EOFetchSpecification

Inherits From:	NSObject
Conforms To:	NSCopying NSObject (NSObject)
Declared In:	EOControl/EOFetchSpecification.h

Class Description

An EOFetchSpecification collects the criteria needed to select and order a group of records or enterprise objects, whether from an external repository such as a relational database or an internal store such as an EOEditingContext. An EOFetchSpecification contains these elements:

- The name of an entity for which to fetch records or objects. This is the only mandatory element.
- An EOQualifier, indicating which properties to select by and how to do so.
- An array of EOSortOrderings, which indicate how the selected records or objects should be ordered when fetched.
- An indicator of whether to produce distinct results or not. Normally if a record or object is selected several times, such as when forming a join, it appears several times in the fetched results. An EOFetchSpecification that makes distinct selections causes duplicates to be filtered out, so that each record or object selected appears exactly once in the result set.
- An indicator of whether to fetch deeply or not. This is used with inheritance hierarchies when fetching for an entity with sub-entities. A deep fetch produces all instances of the entity and its sub-entities, while a shallow fetch produces instances only of the entity in the fetch specification.
- A dictionary of hints, which an EODatabaseContext or other object can use to optimize or alter the results of the fetch.

EOFetchSpecifications are used most often in the **objectsWithFetchSpecification:editingContext:** method defined by EOObjectStore, EOEditingContext, and EODatabaseContext, as well as **objectsWithFetchSpecification:**, defined by EOEditingContext alone. EOAdaptorChannel and EODatabaseChannel also define methods that use EOFetchSpecifications.

Adopted Protocols

NSCopying	– copyWithZone:

Method Types

Creating instances	+ fetchSpecificationWithEntityName:qualifier:sortOrderings: – init – initWithEntityName:qualifier:sortOrderings:usesDistinct: isDeep:hints:
Configuring fetch parameters	– setEntityName: – entityName – setQualifier: – qualifier – setSortOrderings: – sortOrderings – setUsesDistinct: – usesDistinct – setLocksObjects: – locksObjects – setIsDeep: – isDeep – setRefreshesRefetchedObjects: – refreshesRefetchedObjects – setHints: – hints

Class Methods

fetchSpecificationWithEntityName:qualifier:sortOrderings:

+ (EOFetchSpecification *)**fetchSpecificationWithEntityName:**(NSString *)*entityName*
qualifier:(EOQualifier *)*qualifier*
sortOrderings:(NSArray *)*sortOrderings*

Returns an EOFetchSpecification for *entityName*, using *qualifier* to select and *sortOrderings* to sort the results. This EOFetchSpecification doesn't perform distinct selection, is deep, and has no hints.

See also: – **initWithEntityName:qualifier:sortOrderings:usesDistinct:isDeep:hints:**

Instance Methods

entityName

– (NSString *)**entityName**

Returns the name of the entity to be fetched.

See also: – **isDeep**, – **setEntityName:**

hints

– (NSDictionary *)**hints**

Returns the receiver's hints, which other objects can use to alter or optimize fetch operations. EODatabaseContext's **objectsWithFetchSpecification:editingContext:** uses a hint to prefetch the destinations of relationships, for example. See the EODatabaseContext class specification for more information.

See also: – **setHints:**

init

– (id)**init**

Initializes a new EOFetchSpecification with no state, except that it fetches deeply and doesn't use distinct. Use the **set...** methods to add other parts of the specification. This is the designated initializer for the EOFetchSpecification class. Returns **self**.

See also: – **initWithEntityName:qualifier:sortOrderings:usesDistinct:isDeep:hints:**

initWithEntityName:qualifier:sortOrderings:usesDistinct:isDeep:hints:

– (id)**initWithEntityName:**(NSString *)*entityName*
qualifier:(EOQualifier *)*qualifier*
sortOrderings:(NSArray *)*sortOrderings*
usesDistinct:(BOOL)*distinctFlag*
isDeep:(BOOL)*deepFlag*
hints:(NSDictionary *)*hints*

Initializes a new EOFetchSpecification with the given arguments. Returns **self**.

See also: + **fetchSpecificationWithEntityName:qualifier:sortOrderings:**

isDeep

– (BOOL)**isDeep**

Returns YES if a fetch should include sub-entities of the receiver's entity, NO if it shouldn't. EOFetchSpecifications are deep by default.

For example, if you have a Person entity with two sub-entities, Employee and Customer, fetching Persons deeply also fetches all Employees and Customers matching the qualifier, and fetching Persons shallowly fetches only Persons matching the qualifier.

See also: – **setIsDeep:**

locksObjects

– (BOOL)**locksObjects**

Returns YES if a fetch should result in the selected objects being locked in the data repository, NO if it shouldn't. The default is NO.

See also: – **setLocksObjects:**

qualifier

– (EOQualifier *)**qualifier**

Returns the EOQualifier the indicates which records or objects to fetch.

See also: – **setQualifier:**

refreshesRefetchedObjects

– (BOOL)**refreshesRefetchedObjects**

Returns YES if existing objects are overwritten with fetched values when they've have been updated or changed. Returns NO if existing objects aren't touched when their data is refetched (the fetched data is simply discarded). The default is NO.

See also: – **setRefreshesRefetchedObjects:**

setEntityName:

– (void)**setEntityName:**(NSString *)*entityName*

Sets the name of the root entity to be fetched to *entityName*.

See also: – **isDeep**, – **entityName**

setHints:

– (void)**setHints:**(NSDictionary *)*hints*

Sets the receiver's hints to *hints*. Other objects can use these to alter or optimize fetch operations. See the description of the **objectsWithFetchSpecification:editingContext:** method in the EODatabaseContext class specification for an example.

See also: – **hints**

setIsDeep:

– (void)**setIsDeep:**(BOOL)*flag*

Controls whether a fetch should include sub-entities of the receiver's entity. If *flag* is YES, sub-entities are also fetched; if *flag* is NO, they aren't. EOFetchSpecifications are deep by default.

For example, if you have a Person entity/class/table with two sub-entities and subclasses, Employee and Customer, fetching Persons deeply also fetches all Employees and Customers matching the qualifier, while fetching Persons shallowly fetches only Persons matching the qualifier.

See also: – **isDeep**

setLocksObjects:

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

Controls whether a fetch should result in the selected objects being locked in the data repository. If *flag* is YES it should, if NO it shouldn't. The default is NO.

See also: – **locksObjects**

setQualifier:

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

Sets the receiver's qualifier to *qualifier*.

See also: – **qualifier**

setRefreshesRefetchedObjects:

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

Controls whether existing objects are overwritten with fetched values when they've have been updated or changed. If *flag* is YES, they are; if *flag* is NO, they aren't (the fetched data is simply discarded). The default is NO.

– **refreshesRefetchedObjects**

setSortOrderings:

– (void)**setSortOrderings:**(NSArray *)*sortOrderings*

Sets the receiver's array of EOSortOrderings to *sortOrderings*. When a fetch is performed with the receiver, the results are sorted by applying each EOSortOrdering in the array.

See also: – **sortedArrayUsingKeyOrderArray:** (NSArray Additions), – **sortOrderings**

setUsesDistinct:

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

Controls whether duplicate objects or records are removed after fetching. If *flag* is YES they're removed; if *flag* is NO they aren't. EOFetchSpecifications by default don't use distinct.

See also: – **usesDistinct**

sortOrderings

– (NSArray *)**sortOrderings**

Returns the receiver's array of EOSortOrderings. When a fetch is performed with the receiver, the results are sorted by applying each EOSortOrdering in the array.

See also: – **sortedArrayUsingKeyOrderArray:** (NSArray Additions), – **setSortOrderings:**

usesDistinct

– (BOOL)**usesDistinct**

Returns YES if duplicate objects or records are removed after fetching, NO if they aren't. EOFetchSpecifications by default don't use distinct.

See also: – **setUsesDistinct:**