Recordset: Architecture (ODBC)

Article • 08/02/2021

This topic applies to the MFC ODBC classes.

This topic describes the data members that comprise the architecture of a recordset object:

- Field data members
- Parameter data members
- Using m_nFields and m_nParams data members

> **Note**
>
> This topic applies to objects derived from `CRecordset` in which bulk row fetching has not been implemented. If bulk row fetching is implemented, the architecture is similar. To understand the differences, see **Recordset: Fetching Records in Bulk (ODBC)**.

Sample Class

> **Note**
>
> The MFC ODBC Consumer wizard is not available in Visual Studio 2019 and later. You can still create a consumer manually.

When you use the MFC ODBC Consumer Wizard from **Add Class** wizard to declare a recordset class derived from `CRecordset`, the resulting class has the general structure shown in the following simple class:

C++

```
class CCourse : public CRecordset
{
public:
```

```
    CCourse(CDatabase* pDatabase = NULL);
    ...
    CString m_strCourseID;
    CString m_strCourseTitle;
    CString m_strIDParam;
};
```

At the beginning of the class, the wizard writes a set of field data members. When you create the class, you must specify one or more field data members. If the class is parameterized, as the sample class is (with the data member `m_strIDParam`), you must manually add parameter data members. The wizard does not support adding parameters to a class.

Field Data Members

The most important members of your recordset class are the field data members. For each column you select from the data source, the class contains a data member of the appropriate data type for that column. For example, the sample class shown at the beginning of this topic has two field data members, both of type `CString`, called `m_strCourseID` and `m_strCourseTitle`.

When the recordset selects a set of records, the framework automatically binds the columns of the current record (after the `Open` call, the first record is current) to the field data members of the object. That is, the framework uses the appropriate field data member as a buffer in which to store the contents of a record column.

As the user scrolls to a new record, the framework uses the field data members to represent the current record. The framework refreshes the field data members, replacing the previous record's values. The field data members are also used for updating the current record and for adding new records. As part of the process of updating a record, you specify the update values by assigning values directly to the appropriate field data member or members.

Parameter Data Members

If the class is parameterized, it has one or more parameter data members. A parameterized class lets you base a recordset query on information obtained or calculated at run time.

Typically, the parameter helps narrow the selection, as in the following example. Based on the sample class at the beginning of this topic, the recordset object might execute the following SQL statement:

```
SQL

SELECT CourseID, CourseTitle FROM Course WHERE CourseID = ?
```

The "?" is a placeholder for a parameter value that you supply at run time. When you construct the recordset and set its `m_strIDParam` data member to MATH101, the effective SQL statement for the recordset becomes:

```
SQL

SELECT CourseID, CourseTitle FROM Course WHERE CourseID = MATH101
```

By defining parameter data members, you tell the framework about parameters in the SQL string. The framework binds the parameter, which lets ODBC know where to get values to substitute for the placeholder. In the example, the resulting recordset contains only the record from the Course table with a CourseID column whose value is MATH101. All specified columns of this record are selected. You can specify as many parameters (and placeholders) as you need.

> **Note**
>
> MFC does nothing itself with the parameters — in particular, it does not perform a text substitution. Instead, MFC tells ODBC where to get the parameter; ODBC retrieves the data and performs the necessary parameterization.

> **Note**
>
> The order of parameters is important. For information about this and more information about parameters, see **Recordset: Parameterizing a Recordset (ODBC)**.

Using m_nFields and m_nParams

When a wizard writes a constructor for your class, it also initializes the m_nFields data member, which specifies the number of field data members in the class. If you add any parameters to your class, you must also add an initialization for the m_nParams data member, which specifies the number of parameter data members. The framework uses these values to work with the data members.

For more information and examples, see Record Field Exchange: Using RFX.

See also

Recordset (ODBC)
Recordset: Declaring a Class for a Table (ODBC)
Record Field Exchange (RFX)