EODisplayGroup

Inherits From: NSObject

Conforms To: NSCoding
NSObject (NSObject)

Declared In: EOInterface/EODisplayGroup.h

Class at a Glance

Purpose

An EODisplayGroup collects an array of objects from an EODataSource, and works with a group of EOAssociation objects to display and edit the properties of those objects.

Principal Attributes

- Array of objects supplied by an EODataSource
- EOQualifier and EOSortOrderings to filter the objects for display
- Array of selection indexes
- Delegate

Creation

Interface Builder

– init	Designated initializer.

Commonly Used Methods

– allObjects	Returns all objects in the EODisplayGroup.
– displayedObjects	Returns the subset of all objects made available for display.
– selectedObjects	Returns the selected objects.
– setQualifier:	Sets a filter that limits the objects displayed.
– setSortOrdering:	Sets the ordering used to sort the objects.
– updateDisplayedObjects	Filters, sorts, and redisplays the objects.
– insertObjectAtIndex:	Creates a new object and inserts it into the EODataSource.

Class Description

An EODisplayGroup is the basic user interface manager for an Enterprise Objects Framework application. It collects objects from an EODataSource, filters and sorts them, and maintains a selection in the filtered subset. It interacts with user interface objects and other display objects through EOAssociations, which bind the values of objects to various aspects of the display objects.

An EODisplayGroup manipulates its EODataSource by sending it **fetchObjects**, **insertObject:**, and other messages, and registers itself as an editor and message handler of the EODataSource's EOEditingContext. The EOEditingContext allows the EODisplayGroup to intercede in certain operations, as described in the EOEditors and EOMessageHandlers informal protocol specifications. EODisplayGroup implements all the methods of these informal protocols; see the descriptions for **editingContextWillSaveChanges:**, **editorHasChangesForEditingContext:**, and **editingContext:presentErrorMessage:** for more information.

Most of an EODisplayGroup's interactions are with its associations, its EODataSource, and its EOEditingContext. See the EOAssociation, EODataSource, and EOEditingContext class specifications for more information on these interactions.

Creating an EODisplayGroup

You create most EODisplayGroups in Interface Builder, by dragging an entity icon from the EOModeler application, which creates an EODisplayGroup with an EODatabaseDataSource, or by dragging an EODisplayGroup with no EODataSource from the EOPalette. EODisplayGroups with EODataSources operate independent of other EODisplayGroups, while those without EODataSources must be set up in a master-detail association with another EODisplayGroup. See Chapter 4, "Creating an Enterprise Objects Framework Project" in the *Enterprise Objects Framework Developer's Guide* for more information on setting up EODisplayGroups in Interface Builder.

To create an EODisplayGroup programmatically, simply initialize it and set its EODataSource:

```
EODataSource *myDataSource;    /* Assume this exists. */
EODisplayGroup *myDisplayGroup;

myDisplayGroup = [[EODisplayGroup alloc] init];
[myDisplayGroup setDataSource:myDataSource];
```

After creating the EODisplayGroup, you can add associations as described in the EOAssociation class specification under "Setting up an Association Programmatically."

Getting Objects

Since an EODisplayGroup isn't much use without objects to manage, the first thing you do with an EODisplayGroup is send it a fetch message. You can use the basic **fetch** method; the **fetch:** action method, which can be invoked by a control in the EODisplayGroup's nib file; or, you can configure the

EODisplayGroup in Interface Builder to fetch automatically when its nib file is loaded. These methods all ask the EODisplayGroup's EODataSource to fetch from its persistent store with a **fetchObjects** message.

Filtering and Sorting

An EODisplayGroup's fetched objects are available through its **allObjects** method. These objects are treated only as candidates for display, however. The array of objects actually displayed is filtered and sorted by the EODisplayGroup's delegate, or by a qualifier and sort ordering array. You set the qualifier and sort orderings using the **setQualifier:** and **setSortOrdering:** methods. The **displayedObjects** method returns this filtered and sorted array; index arguments to other EODisplayGroup methods are defined in terms of this array.

If the EODisplayGroup has a delegate that responds to **displayGroup:displayArrayForObjects:**, it invokes this method rather than using its own qualifier and sort ordering array. The delegate is then responsible for filtering the objects and returning a sorted array. If the delegate only needs to perform one of these steps, it can get the qualifier or sort orderings from the EODisplayGroup and apply either itself using the NSArray methods **filteredArrayUsingQualifier:** and **sortedArrayUsingKeyOrderArray:**, which are added by the control layer.

If you change the qualifier or sort ordering, or alter the delegate in a way that changes how it filters and sorts the EODisplayGroup's objects, you can send **updateDisplayedObjects** to the EODisplayGroup to get it to refilter and resort its objects. Note that this doesn't cause the EODisplayGroup to refetch.

Changing and Examining the Selection

An EODisplayGroup keeps a selection in terms of indexes into the array of displayed objects. EOAssociations that display values for multiple objects are responsible for updating the selection in their EODisplayGroups according to user actions on their display objects. This is typically done with the **setSelectionIndexes:** method. Other methods available for indirect manipulation of the selection are the action methods **selectNext:** and **selectPrevious:**, as well as **selectObjectsIdenticalTo:** and **selectObjectsIdenticalTo:selectFirstOnNoMatch:**.

To get the selection, you can use the **selectionIndexes** method, which returns an array of NSNumbers, or **selectedObjects**, which returns an array containing the selected objects themselves. Another method, **selectedObject**, returns the first selected object if there is one.

The Delegate

EODisplayGroup offers a number of methods for its delegate to implement; if the delegate does, it invokes them as appropriate. Besides the aforementioned **displayGroup:displayArrayForObjects:**, there are methods that inform the delegate that the EODisplayGroup has fetched, created an object (or failed to create one), inserted or deleted an object, changed the selection, or set a value for a property. There are also methods that request permission from the delegate to perform most of these same actions. The delegate can

return YES to permit the action or NO to deny it. See each method's description, at the end of this class specification, for more information.

Methods for Use by EOAssociations

While most of your application code interacts with objects directly, EODisplayGroup also defines methods for its associations to access properties of individual objects without having to know anything about which methods they implement. Accessing properties through the EODisplayGroup offers associations the benefit of automatic validation, as well.

Associations access objects by index into the displayed objects array, or by **id**. **valueForObjectAtIndex:key:** returns the value of a named property for the object at a given index, and **setValue:forObjectAtIndex:key:** sets it. Similarly, **valueForKey:object:** and **setValue:forObject:key:**access the objects by **id**. EOAssociations can also get and set values for the first object in the selection using **selectedObjectValueForKey:** and **setSelectedObjectValue:forKey:**.

Adopted Protocols

NSCoding	– encodeWithCoder: – initWithCoder:

Method Types

Creating instances	– init
Configuring behavior	– setFetchesOnLoad: – fetchesOnLoad – setSelectsFirstObjectAfterFetch: – selectsFirstObjectAfterFetch – setUsesOptimisticRefresh: – usesOptimisticRefresh – setValidatesChangesImmediately: – validatesChangesImmediately
Setting the data source	– setDataSource: – dataSource
Setting the qualifier and sort ordering	– setQualifier: – qualifier – setSortOrdering: – sortOrdering

Managing queries	– qualifierFromQueryValues – setEqualToQueryValues: – equalToQueryValues – setGreaterThanQueryValues: – greaterThanQueryValues – setLessThanQueryValues: – lessThanQueryValues – qualifyDisplayGroup – qualifyDisplayGroup: – qualifyDataSource – qualifyDataSource: – enterQueryMode: – inQueryMode – setInQueryMode: – enabledToSetSelectedObjectValueForKey:
Fetching objects from the data source	– fetch – fetch:
Getting the objects	– allObjects – displayedObjects
Updating display of values	– redisplay – updateDisplayedObjects
Setting the objects	– setObjectArray:
Changing the selection	– setSelectionIndexes: – selectObjectsIdenticalTo: – selectObjectsIdenticalTo:selectFirstOnNoMatch: – selectObject: – clearSelection – selectNext – selectNext: – selectPrevious – selectPrevious:
Examining the selection	– selectionIndexes – selectedObject – selectedObjects
Inserting and deleting objects	– insertObject:atIndex: – insertObjectAtIndex: – insert: – deleteObjectAtIndex: – deleteSelection – delete:

Adding keys	– setLocalKeys: – localKeys
Getting the associations	– observingAssociations
Setting the delegate	– setDelegate: – delegate
Changing values from associations	– setSelectedObjectValue:forKey: – selectedObjectValueForKey: – setValue:forObject:key: – valueForKey:object: – setValue:forObjectAtIndex:key: – valueForObjectAtIndex:key:
Editing by associations	– associationDidBeginEditing: – association:failedToValidateValue:forKey:object:errorDescription: – associationDidEndEditing: – editingAssociation – endEditing
Querying changes for associations	– contentsChanged – selectionChanged – updatedObjectIndex
Interacting with the EOEditingContext	– editorHasChangesForEditingContext: – editingContextWillSaveChanges: – editingContext:presentErrorMessage:

Instance Methods

allObjects

– (NSArray *)**allObjects**

Returns all of the objects collected by the receiver.

See also: – **displayedObjects**, – **fetch**

association:failedToValidateValue:forKey:object:errorDescription:

– (BOOL)**association:**(EOAssociation *)*anAssociation*
failedToValidateValue:(NSString *)*value*
forKey:(NSString *)*key*
object:(id)*anObject*
errorDescription:(NSString *)*errorDescription*

Invoked by *anAssociation* from its **shouldEndEditingForAspect:invalidInput:errorDescription:...** method to let the receiver handle a validation error. This method opens an attention panel with *errorDescription* as the message and returns NO.

See also: – **displayGroup:shouldDisplayAlertWithTitle:message:** (Methods Implemented By the Delegate)

associationDidBeginEditing:

– (void)**associationDidBeginEditing:**(EOAssociation *)*anAssociation*

Invoked by *anAssociation* when its display object begins editing to record that EOAssociation as the editing association.

See also: – **editingAssociation**, – **associationDidEndEditing:**, – **endEditing**,
– **association:failedToValidateValue:forKey:object:errorDescription:**

associationDidEndEditing:

– (void)**associationDidEndEditing:**(EOAssociation *)*anAssociation*

Invoked by *anAssociation* to clear the editing association. If *anAssociation* is the receiver's editing association, clears the editing association. Otherwise does nothing.

See also: – **editingAssociation**, – **associationDidBeginEditing:**, – **endEditing**,
– **association:failedToValidateValue:forKey:object:errorDescription:**

clearSelection

– (BOOL)**clearSelection**

Invokes **setSelectionIndexes:** to clear the selection, returning YES on success and NO on failure.

contentsChanged

– (BOOL)**contentsChanged**

Returns YES if the receiver's array of objects has changed and not all observers have been notified, NO otherwise. EOAssociations use this in their **subjectChanged** methods to determine what they need to update.

See also: – **selectionChanged**, – **updatedObjectIndex**

dataSource

– (EODataSource *)**dataSource**

Returns the receiver's EODataSource.

See also: – **setDataSource:**

delegate

– (id)**delegate**

Returns the receiver's delegate.

See also: – **setDelegate:**

delete:

– (void)**delete:**(id)*sender*

This action method invokes **deleteSelection**.

See also: – **deleteObjectAtIndex:**

deleteObjectAtIndex:

– (BOOL)**deleteObjectAtIndex:**(unsigned int)*index*

Attempts to delete the object at *index*, returning YES if successful and NO if not. Checks with the delegate using **displayGroup:shouldDeleteObject:**. If the delegate returns NO, this method fails and returns NO. If successful, sends the delegate a **displayGroup:didDeleteObject:** message.

This method performs the delete by sending **deleteObject:** to the EODataSource. If that message raises an exception, this method fails and returns NO.

See also: – **deleteSelection**, – **delete:**

deleteSelection

– (BOOL)**deleteSelection**

Attempts to delete the selected objects, returning YES if successful and NO if not.

See also: – **deleteObjectAtIndex:**, – **delete:**

displayedObjects

– (NSArray *)**displayedObjects**

Returns the objects that should be displayed or otherwise made available to the user, as filtered by the receiver's delegate or by its qualifier and sort ordering.

See also: – **allObjects**, – **updateDisplayedObjects**,
– **displayGroup:displayArrayForObjects:** (Methods Implemented By the Delegate),
– **qualifier**, – **sortOrdering**

editingAssociation

– (EOAssociation *)**editingAssociation**

Returns the EOAssociation editing a value if there is one, NO if there isn't.

See also: – **associationDidBeginEditing:**, – **associationDidEndEditing:**

editingContext:presentErrorMessage:

– (void)**editingContext:**(EOEditingContext *)*anEditingContext*
presentErrorMessage:(NSString *)*errorMessage*

Invoked by *anEditingContext* as part of the EOMessageHandlers informal protocol, this method calls **NSRunAlertPanel()** with no title, *errorMessage* as the message to display, and an OK button.

editingContextWillSaveChanges:

– (void)**editingContextWillSaveChanges:**(EOEditingContext *)*anEditingContext*

Invoked by *anEditingContext* in its **saveChanges** method as part of the EOEditors informal protocol, this method allows the EODisplayGroup to prohibit a save operation. EODisplayGroup's implementation of this method invokes **endEditing**, and raises an NSInternalInconsistencyException if it returns NO. Thus, if there's an association that refuses to end editing, *anEditingContext* doesn't save changes.

editorHasChangesForEditingContext:

– (BOOL)**editorHasChangesForEditingContext:**(EOEditingContext *)*anEditingContext*

Invoked by *anEditingContext* as part of the EOEditors informal protocol, this method returns NO if any association is editing, YES otherwise.

See also: – **editingAssociation**, – **associationDidBeginEditing:**, – **associationDidEndEditing:**

enabledToSetSelectedObjectValueForKey:

– (BOOL)**enabledToSetSelectedObjectValueForKey:**(NSString *)*key*

Returns YES to indicate that a single value association (such as an EOControlAssociation for a NSTextField) should be enabled for setting *key,* NO otherwise. Normally this is the case if the receiver has a selected object. However, if *key* is a special query key (for example, "@query=.name"), then the control should be enabled even without a selected object.

endEditing

– (BOOL)**endEditing**

Attempts to end any editing taking place. If there's no editing association or if the editing association responds YES to an **endEditing** message, returns YES. Otherwise returns NO.

See also: – **editingAssociation**

enterQueryMode:

– (void)**enterQueryMode:**(id)*sender*

This action method invokes **setInQueryMode:** with an argument of YES.

See also: – **inQueryMode**

equalToQueryValues

– (NSDictionary *)**equalToQueryValues**

Returns the receiver's dictionary of equalTo query values. This dictionary is typically manipulated by associations bound to keys of the form @query=.*propertyName*. The **qualifierFromQueryValues** method uses this dictionary along with the lessThan and greaterThan dictionaries to construct qualifiers.

See also: – **setEqualToQueryValues:**, – **greaterThanQueryValues**, – **lessThanQueryValues**,

fetch

– (BOOL)**fetch**

Attempts to fetch objects from the EODataSource, returning YES on success and NO on failure.

Before fetching, invokes **endEditing** and sends **displayGroupShouldFetch:** to the delegate, returning NO if either of these methods does. If both return YES, sends a **fetchObjects** message to the receiver's EODataSource to replace the object array, and if successful sends the delegate a **displayGroup:didFetchObjects:** message.

See also: – **fetch:**

fetch:

– (void)**fetch:**(id)*sender*

This action method invokes **fetch**.

fetchesOnLoad

– (BOOL)**fetchesOnLoad**

Returns YES if the receiver fetches automatically after being loaded from a nib file, NO if it must be told explicitly to fetch. The default is NO. You can set this behavior in Interface Builder using the Inspector panel.

See also: – **fetch**, – **fetch:**, – **setFetchesOnLoad:**

greaterThanQueryValues

– (NSDictionary *)**greaterThanQueryValues**

Returns the receiver's dictionary of greaterThan query values. This dictionary is typically manipulated by associations bound to keys of the form @query>.*propertyName*. The **qualifierFromQueryValues** method uses this dictionary along with the lessThan and equalTo dictionaries to construct qualifiers.

See also: – **setGreaterThanQueryValues:**, – **lessThanQueryValues**, – **equalToQueryValues**

init

– (id)**init**

Initializes a newly allocated EODisplayGroup. The new EODisplayGroup then needs to have an EODataSource set with **setDataSource:**. This is the designated initializer for the EODisplayGroup class. Returns **self**.

See also: – **bindAspect:displayGroup:key:** (EOAssociation)

inQueryMode

– (BOOL)**inQueryMode**

Returns YES to indicate that the receiver is in query mode, NO otherwise. In query mode, controls in the user interface that normally display values become empty, allowing users to type queries directly into them (this is also known as a "Query By Example" interface). In effect, the receiver's "displayedObjects" are replaced with an empty equalTo query values dictionary. When **qualifyDisplayGroup** or **qualifyDataSource** is subsequently invoked, the query is performed and the display reverts to displaying values—this time, the objects returned by the query.

See also: – **setInQueryMode:**, – **enterQueryMode:**

insert:

– (void)**insert:**(id)*sender*

This action method invokes **insertObjectAtIndex:** with an index just past the first index in the selection, or 0 if there's no selection.

See also: – **insertObject:atIndex:**

insertObject:atIndex:

– (void)**insertObject:**(id)*anObject* **atIndex:**(unsigned int)*index*

Inserts *anObject* into the receiver's EODataSource and displayed objects at *index*, if possible. This method checks with the delegate before actually inserting, using **displayGroup:shouldInsertObject:atIndex:**. If the delegate refuses, *anObject* isn't inserted. After successfully inserting the object, this method informs the delegate with a **displayGroup:didInsertObject:** message, and selects the newly inserted object.

Raises an NSRangeException if *index* is out of bounds.

See also: – **insertObjectAtIndex:**, – **insert:**

insertObjectAtIndex:

– (id)**insertObjectAtIndex:**(unsigned int)*anIndex*

Asks the receiver's EODataSource to create a new object by sending it a **createObject** message, then inserts the new object using **insertObject:atIndex:**. If a new object can't be created, this method sends the delegate a **displayGroup:createObjectFailedForDataSource:** message or, if the delegate doesn't respond, opens an attention panel to inform the user of the error.

See also: – **insert:**

lessThanQueryValues

– (NSDictionary *)**lessThanQueryValues**

Returns the receiver's dictionary of lessThan query values. This dictionary is typically manipulated by associations bound to keys of the form @query<.*propertyName*. The **qualifierFromQueryValues** method uses this dictionary along with the greaterThan and equalTo dictionaries to construct qualifiers.

See also: – **setLessThanQueryValues:**, – **greaterThanQueryValues**, – **equalToQueryValues**

localKeys

– (NSArray *)**localKeys**

Returns the additional keys that EOAssociations can be bound to. An EODisplayGroup's basic keys are typically those of the attributes and relationships of its objects, as defined by their EOClassDescription through an EOEntity in the model. Local keys are typically used to form associations with key paths, with arbitrary methods of objects, or with properties of objects not associated with an EOEntity. Interface Builder allows the user to add and remove local keys in the EODisplayGroup Attributes Inspector panel.

See also: – **setLocalKeys:**

observingAssociations

– (NSArray *)**observingAssociations**

Returns all EOAssociations that observe the receiver's objects.

See also: + **observersForObject:** (EOObserverCenter)

qualifier

– (EOQualifier *)**qualifier**

Returns the receiver's qualifier, which it uses to filter its array of objects for display when the delegate doesn't do so itself.

See also: – **updateDisplayedObjects**, – **displayedObjects**, – **setQualifier:**

qualifierFromQueryValues

– (EOQualifier *)**qualifierFromQueryValues**

Builds a qualifier constructed from entries in the three query dictionaries: equalTo, greaterThan, and lessThan. These, in turn, are typically manipulated by associations bound to keys of the form @query=.firstName, @query>.budget, @query<.budget.

See also: – **qualifyDisplayGroup**, – **qualifyDataSource**

qualifyDisplayGroup

– (void)**qualifyDisplayGroup**

Takes the result of **qualifierFromQueryValues** and applies to the receiver using **setQualifier:**. The method **updateDisplayedObjects** is invoked to refresh the display. If the receiver is in query mode, query mode is exited.

See also: – **qualifyDataSource**

qualifyDisplayGroup:

– (void)**qualifyDisplayGroup:**(id)*sender*

This action method invokes **qualifyDisplayGroup:**.

qualifyDataSource

– (void)**qualifyDataSource**

Takes the result of **qualifierFromQueryValues** and applies to the receiver's data source. The receiver then sends itself a **fetch** message. If the receiver is in query mode, query mode is exited. This method differs from **qualifyDisplayGroup** as follows: whereas **qualifyDisplayGroup** performs in-memory filtering of already fetched objects, **qualifyDataSource** triggers a new qualified fetch against the database.

See also: – **qualifyDisplayGroup**

qualifyDataSource:

– (void)**qualifyDataSource:**(id)sender
This action method invokes **qualifyDataSource**.

redisplay

– (void)**redisplay**

Notifies all observing associations to redisplay their values.

See also: – **observingAssociations**

selectedObject

– (id)**selectedObject**

Returns the first selected object in the displayed objects array, or **nil** if there's no such object.

See also: – **displayedObjects**, – **selectionIndexes**, – **selectedObjects**

selectedObjects

– (NSArray *)**selectedObjects**

Returns the objects selected in the receiver's displayed objects array.

See also: – **displayedObjects**, – **selectionIndexes**, – **selectedObject**

selectedObjectValueForKey:

– (id)**selectedObjectValueForKey:**(NSString *)*key*

Returns the value corresponding to *key* for the first selected object in the receiver's displayed objects array, or **nil** if exactly one object isn't selected.

See also: – **valueForKey:object:**

selectionChanged

– (BOOL)**selectionChanged**

Returns YES if the selection has changed and not all observers have been notified, NO otherwise. EOAssociations use this in their **subjectChanged** methods to determine what they need to update.

See also: – **contentsChanged**

selectionIndexes

– (NSArray *)**selectionIndexes**

Returns the indexes of the receiver's selected objects as NSNumbers, in terms of its displayed objects array.

See also: – **displayedObjects**, – **selectedObjects**, – **selectedObject**, – **setSelectionIndexes:**

selectNext

– (BOOL)**selectNext**

Attempts to select the object just after the currently selected one, returning YES if successful and NO if not. The selection is altered in this way:

- If there are no objects, does nothing and returns NO.
- If there's no selection, selects the object at index zero and returns YES.
- If the first selected object is the last object in the displayed objects array, selects the first object and returns YES.
- Otherwise selects the object after the first selected object.

See also: – **selectPrevious**, – **selectNext:**, – **setSelectionIndexes:**

selectNext:

– (void)**selectNext:**(id)*sender*

This action method invokes **selectNext**.

See also: – **selectPrevious:**, – **setSelectionIndexes:**

selectObject:

– (BOOL)**selectObject:**(id)*object*

Returns YES to indicate that the receiver has found and selected *object*, NO if it can't find a match for *object* (in which case it clears the selection). The selection is performed on the receiver's displayedObjects, not allObjects.

selectObjectsIdenticalTo:

– (BOOL)**selectObjectsIdenticalTo:**(NSArray *)*objects*

Attempts to select the objects in the receiver's displayed objects array whose **id**s are equal to those of *objects*, returning YES if successful and NO otherwise.

See also: – **setSelectionIndexes:**, – **selectObjectsIdenticalTo:selectFirstOnNoMatch:**

selectObjectsIdenticalTo:selectFirstOnNoMatch:

– (BOOL)**selectObjectsIdenticalTo:**(NSArray *)*objects* **selectFirstOnNoMatch:**(BOOL)*flag*

Selects the objects in the receiver's displayed objects array whose **id**s are equal to those of *objects*, returning YES if successful and NO otherwise. If no objects in the displayed objects array match *objects* and *flag* is YES, attempts to select the first object in the displayed objects array.

See also: – **setSelectionIndexes:**, – **selectObjectsIdenticalTo:**

selectPrevious

– (BOOL)**selectPrevious**

Attempts to select the object just before the presently selected one, returning YES if successful and NO if not. The selection is altered in this way:

- If there are no objects, does nothing and returns NO.
- If there's no selection, selects the object at index zero and returns YES.
- If the first selected object is at index zero, selects the last object and returns YES.
- Otherwise selects the object before the first selected object.

See also: – **selectNext**, – **selectPrevious:**, – **redisplay**

selectPrevious:

– (void)**selectPrevious:**(id)*sender*

This action method invokes **selectPrevious**.

See also: – **selectNext:**, – **redisplay**

selectsFirstObjectAfterFetch

– (BOOL)**selectsFirstObjectAfterFetch**

Returns YES if the receiver automatically selects its first displayed object after a fetch if there was no selection, NO if it leaves an empty selection as-is.

EODisplayGroups by default do select the first object after a fetch when there was no previous selection.

See also: – **displayedObjects**, – **fetch**, – **selectsFirstObjectAfterFetch**

setDataSource:

– (void)**setDataSource:**(EODataSource *)*aDataSource*

Sets the receiver's EODataSource to *aDataSource*. In the process, it performs these actions:

- Unregisters **self** as an editor and message handler for the previous EODataSource's EOEditingContext, if necessary, and registers **self** with *aDataSource's* EOEditingContext. If the new EOEditingContext already has a message handler, however, the receiver doesn't assume that role.
- Registers **self** for EOObjectsChangedInEditingContextNotification and EOInvalidatedAllObjectsInStoreNotification from the new EOEditingContext.
- Clears the receiver's array of objects.
- Sends **displayGroupDidChangeDataSource:** to the delegate if there is one.

See also: – **dataSource**

setDelegate:

– (void)**setDelegate:**(id)*anObject*

Sets the receiver's delegate to *anObject*, without retaining it.

See also: – **delegate**

setEqualToQueryValues:

– (void)**setEqualToQueryValues:**(NSDictionary *)*values*

Sets to *values* the receiver's dictionary of equalTo query values. The **qualifierFromQueryValues** method uses this dictionary along with the lessThan and greaterThan dictionaries to construct qualifiers.

See also: – **equalToQueryValues**, – **setLessThanQueryValues:**, – **setGreaterThanQueryValues:**

setFetchesOnLoad:

– (void)**setFetchesOnLoad:**(BOOL)*flag*

Controls whether the receiver automatically fetches its objects after being loaded from a nib file. If *flag* is YES it does; if *flag* is NO the receiver must be told explicitly to fetch. The default is NO. You can also set this behavior in Interface Builder using the Inspector panel.

See also: – **fetch**, – **fetch:**, – **fetchesOnLoad**

setGreaterThanQueryValues:

– (void)**setGreaterThanQueryValues:**(NSDictionary *)*values*

Sets to *values* the receiver's dictionary of greaterThan query values. The **qualifierFromQueryValues** method uses this dictionary along with the lessThan and equalTo dictionaries to construct qualifiers.

See also: – **greaterThanQueryValues**, – **setLessThanQueryValues:**, – **setEqualToQueryValues:**

setInQueryMode:

– (void)**setInQueryMode:**(BOOL)*flag*

Sets according to *flag* whether the receiver is in query mode. For more discussion of query mode, see the method description for **inQueryMode**.

See also: – **enterQueryMode:**

setLessThanQueryValues:

– (void)**setLessThanQueryValues:**(NSDictionary *)*values*

Sets to *values* the receiver's dictionary of lessThan query values. The **qualifierFromQueryValues** method uses this dictionary along with the greaterThan and equalTo dictionaries to construct qualifiers.

See also: – **lessThanQueryValues**, – **setGreaterThanQueryValues:**, – **setEqualToQueryValues:**

setLocalKeys:

– (void)**setLocalKeys:**(NSArray *)*keys*

Sets the additional keys to which EOAssociations can be bound to *keys*. Interface Builder allows the user to add and remove local keys in the EODisplayGroup Attributes Inspector panel. See **localKeys** for more information.

setObjectArray:

– (void)**setObjectArray:**(NSArray *)*objects*

Sets the receiver's objects to *objects*, regardless of what its EODataSource provides. This method doesn't affect the EODataSource's objects at all; specifically, it results in neither inserts or deletes of objects in the EODataSource. *objects* should contain objects with the same property names or methods as those accessed by the receiver. This method is used by **fetch** to set the array of fetched objects; you should rarely need to invoke it directly.

After setting the object array, this method restores as much of the original selection as possible by invoking **selectObjectsIdenticalTo:selectFirstOnNoMatch:**. If there's no match and the receiver selects after fetching, then the first object is selected.

See also: – **allObjects**, – **displayedObjects**, – **selectsFirstObjectAfterFetch**

setQualifier:

– (void)**setQualifier:**(EOQualifier *)*aQualifier*

Sets the receiver's qualifier to *aQualifier*. This qualifier is used to filter the receiver's array of objects for display when the delegate doesn't do so itself. Use **updateDisplayedObjects** to apply the qualifier.

If the receiver's delegate responds to **displayGroup:displayArrayForObjects:**, that method is used instead of the qualifier to filter the objects.

See also: + **qualifierWithQualifierFormat:** (EOQualifier class of the control layer),
– **updateDisplayedObjects**, – **displayedObjects**, – **qualifier**, – **qualifierFromQueryValues**

setSelectedObjectValue:forKey:

– (BOOL)**setSelectedObjectValue:**(id)*value* **forKey:**(NSString *)*key*

Invokes **setValue:forObject:key:** with the first selected object, returning YES if successful and NO otherwise. This method should be invoked only by EOAssociation objects to propagate changes from display objects.

See also: – **setValue:forObjectAtIndex:key:**, – **valueForKey:object:**

setSelectionIndexes:

– (BOOL)**setSelectionIndexes:**(NSArray *)*indexes*

Selects the objects at *indexes* in the receiver's array if possible, returning YES if successful and NO if not (in which case the selection remains unaltered). *indexes* is an array of NSNumbers. This method is the primitive method for altering the selection; all other such methods invoke this one to make the change.

This method invokes **endEditing** to wrap up any changes being made by the user. If **endEditing** returns NO, this method fails and returns NO. This method then checks the delegate with a **displayGroup:shouldChangeSelectionToIndexes:** message. If the delegate returns NO, this method also fails and returns NO. If the receiver successfully changes the selection, its observers each receive a **subjectChanged** message.

See also: – **allObjects**

setSelectsFirstObjectAfterFetch:

– (void)**setSelectsFirstObjectAfterFetch:**(BOOL)*flag*

Controls whether the receiver automatically selects its first displayed object after a fetch when there were no selected objects before the fetch. If *flag* is YES it does; if *flag* is NO then no objects are selected.

EODisplayGroups by default do select the first object after a fetch when there was no previous selection.

See also: – **displayedObjects**, – **fetch**, – **selectsFirstObjectAfterFetch**

setSortOrdering:

– (void)**setSortOrdering:**(NSArray *)*orderings*

Sets the EOSortOrdering objects that **updateDisplayedObjects** uses to sort the displayed objects to *orderings*. Use **updateDisplayedObjects** to apply the sort orderings.

If the receiver's delegate responds to **displayGroup:displayArrayForObjects:**, that method is used instead of the sort orderings to order the objects.

See also: + **sortOrderingWithKey:selector:** (EOSortOrdering class of the control layer),
– **displayedObjects**, – **sortOrdering**

setUsesOptimisticRefresh:

– (void)**setUsesOptimisticRefresh:**(BOOL)*flag*

Controls how the receiver redisplays on changes to objects. If *flag* is YES it redisplays only when elements of its displayed objects array change; if *flag* is NO it redisplays on any change in its EOEditingContext. Because changes to other objects can affect the displayed objects (through flattened attributes or custom

methods, for example), EODisplayGroups by default use the more pessimistic refresh technique of redisplaying on any change in the EOEditingContext. If you know that none of the EOAssociations for a particular EODisplayGroup display derived values, you can turn on optimistic refresh to reduce redisplay time.

The default is NO. You can also change this setting in Interface Builder's Inspector panel using the Refresh All check box.

See also: – **usesOptimisticRefresh**

setValidatesChangesImmediately:

– (void)**setValidatesChangesImmediately:**(BOOL)*flag*

Controls the receiver's behavior on encountering a validation error. Whenever an EODisplayGroup sets a value in an object, it sends the object a **validateValue:forKey:** message, allowing the object to coerce the value's type to a more appropriate one or to return an exception indicating that the value isn't valid. If this method is invoked with a *flag* of YES, the receiver immediately presents an attention panel indicating the validation error. If this method is invoked with a *flag* of NO, the receiver leaves validation errors to be handled when changes are saved.

EODisplayGroups by default don't validate changes immediately.

See also: – **saveChanges** (EOEditingContext), – **validatesChangesImmediately**

setValue:forObject:key:

– (BOOL)**setValue:**(id)*value*
forObject:(id)*anObject*
key:(NSString *)*key*

Sets a property of *anObject*, identified by *key*, to *value*. Returns YES if successful and NO otherwise. If a new value is set, sends the delegate a **displayGroup:didSetValue:forObject:key:** message.

This method should be invoked only by EOAssociation objects to propagate changes from display objects. Other application code should interact with the objects directly.

If the receiver validates changes immediately, it sends *anObject* a **validateValue:forKey:** message, returning NO if the object refuses to validate *value*. Otherwise, validation errors are checked by the EOEditingContext when it attempts to save changes.

See also: – **setValue:forObjectAtIndex:key:**, – **setSelectedObjectValue:forKey:**, – **valueForKey:object:**, – **validatesChangesImmediately**

setValue:forObjectAtIndex:key:

– (BOOL)**setValue:**(id)*value*
forObjectAtIndex:(unsigned int)*index*
key:(NSString *)*key*

Invokes **setValue:forObject:key:** with the object at *index*, returning YES if successful and NO otherwise. This method should be invoked only by EOAssociation objects to propagate changes from display objects.

See also: – **setSelectedObjectValue:forKey:**,– **valueForObjectAtIndex:key:**

sortOrdering

– (NSArray *)**sortOrdering**

Returns an array of EOSortOrdering objects that **updateDisplayedObjects** uses to sort the displayed objects, as returned by the **displayedObjects** method.

See also: – **setSortOrdering:**

updateDisplayedObjects

– (void)**updateDisplayedObjects**

Recalculates the receiver's displayed objects array and redisplays. If the delegate responds to **displayGroup:displayArrayForObjects:**, it's sent this message and the returned array is set as the EODisplayGroup's displayed object. Otherwise, the receiver applies its qualifier and sort ordering to its array of objects. In either case, any objects that were selected before remain selected in the new displayed objects array.

See also: – **redisplay**, – **allObjects**, – **displayedObjects**, – **selectedObjects**, – **qualifier**, – **sortOrdering**

updatedObjectIndex

– (int)**updatedObjectIndex**

Returns the index in the displayed objects array of the most recently updated object, or –1 if more than one object has changed. The return value is meaningful only when **contentsChanged** returns YES. Associations can use this method to optimize redisplay of their user interface objects.

usesOptimisticRefresh

– (BOOL)**usesOptimisticRefresh**

Returns YES if the receiver redisplays only when elements of its displayed objects array change, NO if it redisplays on any change in its EOEditingContext. Because changes to other objects can affect the displayed objects (through flattened attributes or custom methods, for example), EODisplayGroups by default use the more pessimistic refresh technique of redisplaying on any change in the EOEditingContext. If you know that none of the EOAssociations for a particular EODisplayGroup display derived values, you can turn on optimistic refresh to reduce redisplay time.

The default is NO. You can change this setting in Interface Builder's Inspector panel using the Refresh All check box.

See also: – **setUsesOptimisticRefresh:**

validatesChangesImmediately

– (BOOL)**validatesChangesImmediately**

Returns YES if the receiver immediately handles validation errors, or leaves them for the EOEditingContext to handle when saving changes. See **setValidatesChangesImmediately:** for more information.

EODisplayGroups by default don't validate changes immediately.

valueForKey:object:

– (id)**valueForKey:**(NSString *)*key* **object:**(id)*anObject*

Returns *anObject*'s value for the property identified by *key.*

See also: – **valueForObjectAtIndex:key:**

valueForObjectAtIndex:key:

– (id)**valueForObjectAtIndex:**(unsigned int)*index* **key:**(NSString *)*key*

Returns the value of the object at *index* for the property identified by *key.*

See also: – **valueForKey:object:**

Methods Implemented By the Delegate

displayGroup:createObjectFailedForDataSource:

– (void)**displayGroup:**(EODisplayGroup *)*aDisplayGroup*
createObjectFailedForDataSource:(id)*aDataSource*

Invoked from **insertObjectAtIndex:** to inform the delegate that *aDisplayGroup* has failed to create a new object for *aDataSource*. If the delegate doesn't implement this method, the EODisplayGroup instead runs an alert panel to inform the user of the failure.

displayGroupDidChangeDataSource:

– (void)**displayGroupDidChangeDataSource:**(EODisplayGroup *)*aDisplayGroup*

Informs the delegate that *aDisplayGroup*'s EODataSource has changed.

displayGroup:didDeleteObject:

– (void)**displayGroup:**(EODisplayGroup *)*aDisplayGroup* **didDeleteObject:**(id)*anObject*

Informs the delegate that *aDisplayGroup* has deleted *anObject*.

displayGroup:didFetchObjects:

– (void)**displayGroup:**(EODisplayGroup *)*aDisplayGroup* **didFetchObjects:**(NSArray *)*objects*

Informs the delegate that *aDisplayGroup* has fetched *objects*.

displayGroup:didInsertObject:

– (void)**displayGroup:**(EODisplayGroup *)*aDisplayGroup* **didInsertObject:**(id)*anObject*

Informs the delegate that *aDisplayGroup* has inserted *anObject*.

displayGroup:didSetValue:forObject:key:

– (void)**displayGroup:**(EODisplayGroup *)*aDisplayGroup*
didSetValue:(id)*value*
forObject:(id)*anObject*
key:(NSString *)*key*

Informs the delegate that *aDisplayGroup* has altered a property value of *anObject*. *key* identifies the property, and *value* is its new value.

displayGroup:displayArrayForObjects:

– (NSArray *)**displayGroup:**(EODisplayGroup *)*aDisplayGroup*
displayArrayForObjects:(NSArray *)*objects*

Invoked from **updateDisplayedObjects**, this method allows the delegate to filter and sort *aDisplayGroup*'s array of objects to limit which ones get displayed. *objects* contains all of *aDisplayGroup*'s objects. The delegate should filter any objects that shouldn't be shown and sort the remainder, returning a new array containing this group of objects. You can use the added NSArray methods **filteredArrayUsingQualifier:** and **sortedArrayUsingKeyOrderArray:** to create the new array.

If the delegate doesn't implement this method, the EODisplayGroup uses its own qualifier and sort ordering to update its displayed objects array.

See also: – **sortOrdering**, – **qualifier**, – **displayedObjects**

displayGroup:shouldChangeSelectionToIndexes:

– (BOOL)**displayGroup:**(EODisplayGroup *)*aDisplayGroup*
shouldChangeSelectionToIndexes:(NSArray *)*newIndexes*

Allows the delegate to prevent a change in selection by *aDisplayGroup*. *newIndexes* is the proposed new selection, an array of NSNumbers. If the delegate returns YES, the selection changes; if the delegate returns NO, the selection remains as it is.

displayGroup:shouldDeleteObject:

– (BOOL)**displayGroup:**(EODisplayGroup *)*aDisplayGroup* **shouldDeleteObject:**(id)*anObject*

Allows the delegate to prevent *aDisplayGroup* from deleting *anObject*. If the delegate returns YES, *anObject* is deleted; if the delegate returns NO, the deletion is abandoned.

displayGroup:shouldDisplayAlertWithTitle:message:

– (BOOL)**displayGroup:**(EODisplayGroup *)*aDisplayGroup*
shouldDisplayAlertWithTitle:(NSString *)*title*
message:(NSString *)*message*

Allows the delegate to prevent *aDisplayGroup* from displaying an attention panel with *title* and *message*. The delegate can return YES to allow *aDisplayGroup* to display the panel, or NO to prevent it from doing so (perhaps displaying a different attention panel).

displayGroup:shouldInsertObject:atIndex:

– (BOOL)**displayGroup:**(EODisplayGroup *)*aDisplayGroup*
shouldInsertObject:(id)*anObject*
atIndex:(unsigned int)*anIndex*

Allows the delegate to prevent *aDisplayGroup* from inserting *anObject* at *anIndex*. If the delegate returns YES, *anObject* is inserted; if the delegate returns NO, the insertion is abandoned.

displayGroup:shouldRedisplayForChangesInEditingContext:

– (BOOL)**displayGroup:**(EODisplayGroup *)*aDisplayGroup*
shouldRedisplayForEditingContextChangeNotification:(NSNotification *)*aNotification*

Invoked whenever *aDisplayGroup* receives an EOObjectsChangedInEditingContextNotification, this method allows the delegate to suppress redisplay based on the nature of the change that has occurred. If the delegate returns YES, *aDisplayGroup* redisplays; if it returns NO, *aDisplayGroup* doesn't. *aNotification* indicates the EOEditingContext that has changed, as well as which objects have changed and how. See the EOEditingContext class specification for information on EOObjectsChangedInEditingContextNotification.

See also: – **redisplay**

displayGroup:shouldRefetchForInvalidatedAllObjectsNotification:

– (BOOL)**displayGroup:**(EODisplayGroup *)*aDisplayGroup*
shouldRefetchForInvalidatedAllObjectsNotification:(NSNotification *)*aNotification*

Invoked whenever *aDisplayGroup* receives an EOInvalidatedAllObjectsInStoreNotification, this method allows the delegate to suppress refetching of the invalidated objects. If the delegate returns YES, *aDisplayGroup* immediately refetches its objects. If the delegate returns NO, *aDisplayGroup* doesn't immediately fetch, instead delaying until absolutely necessary. *aNotification* is an NSNotification. See the EOObjectStore and EOEditingContext class specifications for information on this notification.

displayGroupDidChangeSelection:

– (void)**displayGroupDidChangeSelection:**(EODisplayGroup *)*aDisplayGroup*

Informs the delegate that *aDisplayGroup*'s selection has changed.

displayGroupShouldFetch:

– (BOOL)**displayGroupShouldFetch:**(EODisplayGroup *)*aDisplayGroup*

Allows the delegate to prevent *aDisplayGroup* from fetching. If the delegate returns YES, *aDisplayGroup* performs the fetch; if the delegate returns NO, *aDisplayGroup* abandons the fetch.