ERTaggable

This is an email that Mike Schrag have sent to the Wonder mailing list in February 2008.

Note that I just finished this up and haven't deployed it into my apps yet, so it's not exactly .... production tested .... all that much 🙂 But the examples that I was running appeared to be working properly. Let me know if you get explosions. This port required quite a few additions to ERXSQLHelper to support generation of group by clauses's and having clauses on EOSQLExpressions. I've tested this on FrontBase so far, but that's it. It may require db-specific implementations ...

ERTaggable

ERTaggable is a fairly direct port of the acts_as_taggable Rails mixin to EOF. The framework provides a very easy method of integrating tagging support into arbitrary entities in your applications.

Quick Start

Lets take the example where your model "MyModel" has a "Person" entity that you want to support tagging on.

Create a migration using the helper superclass:

```
public class MyModel0 extends ERTaggableEntity0 {
  public MyModel0() {
   super(Person.ENTITY_NAME);
  }
}
```

Register the taggable in your Application constructor or framework principal:

```
ERTaggableEntity.registerTaggable(Person.ENTITY_NAME);
```

Add convenience methods onto your Person class (optional, but handy):

```
public ERTaggable<Person> taggable() {
 return ERTaggable.taggable(this);
}

public static ERTaggableEntity<Person> taggableEntity() {
 return ERTaggableEntity.taggableEntity(Person.ENTITY_NAME);
}
```

Tag like the wind:

```
Person mike = Person.createPerson(editingContext, "Mike Schrag");
ERTaggable taggableMike = mike.taggable();
taggableMike.addTags("mike employee important");

NSArray matchingPeople =
Person.taggableEntity().fetchTaggedWith("employee", editingContext);

NSDictionary<EOEntity, NSArray<? extends ERXGenericRecord>> matchingItems =
ERTaggableEntity.fetchAllTaggedWith("mike", editingContext);
```

Nitty Gritty

If you choose to deviate from the "magical" route described above, you can provide several override hooks:

By default all entities share the same corpus of tag names. For any entity, you can register a custom ERTag subclass that stores in a separate table instead. This is generally only an issue if you need to provide tag completion and you want to restrict the set of completion offerings. Note that hooking to a custom ERTag table also requires that you use the optional constructor on the migration superclass to specify the name of that entity.

By default, tags are normalized by trimming and lowercasing them. You can override the tag normalizer that is used by setting it on your ERTaggableEntity. An example implementation is

```
public class TagNormalizer implements ERTagNormalizer {
  public String normalize(String tag) {

    String normalizedTag = tag;
    if (normalizedTag != null) {
      normalizedTag = normalizedTag.trim();
    }
    return normalizedTag;
  }
}
```

You can then register that normalizer wherever you have an instance method such as

```
taggableMike = mike.taggable();
TagNormalizer myNormalizer = new TagNormalizer();
taggableMike.taggableEntity().setNormalizer(myNormalizer);
```

You MUST register your taggable entities prior to attempting any tagging operations. The framework will throw an exception scolding you if you do not.
By default, your EOModels are modified on-the-fly to inject tagging support into them. If you don't like magic, you can instead manually create the join entity between your entity and the ERTag entity. You MUST also create a flattened to-many relationship from your entity to the ERTag table through your join entity. If you name that relationship anything other than "tags" (or you use a custom ERTag entity), you must specify the relationship name when you register the entity.
Tags are unique and shared. To ensure this, ERTag commits new tag names in a separate transaction. This can lead to a potentially undesirable (yet mostly harmless) side effect where new tag names may be committed even though you roll back your editing context. Only tag names have this behavior, not tag relationships on your entities.

If you don't want to use the migration, you need to create a join table that contains a "your entity id" (named item_[WONDER:your pk column name] for each pk attribute) and an ERTagID foreign key (named tag_id). You can name the columns whatever you want if you also manually create the join entity in your EOModel.