Article
David Crawford · Jul 26, 2019 3m read

Open Exchange

Dynamic SQL to Dynamic Object

Hello community! I have to work with queries using all kinds of methods like embedded sql and class queries. But my favorite is dynamic sql, simply because of how easy it is to manipulate them at runtime. The downside to writing a lot of these is the maintenance of the code and interacting with the output in a meaningful way. In an effort to have as much dynamism as possible so that we're not rewriting code constantly, and so that we have as *little* code possible (while still making sense and getting the job done), I made a function that takes in any dynamic query and outputs a dynamic object.

The dynamic object has a key/value pair according to the result set's field names and values, which makes it very useful for my needs. Most of the time I will say %ToJSON() on these returned values and it's good to go for an api call response.

I'll say that one thing I'm most disgusted by in programming is hard-coded values. This may not be the case for you, but I see variants of this when code is interacting with a result set returned by a dynamic object:

```
set tResults = []

while rSet.%Next() {
    set tRow = {}
    set tRow.ProcessedDateTime = rSet.ProcessedDateTime
    set tRow.Destination = rSet.Destination
    set tRow.SourceOID = rSet.SourceOID
    set tRow.ResultType = rSet.ResultType
    set tRow.MessageDateTime = rSet.MessageDateTime
    set tRow.PatientLastName = rSet.PatientLastName
    set tRow.PatientFirstName = rSet.PatientFirstName
    set tRow.PatientDOB = rSet.PatientDOB
    set tRow.MRN = rSet.MRN
    set tRow.ResultStatus = rSet.ResultStatus
    set tRow.PatientClass = rSet.PatientClass
    set tRow.DocumentType = rSet.DocumentType
    set tRow.SessionID = rSet.SessionID
    set tRow.DelayEntry = rSet.DelayEntry
    set tRow.DestinationStatus = rSet.DestinationStatus
    set tRow.Report = rSet.Report
    set tRow.Provider = rSet.Provider
    do tResults.%Push(tRow)
}
```

There has to be a better way of doing this, because we're effectively controlling what data we want returned in two different places: the query, and the loop populating the object. I like things to be done in one place, and the query makes the most sense to me. So instead of naming each field, I want the loop to just figure it out on its own. We can accomplish this by using *gasp* one more loop:

```
set tResults = []
 while rSet.%Next() {
            set tRow = {}
            set tMetadata = rSet.%GetMetadata()
```

```
            set tColumnCount = tMetadata.columns.Count()
            for x=1:1:tColumnCount {
                set tColumn = tMetadata.columns.GetAt(x)
                set tColumnName = tColumn.colName
                set $PROPERTY(tRow,tColumnName) = $PROPERTY(rSet,tColumnName)
            }
            do tResults.%Push(tRow)
    }
```

The magic is in getting the correct metadata information, and then using the $PROPERTY function to *indirectly* manipulate the key/value pair. This lets us control what data we're getting out the query using the query itself, so if something is missing we know exactly where to look.

My code on Open Exchange has a couple other goodies to make life easier, such as parameters to select the mode, dialect, and namespace of each function call. Please let me know what you think! This is my first contribution and I'm eager to improve how I code in this language.

Thank you!

#Code Snippet #ObjectScript #SQL #Ensemble #InterSystems IRIS #Open Exchange
Check the related application on InterSystems Open Exchange