How To ... ?

Add properties to an already existing Replicable entity

You have 2 posibilities.

- Make use of the `public Map<String, Object> getMongoProperties(String domain)`. Use this method if the properties you add are not common to multiple entities. If this is the case ...
- Add the properties in a DataSet. You can create a separate one or make use of an already existing one.
  Examples are all over the place. For instance, take a look at *OpenAcdAgent*. It has a bunch of "proprietary" properties and also some common with other OpenACD objects, written in the OpenACD DataSet.

Make an entity Replicable

- you must make sure the entity implements *Replicable* interface. Implement all methods with relevant info.
- make sure the replication is triggered when saving and deleting. One thing to take into account is that the replication is not triggered if the save method is called something other than **saveObject(Object)** (replace Object with any name), because only the methods with that name are intercepted. (Similar with delete.) Another option is to publish an event, but the recommended way is the interceptor way.
- make sure the entities are retrieved by an *ReplicableProvider*. The recommended way to do that is to create a separate ReplicableProvider implementation to implement the *getReplicables* method.

Mongo CLI

There are a bunch of useful commands to use from Mongo CLI (accessible by typing *mongo* in your favourite terminal).

- **`show dbs`**
  prints out all databases.
- **`show collections`**
  prints out all collections in a database (you'll need to `use a_particular_database` first).
- **`db.entity.find()`**
  this will print out all documents in entity collection. Not very useful if there are many as it prints out just a few (you'll notice "has more" at the end).
- **`db.entity.find({_id : /User.*/})`**
  db.entity.find({_id: { $regex: 'User.*', $options: 'i' }})
  regular expression finder. Use your imagination to narrow your search.
  (first example may not work on CLI 2.0.7)
- **`db.entity.find({ent:"user"})`**
  instead of regex you may find all entities of a particular type using **ent** field. This field is the java class name in lower case.
- **db.entity.find({type:"openacdskill", name:"German"})**
  narrow the search by applying multiple "filters".
- **`db.entity.drop()`**
  drop the entire entity collection.
- **`db.entity.find({uid:"200"})`**
  find a particular user document. User names are held in the *uid* field.
- **`db.entity.find({uid:"200"}).forEach(function (d) {printjson(d)})`**
  prints out a formatted version of the document. Pretty useful when looking at a single document. You can also use: `db.entity.find({uid:"200"}).forEach(function (d) {printjson(d.prm)})` to see a formatted version of the **prm** field (permissions).
- **`db.entity.find({uid:"200"}).forEach(function (d) {print("_id: "+ d._id +"; perms: "+d.prm)});`**
  print out just a few fields from a particular document

More: http://docs.mongodb.org/manual/reference/operators/