EOF-Using EOF-Primary Keys

Accessing Primary Keys

When your primary key is not visible as a class property, the documented way to obtain the primary key values is with EOUtilities.primaryKeyForObject(). This works, but in the case where you have a single non-compound primary key, you need a few more lines of code to pull out this pk value. More troubling, the EOUtilities method, when called with a fault, will cause the fault to be resolved. The fault does not really need to be resolved to get the primary key, as the pk value is present even in the fault.

Here's a method that works for EOs which have a single Integer primary key, to pull out that pk value, and without resolving a fault (possibly causing a trip to the db) when called on a fault value:

```
public static Integer singleIntegerPrimaryKeyForObject(EOEnterpriseObject eo) {
    EOEditingContext ec = eo.editingContext();
    if (ec == null) {
        //you don't have an EC! Bad EO. We can do nothing.
        return null;
    }
    EOGlobalID gid = ec.globalIDForObject(eo);
    if (gid.isTemporary()) {
        //no pk yet assigned
        return null;
    }
    EOKeyGlobalID kGid = (EOKeyGlobalID) gid;
    NSArray keyValues = kGid.keyValuesArray();
    if (keyValues.count() != 1 ||
        ! (keyValues.objectAtIndex(0) instanceof Integer))
        return null;

    return (Integer) keyValues.objectAtIndex(0);

}
```

You can also use the EOUtilities method but please note that, at least as of WO5, there are two somewhat undesirable aspects of this EOUtilities code:

- If the EO is a fault, calling the EOUtilities method will cause the fault to be resolved, possibly requiring a trip to the db, even though pk info is already present in the fault and the trip to the db is really unneccesary at this point.
- If the EO has just been inserted and not yet committed, it does not have a pk yet, and the EOUtilities code will throw an exception. And it's not a logical exception corresponding to "no pk yet", but rather some uncaught exception thrown by some surprised Apple code.

Assigning Primary Keys

Usually EOF doesn't get around to generating/inserting primary keys until you actually saveChanges() on an EOEditingContext containing newly created about-to-be saved EOEnterpriseObjects.
But sometimes you want to assign a permanent pk as soon as you create the EOEnterpriseObject, not wait until it's actually committed to the db. Here's some code to do that, which, in a given situation, will generate the pk in the same way that EOF would have generated it later, but do it right away. [WO: Thanks to Pierre Barnard]

```
public void _insertObjectWithGlobalID(EOEnterpriseObject eo, EOGlobalID globalID)
 {
   EOEntity entity = EOUtilities.entityNamed(this, eo.entityName());
   EODatabaseContext dbContext = EOUtilities.databaseContextForModelNamed(this, entity.model().name());
   NSDictionary pkDict = primaryKeyDictionaryForDatabaseContextAndEntity(dbContext, entity);

   if (pkDict == null || pkDict.count() != 1)
   {
     NSLog.err.appendln("Failed to generate primary key for entity " + entity.name() + ", or pk is compound.");
   } else
   {
     Object pk = pkDict.allValues().lastObject();
     globalID = EOKeyGlobalID.globalIDWithEntityName(entity.name(), new Object[] { pk });
   }

   super._insertObjectWithGlobalID(eo, globalID);
 }

 public static NSDictionary primaryKeyDictionaryForDatabaseContextAndEntity(EODatabaseContext dbContext,
EOEntity entity) {
   NSDictionary pk = null;
   try {
     dbContext.lock();
     EOAdaptorChannel adaptorChannel = dbContext.availableChannel().adaptorChannel();
     if (!adaptorChannel.isOpen()) {
       adaptorChannel.openChannel();
     }
     pk = (NSDictionary) adaptorChannel.primaryKeysForNewRowsWithEntity(1, entity).lastObject();
   }
   catch (Exception e) {
     NSLog.err.appendln("Can't get primary keys for entity " + entity.name() + "  " + e);
   }
   finally {
     dbContext.unlock();
     return pk;
   }
 }
```

There are 2 insertObjectWithGlobalID methods in EOEditingContext. One has a name prefixed by an underscore. The regular one seems to call that one. For most uses overriding the regular one should be sufficient. With JavaClient however, the regular one is not called. On the server side only the one with the underscore is called. You'll have to override the method with the underscore in the server side context as the above will work only on the server.

If you use nested ECs in a standard WO application you might have to override the same semi-private method should you want the parent EC to handle PK creation.

BTW, I believe EOF does some optimization by retrieving PKs in batches rather than one by one. You lose that when using the above code.