Artículo
Jose-Tomas Salvador · 14 ago, 2019 Lectura de 5 min

Open Exchange

Jugando con la Sincronización de Objetos

La funcionalidad de Sincronización de Objetos no es nueva, estaba presente en Caché, pero quería explorar un poco más en profundidad cómo funciona. Siempre he pensado que la sincronización automática de una base de datos es compleja en sí misma pero, para algunos escenarios muy particulares quizá no sea tan difícil. Así que he considerado un caso de uso muy simple (OK, quizá el caso típico, no descubro nada... pero si es común y funciona, es bueno). Puedes bajar el código de GitHub y compilarlo en tu sistema, generar automáticamente datos de ejemplo y jugar un poco con ello. Está hecho para ***InterSystems IRIS*** pero debería funcionar también en las últimas versiones de *Caché y Ensemble*.

Ahora vayamos al detalle. Mi aplicación de ejemplo, a nivel de clases (lo siento, no hay interfaz de usuario), considera un pequeño modelo de datos donde tenemos Items, Customers, Employees y Orders. Trabajando *off-line*, si somos comerciales, normalmente limitaremos nuestras operaciones a consultar: *Items*, *Customers* y/o *Employess* (por lo que esos datos los consideramos de solo-lectura); y a añadir o modificar *Orders*. El registro o modificación de *Items*, *Customers* o *Employees* lo podemos considerar como algo que debe hacerse en el sistema de base de datos MASTER y de ahí transmitido a los sistemas de base de datos CLIENT .

Cuando definimos el modelo de clases, tenemos que tener en cuenta que transacciones queremos sincronizar (en nuestro caso *Orders*) y cuales no. En la definición de las clases debemos añadir Parameter OBJJOURNAL = 1; para indicar que el sistema tiene que mantener la relación de los objetos de dicha clase insertados/borrados/modificados. Igualmente, en el/los sistema(s) cliente, la creación de objetos sucederá en momentos diferentes y, si usamos IDKEYs por defecto, tendremos situaciones en las cuales el mismo objeto (tras la sincronización) tendrá un IDKEY diferente en cada base de datos... Como gestionamos eso? ... Necesitamos una forma de identificar global y univocamente un objeto entre bases de datos... y aquí es cuando el GUID viene en nuestra ayuda. InterSystems IRIS puede generar un ID global unico (*Globally Unique ID*) para un objeto que puede ser compartido entre sistemas IRIS y todos ellos pueden usarlo para decidir si están tratando o no con el mismo objeto. Hasta aquí bien. Para que una clase pueda generar automáticamente un GUID para cada nuevo objeto, tenemos que incluir Parameter GUIDENABLED = 1; en su definición. Y con esto, todo nuestro trabajo está hecho.

En nuestro caso de uso tendremos algo similar a esto (la definición completa en GitHub...aquí recortada):

```
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;
    ...

}
```

Como ves, hay una clase: Order; con OBJJOURNAL y GUIDENABLED, y las otras clases sólo tienen GUIDENABLED. ¿Por qué? Porque los objetos de esas clases son referenciadas dentro de cada *Order* y, aunque no mantenemos sus modificaciones, necesitamos saber que la misma *Order* referencia a los mismos objetos sin importar en que sistema(s) estemos mirando *(IMPORTANTE: si no habilitamos GUID en los objetos referenciados, las Orders serán sincronizadas pero sin ninguna información relacionada a las referencias)*.

OK. Dicho esto, puedes empezar a jugar. He implementado algunas utilidades en SampleApps.Synch.Util para generar datos y ejecutar operaciones básicas de interción/modificación/borrado. También encontrarás una API REST básica (SampleApps.Synch.API.v1.RestLegacy) para poder hacerlo desde un *cliente REST* o desde *ObjectScript* indiferentemente.

Dispones de más información junto con todo el resto del código del proyecto aqui en GitHub. Puedes clonar el repositorio o descargar un zip con el código fuente. Una vez tengas el código en tu servidor, carga y compila la clase SampleApps.Synch.Config.Installer desde tu terminal (dentro del *namespace* que elijas), y ejecuta la instalación:

```
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)
```

La mayoría de los argumentos tienen valores por defecto (echa un vistazo a la clase SampleApps.Synch.Config.Installer, pero, como puedes ver en el ejemplo arriba, puedes cambiarlos antes de llamar al método *setup*).

Tras su ejecución, tendrás 2 nuevos *namespaces*: *CORE* y *NODE* (*MASTER* y *CLIENT* por defecto si llamas a *setup* sin argumentos) con datos ya precargados en *Orders, Items, Customers* y *Employees*. Ambas bases de datos son idénticas en este punto. Puedes jugar con los servicios REST para cargar nuevos datos, preparar la sincronización de datos de un *namespace* y después sincronizarlos en el otro. Cualquier conflicto descubierto durante la sincronización será almacenado en SampleApps.Synch.Data.SynchConflict que también puede consultarse a través de un servicio REST (aparte de desde SQL).

En GitHub he incluido más información y también ejemplos para llamar a servicios REST para que así puedas empezar a probar facilmente. Simplemente haz click aquí, descarga el código fuente o clone el repositorio y podrás estar probando en 2 minutos.

Disfruta!!

#Modelo de datos de objetos #ObjectScript #Caché #Ensemble #InterSystems IRIS
Ir a la aplicación en InterSystems Open Exchange

URL de fuente:https://es.community.intersystems.com/post/jugando-con-la-sincronizaci%C3%B3n-de-objetos