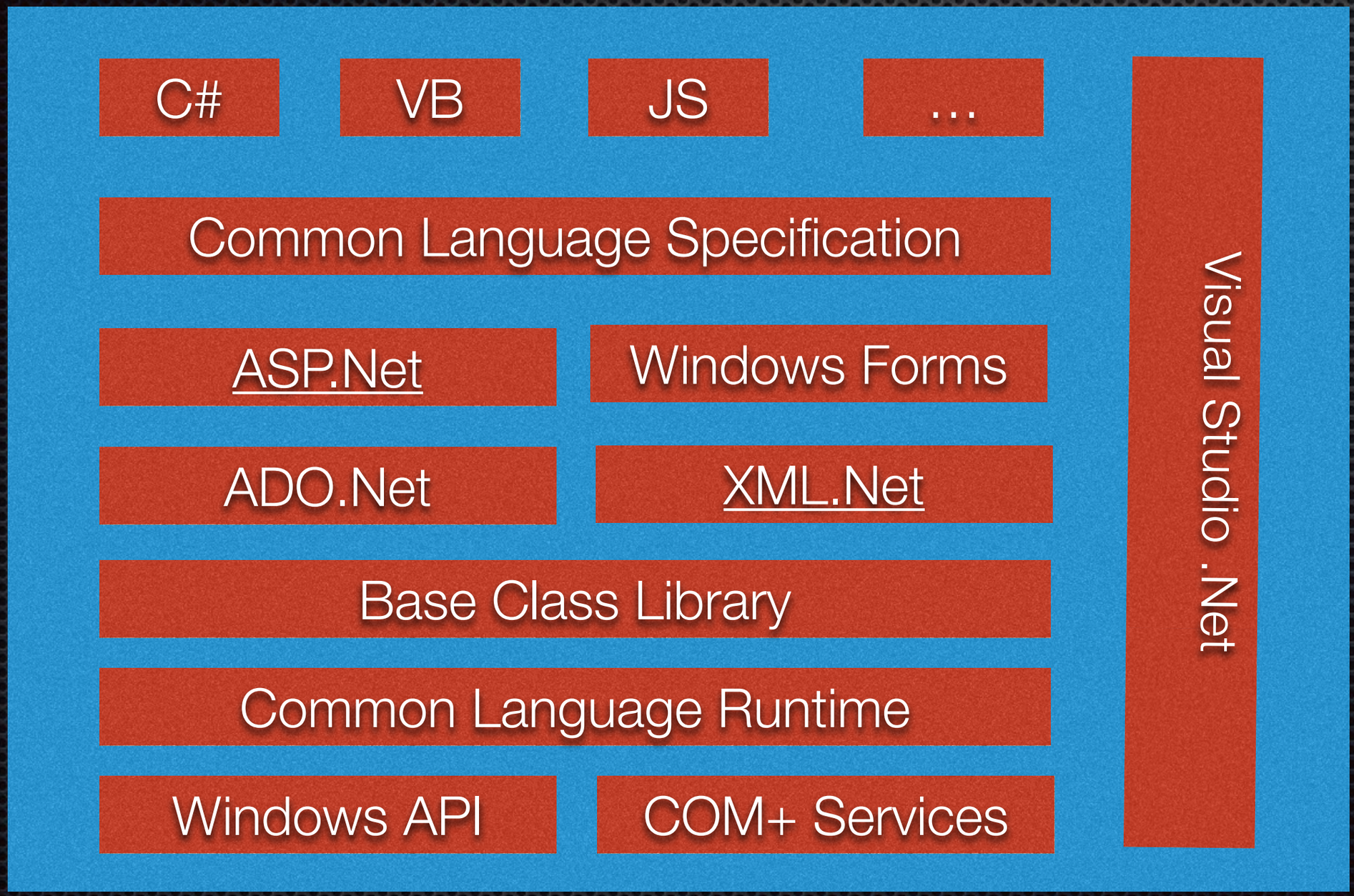
ADO.Net

The ADO.Net object structure
Connecting
Commanding
Readers and DataSets

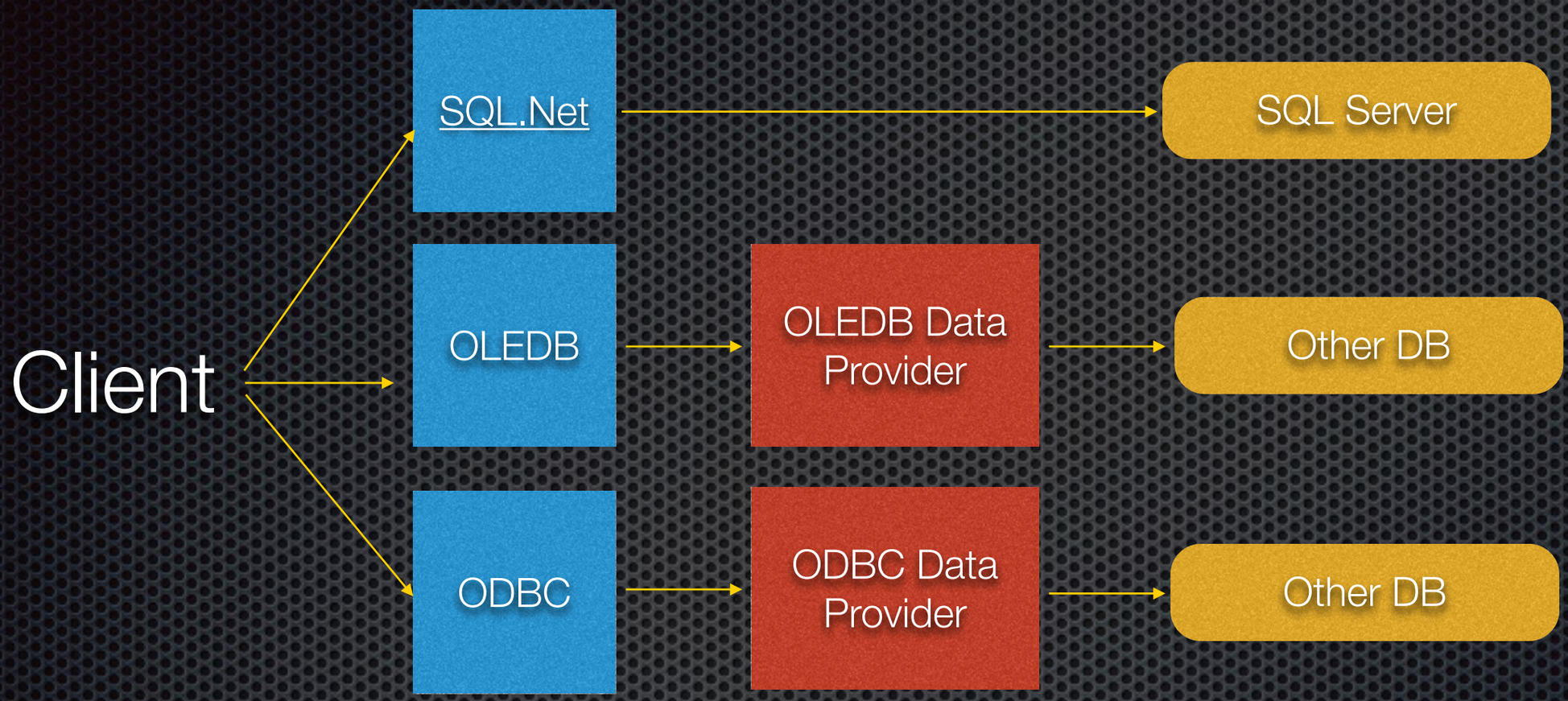
What is ADO.Net

- The data access classes for the .Net framework
- Designed for highly efficient data access
- Support for XML and disconnected record sets

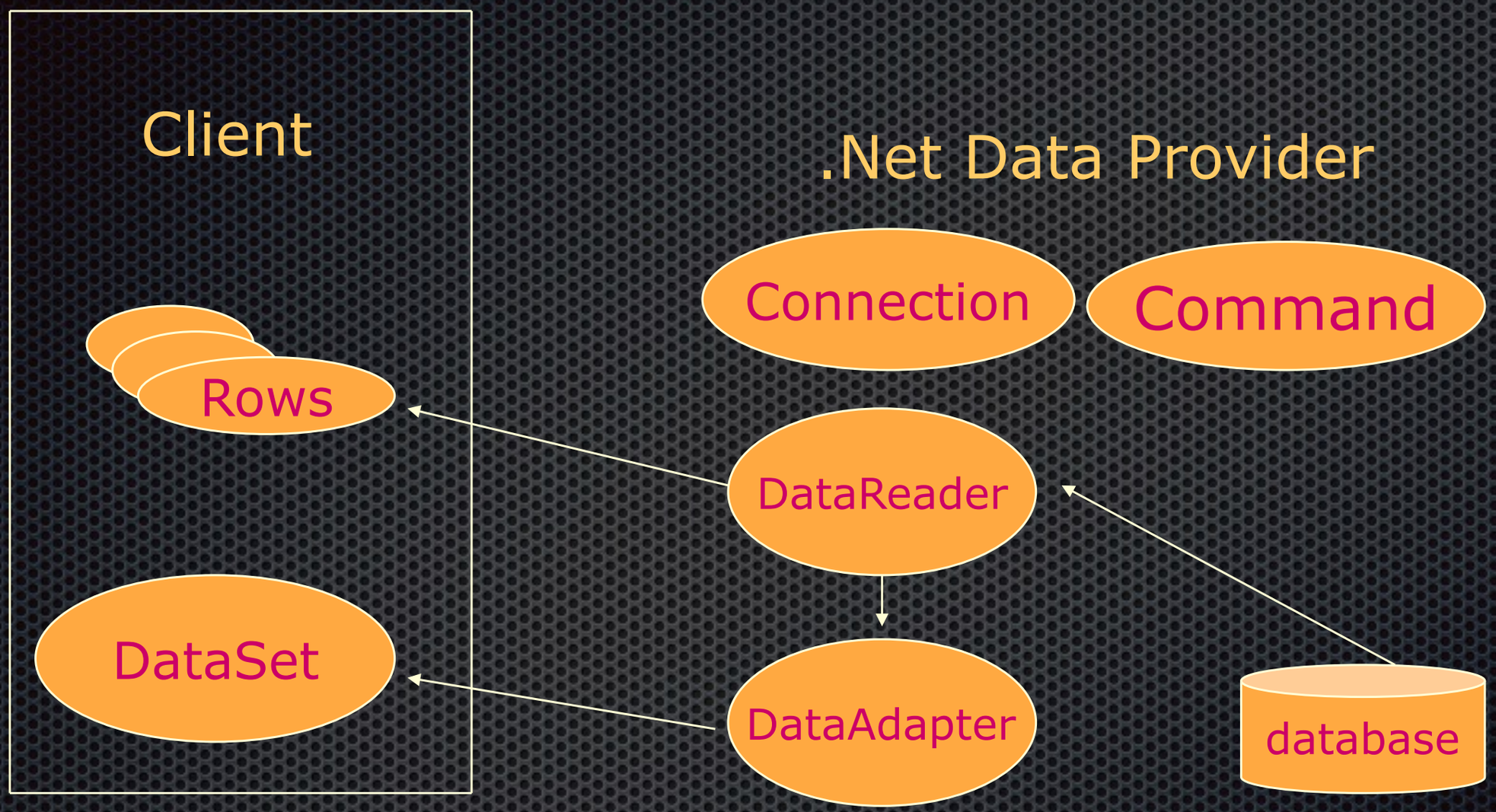
Where does ADO.Net Sit



.Net Data Providers



Data Provider Functionality



Namespaces

- System.Data & System.Data.Common
- System.Data.SqlClient & System.Data.OleDB
- System.Data.SqlTypes
- System.XML & System.XML.Schema

Namespaces

System.Data
System.Data.Common
System.Data.SqlClient
System.Data.OleDb

```
{
SqlConnection
SqlCommand
SqlDataReader
SqlDataAdapter
SqlParameter
SqlParameterCollection
SqlError
SqlErrorCollection
SqlException
SqlTransaction
SqlDbType
}
```

Using Data Adapter

```
using System.Data.SqlClient;

  string sConnectionString =
    "Initial Catalog=Northwind;
    Data Source=localhost;
    Integrated Security=SSPI;";

SqlDataAdapter sqlAdp= new
SqlDataAdapter(sConnectionString);

sqlAdp.Close();
sqlAdp.Dispose();
```

Connection Pooling

- ADO.Net pools connections.
  When you close a connection it is released back into a pool.

```
SqlConnection conn = new SqlConnection();
  conn.ConnectionString =
    "Integrated Security=SSPI;Initial Catalog=northwind";
  conn.Open();  // Pool A is created.
```

```
SqlConnection conn = new SqlConnection();
  conn.ConnectionString =
    "Integrated Security=SSPI;Initial Catalog=pubs";
  conn.Open();
  // Pool B is created because the connection strings differ.

SqlConnection conn = new SqlConnection();
  conn.ConnectionString =
    "Integrated Security=SSPI;Initial Catalog=northwind";
  conn.Open(); // The connection string matches pool A.
```

Demo..

```
string sSelectQuery =
     "SELECT * FROM Categories ORDER BY CategoryID";
   string sConnectionString =  "Initial Catalog=Northwind;
      Data Source=localhost;   Integrated Security=SSPI;";

   SqlConnection objConnect = new SqlConnection(sConnectString);

   SqlCommand objCommand = new   SqlCommand(sSelectQuery,
   objConnect);
   /*
objCommand.CommandTimeout = 15;
   objCommand.CommandType = CommandType.Text;
*/

   objConnect.Open();

   SqlDataReader drResults;
   drResults = objCommand.ExecuteReader()

   drResults.Close();
   objConnect.Dispose();
```

Command Methods

.ExecuteReader() - Returns DataReader

.ExecuteNonQuery() - **Returns # of Rows Affected**

.ExecuteXMLReader() - Returns XMLReader Object to Read XML documentation

.ExecuteScaler() - Returns a Single Value e.g. SQL SUM function.

DataReader object

DataReader objects are highly optimized for fast, forward only enumeration of data from a data command

A DataReader is **not** disconnected.

Access to data is on a per record basis.

Forward only

Read only

Does support multiple record sets

Data Reader Demo..

```
SqlDataReader sqlReader;
sqlReader = sqlCommand.ExecuteReader();
while (sqlReader.Read())
{
  // process, sqlReader("field")
}
sqlReader.Dispose();
```

DataSets

In-memory representation of data contained in a database/XML

Operations are performed on the DataSet, not the data source

A DataSet contains one or more DataTables.

Fields are held within the DataTable.And in DataRows, DataColumns.

Setup SqlConnection
Setup a SqlDataAdapter
Create a DataSet
Call the .Fill() method on the DA

Data Adapter

Pipeline between DataSets and data sources

Geared towards functionality rather than speed

Disconnected by design

Supports select, insert, delete, update commands and methods

Must always specify a select command

All other commands can be generated or specified

Data Adapter Demo..

```
SQLDataAdapter sqlDA =
  new SqlDataAdapter();
sqlDA.SelectCommand =
  new SqlCommand ("select * from authors“,
  sqlConnection);

DataSet sqlDS = new DataSet("authorsTable");
sqlDA.Fill(sqlDS, "authorsTable");
```

?