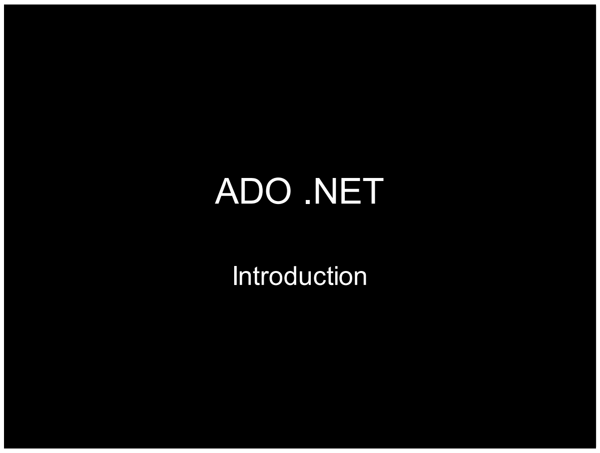
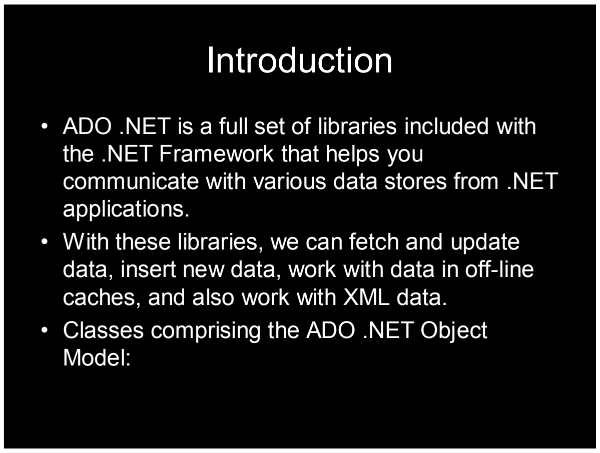


ADO .NET

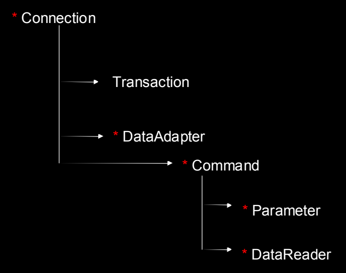
Introduction



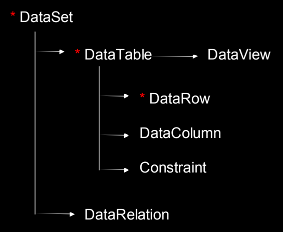
Introduction

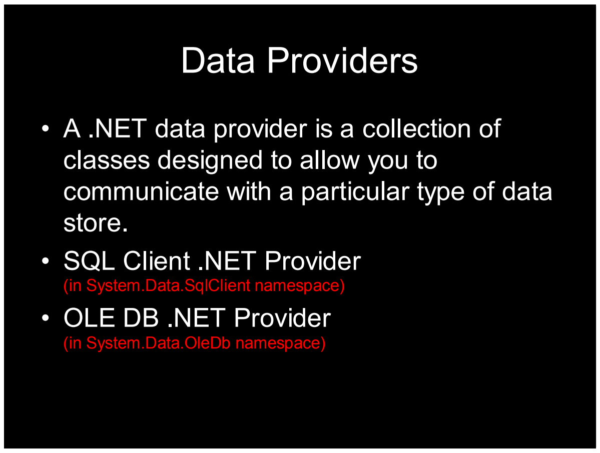
- ADO .NET is a full set of libraries included with the .NET Framework that helps you communicate with various data stores from .NET applications.
- With these libraries, we can fetch and update data, insert new data, work with data in off-line caches, and also work with XML data.
- Classes comprising the ADO .NET Object Model:

Connected Objects



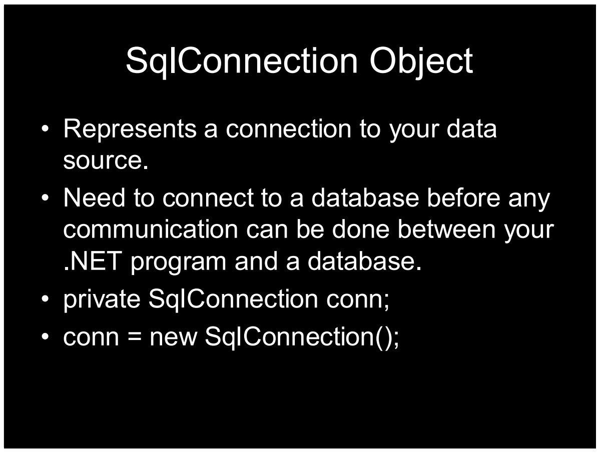
Disconnected Objects





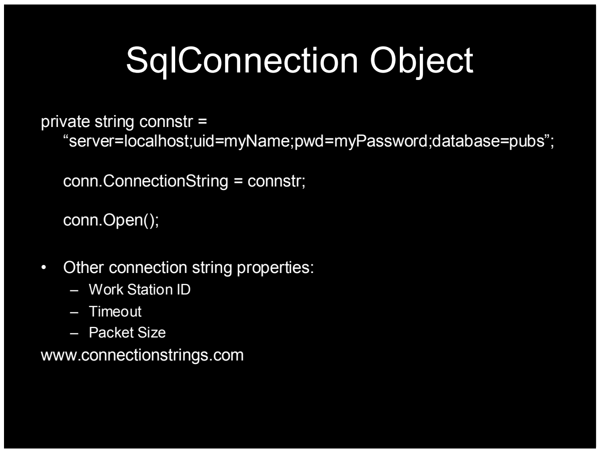
Data Providers

- A .NET data provider is a collection of classes designed to allow you to communicate with a particular type of data store.
- SQL Client .NET Provider
  (in System.Data.SqlClient namespace)
- OLE DB .NET Provider
  (in System.Data.OleDb namespace)



SqlConnection Object

- Represents a connection to your data source.
- Need to connect to a database before any communication can be done between your .NET program and a database.
- private SqlConnection conn;
- conn = new SqlConnection();



SqlConnection Object

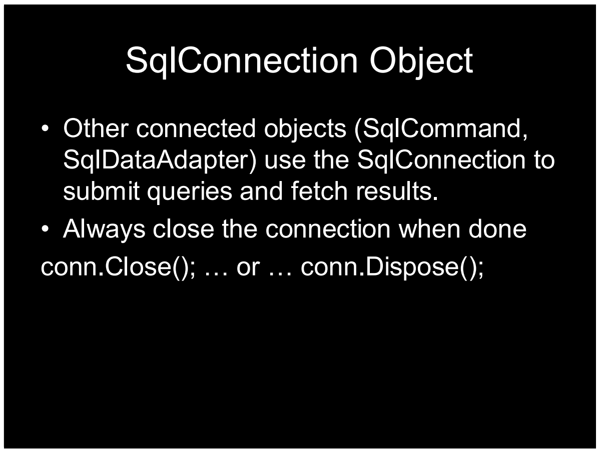
```
private string connstr =
    “server=localhost;uid=myName;pwd=myPassword;database=pubs”;

    conn.ConnectionString = connstr;

    conn.Open();
```

- Other connection string properties:
  - Work Station ID
  - Timeout
  - Packet Size

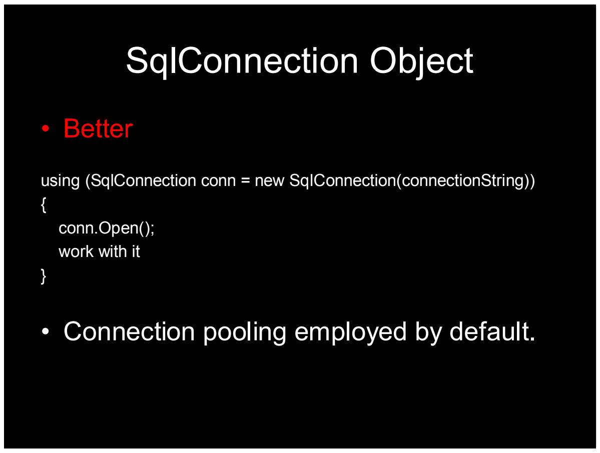
www.connectionstrings.com



SqlConnection Object

- Other connected objects (SqlCommand, SqlDataAdapter) use the SqlConnection to submit queries and fetch results.
- Always close the connection when done

conn.Close(); … or … conn.Dispose();

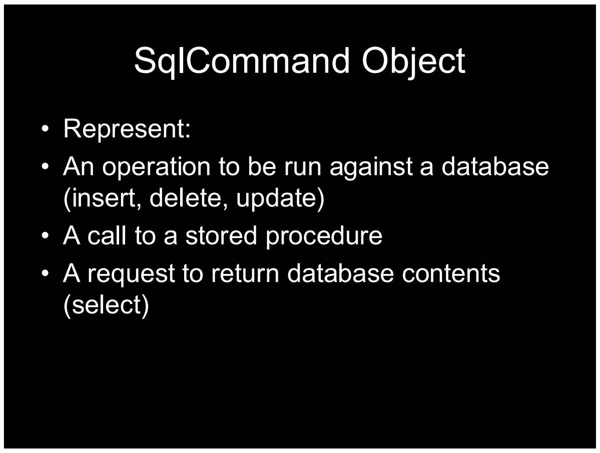


SqlConnection Object

- Better

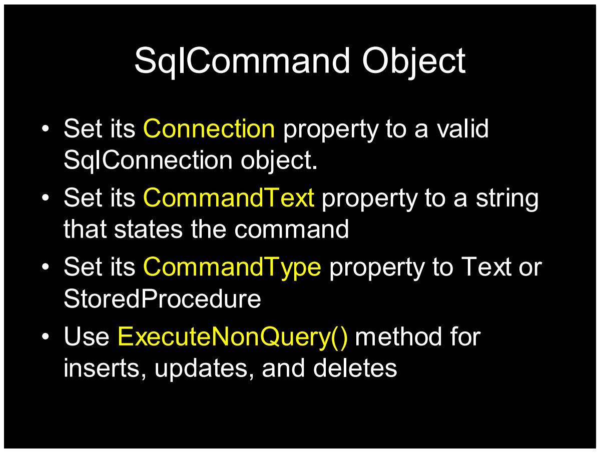
```
using (SqlConnection conn = new SqlConnection(connectionString))
{
   conn.Open();
   work with it
}
```

- Connection pooling employed by default.



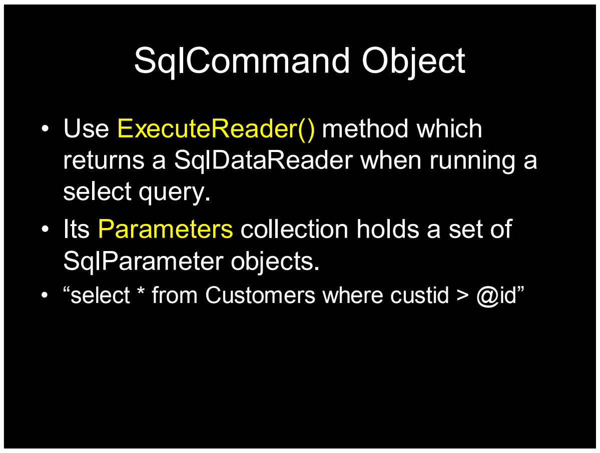
SqlCommand Object

- Represent:
- An operation to be run against a database (insert, delete, update)
- A call to a stored procedure
- A request to return database contents (select)



SqlCommand Object

- Set its Connection property to a valid SqlConnection object.
- Set its CommandText property to a string that states the command
- Set its CommandType property to Text or StoredProcedure
- Use ExecuteNonQuery() method for inserts, updates, and deletes



SqlCommand Object

- Use ExecuteReader() method which returns a SqlDataReader when running a select query.
- Its Parameters collection holds a set of SqlParameter objects.
- “select * from Customers where custid > @id”

SqlDataReader Object

- Designed to let you retrieve and examine the rows returned by a select query as quickly as possible.
- Does not support updating data.
- Operates in a forward moving, read only capability.
- Very light weight and fast.
- Always Close() the reader when done.

SqlTransaction Object

- Let you group a number of changes to your database into one atomic transaction.
- They either all succeed or all fail.
- SqlConnection has a BeginTransaction() method.
- Use the Transaction to then commit or cancel.

SqlParameter Object

- Let you use wildcards in your sql queries.

"Select * from Customers where CustID = @parmname"

- You create a parameter object for each parameter, assign a value, and add them to the SqlCommand Parameters collection.
- Can also use with stored procedures to pass values in and get values out.

SqlDataAdapter Object

- Very Important.
- Bridge between database and disconnected objects.
- Its Fill() method transfers data from a database into a (in memory) DataSet or a DataTable.
- Its Update() method takes pending changes from a (in memory) DataSet and updates the physical database.

SqlDataAdapter Object

- Has properties that let you attach SqlCommand objects to Select, Insert, Update, or Delete data.
- You can use these SqlCommand objects to call on stored procedures in your database.

DataSet Object

- Most important disconnected object.
- Can represent a rich subset of an entire database that is cached on your machine without having a continuous connection to the database.
- Can represent a set of tables with all the metadata necessary to represent relationships and constraints among the tables.

DataSet Objects

- It is comprised of DataTable and DataRelation objects.
- These are accessed as properties of the DataSet object.

DataTable

- Lets you examine data through collections of Rows and Columns.
- You store the results of a query in a DataTable through the SqlDataAdapter’s Fill method.
- SqlDataAdapter daCust = new SqlDataAdapter(sqlstr, sqlConn);
- DataTable tblCust = new DataTable();
- daCust.Fill(tblCust);

DataTable

- Data in tblCust is now disconnected from server.
- Can examine it without any more network traffic between it and server.
- Other properties of interest in a DataTable are *Rows*, *DataColumns*, and *Constraints*.

DataColumn

- Part of a DataTable’s DataColumns collection.
- Corresponds to a column in the data and has information about the structure of a column (metadata) including *Type* property, *ReadOnly*, *AllowDBNull*, *Unique*, *Default*, and *AutoIncrement*.

DataRow

- To get the actual values stored in a DataTable object, you use its *Rows* collection, which contains a series of DataRow objects.
- You use the *Item* property of the appropriate DataRow to read the value for any column.
- Exist several overloaded definitions of the Item property.

DataRow

- DataRow row;
- row = MyTable.Rows[0];
- String id = row[0] row[“CustomerID”]
- foreach (DataRow row in MyTbl.Rows)
-   Console.WriteLine(row[0]);