Xcode 4.2

Includes SDKs for Mac OS X 10.7 Lion and iOS 5

Contents

Introduction

Xcode is the complete developer toolset for creating applications for Mac, iPhone, and iPad. This package installs the Xcode IDE, the Instruments analysis tool, iOS Simulator, and OS framework bundles in the form of Mac OS X SDKs and iOS SDKs.

What's New in Xcode 4.2 for Lion

- Support for Mac OS X 10.7 Lion and iOS SDK 5
- Apple LLVM compiler 3.0 with Automatic Reference Counting (ARC)
- Storyboarding support in Interface Builder to design multi-view workflows for iOS
- OpenGL ES graphical debugger within the main Xcode debugging interface
- LLVM compiler support for C++'0x features using the new LLVM libc++ standard library
- Additional bug fixes, stability, and performance improvements

What's New in Xcode 4

- Xcode 4 has a brand new, single window interface for all major workflows
- Interface Builder is now integrated within the main Xcode IDE
- Assistant shows a paired editor with complementary files, e.g.: header or UI controller
- Live Issues display coding errors as you type, and Fix-it can correct the mistake for you

Compatibility: Xcode 4 requires an Intel-based Mac running Mac OS X 10.7.0 Lion or later, and includes Mac OS X SDK 10.7 and 10.6, and iOS SDK 5. To develop apps targeting prior versions of Mac OS X or iOS, see the section titled About SDKs and the iOS Simulator below.

Developer Resources: The Mac and iOS Developer Programs provide access to the App Store, additional support and documentation, as well as provisioning resources to enable testing and deployment on an iPhone, iPod touch, or iPad device. For more information visit:

for iOS: http://developer.apple.com/programs/ios/
for Mac: http://developer.apple.com/programs/mac/

For discussions about any Apple developer software, including pre-release products, please visit the Apple Developer Forums, at: http://devforums.apple.com/

Top new features in Xcode 4

- Xcode 4 has a brand new, single window interface for all major workflows
- Interface Builder is now integrated within the main Xcode IDE
- Assistant shows a paired editor with complementary files, e.g.: header or UI controller
- Version editor shows a live comparison through SCM history, using Git or Subversion
- Live Issues display coding errors as you type, and Fix-it can correct the mistake for you
- Tabs save a unique screen layout for each task, e.g. debugging, editing, designing
- LLVM compiler 2.0 includes full support for C, Objective-C, and now C++
- LLDB debugger is faster, and uses less memory than the GDB debugging engine
- Instruments adds System Trace, and new iOS instruments including OpenGL ES

Feature overview

- Xcode 4 uses a completely new, single window interface for organizing your project. It is now possible to create a workspace that can contain multiple projects, as well as references to files elsewhere on disk. Common Xcode operations such as search and replace will execute across all files within a workspace. Xcode 4 does not require the use of a workspace, and is compatible with Xcode 3.2 projects.
- The Jump Bar is a path control at the top of the editor windows that allows you to click at any location within the bar and quickly jump to peers of that folder or file.
- The Navigator pane on the left of the window, opened by the menu View -> Navigators, contains access to all the active information in your current project(s). Source files, build errors, logs, debugging stack traces, and more are collected here.
- The Utility pane on the right of the window, opened by the menu View -> Utilities, contains file inspectors, Quick Help for the selected text, and libraries of file templates, code snippets, Interface Builder objects, and your system media.
- Interface Builder is now integrated into the Xcode window. Selecting a .xib/.nib file via the Jump Bar or Project Navigator will open the IB editor. File and object inspectors are available via the View -> Utilities menu item. It is now possible to drag connections from objects on the design canvas directly to code in the accompanying source editor. A small dialog will ask if you wish to create an action or outlet, or you can select existing code to receive the connection. The interface will be connected for you, and the code stub will be generated, if necessary, at the point where you dragged the connection.
- Assistant editor (View -> Editor menu, or toolbar button) will split the Xcode 4 single editor window into a pair of editor panes, with your primary document on the left, and a complementary file on the right. You can use the button on the far left of the Jump Bar at the top of the editor pane to tell the Assistant what type of files you would like to see: counterparts, sub/superclasses, actions or outlets, or others. The Assistant can also auto-select files you may wish to see, as you are coding.
- Version editor (View -> Editor menu, or toolbar button) is part of the completely rebuilt source control management (SCM) support of Xcode 4. The Version editor makes it easy to compare two versions of a file, see commit logs, check blame, and even zoom back along the commit timeline. Xcode's SCM support also adds support for branching

and merging, as well as full support for Git to go along with Subversion. The repository browser is now located in the Organizer window, along with documentation and device management.

- Fix-it works in conjunction with Xcode build errors to present you not just the error, but also an option to immediately fix the problem without writing any code. If a Fix-it is available, clicking the error indicator beside the code will show a pop-up window listing the available fixes. Pick the appropriate fix, and Xcode will repair the mistake for you.
- Live Issues detect and alert you to coding errors as you type; there is no need to build the project first. Live Issues also work with Fix-it, making it easy to see your mistake, correct it, and continue coding without missing a beat.
- The Product -> Analyze menu item can be thought of as “super warnings”, detected by using Xcode’s built-in static analyzer. The Analyze command will do a deep inspection of your code, walking down most possible code paths, to identify potential coding mistakes and logical errors. Found problems can be viewed in the editor as message bubbles which, when clicked, will display arrows that walk through the steps that can create the error. The static analyzer performs an in-depth analysis of application behavior, and is a great complement to the Live Issues and Fix-it features.
- Schemes allow you to preconfigure the build, deploy, run, and test products and configurations for each of the most common tasks during development. For instance, the default scheme setup will configure your project to build in Debug mode when you execute the Run command (CMD-R), allowing you to debug by simply turning on breakpoints. The same scheme will build for Release when you perform the Profile or Archive command. Schemes are setup for you automatically when you create a new project, and you can use the “Manage Schemes” menu to set them up however you like.
- New optional LLVM compiler 2.0 uses the much faster Clang front-end parser coupled with the LLVM back-end compiler for the fastest available compile time, and best performing generated code for C/C++ and Objective-C. For maximum compatibility with GCC use the default LLVM GCC 4.2 compiler which employs the LLVM back-end for superior code generation, but uses the GCC front end parser.

  NOTE: to choose the LLVM compiler, click on your project item in the Project navigator, choose a project or target, and select the Build Settings tab. The build option “C/C++ Compiler Version” is used to select which compiler to use.
- New optional LLDB debugger is available to replace the default GDB debugging engine, providing a faster launch time, and less memory overhead. To enable LLDB, from the Xcode menu bar select Product -> Edit “My Scheme”. The default “Launch” scheme has an option for Debugger which can be selected to GDB (default) or LLDB, to be run when Xcode launches the debugger.
- Instruments now offers the ability to set a time filter, and to zoom in/out with accuracy to the nanosecond. New keyboard shortcuts include:

  - Time Filter: (option + mouse click and drag)
  - Zoom in: (shift + mouse click and drag)
  - Zoom out: (control + mouse click and drag)

- Instruments’ new System Trace template configures to simultaneously record system calls, virtual memory operations, and scheduling events. The System Trace template combined with the new Thread Strategy view makes understanding the behavior of your overall system even easier.

Workflows using the single-window interface of Xcode 4

Xcode 4 includes new features to make your workflows more automated, and easier to manage on screen: tabs, editor panes, and behaviors. These features work together to scale the single window interface of Xcode 4 to effectively use available screen real estate, from the smallest MacBook Air, up to Macs equipped with multiple monitors.

- **Editors:** The built-in editor modes, Standard, Assistant, and Version, are designed to present the most critical information, in context, as you work. In addition to these standard editor layouts, clicking the icon on the far-right of the Jump Bar will split the current editor pane. The split editors have their own history, and Assistant editor splits can each present counterparts as you work in the primary editor. The General preferences pane configures document open behaviors, such as option-click to open in the Assistant editor, or double-click to open the document in a new window or tab.

- **Tabs:** Much like in Safari, each Xcode 4 tab is an independent view of the complete work area. Selecting File -> New Tab (or pressing CMD-T) will create a new tab view of your current workspace, with its own Navigator/Utility geometry, debug area, and editor layout. Tabs and their customized layout are preserved between launches of Xcode 4, and double clicking the tab's label will give it a permanent name. For computers with a lot of screen space, tear a tab off and move it wherever you like. Configure a tab for Interface Builder design tasks, debugging activities, code editing, and more.

- **Behaviors:** Certain tasks are common during development, among them Build, Test, and Run. The Behaviors preferences panel configures specific actions or changes in state within the IDE when you perform these tasks. For example, when a Run starts, Xcode can automatically switch to a tab called Debug (or the IDE will create the tab for you). As described above, each tab can be independently configured, making it easy to trigger a different Xcode layout depending on the behavior active at the time.

About SDKs and the iOS Simulator

An SDK is a collection of frameworks (libraries, headers, and resources) that represent the API for a specific Mac OS X or iOS version. Most of the functionality your app gets from an SDK is actually provided by the host operating system, so the right **Base SDK** and **OS Deployment Target** settings are critical for app compatibility. Conveniently, Xcode automates this configuration for you, building with the latest SDK and targeting the latest OS by default.

If your app doesn't require the latest OS features, you can configure it to run on a previous Mac OS X or iOS version via the **OS Deployment Target** option in the Project settings. If your project was created in a much older version of Xcode, you can let Xcode update your project, which will automatically set **Base SDK** to "Latest" in your build settings.

For iOS, Xcode will automatically switch between the iOS Simulator SDK and the device SDK, depending on where you intend to run your app. You don't need to set this manually.

NOTE: for final qualification on a specific version of iOS you must test on a device.

For the latest security information visit: **http://support.apple.com/kb/HT1222**
For more detailed information please see the complete Xcode release notes.

Installation

The Xcode installer will create the folder `/Developer` on the root of your boot partition, and place the Xcode developer tools and SDKs in this folder. If you have a previous version of Xcode you would like to preserve, you can copy the existing installation from `/Developer` to another folder to prevent having it updated automatically by the installer.

NOTE: The Xcode installer also installs system components and UNIX command line tools in a shared location on your Mac. Only one version of these components can exist on a computer at a time, and the last installed version replaces any previously installed set.

By default, Xcode will download developer documentation in the background for offline reading, and automatically download documentation updates as well. This behavior can be changed after install via Preferences in Xcode.

Uninstalling Xcode Developer Tools

To uninstall Xcode developer tools on the boot volume along with the <Xcode> directory, from a Terminal window type:

```
$ sudo <Xcode>/Library/uninstall-devtools --mode=all
```

To remove the underlying developer content on the boot volume, but leave the <Xcode> directory and supporting files untouched, from a Terminal window type:

```
$ sudo <Xcode>/Library/uninstall-devtools --mode=systemsupport
```

To just remove the UNIX development support on the boot volume, but leave the <Xcode> directory and supporting files untouched, from a Terminal window type:

```
$ sudo <Xcode>/Library/uninstall-devtools --mode=unixdev
```

Finally, to just uninstall the <Xcode> directory you can simply drag it to the trash, or from a Terminal window type:

```
$ sudo <Xcode>/Library/uninstall-devtools --mode=xcodedir
```

NOTE: The uninstaller that ships with previous versions of the Xcode developer tools will not clean everything off of your system properly. You should use the one installed with these Xcode developer tools.

Deprecation Notice

- Distributed builds using distcc are deprecated as of Xcode 4.2. Future versions of Xcode will cease to include this feature. Moving your project to the Apple LLVM compiler will take

advantage of the latest advancements in accelerating build times. Live Issues and Fix-its will help you find and fix problems in your code without the need to build.

- Plug-ins for Interface Builder 3.x (IBPlugins), for integrating custom controls and inspectors, are not supported within Xcode 4. The Xcode 4 design canvas, inspectors, library, and other integrated Interface Builder functionality cannot be used with these plugins to edit existing, or design new .xib/.nib files. Projects that contain existing files originally designed with these plugins continue to build in Xcode 4. If you need to actively support these .xib/.nib files, it is recommended that you also install Xcode 3.2, as described in the Custom Install section.