

# Projects Example

by Steve Herrick, NeXT EOF Team

## Overview

This example shows how to control operations on a many-to-many relationship. The inspector panel uses a pop up to switch between alternate master/detail views for the relationship. Inserts and Deletes can be performed on the master, and Add and Remove operations are permitted on the detail. Add and Remove actually affect entries in an intermediate table, which ties the many-to-many together.

Inserts into the Employee and Project entities use the UniqueKey object to supply integer keys. Instances of UniqueKey object share a separate database channel to insure access for reserving new blocks of keys.

## Program Organization

### Explanation of the nib file

The File's Owner in the .nib file has been set to EOApplication. This was done by dragging in EOApplication.h and setting the File's Owner class in the inspector. EOApplication is necessary to implement autorelease pools under 3.2. This subclass of Application flushes the autorelease pool each cycle through the event queue. Special handling is used to make modal loops work correctly.

As mentioned in the release notes, the \_main file was edited as follows:

```
#import <EOApplication.h>
```

was changed to

```
#import <eointerface/EOApplication.h>
```

Each master/detail view has a two connected controllers. Additionally, a controller is used to provide selection panel choices for additions to the employees on a project and one for projects for an employee.

## Major Classes in the Application

Projects	A coordination object which is the delegate of the Employee and Project controllers. It is used to remove references in the intermediate table when an Employee or a Project is deleted. It also is the target of the Add and Remove buttons in the detail views. Add causes a selection panel to be populated with additional choices from one of the selection controllers. The Projects objects provides the qualifier for the controller that selects choices that are not already present in the detail view.
UniqueKey	Reserves blocks of unique integer keys for a given entity name. UniqueKey uses a separate channel to insure access to its own unique_key entity. This entity (table) is used to store the maxKey for a given entityName. An instance of UniqueKey reserves a block of keys by incrementing maxKey by the block size. Optimistic locking is used for the update, so a re-try consists of a call to re-fetch the current data, increment maxKey, and attempt another update.

**Other Peculiarities**

This example uses the additional shared libraries libEOInterface\_s.a, libEOAccess\_s.a, and libFoundation\_s.a. Also, Makefile.preamble uses:

OTHER\_LDFLAGS = -all\_load

to force the linking of shared library classes that are referenced at runtime by EOPalette.palette objects.

**NOTE for HPPA-Users (EOF Release 1.0 only)**

The Oracle7 adaptor is not available yet for HPPA platforms. You'll need to edit the .eomodel file and replace the Oracle7 string with the Oracle string to use the Oracle6-based adaptor.

**DB Scripts**

installPEOPLE.sqlsybase  
installPEOPLE.sqloracle   Used to create and restore the databases for the Sybase and Oracle versions of this example.  
See PeopleDBScripts directory.

Valid for NEXTSTEP Release 3.2 installed with the Enterprise Objects Framework Release 1.0 and 1.1