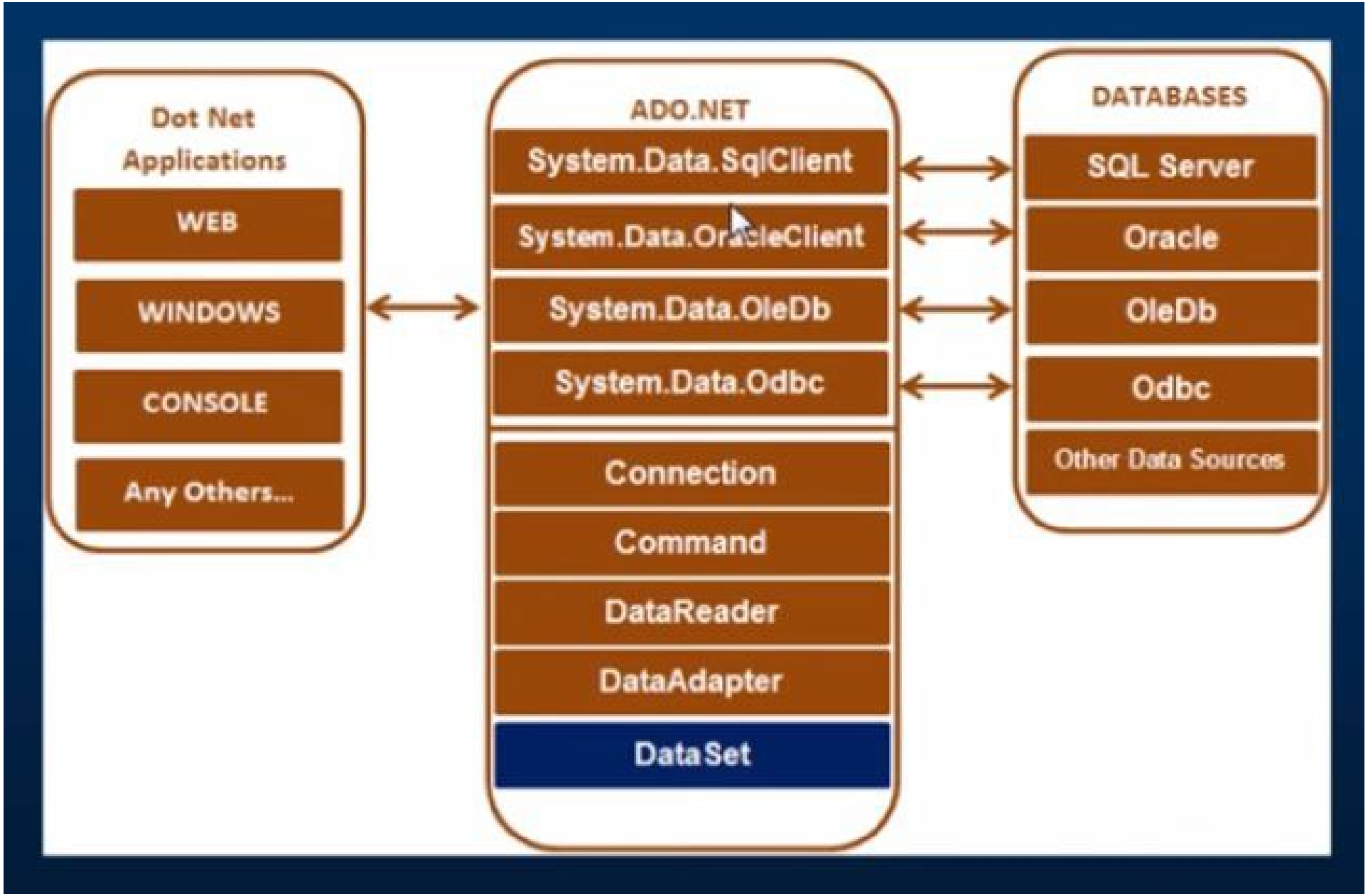
ADO.NET

ADO.NET Architecture



Direct Data Access and Disconnected Data Access

- When you query data with direct data access, you don't keep a copy of the information in memory. Instead, you work with it for a brief period of time while the database connection is open, and then close the connection as soon as possible.
- In disconnected data access, we keep a copy of the data in the DataSet object and work with it even after the database connection has been closed.

- using System.Data;
- using System.Data.SqlClient;

Direct Data Access

To query information

1. Create Connection, Command, and DataReader objects.

2. Use the DataReader to retrieve information from the database, and display it in a control on a web form.

3. Close your connection.

4. Send the page to the user. At this point, the information your user sees and the information in the database no longer have any connection, and all the ADO.NETobjects have been destroyed.

To add or update information, follow these steps:

1. Create new Connection and Command objects.

2. Execute the Command (with the appropriate SQL statement).

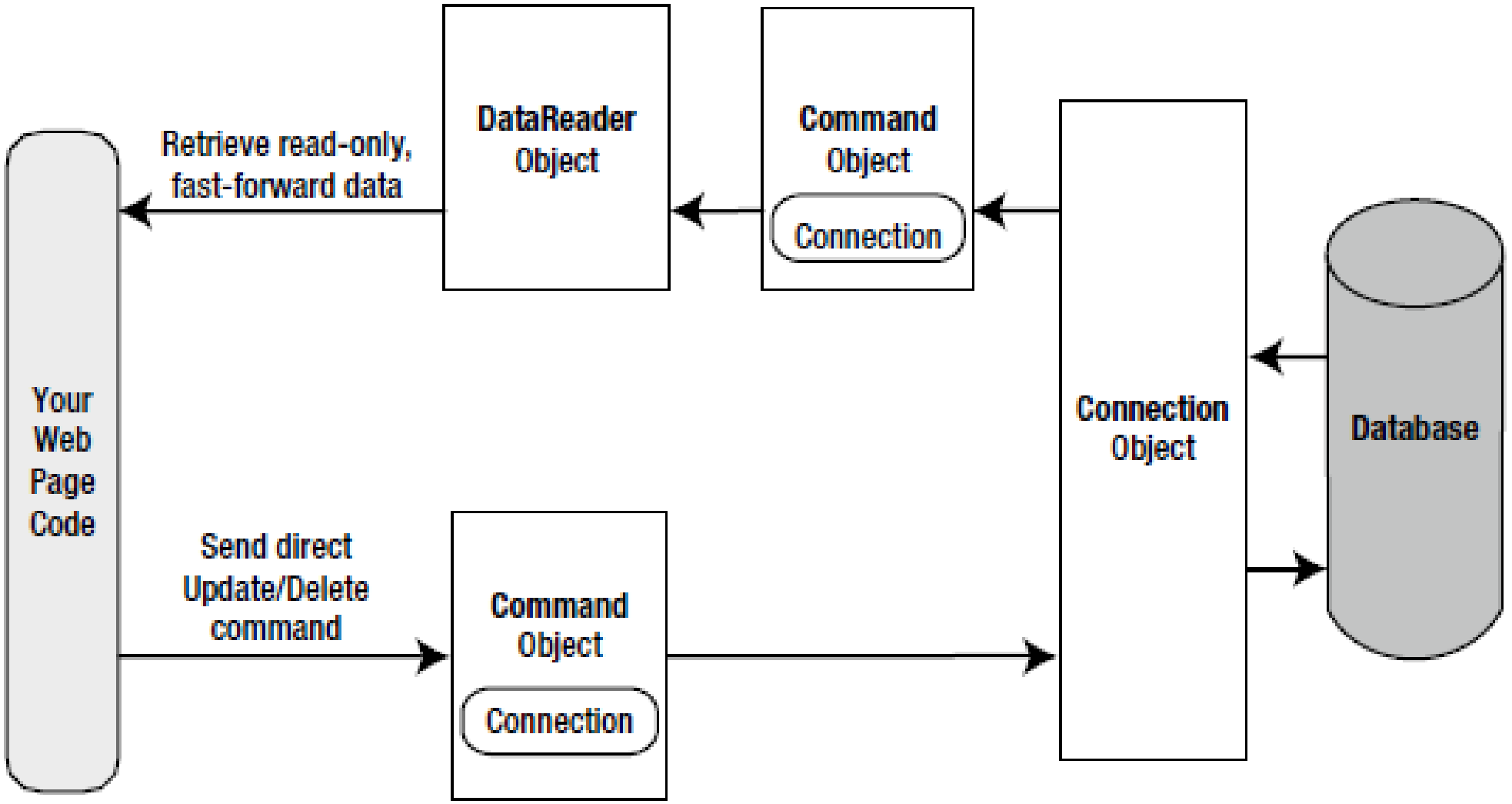


Figure 14-8. Direct data access with ADO.NET

Disconnected Data Access

When you use *disconnected data access*, you use the DataSet to keep a copy of your data in memory. You connect to the database just long enough to fetch your data and dump it into the DataSet, and then you disconnect immediately.

There are a variety of good reasons to use the DataSet to hold onto data in memory. Here are a few:

- You need to do something time-consuming with the data. By dumping it into a DataSet first, you ensure that the database connection is kept open for as little time as possible.
- You want to use ASP.NET data binding to fill a web control (such as a GridView) with your data. Although you can use the DataReader, it won't work in all scenarios. The DataSet approach is more straightforward.
- You want to navigate backward and forward through your data while you're processing it. This isn't possible with the DataReader, which goes in one direction only—forward.
- You want to navigate from one table to another. Using the DataSet, you can store several tables of information. You can even define relationships that allow you to browse through them more efficiently.
- Caching with the DataSet

SqlConnection

- SqlConnection in ADO.NET represents a connection to a SQL Server database
- *Data Provider for SQL Server (System.Data.SqlClient).*
- *Data Provider for MS ACCESS (System.Data.OleDb).*
- *Data Provider for MYSQL (System.Data.Odbc).*
- *Data Provider for ORACLE (System.Data.OracleClient).*

Connection String Parameters

- Data Source: This identifies the Server name, which could be the local machine, machine domain name or IP address
- Initial Catalog: This identifies the database name.
- Integrated Security: *True* for Windows authentication, *False* for Sql Server Authentication
- User Id: Name of the user configured in SQL Server.
- Password: Password matching SQL Server User ID.

Properties of Connection Object

Property	Description
Command Timeout	By Command time out, we can get or set number of seconds to wait, while attempting to execute a command.
Connection Timeout	By Connection time out, we can get or set number of seconds to wait for the connection to open.
Connection String	Connection string is used to establish and create connection to data source by using server name, database name, user id and password.
Mode	By mode property, we can check provider access permission.
Provider	By this property, we can get or set provider name.
State	By this property, we can check your current connection open or close before connection opening or closing
Version	This returns the ADO version number.

Command Class

- **ExecuteReader:** Returns data to the client as rows. This would typically be an SQL select statement or a Stored Procedure that contains one or more select statements. This method returns a DataReader object that can be used to fill a DataTable object or used directly for printing reports and so forth.

- **ExecuteNonQuery:** Executes a command that changes the data in the database, such as an update, delete, or insert statement, or a Stored Procedure that contains one or more of these statements. This method returns an integer that is the number of rows affected by the query.

- **ExecuteScalar:** This method only returns a single value. This kind of query returns a count of rows or a calculated value.

DataReader

- DataReader object provides
- A read only
- A forward only
- High performance mechanism
- to retrieve data from a data store as a data stream, while staying connected with the data source.

DataAdapter Class

- To connect DataSets to databases.
- Useful when using data-bound controls in Windows Forms
- Easy way to manage the connection between your application and the underlying database tables

DataSet Class

- Collection of DataTable objects
- Each object contains a collection of DataColumn and DataRow objects.
- The DataSet also contains a Relations collection that can be used to define relations among Data Table Objects.

Insert

```
protected void Button2_Click(object sender, EventArgs e)
    {
        string connString;
        SqlConnection cnn;
        connString = " Data Source=MUM0219CPU0072\\SQLEXPRESS;Initial Catalog=Student;Integrated Security=True";
        cnn = new SqlConnection(connString);
        cnn.Open();
        SqlCommand command;
        SqlDataAdapter adapter = new SqlDataAdapter();
        String sql = "";
        String newsap = TextBox1.Text;
        String newname = TextBox2.Text;
        String newstream = TextBox3.Text;
        sql = "Insert into Student values('" + newsap + "', '" + newname + "','" + newstream + "')";
        command = new SqlCommand(sql, cnn);
        adapter.InsertCommand = new SqlCommand(sql, cnn);
        adapter.InsertCommand.ExecuteNonQuery();
        command.Dispose();
        cnn.Close();
    }
```

View (Connected)

```
protected void Button3_Click(object sender, EventArgs e)
    {
        string connString;
        SqlConnection cnn;
        connString = " Data Source=MUM0219CPU0072\\SQLEXPRESS;Initial Catalog=Student;Integrated Security=True";
        cnn = new SqlConnection(connString);
        cnn.Open();
        SqlCommand command;
        SqlDataReader dataReader;
        String sql, Output = " ";
        sql = "Select * from Student";
        command = new SqlCommand(sql, cnn);
        dataReader = command.ExecuteReader();
        while (dataReader.Read())
        {
            Output = dataReader.GetValue(0) + " - " + dataReader.GetValue(1) + " - " + dataReader.GetValue(2);
            ListBox1.Items.Add(Output);
        }
        dataReader.Close();
        cnn.Close();
    }
```

View (Disconnected)

```
public DataSet GetEmployeeData()
{
    SqlConnection conString = new SqlConnection("myconnection");
    conString.Open();
    SqlCommand cmdQuery = new SqlCommand("Select * from Employee", co
nString);
    SqlDataAdapter sda = new SqlDataAdapter(cmdQuery);
    DataSet dsData = new DataSet();
    sda.Fill(dsData);
    return dsData;
}
```

Data Binding

```
{

        ListBox1.Visible = true;
        string connString;
        SqlConnection cnn;
        connString = " Data Source=MUM0219CPU0072\\SQLEXPRESS;Initial Catalog=Student;Integrated Security=True";
        cnn = new SqlConnection(connString);
        cnn.Open();
        SqlCommand command;
        String sql;
        sql = "Select * from Student";
        command = new SqlCommand(sql, cnn);
        ListBox1.DataSource = command.ExecuteReader();
        ListBox1.DataTextField="StuName";
        ListBox1.DataValueField="SAPID";
        ListBox1.DataBind();
        command.Dispose();
        cnn.Close();
    }
```

Populate Listbox with Table contents

```
using System.Data;
using System.Data.SqlClient;

namespace WebApplication1
{
    public partial class WebForm5 : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!this.IsPostBack)
            {
                FillAuthorList();
            }
        }

        private void FillAuthorList()
        {
            ListBox1.Items.Clear();
            string selectSQL = "SELECT * from employee";
            SqlConnection con = new SqlConnection("Data Source=MUM15121CPU1935\\DEMO1;Initial Catalog=books;Integrated Security=True");
            SqlCommand cmd = new SqlCommand(selectSQL, con);
            SqlDataReader reader;
            con.Open();
            reader = cmd.ExecuteReader();
            while (reader.Read())
            {
                ListItem newItem = new ListItem();
                newItem.Value = reader["emp_id"].ToString();
                newItem.Text = reader["emp_name"].ToString()+"-"+reader["dept"].ToString();
                ListBox1.Items.Add(newItem);
            }
            reader.Close();
        }
```

```
protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    // Create a Select statement that searches for a record
    // matching the specific author ID from the Value property.
    string selectSQL, result;
    Label1.Text = ListBox1.SelectedItem.Value;
    selectSQL = "SELECT * FROM employee ";
    selectSQL += "WHERE emp_id='" + ListBox1.SelectedItem.Value + "'";
    // Define the ADO.NET objects.
    SqlConnection con = new SqlConnection("Data Source=MUM15121CPU1935\\DEMO1;Initial Catalog=books;Integrated Security=True");
    SqlCommand cmd = new SqlCommand(selectSQL, con);
    SqlDataReader reader;
    con.Open();
    reader = cmd.ExecuteReader();
    reader.Read();

    result = "<b>";
    result += reader["emp_id"];
    result += ", ";
    result += reader["emp_name"];
    result += "</b><br />";
    result += "Dept: ";
    result += reader["dept"];
    result += "<br />";
    result += "Designation: ";
    result += reader["desig"];
    result += "<br />";

    Label1.Text = result;

    reader.Close();
}
```

- 455-456