ADC
Apple Developer Connection
Direct

Cocoa: A New Flavor to Mac OS Development

Mac OS X integrates five separate application runtime environments—Carbon, Cocoa, Classic, Java, and BSD—into one seamless whole, providing developers with many options. This article introduces Cocoa, an important development technology used by Apple, to develop many of the applications that ship with Mac OS X. Cocoa constitutes a new, highly efficient way for developers to create new products for Mac OS X.

A Brief Overview

Cocoa is an object-oriented application framework and runtime environment—a set of software components used to construct applications that run on Mac OS X. Think of Cocoa as a large set of reusable application building blocks that can be used as delivered or extended for your specific needs.

Cocoa is based on OpenStep, an object-oriented technology that was originally introduced in 1987 as NeXTSTEP and has been refined through many iterations. Consequently, Cocoa is mature technology based on a design years ahead of other object-oriented frameworks.

Like any software library, Cocoa has a learning curve for newcomers, but it is not an overly difficult one and productivity comes quickly. The Cocoa framework delivers a great deal of fundamental application functionality so you can spend the majority of your energy working on application features rather than managing the more tedious parts of system interface and user experience implementation.

The primary implementation language used in Cocoa is Objective-C, a superset of ANSI C with specific language features added to allow for object-oriented programming. The language extensions in Objective-C are compact and easy to learn. Cocoa applications make extensive use of the classes and methods in Cocoa component libraries. However, because of Objective-C's compatibility with ANSI C, they can also make use of core functionality contained in traditional C and C++ libraries brought from other application environments.

Features and Development Scope

Cocoa is a peer to the Carbon and Java application development environments in Mac OS X. Cocoa supports all Mac OS X application service features. For example, Cocoa applications can access the Mac OS X native imaging and printing model, multimedia standards QuickTime and OpenGL, and Internet and BSD services, too. Localization and Internationalization are also well supported by Cocoa. The separation of user interface elements from executable code allows you to package your applications for different locales easily, with no code changes. All Cocoa text drawing utilizes the Unicode standards. The text and font systems are particularly flexible and allow you to use sophisticated word processing features with little effort.

An important development advantage that Cocoa offers is the capability to develop programs quickly and easily by assembling reusable components. With reusable components, developers can not only create applications, but also produce:

- Frameworks (sophisticated library structures)
- Bundles of executable code and associated resources which can be loaded and executed dynamically
- Collections of custom user-interface objects

These capabilities support the easy creation and distribution of application plug-ins and extensions.

Tools and Resources

Apple put the tools for Cocoa development directly into the hands of developers by including them in every Mac OS X package on the *Mac OS X Developer Tools CD*. Just install the Developer.pkg file and the complete Apple tool suite is installed and ready for use. This installation includes the Project Builder Integrated Development Environment, Interface Builder—the application for designing and testing user interfaces and establishing the connections between objects and actions—as well as all the interface files, debugging and performance tools, and full online documentation.

Once you have installed the Developer.pkg, open a Finder window and you will find the /Developer directory on your system volume. Inside this directory, in /Applications, are the Project Builder and Interface Builder applications along with many other tools. In /Developer/Documentation are folders containing PDF and HTML help and documentation files for Cocoa and the developer tools.

Finally, in /Developer/Examples are /AppKit and /Foundation sample code folders with Cocoa applications for you to learn from and even reuse for your own project.

Taking Apart the Technology

Cocoa is comprised of two object-oriented frameworks: Foundation

Godfrey DiGiorgi is Technology Manager for Development Tools & Cocoa in Apple Worldwide Developer Relations. He's been associated with Macintosh development for more years than he'd care to admit. He can be reached at ramarren@apple.com.

(Foundation.framework) and Application Kit (AppKit.framework). The Foundation classes provide the low-level objects and functionality that form the basis of the Cocoa environment. The classes in Application Kit provide the functionality users see in the user interface, which respond to system events such as mouse clicks and key presses. The Application Kit is layered directly on Foundation. Here is a brief look at the functionality contained in each of these frameworks.

The Foundation Framework is designed to provide a set of basic utility classes, introduce consistent conventions for paradigms (such as memory management) support Unicode strings, object persistence, and file management. Foundation includes:

- the root object class
- classes representing basic data types such as strings and byte arrays
- collection classes for storing other objects
- classes representing system information such as dates and
- classes representing communication ports

Several paradigms are also defined in Foundation to help avoid confusion in common situations and introduce consistency across class hierarchies. This is done with some standard policies, like the one used for object ownership (answering questions such as: "Who is responsible for disposing of an object?"), and also with abstract classes which enumerate over collections. These paradigms reduce special and exceptional cases in code management and allow reuse of the same mechanisms with various kinds of objects. All together, these paradigms improve development efficiency and productivity.

The Application Kit framework contains all the objects needed to implement the graphical, event-driven user interface: windows, panels, buttons, menus, scrollers, text fields, etc. The Application Kit handles all the details for you as it efficiently draws on the screen, communicates with hardware devices and screen buffers, clears areas of the screen before drawing, and clips views. There are over a hundred classes in the Application Kit, so it might seem a steep learning curve, but many of the Application Kit classes are support classes that are only used indirectly.

For a detailed listing of the Foundation and Application Kit object classes, see the documentation in:
/Developer/Documentation/Cocoa/CocoaTopics.html.

Object-Oriented Programming

For traditional Mac programmers, Cocoa development represents a paradigm shift—from a procedural to an object-oriented development model. The "free" functionality found in the Foundation and Application Kit frameworks helps Mac programmers realize the benefits of this easy, natural way to develop applications.

Object-oriented programming allows the construction of complex applications through assembling small, well-tested, reusable modules called objects. This provides three simple advantages:

1. Greater reliability by breaking complex implementations into smaller, easily testable components.
2. Easier maintainability due to the small, modular nature of objects. (This small size allows one to fix bugs found in testing more easily.)
3. Greater productivity through reuse. The ability to use an existing class over and over again means less redundant work. When you need to extend the functionality of a particular class to meet a specific need, you can do so easily through the use of a mechanism called inheritance. You only need to code the specific functionality you are adding to a class, the rest of the object's behavior is inherited from the preexisting parent class. For more information, see the documentation in:
/Developer/Documentation/Cocoa/ObjectiveC/index.html.

Fundamentals of Cocoa Programming

Cocoa is a rich object-oriented environment. Object-oriented programming makes heavy use of patterns to simplify design and implementation of complex systems. The three most essential patterns to learn when starting are:

1. Model-View-Controller (MVC) - MVC defines three types of objects in an application: model, view, and controller. Model objects hold data and define the logic that manipulates that data. View objects represent user interface elements (a window, for example). Controller objects act as mediators between model objects and view objects. This mediation role allows view objects to be free from the programmatic interfaces of models and vice-versa.
2. Target/Action - This is part of the mechanism by which user interface controls respond to user actions. When a user clicks a user interface control, the control sends an action message to the target object.
3. Delegation - Delegation lets you modify an object's behavior without creating a custom subclass. A delegate acts on behalf of another object. When a delegate receives a message (from a window, a view, etc.), the sender of the message is allowing the delegate to influence its behavior and aid in decision-making (such as: "Should I allow the user to close me?").

Cocoa leverages the dynamic binding and object introspection capabilities of Objective-C and Java by allowing delegate objects to implement just the functionality they want to influence. At runtime, the delegating objects can query their delegates to see what methods have actually been implemented. This saves you from having to subclass some specific parent class (that has all the default implementations), helping to preserve your application's unique class hierarchy.

Continued on page ?? ☛

Project Builder Tips & Tricks

- You can use Build Styles instead of duplicate targets for many things that would require duplicate targets in other environments.
- All text fields in Project Builder that contain file paths support path completion (completion is bound to the F5 key by default).
- Any place where single clicking an item loads that item into the built-in editor, double-clicking will open the item in a separate window. This includes the files list, bookmarks, targets, build results, find results, etc.

New Mac OS X Related Releases

The following software is available from the Download Software area of the ADC Member Site at:
http://connect.apple.com/

• **CarbonLib 1.3d9 SDK**
The latest prerelease version of the CarbonLib 1.3 SDK for Mac OS
• **Mac OS USB Driver Development Kit 1.5**
http://developer.apple.com/hardware/usb/download.htm
• **FireWire 2.8.1 Software Developers Kit (SDK)**
http://developer.apple.com/hardware/FireWire/Developer_Info.html#FireWireSDK

Developer Documentation

• **O'Reilly and Apple Collaborate on Mac OS X Book Series**
O'Reilly & Associates, has announced plans to publish a series of books about Mac OS X development. The books in this series have been technically reviewed by Apple engineers and are recommended by the Apple Developer Connection. The first Mac OS X titles, *Learning Carbon* and *Learning Cocoa*, are available this May. In addition, the O'Reilly Network has established the Mac DevCenter, a web forum for development news and articles.
• **Apple Technical Publications**
Over thirty new and updated documents have been added in the last month to help developers with successful Mac OS X application and peripheral development at:
http://developer.apple.com/techpubs/

TN2015 - Locating Application Support Files Under Mac OS X
TN2014 - Insights on OpenGL
TN2013 - The 'plst' Resource
TN2012 - Building QuickTime Components for Mac OS X

QA1019 - Can't attach during two-machine debugging with GDB
QA1018 - Using AppleScript to send an email with an attachment
QA1013 - Mac OS X and root access

SC - More than thirty new Mac OS X code samples were posted to the ADC web site since the last issue. Please visit the URL below for a complete list of sample code.
http://developer.apple.com/samplecode/

"Built for Mac OS X" Artwork Now Available

Now that customers have Mac OS X in their hands, they'll be looking for great products to run on it. Tell the world that your product runs on Mac OS X by displaying the "Built for Mac OS X" badge on your product's packaging. The artwork, licensing requirements, and usage guidelines are available on the ADC Software Licensing web site.
http://developer.apple.com/mkt/swl/agreements.html#macosx

Upcoming Seminars and Events

For more information on Apple developer events please visit the developer Events page at:
http://developer.apple.com/events/

Training and Seminars

• ***Apple iServices: Cocoa Development Classes***
This five-day course provides comprehensive, hands-on training using real-world examples. With the skills acquired in this course, developers can build full-featured applications using the most advanced software environment on Mac OS X.
http://www.apple.com/iservices/technicaltraining/cocoadev.html
• ***Programming With Cocoa***
Taught by Aaron Hillegass at the Big Nerd Ranch, Ashville, NC and Atlanta, GA. Five-day classes are taught on developing web-based and Mac OS X applications.
http://www.bignerdranch.com/when.html

Developer Related Conferences

•***Worldwide Developers Conference (WWDC) 2001, San Jose, CA***
May 21-25
Register now for Apple's Worldwide Developers Conference 2001, which takes place in San Jose, California from May 21-25. ADC Premier members receive one free pass to the conference. For schedules and other details check out:
http://www.apple.com/developer/wwdc2001/

• ***MacHack Conference, Dearborn, MI***
June 21-23
MacHack, in its sixteenth year, remains centered around cutting edge software development. MacHack's uniqueness derives from the informal feel and the LIVE coding that occurs around-the-clock during the conference.
http://www.machack.com/

Cocoa: A New Flavor to Mac OS Development

Continued from page ??

These patterns are discussed at length in *Inside Cocoa: Object-Oriented Programming and the Objective C-Language* and in other parts of the Cocoa documentation.

Another integral part of Cocoa programming practice is the use of Interface Builder. Interface Builder is a design tool, allowing you to easily define and test a user interface. It is also used with Cocoa to define object classes and "wire" the connections of targets and actions. Interface Builder creates 'nib' files, which are a static representation of objects and their relationships. These nib files are efficiently loaded as needed at runtime. Interface Builder is closely tied to the Project Builder IDE for a smoothly integrated development experience. For more information on Interface Builder and Project Builder, see the documentation available in:
/Developer/Documentation/DeveloperTools
as well as on the Web at:
http://developer.apple.com/tools/projectbuilder/
http://developer.apple.com/tools/interfacebuilder/

Language Support

Cocoa is implemented in Objective-C. As a superset of ANSI C with special syntax and runtime extensions, Objective-C lets you use object-oriented programming techniques while leveraging as much of the use and knowledge of standard ANSI C as possible.

Java can also be used to implement Cocoa applications through the use of the Java API versions of the Foundation and Application Kit frameworks. See the Foundation and Application Kit reference locations.

Where to Go for More Information

- Start with the Cocoa technology web page. It includes News and Updates as well as a list of Resources for Cocoa development.

http://developer.apple.com/cocoa/

- Read Inside Cocoa: Object-Oriented Programming and the Objective-C Language.

http://developer.apple.com/techpubs/macosx/Cocoa/CocoaTopics.html
http://www1.fatbrain.com/documentation/apple/

- Sign up for one of Apple's development-oriented mail lists, such as Cocoa Developer, ProjectBuilder-User, or Java Developer.

http://lists.apple.com

- Check out the mailing lists for Mac OS X and Objective- C developers supported by the Omni Group.

http://www.omnigroup.com/

- Look for the O'Reilly book, Learning Cocoa, due in May, 2001.

http://www.oreilly.com/catalog/learncocoa

In Summary

The demand for Mac OS X applications is huge and Cocoa can help you bring new products to market quickly, using the full power of Mac OS X development tools and object-oriented methodology to facilitate your work.

Whatever Mac OS X development path you choose, Apple is eager to assist you. For the latest Mac OS X news and information, visit the Apple Developer Connection web site at:
http://developer.apple.com/macosx

Did You Know?

Learning Cocoa—and Mastering It

To programmers new to Cocoa--Apple's powerful object-oriented application environment--the road to mastery is a challenging yet rewarding one. Although there are many new things to learn, once you become comfortable with Cocoa, your programming productivity will take off. Guaranteed.

To help you in your Cocoa apprenticeship, Apple provides two great sources of technical information. The first is the book *Learning Cocoa*, published by O'Reilly. Written by insiders at Apple Computer, Learning Cocoa mixes conceptual overviews with hands-on tutorials to give you a crash course in Cocoa application development. The idea is that learning Cocoa should not be just a matter of reading, but doing. *Learning Cocoa* guides you through the creation of several applications, each more complex than the one before. By the end of the book, you'll be prepared to take on serious application development on your own. Look for*Learning Cocoa* in technical bookstores near you; you can also purchase it direct from O'Reilly at:
http://www.oreilly.com/catalog/learncocoa/

The second source of information on Cocoa is Apple's own technical publications, especially its Cocoa programming topics. The programming topics are a hierarchically organized collection of information nodes on such topics as implementing undo, custom drawing, and managing text. Each node brings together (in an HTML frame set) the conceptual, procedural, and reference documentation that illuminates a single programming task. In addition to documentation, a programming topic includes links to example projects, technical notes, and other sources of information. If you have the Developer package installed, you can access the Cocoa programming topics through the Developer Help Center. You can also view them on the ADC Developer Documentation web site at:
http://developer.apple.com/techpubs/macosx/Cocoa/CocoaTopics.html