EODatabaseChannel

Inherits From: NSObject

Declared In: EOAccess/EODatabaseChannel.h

Class Description

An EODatabaseChannel represents an independent communication channel to the database server its EODatabase object is connected to, and fetches records as instances of enterprise object classes specified in its EODatabase's EOModel. EODatabaseContexts use EODatabaseChannels to perform fetches and to lock rows in the database. All of these EODatabase... objects are used automatically by EOEditingContexts and other components of Enterprise Objects Framework. You rarely need to interact with them directly.

Fetching Objects

If you need to fetch objects directly using an EODatabaseChannel, you do so by issuing a **selectObjectsWithFetchSpecification:editingContext:** message to select them, followed by a **while** loop in which you fetch each individual object. This excerpt fetches Employee objects whose salary is greater than 40,000, in no particular order:

```
EODatabaseChannel *myChannel;        /* Assume this exists. */
EOEditingContext *editingContext;    /* Assume this exists. */
EOQualifier *salaryQualifier;
EOFetchSpecification *fetchSpec;

salaryQualifier = [EOQualifier qualifierWithQualifierFormat:@"salary > 40000"];
fetchSpec = [EOFetchSpecification fetchSpecificationWithEntityName:@"Employee"
    qualifier:salaryQualifier sortOrderings:nil];

[myChannel selectObjectsWithFetchSpecification:fetchSpec
    editingContext:editingContext];

while (eo = [myChannel fetchObject]) {
    /* Process the fetched eo. */
}
```

This excerpt first builds an EOFetchSpecification for the Employee entity, using an EOQualifier that matches salaries above 40,000. The **selectObjectsWithFetchSpecification:editingContext:** causes all rows in the database matching the qualifier to be selected, and the **fetchObject** messages retrieve objects for those rows in the order indicated by the EOFetchSpecification. Since this example doesn't use sort orderings, the objects' order is undetermined.

Method Types

Creating instances	– initWithDatabaseContext:
Getting cooperating objects	– adaptorChannel – databaseContext
Fetching objects	– selectObjectsWithFetchSpecification:editingContext: – isFetchInProgress – fetchObject – cancelFetch
Setting internal fetch state	– setCurrentEntity: – setCurrentEditingContext: – setIsLocking: – isLocking – setIsRefreshingObjects: – isRefreshingObjects
Setting the delegate	– setDelegate: – delegate

Class Methods

adaptorChannel

– (EOAdaptorChannel *)**adaptorChannel**

Returns the EOAdaptorChannel used by the receiver for communication with the database server.

cancelFetch

– (void)**cancelFetch**

Cancels any fetch in progress.

See also: – **isFetchInProgress**, – **selectObjectsWithFetchSpecification:editingContext:**, – **fetchObject**

databaseContext

– (EODatabaseContext *)**databaseContext**

Returns the EODatabaseContext that controls transactions for the receiver.

delegate

– (id)**delegate**

Returns the receiver's delegate. An EODatabaseChannel shares the delegate of its EODatabaseContext. See the EODatabaseContext class specification for the delegate methods you can implement.

See also: – **setDelegate:**

fetchObject

– (id)**fetchObject**

Fetches and returns the next object in the result set produced by a **selectObjectsWithFetchSpecification:editingContext:** message; returns **nil** if there are no more objects in the current result set or if an error occurs. This method uses the receiver's EOAdaptorChannel to fetch a row, records a snapshot with the EODatabaseContext if necessary, and creates an enterprise object from the row if the object doesn't already exist. The new object is sent an **awakeFromFetchInEditingContext:** message to allow it to finish setting up its state.

If no snapshot exists for the fetched object, the receiver sends its EODatabase a **recordSnapshot:forGlobalID:** message to record one. If a snapshot already exists (because the object was previously fetched), the receiver checks whether it should overwrite the old snapshot with the new one. It does so by asking the delegate with a **databaseContext:shouldUpdateCurrentSnapshot:newSnapshot:globalID:databaseChannel:** method. If the delegate doesn't respond to this method, the EODatabaseChannel overwrites the snapshot if it's locking or refreshing fetched objects. Further, if the EODatabaseChannel is refreshing fetched objects, it posts an EOObjectsChangedInStoreNotification on behalf of its EODatabaseContext (which causes any EOEditingContext using that EODatabaseContext to update its enterprise object with the values recorded in the new snapshot).

See the EODatabaseContext class specification for information on locking and update strategies, and the EOFetchSpecification class specification for information on refreshing fetched objects.

See also: – **cancelFetch**, – **isFetchInProgress**, – **isLocking**, – **isRefreshingObjects**

initWithDatabaseContext:

– **initWithDatabaseContext:**(EODatabaseContext *)*aDatabaseContext*

Initializes a newly allocated EODatabaseChannel with *aDatabaseContext* as the EODatabaseContext it works in. The new EODatabaseChannel retains *aDatabaseContext*, and creates an EOAdaptorChannel to communicate with the database server.

This is the designated initializer for the EODatabaseChannel class. Returns **self**, or **nil** if no more channels can be associated with *aDatabaseContext*.

isFetchInProgress

– (BOOL)**isFetchInProgress**

Returns YES if the receiver is fetching, NO otherwise. An EODatabaseChannel is fetching if it's been sent a successful **selectObjectsWithFetchSpecification:editingContext:** message. An EODatabaseChannel stops fetching when there are no more objects to fetch or when it's sent a **cancelFetch** message.

isLocking

– (BOOL)**isLocking**

Returns YES if the receiver is locking the objects selected, as determined by its EODatabaseContext's update strategy or the EOFetchSpecification used to perform the select. Returns NO otherwise. This method always returns NO when no fetch is in progress.

See also: – **locksObjects** (EOFetchSpecification), – **setIsLocking:**

isRefreshingObjects

– (BOOL)**isRefreshingObjects**

Returns YES if the receiver overwrites existing snapshots with fetched values and causes the current EOEditingContext to overwrite existing enterprise objects with those values as well. Returns NO otherwise. This behavior is controlled by the EOFetchSpecification used in a **selectObjectsWithFetchSpecification:editingContext:** message.

See also: – **refreshesRefetchedObjects** (EOFetchSpecification), – **fetchObject**, – **setIsRefreshingObjects:**

selectObjectsWithFetchSpecification:editingContext:

– (void)**selectObjectsWithFetchSpecification:**(EOFetchSpecification *)*fetchSpecification*
editingContext:(EOEditingContext *)*anEditingContext*

Selects objects described by *fetchSpecification* so that they'll be fetched into *anEditingContext*. The selected objects compose one or more result sets, each object of which will be returned by subsequent **fetchObject** messages in the order prescribed by *fetchSpecification*'s EOSortOrderings.

Raises an exception if an error occurs; the particular exception depends on the specific error, and is indicated in the exception's description. Some possible reasons for failure are:

- *fetchSpecification* is invalid.
- The receiver's EODatabaseContext has no transaction in progress.
- The delegate disallows the select operation.
- The receiver's EOAdaptorChannel fails to perform the select operation.

This method invokes the delegate methods **databaseContext:shouldSelectObjectsWithFetchSpecification:databaseChannel:**, **databaseContext:shouldUsePessimisticLockWithFetchSpecification:databaseChannel:**, and **databaseContext:didSelectObjectsWithFetchSpecification:databaseChannel:**. See their descriptions in the EODatabaseContext class specification for more information.

See also: – **fetchObject**

setCurrentEditingContext:

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

Sets the EOEditingContext that's made the owner of fetched objects to *anEditingContext*. This method is automatically invoked by **selectObjectsWithFetchSpecification:editingContext:**. You should never invoke it directly.

See also: – **setCurrentEntity:**

setCurrentEntity:

– (void)**setCurrentEntity:**(EOEntity *)*anEntity*

Sets the EOEntity used when fetching enterprise objects to *anEntity*. Subsequent **fetchObject** messages during a fetch operation create an object of the class associated with *anEntity*. This method is invoked automatically by **selectObjectsWithFetchSpecification:editingContext:**. You should never need to invoke it directly.

See also: – **setCurrentEditingContext:**

setDelegate:

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

Sets the receiver's delegate to *anObject*. An EODatabaseChannel shares the delegate of its EODatabaseContext; you should never invoke this method directly. See the EODatabaseContext class specification for the delegate methods you can implement.

See also: – **delegate**

setIsLocking:

– (void)**setIsLocking:**(BOOL)*flag*

Records whether the receiver locks the records it selects. An EODatabaseChannel modifies its interaction with the database server and its snapshotting behavior based on this setting. If *flag* is YES the EODatabaseChannel modifies its fetching behavior to lock objects; if *flag* is NO it simply fetches them.

An EODatabaseChannel automatically sets this flag according to the fetch specification used in a **selectObjectsWithFetchSpecification:editingContext:** message. You might invoke this method directly if evaluating SQL directly with EOAdaptorChannel's **evaluateExpression:** method.

See also: – **locksObjects** (EOFetchSpecification), – **setIsLocking:**

setIsRefreshingObjects:

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

Records whether the receiver causes existing snapshots and enterprise objects to be overwritten with fetched values. If *flag* is YES the receiver overwrites existing snapshots with fetched values and posts an EOObjectsChangedInStoreNotification on behalf of its EODatabaseContext (which typically causes the an existing object's EOEditingContext to replace its values with the new ones). If *flag* is NO, the receiver relies on the delegate to determine whether snapshots should be overwritten, and doesn't cause enterprise objects to be overwritten.

An EODatabaseChannel automatically sets this flag according to the fetch specification used in a **selectObjectsWithFetchSpecification:editingContext:** message. You might invoke this method directly if evaluating SQL directly with EOAdaptorChannel's **evaluateExpression:** method.

See also: – **refreshesRefetchedObjects** (EOFetchSpecification)