ADO.NET in C#.NET

(Communication between .net applications and databases)

Introduction to ADO.NET

ADO.NET (ActiveX Data Objects for .NET) is a data access technology in the .NET framework that enables communication between applications and databases. It provides a set of classes to interact with various data sources such as SQL Server, Oracle, and MySQL.

Features of ADO.NET

- **Disconnected Data Access**: Uses `DataSet` to store data independently of the database.
- **Scalability**: Efficient for large applications as it supports connection pooling.
- **Interoperability**: Works with different database systems using providers.
- **Security**: Provides secure data access through connection authentication.

ADO.NET Architecture

ADO.NET consists of two major components:

1. **Connected Architecture** (Using `DataReader`)
    - Works with `SqlConnection`, `SqlCommand`, `SqlDataReader`
    - Requires an active connection to the database
    - Fast and forward-only data retrieval
2. **Disconnected Architecture** (Using `DataSet`)
    - Uses `SqlDataAdapter`, `DataSet`, `DataTable`
    - Stores data in memory for offline processing
    - Useful for manipulating data without keeping the database connection open

Important ADO.NET Classes (Connected Architecture)

Namespace : Microsoft.Data.SqlClient

1. **SqlConnection**: Establishes a connection to the database.

```
SqlConnection conn = new SqlConnection("your_connection_string");
conn.Open();
```

2. **SqlCommand**: Executes SQL queries and stored procedures.

```
SqlCommand cmd = new SqlCommand("SELECT * FROM Students", conn);
```

Note: Install *Microsoft.Data.SqlClient* package using NuGet Package Manager.

3. **SqlDataReader**: Reads data in a forward-only manner (Connected Mode).

```
SqlConnection conn = new SqlConnection("your_connection_string");

SqlCommand cmd = new SqlCommand("SELECT * FROM Students", conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

while(reader.Read())
{
    Console.WriteLine(reader["StudentName"].ToString());
}
```

Executing Queries in ADO.NET

1. Executing Non-Query (Insert, Update, Delete)

```
SqlCommand cmd = new SqlCommand("INSERT INTO Students (Name, Age) VALUES ('John', 20)", con);
int rowsAffected = cmd.ExecuteNonQuery();
```

2. Executing Scalar Query (Single Value)

```
SqlCommand cmd = new SqlCommand("SELECT COUNT(*) FROM Students", con);
int count = (int)cmd.ExecuteScalar();
```

Perform CRUD Operations

Create (INSERT Operation)

```
 SqlConnection con = new SqlConnection("your_connection_string");
string query = "INSERT INTO Students (Name, Age) VALUES (@Name, @Age)";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@Name", "Alice");
cmd.Parameters.AddWithValue("@Age", 22);
cmd.ExecuteNonQuery();
```

Read

```
SqlConnection conn = new SqlConnection("your_connection_string");

  SqlCommand cmd = new SqlCommand("SELECT * FROM Students", conn);

  conn.Open();

  SqlDataReader reader = cmd.ExecuteReader();

  while(reader.Read())
  {
      Console.WriteLine(reader["StudentName"].ToString());
  }
```

UPDATE

```
 SqlConnection conn = new SqlConnection("your_connection_string");

string query = "UPDATE Students SET Age = @Age WHERE Name = @Name";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@Name", "Alice");
cmd.Parameters.AddWithValue("@Age", 23);

cmd.ExecuteNonQuery();
```

DELETE

```
 SqlConnection conn = new SqlConnection("your_connection_string");

string query = "DELETE FROM Students WHERE Name = @Name";
SqlCommand cmd = new SqlCommand(query, con);
cmd.Parameters.AddWithValue("@Name", "Alice");

cmd.ExecuteNonQuery();
```

Important ADO.NET Classes (DisConnected Architecture)

1. **SqlDataAdapter**: Fills the `DataSet` with database data.

```
SqlDataAdapter da = new SqlDataAdapter("SELECT * FROM Students", con);
DataSet ds = new DataSet();
da.Fill(ds);
```

2. **DataSet**: Stores data in memory as a collection of `DataTable` objects.

```
DataTable dt = ds.Tables[0];
```

3. **DataTable**: Represents a single table of data.

```
foreach (DataRow row in dt.Rows) {
    Console.WriteLine(row["StudentName"].ToString());
}
```

Note: DataSet and DataTable belongs to *System.Data* namespace

Conclusion

- ADO.NET provides a flexible and efficient way to interact with databases in C#.NET applications.
- It supports both connected and disconnected models, making it suitable for various application scenarios.