1 Creating and Managing a Project

I have a bit of FIAT in my soul,
And can myself create my little world.
Thomas Lovell Beddoes

But from these create he can
Forms more real than living man,
Nurslings of immortality!
Percy Bysshe Shelley

I must Create a System, or be enslav'd by another Man's;
I will not Reason and Compare: my business is to Create.
William Blake, *Jerusalem*

Creating a project

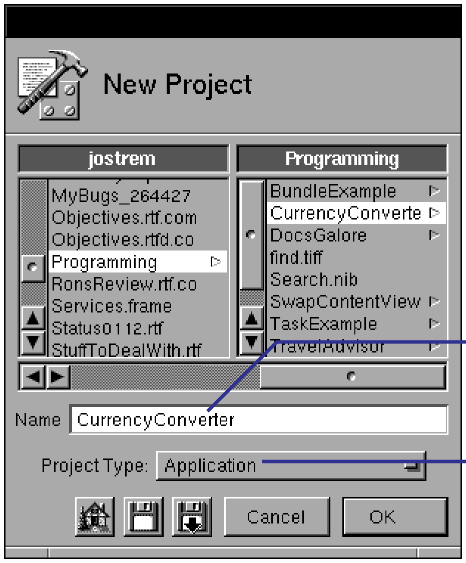
1 **In Project Builder, choose New from the Project menu.**

2 **In the New Project panel, choose the project's type from the Project Type pop-up list.**

3 **Name the project.**

4 **Choose OK.**

A *project* is a set of files that produces a given end product, such as an application, a tool, a library, or a loadable bundle. When you create a project in Project Builder, you create a directory that will hold all of the project's code files and resource files. Project Builder adds several supporting files, such as project makefiles and templates that you can use to create source files, to that directory.



Name: CurrencyConverter

Project Type: Application

*Project Builder creates a directory named **CurrencyConverter** and places the project's supporting files in it.*

Choose the project type from this list. Remember to choose the project type; you can't change the type of an Application project later.

Project Types

Project Builder can create these types of projects:

Application A standalone OPENSTEP application.

Tool A server or 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 library plus resources. See "Frameworks: Easy to Use, Easy to Create" in this chapter.

Palette A static Interface Builder palette—a 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 "Legacy Projects" in this chapter.

Aggregate A collection of loosely related projects. See "Grouping projects" in this chapter.

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.

Use the Open command on the Project menu to open a project that already exists. In the Open panel, select the project's folder.

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 ".m" files that implement an Objective-C class.

Headers The ".h" 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.

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 "Grouping projects" in this chapter.

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 libraries that the project links to, such as the Application Kit. OPENSTEP-supplied frameworks are in **/NextLibrary/Frameworks**. (See "Frameworks: Easy to Use, Easy to Create" in this chapter.)

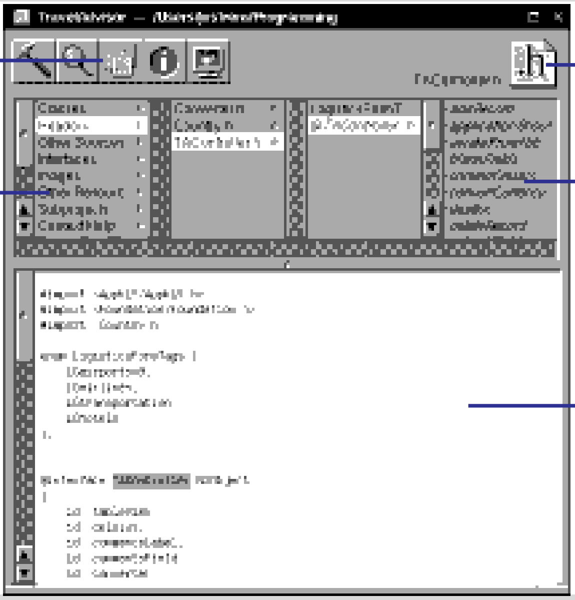
Non Project Files Files that you have opened that aren't part of the project.

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 (".h") and implementation (".m") 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.

This button showas you a list of the files that are already loaded. You can use that list to jump between files quickly.

To add a file, double-click one of these types.



```
#import <AppKit/AppKit.h>
#import <Foundation/Foundation.h>
#import "Country.h"

enum LogisticsFormTags {
    [illegible]
    [illegible]
    LGtransportation
    LGhotels
};


@interface TAController : NSObject
{
    id  tableView;
    id  celsius;
    id  commentsLabel;
    id  commentsField;
    id  [illegible]
```

This is shaded when the file has been modified.

The browser shows the files in the project, the classes in that file, and the methods in that class.

The code editor shows the currently selected file and allows you to edit it.

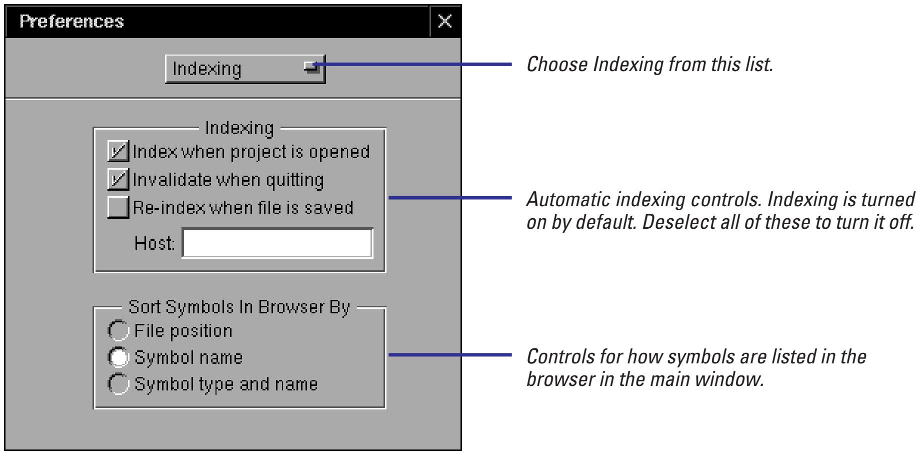
Setting indexing preferences

1 **Choose Info ▶ Preferences.**

2 **In the Preferences panel, choose Indexing from the pop-up list.**

3 **Select or deselect the preferences you want.**

For you to take full advantage of Project Builder, the source code must be indexed. When a project is indexed, Project Builder keeps track of each symbol in the project, what that symbol defines (such as a class or a function), where it is declared, and where it is used. The Indexing Preferences panel allows you to control when and how a project is indexed.



Preferences

Indexing

Indexing
- [✓] Index when project is opened
- [✓] Invalidate when quitting
- [ ] Re-index when file is saved

Host:

Sort Symbols In Browser By
- File position
- Symbol name
- Symbol type and name

Choose Indexing from this list.

Automatic indexing controls. Indexing is turned on by default. Deselect all of these to turn it off.

Controls for how symbols are listed in the browser in the main window.

By default, “Index when project is opened” and “Invalidate when quitting” are selected. These preferences cause Project Builder to create a new index in memory each time it is opened and to delete the index when you quit the application. (This ensures that the index is updated at least when you quit Project Builder.) If you deselect “Invalidate when quitting,” the project index will persist until you reboot.

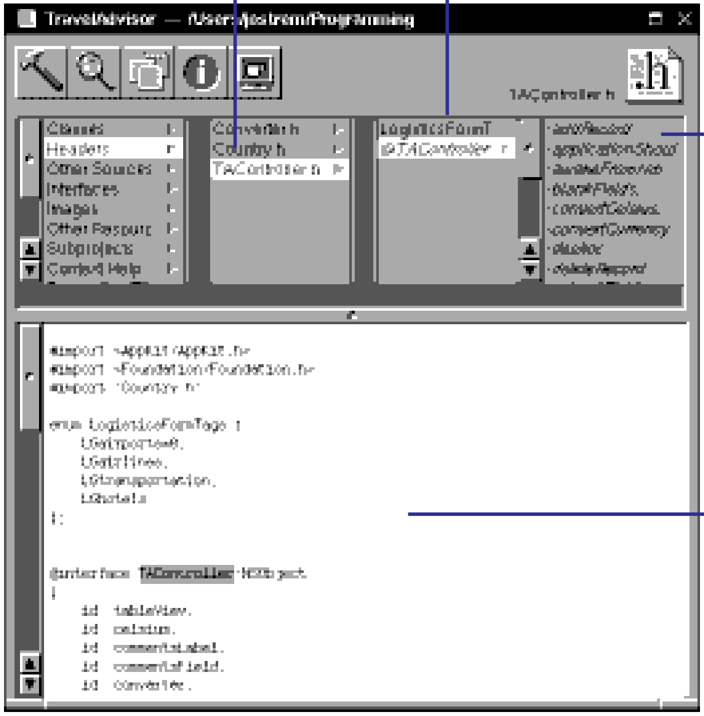
Indexing requires overhead. To improve performance, you can turn indexing off; however if you do this, many useful features, such as project-wide name completion and language-based searching, aren’t available. Another option is to have the indexing process run on another computer on your network. To do this, enter that computer’s host name in Host field of the Indexing Preferences panel.

Chapter 9, Building, describes the Build Attributes inspector and how to build the project on another host.

Tip: If the project indexes incorrectly, make sure that you have set the information in the Project Attributes and Build Attributes inspectors. Use the same host to index the project as you use to build the project.

When you index, the browser can show you information about a file's contents.

The classes, protocols, and other symbols the selected file defines.



```
#import <AppKit/AppKit.h>
#import <Foundation/Foundation.h>
#import "Country.h"

enum LogisticsFormTags {
    LGairports=0,
    LGairlines,
    LGtransportation,
    LGhotels
};


@interface TAController:NSObject
{
    id  tableView;
    id  celsius;
    id  commentsLabel;
    id  commentsField;
    id  Converter;
```

The methods defined for that class or protocol.

The editor jumps to the class or method you select.

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.

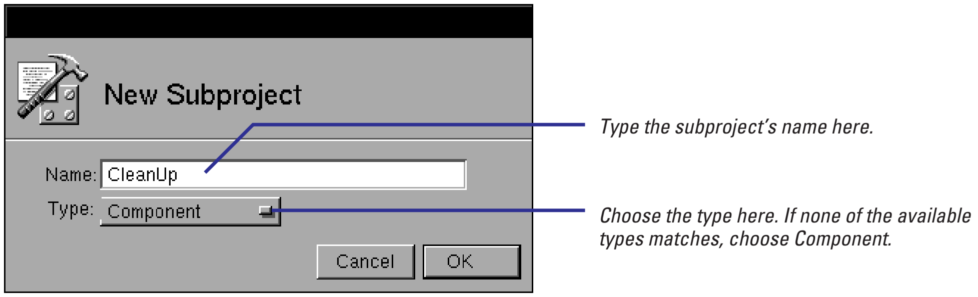
If you need to control the project server, use the commands on the Indexing menu under the Project menu. The command Purge Indices kills the current project server. After you use this command, use Index Source Code to start a new project server and reindex your project.

Grouping projects

- **To incorporate the build result of one project into the build result of another project, create a subproject.**
- **To group together related projects of any type, create an Aggregate project.**

If you have many projects that relate to each other, you may want to organize them as a single project in Project Builder. You can group projects into a single project in two ways: by creating a subproject and by creating an Aggregate project.

If one of the projects is clearly a part of the other, create it as a subproject of that other project. To create a subproject within a project, choose New Subproject from the Project menu to bring up the New Subproject panel, choose the type of subproject you want to create, type a name for the subproject in the Name field, and click OK.


New Subproject
Cancel
OK
Type the subproject's name here.
Choose the type here. If none of the available types matches, choose Component.

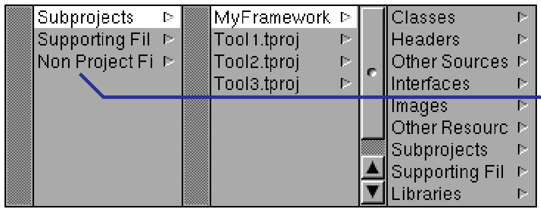
Name: CleanUp
Type: Component

When you create a subproject, you are creating a project whose build result (executable or bundle) is incorporated into the build result of the main project. (A *bundle* is simply a file package directory that contains all of the resources the user needs to execute the program.) If the main project produces a bundle as its build result, the subproject's executable or bundle is placed inside the main project's bundle. Component subprojects produce an object file that is linked with the main project's executable. A component's resources are merged into the resource directory of the main project.

For example, if you are creating an application named MyApp and one of the application's commands invokes a tool named **aTool**, you might want to have the **aTool** project be a subproject of the application project. When you build the application, **aTool** is built as well. After all of the code in both projects has been compiled and linked, Project Builder creates a directory (bundle) named **MyApp.app**. That directory contains **aTool** as well as the application's executable and its resources.

By creating a subproject, you are creating a project that is subordinate to the main project. In some cases, a subordinate relationship doesn't make sense. For example, you can't have one application be a subproject or another application.

If the projects' executables don't need to be tied to each other but you want to group them together as a convenience, create an Aggregate project. Then make those projects be subprojects of the aggregate. (You create an Aggregate project the same way you create any other type of project.)



An aggregate project contains only subprojects and a makefile that builds all of the subprojects at once.

The only purpose of an aggregate project is to group other projects together. The aggregate itself produces no executable or bundle. Because of this, you can have any type of project, even applications and frameworks, be a subproject of an aggregate.

For example, suppose you've created several tools to test a framework that you're working on. If you want to manage all of these projects as a single unit, you can create an aggregate project and include the tools and the framework as subprojects of the aggregate.

Tip: To have all subprojects be listed in the top level of the browser (rather than under Subprojects), set the preference on the Miscellaneous Preferences panel.

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.

Changing a project's name

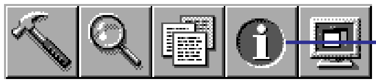
1 **Click the Inspector button.**

2 **In the Project Inspector panel, choose Project Attributes.**

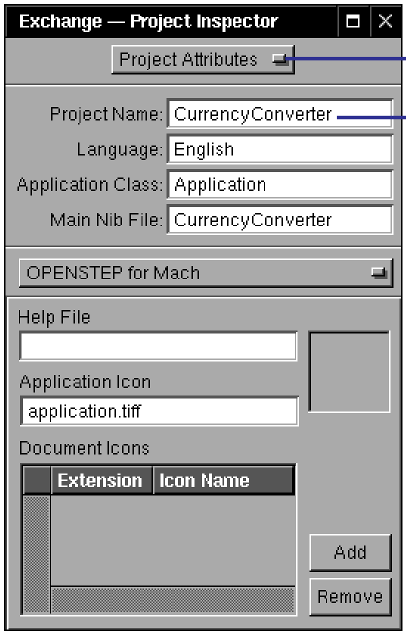
3 **Type the new name in the Project Name field.**

4 **Press Return.**

By default, the project's name is the name of the directory you chose when you created the project. The same name is used for the executable or bundle that the project creates. If this isn't the name you want to use, change it in the Project Attributes inspector.



Click this button, or choose Inspector from the Tools menu.



Exchange — Project Inspector

Project Attributes

Project Name: CurrencyConverter
Language: English
Application Class: Application
Main Nib File: CurrencyConverter

OPENSTEP for Mach

Help File

Application Icon
application.tiff

Document Icons

Extension	Icon Name

Select Project Attributes.

Type the new name here.

The Project Attributes inspector contains information Project Builder needs to know to build the project and to maintain the project makefiles. Different types of projects have different Project Attributes inspectors, but all of them have the Project Name attribute.

For application projects, this panel contains the language used to develop the application, the class for the application object, the name of the main nib file, and the icons used by the application in addition to the project name. You can change these attributes as well.

Setting the application icon

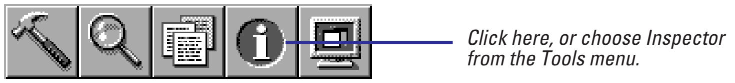
1 **Add the icons to the project under Images.**

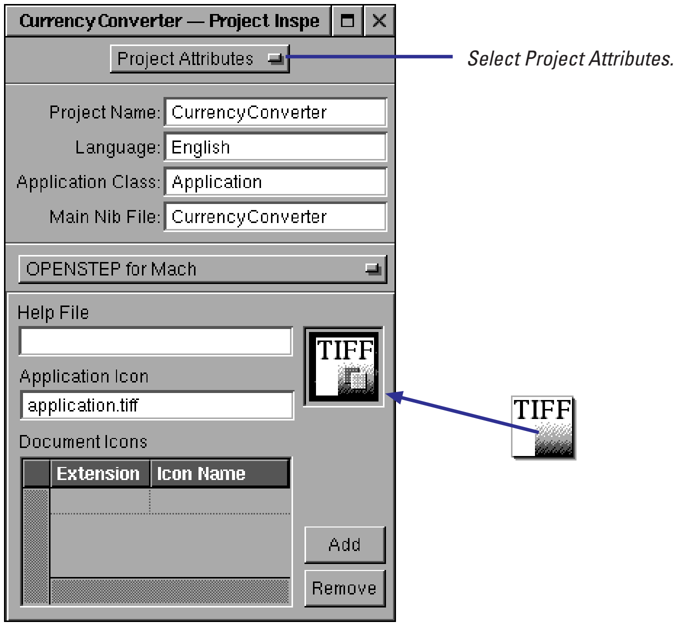
2 **Bring up the Inspector and choose Project Attributes.**

3 **Place the cursor in the Application Icon field.**

4 **Drag the application's icon from the main window into the icon well.**

For most applications, you'll want to use a unique icon. You can create your application icon using any graphics tool, but it must be a 48 X 48 pixel TIFF image. Once you've created the icon for your application, add it to the project and to the Project Attributes inspector.





Project Name: CurrencyConverter
Language: English
Application Class: Application
Main Nib File: CurrencyConverter
Application Icon: application.tiff

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.

Setting document icons

1 **Add the icons to the project under Images.**

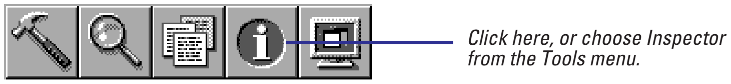
2 **Bring up the Project Inspector and choose Project Attributes.**

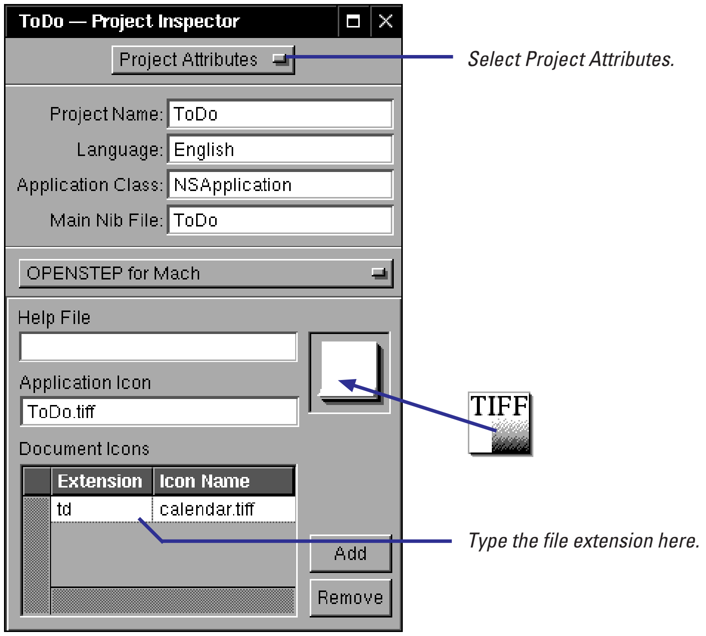
3 **Click Add Row.**

4 **Drag the icon from the main window into the icon well.**

5 **Type the file extension to be represented by this icon in the Name field of the Document Icons table.**

If an application creates documents of a unique type, you should create an icon for that document type in addition to the icon for the application itself. Like the application icon, the document icon must be a 48 X 48 pixel TIFF image. Again like the application icon, you add the document icon to the project, and then to the Project Attributes inspector.





Project Name: ToDo
Language: English
Application Class: NSApplication
Main Nib File: ToDo
Application Icon: ToDo.tiff

Extension	Icon Name
td	calendar.tiff

Are you creating a multi-document application? Be sure to read the section on multi-document applications in *Discovering OPENSTEP.*

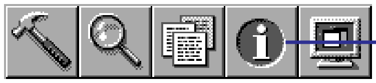
Setting system-defined document icons

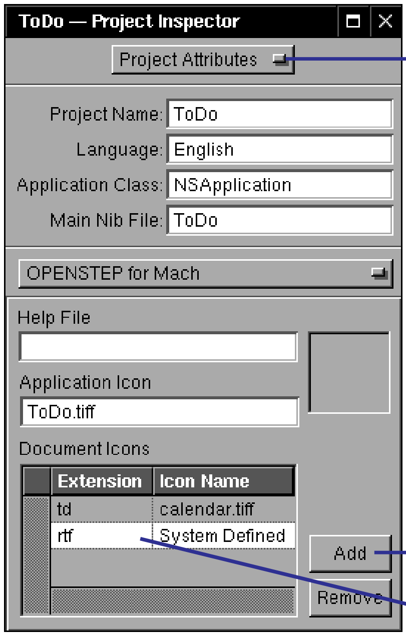
1 **Bring up the Project Inspector and choose Project Attributes.**

2 **Click Add Row.**

3 **Type the file extension in the Name field of the Document Icons table.**

In addition to listing the file types that the application can create, the Document Icons table should list the file types that the application can read but cannot create. For example, if you are creating a word processing application that can open RTF files and translate them into its own unique file type, you should list the RTF extension in this table.





Project Name: ToDo
Language: English
Application Class: NSApplication
Main Nib File: ToDo

Application Icon: ToDo.tiff

Extension	Icon Name
td	calendar.tiff
rtf	System Defined