Building ApiCode Program On DBSync

Deploying

DBSync is an integration platform used to integrate databases, Software as a Service (SaaS) and, other software applications as it sees fit.

For running your ApiCode in the DBSync environment, you need to have an On-Premise instance of DBSync installed in your box.

For more details about downloading, and configuring, DBSync, please refer to our wiki documentation to get an Introduction to DBSync. Do take a look into the Getting Started section. There, register for the trial installation and do a quick installation/configuration of the same.

http://help.mydbsync.com/docs/display/dbsync/1+-+Introduction

http://help.mydbsync.com/docs/display/dbsync/2+-+Getting+Started

Deploying ApiCode on DBSync platform

Before we can deploy on the DBSync server, we need to understand the structure of appcode-deploy.xml. This file is under "src/deploy" folder. Open the appcode-deploy.xml and you can view the parameters there.

1. Log into AppCenter and Launch DBSync2.
2. Add the Java and DB connector .
3. Before we configure the adapters, it is important that we deploy our custom 'HelloWorld' AppCode to the DBSync2 environment. So that, when we configure the Java Adapter, it has the relevant class files packaged. As mentioned, let us edit appcode-deploy.xml to point to the correct profile name:

```
<property name="profile" value="HelloProfile" />
```

5. Now we need to run the ant script for appcode-deploy.xmlwith target. As deploy, we will see the output as follows:

6. The log shows a successful deployment. Now let us configure properties of Java Connector. In our case, the only property that we need to set is the environment name setting in the Classpath property.
As a result, we need to set it to "sandbox".
7. Similarly for DB adapter, set the connection and DB properties to be used while sync. Let us go ahead and configure our use-case. We need to configure the DB adapter to be able to read from the DB table and write it out in our 'Hello World' program. Before we can use the connectors in our project, we need to validate them.
8. Now define and save the reader (DB reader) properties by using the metadata exposed through the Lookup button. Let us define the sql query that we will be using to fetch data from the selected table. In this case, we use the following query: "select * from test_customer".

Note :
In this example, we are using the AppCode as a writer and not a reader. However, as explained in the previous sections, we can use the AppCode as a source as well from where we do a ***read*** *operation. In that case, we need to pass a SELECT statement as an input query. This would then be converted into a filter. The filter params are passed in the where condition as key='value' comma separated. The structure of the same is given below:*
Select * from <tablename> where []SaveResult[] <methodName>(List<InputMetadataParameter>).

We set the batchSize to 100 and the table name to "writeHello". This will we defined in our HelloWorld AppCode in this example.
After selecting the table, and saving the properties, we need to map the reader and the writer properties.
For that, open properties of the Map box and then, set them. We need to define the map sequence and the description.
In this case, we set Map Sequence as 1 and the description as "Test".

Let us edit the Map where we map out the Reader metadata elements to that of the Writer.
We need to select the fields from the Source and Target. Then, we map them out from the Map Screen. To get details about the whole process of Mapping, you can go through WIKI documentation in the DBSync section. Now as per the mapping, we have mapped the **"id"** field in the Writer to the **"number"** field in the Reader. In addition to that, we mapped the **"name"** field of the Writer to the **"name"** field of the "Reader.
After "closing" the Map window, we go back to the properties window of the "Map" state and "Save" it. Once "Saved", we need to run the Sync and view the results. For that, click on the "Run" button.
However, before you click on 'Run', we need to ensure the LogLevel is SET properly.

Click on the "Processes" tab to view the logs. Then, click on the individual log file in the list to view - the file is in the LogViewer.

On Successful run, we can view the output.

The "Writer" output in the console looks something like this:

```
Received input parameter:id:C02710:name:Name07_Updated
Writing value:Hello:Name07_Updated:a93d1c72-4b4b-43aa-af42-5c9d27629b0e
Received input parameter:id:C02720:name:Name08_Updated
Writing value:Hello:Name08_Updated:a93d1c72-4b4b-43aa-af42-5c9d27629b0e
Received input parameter:id:C02730:name:Cronus Cardoxy Sales
Writing value:Hello:Cronus Cardoxy
Sales:a93d1c72-4b4b-43aa-af42-5c9d27629b0e
Received input parameter:id:C02740:name:Cronus Cardoxy Procurement
Writing value:Hello:Cronus Cardoxy
Procurement:a93d1c72-4b4b-43aa-af42-5c9d27629b0e
Received input parameter:id:IC1020:name:Cronus Cardoxy Sales
Writing value:Hello:Cronus Cardoxy
Sales:a93d1c72-4b4b-43aa-af42-5c9d27629b0e
Received input parameter:id:IC1030:name:Cronus Cardoxy Procurement
Writing value:Hello:Cronus Cardoxy
Procurement:a93d1c72-4b4b-43aa-af42-5c9d27629b0e
Hello World AppCode close called:a93d1c72-4b4b-43aa-af42-5c9d27629b0e
```

The “Reader” output in the console looks something like this:

```
<row db__id="65">
  <id type="string">C02580</id>
  <name>London Candoxy Storage Campus</name>
  <value />
</row><row db__id="66">
  <id type="string">C02590</id>
  <name>Elkhorn Airport</name>
  <value />
</row><row db__id="67">
  <id type="string">C02600</id>
  <name>Candoxy Canada Inc.</name>
  <value />
</row><row db__id="68">
  <id type="string">C02610</id>
  <name>New Concepts Furniture</name>
  <value />
</row>
```

If you look closely at the output, the DBSync processor queried the table data using the DB Adapter and wrote it out in our "HelloWorld" AppCode "writer" output.

This document now contains a complete step-by-step procedure. Name, (1) create custom AppCode; deploy it on DBSync; and (3) run the sync.