EOFaultHandler

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

Class Description

EOFaultHandler is the abstract class defining the mechanism that helps an EOFault to fire. Subclasses of EOFaultHandler perform the specific steps necessary to get data for the EOFault and fire it. The Access layer, for example, uses private subclasses to fetch data using an EODatabaseContext. Most of EOFaultHandler's methods are properly defined; you need only override **completeInitializationOfObject:** to provide appropriate behavior. In addition, however, you can optionally implement **faultWillFire:** to prepare for conversion, and **shouldPerformInvocation:** to intercept particular messages sent to the EOFault without causing it to fire.

You create an EOFaultHandler using the standard **alloc** and **init** methods, possibly using a more specific **init** method with your subclass. To create an EOFault, you invoke EOFault's **makeObjectIntoFault:withHandler:** class method with the object to turn into an EOFault and the EOFaultHandler. An EOFaultHandler belongs exclusively to a single EOFault, and shouldn't be shared or used by any other object.

Firing an EOFault

When an EOFault receives a message that requires it to fire, it sends a **completeInitializationOfObject:** method to its EOFaultHandler. This method is responsible for invoking EOFault's **clearFault:** class method to revert the EOFault to its original state, and then do whatever is necessary to complete initialization of the object. Doing so typically involves fetching data from an external repository and passing it to the object.

As a trivial example, consider a subclass called FileFaultHandler, that simply stores a filename whose contents it reads from disk. Its initialization and **completeInitializationOfObject:** methods might look like these:

```
- (id)initWithFile:(NSString *)path
{
    self = [super init];
    filename = [path copy];
    return self;
}
```

```
- (void)completeInitializationOfObject:(id)anObject
{
    NSString *fileContents;

    [EOFault clearFault:anObject];
    // This releases self; be sure to retain if needed!

    fileContents = [NSString stringWithContentsOfFile:filename];
    [anObject takeValue:fileContents forKey:@"fileContents"];
    return;
}
```

initWithFile: just stores the path of the file to read in the instance variable **filename**.
completeInitializationOfObject: invokes EOFault's **clearFault:** method, which reverts the EOFault into its original state (and also releases the fault handler, so references to **self** after this are illegal). It then gets the contents of the file it was created with and passes them to the reverted object. Note that this implementation doesn't assume the class of the cleared EOFault, instead using the generic **takeValue:forKey:** method to assign the file contents to it.

Method Types

Setting the target class and extra data	– setTargetClass:extraData: – targetClass – extraData
Reference counting	– incrementExtraRefCount – decrementExtraRefCountWasZero – extraRefCount
Getting the original class	– classForFault:
Firing a fault	– completeInitializationOfObject: – faultWillFire: – shouldPerformInvocation:
Getting a description	– descriptionForObject:
Checking class information	– isKindOfClass:forFault: – isMemberOfClass:forFault: – conformsToProtocol:forFault: – methodSignatureForSelector:forFault: – respondsToSelector:forFault:

Instance Methods

classForFault:

– (Class)**classForFault:**(id)*aFault*

Returns the target class of the receiver's EOFault, which must be passed as *aFault* in case the receiver needs to fire it (EOFaultHandlers don't store back pointers to their EOFaults). For example, to support entity inheritance, the Access layer fires EOFaults for entities with subentities to confirm their precise class membership.

See also: – **completeInitializationOfObject:**

completeInitializationOfObject:

– (void)**completeInitializationOfObject:**(id)*aFault*

Implemented by subclasses to revert *aFault* to its original state and complete its initialization in whatever means is appropriate to the subclass. For example, the Access layer subclasses of EOFaultHandler fetch data from the database and pass it to the object. This method is invoked automatically by an EOFault when it's sent a message that it can't handle without reverting to its original class. EOFaultHandler's implementation merely raises an exception.

conformsToProtocol:forFault:

– (BOOL)**conformsToProtocol:**(Protocol *)*aProtocol* **forFault:**(id)*aFault*

Returns YES if the target class of the receiver's EOFault conforms to *aProtocol*. This EOFault must be passed as *aFault* in case the receiver needs to fire it (EOFaultHandlers don't store back pointers to their EOFaults). For example, to support entity inheritance, the Access layer fires EOFaults for entities with subentities to confirm their precise class membership.

See also: – **completeInitializationOfObject:**

decrementExtraRefCountWasZero

– (BOOL)**decrementExtraRefCountWasZero**

Used by EOFaultHandler's internal reference counting mechanism, this method functions as the Foundation function **NSDecrementExtraRefCountWasZero()** for the receiver's EOFault.

descriptionForObject:

– (NSString *)**descriptionForObject:**(id)*aFault*

Returns a string naming the original class of the receiver's EOFault and giving *aFault*'s **id**, and also noting that it's a fault; for example: "<Employee (Fault 0x3a07)>". (The EOFault must be passed as *aFault* because EOFaultHandlers don't store back pointers to their EOFaults.)

extraData

– (void *)**extraData**

Returns the bytes replaced by the receiver's **id** in the original object's state, as a pointer to **void**. When the receiver's EOFault is reverted to its original state, both its **isa** pointer and this data are replaced.

extraRefCount

– (unsigned int)**extraRefCount**

Used by EOFaultHandler's internal reference counting mechanism, this method functions as the Foundation function **NSExtraRefCount()** for the receiver's EOFault.

faultWillFire:

– (void)**faultWillFire:**(id)*aFault*

Invoked from EOFault's **clearFault:** method to inform the receiver that *aFault* is about to be reverted to its original state. EOFaultHandler's implementation does nothing.

incrementExtraRefCount

– (void)**incrementExtraRefCount**

Used by EOFaultHandler's internal reference counting mechanism, this method functions as the Foundation function **NSIncrementExtraRefCount()** for the receiver's EOFault.

isKindOfClass:forFault:

– (BOOL)**isKindOfClass:**(Class)*aClass* **forFault:**(id)*aFault*

Returns YES if the target class of the receiver's EOFault is *aClass* or a subclass of *aClass*. This EOFault must be passed as *aFault* in case the receiver needs to fire it (EOFaultHandlers don't store back pointers to

their EOFaults). For example, to support entity inheritance, the Access layer fires EOFaults for entities with subentities to confirm their precise class membership.

See also: – **completeInitializationOfObject:**

isMemberOfClass:forFault:

– (BOOL)**isMemberOfClass:**(Class)*aClass* **forFault:**(id)*aFault*

Returns YES if the target class of the receiver's EOFault is *aClass*. This EOFault must be passed as *aFault* in case the receiver needs to fire it (EOFaultHandlers don't store back pointers to their EOFaults). For example, to support entity inheritance, the Access layer fires EOFaults for entities with subentities to confirm their precise class membership.

See also: – **completeInitializationOfObject:**

methodSignatureForSelector:forFault:

– (NSMethodSignature *)**methodSignatureForSelector:**(SEL)*aSelector* **forFault:**(id)*aFault*

Returns the NSMethodSignature for *aSelector* in the target class of the receiver's EOFault, which must be passed as *aFault* in case the receiver needs to fire it (EOFaultHandlers don't store back pointers to their EOFaults). For example, to support entity inheritance, the Access layer fires EOFaults for entities with subentities to confirm their precise class membership.

See also: – **completeInitializationOfObject:**

respondsToSelector:forFault:

– (BOOL)**respondsToSelector:**(SEL)*aSelector* **forFault:**(id)*aFault*

Returns YES if the target class of the receiver's EOFault responds to *aSelector*. This EOFault must be passed as *aFault* in case the receiver needs to fire it (EOFaultHandlers don't store back pointers to their EOFaults). For example, to support entity inheritance, the Access layer fires EOFaults for entities with subentities to confirm their precise class membership.

See also: – **completeInitializationOfObject:**

setTargetClass:extraData:

– (void)**setTargetClass:**(Class)*targetClass* **extraData:**(void *)*extraData*

Stores *targetClass* and *extraData* as state of the original object overwritten when an EOFault is created by EOFault's **makeObjectIntoFault:withHandler:** method, which replaces *targetClass* with the EOFault class, and *extraData* with the EOFaultHandler's **id**.

shouldPerformInvocation:

– (BOOL)**shouldPerformInvocation:**(NSInvocation *)*anInvocation*

Overridden by subclasses to circumvent reversion of an EOFault to its original state. Returns YES if the EOFault should revert and perform *anInvocation*, NO if it shouldn't. If this method returns NO, the receiver should set *anInvocation*'s return value appropriately. EOFaultHandler's implementation returns YES.

See also: – **setReturnValue:** (NSInvocation class of the Foundation Framework)

targetClass

– (Class)**targetClass**

Returns the target class of the receiver's EOFault. The EOFault may, however, be converted to a member of this class or of a subclass of this class. For example, to support entity inheritance, the Access layer fires EOFaults for entities with subentities into the appropriate class on fetching their data.