Article
Jose-Tomas Salvador · Aug 14, 2019 5m read

Open Exchange

Playing with Object Synchronization

Object Synchronization is a feature that has been around for a while, since Caché days, but I wanted to explore a bit more how it works. I've always thought that database automatic synchronization is complex by nature but, for some particular scenarios shouldn't be so hard. So I considered a very simple use case (OK, perhaps the typical one, I'm not discovering anything... but if it's common and it works, it's good). You can download from GitHub and compile it into your system, generate sample data and play a bit with it. It's done for ***InterSystems IRIS*** but it also should work in last versions of *Caché & Ensemble*.

Now, let's go into the details. My sample app, at class level (no UI implemented sorry), is considering a small data model where we have Items, Customers, Employees and Orders. Working off-line, if we're a sales person, we'll normally just limit our operations to query for: *Items*, *Customers* and or *Employess* (so that data could be considered read-only for us); and to add or update *Orders,* which is our read-write data that we want to synchronize. Then, registering or modifying new *Items*, *Customers* or *Employees* can be considered something to be done in the MASTER database system and spread it over the CLIENT databases.

When defining the class model, we have to take into account what transactions we want to synchronize (in our case *Orders*) and which ones don't. In the class definition we should add Parameter OBJJOURNAL = 1; so to indicate that the system has to keep track of insert/deletion/changes of objects belonging to that particular class. Also, in separate systems, objects creation will happen at different moments and, if we use default IDKEYs, will have situations in which that same object (after syncrhonization) has a different IDKEY in each database... how can we deal with that?... We need a way to globally and uniquely identify an object across the databases... here is when GUID comes to our help. IRIS can generate a Globally Unique ID for an object that can be shared between IRIS systems and all of them can use it to decide if they're dealing with the same object or not. So far so good. How do we get a class to automatically generate a GUID for each new object?. Easy. We just have to incude Parameter GUIDENABLED = 1; in its definition. And with this, all our work is done.

In our use case we'll have something like (full definition in GitHub...here is truncated):

```
Class SampleApps.Synch.Data.Order Extends (%Persistent,%Populate,%JSON.Adaptor)

{
   /// We don't want to force other classes reference to use %JSON.Adaptor. Then we
decide export the
   /// GUID of referenced objects (by default it will include the JSON of object
referenced)
   Parameter %JSONREFERENCE As STRING [ Constraint = "OBJECT,ID,OID,GUID", Flags =
ENUM ] = "GUID";

   ///We'll keep track of this class for synchronization (assuming that in off-line
mode is not
   ///allowed by the app to modify master classes

   Parameter OBJJOURNAL = 1;
   Parameter GUIDENABLED = 1;
   ...
}
```

```
Class SampleApps.Synch.Data.Customer Extends (%Persistent, %Populate)
{

    ///This allows the class to be stored with the GUIDs
    ///Also is a class referenced in Order class which is to be synchronized,
    ///so this class objects need a GUID to be able to synchronize their references
    Parameter GUIDENABLED = 1;
    ...
}
Class SampleApps.Synch.Data.Item Extends (%Persistent, %Populate)
{

    Parameter GUIDENABLED = 1;
    ...
}

Class SampleApps.Synch.Data.Employee Extends (%Persistent,%Populate)
{

    Parameter GUIDENABLED = 1;
    ...

}
```

As you can see, there is one class: Order; with OBJJOURNAL and GUIDENABLED, and the other classes just have GUIDENABLED. Why? Because the objects of those classes are referenced within each *Order* and, even though we don't keep track of modifications, we need to be sure that the same *Order* references to the same objects no matter in which systems we are looking at *(IMPORTANT: if we don't enable GUID in objects referenced, the Orders will be synchronized but without any info related to references).*

OK. This said, you can just play with it. I've implemented some utils in SampleApps.Synch.Util to populate data and execute basic insert/update/delete operations. Also a basic API Rest (SampleApps.Synch.API.v1.RestLegacy) so to be able to do it from *ObjectScript* or from a *REST Client.*

More information together with all the project code is available in GitHub here. Basically you can clone the repo or download a zip with the code. Once you get the code in your server, load and compile the class SampleApps.Synch.Config.Installer from your terminal (within the namespace that you choose), an run the installation:

```
TEST>do $system.OBJ.Load
("c:/Temp/SampleApps/Synch/Config/Installer.cls","ck")
TEST>set args("InstallingFromNS")="TEST"
TEST>set args("NamespaceMaster")="CORE"
TEST>set args("NamespaceClient")="NODE"
TEST>do ##class(SampleApps.Synch.Config.Installer).setup(.args)
```

Most of arguments have default values (take a look at *Installer* class, but, as you can see in the sample above, you can change most of them before calling the *setup* method.

Once executed, you'll have two namespaces: *CORE* and *NODE* (*MASTER* and *CLIENT* by default if you call setup without arguments) with populated Orders, Items, Customers and Employees. Both databases are identical at this point. You can play with REST services to populate with more data, prepare a synchronization data from one namespace and then synchronizing it in the other. Whatever conflict discovered during synchronization will be stored in SampleApps.Synch.Data.SynchConflict that also can be queried through a REST service (apart from

SQL).

In GitHub I've included more info and also examples to call REST services so you can start testing easily. Just click here, download the source code or clone the repository and you'll be testing in 2 minutes.

Enjoy!!

#Object Data Model #ObjectScript #REST API #Caché #Ensemble #InterSystems IRIS
Check the related application on InterSystems Open Exchange