Anuncio
Neerav Adam Verma · 13 feb, 2020

Class Queries

Class Queries in InterSystems IRIS (and Cache, Ensemble, HealthShare) is a useful tool that separates SQL queries from Object Script code. Basically, it works like this: suppose that you want to use the same SQL query with different arguments in several different places.In this case you can avoid code duplication by declaring the query body as a class query and then calling this query by name.

They are declared as Query items in class definitions (similar to Method or Property) in the following way:

- Type: %SQLQuery
- All arguments of your SQL query must be listed in the list of arguments
- Query type: SELECT
- Use the colon to access each argument (similar to static SQL)
- Define the ROWSPEC parameter which contains information about names and data types of the output results along with the order of fields
- (Optional) Define the CONTAINID parameter which corresponds to the numeric order if the field containing ID. If you don't need to return ID, don't assign any values to CONTAINID
- (Optional) Define the COMPILEMODE parameter which corresponds to the similar parameter in static SQL and specifies when the SQL expression must be compiled. When this parameter is set to IMMEDIATE (by default), the query will be compiled simultaneously with the class. When this parameter is set to DYNAMIC, the query will be compiled before its first execution (similar to dynamic SQL)
- (Optional) Define the SELECTMODE parameter which specifies the format of the query results
- Add the SqlProc property, if you want to call this query as an SQL procedure.
- Set the SqlName property, if you want to rename the query. The default name of a query in SQL context is as follows: PackageName.ClassNameQueryName
- Caché Studio provides the built-in wizard for creating class queries

Sample definition of the Person class with the ByName query which returns all user names that begin with a specified letter

```
Class NV.Person Extends %Persistent
{
Property Name As %String;
Property DOB As %Date;
Property SSN As %String;
Query ByName(name As %String = "") As %SQLQuery
    (ROWSPEC="ID:%Integer,Name:%String,DOB:%Date,SSN:%String",
     CONTAINID = 1, SELECTMODE = "RUNTIME",
     COMPILEMODE = "IMMEDIATE") [ SqlName = SP_NV_By_Name, SqlProc ]
{
SELECT ID, Name, DOB, SSN
FROM NV.Person
WHERE (Name %STARTSWITH :name)
ORDER BY Name
}
}
```

You can call this query from Caché Object Script in the following way:

```
Set statement=##class(%SQL.Statement).%New()
Set status=statement.%PrepareClassQuery("NV.Person","ByName")
If $$$ISERR(status) {
    Do $system.OBJ.DisplayError(status)
}
Set resultset=statement.%Execute("A")
While resultset.%Next() {
    Write !, resultset.%Get("Name")
}
```

Alternatively, you can obtain a resultset using the automatically generated method queryNameFunc:

```
Set resultset = ##class(NV.Person).ByNameFunc("A")
While resultset.%Next() {
    Write !, resultset.%Get("Name")
}
```

Alternative approach: %SQL.CustomResultSet

Alternatively, you can create a query by subclassing from the %SQL.CustomResultSet class. Benefits of this approach are as follows:

- Slight increase in speed
- ROWSPEC is unnecessary, since all metadata is obtained from the class definition
- Compliance with the object-oriented design principles

To create query from the subclass of %SQL.CustomResultSet class, make sure to perform the following steps:

1. Define the properties corresponding to the resulting fields
2. Define the private properties where the query context will be stored
3. Override the %OpenCursor method (similar to queryNameExecute) which initiates the context. In case of any errors, set %SQLCODE and %Message as well
4. Override the %Next method (similar to queryNameFetch) which obtains the next result. Fill in the properties. The method returns 0 if all the data has been processed and 1 if some data is still remaining
5. Override the %CloseCursor method (similar to queryNameClose) if necessary

Example of %SQL.CustomResultSet for Utils.CustomQuery:

```
Class Utils.CustomQueryRS Extends %SQL.CustomResultSet
{
Property Id As %String;
Property Prop1 As %String;
Property Prop2 As %Integer;
Method %OpenCursor() As %Library.Status
{
    Set ..Id = ""
    Quit $$$OK
}

Method %Next(ByRef sc As %Library.Status) As %Library.Integer [ PlaceAfter = %Execute
 ]
{
    Set sc = $$$OK
```

```
    Set ..Id = $Order(^Utils.CustomQueryD(..Id),1,val)
    Quit:..Id="" 0
    Set ..Prop1 = $Lg(val,2)
    Set ..Prop2 = $Lg(val,3)
    Quit $$$OK
}
}
```

You can call it from Caché Object Script code in the following way:

```
 Set resultset= ##class(Utils.CustomQueryRS).%New()
       While resultset.%Next() {
        Write resultset.Id,!
 }
```

#SQL #Ensemble #InterSystems IRIS

URL de fuente:https://es.community.intersystems.com/node/473966