**Tutorial. Accessing Database Using C# Part 2/3**

**Disconnected Mode**

V1.0

31/05/2022

In this tutorial you will study ADO.Net disconnected mode. Below are the topics of this tutorial.

**Topics**

[**1.** **What is disconnected mode** 1](#_Toc104853262)

[**2.** **Setup the project** 2](#_Toc104853263)

[**Step 1.** **Check the DB file** 2](#_Toc104853264)

[**Step 2.** **Create a C# console application** 3](#_Toc104853265)

[**Step 3.** **Create data models** 3](#_Toc104853266)

[**Step 4.** **Create DAL classes** 4](#_Toc104853267)

[**Step 5.** **Install “System.Data.SQLite”** 5](#_Toc104853268)

[**3.** **CRUD in connected mode** 5](#_Toc104853269)

[**The steps** 5](#_Toc104853270)

[**The StudentReadAll method** 6](#_Toc104853271)

[**4.** **CRUD in disconnect mode** 7](#_Toc104853272)

[**The steps** 7](#_Toc104853273)

[**The StudentReadAll method** 7](#_Toc104853274)

[**The StudentCreate method** 8](#_Toc104853275)

[**The StudentUpdate method** 9](#_Toc104853276)

[**The StudentDelete method** 10](#_Toc104853277)

[**5.** **Batch operation in the disconnected mode** 12](#_Toc104853278)

1. **What is disconnected mode**

Here is how does ADO.Net work

* Program build “Connection” to access databases
* There are two ways for program to read & update\*\* data
  + Build “Command” with input data 🡪 execute command to read / update data through connection 🡪 display results to output
  + Create “DataSet” and “DataAdapter”🡪 fill DataSet with data from data source 🡪 read / update data in DataSet locally 🡪 update changes to data source

\*\* Here, “update data” means insert, modify, delete

Data Source

Input

Output

Connection

Command

DataAdapter

DataSet

Execute

Build

Read / update

Fill / update

ADO.Net Framework

The way that using “Command” for data manipulation is called “connected mode”, while the other way that using “DataSet” and “DataAdapter” is called “disconnected mode”. Refer to the follow articles for more information about the two working modes.

<https://www.c-sharpcorner.com/UploadFile/8a67c0/connected-vs-disconnected-architecture-in-C-Sharp/>

And should you use connect mode or disconnected mode?

<https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff647768(v=pandp.10)?redirectedfrom=MSDN#dataset-vs-datareader>

In the next part, we will create an application to demonstrate using the two modes for database access.

1. **Setup the project**

We are going to create a console application to implement the CRUD operations for the following database.

Student

StudentID (PK)

Name

Subject

SubjectID (PK)

Title

Enrollment

ID (PK)

StudentID\_FK (FK)

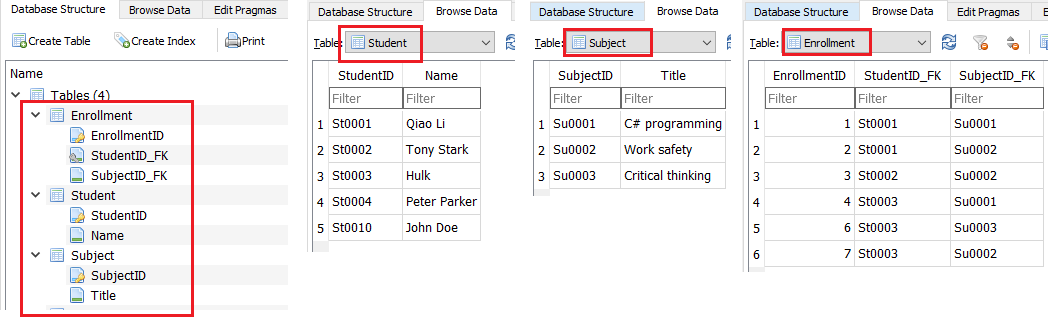
SubjectID\_FK (FK)

And here are the possible sample data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Table Student** | | | | **Student ID (PK)** | **Name** | | St0001 | Qiao Li | | St0002 | Tony Stark | | St0003 | Hulk | | St0004 | Peter Parker | | |  |  | | --- | --- | | **Table Subject** | | | **Subject ID (PK)** | **Title** | | Su0001 | C# programming | | Su0002 | Work safety | | Su0003 | Critical thinking | | |  |  |  |  | | --- | --- | --- | --- | | **Table Enrollment** | | | | | **Enrollment ID (PK)** | **Student ID (FK)** | **Subject ID (FK)** | | 1 | St0001 | Su0001 | | 2 | St0001 | Su0002 | | 3 | St0002 | Su0002 | | 4 | St0003 | Su0001 | |

* 1. **Check the DB file**

Open the DB file with SQLite DB Browser and verify the contents.



* 1. **Create a C# console application**
  2. **Create data models**

Create a folder called models and three mode classes

Text

Description automatically generated

**Models\Student.cs**

Text

Description automatically generated

**Models\Subject.cs**

Text

Description automatically generated

**Models\Enrollment.cs**Text

Description automatically generated

* 1. **Create DAL classes**

Create the two DAL class files and keep them empty for now. And here is the file structure by the end of this stage

Text

Description automatically generated

* 1. **Install “System.Data.SQLite”**

|  |
| --- |
| **Please note**  In the previous tutorial, C# SQLite library “Microsoft.Data.Sqlite” was used. This library is developed by Microsoft, however it does NOT support the disconnected mode. An alternative C# SQLite library is “System.Data.SQLite”, which is maintained by SQLite team, and it support the disconnected mode. You may find more information in the following web pages  <https://docs.microsoft.com/en-us/dotnet/standard/data/sqlite/compare>  <https://stackoverflow.com/questions/51933421/system-data-sqlite-vs-microsoft-data-sqlite#answer-52025556>  In this tutorial, we will be using “System.Data.SQLite”. |

In NuGet Packet Management GUI too, search and install the package, like below

A screenshot of a computer

Description automatically generated with medium confidence

Now you are all set. In the next parts, we will implement two types of connection, connected mode and disconnected mode, for the application to communicate with the database. Then implement CRUD functionalities in both modes.

1. **CRUD in connected mode**

We will start with the connected mode. You have studied disconnected in the part 1 of this tutorial. The same methods will be applied in this tutorial. Let review the steps again

**The steps**

1. Create the connection object
2. For each operation
   1. Open the connection
   2. Construct the SQL query
   3. Execute the SQL query
   4. Read data from query results
   5. Close the connection

**The StudentReadAll method**

Code the DALConnected class like so in the “DALConnected.cs” file

Text

Description automatically generated

Then test the class with the following codes

Text

Description automatically generated

And the result

Text

Description automatically generated

All other operations using connected mode follow the same steps as above and have also been demonstrated in a previous tutorial. The coding will not be implemented in this example. Complete all CRUD for all three entities in your own time. The focus of this tutorial is the disconnected mode.

1. **CRUD in disconnect mode**

With disconnected mode a subset of the database is loaded to a representation variable, called DataSet, in the application through an SQL adapter object. All operations are done with the DataSet, without any connection to database. And the adapter will be able to reflect the changes back to the database. Also, the adapter is able to synchronize the data between the Database and the DataSet.

**The steps**

1. Create the connection object
2. Create DataSet, DataAdapter
3. Populate DataSet with data from database
4. Read
   1. Refresh DataSet with newest data from data source
   2. Read data from DataSet
   3. Display data to output
5. Update (insert, modify, delete)
   1. Refresh DataSet with newest data from data source
   2. Insert, modify, delete data in DataSet
   3. Send the changes to data source

**The StudentReadAll method**

1. In DALDisconnected class, define adapter, dataset and a query string

Text

Description automatically generated

1. In the constructor, create all required objects, populate data to data set

Text

Description automatically generated

1. Implement the StudentReadAll method

Text

Description automatically generated

1. Test the disconnected StudentReadAll method

Text

Description automatically generated

1. Check the results

Text

Description automatically generated

**The StudentCreate method**

1. Create the StudentCreate method

Text

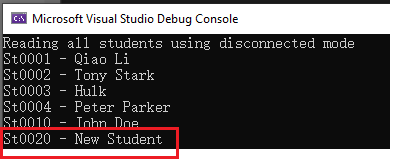
Description automatically generated

1. Test the above method

A screenshot of a computer

Description automatically generated

1. Check the results



**The StudentUpdate method**

1. Create the StudentUpdate method

Text

Description automatically generated

1. Test the above method

Text

Description automatically generated

1. Check the results

Text

Description automatically generated

**The StudentDelete method**

1. Create the StudentDelete method

Text

Description automatically generated

1. Test the above method

Text

Description automatically generated

1. Check the results

Text

Description automatically generated

1. **Batch operation in the disconnected mode**

With disconnected mode, we could easily apply multiple operations to the DataSet and update all changes at last. Here is an example

1. Create the StudentBatchOperation method

|  |
| --- |
| public void StudentBatchOperation()  {  var studentTable = DBDataSet.Tables["Student"];  Console.WriteLine("Add a student - St1000");  // create a row object  DataRow row1 = studentTable.NewRow();  // set the values for the new row  row1["StudentID"] = "St1000";  row1["Name"] = "Batch 1";  // add the new row in student table  studentTable.Rows.Add(row1);  Console.WriteLine("Add another student - St1000");  DataRow row2 = studentTable.NewRow();  // set the values for the new row  row2["StudentID"] = "St2000";  row2["Name"] = "Batch 2";  // add the new row in student table  studentTable.Rows.Add(row2);  Console.WriteLine("Update a student - St3000");  foreach (DataRow row in studentTable.Rows)  {  if (String.Equals(row["StudentID"].ToString(), "St3000"))  {  // update the row  row["Name"] = "Jane updated";  break;  }  }  Console.WriteLine("Delete a student - St4000");  foreach (DataRow row in studentTable.Rows)  {  if (String.Equals(row["StudentID"].ToString(), "St4000"))  {  // update the row  row.Delete();  break;  }  }  // automatically build required insert/update/delete command  new SQLiteCommandBuilder(DataAdapter);  // update the database  DataAdapter.Update(studentTable);  } |

1. Test the above method

Text

Description automatically generated

1. Check the results

Text

Description automatically generated

End of this tutorial