Module 2

ADO.NET

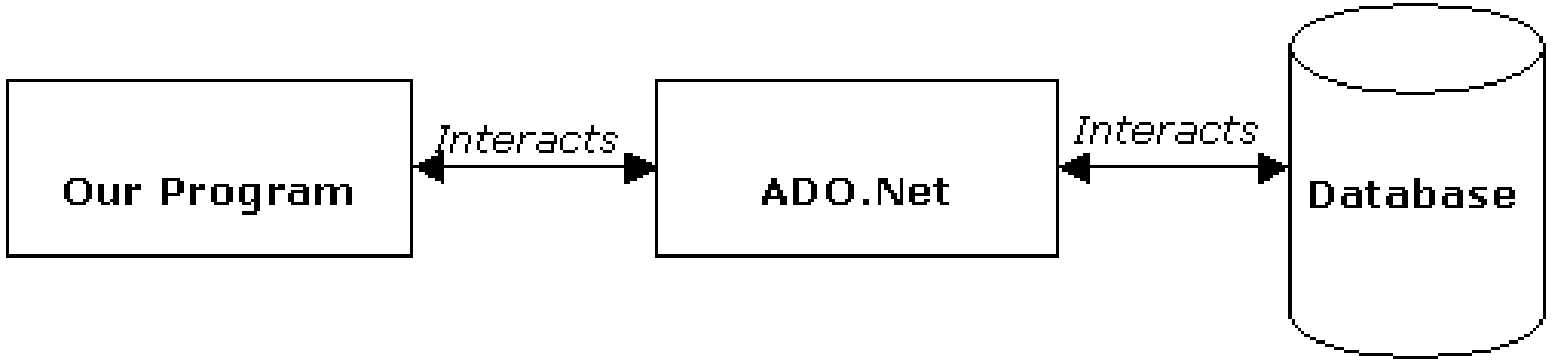
.NET Data Access and Manipulation

What is ADO.NET?

- A data-access technology that enables applications to connect to data stores and manipulate data contained in them in various ways
- Former version was ADO (ActiveX Data Object)

What is ADO.NET?

- An object oriented framework that allows you to interact with database systems



Introduction

- ADO.NET provides consistent access to data sources such as Microsoft SQL Server, as well as data sources exposed through OLE DB and XML.
- Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data.
- ADO.NET cleanly factors data access from data manipulation into discrete components that can be used separately or in tandem.
- ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results.
- Those results are either processed directly, or placed in an ADO.NET **DataSet** object in order to be exposed to the user in an ad-hoc manner, combined with data from multiple sources, or remoted between tiers.
- The ADO.NET **DataSet** object can also be used independently of a .NET Framework data provider to manage data local to the application or sourced from XML.

- ADO.NET is the data access component for the .NET Framework.
- This model enables the communication with the databases and as well as other structures like arrays and collections also.
- It supports a data centric application development and uses the disconnected architecture for accessing the data. Thus it conserves the system resources and reduces the overhead of network traffic.
- ADO.NET is a data access technology which provides communication between relational and non relational database systems. This leverages the power of XML to provide disconnected data accessibility.
- ADO.NET is establishing a connection with a data source, send queries and update statements to the data source and the returns the result.
- It is a collection of interfaces, classes and structures for managing the data access from different databases.
- The ADO.Net Object model is based upon the objects of System.Data namespace

Features

- **Disconnected Architecture** : ADO.NET uses the disconnected Architecture . Applications connect to the database only when retrieving and updating data. After that connection is closed. When the database needs to be updated the connection is re established.
- **Data cached in datasets** : dataset is a common method of accessing data in a disconnected architecture. Dataset is a catched set of records.
- ADO.NET supports scalability by working with datasets. Operations are performed in the set instead of database, so resources are saved.

- **Data transfer in XML format** : Data is transferred from a database into a dataset and from the dataset to another component by using XML. We can use XML as a data source and store data from it in a dataset. Knowledge of XML is not required.
- **Interaction through commands :All** operations are performed using commands. A data command can be a SQL statement or a stored procedure. We can retrieve , insert, delete or modify data from a database using commands.
- Automatic Connection Pooling
- Enables integration of data from multiple heterogeneous data sources

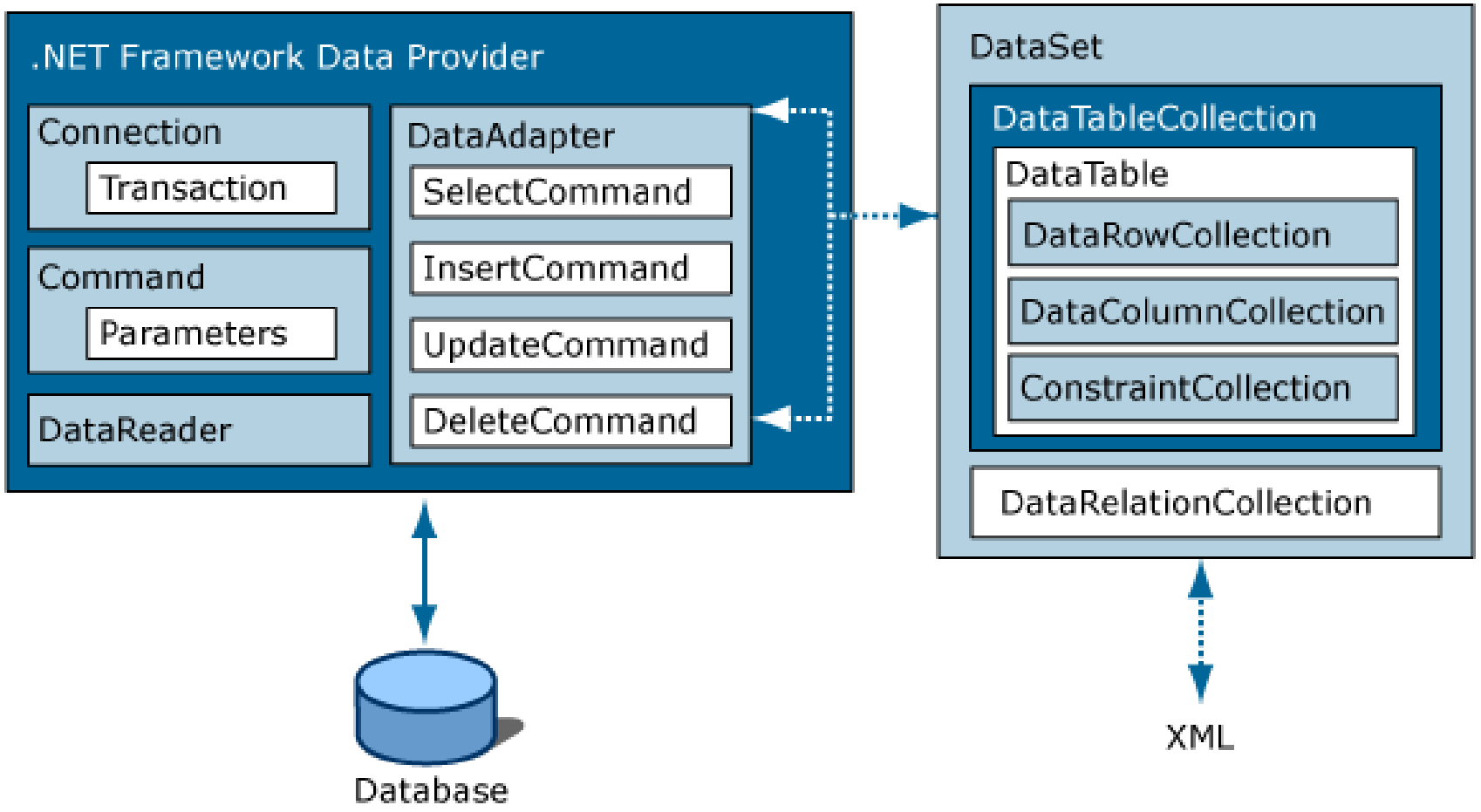
ADO Vs ADO.NET

ADO.NET	**ADO**
Object Oriented	**O**bject based
Disconnection Oriented Model	Connection Oriented Model
Dataset object is used	ADO uses Recordset object
One dataset is a collection of one or more tables and their relations.	Recordset stores only one table from the database. Join query is used for multiple tables
ADO.Net uses XML, XSD for interchange of data	Data exchange between application is provided through COM
Richer data types support for data exchange	Because of COM Marshalling, limited set of data types
Full support for XML	Limited Support
Communication to the databases is established by data adapter. Data adapter calls the OLEDB provider.	Communication to the databases are established by making calls to OLEDB provider

Objective of ADO.NET

- Support disconnected data architecture
- Tight integration with XML
- Common data representation
- Ability to combine data from multiple and varied data sources
- Optimizedfacilities for interacting with a database

ADO.NET Architecture



ADO.NET Core Objects

- Core namespace: System.Data
- .NET Framework data providers:

Data Provider	Namespace
SQL Server	`System.Data.SqlClient`
OLE DB	`System.Data.OleDb`
ODBC	`System.Data.Odbc`
Oracle	`System.Data.OracleClient`

ADO.NET Core Objects

Object	Description
`Connection`	Establishes a connection to a specific data source. (Base class: DbConnection)
`Command`	Executes a command against a data source. Exposes **Parameters** and can execute within the scope of a **Transaction** from a **Connection**. (The base class: DbCommand)
`DataReader`	Reads a forward-only, read-only stream of data from a data source. (Base class: DbDataReader)

Steps of Data Acces : Connected Environment

1. Create connection
2. Create command (select-insert-update-delete)
3. Open connection
4. If SELECT -> use a **DataReader** to fetch data
5. If UPDATE,DELETE, INSERT -> use command object’s methods
6. Close connection

```
static void Main()
{
    string connectionString =
                        Properties.Settings.Default.connStr;
    string queryString = "SELECT CategoryID, CategoryName FROM
                                            dbo.Categories;";
    SqlConnection connection = new
                                SqlConnection(connectionString);

    SqlCommand command = new SqlCommand(queryString,connection);
    try
    {
        connection.Open();
        SqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine("\t{0}\t{1}",reader[0],reader[1]);
        }
        reader.Close();
         connection.close();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

EXAMPLE

Connected – Update, Delete, Insert

- Command class core methods:
  - **`ExecuteNonQuery`** : Executes a SQL statement against a connection object
  - **`ExecuteReader`**: Executes the CommandText against the Connection and returns a **`DbDataReader`**
  - **`ExecuteScalar`**: Executes the query and returns the first column of the first row in the result set returned by the query

Connected – Update, Delete, Insert

```
string connString =
       Properties.Settings.Default.connStr;
SqlConnection conn = new
        SqlConnection(connString);
SqlCommand cmd = new SqlCommand("delete from
    Customers" + "where custID=12344", conn);
conn.Open();
cmd.ExecuteNonQuery();
conn.Close();
```

Can be an update or insert command