EOF-Using EOF-EOGenerator

This is deprecated information!

Contents

Overview

If you've ever used EOModeler's Java source code generator, you know how much of a pain it can be when you make changes to your model objects and have to merge changes in later. One solution for this is to use EOGenerator, an application developed by Rubicode Software, which uses the Generation Gap pattern to create your Java files from your EOModels. EOGenerator produces TWO java files for each Entity rather than one. Take the example of a Person entity. The first java file is _Person.java, which contains all of the autogenerated methods. The second java file is Person.java, and Person extends _Person. The second file is where you place all of your customizations. Any time your model changes, only your _Xxx.java files are updated, and your customizations are left untouched. Additionally, EOGenerator allows for the creation of extensive custom templates for your files, which provides the ability to place convenience methods in your _Xxx.java files.

EOGenerator doesn't work on Mac OS X 10.5. You have to use JavaEOGenerator or Velocity EOGenerator.

Advantages

There are several advantages to using EOGenerator over EOModeler's default Java file generation and merging with FileMerge.

- EOGenerator uses the Generation Gap pattern, which provides a much cleaner separation of autogenerated vs customized code with no need to deal with merging at all. There are border cases with FileMerge that can cause you to deal with annoying conflicts.
- EOGenerator uses the MiscMerge language for its templates. This allows you to extend the core templates with extensive customizations (see the EOGenerator Mods section below), better supporting your own custom development process and workflow.
- As David LaBer put it, "all the cool kids use it - and we all know looking cool is the **most** important criteria".

How To Use It

Kieran Kelleher has writttten an Introduction to EOGenerator on his blog.

It's actually very simple to use. The quick start is:

- Download and untar EOGenerator from the Rubicode site
- Run the following command:

```
eogenerator -model /path/to/model/YourModel.eomodeld -destination /path/to/source/folder
-subclassDestination /path/to/source/folder -templatedir /path/to/EOGenerator/templates -java -packagedirs
```

Voila. EOGenerator will spit out your Java files for you. Let's break down the commands you can pass in:

- -define-EOGenericRecord <class>, allows you to specify the _Person class's superclass. For instance, if you use Project Wonder, you would specify -define-EOGenericRecord er.extensions.ERXGenericRecord
- -destination <path>, the folder that _Person.java-style java files will be produced in (the non-editable files)
- -java, produce java files
- -javaTemplate <filename>, the name of the Java template to use inside of the template dir (_Person)
- -model <path>, Passes in the path of a .eomodeld you would like to generate Java files for. You can actually include multiple -model commands on the commandline
- -packagedirs, produce package directory for any package statements defined in your Java files (not necessary if you don't specify package names on your entities. By the way, you should specify packages on your entities 🙂)
- -refmodel <path>, Passes in the path of an .eomodeld that is required for generating Java files, but that won't actually have Java files generated for it. For instance, you should -refmodel any prototypes, or any models in other frameworks that you depend on
- -subclassDestination <path>, the folder that Person.java-style java files will be produced in (the editable files)
- -subclassJavaTemplate <filename>, the name of the Java subclass template to use inside of the template dir (Person)
- -templatedir <path>, the path to the folder that contains EOGenerator templates
- -verbose, turn on verbose output

Custom EOGenerator Mods

Zak Burke

Allow setting nulls on a to-one relationship (and turn it into a remove). Note, this is also included in Jonathan Rentzsch's templates.

```
public void save<$ToOneRelationship.name.initialCapitalString$>(<$ToOneRelationship.destinationEntity.
referenceJavaClassName$> value)
  {
    if (value == null)
    {
      <$ToOneRelationship.destinationEntity.referenceJavaClassName$> object = <$ToOneRelationship.name$>();
      if (object != null)
        removeObjectFromBothSidesOfRelationshipWithKey(object, "<$ToOneRelationship.name$>");
    }
    else
    {
      addObjectToBothSidesOfRelationshipWithKey(value, "<$ToOneRelationship.name$>");
    }
  }
```

Chuck Hill

Return the list of changes between the current EO and the last committed version of the EO:

```
public NSDictionary changedProperties() {
    NSDictionary commitedValues = editingContext().committedSnapshotForObject(this);
    return changesFromSnapshot(commitedValues);
  }
```

Jonathan Rentzsch

Jonathan Rentzsch has provided his base EOGenerator templates, which are a must-have:

http://rentzsch.com/share/eogenerator52templates.zip

Markus Ruggiero

Constants for all attributes and relationships. This allows compile time error checking in situations like addObjecttoBothSidesOfRelationshipWithKey(myObject, Person.TO_MANY_Children)

```
<$foreach attribute classAttributes.@reversedArray do$>
    public static final String ATTRIBUTE_<$attribute.name$> = "<$attribute.name$>";<$endforeach do$>

  <$foreach ToOneRelationship classToOneRelationships.@reversedArray do$>
    public static final String TO_ONE_<$ToOneRelationship.name$> = "<$ToOneRelationship.name$>";<$endforeach
do$>

  <$foreach ToManyRelationship classToManyRelationships.@reversedArray do$>
    public static final String TO_MANY_<$ToManyRelationship.name$> = "<$ToManyRelationship.name$>";<$endforeach
do$>
```

We also make heavy use of the user info dictionary on entity and attribute level. Allows to generate customized methods and what not. One example is booleans that are stored in the DB as strings with values "true" and "false".

```
<$if attribute.userInfo.usage h1.  booleanFlag $>    // boolean accessors
    public void <$attribute.userInfo.setterName$>(boolean newBoolean) {
      set<$attribute.name.initialCapitalString$>(newBoolean ? "true" : "false");
    }

    public boolean <$attribute.userInfo.getterName$>() {
      return "true".equals(<$attribute.name$>()) ? true : false;
    }

    // validation
    public String validate<$attribute.name.initialCapitalString$>(String newValue) {
      if ( newValue  null ) {
        return "false";
      } else if ( !newValue.equals("true") && !newValue.equals("false") ) {
        String errorMessage = MessageHandler.format("INVALID_BOOLEAN_FLAG <$classNameWithoutPackage$>.
<$attribute.name$>", null);
        throw new NSValidation.ValidationException(errorMessage);
      }
      return newValue;
    }
  <$endif$>
```

Mike Schrag

Add a constant that represents the name of the entity so that you can refer to Person.ENTITY_NAME in fetches rather than the String (allows refactoring support in Eclipse):

```
public static final String ENTITY_NAME = "<$name$>";
```

Add a static factory method to your EO's (Person createPerson(...)) that shows you what required attributes and relationships are configured for you entity (attempts to provide a replacement "constructor" since EO constructors are empty):

```
public static <$classNameWithoutPackage$> create<$classNameWithoutPackage$>(EOEditingContext
_editingContext<$foreach Attribute classAttributes.@sortedNameArray do$><$if !Attribute.allowsNull$>,
<$Attribute.javaValueClassName$> _<$Attribute.name$><$endif$><$endforeach do$><$foreach ToOneRelationship
classToOneRelationships.@sortedNameArray do$><$if ToOneRelationship.isMandatory$>, <$ToOneRelationship.
destinationEntity.referenceJavaClassName$> _<$ToOneRelationship.name$><$endif$><$endforeach do$>) {
   <$classNameWithoutPackage$> eoObject = (<$classNameWithoutPackage$>)EOUtilities.createAndInsertInstance
(_editingContext, <$GEN_PREFIX$><$classNameWithoutPackage$>.ENTITY_NAME);<$foreach Attribute classAttributes.
@sortedNameArray do$><$if !Attribute.allowsNull$>
   eoObject.set<$Attribute.name.initialCapitalString$>(_<$Attribute.name$>);<$endif$><$endforeach do$><$foreach
ToOneRelationship classToOneRelationships.@sortedNameArray do$><$if ToOneRelationship.isMandatory$>
   eoObject.set<$ToOneRelationship.name.initialCapitalString$>Relationship(_<$ToOneRelationship.name$>);
<$endif$><$endforeach do$>
   return eoObject;
}
```

Here's a little bitty fancier (read: nastier) version that also handles superclass mandatory attributes and fields (one level). It skips any attribute that is referenced in the restricting qualifier of your subclass (since you are probably going to set that in your awakeFromInsertion):

```
public static <$classNameWithoutPackage$> create<$classNameWithoutPackage$>(EOEditingContext
editingContext<$foreach Attribute classAttributes.@sortedNameArray do$><$if !Attribute.allowsNull$>,
<$Attribute.javaValueClassName$> <$Attribute.name$><$endif$><$endforeach do$><$foreach Attribute parentEntity.
classAttributes.@sortedNameArray do$><$if !Attribute.allowsNull$><$set RestrictingQualifierKey =
false$><$foreach QualifierKey restrictingQualifier.allQualifierKeys do$><$if Attribute.name =
QualifierKey$><$set RestrictingQualifierKey = true$><$endif$><$endforeach do$><$if RestrictingQualifierKey =
false$>, <$Attribute.javaValueClassName$> <$Attribute.name$><$endif$><$endif$><$endforeach do$><$foreach
ToOneRelationship classToOneRelationships.@sortedNameArray do$><$if ToOneRelationship.isMandatory$>,
<$ToOneRelationship.destinationEntity.referenceJavaClassName$> <$ToOneRelationship.name$><$endif$><$endforeach
do$><$foreach ToOneRelationship parentEntity.classToOneRelationships.@sortedNameArray do$><$if
ToOneRelationship.isMandatory$>, <$ToOneRelationship.destinationEntity.referenceJavaClassName$>
<$ToOneRelationship.name$><$endif$><$endforeach do$>) {
   <$classNameWithoutPackage$> eoObject = (<$classNameWithoutPackage$>)EOUtilities.createAndInsertInstance
(editingContext, <$GEN_PREFIX$><$classNameWithoutPackage$>.ENTITY_NAME);<$foreach Attribute classAttributes.
@sortedNameArray do$><$if !Attribute.allowsNull$>
   eoObject.set<$Attribute.name.initialCapitalString$>(<$Attribute.name$>);<$endif$><$endforeach do$><$foreach
ToOneRelationship classToOneRelationships.@sortedNameArray do$><$if ToOneRelationship.isMandatory$>
   eoObject.set<$ToOneRelationship.name.initialCapitalString$>Relationship(<$ToOneRelationship.name$>);
<$endif$><$endforeach do$><$foreach Attribute parentEntity.classAttributes.@sortedNameArray do$><$if !Attribute.
allowsNull$><$set RestrictingQualifierKey = false$><$foreach QualifierKey restrictingQualifier.allQualifierKeys
do$><$if Attribute.name = QualifierKey$><$set RestrictingQualifierKey = true$><$endif$><$endforeach do$><$if
RestrictingQualifierKey = false$>
   eoObject.set<$Attribute.name.initialCapitalString$>(<$Attribute.name$>);<$endif$><$endif$><$endforeach
do$><$foreach ToOneRelationship parentEntity.classToOneRelationships.@sortedNameArray do$><$if
ToOneRelationship.isMandatory$>
   eoObject.set<$ToOneRelationship.name.initialCapitalString$>Relationship(<$ToOneRelationship.name$>);
<$endif$><$endforeach do$>
   return eoObject;
}
```

Add a bunch of convience fetch methods (fetchAllPersons, fetchRequiredPerson, and other variants). It's not smart about pluralization, so it's just going to put an "s" on the end of the entity name:

```
public static NSArray fetchAll<$classNameWithoutPackage$>s(EOEditingContext _editingContext) {
    return <$GEN_PREFIX$><$classNameWithoutPackage$>.fetchAll<$classNameWithoutPackage$>s(_editingContext,
null);
  }
```

```
public static NSArray fetchAll<$classNameWithoutPackage$>s(EOEditingContext _editingContext, NSArray
_sortOrderings) {
    return <$GEN_PREFIX$><$classNameWithoutPackage$>.fetch<$classNameWithoutPackage$>s(_editingContext, null,
_sortOrderings);
  }
```

```
public static NSArray fetch<$classNameWithoutPackage$>s(EOEditingContext _editingContext, EOQualifier
_qualifier, NSArray _sortOrderings) {
    EOFetchSpecification fetchSpec = new EOFetchSpecification(<$GEN_PREFIX$><$classNameWithoutPackage$>.
ENTITY_NAME, _qualifier, _sortOrderings);
    fetchSpec.setIsDeep(true);
    NSArray eoObjects = _editingContext.objectsWithFetchSpecification(fetchSpec);
    return eoObjects;
  }
```

```
public static <$classNameWithoutPackage$> fetch<$classNameWithoutPackage$>(EOEditingContext _editingContext,
String _keyName, Object _value) {
    return <$GEN_PREFIX$><$classNameWithoutPackage$>.fetch<$classNameWithoutPackage$>(_editingContext, new
EOKeyValueQualifier(_keyName, EOQualifier.QualifierOperatorEqual, _value));
  }
```

```
public static <$classNameWithoutPackage$> fetch<$classNameWithoutPackage$>(EOEditingContext _editingContext,
EOQualifier _qualifier) {
    NSArray eoObjects = <$GEN_PREFIX$><$classNameWithoutPackage$>.fetch<$classNameWithoutPackage$>s
(_editingContext, _qualifier, null);
    <$classNameWithoutPackage$> eoObject;
    int count = eoObjects.count();
    if (count == 0) {
      eoObject = null;
    }
    else if (count == 1) {
      eoObject = (<$classNameWithoutPackage$>)eoObjects.objectAtIndex(0);
    }
    else {
      throw new IllegalStateException("There was more than one <$classNameWithoutPackage$> that matched the
qualifier '" + _qualifier + "'.");
    }
    return eoObject;
  }
```

```
public static <$classNameWithoutPackage$> fetchRequired<$classNameWithoutPackage$>(EOEditingContext
_editingContext, String _keyName, Object _value) {
    return <$GEN_PREFIX$><$classNameWithoutPackage$>.fetchRequired<$classNameWithoutPackage$>(_editingContext,
new EOKeyValueQualifier(_keyName, EOQualifier.QualifierOperatorEqual, _value));
  }
```

```
public static <$classNameWithoutPackage$> fetchRequired<$classNameWithoutPackage$>(EOEditingContext
_editingContext, EOQualifier _qualifier) {
    <$classNameWithoutPackage$> eoObject = <$GEN_PREFIX$><$classNameWithoutPackage$>.
fetch<$classNameWithoutPackage$>(_editingContext, _qualifier);
    if (eoObject ==  null) {
      throw new NoSuchElementException("There was no <$classNameWithoutPackage$> that matched the qualifier '"
+ _qualifier + "'.");
    }
    return eoObject;
  }
```

Add methods for getting local instances of EO's. The static one is handy if you have a reference to an EO that might be null (it does a null check first):

```
public <$classNameWithoutPackage$> localInstanceOf<$classNameWithoutPackage$>(EOEditingContext _editingContext)
{
    return (<$classNameWithoutPackage$>)EOUtilities.localInstanceOfObject(_editingContext, this);
  }
```

```
public static <$classNameWithoutPackage$> localInstanceOf<$classNameWithoutPackage$>(EOEditingContext
_editingContext, <$classNameWithoutPackage$> _eo) {
    return (_eo == null) ? null : (<$classNameWithoutPackage$>)EOUtilities.localInstanceOfObject
(_editingContext, _eo);
  }
```

If you've ever wanted to be able to qualify a toMany relationship on your EO's, but sometimes you want to fetch them w/ a fetch spec, and sometimes you want to filter in-memory, you can use the following:

```
<$if !ToManyRelationship.inverseRelationship$>
          public NSArray <$ToManyRelationship.name$>(EOQualifier qualifier) {
                  return <$ToManyRelationship.name$>(qualifier, null);
          }
  <$endif$>
  <$if ToManyRelationship.inverseRelationship$>
          public NSArray <$ToManyRelationship.name$>(EOQualifier qualifier) {
                  return <$ToManyRelationship.name$>(qualifier, null, false);
          }

          public NSArray <$ToManyRelationship.name$>(EOQualifier qualifier, boolean fetch) {
                  return <$ToManyRelationship.name$>(qualifier, null, fetch);
          }
  <$endif$>

          public NSArray <$ToManyRelationship.name$>(EOQualifier qualifier, NSArray sortOrderings<$if
ToManyRelationship.inverseRelationship$>, boolean fetch<$endif$>) {
                  NSArray results;
  <$if ToManyRelationship.inverseRelationship$>
                  if (fetch) {
                          EOQualifier fullQualifier;
                          EOQualifier inverseQualifier = new EOKeyValueQualifier(<$ToManyRelationship.
destination.className$>.
                 <$ToManyRelationship.inverseRelationship.name.uppercaseUnderbarString$>_KEY, EOQualifier.
QualifierOperatorEqual, this);
                          if (qualifier == null) {
                                  fullQualifier = inverseQualifier;
                          }
                          else {
                                  NSMutableArray qualifiers = new NSMutableArray();
                                  qualifiers.addObject(qualifier);
                                  qualifiers.addObject(inverseQualifier);
                                  fullQualifier = new EOAndQualifier(qualifiers);
                          }
                          results = <$ToManyRelationship.destination.className$>.fetch<$ToManyRelationship.
destination.name$>s(editingContext(), fullQualifier, sortOrderings);
                  }
                  else {
  <$endif$>
                          results = <$ToManyRelationship.name$>();
                          if (qualifier != null) {
                                  results = EOQualifier.filteredArrayWithQualifier(results, qualifier);
                          }
                          if (sortOrderings != null) {
                                  results = EOSortOrdering.sortedArrayUsingKeyOrderArray(results,
sortOrderings);
                          }
  <$if ToManyRelationship.inverseRelationship$>
                  }
  <$endif$>
                  return results;
          }
```

John Huss

I wanted to share a wonderful bit of knowledge I learned today. If you're using Java 1.5 you can add @SuppressWarnings("all") to the template for your _EO base classes and eliminate annoying compiler messages (usually uneeded import statements).

```
@SuppressWarnings("all")
  public class _Invoice extends ERXGenericRecord {
  }
```

Guido Neitzer

Create awakeFromInsertion() and awakeFromFetch() in your EOGenerator template as a method stub that only calls super() and has a comment for "initialize your object here ...". You only have to put in the code at that place and can't possibly forget to call super(). Here is an example:

```
/**
 * Initialization of the instance while inserting it into an editing context
 */
public void awakeFromInsertion (EOEditingContext editingContext) {
    super.awakeFromInsertion (editingContext);
    // initialize your object here
}
```

This is from my JavaSubclassSourceTemplate.eotemplate