Query by Example

NextAge Consulting
Pete Halsted
110 East Center St. #1035
Madison, SD 57042
pete@thenextage.com
www.thenextage.com
www.thenextage.com/wordpress

Table of Contents

BSD 3 License

Copyright (c) 2012, NextAge Consulting (www.thenextage.com)
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the NextAge nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NextAge Consulting - Pete Halsted

Pete Halsted has been developing custom business management applications for small to medium-sized companies, since 1987. His focus is on client/server, distributed and cloud based development utilizing WinDev, WebDev, and PostgreSQL. Pete is a Clarion Certified Developer with 25 years in the industry, has spoken at several Developers conferences, and provided Developer training and mentoring on a one on one basis. He has served companies both large and small as Project Manager, Lead Architect, Lead Developer and Chief Technology Officer. Pete tours the country full-time by motor home with his wife and dog, enjoying the freedom provided by cloud based technologies. Pete is available for Project Management, Custom Design, Development, Training, and Speaking assignments. For more information please visit www.thenextage.com or follow his blog at www.thenextage.com/wordpress

General Information

The Query by Example class provide a method of allow end users to filter their data using a simple QBE interface. It allows the them to save their queries for later reuse as well.

A QBE are three basic modes of QBE HFitler, SQL, and Advanced SQL. Hfilter and SQL work very similarly except one is accomplishing the task via Hfilter, and one uses an SQL statement with HExecuteSQLQuery. The one advantage to the SQL method is the Multiple File Fields will be expose to the QBE interface See the special notes section for additional information.

The Advanced SQL method allows complete control of the SQL statement. This requires a table that is set to be filled by programming. With the Advanced SQL mode the data can be fully loaded on initialization or using a Timer function so that the user is able to interact with the browse while records are still loading.

Software Design Consulting
The QBE class was original inspired and some of the base code comes from the training material provided by Glenn Rathke of Software Design Consulting. We highly recommend his training materials to anyone using the WX suite of tools.

Soft Design Consulting, LLC is dedicated to providing the best training in WinDev, WebDev, andWinDev Mobile, and that means we bring numerous years of creating course content and presentation to the WinDev community.

Glenn Rathke, owner of Soft Design Consulting is a seasoned developer and former professor at Roosevelt University in Chicago, IL. Glenn has trained hundreds of developers and students over the years. He incorporates concepts from many proven programming courses as well as his own unique 'take' on the WinDev language.
Soft Design Consulting provides and teaches:

1. Elegantly simple user interface which shortens the end user learning curve.
2. Data access across all methods; tables, HFilter, Queries, HExecuteSQLQuery, Stored Procedures, and more...
3. Fast data access which increases productivity.
4. Reusable QBE
5. Implementation of Dynamic Business Rules

Training is available as self study courses which include Our unique real world examples, manual, and streaming videos. We also offer 4 days of on-site training at your facility as well in the St. Louis area.

More info can be found at www.sdcdev.net

Adding to a Browse (HFilter or SQL Mode)

1. Add the QBE Button control template to the window. There shouldn't be any need to change the code behind these controls but I will document them here so you know what

they are doing. By Default when you add a control template to a window WinDev names it something like CTPL_NoName1, if you rename it to TPL_QBE, it will match the reset of the code bricks and control templates and there will be fewer changes to make.

a. Start New Session Button
    - QBE.QBEButton()
b. StatusMessage Static
    - No code, the class will update this static to provide feedback about the current activity of the QBE

2. Use the QBE Setup Code Brick to add the setup code to the window *Global Declarations Process*.
    a. `QBE is QueryByExample`
        - Instantiates a copy of the Query By Example class
    b. `QBE.QBEMode =QueryByExample.ModeHFilter`
        - The filter mode to use either ModeHFilter or ModeSQL
    c. `QBE.ExcludeHiddenColumns =True`
        - Should Hidden Columns be excluded from the QBE interface?
    d. `QBE.ExclusionList  = ""`
        - A comma delimited list of fields to exclude, this property is optional. Useful if you have hidden primary key fields etc in the selection list but don't need to have in the QBE selection screen
    e. `QBE.TheTableControl = TBL_xxx..FullName`
        - Where Tbl_xxx is the name of the table control, the order of the properties should be set the same as they are listed here otherwise the class will perform some setup actions more than once, which will not cause any functional issues but could slow down the initial screen load.
    f. `QBE.AllowEmptySearchConditions = False`
        - Can the user issue a QBE with no conditions?
    g. `QBE.StatusMessageControl = TPL_QBE.StatusMessage..FullName`
        - The name of the status control from the control template
    h. `QBE.StartEmpty =False`
        - Should the table control be loaded initially or shown empty until the user starts a QBE session. Handy for tables with a large number of records.
    i. `QBE.AdditionalConditions =  ""`
        - Additional conditions to be applied. These conditions can not be overridden by the user. Handy to force the table to only show records for the current users, etc. This property is optional.
3. In the Initialization of the Table control process code place the following line
    a. `QBE.TableInitialize()`
        - Initializes the table and fills it based on the QBE settings

Adding to a Browse (SQL Advanced Mode without Timer)

The advanced mode allows gives you complete control over the SQL statement uses for the selection of records. This requires that the table be set to be filled by programming, and for you to specify the Initial SQL statement. The class includes a “utility” function to assist with this.

Start by defining the table as you would normally using tables (not queries) and multiple table fields as needed. One the table is defined as desired. Place the following lines of code in the global declarations of the procedure and run the application. This will provide you will a message containing the resulting SQL statement as well as put it in the clipboard. You can use this statement as the basis for creating your statement, just be certain the the selection order and field types match those of the table when done.

```
QBE is QueryByExample
QBE.QBEMode =QueryByExample.ModeSQL
QBE.TheTableControl = TBL_xxx..FullName
QBE.GetSQLStatementForTable()
```

The code brick includes the above lines to assist you. Once you have captured the SQL statement, the above lines can be removed, set the table control to be filled by programming and then follow the rest of the instructions to setup the actual browse.

1. Add the QBE Button control template to the window. There shouldn’t be any need to change the code behind these controls but I will document them here so you know what they are doing.
    a. Start New Session Button
        i. QBE.QBEButton()
    b. StatusMessage Static
        i. No code, the class will update this static to provide feedback about the current activity of the QBE
2. Use the QBE Setup Advanced Mode Code Brick to add the setup code to the window *Global Declarations Process*. There will be a few properties not used unless using the TimerLoad Feature.
    a. `QBE is QueryByExample`
        i. Instantiates a copy of the Query By Example class
    b. `QBE.QBEMode = QueryByExample.ModeSQLAdvanced`
        i. The filter mode must be set to advanced.
    c. `QBE.ExcludeHiddenColumns =True`
        i. Should Hidden Columns be excluded from the QBE interface?
    d. `QBE.ExclusionList = ""`
        i. A comma delimited list of fields to exclude, this property is optional. Useful if you have hidden primary key fields etc in the selection list but don’t need to have in the QBE selection screen. With Advanced mode this should be the name of the column not the name of the database field.

e. `QBE.InitialSQLStatement = [`
   `This SQL statement created above and tweaked as desired.`
   `]`
f. `QBE.TheTableControl = TBL_xxx..FullName`
    i. Where Tbl_xxx is the name of the table control, the order of the properties should be set the same as they are listed here otherwise the class will perform some setup actions more than once, which will not cause any functional issues but could slow down the initial screen load.
g. `QBE.AllowEmptySearchConditions = False`
    i. Can the user issue a QBE with no conditions?
h. `QBE.StatusMessageControl = TPL_QBE.StatusMessage..FullName`
    i. The name of the status control from the control template
i. `QBE.StartEmpty =False`
    i. Should the table control be loaded initially or shown empty until the user starts a QBE session. Handy for tables with a large number of records.
j. `QBE.AdditionalConditions =  ""`
    i. Additional conditions to be applied. These conditions can not be overridden by the user. Handy to force the table to only show records for the current users, etc. This property is optional.

3. In the Initialization of the Table control process code place the following line
    a. `QBE.TableInitialize()`
        i. Initializes the table and fills it based on the QBE settings

Adding to a Browse (SQL Advanced Mode with Timer)

The advanced mode with timer is exactly the same as without except that it loads the records a page at a time, returning control back to the user immediately allowing them to interact with the table while it is still loading records. This is very handy will working with large database tables. The initial setup is the same with or without the Timer, there are just a few extra properties to set, the instructions that follow are only for the additional properties you should refer to the without Timer section for the general setup of the SQL Advanced Mode.

1. Add the QBETimed control template to the window. There shouldn't be any need to change the code behind these controls but I will document them here so you know what they are doing.
    a. Start New Session Button
        i. QBE.QBEButton()
    b. Stop QBE Button
        i. QBE.StopButton()
    c. StatusMessage Static
        i. No code, the class will update this static to provide feedback about the current activity of the QBE
    d. ProgressBar
        i. No code, the class will update this progress bar with a record count as it loads records with it interval of the timer.
2. Use the QBE Setup Advanced Mode Code Brick to add the setup code to the window *Global Declarations Process*. Below a the specific properties used with the TimerLoad Feature.
    a. `QBE.UseTimer =True`
        i. Uses the timer feature
    b. `QBE.ProgressBarControl = TPL_QBE.ProgressBar..FullName`
        i. The name of the progress bar from the control template.
    c. `QBE.LoadInterval = 1`
        i. The number of seconds between each Page load, default is 1 second
    d. `QBE.RecordsPerPage   = 50`
        i. Number of records to load with each Page. Default is 50.
3. In the Initialization of the Table control process code place the following line
    a. `QBE.TableInitialize()`
        i. Initializes the table and fills it based on the QBE settings

Special Notes Concerning Large Datasets, Queries and Query Parameters.

When you get near a 100,000 records (or you are doing a remote connection to the database) my testing using the actual file for the table control source versus a project query shows some advantages to using the actual files. This is because of the nature of the QBE class.

The QBE class is designed to work Query parameters. I am not sure how parameters would be support and still keep the QBE code as generic and easy to implement as it currently is. I welcome input and suggestions for others.

When not using Query parameters, my testing seems to indicate that HF actually builds the entire dataset and then filters that with anything specified via an HFilter or a SQL statement. So when the records sets gets larger this will cause a delay as the Query is built.

For instance in my testing a table with over a million records. When I used a project query, even with the option set to start with an empty data set, the table still took between 10-20 seconds to completely load and allow the user to interact.I believe this is because the Query is being built with the full record set and then filtered from there.

When using the actual file for the table control source, I did not see this slow down. Because it appears with an actual file the filter is being performed by the server.

Special Notes Concerning QBE Mode and Files vs Queries

The QBE Mode of HFILTER or SQL are both supported for table control that have a source of a file or a project query. The nature of embedded queries does not not allow them to be used in SQL mode. Unless you are working with large record sets this likely will not make a difference.

One other note about QBE Mode and Query vs File. If you are using an actual File as the table source and QBE mode of HFILTER, and you have a field included in the table that is using a multi-file link to show a value from another file, that column will not be available via QBE. Since HFILTER is only applied to one file, it naturally can only be applied to the primary file of the table.

However if you were to use the QBE mode of SQL, then the field would be available because in SQL mode the QBE class is generating an actual SQL statement and can generate a where clause using field from more than one file.

The other option for this situation would be to create a Query that would “flatten” the files into one query. Which would again allow the field to be expose to QBE. Which method you use will be a matter of personal preference and the size of the dataset based on the information concerning large datasets and queries mentioned elsewhere in the document..

Database Table

The class has been written in a generic fashion that allows the file and field names to be set via properties if desired. The class also uses Hxxxx commands for all file access, meaning that the class should work with any database that WX supports. The suggested definition for the files follows. If you do not use this definition you will have to use the associated properties to point to the correct values (*See the Special Properties*).

QBEQuery

Holds the saved queries, queries can either be global or specific to a user.

- QBEQueryID
    - Automatic ID
- UserID - Primary key to the User Table
    - Integer
- QueryName - Name of the saved Query
    - Text (50)
- TableControl - The name of the table control managed by this query
    - Text (100)
- SavedQuery - The actual filter or SQL statement for the Query.
    - Text Memo

QBEFieldValue

Holds the individual field conditions for a saved query.

- QBEFieldValueID
    - Automatic ID
- QBEQueryID - Link to the QBE Query File
    - Integer
- Item- The Field Name
    - Text (50)
- Caption - Caption of the field (Provides a user friendly name)
    - Text (50)
- Condition - The Condition dropdown value (Equal, not equal, etc) is translated to a number.
    - Integer
- FirstValue- First value to check
    - Text (50)
- And_Or - And or Or for some conditions
    - Text (50)
- SecondValue - Second Value to check if And or Or condition
    - Text(50)

Users

While not technically a file of the Query by Example, it does use the user table to retrieve the user name to show for saved queries, you table will likely have several other fields but will need at least these two fields, remember you can use the special properties to override the default names.

- UserID
    - Automatic ID
- LoginName- The users Name
    - Text (50)

Project Code

The QBE class is instantiated for each browse it is used on, however there are a few global properties that should be set in your project initialization code, so these values will be the same for all instances of the class. If you are using the special properties to change the File or field names that should be done here as well.

- `QueryByExample.UserIDVariable = "GLO.UserID"`
    - The project variable that stores the User Id
- `QueryByExample.SilentErrors = False`
    - If set to True Error Messages in the class will not be displayed. The default value is False. Note: If using the System Log class the error will still be logged even if an error is not displayed
- `QueryByExample.TheSystemLogClass = SysLog`
    - Pointer to the System Log class for logging errors.

Properties

Note: Private Properties are not documented.

AdditionalConditions
This allows you to specify a condition that they user doesn't have access to and can not override. For instance if you want to place QBE on the customer table, but some users are only allowed to see a subset of customers, the condition for that subset could be added via the *AdditionalConditions* property and the user would not be able to query records that they should not have access to.

AllowEmptySearchConditions
By default,If the user doesn't specify any conditions the class does not perform a data fetch. This is to avoid loading a large number of records unnecessarily. This property will allow the data fetch to be performed without any conditions being specified. Default is False

ExcludeHiddenColumns
If True then hidden columns are not available to QBE or note. Default is True

ExclusionList
This is a comma delimited list of field names that you don't want to to be available for the user to select for QBE Conditions. Note: if you have a multiple file browse and only include a field name in the ExclusionList than that field would be excluded from all files, however if you prefix the field name with the file name in the exclusion list (User.UserID) than only the field from that file will be excluded if it exists in more than one file.

InitialSQLStatement
Complete Select statement including joins that will be used as the basis for the statements generated when QBE Mode is SQLADVANCE

JoinStyle
When using Multiple File columns in the table, you can specify how you want the joins to be generated. Choices are INNER, OUTER, WHERE. The default is INNER.

LoadInterval
Only valid when UseTimer is True. The number of seconds between each record load. Default is 1

ProgressBarControl
Only valid when UseTimer is True. The name of the progress control to display the progress of the timed load.
QBEMode

The type of Filtering Applied (HFILTER,SQL, or Advanced) Default is HFILTER

RecordsPerPage
Only valid when UseTimer is True. The number of records to read with each record load. Default is 50

SQLConnection
Name of the Connection to use when using SQL QBEMode. When specified the SQL statements generated by QBE will be executed against the connection using HQueryWithoutCorrection. If not specified, the queries will be ran using the non connection syntax of HExecuteSQLQuery which does not allow HQueryWithoutCorrection.

StartEmpty
Boolean value, determines if the table control is initially loaded or empty when the screen opens. Default is False

StatusMessageControl
The name of the control that displays the QBE status.

TheTableControl
The name of the table control being managed.

UseTimer
Only valid for SQLAdvanced mode. When enabled data is loaded via a timer loop, allowing the user to interactive with the table control, while records are still loading.

Methods

Note: Private Methods are not documented.

BuildSearchConditions
Pulls all of the QBE settings together and creates Filter and / or SQL statement, called by the execute button of the QBE window and should not need to be called directly.

DeleteSavedQuery
Deletes a selected saved query, should not need to be called directly.

ExecuteSavedButton
Loads the Search Conditions with a saved query, used by the QBE window and should not need to be called directly.

GetSQLStatementForTable
This method is not intended for use in production. It is used to assist you with creating an SQLADVANCED QBE browse. If you create the browse using standard files as the source, then run the application calling this method and it will give you the InitialSQLStatement that will match the browse. You can then convert the browse to be filled by programming, and tweak the InitialSQLStatement as needed, making sure that the field types and order remain intact.

InitializeSavedQueries
Populates the SavedQueries array for the current table

LoadSaveQuery
Populates the QBE window with a selected saved query, should not need to be called directly.

QBEButton
Initiates a QBE Search Session, Displays QBE Window, and then manages the filtering of the browse. Generally called via one of the Control Templates.

StopButton
Only valid when UseTimer is True. Stop the current timer and halts loading records.

TableInitialize
Call should be placed in the Initialization Code of the table control. This method handles all the logic of applying the filter and resetting the table. Can also be called from other areas, passing in a condition to manually start a QBE session.

UpdateTableRow
Used by the Browse Form Manager to update the browse after an update, if SQL Advanced mode is being used.

ValidateConditionCombo
Used to validate the Condition Drop Combo on the QBE window, should not need to be called directly.

EQUATES

Equates are provided to assist with setting some of the properties

ModeHFilter
QBEMode set to use HFilter technique. Default Value of QBE Mode

ModeSQL
QBEMode set to use SQL technique

ModeSQLAdvanced
QBEMode set to use the Advanced SQL technique that allows you full control of the SQL statement.

JoinStyleInner
Set JoinStyle to create Inner Joines. Default Value of JoinStyle

JoinStyleOuter
Set JoinStyle to create Outer Joins

JoinStyleWhere
Set JoinStyle to use “old style” joins via the Where clause

Code Bricks

QBE - QBE Setup
Instantiates a copy of the Query by Example class and sets it up.

QBE - QBE Setup Advanced Mode
Instantiates a copy of the Query by Example class and sets it up for Advanced Mode with Timer

Special Properties

The below properties can be used to change the default file name and field names used. They are global properties and should be set in the project initialization code using the QueryByExample.PropertyName syntax.

QBEQueryFile
The name of the QBEQuery file. Default = “QBEQuery”

QBEQueryQBEQueryIDField
Default ="QBEQueryID"

QBEQueryUserIDField
Default = "UserID"

QBEQueryQueryNameField
Default = "QueryName"

QBEQueryTableControlField
Default = "TableControl"

QBEQuerySavedQueryField
Default ="SavedQuery"

QBEFieldValueFile
Default = "QBEFieldValue"

QBEFieldValueQBEQueryIDField
Default = "QBEQueryID"

QBEFieldValueItemField
Default = "Item"

QBEFieldValueCaptionField
Default ="Caption"

QBEFieldValueConditionField
Default = "Condition"

QBEFieldValueFirstValueField
Default = "FirstValue"

QBEFieldValueAnd_OrField
Default = "And_Or"

QBEFieldValueSecondValueField
Default = "SecondValue"

UsersFile
Default = "Users"

UsersUserIdField
Default = "Users.UserID"

UsersLoginName
Default = "Users.LoginName"

The following properties are global properties that should be set in the project code so thier value will be the same for all instances of the class.

SilentErrors

If set to True Error Messages in the class will not be displayed. The default value is False. Note: If using the System Log class the error will still be logged even if an error is not displayed.

TheSystemLogClass

Pointer to the System Log class for logging errors.

- The System Log class is another Open Source class available from NextAge. The Security class will perform without it. However if you don't not want to include the System Log class in your project you will have to comment out the related lines of the class. Unfortunately WX does not provide the ability to optionally compile code based on the existence of another class. As long as the class is include in your project you will not have compile errors and the class calls will be skipped, however if you remove the System Log class from your project, you will have compile errors in the Security class.

UserIdVarialbe

The project variable that stores the User Id

Control Templates

QBEButton
Adds a QBE Search Button to a screen, the button makes a call to QBE.QBEButton()

QBESavedQueries
Adds a drop list to a QBE browse that allows the user select select a predefined query. There is a template variable CanViewAllQueries that can be overridden to show all queries versus just the current users.

QBETimed
Adds a QBE Search and Stop Button to a screen and progress bar, this version should be used with the UserTimer SQL Advanced Mode

Windows

Win_Find_Records_QBE
This is the primary QBE interface with the user will configure their QBE request. This is based on the original work provided by Glenn Rathke's training examples, but has been modified to be SQL specific and perform background data loads. If you want to use QBE with Hyper File or with direct data loads, I suggest purchasing his training materials.

Win_Saved_QBE
Called from *Win_Find_Records_QBE*, and provides an interface to allow users to save their queries for future use.

Change Log

1.0 - January 9, 2013

Initial Release

1.01 - January 16 , 2013

1. Fixed direct reference to file in DeleteSavedQuery Method
2. Added some methods used by Browse Form Manager to refresh the table after an update
3. Resolved issue with Additional Conditions not being applied if no search conditions supplied
4. Resolved issue with Get Record Count not handling From as first word of line
5. Resolved issue with Advance SQL where clause not being correct if there are ambiguous field names