tFetchGroup:aFetchGroup Initializes associates aFetchGroup with aDest
expression:anExpr
destination:aDest

stinationThe association's user interface object
chGroupThe DBFetchGroup that owns the association
pressionThe DBExpression that selects the properties displayed
Destination:newDestinationSets the user association's user interface object

ntentsDidChangeNotice to redisplay because the value changed
rentRecordDidDeleteNotice to redisplay because a record was deleted
lEditingNotification that editing the destination must end
Value:valueGets an object containing the association's data
signed int)selectedRowAfter:(unsigned int)previousRow
The index of the next selected row
ectionDidChangeNotice that the user has changed the selection
Value:valueSets the association's data
idateEditingNotice to validate changes the user has made

ociation:associationGets an object containing the association's data
getValue:(DBValue *)value
ociation:associationSets the association's data
setValue:(DBValue *)value
ociationContentsDidChange:associationNotice to redisplay because the value changed
ociationCurrentRecordDidDelete:association
Notice to redisplay because a record was deleted
ociationSelectionDidChange:associationNotice that the user has changed the selection

tInitializes a new DBBinder instance
tForDatabase:aDBDatabase Initializes database, properties, qualifier frees DBBinder

st *)getProperties:(List *)aListGets and returns the DBBinder's properties
st *)setProperties:(List *)aListSets and returns the DBBinder's properties
lProperty:anObjectAdds an object to the list of properties
ovePropertyAt:(unsigned int)indexDeletes one of the objects from the list of properties

BQualifier *)qualifierThe DBBinder's qualifier
Qualifier:(DBQualifier *)aQualifierSets the DBBinder's qualifier

<DBContainers>)containerThe DBBinder's container
Container:(id <DBContainers>)anObjectSets the DBBinder's container
FlushEnabled:(BOOL)flagSets whether flushing the DBBinder is permitted
OOL)isFlushEnabledReports whether flushing is enabled default YES
FreeObjectsOnFlush:(BOOL)flagSets whether the DBBinder is freed when flushed
OOL)areObjectsFreedOnFlushYES if container objects freed when DBBinder is flushed

RecordPrototype:anObjectMakes anObject the prototype for the DBBinder's records
ateRecordPrototypeCreate default prototype object for the DBBinder's records
OOL)ownsRecordPrototypeYES if createRecordPrototype will work (no prototype)
ordPrototypeThe DBBinder's record prototype
ociateRecordIvar:(const char *)variableName Makes variableName report the value of aProperty
withProperty:(id <DBProperties>)aProperty
ociateRecordSelectors:(SEL)set Sets the selectors for storing and retrieving aProperty
:(SEL)get
withProperty:(id <DBProperties>)aProperty
BValue *)valueForProperty:(id <DBProperties>)aProperty
The value of aProperty for the current record

lRetrieveOrder:(DBRetrieveOrder)anOrder for:(id <DBProperties>)aProperty
Appends aProperty to the retrieve ordering criteria
ioveRetrieveOrderFor:(id <DBProperties>)aProperty
Removes aProperty from the list of ordering criteria
BRetrieveOrder)retrieveOrderFor:(id <DBProperties>)aProperty
The direction in which aProperty is sorted on retrieve
signed int)positionInOrderingsFor:(id <DBProperties>)aProperty
The rank order of aProperty in the list of order criteria
OOL)ignoresDuplicateResultsYES if duplicate records are ignored during select
IgnoresDuplicateResults:(BOOL)flagSets whether duplicate records will be ignored in select

OOL)evaluateString:(const unsigned char *)aString
Tells the adaptor to evaluate aString (without qualifier)
OOL)adaptorWillEvaluateString:(const unsigned char *)aString
YES if delegate permits evaluation of aString default YES

chInThreadStarts an asynchronous fetch to the container
ncelFetchAborts an asynchronous fetch
eckThreadedFetchCompletion:(double)timeout
Sends binderDidFetch: if an asynchronous fetch completes within timeout seconds

MaximumRecordsPerFetch:(unsigned int)limit
Sets maximum records per synchronous fetch
nsigned int)maximumRecordsPerFetchReturns maximum records per fetch default unlimited
OOL)recordLimitReachedYES if the previous fetch stopped for the record limit

SharesContext:(BOOL)flagSet whether this binder uses the shared cursor
OOL)sharesContextYES if this binder uses the shared cursor

etCancels any fetch, then flushes and frees objects
OOL)flushIf enabled, empties the container
XZone *)scratchZoneThe zone the DBBinder is now using for allocations

egateThe object that receives notification messages
Delegate:anObjectSets the object to receive notification messages

d:(NXTypedStream *)streamCreates an instance by reading from a typed stream
te:(NXTypedStream *)streamArchives an instance by writing to a typed stream

der:aBinder didEvaluateString:(const unsigned char *)aString
Notification that aString was evaluated by the adaptor
OOL)binder:aBinder willEvaluateString:(const unsigned char *)aString
Notification that aString will be sent the adaptor YES lets evaluation proceed
derDidDelete:aBinderNotification that records were deleted from the database
derDidFetch:aBinderNotification that records were fetched from the database
derDidInsert:aBinderNotification that records were inserted in the database
derDidSelect:aBinderNotification that records were selected (but not fetched)

tFromFile:(const char *)aPathInitializes and loads information from a model file

nst char *)directoryThe directory from which the model was loaded
nst char *)nameThe model's name in the class's name table
OOL)setName:(const char *)aStringSets the model's name in the class's name table
nst char *)currentAdaptorNameThe name of the current database adaptor
nst char *)defaultAdaptorNameThe name of the model's default adaptor
nst unsigned char *)defaultLoginStringThe model's default login string
nst unsigned char *)currentLoginStringThe current login string
nst unsigned char *)loginStringForUser:(const char *)aUser
The the model's login string for user aUser

<DBEntities>)entityNamed:(const char *)aName
urns an object embodying entity aName
st *)getEntities:(List *)aListReturns a list of the names of the model's entities

ptyDataDictionaryFrees the current data dictionary
dDefaultDataDictionaryReplaces the data dictionary by querying the database

OOL)connect Opens a connection to database using the default login
OOL)connectUsingString:(const unsigned char *)aString
Opens database connection to database by sending aString
OOL)connectUsingAdaptor:(const char *)aClassname
andString:(const unsigned char *)aString
OOL)disconnect Disconnects from the database
OOL)disconnectUsingString:(const unsigned char *)aString
Disconnects from the database by sending it aString

OOL)areTransactionsEnabledYES if transactions are enabled
OOL)enableTransactions:(BOOL)flagEnable/disable transaction returns YES if successful

egateThe object that receives notification messages
Delegate:anObjectSets the object that receives notification messages

OOL)evaluateString:aStringReturns YES if the adaptor evaluates the string

OOL)arePanelsEnabledYES if UI panels can respond to problems
PanelsEnabled:(BOOL)flagEnable/disable response by UI panels

d:(NXTypedStream *)streamCreates an instance by reading from a typed stream
te:(NXTypedStream *)streamArchives an instance by writing to a typed stream

aDatabaseNotification of log message sent by aDatabase
log:(const char*) fmt, ¼
OOL)db:aDatabase Notification of a message received from aDatabase
notificationFrom:anObjectReturns YES when the user acknowledges the notifiation.
message:(const unsigned char*)msg
code:(int)n
OOL)db:aDatabase Notice that aString will be evaluated YES lets it proceed
willEvaluateString:(const char *)aString
usingBinder:(const char *)aBinder
DidRollbackTransaction:senderNotification that database rolled back a transaction
DidCommitTransaction:senderNotification that database committed a transaction
WillCommitTransaction:senderNotification that database will commit a transaction
WillRollbackTransaction:senderNotification that database will roll back a transaction

Initializes a new instance
eFrees the space an instance formerly used

tReturns the font used in the editable display
Font:aFontSets the font used in the editable display

:(BOOL)useColumnPos
OOL)editFieldAt:(unsigned int)row Displays and prepares to edit one field of the data source's
:(unsigned int)column current record, taken from row or column of dynamic axis,
inside:(NXRect *)frameusing rowAttrs or colAttrs to identify static attributes,
inView:aViewand flags useRowPos and useColumnPos to select which
withAttributes:(id <DBTableVectors>) rowAttrs
:(id <DBTableVectors>) columnAttrsreturns YES if editing was permitted
usePositions:(BOOL)useRowPos
:(BOOL)useColumnPos
onEvent:theEvent

rtEditingForces an end to editing and discards changes returns self
OOL)endEditingEnds editing when user clicks elsewhere Returns YES if that becomes first responder

d:(NXTypedStream *)streamCreates an instance by reading from a typed stream
te:(NXTypedStream *)streamArchives an instance by writing to a typed stream
ishUnarchivingAutomatically invoked final step in unarchiving

tForEntity:(id <DBEntities>)anEntity Initializes for anEntity, with description string shown
fromDescription:(const unsigned char *)descriptionFormat, ¼
tForEntity:(id <DBEntities>)anEntity Initializes anEntity, from property aName,
fromName:(const char *)aName to have data type aType
usingType:(const char *)aType
pyFromZone:(NXZone *)zoneReturns new copy of receiver, allocated from zone
eFrees the space that an instance formerly used

Entity:(id <DBEntities>)anEntity Sets anEntity, with the description string shown
andDescription:(const unsigned char *)descriptionFormat, ¼

d:(NXTypedStream *)streamCreates an instance by reading from a typed stream
te:(NXTypedStream *)streamArchives an instance by writing to a typed stream

nst char *)nameReturns the name (set to match the attribute it fetches)
duleThe DBModule that owns the fetch group
ityThe DBEntity for which the fetch group fetches
ordListThe DBRecordList in which fetched records are stored
signed int)currentRecordThe index within the DBRecordList of the current record
signed int)recordCountThe number of records in the DBRecordList

Autoselect:(BOOL)flagIf YES, fetch selects first row, delete selects next rowf
OOL)doesAutoSelectReturns flag set by setAutoselect: default YES
CurrentRecord:(unsigned int)newIndexSets a position within the DBRecordList
arCurrentRecordDeselects current record
signed int)selectedRowAfter:(unsigned int)previousRow
Index of first selected row after previousRow
isplayEverythingDisplays all of DBFetchGroup's DBAssociations

eteCurrentSelectionDeletes the selected records from the DBRecordList
OOL)insertNewRecordAt:(unsigned int)indexInserts a (default) record in the DBRecordList at index
chContentsOf:aSource usingQualifier:aQualifier
Replaces all records by reading aSource using aQualifier

OOL)hasUnsavedChangesYES if the DBRecordList has been changed but not saved
OOL)validateCurrentRecordYES unless delegates for editor or DDModule object
eChangesSaves changes in this or subordinate fetch groups
cardChangesDiscards changes in this and subordinate fetch groups

lExpression:newExpressionAdds newExpression to the list of expressions to fetch
eValueFromAssociation:anAssociationPuts the displayed value into the DBRecordList
lAssociation:newAssociationAdds newAssociation to the list of associations
moveAssociation:anAssociationRemoves anAssociation from the list of associations

egateThe object that receives notification messages
Delegate:anObjectSets the object to receive notification messages

chGroup:fetchGroup
didInsertRecordAt:(int)indexNotification of a new record in the DBRecordList

chGroupWillFetch:fetchGroupNotification that fetch will change the DBRecordList
OOL)fetchGroupWillSave:fetchGroupNotification of pending save YES lets it proceed

aSourceReturns the DBRecordList (or other source)
DataSource:newDataSourceMakes newDataSource the place to get values for display

ValueAt:(unsigned int) row Returns a DBValue from the DBRecordList,
:columntaking it from position row or column of the dynamic axis,
inside:(NXRect *)frameusing rowAttrs or colAttrs to identify static attributes,
inView:aView and flags useRowPos and useColumnPos to select which
withAttributes:(id <DBTableVectors>) rowAttrs
:(id <DBTableVectors>)columnAttrs
usePositions:(BOOL)useRowPos
:(BOOL)useColumnPos

wFieldAt:(unsigned int) row Displays one field of the data source's current record
:columntaken from position row or column of the dynamic axis,
inside:(NXRect *)frameusing rowAttrs or colAttrs to identify static attributes,
inView:aView and flags useRowPos and useColumnPos to select which
withAttributes:(id <DBTableVectors>) rowAttrs
:(id <DBTableVectors>)columnAttrs
usePositions:(BOOL)useRowPos
:(BOOL)useColumnPos

ginBatching:(id <DBTableVectors>) attrsNotification that format attrs apply to all following items
lBatchingMarks the end of a block of items formatted the same way
etBatching:(id <DBTableVectors>)attrsBegin batching if not already started

egateThe object that receives notification messages
Delegate:anObjectSets the object to receive notification messages

Initializes a new instance
eFrees the space an instance formerly used

:columnIntaken from position row or column of the dynamic axis,
inside:(NXRect *)frameusing rowAttrs or colAttrs to identify static attributes,
inView:aView and flags useRowPos and useColumnPos to select which
withAttributes:(id <DBTableVectors>) rowAttrs
:(id <DBTableVectors>)columnAttrs
usePositions:(BOOL)useRowPos
:(BOOL)useColumnPos

d:(NXTypedStream *)streamCreates an instance by reading from a typed stream
te:(NXTypedStream *)streamArchives an instance by writing to a typed stream

tFrame:(const NXRect *)frameRectInitializes the view in the frame coordinates
wSelf:(const NXRect *)rects Called by display to draw the image
:(int)rectCount

ageReturns the image being displayed
Image:newImageMakes newImage the image to display

Style:(int)newStyleSets the style of border for the image
t)styleReturns a constant indicating the border style

OOL)isEditableYES if the image can be deleted or replaced
Editable:(BOOL)flagAllow/prohibit deleting or replacing the image

abaseThe DBModule's DBDatabase
ityThe DBModule's DBEntity
tFetchGroupThe DBModule's root DBFetchGroup
ociationForObject:anObjectThe DBAssociation that handles UI object anObject
tingAssociationThe DBAssociation currently involved in editing
FetchGroups:(List *)aListReturns a list of all the DBModule's DBFetchGroups
chGroupNamed:(const char *)aNameReturns the DBFetchGroup for the property named aName

chAllRecords:senderFetches all records for the DBModule's root fetch group
veChanges:senderSaves in the database changes made to the fetched records
cardChanges:senderDiscard changes proposed for the fetched records
eteRecord:senderDelete one of the fetched records
pendNewRecord:senderAppend a new (default) record to those fetched
ertNewRecord:senderInsert a new (default) record at the current position
xtRecord:senderSelect the next of the fetched records
viousRecord:senderSelect the preceding of the fetched records
eValueFrom:senderUI object has a new value, so fetched record is revised
tDidEnd:textObjectUser has finished editing a text field
endChar:(unsigned short)whyEnd
OOL)textWillChange:textObjectUser has entered an editable field YES lets editing proceed
OOL)textWillEnd:textObjectNotification that an editable field will relinquish first responder YES lets the cha

egateThe object that receives notification messages
Delegate:anObjectSets the object to receive notification messages

duleDidSave:moduleCalled when module has completed a save to the database
OOL)moduleWillLoseChanges:moduleCalled when module is about to discard user's changes
OOL)moduleWillSave:moduleCalled when module is about to save to the database

tForEntity:(id <DBEntities>)anEntityInitializes a new instance to select from anEntity
tForEntity:(id <DBEntities>)anEntityInitializes to select from anEntity by descriptionFormat
fromDescription:(const unsigned char *)descriptionFormat, ¼
pyFromZone:(NXZone*)zReturns a copy of the DBQualifier, allocating from z
eFrees space that a DBQualifier formerly used

dDescription:(const unsigned char *)descriptionFormat, ¼
Appends descriptionFormat to the qualifier descriptions
Entity:(id <DBEntities>)anEntitySets both anEntity and qualifying descriptionFormat
andDescription:(const unsigned char *)descriptionFormat, ¼
OOL)setName:(const char *)aNameAssigns the DBQualifier aName and returns YES
OOL)emptyDeletes the qualifying descriptions and returns YES

te:(NXTypedStream *)streamArchives an instance by writing to a typed stream

tInitializes a new instance of DBRecordList
eFrees the space a DBRecordList formerly used
arEmpties the record list and lists of properties

RetrieveMode:(DBRecordListRetrieveMode)aMode
Sets the DBRecordList's retrieval strategy
BRecordListRetrieveMode)currentRetrieveMode
Returns a constant identifying the retrieval strategy

chRecordForRecordKey:(DBValue *)aValueFetches records qualified by matching aValue
chUsingQualifier:(DBQualifier *)aQualifierEmpties, then fetches records selected by aQualifier
chUsingQualifier:(DBQualifier *)aQualifierFetches records selected by aQualifier
empty:emptyFirstif emptyFirst is YES, first empties the record list
signed int)recordLimitReturns the maximum number of records to fetch
RecordLimit:(unsigned int)countSets the maximum number of records to fetch

Value:(DBValue *)aValuePuts the current record's value for aProperty into aValue
forProperty:aProperty
Value:(DBValue *)aValuePuts value of aProperty for the record at index into aValue
forProperty:aProperty
at:(unsigned int)index
RecordKeyValue:(DBValue *)aValuePuts the key value for the current record into aValue
RecordKeyValue:(DBValue *)aValuePuts the key value for the record at index into aValue
at:(unsigned int)index

Value:(DBValue *)aValueSets the current record's value of aProperty to aValue
forProperty:aProperty
Value:(DBValue *)aValue Sets value of aProperty for record at index to aValue
forProperty:aProperty
at:(unsigned int)index
ertRecordAt:(unsigned int)indexInserts a (default) record ahead of the record at index
pendRecordInserts a (default) record after the last one
wRecordInserts a (default) record to precede the current record

signed int)positionForRecordKey:(DBValue *)aValue
Returns the index of the record whose key is aValue
veRecordAt:(unsigned int)sourceIndexMoves record at sourceIndex to precede the record now
to:(unsigned int)destinationIndex at destinationIndex
apRecordAt:(unsigned int)anIndexTransposes the positions of the two records
withRecordAt:(unsigned int)anotherIndex

signed int)saveModificationsSaves to the database any changes since the fetch returns code for success, partia

Initializes a new instance
eFrees space formerly used by a DBRecordStream

dRetrieveOrder:(DBRetrieveOrder)anOrderAppends anOrder (up/down) to sort criteria for aProperty
for:(id <DBProperties>)aProperty
st *)setProperties:(List *)propertyList Sets/returns list of properties wanted from entity aSource
ofSource:aSource
st *)getProperties:(List *)propertyListReturns and puts into propertyList the stream's properties
st *)setKeyProperties:(List *)propertyListSets and returns propertyList as the stream's key properties
st *)getKeyProperties:(List *)keyListReturns/ puts into propertyList the stream's key properties

chUsingQualifier:(DBQualifier *)aQualifierStarts fetching records that pass aQualifier
celFetchStops fetching and sends fetchDone to DBDatabase
BRecordRetrieveStatus)currentRetrieveStatusDB_Ready/NotReady, DB_FetchInProgress/Completed

Value:(DBValue *)aValue forProperty:aProperty
Puts current record's aProperty's value into aValue
RecordKeyValue:(DBValue *)aValuePuts current record's key value into aValue
NextMakes next record available nil if none left

Value:(DBValue *)aValue forProperty:aPropert
Sets the current record's aProperty to aValue
wRecordInserts new, empty record at the current record
eteRecordDeletes the current record

arResets everything except the delegate

egateThe object that receives notification messages
Delegate:anObjectSets the object that will receive notification messages
derDelegateThe object that receives notification messages for binders
BinderDelegate:anObjectSets the object to receive notification messages for binders

OOL)recordStream:sender Invoked when changes can't be saved aCode tells why
willFailForReason:(DBFailureCode) aCodeYES acknowledges failure NO tries to proceed with those reco
OOL)recordStreamPrepareCurrentRecordForModification:aRecordStream
Invoked when a record will be modififed or deleted YES permits modification to proceed

Identifier:anIdentiferInitialize a DBTableVector for property anIdentifier
eFree the space formerly allocated to a DBTableVector

tFrame:(const NXRect *)newFrameInitializes an instance located within newFrame
eFrees space formerly used by a DBTableView

DataSource:aSourceSets the object that will provide data for the display
aSourceThe object that provides data for the display
Delegate:delegateSets the object that will receive notification messages
egateThe object that receives notification messages

awSelf:(const NXRect *) rects :(int) count

matterAt:(unsigned int)row :(unsigned int)column
Returns the DBFormatter for the field at row and column
OOL)dynamicRowsYES if rows are dynamic

OOL)doesAcceptArrowKeysYES if arrow keys are accepted for navigation
owVectorReordering:(BOOL)flagLets/prevents user drag static row/column to new position
OOL)doesAllowVectorReorderingYES if user is permitted to reorder static row or column
owVectorResizing:(BOOL)flagLets/prevents user drag the width of static row or column
OOL)doesAllowVectorResizingYES if user can drag row or column to change width

oadData:senderRedraw because data may have changed
outChanged:senderRedraw because row or column spacing changed
wsChangedFrom:(unsigned int)startRowRedraw because data changed in a block of rows
to:(unsigned int)endRow
umnsChangedFrom:(unsigned int)startColumn Redraw because data changed in a block of columns
to:(unsigned int)endColumn

signed int)columnCountTotal number of columns
signed int)rowCountTotal number of rows
<DBTableVectors>)rowAt:(unsigned int)aPosition
Object specifying format of the static row at aPosition
<DBTableVectors>)columnAt:(unsigned int)aPosition
Object specifying format of the static column at aPosition
dColumn:identifier Adds an static column at aPosition
at:(unsigned int)aPosition
dColumn:identifier Adds a static column with title at aPosition
withTitle:(const char *)title
dColumn:identifierAdds a static column with title and formatter at aPosition
withFormatter:formatter
andTitle:(const char *)title
at:(unsigned int)aPosition
moveColumnAt:(unsigned int)columnPositionDeletes a static column
OOL)moveColumnFrom:(unsigned int)oldPosChanges a static column's position
to:(unsigned int)newPos
dRow:identifier Adds an static row at aPosition
at:(unsigned int)aPosition
dRow:identifier Adds a static row with title at aPosition
withTitle:(const char *)title
dRow:identifierAdds a static row with title and formatter at aPosition
withFormatter:formatter
andTitle:(const char *)title
at:(unsigned int)aPosition
moveRowAt:(unsigned int)rowPositionDeletes a static row
OOL)moveRowFrom:(unsigned int)oldPosChanges a static row's position
to:(unsigned int)newPos
signed int)indedOfColumnWithIdentifier:anIdentifier
Position in sequence of static column anIdentifier
signed int)indexOfRowWithIdentifier:anIdentifier
Position in sequence of static row anIdentifier

t)modeReturns DB_NOSELECT/RADIOMODE/LISTMODE
owEmptySel:(BOOL)flagAllow/prohibit user to leave nothing selected
OOL)doesAllowEmptySelYES if user may leave nothing selected
signed int)selectedRowCountNumber of rows currently selected
signed int)selectedColumnCountNumber of columns currently selected
t)selectedRowThe row number of the selected row
t)selectedColumnThe column number of the selected column
OOL)isRowSelected:(unsigned int)rowYES if row is selected
OOL)isColumnSelected:(unsigned int)columnYES if column is selected
electAll:senderMakes nothing selected.
ectAll:senderMakes all rows and columns selected.
RowSelectionOn:(unsigned int)start Sets block of rows to selected (YES) or deselected (NO)
:(unsigned int)end
to:(BOOL)flag
ColumnSelectionOn:(unsigned int)start Sets block of columns to selected (YES) or deselected
:(unsigned int)end
to:(BOOL)flag
ectRow:(unsigned int)rowSelects row, or extends selection if flag is YES
byExtension:(BOOL)flag
ectColumn:(unsigned int)columnSelects row, or extends selection if flag is YES
byExtension:(BOOL)flag
electRow:(unsigned int)rowDeselects the indicated row
electColumn:(unsigned int)columnDeselects the indicated column
signed int)selectedRowAfter:(unsigned int)aRow
Index of the first selected row after aRow
signed int)selectedColumnAfter:(unsigned int)aColumn
Index of the first selected column after aColumn
dAction:(SEL)anActionSends anAction to anObject for each selected row
to:anObjectif YES, does it for each selected row
forSelectedRows:(BOOL)flag
dAction:(SEL)anActionSends anAction to anObject for each selected column
to:anObjectif YES, does it for each selected column
forSelectedColumns:(BOOL)flag

wHeadingReturns the row heading view
RowHeading:newRowHeadingMakes newRowHeading the row heading view
RowHeadingVisible:(BOOL)flagMakes the row heading visible or not
umnHeadingReturns the column heading view
ColumnHeading:newColumnHeadingMakes newColumnHeading the column heading view
ColumnHeadingVisible:(BOOL)flagMakes the column heading visible or not

playDisplays the DBTableView
ollClip:aClip Sets aClip's origin to be newOrigin in the content view
to:(const NXPoint *)newOrigin

ollColumnToVisible:(unsigned int)columnScroll the content so that column is visible in the scroll clip
OOL)acceptsFirstResponderYES if the DBTableView will handle keyboard events

Action:(SEL)aSelectorMakes aSelector the action in response to a click
EL)actionThe action to be sent on a click
DoubleAction:(SEL)aSelectorMakes aSelector the action in response to a double click
EL)doubleActionThe action in response to a double click
Target:anObjectMakes anObject the target for an action message
getThe target for an action message

d:(NXTypedStream *)streamCreates an instance by reading from a typed stream
te:(NXTypedStream *)streamArchives an instance by writing to a typed stream
ishUnarchivingAutomatically invoked final step in unarchiving

tInitializes a new DBTextFormatter instance
eFrees the space allocated to a DBTextFormatter.

tReturns the formatter's font
Font:aFontMakes aFont the formatter's font

ginBatching:(id <DBTableVectors>) attrsThe format attrs applies to all following records
etBatching:(id <DBTableVectors>) attrsBegins batching if not already in effect
lBatchingCompletes sequence of records in same format

d:(NXTypedStream *)streamCreates an instance by reading from a typed stream
te:(NXTypedStream *)streamArchives an instance by writing to a typed stream

tInitialize a DBValue instance

StringValue:(const char *)aStringSets the object's value to aString
StringValueNoCopy:(const char *)aStringSets the object's value so that it points to aString
ValueFrom:(DBValue *)aValueSets the object's to have the same value as aValue
NullSets the object's value to NULL

<DBTypes>)valueTypeReturns the type of value the object contains
OOL)isEqual:(DBValue *)anotherValueYEs if this object has same type and value as anotherValue
uble)doubleValueReturns the object's value as a double
at)floatValueReturns the object's value as a float
)intValueReturns the object's value as an int
ectValueReturns the object's value as an object
nst char *)stringValueReturns the object's value as a string
OOL)isNullYES if the object's value is NULL

d:(NXTypedStream *)streamCreates an instance by reading from a typed stream
te:(NXTypedStream *)streamArchives an instance by writing to a typed stream