EOEntityClassDescription

Inherits From:	EOClassDescription : NSObject
Conforms To:	NSObject (NSObject)
Declared In:	EOAccess/EOEntity.h

Class Description

EOEntityClassDescription is the subclass of EOClassDescription provided by the Enterprise Objects Framework access layer. The EOClassDescription class provides a mechanism for extending classes by giving them access to metadata not available in the Objective-C run-time system. EOEntityClassDescription extends the behavior of enterprise objects by deriving information about them (such as NULL constraints and referential integrity rules) from an associated EOModel file.

In the typical scenario in which an enterprise object has a corresponding model file, the first time a particular operation is performed on a class (such as validating a value), an EOClassDescriptionNeeded notification is broadcast. When an EOModel object receives this notification it registers the metadata (class description) for the EOEntity on which the enterprise object is based. This class description is used from that point on.

For a more detailed discussion of this subject, see the EOClassDescription class specification.

Method Types

Initializing an EOEntityClassDescription	– initWithEntity:
Returning an EOEntityClassDescription’s entity	– entity
Allocating new object instances	– createInstanceWithEditingContext:globalID:zone:

Returning information from the EOEntityClassDescription	– entityName – attributeKeys – classDescriptionForDestinationKey: – toManyRelationshipKeys – toOneRelationshipKeys – inverseForRelationshipKey: – ownsDestinationObjectsForRelationshipKey: – deleteRuleForRelationshipKey:
Performing validation	– validateObjectForDelete: – validateObjectForSave: – validateValue:forKey:
Handling newly inserted objects	– awakeObject:fromInsertionInEditingContext:

Instance Methods

attributeKeys

– (NSArray *)**attributeKeys**

Overrides EOClassDescription's **attributeKeys** method to return all of the EOAttributes that are class properties of the receiver's EOEntity.

See also: – **entityName**, – **toOneRelationshipKeys,** – **toManyRelationshipKeys**

awakeObject:fromInsertionInEditingContext:

– (void)**awakeObject:**(id)*object*
fromInsertionInEditingContext:(EOEditingContext *)*anEditingContext*

Overrides EOClassDescription's **awakeObject:fromInsertionInEditingContext:** method to propagate inserts for the newly inserted *object* in *anEditingContext*. More specifically, if *object* has a relationship (or relationships) that propagates the object's primary key and if no object yet exists at the destination of that relationship, creates the new object at the destination of the relationship.

classDescriptionForDestinationKey:

– (EOClassDescription *)**classDescriptionForDestinationKey:**(NSString *)*detailKey*

Overrides EOClassDescription's **classDescriptionForDestinationKey:** method to return the EOClassDescription for objects at the destination of the EORelationship identified by *detailKey.* For example, the statement:

```
[project classDescriptionForDestinationKey:@"leader"];
```

might return the class description for the Employee entity.

createInstanceWithEditingContext:globalID:zone:

– (id)**createInstanceWithEditingContext:**(EOEditingContext *)*anEditingContext*
globalID:(EOGlobalID *)*globalID* **zone:**(NSZone *)*zone*

Overrides EOClassDescription's **createInstanceWithEditingContext:globalID:zone:** method to allocate an object of the appropriate class in *anEditingContext*, with *globalID*, in *zone*. Enterprise Objects Framework uses this method to allocate new instances of objects when fetching existing enterprise objects or inserting new ones in an EODisplayGroup.

deleteRuleForRelationshipKey:

– (EODeleteRule)**deleteRuleForRelationshipKey:**(NSString *)*relationshipKey*

Overrides EOClassDescription's **deleteRuleForRelationshipKey:** method to return the EODeleteRule for the EORelationship specified by *relationshipKey*. This EORelationship is defined for an EOEntity in the receiver. The returned EODeleteRule is one of the following:

EODeleteRuleNullify
EODeleteRuleCascade
EODeleteRuleDeny

For example, suppose you have a department with multiple employees. When a user tries to delete the department, your application could:

- Delete the department and remove any back pointers from the employee to the department (nullify)
- Delete the department and all of the employees it contains (cascade)
- Refuse the deletion if the department contains employees (deny)

entity

– (EOEntity *)**entity**

Returns the entity associated with the receiver.

entityName

– (NSString *)**entityName**

Overrides EOClassDescription's **entityName** method to return the name of the receiver's EOEntity.

See also: – **attributeKeys**, – **toOneRelationshipKeys,** – **toManyRelationshipKeys**

initWithEntity:

– **initWithEntity:**(EOEntity *)*anEntity*

Initializes a newly allocated EOEntityClassDescription with *anEntity.* Returns **self**.

inverseForRelationshipKey:

– (NSString *)**inverseForRelationshipKey:**(NSString *)*relationshipKey*

Overrides EOClassDescription's **inverseForRelationshipKey:** method to return the name of the EORelationship pointing back at the receiver from the destination of the EORelationship specified by *relationshipKey.* This method works by returning the name of the EORelationship returned by the corresponding EORelationship's **inverseRelationship** method.

For example, suppose an Employee object has a relationship called **department** to a Department object, and Department has a relationship called **employees** back to Employee. The statement

```
[employee inverseForRelationshipKey:@"department"];
```

returns the string "employees".

ownsDestinationObjectsForRelationshipKey:

– (BOOL)**ownsDestinationObjectsForRelationshipKey:**(NSString *)*relationshipKey*

Overrides EOClassDescription's **ownsDestinationObjectsForRelationshipKey:** method to return YES or NO to indicate whether the objects at the destination of the EORelationship specified by *relationshipKey* should be deleted if they are removed from the relationship (and not transferred to the corresponding relationship of another object). For example, an Invoice object owns its line items. If a LineItem object is removed from an Invoice it should be deleted since it can't exist outside of an Invoice.

This method works by returning the result of the EORelationship's **ownsDestination** method.

toManyRelationshipKeys

– (NSArray *)**toManyRelationshipKeys**

Overrides EOClassDescription's **toManyRelationshipKeys** method to return all of the to-many EORelationships that are marked as class properties in the receiver's EOEntity.

See also: – **entityName**, – **toOneRelationshipKeys,** – **attributeKeys**

toOneRelationshipKeys

– (NSArray *)**toOneRelationshipKeys**

Overrides EOClassDescription's **toOneRelationshipKeys** method to return all of the to-one EORelationships that are marked as class properties in the receiver's EOEntity.

See also: – **entityName**, – **toManyRelationshipKeys,** – **attributeKeys**

validateObjectForDelete:

– (NSException *)**validateObjectForDelete:**(id)*object*

Overrides EOClassDescription's **validateObjectForDelete:** method to determine whether it's permissible to delete the object. Returns **nil** if the delete operation should proceed, or an unevaluated exception containing a user-presentable (localized) error message if not.

validateObjectForSave:

– (NSException *)**validateObjectForSave:**(id)*object*

Overrides EOClassDescription's **validateObjectForSave:** method to determine whether the values being saved for the object are acceptable. Returns **nil** if the values are acceptable and the save operation should therefore proceed, or an unevaluated exception containing a user-presentable (localized) error message if not.

validateValue:forKey:

– (NSException *)**validateValue:**(id *)*valueP* **forKey:**(NSString *)*key*

Overrides EOClassDescription's **validateValue:forKey:** method to validate the value associated with *key*, and pointed to by *valueP*. Looks up the corresponding EORelationship or EOAttribute in the EOEntity for the receiver and forwards the validation request to it. For example, for an EOAttribute this method checks that the data type for the value matches the type in the EOModel and that width and "allows NULL" constraints have not been violated.

Returns **nil** if the value is acceptable, or an unevaluated exception containing a user-presentable (localized) error message if not.