Article
Brendan Bannon · Aug 29, 2016 7m read

The Art of Mapping Globals to Classes 1 of 3

The Art of Mapping Globals to Classes 1 of 3

Looking to breathe new life into an old MUMPS application? Follow these steps to map your existing globals to classes and expose all that beautiful data to Objects and SQL.

By following the simple steps in this article and the next two you will be able to map all but the craziest globals to Caché classes. For the crazy ones I will put up a zip file of different mappings I have collected over the years. This is NOT for new data; if you don ' t already have existing global please just use the default storage.

If you still can ' t make heads or tails of your globals, send some example data to Support@InterSystems.com and we will be happy to help you figure it out.

Steps for mapping a global to a class:

1. Identify a repeating pattern in the global data.
2. Identify what makes up a unique key.
3. Identify the properties and their types.
4. Define the properties in the class (don ' t forget the properties from the variable subscripts).
5. Defined the IdKey index.
6. Define the Storage Definition:
    a. Define the Subscripts up to and including the IdKey.
    b. Define the Data section.
    c. Ignore the Row ID section. 99% of the time the default is what you want so let the system fill that in.
7. Compile and test your class / table.

On to the example!

Say you have 2 globals that look something like this:

^mapping("Simple",1)="Brendan Bannon^55192^SQL Manager"

^mapping("Simple",2)="Nicole Aaron^63375^Support Specialist"

^mapping("Simple",3)="Kyle Baxter^61155^Senior Support Specialist"

^mapping("Simple",4)="Prasad Kari^58471^Support Specialist"

^mapping("Simple",5)="Clive Morgan^57982^Senior Support Specialist"

^index("Simple","HireDate",55192,1)=""

^index("Simple","HireDate",57982,5)=""

^index("Simple","HireDate",58471,4)=""

^index("Simple","HireDate",61155,3)=""

^index("Simple","HireDate",63375,2)=""

Let's go through the 7 steps to create a class that will let use look at this data via SQL and Objects.

Step 1

The ^mapping global is about as simple as it gets. Each node looks like it contains the same type of data. We will deal with the ^index global after we finish with the ^mapping global.

Step 2

The first subscript is just a constant. The second subscript is an incremented counter that looks to be unique for each row of data.

Step 3

Looks like the properties could be: Name, HireDate, Title. Without looking at the ^index global or having someone that knows how the data is used it would be hard to know that 55192 is a date, let alone a hire date. Don ' t forget to define a property for the value in the second subscript.

Step 4

Property Name As %String;

Property HireDate As %Date;

Property Title As %String;

Property Sub2 As %Integer;

Step 5

You must define an IdKey index for every class that uses Caché SQL Storage

Index Master On Sub2 [IdKey];

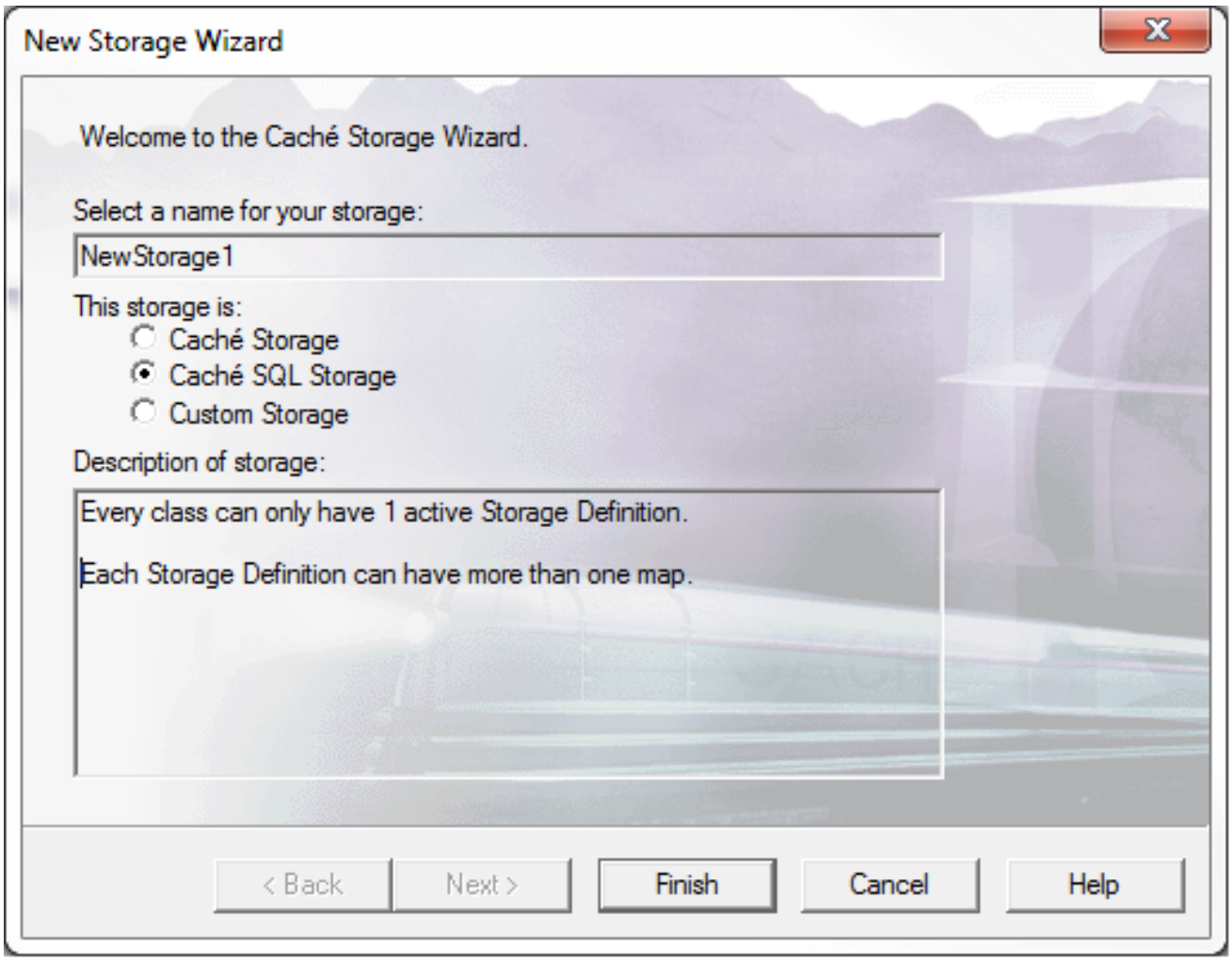
It is not required to define an index for the index maps, but it is a good idea.

Index hireDateindex On HireDate;

Step 6

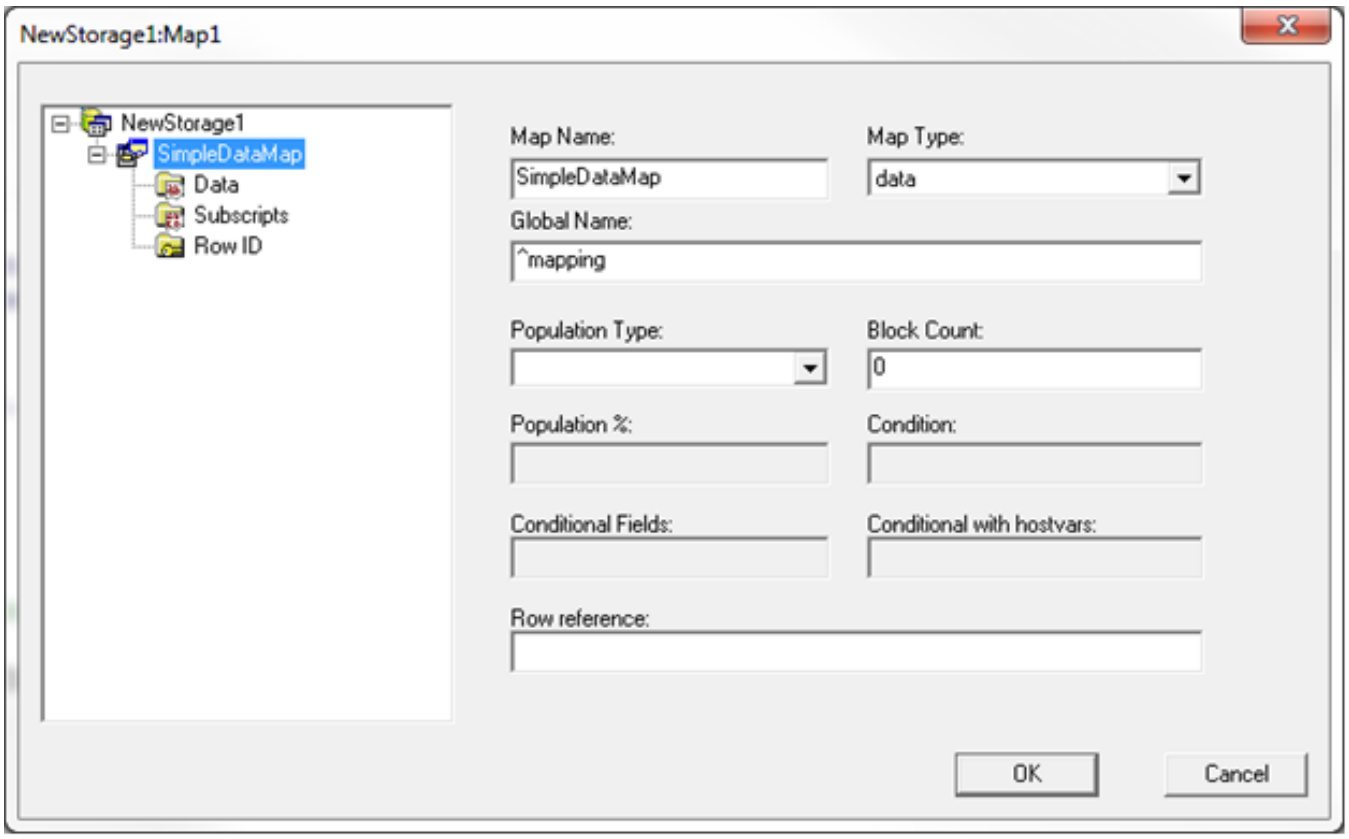
Now we need to create the mapping between the class and the global. I like the wizard so I will show the screen shots for that, but if you are good at typing xml you can define the Caché SQL Storage definition manually.

Click on the Storage icon. You can call the Storage Definition anything you want, I went with the default. Click on Caché SQL Storage and click finish.



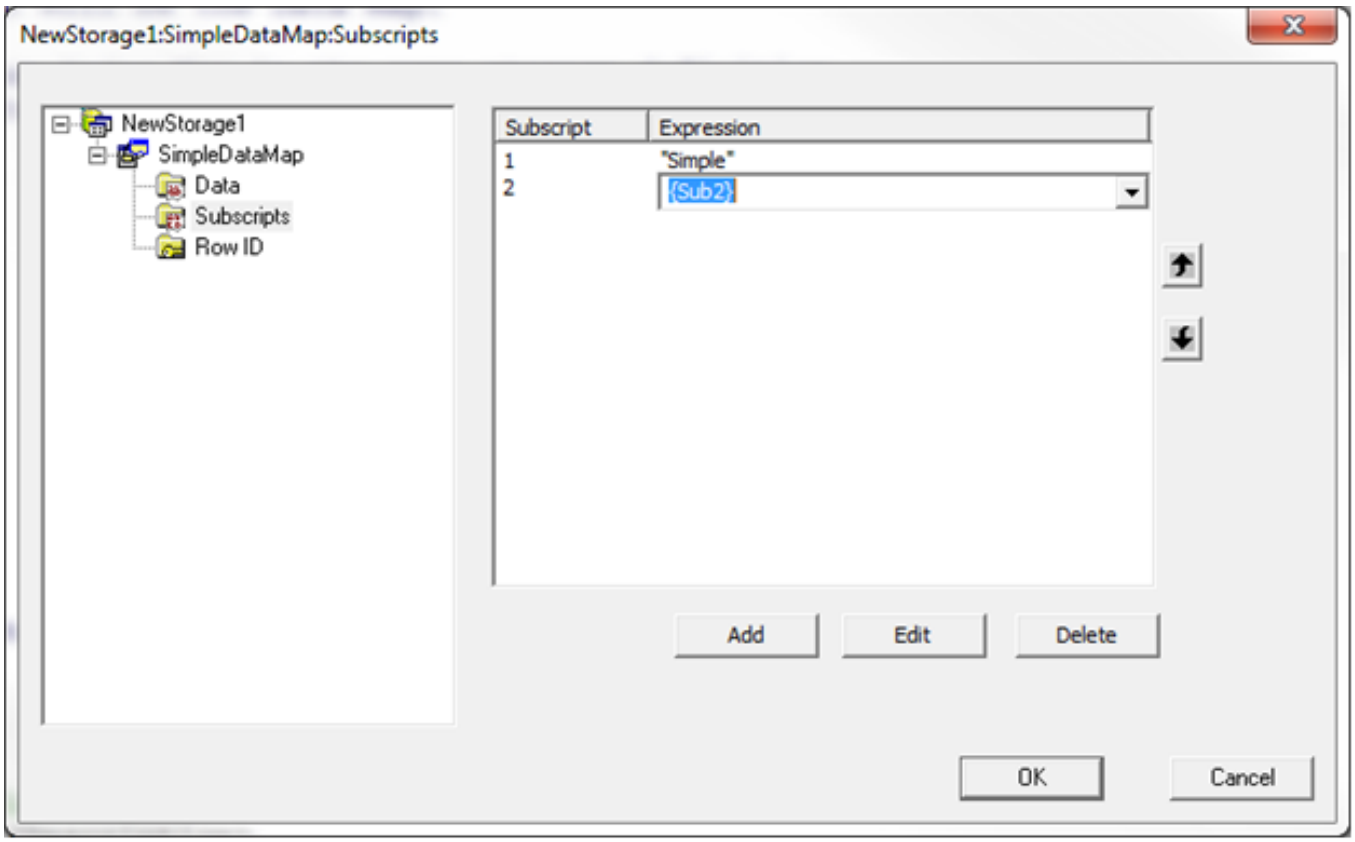
In the window below I changed the Map Name, but Map1 works as well. The only thing you need to fill in is the Global Name. Don ' t forget the " ^ ". Every persistent class needs to have one data map. It can have multiple index maps.

Clicking OK will exit you from the wizard so stay away from that until you are all done.



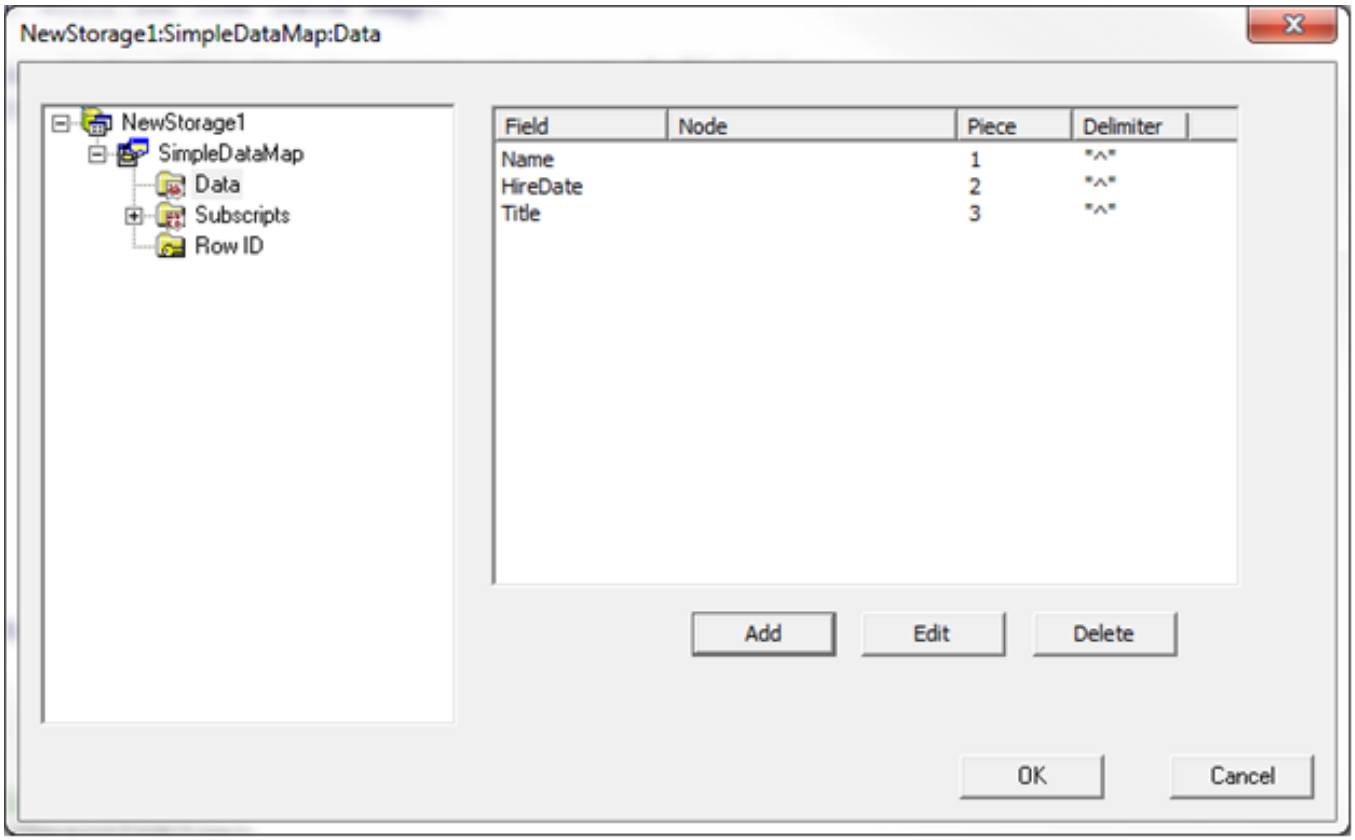
Step 6a

I like to do Subscripts next. In here we are going to define all the global subscripts up to and including the IdKey (the IdKey can be more than one subscript). In the expression box you refer to a field by putting it in {}. You can put in any valid COS expression here. We are keeping it simple so we just have a constant a Subscript 1 and a field at Subscript 2.



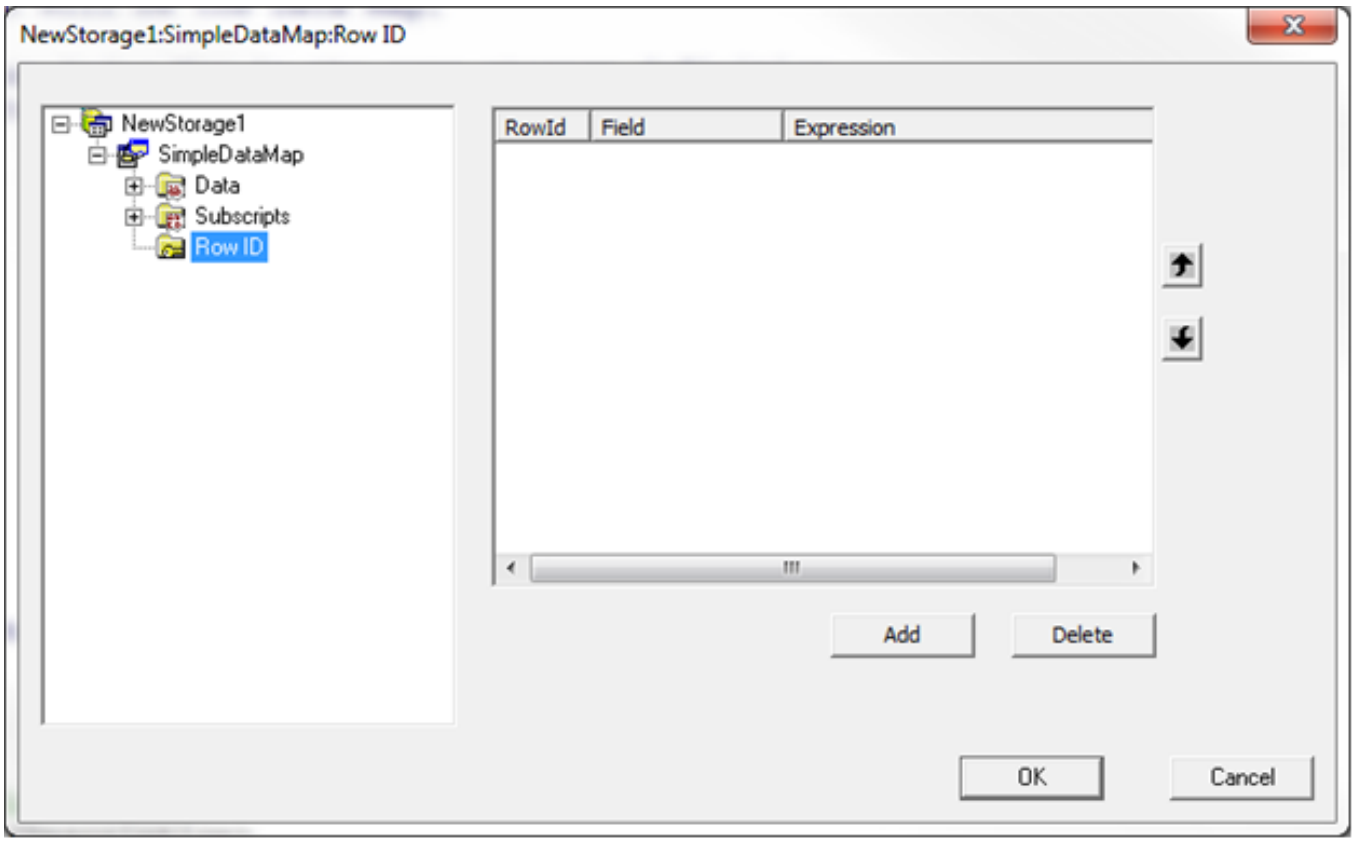
Step 6b

For data we are describing any subscripts that come after the IdKey plus all the data to the right of the equal sign. The Node column would be used for any additional subscripts (my next post will have an example of this). Piece and Delimiter are describing the location of the properties in the global. The default is to use the $PIECE command to parse the global data.

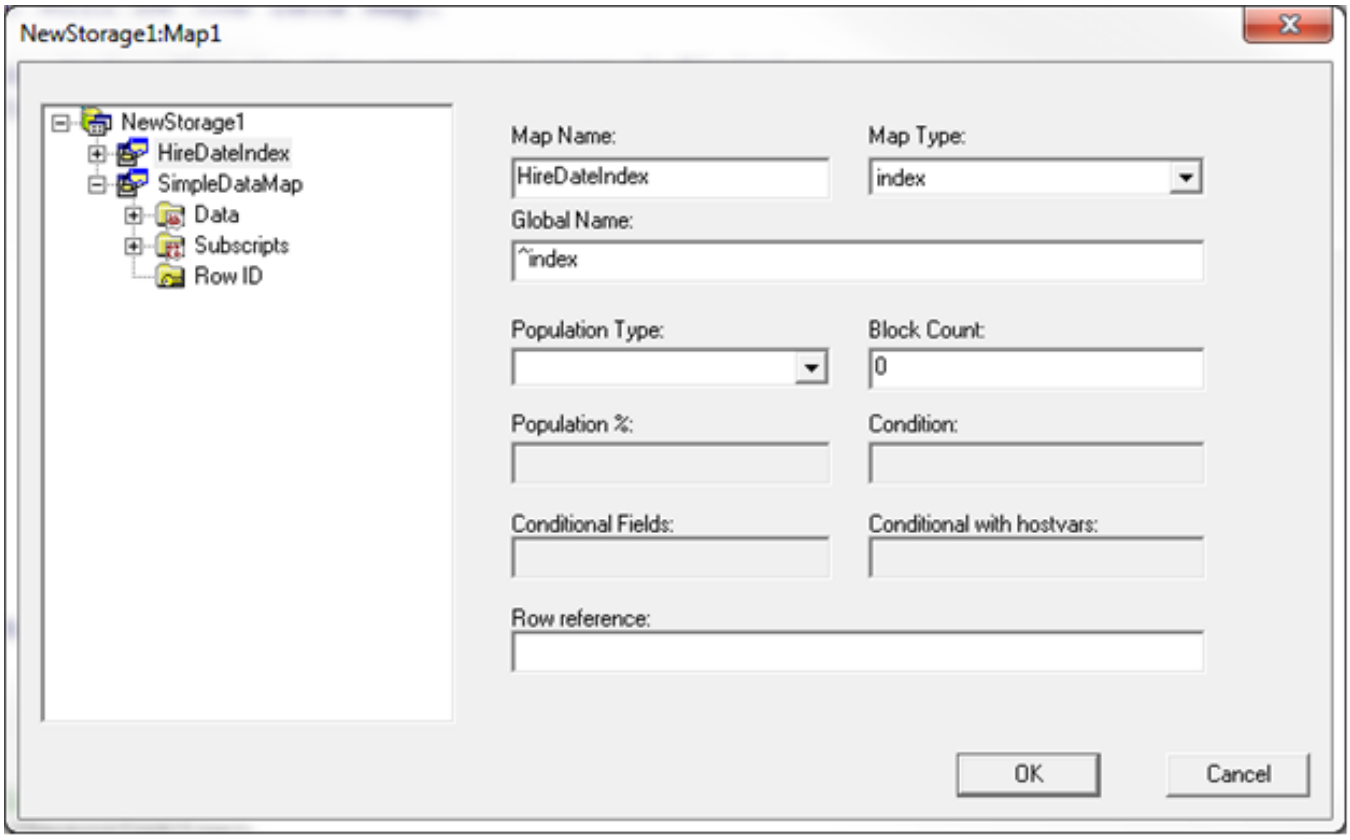


Step 6c

Nothing to see here, I told you to leave it blank.

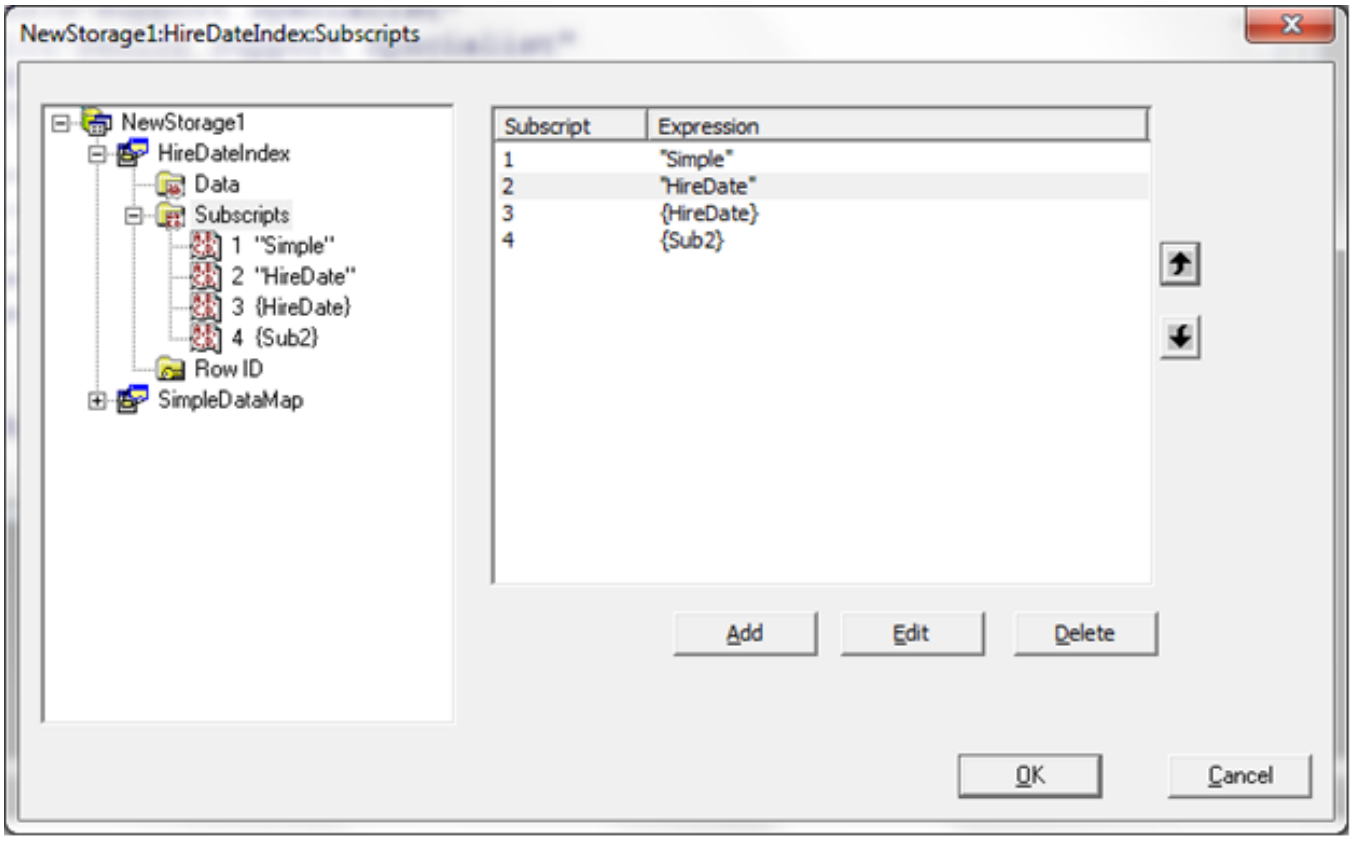


Now repeat step 6 for the ^index global.



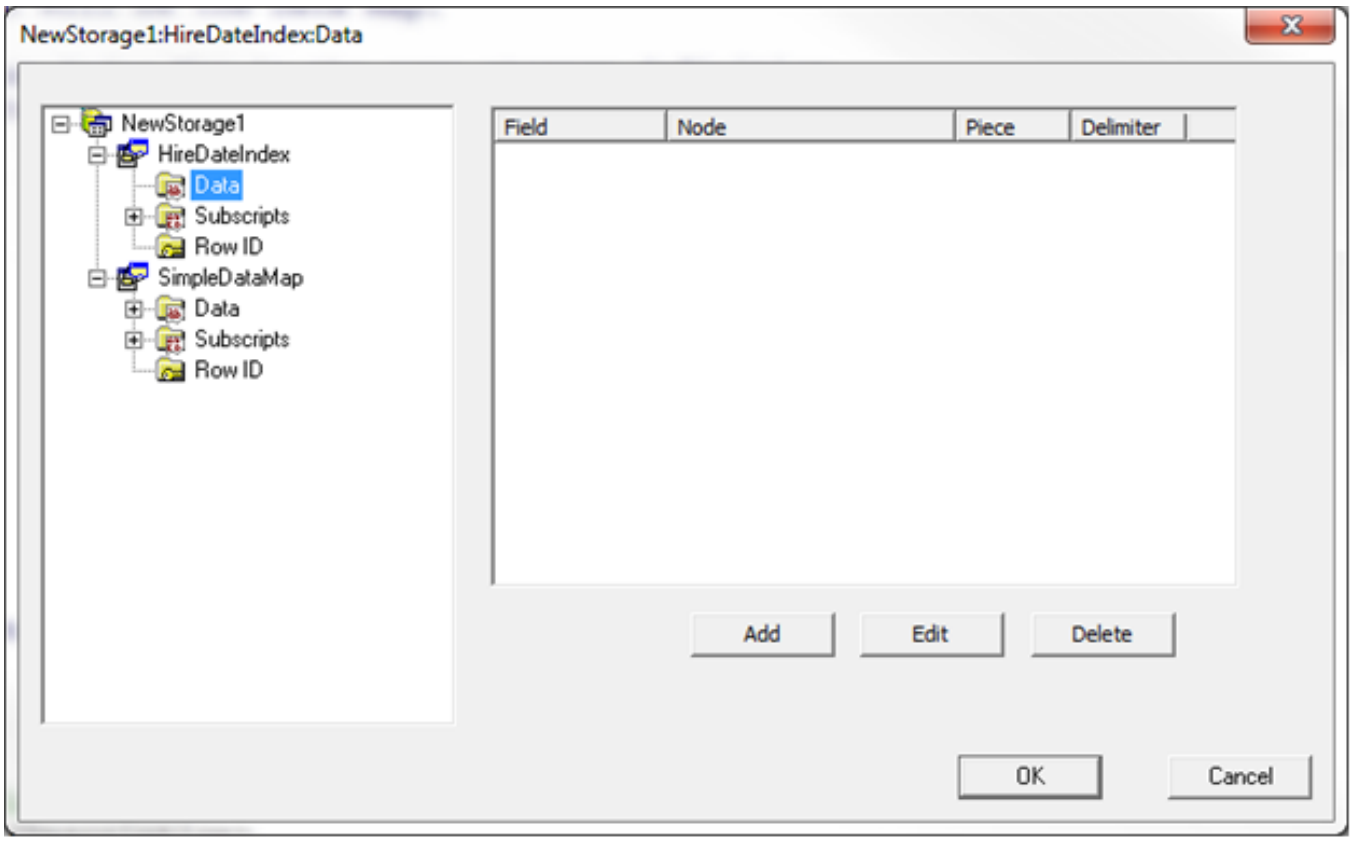
Step 6a

Four subscripts this time: two constants, HireDate and the IdKey (Sub2). Every Index map must be able to construct the IdKey. Most of the time it will be in the subscripts, but it could be part of the data.



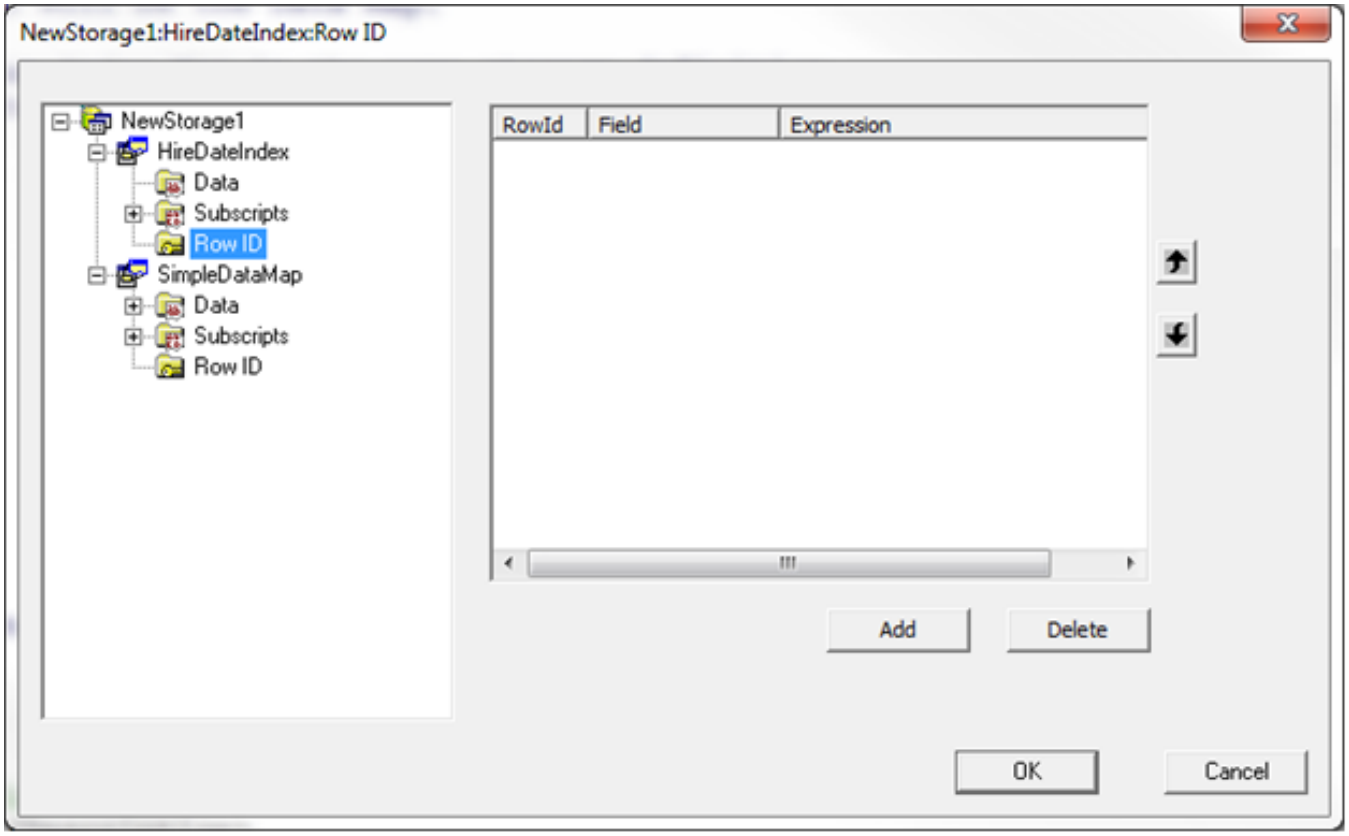
Step 6b

There are no additional subscripts and no data so this one is blank



Step 6c

Still nothing to do here.



Step 7

All that is left now is to compile the class and then try to query the table to make sure it correctly displays the data.

```
Compilation started on 08/15/2016 15:20:58 with qualifiers 'fck /checkuptodate=expandedonly'
Compiling class Mapping.Example1
Compiling table Mapping.Example1
Compiling routine Mapping.Example1.1
Compilation finished successfully in 0.270s.
```

```
SELECT Sub2, Name, HireDate, Title FROM Mapping.Example1
```

Sub2	Name	HireDate	Title
1	Brendan Bannon	1992-02-10	SQL Manager
2	Nicole Aaron	2014-07-07	Support Specialist
3	Kyle Baxter	2008-06-08	Senior Support Specialist
4	Prasad Kari	2001-02-01	Support Specialist
5	Clive Morgan	1999-10-01	Senior Support Specialist

And for those of you who like to type here is what the storage definition looks like in xml

```
/// Every class can only have 1 active Storage Definition.
/// Each Storage Definition can have more than one map.
Storage NewStorage1
{

<SQLMap name="HireDateIndex">
<ConditionalWithHostVars></ConditionalWithHostVars>
<Global>^index</Global>
<Subscript name="1">
<Expression>"Simple"</Expression>
</Subscript>
<Subscript name="2">
<Expression>"HireDate"</Expression>
</Subscript>
<Subscript name="3">
<Expression>{HireDate}</Expression>
</Subscript>
<Subscript name="4">
<Expression>{Sub2}</Expression>
</Subscript>
<Type>index</Type>
</SQLMap>
<SQLMap name="SimpleDataMap">
<Data name="HireDate">
<Delimiter>"^"</Delimiter>
<Piece>2</Piece>
</Data>
<Data name="Name">
<Delimiter>"^"</Delimiter>
<Piece>1</Piece>
</Data>
<Data name="Title">
<Delimiter>"^"</Delimiter>
<Piece>3</Piece>
</Data>
<Global>^mapping</Global>
<Subscript name="1">
<Expression>"Simple"</Expression>
</Subscript>
<Subscript name="2">
<Expression>{Sub2}</Expression>
</Subscript>
<Type>data</Type>
</SQLMap>
<StreamLocation>^Mapping.Example1S</StreamLocation>
<Type>%CacheSQLStorage</Type>
}
```

For those that don't want to type here is a file with the globals and class: mappingexample1.zip

If you want to learn more have a look at Part 2

#Best Practices #Globals #Mapping #SQL #Caché