QueryByExample

by Craig Federighi, EO Development Team
and Mai Nguyen, NeXT Developer Support

Overview

QBE works by temporarily taking over the UI managed by an EOController and allowing the user to specify a query by filling out some subset of the UI elements. Then QBE contructs an EOQualifier to match those entries, applies it the the controller's EODatabaseDataSource, and returns the UI to normal operation.

The Example Folder

The QueryByExample is composed of the following:

1) A **QBE** object to do the query by example (Files QBE.[hm])
2) A **SortOrderSetter** object to do sorting on any entity of an eomodel (Files SortOrderSetter.[hm])
3) A **Dictionary** data source object based on the foundation class NSMutableDictionary (Files DictionaryDataSource.[hm])

Note that you can also use the 3 functionalities listed above separately.

The **QBEPalette** builds an IB palette for the QBE and SortOrderSetter objects.

How QBE works

Taking over the UI

The QBE object allows you to toggle between QBE mode and simple display mode.
When running in QBE mode, the QBE object will control the User Interface's behavior.
This control involves several steps:

· Replacing the Data Source:
The controller's EODatabaseDataSource is temporarily replaced with a DictionaryDataSource used to collect the query specification.

· Disabling linked controllers:
During query entry, linked controllers (i.e. Detail controllers and the nextController chain) are unlinked so they will not update incorrectly. Note that otherwise the update methods are being forwarded to detail controllers and next controllers.

· Reconfiguring control editability:
All UI objects associated with keys that can be queried in the data source are temporarily made editable to accomodate query entry.

Query Construction

Query contruction is accomplished by constructing a qualifier to match each attribute specified by the user and then conjoining or disjoining them to produce a composite fetch qualifier. Range queries are supported by parsing

for comparison operators in the prefix of value string and using them in the query construction.

There are 3 types of action methods that you can connect from a Button to the QBE object:

· enterQueryMode:
This method will put the user interface into QBE mode, i.e., now all associations to the EOController will be made editable to allow query entry. If the controller is already in QBE mode when this message is sent, another row is added to the dictionary data source to allow entry of another qualifier. QBE will "OR" all such qualifiers together when constructing the fetch qualifier.

· applyQualifier:
Builds a qualifier ANDING all the current attribute settings, and applies it to the dataSource. After the qualifier is built, this method will exit query mode, restoring the old data source and controller settings. It then performs a fetch on the controller to display the results from the query.

· toggleQueryFetch:
This method offers a convenient way for a single user interface button to support entering and exiting QBE mode. The first time it is called it issues **enterQueryMode:**, the next time **applyQualifier:**, and so on.

Possible Enhancements
When QBE puts a controller in query mode, it should do the same for all linked controllers that have DatabaseDataSources (and disable the others).

SortOrderSetter

SortOrderSetter allows the user to easily apply a sort ordering to EOController by selecting columns in an NXTableView. When a **sortAscending:** or **sortDescending:** message is sent to the SortOrderSetter object, it determines the currently active TableView and notes the keys corresponding to its selected columns. These keys are then used to construct an EOAttributeOrdering array which is applied to the EODatabaseDataSource for the TableView's EOController.

DictionaryDataSource

The DictionaryDataSource is an example of a non-database object that conforms to the EODataSources protocol, and is implemented as an array of dictionary objects. The QBE object uses the DictionaryDataSource to accumulate the user's query specification when in query mode.

How to build a QBE Demo

To use the SortOrderSetter palette

Load the QBEPalette.palette into IB.

Drag the SortOrderSetter palette SortOrderSetter.tiff ¬

into your IB file window. If you already set up a NXTableView with associations to an EOController with a EODatabaseDataSource, you will only need to connect a button to one of the 2 target/action methods, **setOrderAscending:** or **setOrderDescending:**. Now, you can fetch and reorder the

tableview data by selecting table view columns and pressing that button.

To use the QBE palette

Load the QBEPalette.palette into IB.
Drag the QBE palette QBE.tiff ¬
into your IB file window. Drag an eomodel file into IB. This model file will turn into an EOController. Connect the QBE controller outlet to this EOController (This step is *important!).*
Drag a NXTableView into your Application window. Make the desired associations from the controller to the tableview. Connect a button to the QBE object to the **toggleQueryFetch:** target/action method (See the description of QBE above on how to use the other methods). You are now ready to run. After performing fetch, press the QBE button (**toggleQueryFetch:**) to enter QBE mode and enter a query string into a cell of the table view (for example > 5000 for the EMPLOYEE SALARY attribute). Now push the QBE button again to apply the qualifier. To clear the qualifier (fetch all records), exit the query mode with a blank query specification.

Building a Project that uses QBE

To link a project that uses QBE, you will either need to add the [.hm] files that correspond to the QBE and DictionaryDataSource objects to your project, or change your Makefile.preamble to link in the libQBE.a library.

Known Limitations

The current QBE object only supports queries for strings, numbers, and dates. It may have to be extended to handle other data types.

Other Peculiarities

QBEPalette produces a **libQBE.a** located under ./Lib that you can re-use in your other projects. You will need to include both **libQBE.a** and the Headers in ./Headers to compile successfully.

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

Change History

16/February/95 Last edited for EOF Release 1.1 (mai nguyen)