Article
Thomas Carroll · Nov 12, 2016 4m read

Using Embedded SQL in Caché Object Script Part #1

Embedded SQL is a tool that allows us to execute SQL statements in Caché Object Script. For example, to select the name of a person with a particular SSN from the Sample.Person class we can do the following:

```
&SQL(
SELECT Name into :tName
From Sample.Person
Where SSN = :tSSN
)
```

The colon syntax is used to identify local variables, in this case tName and tSSN. Here, tSSN will be defined **prior to execution** and tName will be set **during execution** of the statement. &SQL indicates to our compiler that this is SQL syntax. At compile time, this statement will be optimized to executable Object Script code. More about this syntax can be found here.

http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GSQLesql

SQLCODE

When using embedded SQL, the **only** way to determine whether an operation was successful is to check the **SQLCODE** variable after the statement is executed. SQLCODE is a special variable that holds an integer value describing the status of the last executed statement. The safe way to use embedded SQL is to religiously check this variable.

There are two success values that SQLCODE can have: 0 and 100:

SQLCODE	**Description**
0	Successful Completion
100	No (More) Data

The 0 value means that the operation was successful. However, 100 is a little different. It indicates that the operation was successful, but there was no data to act upon. This may have different meanings depending on the operation.

Operation	**Potential Meaning**
SELECT	Table contains no data
	Table contains no data meeting query criteria
	Row retrieval reached the final row (Cursors)
UPDATE and DELETE	Table contains no data

	Table contains no row satisfying the WHERE criteria

In cases where the SQLCODE is 100, the %ROWCOUNT, another SQL specific variable, will be 0 as well. Any value but 0 or 100 is an error. These error codes are well documented.

http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=RERRsqltable

Examples

Here is a statement where the SQLCODE will be 0 after execution:

```
NEW SQLCODE
&SQL(
Select Count(Name) into :tCount
From Sample.Person
)
```

After the following query the SQLCODE will be 100, because no error will occur and there is no row with an empty ID:

```
NEW SQLCODE
&SQL(
Select Name into :tName
From Sample.Person
Where ID=''
)
```

Using SQLCODE

The best practice for using SQLCODE with embedded SQL is to NEW it before executing an embedded SQL operation and to check it immediately after execution. This is true even within procedure blocks, especially if multiple SQL statements are being executed or if a CURSOR is defined.

Here is an example of an operation where the developer **must** check SQLCODE to be confident the UPDATE succeeded:

```
NEW SQLCODE
&SQL(
UPDATE Sample.Person
SET Name = 'Mickey Mouse'
WHERE SSN = '123-456-7890'
)
```

When this code is executed, if there is no record with SSN = ' 123-456-7890 ' then **no error** will occur. Even if the

developer is sure that this record exists, it is still poor practice to not check SQLCODE. The following is more robust:

```
NEW SQLCODE
&SQL(
UPDATE Sample.Person
SET Name = 'Mickey Mouse'
WHERE SSN = '123-456-7890'
)

If (SQLCODE=0){
    //Successful UPDATE!
}
ElseIf (SQLCODE=100){
    //No records with that SSN
}
Else {
    Write !,"Error Performing UPDATE"
}
```

Checking the SQLCODE allows the code to behave differently depending on whether or not there was a record that met our WHERE clause, and is the **only** way to know whether an error occurred.

However, more catastrophic mistakes can occur if the developer is not being careful. For example, if we have one record in the Sample.Person table with the name **" Adams,Alvin T."** and we write the following:

```
NEW SQLCODE
&SQL(
DECLARE C1  CURSOR FOR
SELECT TOP 1  Name
INTO :tName
FROM Sample.Person
WHERE Name %STARTSWITH 'Z'
)

&SQL(OPEN C1)
&SQL(FETCH C1)

Write tName

&SQL(CLOSE C1)
```

A developer might expect tName to be empty, because there are **no records** that have a name that starts with the character ' Z '. This is **not** the case. The value of tName in the above example is the string **" Adams,Alvin T."** which is the **last** record in the table**.** This is a simple example, but the point is that local variables used in embedded SQL can have unexpected values. Of course, if the developer checks SQLCODE the problem is avoided.

```
NEW SQLCODE
&SQL(
DECLARE C1 CURSOR FOR
SELECT TOP 1 Name
INTO :tName
```

```
FROM Sample.Person
WHERE Name %STARTSWITH ‘Z’
)

&SQL(OPEN C1)
&SQL(FETCH C1)

If (SQLCODE=0){
    Write tName
}
Else {
    Write !,“No records with that start with ‘Z’"
}

&SQL(CLOSE C1)
```

In the second part of this series we will go more in depth regarding how embedded SQL works, introduce the other special SQL variables, and discuss methodologies to implement error trapping with this syntax.

#ObjectScript #SQL #Caché