Copyright ©1996 by NeXT Software, Inc. All Rights Reserved.

ProjectTypes;¬Project Types

Project Builder can create these types of projects:

Application€ A standalone OPENSTEP application.

Tool€ A server or UNIX command-line tool.

Loadable Bundle A directory of resources, such as images, sounds, character strings, nib files, and dynamically loadable executable code, to be used by one or more applications.

Library A static or dynamic shared library.

Framework A bundle that contains a dynamic shared libraryplus resources. See ªFrameworks: Easy to Use, Easy to Createº in this chapter. ;CreatingManagingConcepts.rtfd;FrameworksEasytoUseEasytoCreate;¬

Palette A static Interface Builder paletteÐa palette with code that you must compile before it can be used.

Legacy A project for which Project Builder doesn't maintain the makefile. Use this when you have created your own makefile. See ªLegacy Projectsº in this chapter. ;CreatingManagingConcepts.rtfd;LegacyProjects;¬

Aggregate A collection of loosely related projects. See ªGrouping projectsº in this chapter. ;GroupingProjects.rtfd;;¬

For the most part, the only difference between project types is the kind of executable they produce. However, there are some special issues involved in frameworks and libraries. See Chapter 12 for more information. ;../../05_SpecialTasks/12_FrameworksLibraries/FrameworksLibraries.rtfd;;¬

28012_TableRule.eps ¬

ManagingProjectFilesWithProjectBuilder;¬Managing Project Files With Project Builder

Project Builder makes it easy for you to manage a project's files. It organizes your project for you by grouping

files into the following types (not all types are available for all projects):

Classes The ªmº files that implement an Objective-C class.

Headers The ªhº files that define a class's interface or declare C functions, data types, and variables.

Documentation The documentation files for framework projects, which must be RTF files.

Context Help The help files for this project, which must be RTF files. This only exists for application projects and subprojects.

Other Sources Objective-C files that don't implement a class or files containing code in other languages, such as C, C++, Objective-C++, or PostScript.

Interfaces Interface Builder nib files that define the user interface. Nib files are described further in Chapter 2. ;../../02_CreatingTheInterface/02_Composing/Composing.rtfd;;¬

Images Image files, such as TIFF or EPS files, other than icons.

Other Resources Files containing other resources (such as sound files or eomodels) that the project uses.

Subprojects Subprojects. See ªGrouping projectsº in this chapter. ;GroupingProjects.rtfd;;¬

Supporting Files Makefiles and other files the project does not use directly.

Libraries Libraries that the project links to, such as those in **/usr/lib** or **/lib**.

Frameworks Dynamic shared librariesthat the project links to, such as the Application Kit. OPENSTEP-supplied frameworks are in **/NextLibrary/Frameworks**. (See ªFrameworks: Easy to Use, Easy to Createº in this
chapter. ;CreatingManagingConcepts.rtfd;FrameworksEasytoUseEasytoCreate;¬)

Non Project Files Files that you have opened that aren't part of the project.

In addition, Project Builder creates and maintains some files for you. For all projects, it creates a makefile and templates for a class's interface (ªhº) and implementation (ªmº) files. For application projects, it also creates

a *Project*_**main.m** file, which contains the **main** function, and a nib file, which defines your application's interface. You can customize both of these files for your application.

Of course, you can add your own files to the project. You can also remove files, rename files, create new files, and even open files that aren't part of the project using commands on the File menu.

MainWindow1.eps ¬

28012_TableRule.eps ¬

TheProjectServer;¬The Project Server

Indexing is actually performed by a background process called the project server. The project server is the brains behind Project Builder. When you request information stored in the index, Project Builder asks the project server for that information, then relays it to you.

The project server starts up as soon as you open a project. In just a short time, project server can build a cache of symbol information about your project.

The project server is a continually running process. Even after you quit Project Builder, the project server may continue
running. This saves time at startup; Project Builder only has to start up a project server when you reboot.

When you set the Host attribute on the Preferences panel, the project server runs on that host. If other people on the network use the same computer for their project servers, one project server is created per user.

If you need to control the project server, use the commands on the Indexing menu under the Project menu. The command Purge Indices kills the current project server. After you use this command, use Index Source Code to start a new project server and reindex your project.

28012_TableRule.eps ¬

FrameworksEasytoUseEasytoCreate;¬Frameworks: Easy to Use, Easy to Create

Frameworks are new in OPENSTEP 4.0. A framework is just a simpler, more convenient way to package a dynamic shared library and the resources associated with it.

In previous releases, you had to install essential library components in three different locations: the library file in **/usr/local/lib**, header files in **/LocalDeveloper/Headers**, and documentation in **/LocalLibrary/Documentation**. Now, you can store all three of these components in one directory, the framework directory, which you install in **/LocalLibrary/Frameworks**. Plus, you can package other resources (such as nib files and images) in your framework.

All OPENSTEP kits, including the Application Kit, are now distributed as frameworks. You can find these in **/NextLibrary/Frameworks**.

Another big advantage to frameworks is that Project Builder can see inside the framework package. For example, if you select **AppKit.framework** in Project Builder's browser, you can access its header files and documentation. This means you can look up how to use a method without ever leaving Project Builder!

To find out how to create your own framework, see Chapter 12 in this book.
;../../05_SpecialTasks/12_FrameworksLibraries/FrameworksLibraries.rtfd;;¬

28012_TableRule.eps ¬

LegacyProjects;¬Legacy Projects

If Project Builder can't understand a project's Makefile, it decides that the project is a *legacy project*, a project created using
a previous release. Project Builder won't create or maintain the Makefile for legacy projects; you must do that yourself.

You don't have to use the legacy project type every time you want to control the Makefile. Instead, you can make your changes in the

Makefile.preamble or **Makefile.postamble** files. The project Makefile includes **Makefile.preamble** at the beginning of the build and **Makefile.postamble** at the end of the build. Project Builder won't overwrite these files, so making changes to them is safe.

For more information about Makefiles, see Chapter 9 in this book.
;../../04_BuildingDebugging/09_Building/Building.rtfd;;¬