EOF-Using EOF-Batch Fetching

Batch Fetching

Petite Abeille

Here is a little example about how to batch fetch a bunch of EOFaults:

- First, given an array of faults, how to retrieve meta information without triggering the fault:

```
EOEnterpriseObject aReferenceObject = (EOEnterpriseObject)someFaults.lastObject();
EOAccessFaultHandler aFaultHandler = (EOAccessFaultHandler) EOFaultHandler.handlerForFault(aReferenceObject);
EOEditingContext anEditingContext = aFaultHandler.editingContext();
String anEntityName = aFaultHandler.globalID().entityName();
EOEntity anEntity =
EOModelGroup.defaultGroup().entityNamed( anEntityName );
```

- Second, for each EOFault build a qualifier:

```
EOEnterpriseObject aFault = (EOEnterpriseObject) someFaults.objectAtIndex(index);
EOKeyGlobalID aGlobalID = (EOKeyGlobalID) anEditingContext.globalIDForObject( aFault );
NSDictionary aPrimaryKey = anEntity.primaryKeyForGlobalID( aGlobalID );
EOQualifier aQualifier = anEntity.qualifierForPrimaryKey( aPrimaryKey );
```

- Finally, resolve the qualifier:

```
EOQualifier aQualifier = new EOOrQualifier( someQualifiers );
  EOFetchSpecification aFetchSpecification = new EOFetchSpecification( anEntityName, aQualifier, null);
  anEditingContext.objectsWithFetchSpecification?(aFetchSpecification?);
```

- Attached is the entire method.

```
// ============================================================================
//     Batch fetching Class method(s)
// ----------------------------------------------------------------------------
public static void batchFetchObjects(NSArray someObjects) {
    if ( ( someObjects != null ) && ( someObjects.count() > 0 ) ) {
        NSMutableArray  someFaults = new NSMutableArray();
        int    count = someObjects.count();

        for ( int index = 0; index < count; index++ ) {
            Object  anObject = someObjects.objectAtIndex(index);

            if ( EOFaultHandler.isFault( anObject ) == true ) {
                someFaults.addObject( anObject );
            }
        }

        if ( someFaults.count() > 0 ) {
            NSMutableArray    someQualifiers = new NSMutableArray();
            EOEnterpriseObject  aReferenceObject = (EOEnterpriseObject) someFaults.lastObject();
            EOAccessFaultHandler  aFaultHandler = (EOAccessFaultHandler) EOFaultHandler.handlerForFault
(aReferenceObject);
            EOEditingContext  anEditingContext = aFaultHandler.editingContext();
            String      anEntityName = aFaultHandler.globalID().entityName();
            EOEntity    anEntity = EOModelGroup.defaultGroup().entityNamed( anEntityName );

            count = someFaults.count();

            for ( int index = 0; index < count; index++ ) {
                EOEnterpriseObject  aFault = (EOEnterpriseObject) someFaults.objectAtIndex(index);
                EOKeyGlobalID    aGlobalID = (EOKeyGlobalID) anEditingContext.globalIDForObject( aFault );
                NSDictionary    aPrimaryKey = anEntity.primaryKeyForGlobalID( aGlobalID );
                EOQualifier    aQualifier = anEntity.qualifierForPrimaryKey( aPrimaryKey );

                someQualifiers.addObject( aQualifier );
            }

            if ( someQualifiers.count() > 0 ) {
                EOQualifier    aQualifier = new EOOrQualifier( someQualifiers );
                EOFetchSpecification  aFetchSpecification = new EOFetchSpecification( anEntityName, aQualifier,
null);
                anEditingContext.objectsWithFetchSpecification(aFetchSpecification);
            }
        }
    }
}
```

Jonathan Rochkind

Can you explain the advantages of fetching faults instead of just letting the editingContext return them? -SamBarnum
A 'fault' will not actually be fetched until it is asked for. It's just an object in memory that says 'if anyone tries to access my data, I will fetch it from the db.' You usually don't run into a situation where you want to create faults instead of just fetching objects from the EC---but when you're doing complicated programming, it will be obvious when you do. I think just once have I actually manually created faults. [WO:When I had a semantic relationship, but couldn't actually model the relationship, and implemented a sort of half baked relationship scheme of my own instead.]
However, I'm not sure why you'd want to use the above code when the EOEditingContext method faultForRawRow is available. Make an NSDictionary which has pk attribute names as keys and desired values as their values, pass it to yourEC.faultForRawRow?, you've got a fault. A heck of a lot easier than the above.

Preloading Relationships

This method can be used to prefetch relationships in a batch instead of having them fire one by one. This can be used to fine tune batch fetching in code instead of doing it in the model. Be careful of how you use this method. It will force a fetch from the database even if the objects are already cached in the object store. Used incorrectly this can reduce, instead of increase, performance.

```
public static void preloadRelationship(NSArray sourceEOs, String relationshipName) {
    if (sourceEOs.count() != 0) {
        EOEnterpriseObject sampleEO = (EOEnterpriseObject) sourceEOs.objectAtIndex(0);
        EOEditingContext ec = sampleEO.editingContext();

        EOObjectStoreCoordinator osc = (EOObjectStoreCoordinator) ec.rootObjectStore();
        osc.lock();
        try {
            EOEntity entity = EOModelGroup.modelGroupForObjectStoreCoordinator(osc)
                    .entityNamed(sampleEO.entityName());

            EODatabaseContext databaseContext =  EODatabaseContext.
                    registeredDatabaseContextForModel(entity.model(), osc);

            EORelationship relationship = entity.relationshipNamed(relationshipName);

            databaseContext.batchFetchRelationship(relationship, sourceEOs, ec);
        } finally {
            osc.unlock();
        }
    }
}
```