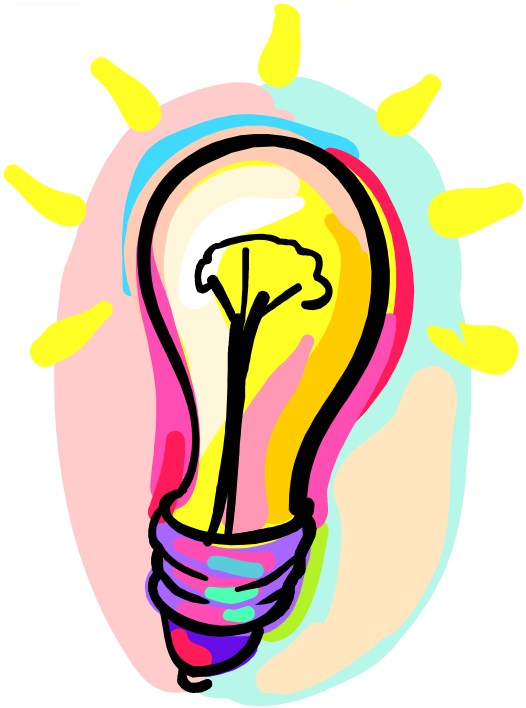
ADO.NET

Agenda

- SqlDataReader
- SqlDataAdapter
- DataSet
- DataTable
- Best Practice



Section 1

SQLDATAREADER

In previous session…

```
command.CommandType = System.Data.CommandType.Text;
command.CommandText = "SELECT Name, Salary FROM Employee";
var sqlDataReader = command.ExecuteReader();
```

- How do you hand returned values from ExecuteReader?

SqlDataReader

- An object is used to obtain the results of a SELECT statement from a command object.

SqlDataReader

- An object is used to obtain the results of a SELECT statement from a command object.
- For performance reasons, the data returned from a data reader is a forward-only stream of data.

SqlDataReader

- An object is used to obtain the results of a SELECT statement from a command object.
- For performance reasons, the data returned from a data reader is a forward-only stream of data.
- Data can be accessed from the stream in a sequential manner.

SqlDataReader

- An object is used to obtain the results of a SELECT statement from a command object.
- For performance reasons, the data returned from a data reader is a forward-only stream of data.
- Data can be accessed from the stream in a sequential manner.
- DataReader can not modify data.

Explain code line by line

```
SqlDataReader sqlDataReader = command.ExecuteReader();
if (sqlDataReader.HasRows)
{
    while (sqlDataReader.Read())
    {
        //// To get Name as the first column in SELECT command
        var name = sqlDataReader.GetString(0);
        //// To get Salary as the second column in SELECT command
        var salary = sqlDataReader.GetDecimal(1);

        /* your code to use name and salary */
    }
}
//// Always call the Close method when you have finished using the DataReader object.
sqlDataReader.Close();
```

Explain code line by line

```
//// Use SqlDataReader to take result from ExecuteReader method.
SqlDataReader sqlDataReader = command.ExecuteReader();

//// Check the result has data or not
if (sqlDataReader.HasRows)



//// Loop to get data row by row
//// Why do we use while loop in this situation?
//// Can we use for or foreach? If can, how?
//// Can we use do-while loop? If can, how?
while (sqlDataReader.Read())
```

Explain code line by line

```
//// Always call the Close method when you have finished using the DataReader object.
sqlDataReader.Close();

//// Developer often forget to close object. How to avoid this?
```

Explain code line by line

```
using (SqlDataReader sqlDataReader = command.ExecuteReader())
{
    if (sqlDataReader.HasRows)
    {
        while (sqlDataReader.Read())
        {
            /* your code to use name and salary */
        }
    }

     //// Developer should not need to close object manually
}
```

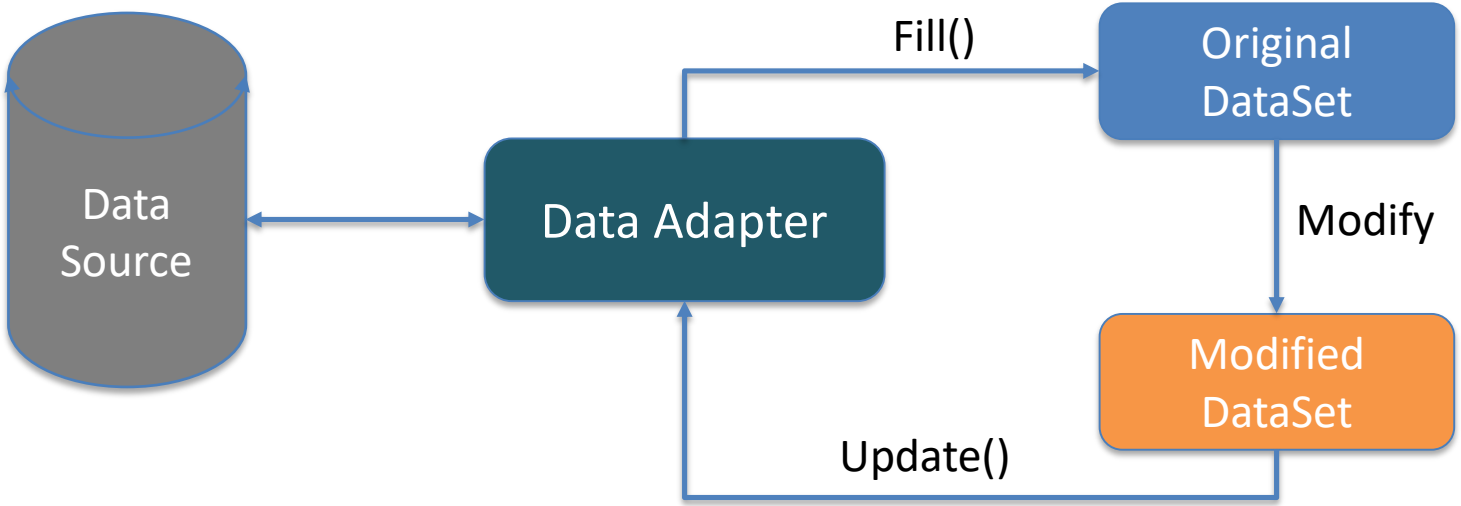
Section 2

SQLDATAADAPTER

SqlDataAdapter

- is used to retrieve data from a data source and populate tables within a DataSet
- resolves changes made to the DataSet back to the data source
- uses Command objects

Retrieve and Update data



DataSet

- is a memory-resident representation of data that provides a consistent relational programming model independent of the data source.
- represents a complete set of data that includes tables, constraints, and relationships among the tables.
- a screenshot of the SQL database

DataReader vs DataSet

- Used in a connected architecture
- Provides better performance
- Read-only access
- Can't create a relation in a data reader

- Used in a disconnected architecture
- Provides lower performance
- Read/write access
- Can create relations in a dataset

Populating a DataSet

```
//// Build command for adapter
SqlCommand command = new SqlCommand();
command.Connection = sqlConnection;
command.CommandType = System.Data.CommandType.Text;
command.CommandText = "SELECT Name, Salary FROM Employee";
```

Populating a DataSet

```
//// Build command for adapter
SqlCommand command = new SqlCommand();
command.Connection = sqlConnection;
command.CommandType = System.Data.CommandType.Text;
command.CommandText = "SELECT Name, Salary FROM Employee";

//// Declare DataAdapter
SqlDataAdapter sqlAdapter = new SqlDataAdapter();
sqlAdapter.SelectCommand = command;
//// Fill data to DataSet
DataSet dataSet = new DataSet();
sqlAdapter.Fill(dataSet, "EmployeeTable");
```

Update data in the DataSet

```
//// Get DataTable from DataSet by name
DataTable employeeTable = dataSet.Tables["EmployeeTable"];
//// Loop in table to change some value
foreach(DataRow row in employeeTable.Rows)
{
    var name = Convert.ToString(row["Name"]);
    if(name == "Peter")
    {
        row["Salary"] = 1234;
    }
}
```

Update data back to SQL

```
//// Build command for adapter
SqlCommand updateCommand = new SqlCommand();
updateCommand.Connection = sqlConnection;
updateCommand.CommandType = System.Data.CommandType.Text;
updateCommand.CommandText = "UPDATE Employee SET Salary = @Salary WHERE Name = @Name";
updateCommand.Parameters.Add("@Salary", SqlDbType.Decimal, 255, "Salary");
updateCommand.Parameters.Add("@Name", SqlDbType.NVarChar, 255, "Name");

//// Assign UpdateCommand for the DataAdapter
sqlAdapter.UpdateCommand = updateCommand;

//// Execute Update action
sqlAdapter.Update(dataSet, "EmployeeTable");
```

Section 3

ADO.NET BEST PRACTICES

Connection String Storage

- **DONOT** hardcode connection string in your code. Put it in configurable file (App.config, Web.config, …)
- Store connection strings securely

Database Connections

- Two golden rules characterize any code working with connections.
  - ✓ First: open the connection as late as possible.
  - ✓ Second: close the connection as early as possible.
- That means: application works with connections for the shortest time possible.

Security

- Always use Parameter for commands
- Validate all parameters before pass them to command
- Use difference connection for difference purposes/roles

Performance

- Group commands to reduce number of Open/Close connection
- Optimize command to reduce number of execute command
- Choose appropriate Execute Method.
- Decide to use DataSet OR DataReader

Thank you