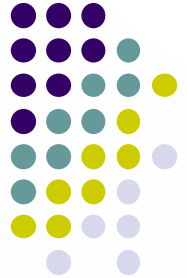
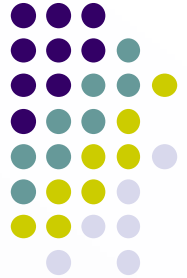


DATABASE CONNECTION

2020-11-15

Database Access



- ODBC Series

ODBC	**Lower level approach**
DAO	Database engine for Access
RDO	Overcome the shortcomings of DAO: access ODBC API directly
OLE DB	component structure : SQL data source and non-sql data source
ADO	VB programmer
ADO.NET	.NET era

ODBC

- ODBC（Open Database Connectivity）

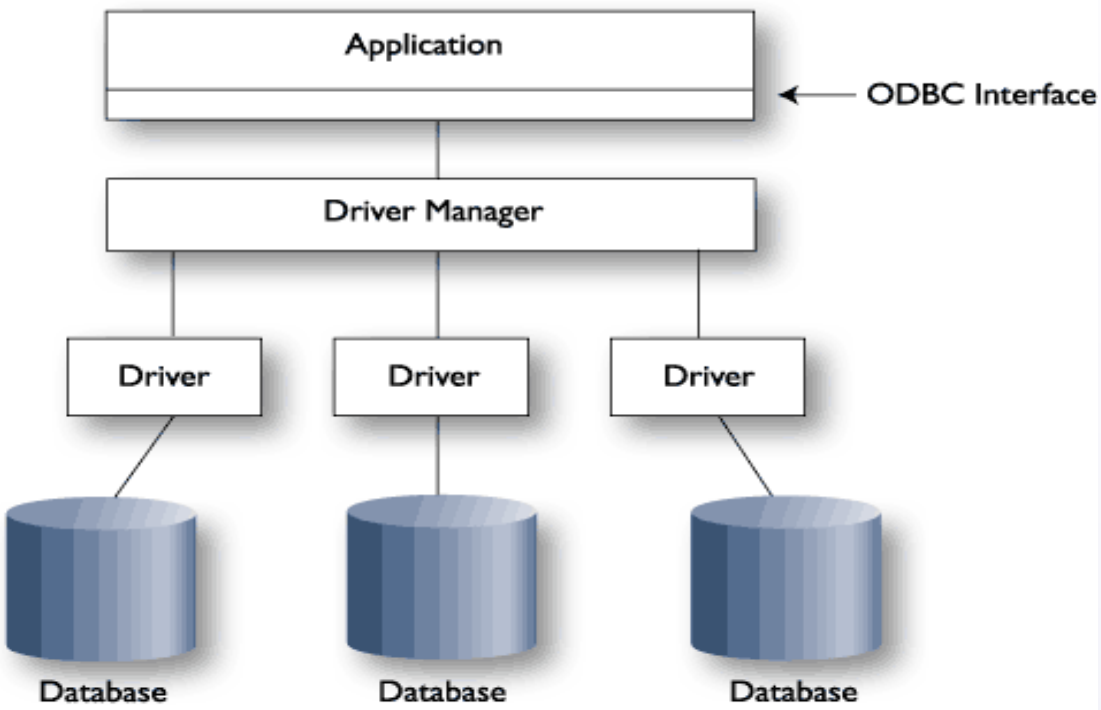


FIGURE 1: ODBC architecture

DAO

- Data access - DAO

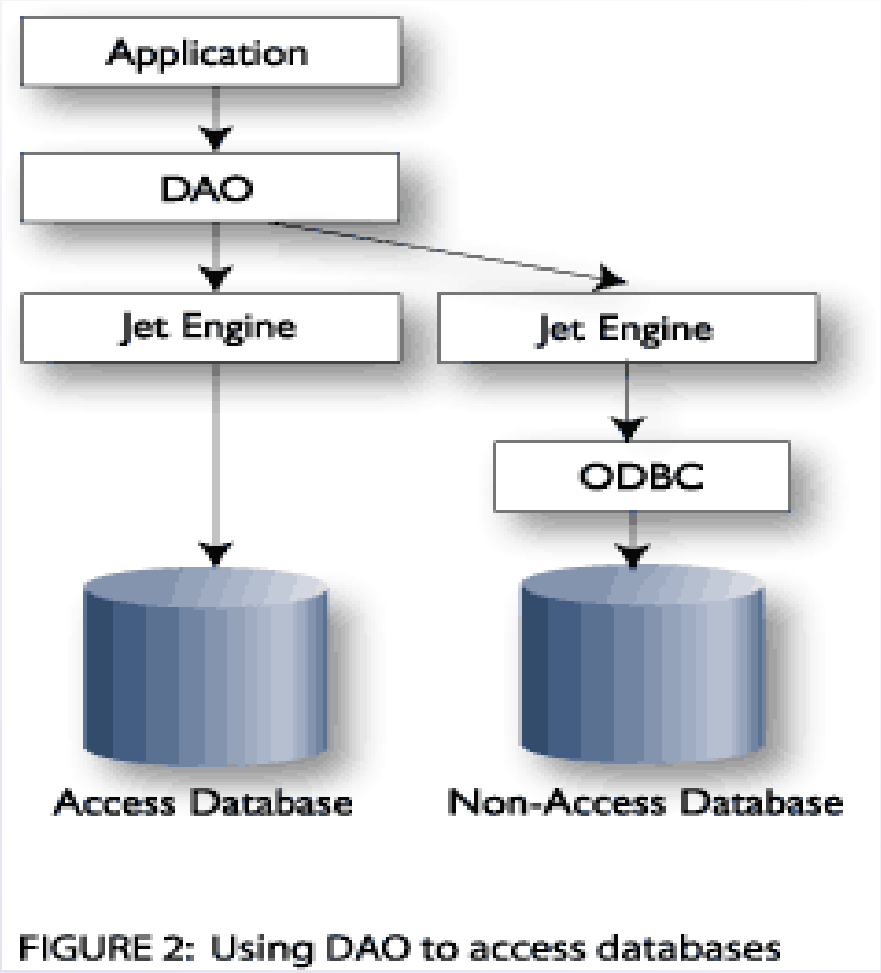


FIGURE 2: Using DAO to access databases

RDO

- Data access – RDO

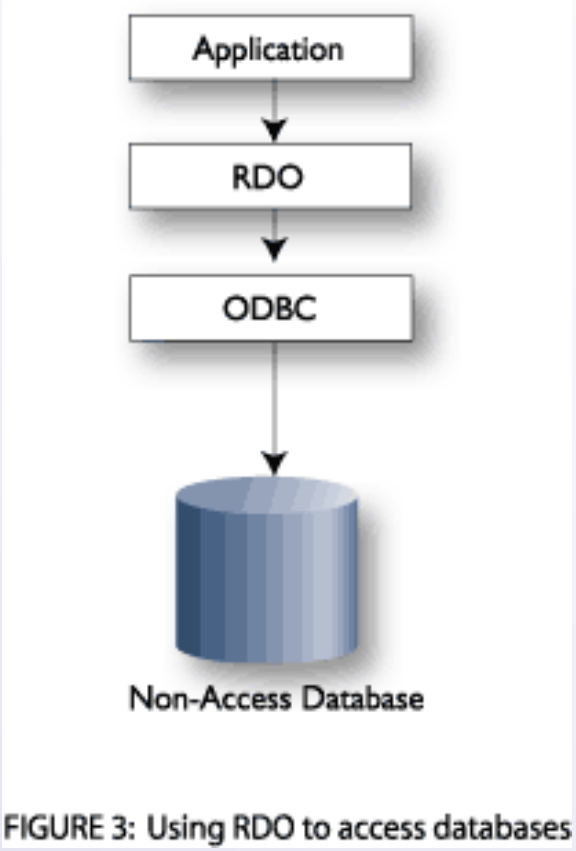


FIGURE 3: Using RDO to access databases

OLDEB

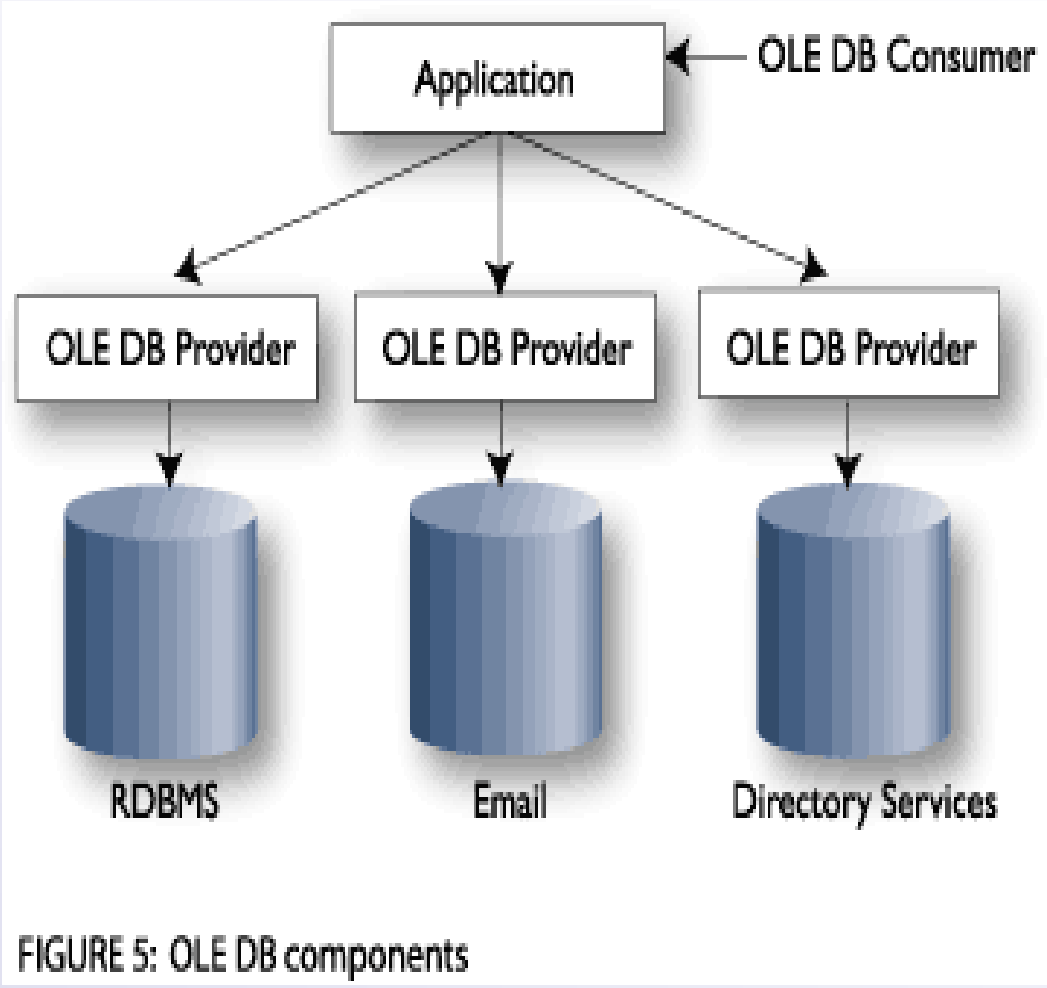


FIGURE 5: OLE DB components

ADO

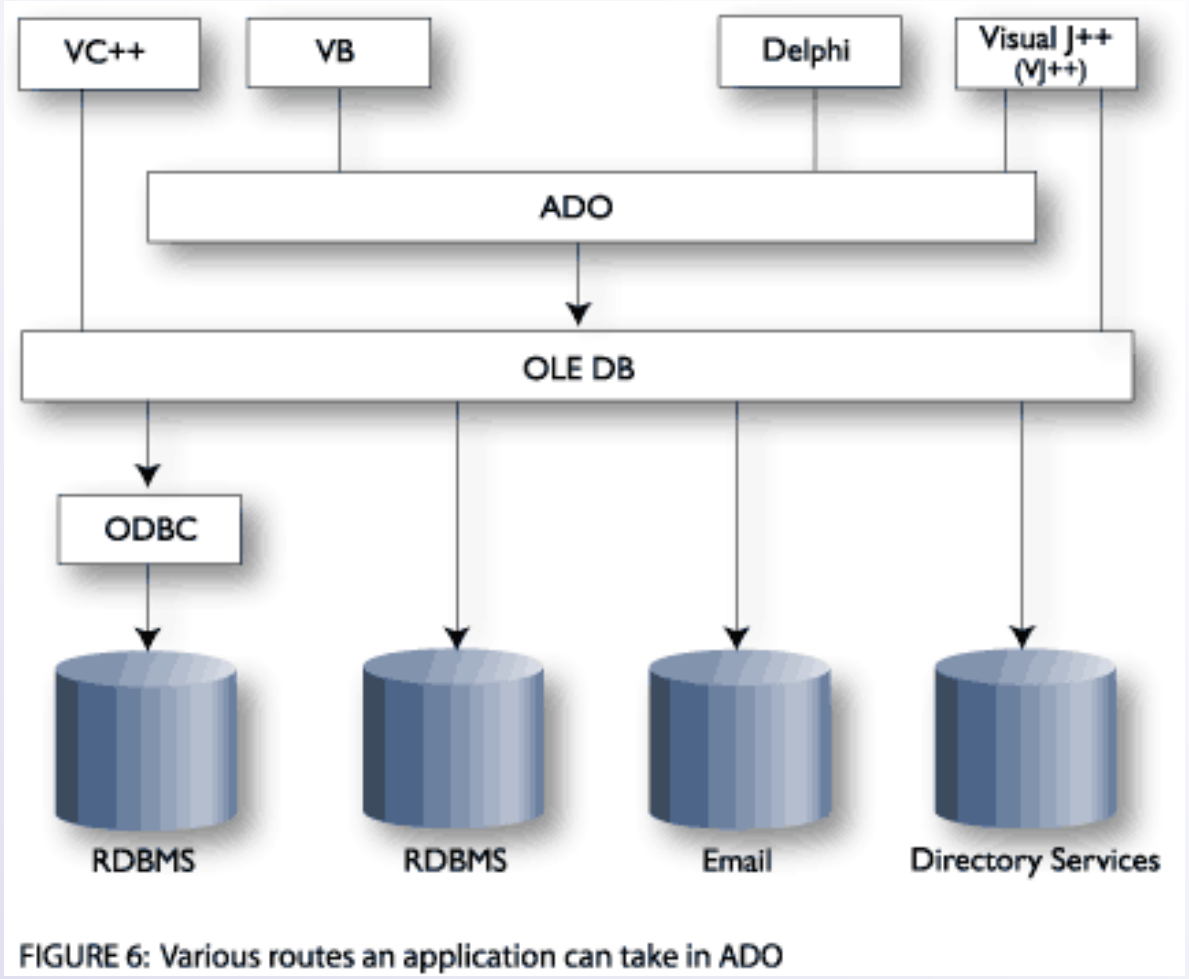
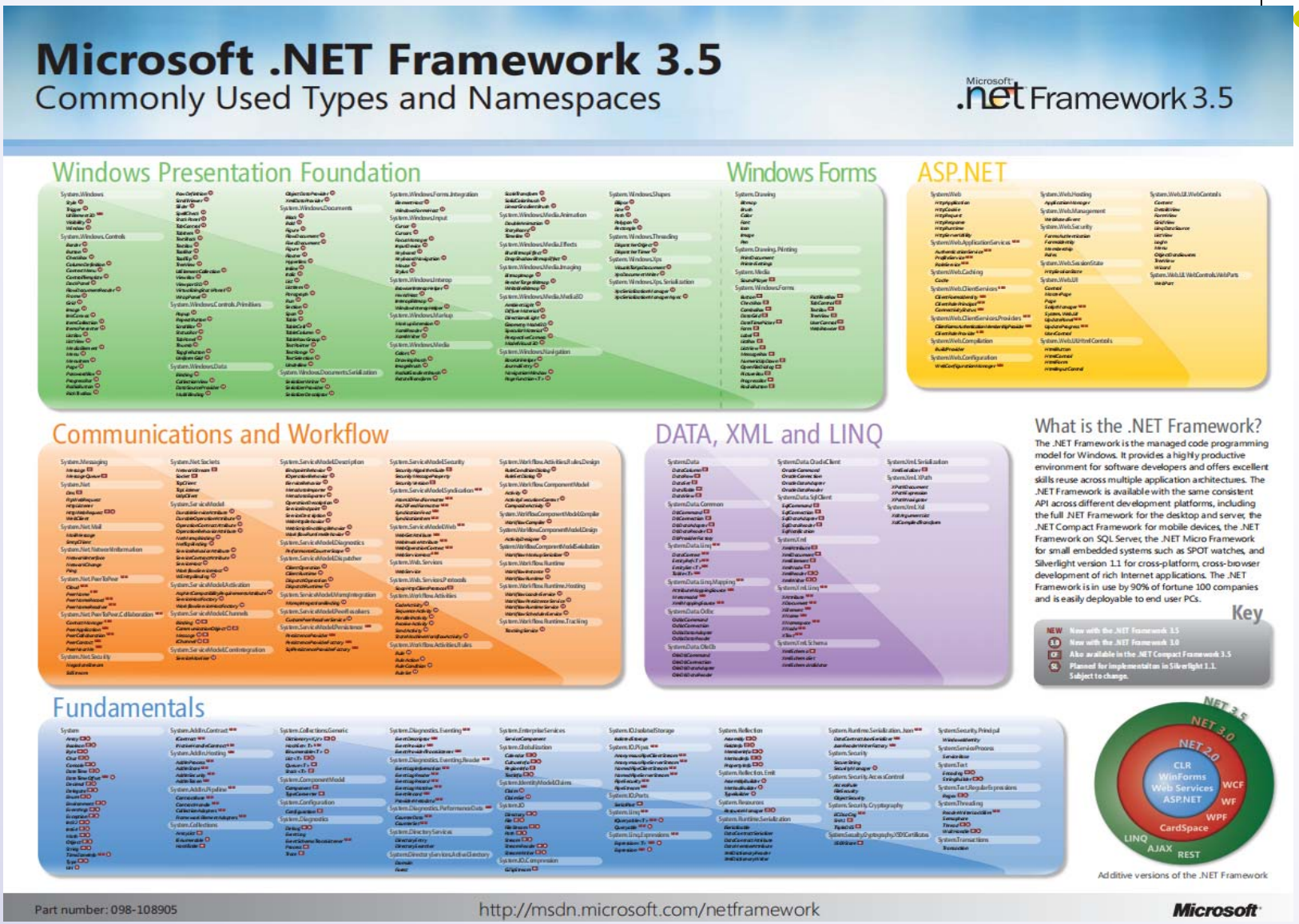
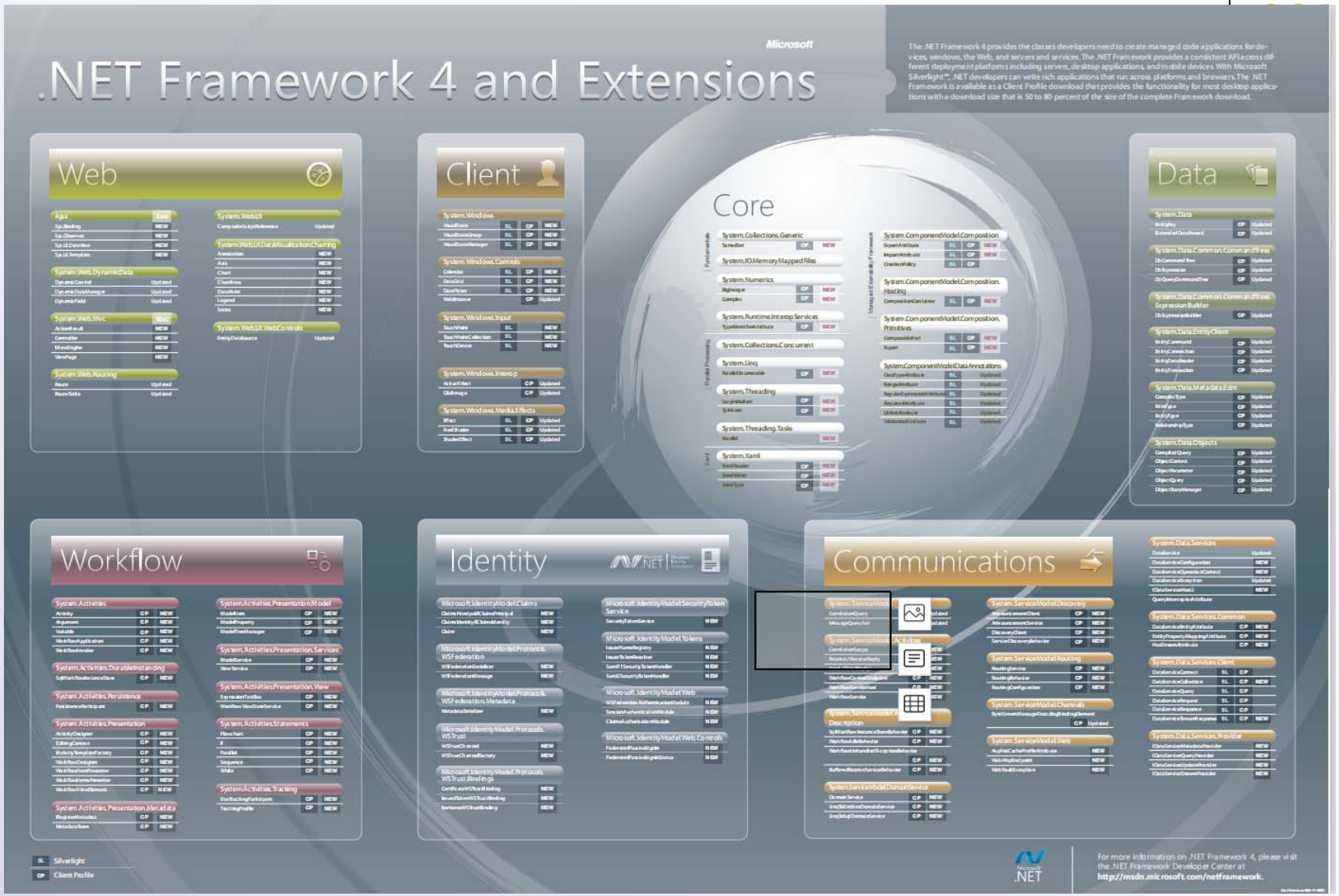


FIGURE 6: Various routes an application can take in ADO

ADO. NET

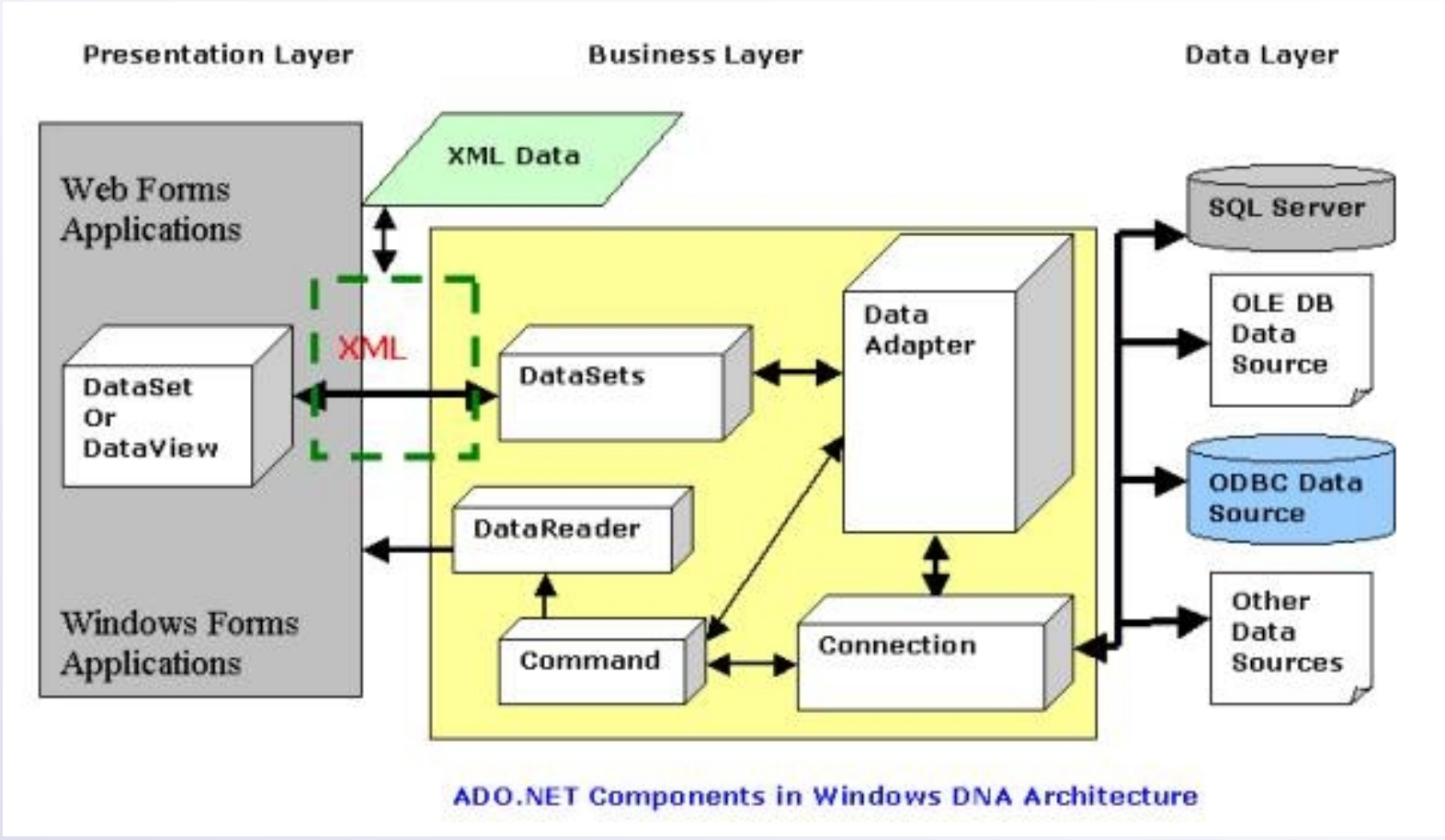

Microsoft .NET Framework 3.5
Commonly Used Types and Namespaces
Microsoft .net Framework 3.5
Windows Presentation Foundation
Windows Forms
ASP.NET
Communications and Workflow
DATA, XML and LINQ
What is the .NET Framework?
The .NET Framework is the managed code programming model for Windows. It provides a highly productive environment for software developers and offers excellent skills reuse across multiple application architectures. The .NET Framework is available with the same consistent API across different development platforms, including the full .NET Framework for the desktop and server, the .NET Compact Framework for mobile devices, the .NET Framework on SQL Server, the .NET Micro Framework for small embedded systems such as SPOT watches, and Silverlight version 1.1 for cross-platform, cross-browser development of rich Internet applications. The .NET Framework is in use by 90% of fortune 100 companies and is easily deployable to end user PCs.
Key
NEW Now with the .NET Framework 3.5
3.0 Now with the .NET Framework 3.0
CF Also available in the .NET Compact Framework 3.5
SL Planned for implementation in Silverlight 1.1. Subject to change.
Fundamentals
NET 3.5
NET 3.0
NET 2.0
CLR
WinForms
Web Services
ASP.NET
WCF
WF
WPF
CardSpace
LINQ
AJAX
REST
Additive versions of the .NET Framework
Part number: 098-108905
http://msdn.microsoft.com/netframework
Microsoft

ADO. NET


Microsoft
.NET Framework 4 and Extensions
The .NET Framework 4 provides the classes developers need to create managed code applications for devices, windows, the Web, and servers and services. The .NET Framework provides a consistent API across different deployment platforms including servers, desktop applications, and mobile devices. With Microsoft Silverlight™, .NET developers can write rich applications that run across platforms and browsers. The .NET Framework is available as a Client Profile download that provides the functionality for most desktop applications with a download size that is 50 to 80 percent of the size of the complete Framework download.
Web
Client
Core
Data
Workflow
Identity
Communications
Silverlight
Client Profile
For more information on .NET Framework 4, please visit the .NET Framework Developer Center at
http://msdn.microsoft.com/netframework.

ADO. NET

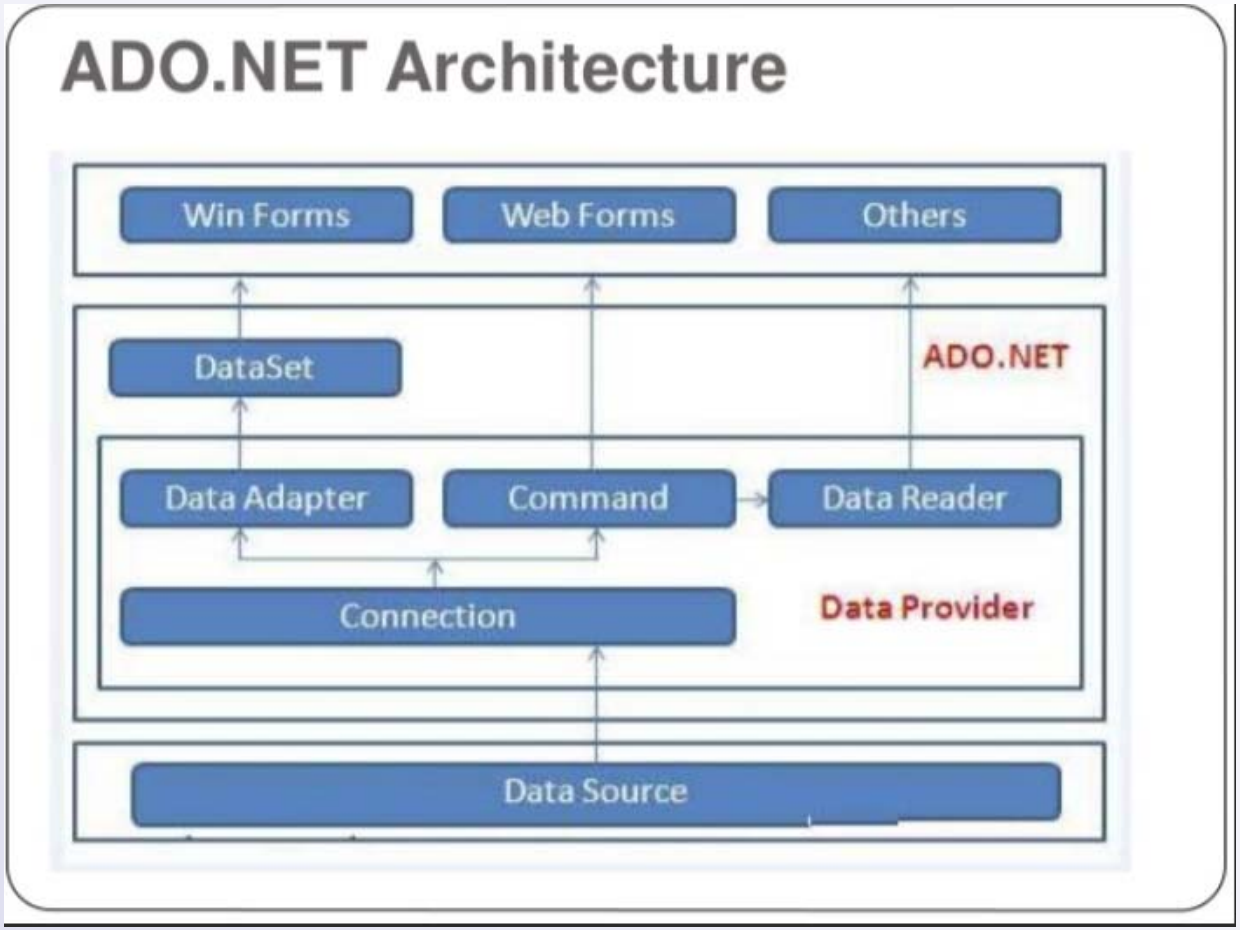
Architecture



ADO.NET Components in Windows DNA Architecture

ADO. NET

- Architecture



ADO.NET

ADO.NET Interface

- Interface
  - -- SQL Server
  - -- OLE DB
  - -- Npgsql (PostgreSQL)

- NameSpaces
  - --- System.Data
  - --- System.Data.Oledb
  - --- System.Data.SqlClient
  - --- Npgsql

ADO.NET – Connection

Connection– SQL Server

- //Namespace
  using System.Data;
  using System.Data.SqlClient;

- // Method1
  string connectionString =
  “server=myServer;uid=sa;pwd=;database=northwind”;

  SqlConnection conn = new SqlConnection (connectionString) ;

- //Method2
  SqlConnection conn = new SqlConnection () ;

  string connectionString =
  “server=myServer;uid=sa;pwd=;database=northwind”;
  conn.ConnectionString = connectionString

ADO.NET – Connection

Connection– PostgreSQL

- //Namespace
  using Npgsql;

- //Method1
  string connString =
  "Host=localhost;Username=postgres;Password=123456;Database=student";
  NpgsqlConnection conn = new NpgsqlConnection (connString);

- // Method2
  NpgsqlConnectionconn conn = new NpgsqlConnection () ;
  string connString =
  "Host=localhost;Username=postgres;Password=123456;Database=student";
  conn.ConnectionString = connString;

ADO.NET – Connection

Connection– Access

- //Namespace

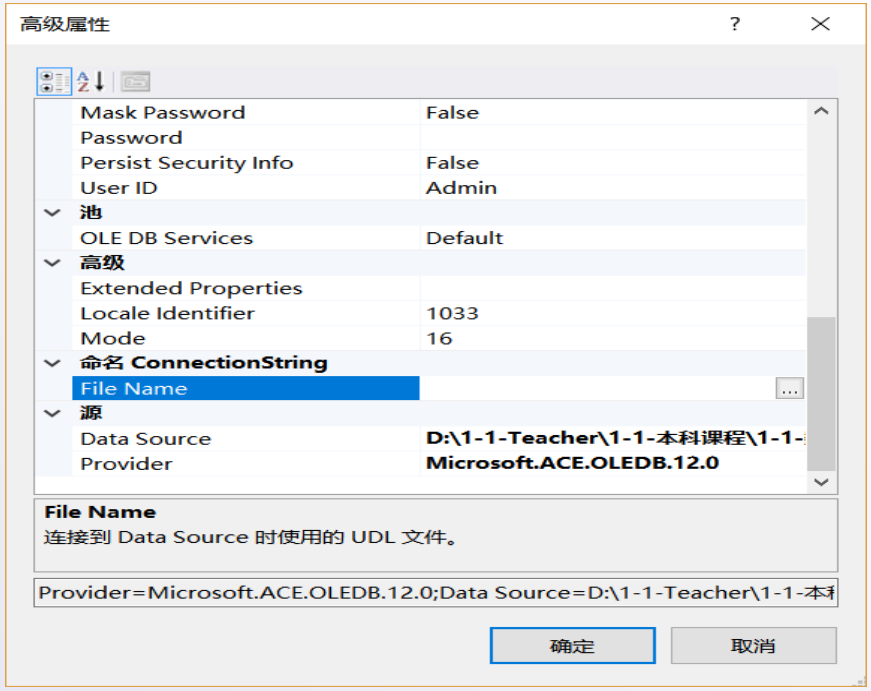
```
using System.Data;
using System.Data.OleDb;
```

- //New connection

```
string connectionString =
@"Provider=""Microsoft.Jet.OLEDB.4.0""; Data Source =
            F:\\DataBase\\employ.mdb";
oleDbConnection conn = new oleDbConnection(connectionString )
```

ADO.NET – Connection

- Connection– Access



Access connection string for different version:

Provider=""Microsoft.ACE.OLEDB.12.0"";...(office2010)

Provider=""Microsoft.ACE.OLEDB.15.0"";...(office2013)

Provider=""Microsoft.ACE.OLEDB.16.0"";...(office2016)

ADO.NET - Open/Close

- After Connection

1) Open database:

```
conn.Open()
```

2) Operation on data

3) Close database:

```
conn.Close()
```

ADO.NET – Command

Command – SQL Server

- Command: SQL execute。

- //Method1

```
SqlCommand cmd = new SqlCommand();
cmd.CommandType = CommandType.Text;
cmd.CommandText = "Select * from Customers";
cmd.Connection = conn;
```

- // Method2

```
SqlCommand cmd = new SqlCommand ("Select * from Customers",  conn);
```

- //execute SQL

```
Int32 recordsAffected = cmd.ExecuteNonQuery();
```

ADO.NET - Command

- Command – procedure (SQL Server)

```
sqlCmd.CommandType = CommandType.StoredProcedure;
sqlCmd.CommandText = "up_getsname";
//parameter
SqlParameter pNo = new SqlParameter( "@sno", '1001');
sqlCmd.Parameters.Add( paramNo ); //input parameter
SqlParameter paramName = new SqlParameter(); // output parameter
paramName.ParameterName = "@sname";
paramName.SqlDbType = SqlDbType.VarChar;
paramName.Size = 10;
paramName.Direction = ParameterDirection.Output;
sqlCmd.Parameters.Add( paramName );
//call procedure
sqlCmd.ExecuteNonQuery();
paramName.Value.ToString();
```

ADO.NET – Command

Command – PostgreSQL

- //Command: method1

```
NpgsqlCommand command = new NpgsqlCommand();
command.CommandText = “select * from s”;
command.Connection = conn;
```

- //command: method2

```
NpgsqlCommand cmd = new NpgsqlCommand(“select * from s”,  conn);
```

- //SQL excute

```
cmd.ExecuteNonQuery();
cmd.CommandText = "delete from s where sno='1001'";
cmd.ExecuteNonQuery();
```

ADO.NET – Command

Command – Access

- //Method1

```
OleDbCommand command = new OleDbCommand();
command.CommandText = “Select ContactName from Customers”;
command.Connection = conn;
```

- // Method2

```
OleDbCommand command = new OleDbCommand
        (“Select ContactName from Customers”,  conn);
```

ADO.NET - DataReader

- DataReader - SQL Server

Read data

```
SqlCommand command = new SqlCommand
        (“Select ContactName from Customers”,  conn);
conn.Open();
SqlDataReader reader = command.ExecuteReader()
string result;
while(reader.Read())
{
  result  = reader.GetString(0);
}
reader.Close();
conn.Close();
```

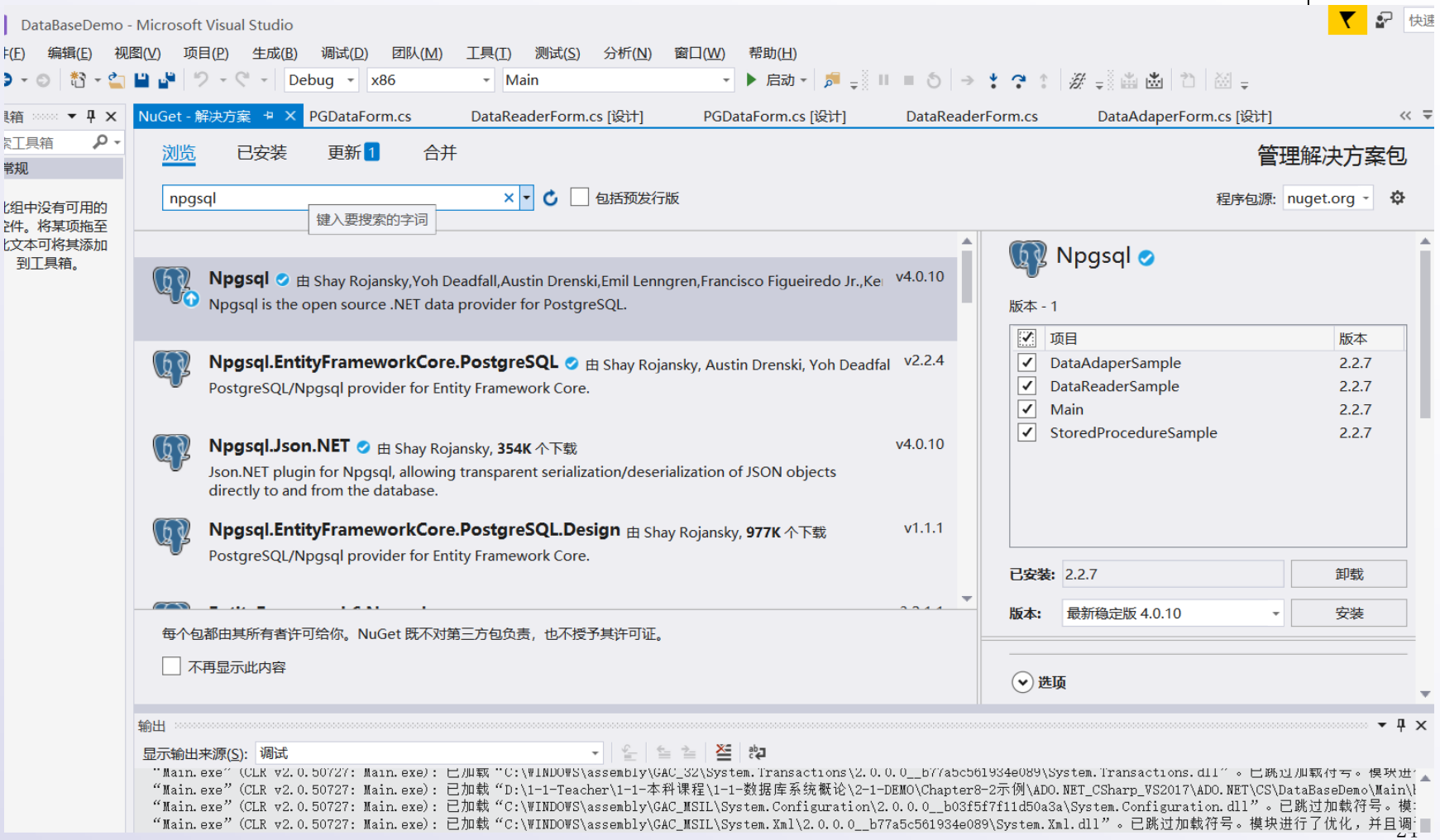
ADO.NET - DataReader

- DataReader - PostgreSQL

```
NpgsqlCommand command = new NpgsqlCommand
          (“select * from s”,  conn);
conn.Open();
NpgsqlDataReader reader = command.ExecuteReader()
string result;
while(reader.Read())
{
  result  = reader.GetString(0);
}
reader.Close();
conn.Close();
```

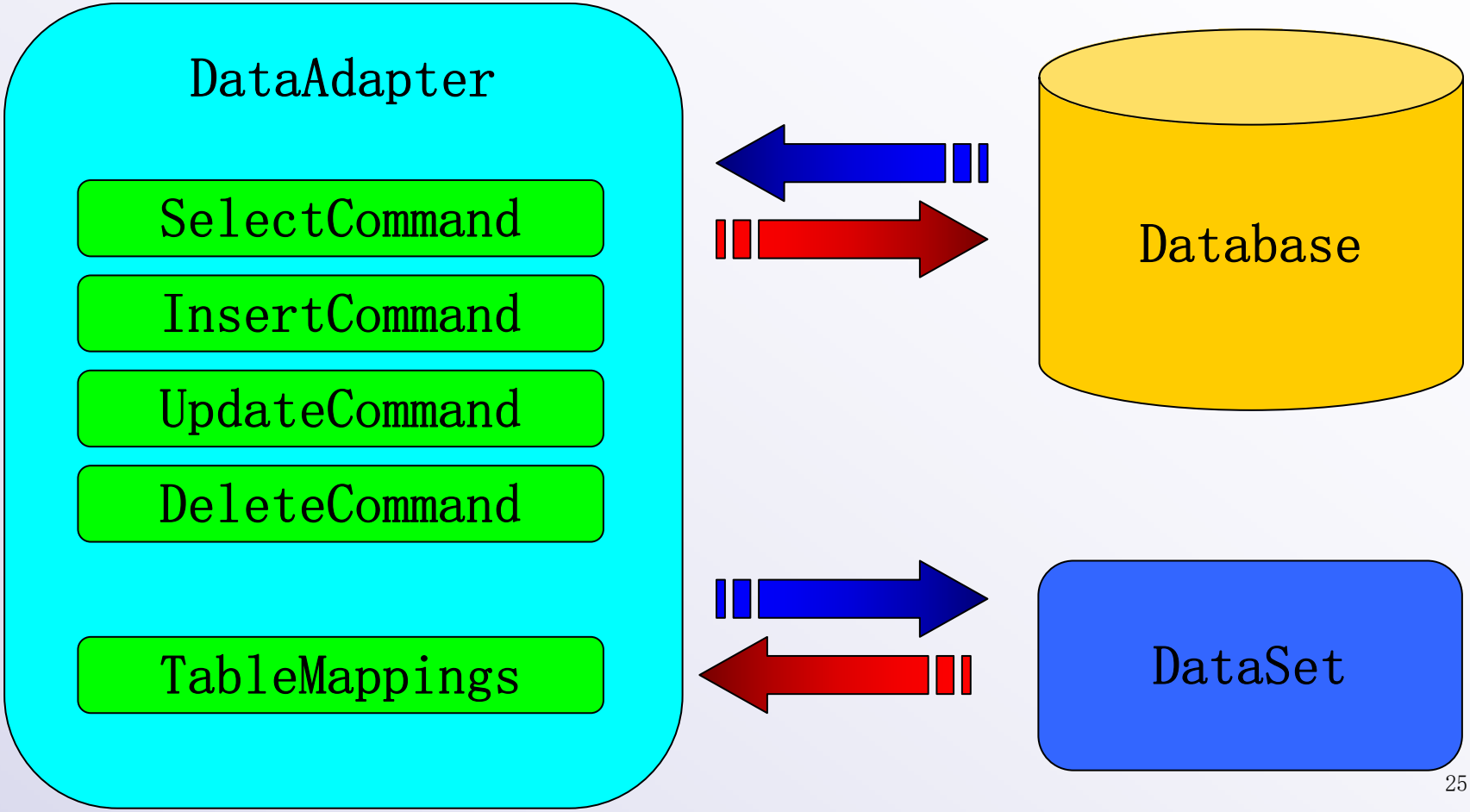
ADO.NET - DataReader

DataReader - PostgreSQL



ADO.NET - DataAdapter

- DataAdapter



ADO.NET - DataSet

DataSet

- DataSet: data in memory。
- DataSet:include several tables (similar with tables in db)。
- DataSet: Can build it according to requirement
- DataSet/Data Source: When connect database by DataAdapter, DataSet will disconnect from database after fetch data, any changes will be in memory before commit.

ADO.NET - DataAdapter

- DataAdapter and DataSet

	列名	数据类型	长度	允许空
🔑	CustomerID	nchar	5	
	CompanyName	nvarchar	40	
	ContactName	nvarchar	30	✓
	ContactTitle	nvarchar	30	✓
	Address	nvarchar	60	✓
	City	nvarchar	15	✓
	Region	nvarchar	15	✓
	PostalCode	nvarchar	10	✓
	Country	nvarchar	15	✓
	Phone	nvarchar	24	✓
	Fax	nvarchar	24	✓

ADO.NET - DataAdapter

DataAdapter and DataSet

```
SqlDataAdapter adapter = new SqlDataAdapter(
                    “select * from Customers” ,  conn);
DataSet dataSet = new DataSet();
Adapter.Fill(dataSet, "Customers");
string cName = dataSet.Tables["Customers"].Columns[0].ToString();
string name =
dataSet.Tables["Customers"].Rows[0]["CompanyName"].ToString();
dataSet.Tables["Customers"].Rows[0]["CompanyName"] = “co-think”
```

Fill (DataSet, Table)
Update (DataSet, Table)