Visual C# .Net using framework 4.5

Eng. Mahmoud Ouf

Lecture 10

Dealing with Database

To Deal with the Database, there are two approaches:

Connected Model:

In which we maintain an open connection with the database while performing Database transactions

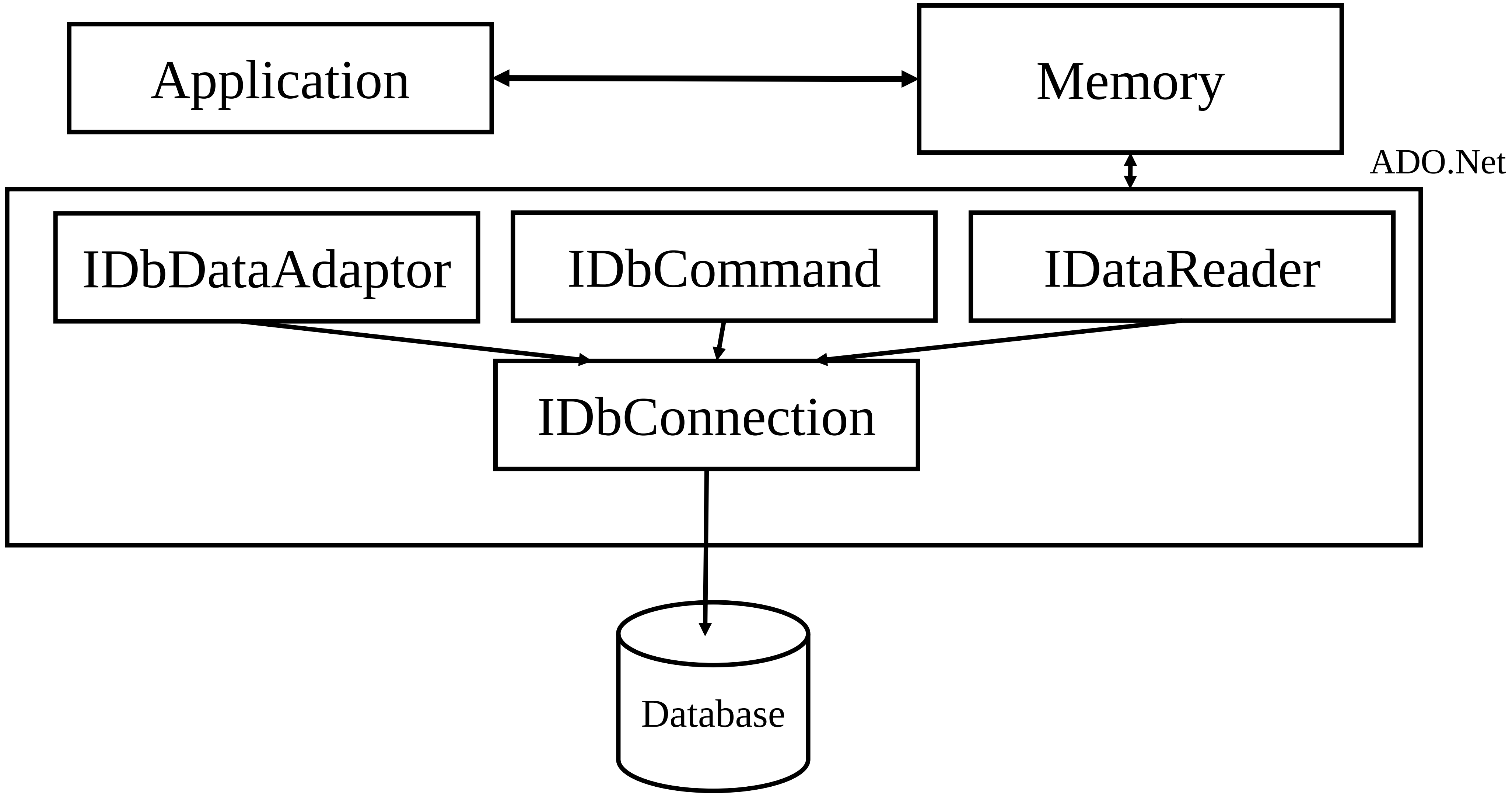
Disconnected Model:

In which we open connection with the database, retrieve the database (Tables, Relations, Columns, Rows, …) to the Memory then close the connection.

Perform the database transaction on the database copy in the memory

Then, open the database connection and return the updates to the real Database

ADO .Net Data Architecture



Working with Disconnected Model (DataSet)

It is a disconnected, in-memory representation of data. When appropriate, The Data Set can act as a template for updating the central database.

The DataSet object contains a collection of zero or more DataTable objects. Each is an in memory representation of a single table.

The DataTable is defined by DataColumns collection and the Constrains Collections. These two makes up the schema of the table. A DataTable contains a DataRows Collections.

The DataSet contains a DataRelations collections (allows the creation of association between one table and rows in another tables.

Working with Disconnected Model (DataSet)

Create a DataSet

DataSet ds = new DataSet(“MyDataSet”);

Create 2 tables (Employee) and (Department)

DataTable Emp = new DataTable(“Employee”);

DataTable Dept = new DataTable (Department”);

Create 5 column

DataColumn EmpId = new DataColumn(“ID”, Type.GetType(“System.Int32”));

DataColumn EmpName = new DataColumn(“Name”, Type.GetType(“System.string));

DataColumn EmpDeptId = new DataColumn(“DeptID”, Type.GetType(“System.Int32”));

DataColumn DeptId = new DataColumn(“ID”, Type.GetType(“System.Int32”));

DataColumn DeptName = new DataColumn(“Name”, Type.GetType(“System.string));

Working with Disconnected Model (DataSet)

Add 3 to table “Employee” and 2 to table “Department”

```
		Dept.Columns.Add(DeptId);
		Dept.Columns.Add(DeptName);
		Emp.Columns.AddRange(New DataColumns[]{EmpId,	EmpName,
EmpDeptId});
```

Add the 2 tables to the dataset

```
		DataSet.Tables.Add(Emp);
		DataSet.Tables.Add(Dept);
```

Create a Relation

```
		DataRelation dr = new DataRelation(“EmpDept”, DeptId,	EmpDeptId);
```

Add the relation to the DataSet

```
		ds.DataRelations.Add(dr);
```

Working with Disconnected Model (DataSet)

Create Rows

```
DataRow dRow = Emp.NewRow();
dRow[0] = 1;
dRow[“Name”] = “Aly”;
```

Add rows to the Table

```
Emp.Rows.Add(dRow);
```

Working with Connected Model

The Data Provider:

Data provider is a set of related component that work together.

.Net has 2 data provider:

    Sql server (using System.Data.SqlClient)
    OleDb   (using System.Data.OleDb)

Establish a connection using OleDbConnection:

Create object from OleDbConnection

```
OleDbConnection cn = new OleDbConnection();
```

Set the ConnectionString property:

```
cn.ConnectionString = “Provider = ……” + “Data Source=……”;
```

    Provider: Microsoft.Jet.OLEDB.4.0 (access)
              SQLOLEDB   (sql server below 6.5)

Open Connection

```
cn.Open();
```

Working with Connected Model

Building a SQL command using OleDbCommand:

Create an object from OleDbCommand

```
OleDbCommand m_Com = new OleDbCommand();
```

Create the SQL Command:

```
string str = “Select Make from Inventory where color = red”;
```

Set the OleDbCommand object properties

```
m_Com.Connection = cn;
m_Com.CommandText = str;
```

Open the connection and execute the command:

ExecuteReader: Return an instance of OleDbDataReader (select)

ExecuteScalar: Return a single value from database query (aggregate function)

ExecuteNonQuery: command with no return value (insert, update, delete)

Working with Connected Model

Working with OleDbDataReader:

Provide a forward only, readonly connected record set.

It can’t be instantiated, it comes from the execution of ExecuteReader method

It only return one record at a time in the memory (we have to iterate over it)

```
OleDbDataReader myReader;
myReader = m_Com.ExecuteReader();

While(myReader.Read())
{
        Console.WriteLine(“Red Car:”+myReader[“Make”].ToString());
}
myReader.Close()
Cn.Close();
```

Working with DisConnected Model

Working with OleDbDataAdapter:

OleDbDataAdapter dAdapt = new OleDbDataAdapter(cn, m_Com);
DataSet ds = new DataSet();
dAdapt.Fill(ds);
OleDbDataAdapter provides 4 properties represent the database command:
SelectCommand
InsertCommand
UpdateCommand
Delete Command

After finishing, to return values to database use:
dAdapt.update();