RecordManager

(Original code by Phil Zakhour, Revised by Mai Nguyen)

Overview

This example demonstrates usage of the Indexing Kit at the IXRecordManager level. At start-up, a new database **.data** will be created in your home directory.

You can then enter records into that database which is managed by an IXRecordManager and perform searches based on the record attributes. In this example, the DataRecord class is the main record type that gets stored in the database. It only has 2 attributes: a string and an integer.

The main window of the application is divided into 2 sections: data entry at the top and searching at the bottom. The "Record ID" field displays the handle of the current record. You can enter values for the attributes and hit "Add Record" to enter a new record into the database. You can also edit an existing record or delete it.

The search panel shows how to do record look-up and retrieval. Just enter a value to search on and hit "Search". The found handles are displayed in the browser. Selecting a handle in the browser will retrieve the record and display its attributes in the upper portion. If the "All" switch is set, the search fieds are ignored and all entries in the database are retrieved.

Topics Of Interest

How to manage custom records with the IXRecordManager object
See the methods **addRecord:**, **deleteRecord:**, and **editRecord:** in Controller.m

How to retrieve records based on their attributes
See the method **search:** in Controller.m

Other References

Please refer also to the Indexing Kit documentation on-line. You should also take care of the endianness of the data in certain cases. Here is a documentation excerpt about this issue (see

/NextLibrary/Documentation/NextDev/GeneralRef/07_IndexingKit/IntroIndexingKit.rtfd)

Portability Issues

The Indexing Kit makes accessing data as architecture-independent as possible, but since it also allows your application to directly access file-based data at a very low level, you need to be aware of portability issues when using file-based data. For an introduction to application and data portability, see **/NextLibrary/Documentation/NextDev/Concepts/PortabilityGuide.rtf**.

With regard to portability, there are two types of data in the Indexing Kit: data whose type is known, and data whose type isn't known. The Indexing Kit automatically converts the byte-order of typed arguments to methods or functions. Any argument declared as an untyped **void ***, however, must be handled by your application.

The Indexing Kit expects all data in a store file to be big-endian, so your application should always write data as big-endian, and convert it from big-endian if needed when reading. If an Indexing Kit method or function requires you to swap byte-order of arguments or results, the documentation will contain a note to that effect.

In addition to byte-order, your application must handle alignment issues for structures stored in files. The Portability Guide outlines the proper way to declare portable structures under ªMemory-mapped Data.º If you follow its suggestions, you should have few problems.

Notes

Valid for 3.1