ArchivingObject

by Daniel Willhite, NeXT Engineering
and Mai Nguyen, NeXT Developer Support

Overview

This example shows how to archive enterprise objects that descend from Object using the traditional archiving scheme. Please see also the example ArchivingNSObject which uses the new archiving scheme to archive objects that descend from NSObject.

Both examples use the Sybase pubs database as the underlying data model.

Program Organization

How to build the eomodel file and Author.[hm]

Important note: The Enterprise Object class *Author* should be specified in the EOModeler inspector window of the authors entity to ensure the proper archiving of the Author objects. All user-defined column names in the authors table should match with the names used in the Author code.

User Interface

To show how archiving/unarchiving works, you need to first archive the selected object, then unarchive it. The text scrollview will show the contents of the archived object. A temporary file named "ObjectArchive" will be used during the archiving process.

Major Classes in the Application

Owner	The heart of the functionality of the program. It handles the archiving and unarchiving at the UI level.
Author	An example of an enterprise object that descends from Object.

Notes on Archiving

Archiving Objects and NSObjects

This program illustrates a technique for archiving graphs of objects that consist of descendents from both Object and NSObject. Archives that use this technique will be readable by future releases of NEXTSTEP. While this technique is very useful in certain situations it has certain restrictions that prevent it from being totally general.

New Archiving Scheme

Archiving in the NSObject world is similar to archiving in NEXTSTEP 3.2. NSObjects are archived using two classes from the Foundation library: NSCoder and NSArchiver.
Every class that inherits from NSObject should implement the following two methods which are analogous to the methods read: and write: in NEXTSTEP 3.2. These methods are:

```
- (void)encodeWithCoder:(NSCoder *)aCoder;
```

which is corresponds to the write: method in NEXTSTEP 3.2 and

```
- initWithCoder:(NSCoder *)aDecoder;
```

Details of these two methods can be obtained from the NSObject.h file in /NextDeveloper/Headers/foundation.

Compatibility Methods

There are some cases when an object that inherits from Object will have an outlet that holds an object which descends from NSObject. In this case, the outlet can be archived along with the object by invoking the following functions from the Object's write: method.

```
extern void NXWriteNSObject(NXTypedStream *typedStream, NSObject *object);
```

which is analogous to the function NXWriteObject() and

```
extern NSObject *NXReadNSObject(NXTypedStream *typedStream);
```

which is analogous to the fuction NXReadObject();

In the inverse case an NSObject may need to archived which has an outlet that holds an object which descends from Object. In this case the outlet can be archived along the NSObject by invoking the following methods from on the NSCoder object:

```
- (void)encodeNXObject:(Object *)object;
```

and

```
- (Object *)decodeNXObject;
```

Restrictions

There are severe restrictions on the normal archiving functionality when archiving objects from both the Object and NSObject world. These are:

1) There is no sharing of information between the two worlds. Normally, if you archive a complex graph that has cycles are where several objects reference a single object, NEXTSTEP keeps enough information about the objects that cycles are detected and objects that are pointed at by many other objects are only archived once. This is still true as long as the graph of objects being archived resides entirely in the Object world or in the NSObject world. However there is no sharing of objects information across worlds so care must be taken not to have cycles in a graph of objects that transcends both worlds.

2) Container objects can not contain objects from the other world when being archived. NSArray, NSDictionary, NSValue, etc. May not be archived if they contain any descendents of Object. Inversely, List and the other containers from the Object world may not be archived

if they contain any descendents of NSObject.

Forward Compatibility
Whenever possible, try not to mix objects from the different hierarchies in your object graphs. Archiving a mixed-world graph of objects will be much slower, take up more space, and be less reliable (due to the lack of object sharing) than archiving graphs of objects that all inherit from the same root class.

This is the only technique for archiving graphs of mixed objects that is guaranteed to compatible with later releases of NEXTSTEP. Pre-existing archives consisting of objects which inherit from from Object and new archives consisting entirely of objects that inherit from NSObject will also be forward compatible.

Other References

See also the Foundation Reference.

Valid for Enterprise Objects Framework Release Version 1.0 with NEXTSTEP Developer Release 3.2

Change History

August 94	Created example for EOF Version 1.0