EOF-Using EOF-The EOF Commandments

Some things to avoid when working with EOF. Some of these things are contraindicated in Apple documentation, others are not. But all are things that experience shows EOF does not expect, and can lead to all sorts of trouble, including mysterious exceptions, and EOF getting confused about what changes must be saved to the database.

1. ## Don't use an EO's Constructor.
   Don't set EO properties in the EO constructor - use `awakeFromInsertion(...)` or `awakeFromFetch(...)` instead.
   From Apple's Documentation:

   > *You may wonder why it's not recommended to initialize an enterprise object's values in an object's constructor. An enterprise object's constructor represents the earliest state in the life of a particular object. The state of an enterprise object at the point of construction is not complete; the object is not fully initialized. It may not yet have been inserted into an editing context and it might not even have been assigned a global ID. You don't ever want to manipulate an enterprise object that isn't in an editing context or that doesn't have a global ID.*
   >
   > *Therefore, any logic you add in an enterprise object's constructor may fail or be invalidated while the object finishes initializing. By providing custom logic in awakeFromInsertion or awakeFromFetch, however, you are guaranteed that an enterprise object is fully initialized.*

   When overriding `awakeFromInsertion(...)` there are two times that this method can be called when the EO being inserted is not actually being initialized for the first time.
   a. If it's being reinserted to a EC after it was deleted from another or the same EC. Only new EOs will have a temporary GlobalID.
   b. If it's being saved to the server-side application by a Java Client WebObjects application. These EOs will have been inserted on the client-side and likely already have values set. To avoid overriding these values, you should verify that the attribute's value is null first.

   ```
   @Override
   public void awakeFromInsertion(EOEditingContext ec) {
       super.awakeFromInsertion(ec);
       EOGlobalID gid = ec.globalIDForObject(this);
       if(gid.isTemporary()) {
           // only set default values when we're inserted into an EC for the first time.
           if(createdDate() == null) {
               // only set value if it hasn't already been set by a Java Client applciation.
               setCreatedDate(new NSTimestamp());
           }
       }
   }
   ```

2. ## Don't change the value of an attribute or behavior of a relationship in its accessor method
   EOF calls those accessor methods many times over the life of the EO. An accessor method is the last place that you want to be making **any** changes or doing any computations to determine the value. Don't:
   - **Don't** provide default values -
     Do use `awakeFromInsertion(...)` or `awakeFromFetch(...)`
   - **Don't** sort a to-many relationship's values -
     Do write a convenience method like `relatedObjectsSorted()`
   - **Don't** format a String or Date or Number attribute -
     Do formatting in helper classes or in the Component
   - **Don't** change an attribute's data type
   - **Don't** change an attribute's value when reading a different attribute. For example: Don't call `setFullName(...)` when you call `firstName()`
3. ## Always Insert First.
   Don't do anything to an EO before inserting it into an editing context. Always insert EOs into ECs immediately. See rule #1.
4. ## Don't modify any EO properties in `validateFor...(...)` methods
   Doing this in `validateValueForKey(...)` is ok as Chuck Hill noted in the list.
5. ## Always invoke `super` when overriding EOF methods
   If overriding `awakeFromInsertion(...)`, remember to call the superclass implementation. Same with `awakeFromFetch(...)`.
6. ## Don't use EOF in model class static initializers
   Doing so forces EOF into operational mode before the frameworks are initialized. Use lazy loading of the entity instead.
7. ## Don't use mutable classes as attributes

   Attributes of (`NSMutableArray`, `NSMutableDictionary` or any other class that can change internal state after creation) will confuse EOF. If you want this effect, use immutable classes and provide cover methods to replace the immutable instance with an updated instance. For example:

```
public void addAppointment(String time, String reason) {
    super.willChange();
    NSMutableDictionary mutableAppointments = appointments().mutableClone();
    mutableAppointments.setObjectForKey(reason, time);
    setAppointments(mutableAppointments.immutableClone());
}
```

8. ### Always Lock your EditingContext

   Don't call any EOEditingContext methods, or any methods on an EO in an EOEditingContext without first ensuring that the EOEditingContext is locked. That includes bindings in a WOD. It's your job to make sure the EO's that you bind to in a page are locked.

   Project Wonder

   Project Wonder provides auto-locking, so you don't have to worry about this.
9. ### Don't Expose Foreign Keys as Class Attributes
   Doing this will result in buggy unexpected behavior in EOF. Same goes for compound PKs (Primary Keys) where the compound PK values are set automatically by EOF (for example in a many-to-many join entity). However exposing compound PK attributes as class attributes is OK for an entity where you are manually setting the PK values yourself. Interestingly, even though it is not recommended, exposing a Primary Key as a class attribute will not break EOF.
10. ### Always use .immutableClone() when iterating over a to-many relationship with a for-each loop.
    It doesn't matter if you use the relationship directly or use a local variable that was populated by `myRelatedObjects()` or `getValueForKey("myRelatedObjects")` you should always use an imutableClone of the array, otherwise you will likely just have a pointer to the actual relationship, which can easily change.