Article
Robert Cemper · Apr 25, 2020 2m read

Semi-Persistent Classes and Tables

If you define a Persistent Class / Table the class compiler generates for you an appropriate Storage definition.
A different option is to define a SQL mapping for an already existing Global storage. This has been excellently explained already in a different series of articles. The Art of Mapping Globals to Classes 1 of 3

Once your storage map is defined it might be extended by the class compiler but the fundamental storage parameters will not change.
This does not mean that you can't change it manually yourself.

My article on The adopted Bitmap includes such a case. And in combination with some Static WHERE Conditions as described earlier you make it available also to SQL.

The typical definition of your storage globals may look like this example:

```
<DataLocation>^User.PersonD</DataLocation>
<IdLocation>^User.PersonD</IdLocation>
<IndexLocation>^User.PersonI</IndexLocation>
<StreamLocation>^User.PersonI</StreamLocation>
```

Now we change this definition into a dynamic one

```
<DataLocation>@%MyData</DataLocation>
<IdLocation>@%MyId</IdLocation>
<IndexLocation>@%MyIndex</IndexLocation>
<StreamLocation>@%MyStream</StreamLocation>
```

and we add some comfort

```
Parameter MANAGEDEXTENT As INTEGER = 0;

ClassMethod SetStorage(ref As %String) As %Integer [ SqlName = SetStorage, SqlProc ]
{
  set %MyData=ref_"D"
  , %MyId=%MyData
  , %MyIndex=ref_"I"
  , %MyStream=ref_"S"
  quit $$$OK
}
```

For object access we direct our storage and fill it

```
write ##class(Person).SetStorage("^mtemp.Person")
write ##class(Person).Populate(5)
```

and in SQL:

```
SELECT  * from Person where SetStorage('^mtemp.Person')=1
```

It works with PPG (**^||Person**)
across namespace (**^|"USER"|Person**)
also subscripted as used for The adopted Bitmap with another variant of **ClassMethod SetStorage()**

For object access (without SQL) it even works for local variables.
It's not the intended use but it demostrates how much flexibility is in this feature.

But take care. It will **NOT** work with Sharding. But that should not be surprising.

#ObjectScript #Other