)로 만든 객체의 세부 사항을 저장할 수는 있지만 사용자 클래스의 코드를 액세스할 수 있는 방법이 없기 때문에 인스턴스를 저장할 수 없습니다. 이들 클래스에 대해 Interface Builder 는 클래스 정보를 부착한 프록시 객체를 저장합니다.

접속 정보 객체 계층 안에서 상호간의 연결 정보입니다. Interface Builder 의 특별한 커넥터 객체가 이 정보를 저장합니다. 문서를 저장하면, 커넥터 객체는 그 커넥터 객체가 연결하는 객체와 함께 저장됩니다.

Main Nib 파일 열기

1. Project Builder 에서 리소스 그룹에 있는 MainMenu.nib 을 찾으십시오.

2. MainMenu.nib 을 이중 클릭하여 여십시오. Interface Builder 가 실행되어 nib 파일이 열립니다.

Nib 파일이 열리면 기본 메뉴 바와 “Window”라는 타이틀을 가진 윈도우가 나타납니다.

Cocoa 의 윈도우

Cocoa 의 윈도우는 Windows 나 Mac OS 9 사용자 환경의 윈도우와 거의 유사합니다. 이는 응용 프로그램의 컨트롤, 필드, 텍스트, 그래픽 등을 보여주는 스크린 상의 사각형 영역입니다. 윈도우는 스크린 상에서 이동이 가능하며, 종이처럼 다른 윈도우 위에 겹쳐 놓을 수 있습니다. 전형적인 Cocoa 윈도우에는 타이틀 바, 컨텐트 영역, 그리고 몇 가지 컨트롤 객체가 있습니다.

NSWindow 와 Window Server

표준 윈도우를 비롯한 대부분의 사용자 인터페이스 객체가 윈도우입니다. 메뉴, 팝업 리스트, 풀 다운 리스트는 기본적으로 윈도우이고 경고 패널, “Show Info” 윈도우, 툴 팔레트 등의 다양한 패널도 마찬가지입니다. 사실상 스크린에 그려지는 모든 것이 윈도우 안에 나타나야 합니다. 그러나 사용자들은 그것이 “윈도우”라는 사실을 정확하게 인식하지는 못합니다.

두개의 상호 작용하는 시스템은 Cocoa 윈도우를 제작하고 관리합니다. 한편, 윈도우는 Window Server 에 의해 생성됩니다. Window Server 는 Quartz(하부 레벨 드로잉 시스템)의 내부 윈도우 관리를 활용하여 Quartz 그래픽 원형을 사용해서 윈도우를 드로잉하고, 크기를 조정하고, 숨기거나 이동시키는 프로세스입니다. Window Server 는 (마우스 클릭과 같은) 사용자 이벤트도 감지하여 프로그램에 전달합니다.

Window Server 가 생성한 윈도우는 응용 프로그램 키트가 제공하는 객체와 한 조를 이룹니다. NSWindow 클래스의 인스턴스, Cocoa 프로그램에서 각각의 물리적인 윈도우는 NSWindow(또는 서브클래스) 인스턴스가 관리합니다.

NSWindow 객체를 만들면, Window Server 는 NSWindow 객체가 관리할 물리적인 윈도우를 생성합니다. Window Server 는, 윈도우는 윈도우 번호로, NSWindow 는 고유 식별자로 구분합니다.

응용 프로그램, 윈도우, 뷰

Cocoa 응용 프로그램을 실행하면, NSWindow 객체는 NSApplication 과 프로그램의 뷰 사이에서 중간에 위치합니다. (뷰는 자신을 드로우 할 수 있고, 사용자 이벤트를 감지할 수 있는 객체입니다.) NSApplication 객체는 윈도우의 리스트를 가지고 있어서 각각의 현재 상황을 추적합니다. 한편으로 각각의 윈도우는 윈도우 뿐 아니라 뷰의 계층도 관리합니다.

여기에서 최고 계층은 컨텐트 뷰로서 윈도우의 컨텐트 사각 영역 안에 딱 들어 맞습니다. 컨텐트 뷰는 자신의 하위 계층에 있는 다른 모든 뷰(서브뷰)를 포함합니다. NSWindow 는 계층상의 뷰에 이벤트를 전달하고 이들 사이에서 변환을 조절합니다.

또 하나의 사각형인 프레임은 윈도우의 외부 경계를 정의하며, 타이틀 바와 윈도우의 컨트롤이 프레임에 포함되어 있습니다. Cocoa 는 스크린의 좌표계에서 윈도우의 상대적인 위치를 정의하는 왼쪽 하단 모서리를 이용하여 윈도우를 나타내기 위한 기본 좌표계를 설정합니다. 이는 왼쪽 상단을 좌표계의 기본으로 하는 Carbon 과 Classic 응용 프로그램과는 다른 점입니다. 뷰는 이 기본 좌표계로부터 변형된 좌표계에서 자신을 드로우 합니다.

Key 와 Main 윈도우

윈도우는 수많은 특성을 가지고 있습니다. 이는 스크린 상에 보이는 것도 있고, 보이지 않는 것도 있습니다. 스크린 상의 윈도우는 Window Server 에 의해 관리되는 층(tier)으로 스크린 상에서 겹치게 놓일 수 있습니다. 스크린 상의 윈도우는 key 또는 main 의 상태를 가질 수 있습니다.

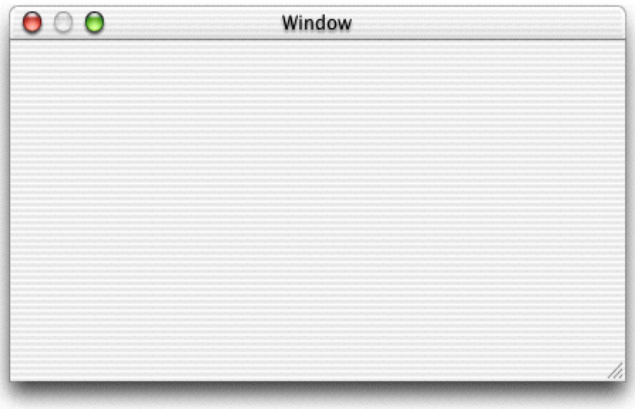
Key 윈도우는 프로그램의 키 입력에 반응하며, 메뉴와 패널에서 들어오는 메시지의 주요 수용체입니다. 일반적으로 사용자가 키를 클릭하면 윈도우가 키를 감지합니다. 각각의 프로그램은 단지 하나의 key 윈도우만 가질 수 있습니다.

응용 프로그램은 하나의 main 윈도우를 가지고 있으며, 키 상태도 함께 가질 수 있습니다. 메인 윈도우는 프로그램에 대한 사용자의 액션에 주로 초점을 맞춥니다. 모달 타입의 key 윈도우(Font 패널이나 Info 윈도우와 같은 패널)에서의 사용자 액션은 main 윈도우에 직접적인 영향을 주기도 합니다.

윈도우 크기 조정

1. 윈도우의 오른쪽 하단 모서리를 안쪽으로 드래그하여 작게 만드십시오.

그림 3-5 Interface Builder 에서 윈도우 크기 조정



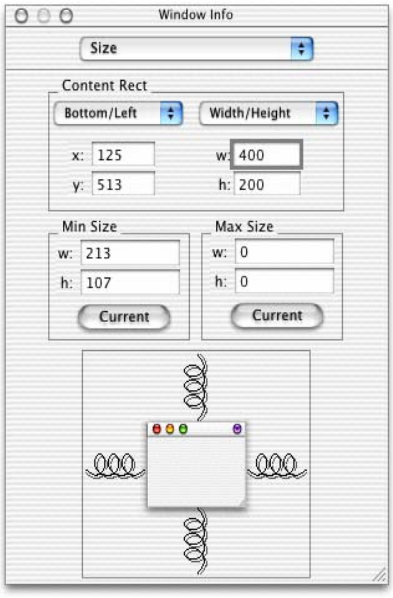
Window Info 윈도우의 Size 메뉴를 이용하면 윈도우의 크기를 더 정확하게 조절할 수 있습니다.

1. Tools 메뉴에서 Show Info 를 선택하십시오.

2. 팝업 메뉴에서 Size 를 선택하십시오.

3.”Content Rect” 영역에 있는 오른쪽 팝업 메뉴에서 “Width/Height”를 선택하십시오. 그림 3-6 과 같이 Width/Height 메뉴 아래에 있는 텍스트필드에 너비(w) 400, 높이(h) 200 을 입력하십시오.

그림 3-6 Interface Builder 의 Info 팔렛트로 윈도우 크기 조정



윈도우의 타이틀과 속성 설정

Info 윈도우가 열리면 윈도우에 대한 다른 속성도 지정하십시오.

1. Info 윈도우의 팝업 메뉴에서 속성을 선택하고 윈도우의 타이틀을 “CurrencyConverter”로 바꾸십시오.

2. “Visible at launch time” 옵션을 선택하십시오.

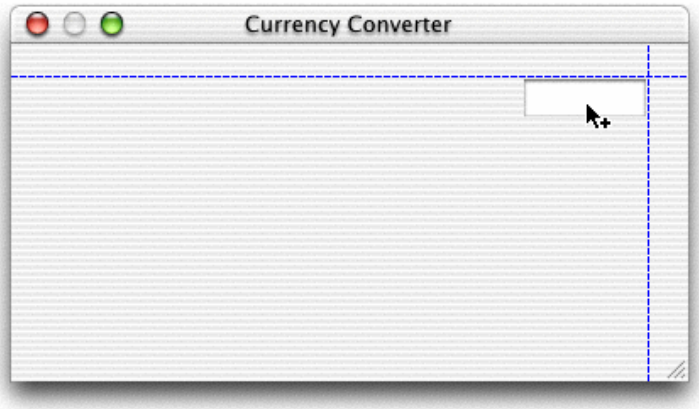
3. “Control” 영역에서 “Resize” 체크 박스의 선택을 해제하십시오.

텍스트 필드 배치, 크기 조정 및 초기화

1. 텍스트 필드 객체를 CurrencyConverter 윈도우 위로 드래그하십시오. Interface Builder 는 객체가 윈도우의 가장자리나 다른 객체에 가까이 접근하면 적당한 지점에서 팝업 가이드를 보여주어 Aqua Interface Guidelines 에 따라 객체의 위치를 지정할 수 있게 해줍니다.

2. 텍스트 필드를 재조정하려면 핸들을 잡고서 원하는 방향으로 드래그하십시오. 예를 들어 텍스트 필드를 왼쪽으로 확대하려면 왼쪽 핸들을 드래그하면 됩니다.

그림 3-7 Interface Builder 에 텍스트 필드 추가하기



Currency Converter 에는 방금 만든 것과 같은 크기의 텍스트 필드가 두 개 더 필요합니다. 새로운 텍스트 필드는 팔레트에서 또 하나의 객체를 드래그하여 크기를 동일하게 조정하거나, 처음의 객체를 복사해서 만들 수 있습니다.

객체 복사

1. 텍스트 필드가 선택되어 있지 않으면 텍스트 필드를 선택하십시오.

2. Edit 메뉴에서 Duplicate(커맨드-D)를 선택하십시오. 새로운 텍스트 필드가 원본 필드 위에 약간 겹치게 생성되어 나타납니다.

3. 새로운 텍스트 필드를 첫번째 텍스트 필드 아래에 위치 시키십시오. 안내 정보가 나타나 두번째 텍스트 필드를 제자리에 “snapping”하도록 도와줄 것입니다.

4. 세번째 텍스트 필드를 만들려면 커맨드-D 를 한번 더 입력하십시오. IB는 이전의 Duplicate 명령을 기억하여 자동적으로 새로 생성한 텍스트 필드에 적용시킵니다.

텍스트 필드의 속성 변경

맨 아래의 텍스트 필드는 계산 결과를 보여주게 될 것이므로 다른 두개의 텍스트 필드와 다른 속성을 가져야 합니다. 이 텍스트 필드는 편집하거나 선택할 수 없어야 합니다.

1. 세번째 텍스트 필드를 선택하십시오.

2. Info Window 를 불러와서 팝업 메뉴에서 Attributes 를 선택합니다.

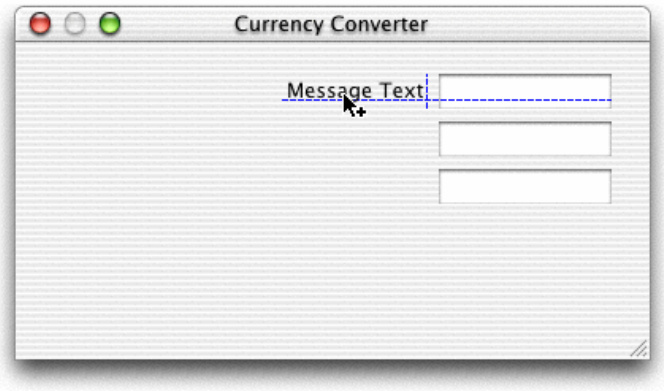
3. Info Window 의 Options 에서 Editable 속성을 해제하면 사용자가 필드의 내용을 수정할 수 없게 됩니다. Selectable 속성은 그대로 두어 사용자가 다른 프로그램에 복사하여 붙일 수(카피앤페이스트) 있도록 합니다.

필드에 레이블 부여

텍스트 필드에 레이블이 없으면 구별할 수 없으므로 View 팔레트에서 미리 만들어져 있는 레이블 객체를 사용하여 레이블을 추가합니다.

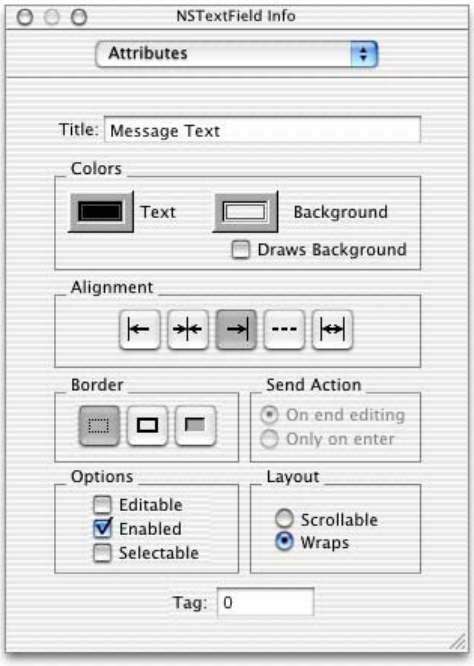
1. Message Text 객체를 View 팔레트에서 윈도우 위로 드래그하십시오.

그림 3-8 Interface Builder 에 텍스트 레이블 추가하기



2. Info Window 의 Alignment 영역에서 왼쪽에서 세 번째 버튼을 클릭하여 글자가 오른쪽 정렬이 되게 하십시오.

그림 3-9 Interface Builder 에 텍스트 레이블 오른쪽 정렬하기



3. 텍스트 레이블을 두 번 복제하여 각각에 다음과 같이 입력하고, 정렬하십시오.

그림 3-10 Interface Builder 에서 텍스트 필드 및 레이블 정렬하기

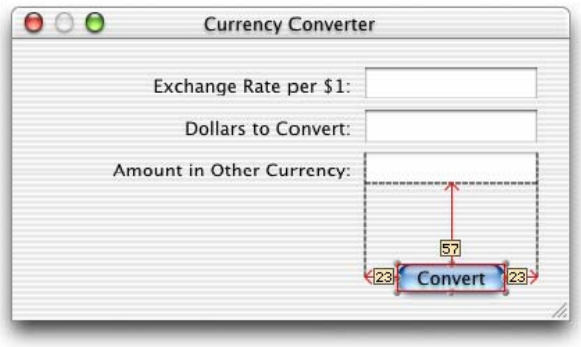
인터페이스에 버튼 추가 및 초기화

통화 변환기는 버튼이나 리턴 키를 눌러서 실행할 수 있습니다.

1. View 팔레트에서 버튼 객체를 드래그하여 윈도우의 오른쪽 하단에 놓으십시오.
2. 버튼의 타이틀을 이중 클릭하여 버튼의 텍스트 필드를 선택하고, 타이틀을 “Converter”로 바꾸십시오.
3. NSButton Info Window 에서 Attributes 를 선택하고, “Equiv:”라는 팝업 메뉴에서 Return 을 선택하십시오. 이렇게 하면 버튼이 마우스 클릭뿐 아니라 Return 키에도 응답할 수 있게 됩니다.
4. 텍스트 필드 아래에서 버튼을 정렬하십시오.

   먼저, 버튼을 Aqua 가이드가 나타날 때까지 아래 방향으로 드래그합니다. 버튼을 선택한 채로 옵션 키를 누릅니다. 커서를 Interface Builder 주위로 옮기면 버튼에서 커서가 가리키는 객체의 위치까지의 거리가 나타납니다. 옵션 키를 계속 누르고 맨 아래의 텍스트 필드에 커서를 둔 채로 버튼이 텍스트 필드의 정 중앙에 오도록 방향 키로 조금씩 밀어 조정합니다.

그림 3-11 Interface Builder 에서 거리 측정하기



수평선 추가

Currency Converter 의 텍스트 필드와 버튼 사이에 있는 깔끔한 선을 추가해 봅시다.

1. 수평 구분자 객체를 View 팔레트에서 인터페이스 위로 드래그하십시오. 이것은 View 팔레트의 오른쪽 하단의 Box 객체 바로 아래에 있습니다.

2. 라인이 윈도우를 가로질러 확장할 때까지 라인의 엔드포인트를 드래그하십시오.

그림 3-12 Interface Builder 에서 Currency Converter 에 요소 추가하기

3. Convert 버튼을 Aqua 버튼이 나타날 때까지 다시 위로 이동시키십시오.

Aqua 레이아웃 및 객체 정렬

산뜻한 유저 인터페이스를 만들기 위해서는 인터페이스 객체를 줄과 행으로 시각적으로 정렬할 수 있어야 합니다. 눈이 번쩍 뜨일만한 디자인을 구성하는 것은 매우 힘들며, x/y 좌표를 직접 입력해야 하는 지루한 작업에 많은 시간을 들여야 합니다. Aqua 인터페이스 정렬은 객체에 섀도우가 있고, UI 가이드라인 기준이 일반적으로 섀도우를 고려하지 않으므로 더욱 어렵습니다.

Cocoa 에서 모든 드로잉은 객체 프레임의 경계 내에서 이루어 집니다. 인터페이스 객체는 섀도우를 가지고 있기 때문에, Mac OS 9 에서처럼 프레임의 가장자리에 맞춘다면 시각적으로 바르게 정렬되지 않습니다. 예를 들어 Aqua UI 가이드라인을 보면, 누름단추는 20 픽셀이어야 한다고 나와 있지만, 버튼과 섀도우가 제대로 보이기 위해서는 32 픽셀이 필요합니다. 레이아웃 사각형은 반드시 정렬해야 합니다.

Layout 메뉴(커맨드-L)에 있는 “Show Layout Rectangles”를 사용하여 IB 에 있는 객체의 사각 레이아웃(레이아웃 rectangles)을 볼 수 있습니다. 또한 IB 크기 판(size pane)은 프레임과 레이아웃 사각형 사이를 변환할 수 있는 팝업 메뉴를 가지고 있어서 적절한 때 수작업으로 값을 설정할 수 있습니다.

Interface Builder 는 객체를 윈도우에서 다양한 방법으로 정렬할 수 있습니다.

- 객체를 마우스로 Aqua 가이드에 맞춰 드래그
- 방향키 누르기 (그리드는 해제한 채, 선택된 객체를 한 픽셀씩 이동)
- 레퍼런스 객체를 사용하여 선택한 객체를 열과 행에 맞춰 두기
- 내장된 정렬 기능 사용
- Info Window 의 Size 디스플레이에서 원점을 지정
- 그리드 사용

Alignment 와 Guides 에 있는 다양한 Layout 서브 메뉴에서 정렬 명령어와 툴을 보십시오. 또한 버튼이 있는 플로팅 윈도우를 제공하여 다양한 정렬을 해주는 정렬 툴(Tools 메뉴에 있는 Alignment 선택)을 사용할 수 있습니다.

윈도우 레이아웃 완성

통화 변환기의 인터페이스는 거의 완성이 되었습니다. 윈도우 크기를 조정하여 모든 객체가 중앙에, 위치하도록 하며, 각 객체의 가장자리를 정렬시키십시오. 현재 객체들은 상단과 오른쪽 가장자리를 기준으로만 정렬 되어 있습니다.

통화 변환기에서 자동화된 Aqua 가이드를 몇 개의 Layout 명령어와 함께 사용할 것입니다.

1. 세번째 텍스트 레이블 “Amount in Other Currency”를 선택하고, 시프트 키를 누른 채 클릭하여 다른 두 개의 텍스트 필드까지 선택합니다.

2. 레이아웃 메뉴에서 “Size to Fit”을 선택하여 모든 레이블의 폭을 가장 작게 줄이십시오.

3. 레이아웃 메뉴에서 “Same Size”를 선택하고 선택된 텍스트 레이블을 같은 크기로 만드십시오.

4. 레이블을 윈도우의 왼쪽 가장자리로 드래그하고 Aqua 가이드가 나타나면 놓으십시오.

5. 세 텍스트 필드 모드를 선택하고 왼쪽으로 드래그하여 다시 가이드를 사용하여 적절한 위치에 두십시오.

6. 수평의 분리자를 짧게 하여 버튼을 다시 텍스트 필드 아래로 이동시키십시오.

7. 오른쪽에 있는 텍스트 필드와 아래의 Convert 버튼에서 적절한 거리에 나타나는 가이드를 사용하여 윈도우 크기를 재조정하십시오.

이 지점에서 프로그램의 윈도우는 다음과 같이 보입니다.

그림 3-13 Currency Converter의 최종 사용자 인터페이스

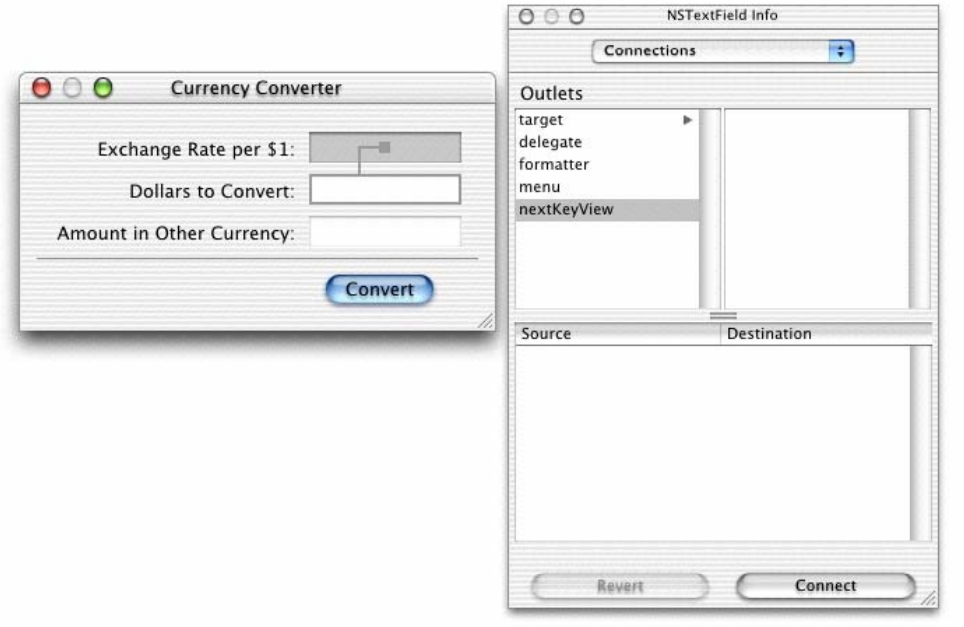
텍스트 필드 간 탭 이동하기

통화 변환기 인터페이스 구성에서 마지막 단계는 외관보다는 동작과 관련이 있습니다. 사용자가 탭 키를 사용하여 편집 가능한 필드에서 다음 필드로 이동하고 처음으로 다시 되돌아올 수 있도록 만들어야 합니다. Interface Builder의 팔레트에 있는 많은 객체는 nextKeyView라는 인스턴스 변수를 가지고 있습니다. 이 변수는 사용자가 탭 키를 누르면 키보드의 이벤트를 받아 다음 객체를 인식합니다.(시프트-탭을 누르면 이전 객체로) 키보드 이벤트를 받은 다음 객체를 확인합니다. 필드 사이의 탭 이동을 원한다면 필드를 nextKeyView 변수를 통해 연결해야 합니다.

1. 첫번째 텍스트 필드를 선택하십시오.

2. 컨트롤 키를 누른 채로 드래그하여 두번째 텍스트 필드로 선을 연결하십시오.

그림 3-14 nextKeyView 아웃렛 연결하기



3. Info 윈도우에서 nextKeyView 를 선택하고 Connect 를 클릭합니다. NextKeyView 아웃렛은 탭 키를 누른 후 이벤트에 응답하는 다음 객체를 찾아냅니다.

4. 같은 과정을 반복하여 두번째 필드에서 첫번째 필드로도 연결합니다.

윈도우용 initialFirstResponder 설정하기

앞 섹션에서 Interface Builder 를 사용하여 두 개 텍스트 필드의 nextKeyView 아웃렛을 연결을 통해 Key view 를 설정했습니다. 이제, 윈도우의 initialfirstResponder 아웃렛을 윈도우가 화면에 처음으로 배치될 때 선택하는 텍스트 필드에 설정해야 합니다. 이 아웃렛을 설정하지 않으면, 윈도우는 키 루프(지정한 것과는 다른 키 루프)를 설정하고, 기본적인 초기 응답자를 선택합니다.

그림 3-15 Interface Builder 에서 initialFirstResponder 아웃렛 설정하기

3. Info 윈도우에서 initialFirstResponder 를 선택하고, Connect 를 클릭하십시오.

인터페이스 테스트

통화 변환기 인터페이스는 이제 완성되었습니다. Interface Builder 는 한 줄의 코드도 작성하지 않고 테스트할 수 있게 해줍니다.

1. File 메뉴에서 Save All 을 선택하여 작업한 것을 저장하십시오.

2. File 메뉴에서 Test Interface 를 선택하십시오.

3. 인터페이스에서 텍스트 필드 간의 탭 이동, 컷앤페이스트 등 다양한 작업을 시험해 보십시오.

4. 작업을 마친 후 Interface Builder 프로그램 메뉴에서 Quit New Application 을 선택하여 테스트 모드에서 빠져 나오십시오.

Interface Builder 에서 프로그램이 처음 실행될 때 나타나는 초기 위치로 사용되는 CurrencyConverter 윈도우의 스크린 위치에 주의하십시오. 윈도우의 초기 위치는 화면의 왼쪽 상단 근처가 적당합니다.

Cocoa 를 이용한 다양한 기능

가장 간단한 Cocoa 응용 프로그램으로서 단 한 줄의 코드도 추가하지 않고서 풍부하고 다양한 기능을 얻을 수 있습니다. 이 기능을 직접 프로그래밍해 보지 않아도 됩니다. Interface Builder 에서 인터페이스를 시험하면서 볼 수 있습니다.

응용 프로그램과 윈도우 동작

테스트 모드에서의 통화 변환기는 화면 상에서 다른 프로그램들과 유사하게 작동합니다. 화면을 클릭하면 통화 변환기는 비활성화되고, 완전히 또는 부분적으로 다른 프로그램의 윈도우에 의해 희미하게 변합니다.

통화 변환기의 윈도우를 클릭하여 활성화하십시오. 타이틀 바를 이용하여 윈도우를 이리 저리 이동해 보십시오. 다음에 테스트할 수 있는 몇 가지가 있습니다.

1. Edit 메뉴를 클릭하십시오. 다른 프로그램들과 마찬가지로 마우스 버튼을 눌렀다 놓으면 해당 항목이 나타났다 사라집니다.

2. 최소화 버튼을 클릭하십시오. 독에서 윈도우의 아이콘을 클릭하여 응용 프로그램이 다시 나타나도록 하십시오.

3. 종료 버튼을 클릭하면 통화 변환기 윈도우가 사라집니다. (메인 메뉴에서 Quit 을 선택하여 윈도우가 다시 테스트 모드로 돌아가도록 합니다.)

Interface Builder 에서 통화 변환기의 윈도우가 크기 재조정 박스를 제거하도록 설정하지 않았다면 크기를 조정할 수 있습니다. 또한 윈도우의 자동 크기 재조정 속성을 설정하여 윈도우의 객체를 비례적으로 조정하거나 초기 크기를 보존할 수 있습니다.(자동 크기 재조정에 대한 더 자세한 정보는 Interface Builder Help 참조)

컨트롤과 텍스트

통화 변환기의 버튼과 텍스트 필드에는 많은 내장된 동작이 있습니다. 리턴 키와 함께 연관된 Aqua 버튼의 기본값인 Convert 버튼 "throbs"를 주목하십시오. 버튼이 끊임 없이 깜박이는 것을 주목하십시오.

다양한 버튼 스타일을 가졌다면 마우스 클릭에도 각각 다른 방법으로 응답할 것입니다.

이제 텍스트 필드 중 하나를 클릭합니다. 필드 내에서 커서가 깜박이는 것을 보십시오. 텍스트를 입력하고 이를 선택합니다. Edit 메뉴에 있는 명령을 사용하여 복사하고 다른 텍스트 필드에 붙입니다.

NextKeyView 로 통화 변환기의 텍스트 필드 사이를 연결한 것을 기억할 것입니다. 텍스트 필드에 커서를 놓고 탭 키를 눌러 커서가 필드에서 필드로 이동하는지 확인하십시오.

메뉴 명령어

Interface Builder 는 모든 새로운 응용 프로그램에 Application, File, Edit, Window, Help 등의 기본 메인 메뉴를 제공합니다. Info 와 같은 이들 중 몇 가지는 미리 만들어진 일련의 명령어들을 가지고 있습니다. 예를 들어 Services 서브 메뉴(서브 메뉴의 아이템은 런타임시 다른 프로그램에 의해 추가됩니다.)로 다른 Cocoa 응용 프로그램과 소통이 가능하며 Window 메뉴로 자신의 프로그램 윈도우를 관리할 수 있습니다.

통화 변환기는 Quit 과 Hide 명령어, Edit 메뉴의 Copy, Cut, Paste 명령어 등, 단지 몇 가지의 명령어만 필요합니다. 원치 않는 명령어는 삭제할 수 있습니다. 새로운 것을 추가하여 또 다른 동작을 만들 수도 있습니다. Interface Builder 에서 설계한 응용 프로그램은 컴파일 할 필요 없이 단순히 메뉴나 메뉴 명령어를 추가해서 기능을 얻을 수 있습니다. 예를 들어

- Font 서브 메뉴는 NSText 객체에 있는 텍스트에 폰트를 적용하기 위하여 DataViews 팔레트에 있는 스크롤 뷰 객체 중 하나처럼 동작을 추가합니다.
- 텍스트 서브 메뉴는 텍스트가 편집될 수 있는 어느 곳에서나 텍스트 정렬, 탭과 들여쓰기가 가능하며, 정렬을 위해 NSText 객체에 있는 룰러가 보이게 해줍니다.

- 텍스트나 이미지를 보여 주는 많은 객체는 가지고 있는 컨텐트를 PDF 데이터로 프린트할 수 있습니다.

도큐멘트 관리

많은 응용 프로그램은 도큐멘트라는 반복 가능하고, 반 자율적인 객체를 생성하고 관리할 수 있습니다. 도큐멘트는 분리된 정보들을 가지고 있으며, 정보의 등록과 유지 보수를 지원합니다. 워드프로세싱 문서가 전형적인 예입니다. 사용자의 조정에 따라 응용 프로그램은 도큐멘트를 만들고, 저장하고, 닫거나 관리합니다.

파일 관리

응용 프로그램은 응용 프로그램 키트에 의해 생성되고 관리되는 Open 패널을 사용하여 사용자가 파일 시스템에서 파일을 찾아 내어 열 수 있도록 도움을 줍니다. 또한 Save 패널을 사용하여 파일에 정보를 저장합니다. Cocoa 는 또한 파일 시스템에서 파일 관리(생성, 비교, 복사, 이동 등)와 사용자 기본값을 관리하기 위한 클래스를 제공합니다.

다른 응용 프로그램과 소통하기

Cocoa 는 응용 프로그램이 다른 응용 프로그램과 정보를 교환할 수 있도록 여러 방법을 지원합니다.

- **Pasteboard** pasteboard 는 응용 프로그램 사이에서 정보를 공유하기 위한 일반적인 설비입니다. 응용 프로그램은 pasteboard 로 사용자가 잘라내거나 복사한 데이터를 다른 응용 프로그램에 붙일 수 있습니다.

- **Services** 어떤 응용 프로그램이라도 텍스트와 같은 특정 데이터 형 기반의 다른 프로그램이 제공하는 서비스를 액세스할 수 있습니다. 응용 프로그램은 암호화, 언어 변환, 레코드 페치 등의 서비스를 다른 프로그램에도 제공할 수 있습니다.

- **드래그앤드롭** 응용 프로그램이 적절한 프로토콜을 수행할 경우, 객체를 다른 프로그램의 인터페이스에서 가져 오거나 내보낼 수 있습니다.

커스텀 드로잉과 애니메이션

Cocoa 는 컨텐트와 사용자의 작업에 대한 응답을 드로잉하기 위해 자신만의 커스텀 뷰를 제작할 수 있습니다. 이를 지원하기 위해 Cocoa 는 NSBezierPath 클래스와 같은 드로잉을 위한 객체와 함수를 제공합니다.

로컬라이제이션

Cocoa 는 응용 프로그램의 일부인 nib 파일, 스트링, 이미지, 사운드의 로컬라이제이션을 지원하기 위해 API 와 툴을 지원합니다.

편집 지원

Interface Builder 에서 특정 메뉴를 자신이 만드는 프로그램의 메뉴 바에 추가하면 여러 가지 패널(기능과 결합된)을 얻을 수 있습니다. 이러한 "add-ons"는 Font 패널(과 폰트 관리), Color 패널(과 색상 관리)을 포함하고 있으며, 패널이 아닐 경우에도 Text 메뉴가 가지고 있는 텍스트 룰러와 탭과 들여쓰기 기능도 포함하고 있습니다.

Formatter 클래스는 응용 프로그램에 숫자, 날짜 등 필드 값에 대한 포맷을 지원합니다. 필드의 컨텐트가 유효한지 확인할 수도 있습니다.

프린트

Cocoa 는 Interface Builder 에서 쉽게 그래픽이나 텍스트를 포함하고 있는 뷰의 프린트를 자동화합니다. 사용자가 컨트롤을 클릭하면 해당하는 패널이 프린트 작업 구성을 돕습니다. 출력물은 WYSIWYG 방식입니다.

응용 프로그램 키트(Application Kit)의 몇몇 클래스는 도큐멘트, 폼 등의 프린트를 더 잘 컨트롤하고, 페이징과 페이지 기반 기능도 지원합니다.

도움말

Interface Builder 에서 Info Window 를 사용하여 손쉽게 응용 프로그램에 context-sensitive 도움말("tool tips"라고도 함)을 만들 수 있습니다. 툴팁 텍스트를 입력한 후에 응용 프로그램 인터페이스 상의 객체 위에 커서를 올려놓으면 작은 윈도우가 나타나서 이 객체에 관한 간략한 정보를 보여줍니다.

플러그인 아키텍쳐

사용자가 나중에 새로운 모듈을 결합할 수 있도록 응용 프로그램을 설계할 수 있습니다. 예를 들어, 드로잉 프로그램의 툴 팔레트에 펜슬, 브러시, 지우개 등이 있습니다. 새로운 툴을 만들어 사용자가 이를 설치하도록 할 수 있습니다. 응용 프로그램을 다시 시작하면 이 툴이 팔레트에 나타납니다.

Currency Converter 의 클래스 정의하기

Interface Builder 는 프로그램 개발자에게 매우 유용한 도구입니다. 이는 프로그램의 그래픽 유저 인터페이스를 구성할 뿐 아니라 응용 프로그램 클래스에서 프로그램적인 인터페이스의 많은 부분을 구성하고 이들 클래스로부터 생성된 객체를 연결하는 방법을 제공합니다.

Currency Converter 의 클래스를 정의하기 위한 세가지 단계입니다.

“서브 클래스 정의”

“클래스의 아웃렛 정의”

“클래스의 액션을 정의”

클래스와 객체

입문자들에게는 객체 지향 용어에서 “객체”와 “클래스”라는 용어를 서로 혼동하여 사용하는 경우가 종종 있습니다. 객체와 클래스는 같은 것일까요? 같지 않다면 어떻게 다를까요? 어떤 관계일까요?

객체와 클래스는 둘 다 프로그램의 구성 요소입니다. 이들은 밀접한 관계에 있으나, 프로그램 내에서는 전혀 다른 역할을 합니다.

우선 클래스는 객체를 구분하여 카테고리화하는 유용한 방법을 지원합니다. 마치 소나무를 특정 나무라고 말하는 것처럼 특정 소프트웨어 객체를 그 클래스로 구분할 수 있습니다. 따라서 클래스의 목적과 메시지를 알 수 있으며 보낼 수도 있습니다. 다시 말해서 클래스는 객체의 타입을 설명합니다.

두번째로 클래스는 인스턴스나 객체를 생성합니다. 클래스는 데이터 구조와 인스턴스의 동작을 정의하며 런타임 시 이 인스턴스를 생성하고 초기화시킵니다.

클래스와 클래스의 인스턴스의 주된 차이점은 데이터입니다. 인스턴스는 자신만의 고유한 데이터 셋을 가지고 있지만, 클래스는 엄격히 말하자면 데이터가 없습니다. 클래스는 인스턴스가 가지게 될 데이터의 구조를 정의하지만 인스턴스만이 데이터를 갖게 됩니다.

반면에 클래스는 실행 프로그램에서 클래스의 인스턴스의 모든 동작을 구현합니다. 객체를 나타내기 위한 도넛 형태의 심볼은 여기에는 적당하지 않습니다. 이는 각각의 객체가 자신만의 코드를 복사하여 가지고 있다는 것을 나타내기 때문입니다. 이것은 좋은 예가 아닙니다. 이 코드는 복제되는 것이 아니라 프로그램에서 현재 모든 인스턴스 간에 공유되어 있는 것입니다.

구분의 개념에서 절대적인 것은 클래스의 중요한 속성인 상속입니다. 클래스는 다른 클래스와 계층적인 관계를 맺으며 존재합니다. 서브클래스는 수퍼클래스로부터 동작과 데이터 구조를 상속받으며, 수퍼클래스는 자신의 수퍼클래스로부터 이들을 상속 받습니다.

클래스와 클래스의 기타 특징에 관한 더 자세한 사항은 부록, “Object-Oriented Programming”을 참조하십시오.

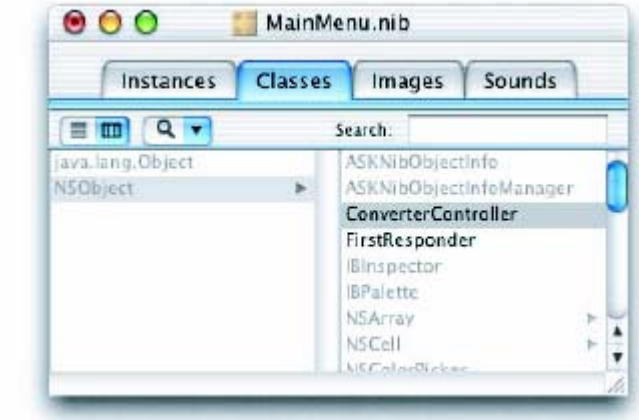
서브 클래스 정의

이전 장에서 클래스를 정의하기 위해서 nib 파일 윈도우의 클래스 디스플레이로 가야 한다는 것을 기억할 것입니다. ConverterController 클래스부터 시작해 봅시다.

1. Interface Builder 에서 MainMenu.nib 파일의 Classes display 를 선택하십시오.

2. 패인의 왼쪽 세로열에서 NSobject 를 클릭하고, Return 을 눌러 “MyObject”라는 NSObject 를 만드십시오.

그림 3-16 NSObject 서브클래스화하기



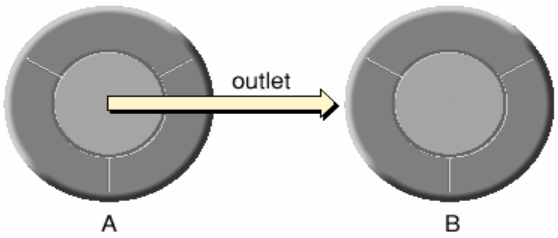
객체 커뮤니케이션을 위한 통로: 아웃렛, 타겟, 액션

Interface Builder 에서 ConverterController 객체와 다른 객체인 아웃렛과 액션 사이의 메시지 교환 통로를 지정할 수 있습니다.

아웃렛

아웃렛은 객체를 구별해주는 하나의 인스턴스 변수입니다.

그림 3-17 다른 도넛 모양을 가리키는 아웃렛



응용 프로그램에서 아웃렛으로 메시지를 보내서 다른 객체와 커뮤니케이션 할 수 있습니다.

아웃렛은 프로그램 내에서 텍스트 필드, 버튼, 윈도우, 패널과 같은 사용자 인터페이스 객체, 응용 프로그램 객체 자체까지 어떠한 객체라도 참조할 수 있습니다. 아웃렛을 구분해주는 것은 Interface Builder 와의 관계입니다.

아웃렛은 다음과 같이 선언됩니다.

IBOutlet id variableName;

id 를 객체의 타입으로 사용할 수 있습니다. id 를 객체 타입으로 가진 객체는 다이내믹하게 타입이 결정됩니다. 이는 객체의 클래스가 실행시 결정된다는 것입니다.

객체 타입을 다이내믹하도록 결정할 필요가 없다면 스태틱하게 객체의 포인터로 타입을 결정해야 합니다.

IBOutlet NSButton* myButton;

Interface Builder 는 코드에서의 선언으로 아웃렛을 인식하고 초기화할 수 있습니다. 일반적으로 Interface Builder 내에서 아웃렛의 값은 객체 사이에 연결 선을 이어 지정합니다. 응용 프로그램에서 아웃렛 외에도 객체를 참조하는 방법이 몇 가지가 있습니다. 그러나 아웃렛과 Interface Builder 의 초기화 능력은 대단한 강점입니다.

Interface Builder 에서 연결하기

모든 인스턴스 변수처럼, 아웃렛도 런타임시 적당한 값으로 초기화되어야 합니다. 이 경우, 객체의 식별자(id 값). Interface Builder 덕분에 프로그램은 nib 파일을 로드할 때 아웃렛을 초기화할 수 있습니다.

Interface Builder 에서 연결을 생성할 때 특별한 커넥터 객체가 소스와 연결할 대상 객체에 관한 정보를 가지고 있습니다. (소스 객체는 아웃렛 객체를 가진 객체입니다.) 그리고나서 이 커넥터 객체는 nib 파일에 저장됩니다. nib 파일이 로드되면 프로그램은 커넥터 객체를 사용하여 소스 객체의 아웃렛을 대상 객체의 id 값에 맞춥니다.

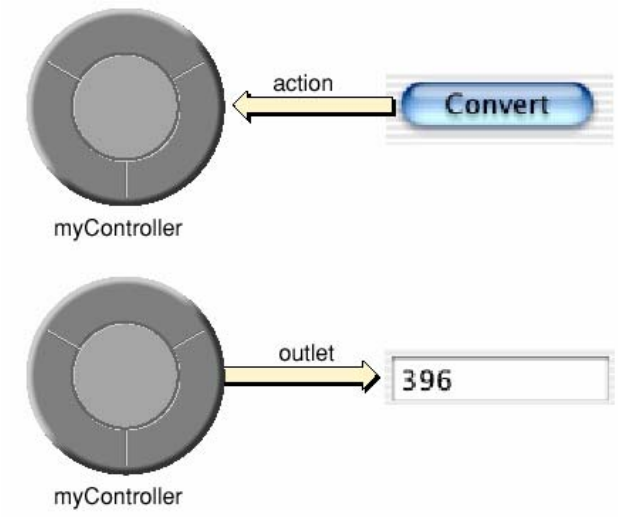
전기 콘센트(nib 파일 윈도우에서 선으로 연결하여 클래스를 디스플레이하는 방법처럼)를 상상하면 이러한 연결을 이해하는데 도움이 될 것입니다. 원본 객체에 있는 콘센트에서 전선을 연결하는 것을 떠올려 보십시오. 연결이 생성되기 전에 코드는 빠져있는 상태이며, 코드를 꽂아 연결이 생성되면 대상 객체의 id 값은 원본 객체의 콘센트로부터 할당됩니다.

Interface Builder 에서 Target/Action

곧 배우게 되겠지만, Interface Builder 의 Info Window 에서 target/action 연결을 볼 수 있습니다. 이 인터페이스는 사용하기 쉽지만 타겟과 액션의 관계는 그리 명백하지 않습니다. 먼저 타겟은 셀 객체의 아웃렛으로서 액션 메시지의 수신자를 식별합니다. 그렇다면 셀 객체란 무엇이며, 버튼과 무슨 관계에 있을까요?

하나 또는 더 많은 셀 객체들은 항상 컨트롤 객체(버튼과 같은 NSControl 에서 상속한 객체)와 연관되어 있습니다. 컨트롤 객체는 액션 메소드의 호출을 “drive”하지만, 셀로부터 타겟과 액션을 받습니다. NSActionCell 은 타겟과 액션 아웃렛을 정의하고, Application Kit 에 있는 모든 종류의 셀들은 이들 아웃렛을 상속합니다.

그림 3-18 타켓-액션 패러다임 관계



예를 들어, 사용자가 Currency Converter 의 Convert 버튼을 클릭하면 버튼은 그 셀로부터 필요한 정보를 얻고 메시지를 커스텀 클래스 ConverterController 의 인스턴스인 타겟 아웃렛으로 보냅니다.

Connection Info 윈도우의 Actions 컬럼에서 모든 액션 메소드는 타겟 객체의 클래스에 의해 정의되고, Interface Builder 에 의해 알려집니다. Interface Builder 는 다음의 신택스를 따르는 정의를 사용하기 때문에 액션 메소드를 식별합니다.

(void)doThis(id)sender;

이것은 특별히 sender 인자를 찾아내 줍니다.

어느 방향으로 연결할 것인가?

일반적으로 연결하는 아웃렛과 액션은 NSObject 의 커스텀 서브클래스에 속합니다. 이런 경우, Interface Builder 에 있는 연결선을 어느 방향으로 그릴 것인지 간단히 결정하는 방법이 필요합니다. 메시지가 흘러가는 방향으로 드래그하십시오.

- 액션 연결을 하기 위해서 사용자 인터페이스에 있는 컨트롤 객체(버튼 또는 텍스트 필드)에서 액션 메시지를 받을 커스텀 인스턴스로 선을 드로우 하십시오.

- 아웃렛 연결을 하기 위해서 커스텀 인스턴스에서 프로그램에 있는 다른 객체로 선을 드로우 하십시오.

이들은 일반적인 경우에 적용되는 방법으로, 모든 상황에서 적용할 수는 없습니다. 예를 들어, Cocoa 객체 중 많은 수가 대리 아웃렛을 가졌는데 이를 연결하기 위해서는 Cocoa 객체에서 자신이 직접 만든 커스텀 객체로 선을 연결해야 합니다.

연결을 명확히 하는 또 다른 방법은 누가 무엇을 찾는지 고려하는 것입니다. 커스텀 객체는 아웃렛을 이용하여 다른 객체를 찾으므로, 커스텀 객체로부터 다른 객체에 연결해야 합니다. 컨트롤 객체는 액션을 이용하여 커스텀 객체를 찾으므로 컨트롤 객체로부터 커스톰 객체에 연결해야 합니다.

클래스의 아웃렛 정의

1. 메소드 이름 convert 를 입력하고, Return 을 누르십시오. IB 는 ":"를 추가합니다.

2. Classes 윈도우에서 ConverterController 를 선택하십시오.

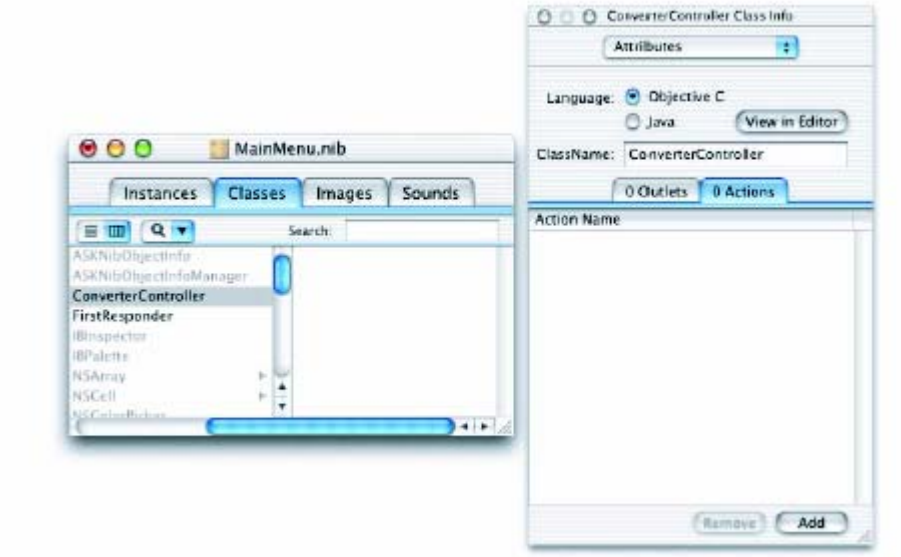
3. Classes 메뉴에서 Add Outlet 을 선택하십시오.

   a. Info 윈도우가 보이고, Attributes 패인이 선택되었는지 확인하십시오.

   b. " 0 Outlets" 라는 탭이 활성화되었는지 확인하십시오.

   c. Add 버튼을 클릭하십시오.

그림 3-19 Interface Builder Info 윈도우의 아웃렛과 액션



4. 이 아웃렛을 rateField 라고 이름을 바꾸고, Return 키를 누르십시오.

5. rateField 아웃렛이 여전히 선택되어 있기 때문에 더 많은 아웃렛을 만들기 위해서는 Return 을 누르기만 하면 됩니다. 이 과정을 반복하여 dollarField 아웃렛, totalField 아웃렛을 만듭니다.

아웃렛 테이블의 Type 세로열을 확인하십시오. 기본적으로, 아웃렛 타입은 id 에 설정되어 있습니다. Objective-C 는 다이내믹하게 언어를 지정하기 때문에 id 로 설정된 채로 있습니다. 그러나, 스태틱하게 지정한 인스턴스 변수는 훨씬 우수한 컴파일 타임 오류 검사를 받기 때문에 아웃렛 타입을 설정하는 습관을 갖는 것은 좋은 현상입니다. Id 로 설정된 콤보 박스 목록에서 id 를 선택하여 세 개의 아웃렛 유형을 NSTextField 로 변경하십시오.

ConverterController 는 인터페이스의 텍스트 필드를 액세스해야 하므로 이를 위해 방금 아웃렛을 만들었습니다. 그러나 ConverterController 는 Converter 클래스(아직 정의되지 않은)와도 소통해야 합니다. 이를 위해 ConverterController 에 converter 라는 아웃렛을 추가합니다.

클래스의 액션 정의

ConverterController 는 convert:라는 하나의 액션 메소드를 가지고 있습니다. 사용자가 Convert 버튼을 클릭하면 convert: 메시지는 ConverterController 의 한 인스턴스인 타겟 객체에 보내집니다. 액션은 사용자가 버튼을 클릭하거나 다른 컨트롤 객체를 작동할 때 객체에 보내는 메시지와 호출된 메소드를 모두 참조합니다.

1. Classes 메뉴에서 Add Action 을 선택하십시오.

   a. Info 윈도우가 보이고, Attributes 패인이 선택되었는지 확인하십시오.

   b. “ 0 Outlets” 라는 탭이 활성화되었는지 확인하십시오.

   c. Add 버튼을 클릭하십시오.

2. 메소드 이름을 convert 라고 입력하십시오. IB 가 :를 추가합니다.

ConverterController 를 인터페이스에 연결하기

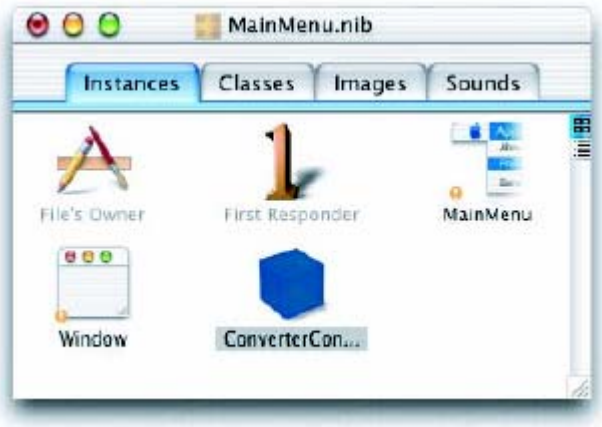
클래스의 인스턴스 생성

Interface Builder 에서 클래스를 정의하는 마지막 단계로서 클래스의 인스턴스를 생성하여 아웃렛과 액션에 연결해봅시다.

1. Classes 윈도우에서 ConverterController 가 선택되어 있지 않다면 이를 선택하십시오.

2. Classes 메뉴에서 Instantiate 를 선택하십시오. 그림 3-20 과 같이 Instances 뷰에서 해당 인스턴스가 하이라이트되어 나타날 것입니다.

그림 3-20 새롭게 인스턴스화된 ConverterController 객체



커스텀 클래스를 인터페이스에 연결하기

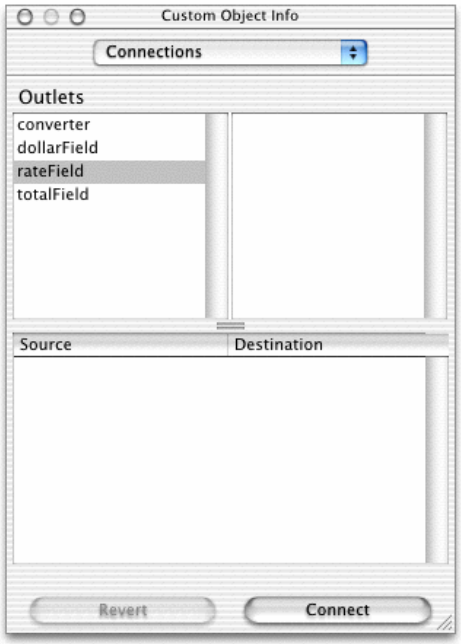
이제 ConverterController 객체를 사용자 인터페이스에 연결할 수 있습니다. 이를 인터페이스에 있는 특정 객체에 연결함으로써 아웃렛을 초기화할 수 있습니다. ConverterController 는 이들 아웃렛을 사용하여 인터페이스에서 값을 주고 받을 것입니다.

1. nib 파일 윈도우의 Instances 디스플레이에서, 그림 3-21 과 같이 ConverterController 인스턴스로부터 첫번째 텍스트 필드로 컨트롤을 누른 채로 드래그하여 선을 연결합니다. 텍스트 필드가 선택되면 마우스 버튼을 놓습니다.

그림 3-21 rateField 아웃렛 연결하기

2. Interface Builder 는 Info 윈도우의 Connections 디스플레이를 불러 옵니다. 여기에서 첫번째 필드(rateField)에 대응하는 아웃렛을 선택하십시오.

그림 3-22 rateField 아웃렛 선택하기



3. Connect 버튼을 클릭하십시오.

4. 같은 방법으로 ConverterController 의 dollarField 와 totalField 아웃렛도 적절한 텍스트 필드에 연결하십시오.

인터페이스 컨트롤을 클래스의 액션에 연결하기

1. Convert 버튼에서 nib 파일 윈도우에 있는 ConverterController 인스턴스로 컨트롤을 누른 채로 드래그하여 선을 연결합니다. 인스턴스가 선택되면 마우스 버튼을 놓습니다.

2. Connections 디스플레이에서 아웃렛 컬럼에 있는 타겟이 반드시 선택되도록 합니다.

3. Actions 컬럼에서 convert 를 선택하십시오.

4. Connect 버튼을 클릭하십시오.

5. nib 파일을 저장하십시오.

Converter 클래스 정의하기

ConverterController 의 아웃렛을 연결하는 동안 converter 아웃렛을 연결하지 않은 것을 알고 있을 것입니다. 이 아웃렛은 통화 변환기 응용 프로그램에 있는 Converter 클래스의 인스턴스를 식별하지만, 이 인스턴스는 아직 존재하지 않습니다.

이제 Converter 클래스를 정의해 봅시다. 이는 Converter 가 모델-뷰-컨트롤러(MVC) 패러다임 내에 있는 모델 클래스이기 때문에 매우 쉬운 작업입니다. 이러한 타입의 클래스 인스턴스는 인터페이스와 직접적으로 통신하지 않기 때문에 아웃렛이나 액션이 필요 없습니다. 다음은 완성 단계입니다:

1. Classes 디스플레이에서 Converter 를 NSObject 의 서브클래스로 만드십시오.

2. Converter 클래스의 인스턴스를 만드십시오.

3. ConverterController 와 Converter 사이에 아웃렛 연결을 만드십시오. Hint: ConverterController 인스턴스에서 Converter 인스턴스로 컨트롤을 누른 채로 드래그하십시오.

4. MainMenu.nib 파일을 저장하십시오.

통화 변환기의 클래스 구현하기

통화 변환기 프로그램 제작의 마지막 단계는 이전 단계에서 정의한 클래스를 구현하는 것입니다.

소스 파일 생성하기

1. nib 파일 윈도우의 Classes 디스플레이로 가십시오.

2. ConverterController 클래스를 선택하십시오.

3. Classes 메뉴에서 Create Files 를 선택하십시오.

4. .h 와 .m 파일 옆에 있는 Create 컬럼의 체크 박스가 선택되었는지 확인하십시오.

5. “Insert into CurrencyConverter” 옆에 있는 체크 박스가 선택되었는지 확인하십시오.

6. Choose 버튼을 클릭하십시오.

7. Converter 클래스에도 같은 방법을 적용하십시오.

8. nib 파일을 저장하십시오.

이제 Interface Builder 를 떠날 시간입니다. Project Builder 을 사용하여 프로그램을 마저 완성해 보겠습니다.

Objective-C Quick Reference

Objective-C 언어는 특별한 구문과 런타임 확장성을 지녀 객체 지향 프로그래밍을 가능하게 해주는 ANSI C 의 수퍼세트입니다. Objective-C 구문은 단순하면서도 강력한 힘을 발휘합니다. 아마도 표준 C 와 Objective-C 의 코드를 혼용할 수도 있을 것입니다.

다음은 언어의 더욱 근본적인 단면에 대해 요약하여 설명합니다. 더 자세한 내용은 Inside Mac OS X: The Objective-C Programming Language"를 참조하십시오.

메시지와 메소드 구현

메소드는 그 객체를 위한 클래스에 의해 구현됩니다.(또는 클래스 메소드의 경우에는 특정 인스턴스에 매이지 않고 기능을 제공) 메소드는 public 또는 private 의 속성을 가질 수 있는데, public 메소드는 클래스의 헤더 파일에서 정의됩니다.(위에서 보았듯이) 메시지는 객체의 메소드를 호출하며, 이름으로 메소드를 식별합니다.

메시지 표현은 호출하기를 원하는 메소드의 이름에 따라 받는 객체를 구별하는 변수로 이루어져 있습니다. 괄호 안에 표현을 담고 있습니다.

```
[anObject doSomethingWithArg:this ];
```

표준 C 언어처럼 문장은 세미콜론으로 종료됩니다.

메시지는 종종 호출된 메소드로부터 반환된 값을 도출합니다. 이 값을 왼쪽에 할당하기 위해서는 적절한 타입의 변수를 가지고 있어야 합니다.

```
int result =[anObj calcTotal ];
```

다른 메시지 표현 안에 메시지 표현을 중첩(nest)할 수 있습니다. 이 예제는 폼 객체의 윈도우를 얻어서, 반환된 NSWindow 객체를 다른 메시지의 receiver 로 만듭니다.

[form window] makeKeyAndOrderFront::self];

하나의 메소드는 함수 구조와 비슷합니다. 메소드의 선언 다음에 중괄호로 둘러 싸인 바디 구현부분이 오게 됩니다.

nil 을 사용하여 null 객체를 지정합니다:이는 null 포인터와 유사합니다. 어떤 Cocoa 메소드는 nil 을 아규먼트로 받아들이지 않습니다.

하나의 메소드는 유용한 두 개의 명시적인 식별자, self 와 super 를 참조합니다. 두가지 모두 메시지를 받는 객체를 식별하지만 메소드 구현이 이루어지는 장소가 다릅니다. self 는 메시지를 받는 클래스에서 검색을 시작하지만 super 는 그 수퍼클래스에서 시작합니다.

[super init];

위 코드는 수퍼클래스의 init 메소드를 호출합니다.

메소드에서 클래스 인스턴스의 인스턴스 변수를 직접 액세스할 수 있지만, 특별한 경우를 제외하고는 직접 액세스하는 대신에 액세스 메소드를 사용하는 것을 권장합니다.

선언

객체를 id 로 선언하여 다이내믹하게 타입을 지정:

id myObject;

다이내믹하게 타입이 지정된 객체의 클래스는 타입이 런타임시 결정되므로, 코드 상에서 이전에 어떤 클래스에 속했는지 알 필요 없이 참조할 수 있습니다. 만약 다형성과 다이내믹 바인딩에 관련되어 있다면 아웃렛과 객체의 타입을 이러한 방식으로 지정하십시오.

클래스에 포인터 타입의 객체를 설정합니다.

```
NSString *mystring;
```

객체의 타입을 스태틱하게 설정하여 컴파일 시간 타입 체크를 쉽게 하고 코드를 이해하기 쉽게 만들 수 있습니다.

인스턴스 메소드의 선언은 -(마이너스) 기호로 시작하며, - 기호 다음에 오는 스페이스는 없어도 상관없습니다.

- (NSString *)countryName;

-(또는 +) 기호와 메소드 이름이 시작되는 사이의 괄호 안에 메소드에 의해 반환되는 값의 타입을 넣으십시오.(위의 예시처럼) 반환값이 없는 메소드는 반환형이 void 이어야 합니다.

메소드 인자(argument)형은 괄호 안에 오며 인자의 키워드와 인자 사이에 위치합니다.

```
- (id)initWithName:(NSString *)name andType:(int)type;
```

모든 선언은 세미콜론으로 종결해야 한다는 것을 기억하십시오.

기본값으로 인스턴스 변수의 영역은 보호되며, 이 변수를 선언한 클래스나 그 서브클래스의 객체에만 직접적인 접근이 가능합니다. 인스턴스 변수를 private(선언한 클래스에서만 접근 가능)으로 만들려면 선언하기 전에 @private directive 를 넣으면 됩니다.

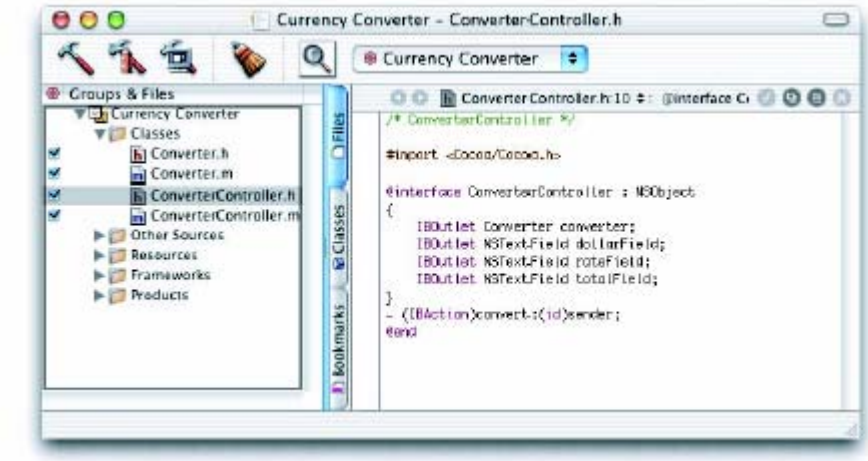
Project Builder 에서 인터페이스(헤더) 파일 살펴보기

Interface Builder 가 헤더와 소스 파일을 CurrencyConverter 프로젝트에 추가할 때 그리고 파일이 Interface Builder 에서 추가되었을 때 선택된 어느 그룹에서든 소스 파일이 나타납니다. 이들 파일은 클래스 구현이므로 Classes 그룹에 추가시킵니다.

1. Project Builder 의 메인 윈도우를 클릭하여 활성화시키십시오.

2. 프로젝트 브라우저에서 네 파일 모두를 선택하여 Classes 그룹으로 드래그합니다.

그림 3-23 Currency Converter 의 Classes 그룹의 소스 파일



메소드 선언 추가하기

Interface Builder 가 생성한 헤더 파일에 인스턴스 변수나 메소드 선언을 추가할 수 있습니다. 일반적으로는 이것을 수행하지만, ConverterController 의 경우에는 필요하지 않습니다. 그러나 ConverterController 객체가 결과값을 얻기 위해 호출할 수 있도록 Converter 클래스에 메소드를 추가할 필요가 있습니다. Converter.h 에 메소드를 선언하여 시작해 봅시다.

1. 프로젝트 브라우저에서 Converter.h 를 선택하십시오.

2. convertAmout:atRate:의 선언을 삽입하십시오.

```
#import <Cocoa/Cocoa.h>

@interface Converter:NSObject

{

}

- (float)convertAmount:(float)amt atRate:(float)rate;

@end
```

이 선언을 convertAmout:atRate:가 부동형의 인자 두 개를 취하고, 부동 값을 반환한다는 것을 나타냅니다. convertAmout:와 atRate:처럼 메소드 이름에 콜론이 붙으면 인자를 소개하는 키워드입니다.(이들은 “C”언어와는 다른 의미의 키워드입니다.)

이제 두 개의 구현 파일을 업데이트해야 합니다.

통화 변환기의 클래스 구현하기

Converter 클래스를 위해 바로 전에 Converter.h 에 선언한 메소드를 구현하십시오. 메소드 구현은 @implementation 〈class name〉과 @end 사이에 오게 됩니다. 다음과 같이 Converter 에 코드를 추가하면 됩니다.

1.Project Builder 의 메인 윈도우에 있는 Classes 그룹에서 Converter.m 을 선택하십시오.

2.convertAmount:에 대한 코드를 삽입하십시오.

```
#import "Converter.h "

@implementation Converter

- (float)convertAmount:(float)amt atRate:(float)rate

{

return (amt *rate);

}

@end
```

메소드는 단순히 두 인자를 곱해서 결과를 반환합니다. 아주 간단합니다.

1. 다음으로 Interface Builder 가 생성한 ConverterController.m 에 있는 convert:메소드의 “empty” 구현을 업데이트하십시오.

```
- (IBAction)convert:(id)sender
{
float rate, amt, total;
amt = [dollarField floatValue];
rate = [rateField floatValue];
total = [converter convertAmount:amt atRate:rate];
[totalField setFloatValue:total];
[rateField selectText:self];
}
```

2. ConverterController.m 의 소스 파일 제일 위에 다음 코드를 추가하여 Converter.h 를 임포트하십시오.

```
#import "Converter.h."
```

convert: 메소드는 다음의 역할을 합니다.

- Rate 와 dollar-amount 필드로 형식이 지정된 부동 포인트 값을 얻습니다.
- convertAmount:atRate:메소드를 호출하고 반환 값을 얻습니다.
- setFloatValue:를 사용하여 다른 Currency 텍스트 필드(totalField)에 반환 값을 적습니다.
- selectText:를 rate 필드에 보냅니다. 텍스트 필드에서 어떤 텍스트를 선택했거나 텍스트가 선택이 안되었어도 커서를 삽입하여 사용자가 다른 계산을 수행할 수 있게 합니다.

Convert: 메소드의 각 라인(float 선언을 제외하고)은 메시지입니다. 메시지 표현의 왼쪽에 있는 “word”는 메시지를 받는 객체 “receiver”를 식별합니다. 이들 객체는 여러분이 정의하고 연결한 아웃렛에 의해 식별됩니다. Receiver 이후에 보내는 객체(“sender”)가 호출하기 원하는 메소드의 이름이 옵니다. 메시지는 종종 결과값을 반환합니다; 위의 예제에서 지역 변수 rate, amt, total 은 이들 값을 갖게 됩니다.

awakeFromNib 메소드 구현하기

프로젝트를 빌드하기 전에 ConverterController.m 에 약간의 코드를 추가하여 사용자들이 조금 더 쉽게 사용할 수 있게 만들어 보겠습니다. 응용 프로그램이 시작될 때, 통화 변환기의 윈도우가 선택되어 커서가 “Exchange Rate per $1”에 오도록 할 것입니다. 이는 nib 파일이 언아카이브되어 텍스트 필드 rateField 에 연결이 성립된 후라야만 가능합니다. 셋업 과정을 이렇게 가능하게 하기 위해서는 언아카이브가 종결될 때 AwakeFromNib 메소드가 모든 객체로 보내져야 합니다. 적절한 액션을 취하기 위해 이 메소드를 구현하십시오.

1. 다음 코드를 ConverterController.m 에 추가하십시오.

```
- (void)awakeFromNib
{
[[rateField window] makeKeyAndOrderFront:self];
[rateField selectText:self];
}
```

makeKeyAndOrderFront: 메시지는 그 이름에 맞는 역할을 합니다. 이는 수신 윈도우를 key 윈도우(키보드로부터 입력을 받는 윈도우)로 만들어 주며 화면 상에서 다른 모든 윈도우보다 앞에 둡니다. 이 메시지는 또한 다른 메시지, [rateField window]에 중첩할 수 있습니다. 이 메시지는 텍스트 필드가 속하는 윈도우를 반환하고, 그리고 나서 makeKeyAndOrderFront: 는 반환된 객체에 보내집니다.

이전에 convert: 구현에서 selectText: 메시지를 확인했습니다. 이는 텍스트 필드에서 텍스트를 선택하여 메시지를 반환하고 텍스트가 없으면 커서를 삽입합니다.

Project Builder 로 강력한 코딩하기

Project Builder 로 코드를 작성하면, "작업대"와 같은 기능을 하는 툴들을 마음껏 사용할 수 있습니다.

Project Find

Project Find (Project Builder 의 Find 패인에서 사용)를 사용하면 프로젝트 코드와 시스템 헤더를 찾을 수 있습니다. Project Find 는 디스크에 모든 프로젝트 식별자(클래스, 메소드, 함수, 상수 등)를 저장하는 프로젝트 인덱스를 사용합니다. , 기타 심볼에 대한 참조나 정의에 대해 검색할 수 있게 해줍니다.

C 기반 언어의 경우, Project Builder 는 소스 파일이 컴파일되는 동안 자동으로 인덱스 정보를 수집하여, 개발자가 Project Find 를 사용하기 전에 인덱스를 만들 수 있는 프로젝트를 구축합니다.

통합 문서 보기

Project Builder 는 응용 프로그램에서 직접 HTML 문서를 볼 수 있도록 지원합니다. 사용자는 문서 및 Project Builder, 다른 개발자 툴, Carbon, Cocoa, AppleScript Studio 에 관한 릴리즈 노트, 그리고 UNIX 맨 페이지에 접근할 수 있습니다.

또한, 전체 또는 부분적으로 완성된 식별자에서 개발자 문서 및 헤더 파일로 바로 갈 수 있습니다. HTML 문서를 검색하려면, 옵션 키를 누른 채 이중 클릭하고, 헤더 파일 선언을 검색하려면, 커맨드 키를 누른 채 이중 클릭하십시오.

들여쓰기

Preferences 에서 자동적인 들여쓰기가 일어나는 캐릭터, 들여쓰기 당 스페이스 수, 기타 전체 들여쓰기 특성을 지정할 수 있습니다. Edit 메뉴는 Indentation 서브메뉴를 포함하고 있어 상황에 따른 들여쓰기나 코드 블록을 가능하게 해줍니다.

괄호 검사

중괄호({},왼쪽이나 오른쪽 모두 상관 없음)를 이중 클릭하여 대응하는 중괄호를 알 수 있습니다. 중괄호 사이의 코드가 하이라이트됩니다. 비슷한 방법으로 메시지 표현에서 대응하는 []와 ()을 찾기 위해 []와 ()을 이중 클릭하여 알 수 있습니다. 대응하는 구분자가 없다면 Project Builder 가 경고음을 냅니다.

Emacs 바인딩

Project Builder 의 코드 에디터에서 통용되는 대부분의 Emacs 명령어를 사용할 수 있습니다.(Emacs 는 코드 작성을 위한 대중적인 에디터입니다.) 예를 들자면 page-forward (컨트롤-v), word-forward (Meta-f), delete-word (Meta-d), kill-forward (컨트롤-k), yank from kill ring (컨트롤-y) 명령어가 있습니다.

어떤 Emacs 명령어는 표준 매킨토시 바인딩의 어떤 부분과는 충돌 가능성이 있습니다. 코드 에디터가 다른 명령어 키(Option, Shift, Control 과 같은)를 Emacs 의 Control, Meta 키로 대체하기 위해 사용하는 키 바인딩을 수정할 수 있습니다. 커스텀 키 바인딩에 대한 안내는 다음을 확인하십시오.

통화 변환기의 빌드, 디버그, 실행하기

이 섹션은 응용 프로그램의 빌드, 디버그, 실행 방법에 대해 설명합니다.

응용 프로그램 빌드할 때 무엇이 일어나는가

Project Builder 에서 Build 버튼을 클릭하여 빌드 툴을 실행하십시오. 디폴트 빌드 툴은 jam 이지만, Project Builder 에서 설정을 바꾸어 어떤 빌드 유틸리티라도 사용할 수 있습니다. 빌드 툴은 컴파일과 링크 과정을 거쳐 실행 파일을 생성합니다. 또한 프로그램을 빌드하기 위해 필요한 다른 일도 수행합니다.

빌드 툴은 컴파일러를 호출하여 프로젝트의 소스 코드 파일을 전해줍니다. 이들 파일을(Objective-C, C++, 표준 C 언어) 컴파일하면 프로그램 구조를 위한 기계어 객체 파일 또는 빌드하기 위한 특정 프로그램의 구조를 생성합니다.

빌드의 링크 단계에서, 빌드 툴은 링커를 실행하여 라이브러리와 프레임웍을 객체 파일과 링크하여 줍니다. 프레임웍과 라이브러리는 어느 응용 프로그램에서라도 사용할 수 있는 프리 컴파일된 코드를 포함하고 있습니다. 링킹은 라이브러리, 프레임웍, 객체 파일의 코드를 통합하여 응용 프로그램의 실행파일을 생성합니다.

빌드 툴은 또한 프로젝트로부터 응용 프로그램 패키지의 적당한 위치로 nib 파일, 사운드, 이미지, 기타 리소스를 복사합니다. 응용 프로그램 패키지는 실행 응용 프로그램과 응용 프로그램의 실행에 필요한 리소스를 포함하고 있는 디렉토리입니다. 이 디렉토리는 Finder 에서 하나의 파일로 나타나게 되며 이중 클릭으로 프로그램을 실행할 수 있습니다.

프로젝트 빌드하기

Project Build 패널에서 빌드를 시작할 수 있습니다.

1. 소스 코드 파일과 변경 사항을 프로젝트에 저장하십시오.

2. 그림 3-24 와 같이 메인 윈도우 상의 Build 버튼을 클릭하십시오.

그림 3-24 Project Builder 의 Build 버튼

Build 버튼을 클릭하면, 빌드 과정이 시작됩니다. Project Builder 가 에러 없이 과정을 완료하면 “Build succeeded”라는 메시지가 프로젝트 윈도우의 왼쪽 하단에 나타납니다.

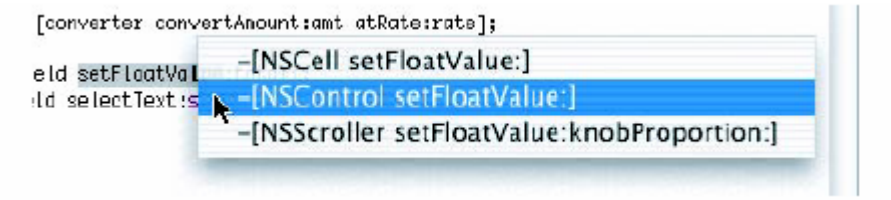
참조 문서

Project Builder 가 프로젝트를 구축할 때, 프레임웍과 관련 문서가 포함된 소스 파일의 인덱스를 작성합니다. Project Builder 내에서 바로 문서 및 헤더 파일에 접근할 수 있습니다. 다음을 따라 해보십시오.

1. ConverterController.m 으로 가십시오.

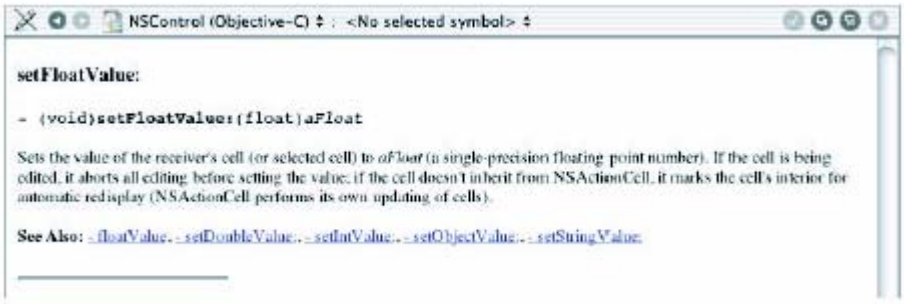
2. setFloatValue 를 옵션 키를 누른 채 이중 클릭하십시오. -[NSControl setFloatValue:] 라는 메소드 이름을 가진 팝업 메뉴가 나타납니다.

그림 3-25 문서 팝업 메뉴



3. NSControl 과 관련된 메소드를 선택하십시오. Project Builder 는 에디터 패인에서 바로 HTML 양식의 문서를 열고, 요청한 메소드에 대한 설명을 제공합니다.

그림 3-26 Project Builder 내 문서

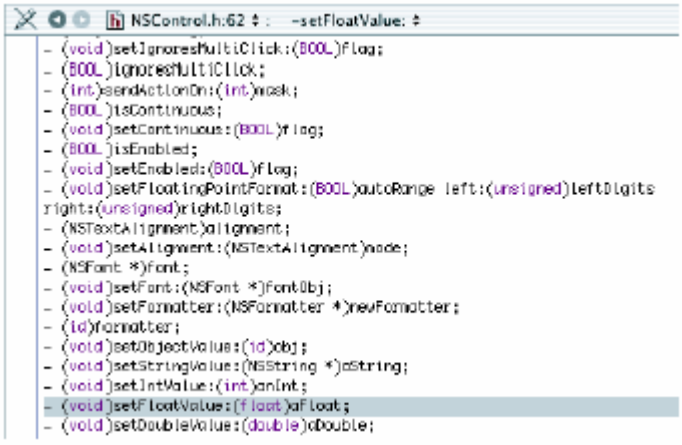


4. Back 버튼(에디터 패인 상단의 왼쪽 화살표)을 클릭하여 코드로 가십시오.

5. 같은 코드를 커맨드 키를 누른 채 이중 클릭하십시오. 유사한 팝업 메뉴가 나타납니다.

6. 다시, -[NSControl setFloatValue:]메소드를 선택하십시오. 이 때, Project Builder 는 관련 헤더 파일의 메소드 선언으로 안내합니다.

그림 3-27 Project Builder 내 헤더 참조



7. Back 버튼을 클릭하십시오.

팝업 메뉴가 나타나지 않으면, 프로젝트는 올바르게 인덱싱되지 않았습니다. 프로젝트를 인덱싱하려면, 다시 구축하십시오.

프로젝트 디버그하기

물론 처음부터 프로젝트가 완벽하게 실행되는 경우는 드뭅니다. Project Builder 는 프로젝트를 처음으로 빌드할 때 몇 가지 에러가 발생하기 쉽습니다. Project Builder 의 에러 체크 기능으로 잘못된 코드를 확인하십시오.

1. 코드에서 세미 콜론을 삭제하여, 에러를 만드십시오.

2. Project Build 패널에서 Build 버튼을 클릭하십시오.

3. 빌드 에러 브라우저에 나타난 에러 발생을 알려 주는 라인을 클릭하십시오.

4. 코드에서 에러를 바로 잡으십시오.

5. 프로젝트를 다시 빌드하십시오.

그래픽 디버거 사용하기

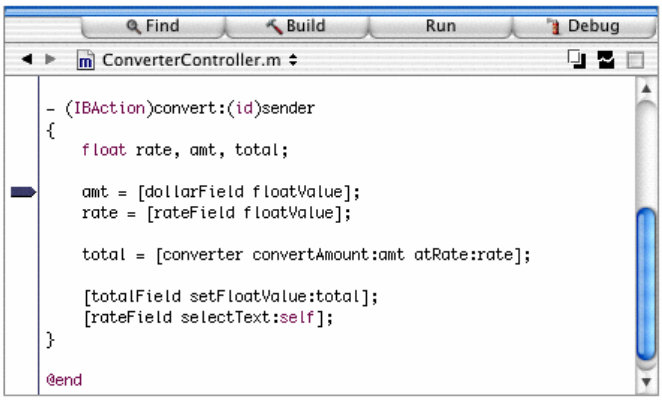
디버깅 작업을 순조롭게 하기 위해서 Project Builder 는 GNU 디버거, GDB 를 통해 그래픽 유저 인터페이스를 이용합니다. 디버거와 친숙해지기 위해(모든 과정이 통화 변환기 프로젝트처럼 쉽게 진행되지는 않을 것입니다.) 통화 변환 코드를 추가하는 각 단계마다 사용하시기 바랍니다.

1. Project Builder 의 Group 과 Files View 에 있는 ConverterController.m 를 클릭하십시오.

2. convert: 메소드를 찾으십시오.

3. 코드 리스트의 왼쪽에 있는 컬럼을 클릭하여 브레이크 포인트를 지정하십시오.

그림 3-28 Project Builder 의 브레이크포인트 설정



4. Debug 버튼이나 커맨드-R 을 눌러 디버거를 실행하십시오.

5. 값을 입력하고 Convert 버튼을 클릭하여 변환을 테스트 해보십시오.

6. Convert 를 클릭하면 디버거가 활성화되어 코드의 줄 단위에 따른 각각의 단계별 결과를 살펴볼 수 있습니다.

복잡한 디버깅 작업을 수행하는 데에 GDB 콘솔을 사용할 수 있습니다. 화면의 우측 상단에 있는 Console 탭을 클릭하여 콘솔을 나타나게 하십시오. Console 윈도우를 클릭할 때, 리턴을 누르면 (gdb)프롬프트가 나타납니다.

사용자 인터페이스에는 나타나지 않는 프롬프트에서 많은 GDB 명령어를 입력할 수 있습니다. 이들 명령어에 관한 온라인 정보를 보려면 프롬프트에서 “help”를 누르면 됩니다.

통화 변환기 실행하기

몇 가지 환율과 달러를 입력하고 Convert 를 클릭하십시오. 또한 필드에서 텍스트를 선택하고, Application 메뉴에 있는 Services 서브 메뉴를 선택하십시오: 이 메뉴는 이제 선택된 텍스트로 무언가를 할 수 있는 다른 응용 프로그램을 나열합니다.

물론 프로그램이 더욱 복잡해짐에 따라 더 철저히 많은 테스트를 거쳐야 할 것입니다. 전체적인 설계, 인터페이스, 커스텀 클래스 정의나 메소드와 함수의 구현을 변경해야 할 에러나 결점을 발견할 수 도 있습니다.

통화 변환기는 단순한 응용 프로그램이지만 이전 챕터에서 소개한 모든 개념과 테크닉을 통합하였습니다. 이제까지 Cocoa 응용 프로그램을 개발하기 위해 필요한 기술에 대해 잘 이해할 수 있었을 것입니다. 배운 것을 함께 정리해 봅시다:

- Interface Builder 로 그래픽 유저 인터페이스(GUI) 구성하기
- 인터페이스 시험하기
- 모델-뷰-컨트롤러 패러다임을 이용하여 응용 프로그램 설계
- 클래스의 아웃렛과 액션 지정
- 클래스 인스턴스를 아웃렛과 액션을 통해 인터페이스에 연결
- 클래스 구현 기초
- 응용 프로그램 빌드와 에러 해결

4 장. Cocoa 리소스

이 장에서는 Cocoa 를 학습하는 중에 도움을 줄 수 있도록 개발자 문서, 웹 사이트, 이메일 목록, 관련 서적 등 수 많은 리소스를 제공합니다.

Cocoa 개발 문서

Apple 의 Cocoa 개발 문서에는 Cocoa API 에 대한 참조 문서가 들어 있습니다. 또한, 다양한 프로그래밍 도메인에 대한 자료 및 특정 작업이 포함된 Programming Topic 이 들어 있습니다. Cocoa 프로그래밍을 처음 접하시는 분들은 " 시작하기" 섹션을 참조하십시오.

다음과 같은 방법으로 Cocoa 개발 문서를 참조하실 수 있습니다.

■ Project Builder 의 Help 메뉴 혹은 매킨토시 도움말의 개발자 도움말 센터에서 Cocoa Help 를 선택하십시오.

■ 하드 디스크에서 /Developer/Documentation/Cocoa 를 검색하십시오.

■ 최신 버전의 문서를 제공하는 http://developer.apple.com/techpubs/macosx/Cocoa/CocoaTopics.html 을 방문하십시오.

■ Project Builder 의 Find 패인에서 소스 파일 식별자, 개발자 문서, 헤더 파일 등을 검색하십시오. Find 패인은 Single Window 모드의 메인 윈도우나 Find 메뉴에서 Show Batch Find 를 선택하여 이용할 수 있습니다.

■ Project Builder 에디터 패인에서 식별자를 옵션 키를 누른 채 이중 클릭하여 문서를 검색하고, 커맨드 키를 누른 채 이중 클릭하여 관련 헤더 파일을 찾으십시오.

또한, Project Builder 에서 바로 프로젝트 내 프레임웍을 찾을 수 있습니다. Cocoa 개발자들은 Groups & Files 패인의 타켓 그룹에 위치한 Foundation.framework >Headers 와 AppKit.framework>Headers inside Frameworks > Other Frameworks 를 검색하십시오.

개발 툴에 대한 도움말

Apple 은 Mac OS X 에서 Project Builder, Interface Builder, 커맨드 라인 툴 등 수 많은 개발자 툴용 도움말 문서를 제공합니다.

Project Builder 에 관한 도움말은 Project Builder 의 Help 메뉴에서 Project Builder Help 를 선택하십시오.

Interface Builder 에 관한 도움말은 Interface Builder 의 Help 메뉴에서 Interface Builder Help 를 참조하십시오.

다음과 같이 개발자 툴을 제공합니다.

■ Project Builder 의 Help 메뉴 혹은 매킨토시 도움말의 개발자 도움말 센터에서 Developer Tools Help 를 참조하십시오.

■ 하드 디스크에서 /Developer/Documentation/DeveloperTools 를 검색하십시오.

■ 최신 버전의 문서를 제공하는 http://developer.apple.com/techpubs/macosx/DeveloperTools/devtools.html을 방문하십시오.

또한, Project Builder 와 Interface Builder 는 툴 팁이라는 도움말 태그를 제공하여 마우스 화살표가 그 부분 위에 있으면 인터페이스에 관한 설명을 제공합니다.

메일링 리스트

Apple 은 수 많은 메일링 리스트를 지원하여 개발자와 사용자가 다양한 주제로 토론할 수 있도록 합니다. cocoa-dev 메일링 리스트는 초보부터 전문가에 이르기까지 Cocoa 개발자들을 위한 우수한 리소스입니다. studentdev 리스트는 Cocoa 를 포함한 Apple 제품 개발에 관해 학생들이 토론할 수 있는 장을 제공합니다.

다음 주소에서 메일링 리스트에 등록하여 리스트 아카이브를 검색할 수 있습니다.

http://lists.apple.com

관련 서적

O'Reilly & Associates 는 Cocoa 개발에 관련된 두 권의 서적을 출판하였습니다. 한 권은 Apple 에서 저술했습니다.

- Apple Computer,Inc 의 Learning Cocoa
- Simson Garfinkel 및 Michael K.Mahoney 의 Building Cocoa Applications: A Step-by-Step Guide

Cocoa 개발과 관련된 여러 권의 서적들이 출판되었으며, 향후에도 출판될 예정입니다. 서점의 프로그래밍 섹션을 방문해보십시오.

기타 리소스

- ADC 는 Cocoa 개발자들을 위해 유용한 기술 및 비즈니스 서비스를 제공합니다. 개발자 문서, 샘플 코드, 개발자 프로그램 정보 및 비즈니스 관련 지원은 http://developer.apple.com을 방문하십시오.

■ Cocoa 개발과 관련된 써드 파티 웹 사이트가 있습니다. Google 을 검색하여 웹 상에서 Cocoa 리소스를 찾아보십시오.

http://www.goole.com/search?hl=en&lr=&q=cocoa+programming