Working with Disconnected Model in ADO.NET

ADO.NET Disconnected Model

The disconnected model utilizes `DataSet` and `SqlDataAdapter` to fetch and update data **without keeping the database connection open continuously**.

Database Setup (SQL Server)

Run the following SQL script to create the `Students` table in the database.

```
CREATE DATABASE StudentDB;
GO

USE StudentDB;
GO

CREATE TABLE Students (
    StudentID INT IDENTITY(1,1) PRIMARY KEY,
    Name NVARCHAR(100),
    Age INT,
    Course NVARCHAR(50)
);
```

CRUD Operatoins using Disconnected Model

1. Loading Data

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

static string connectionString = "CONNECTION_STRING";

static DataSet studentDataSet = new DataSet();
static SqlDataAdapter dataAdapter;
```

```
static void LoadData()
{
```

```
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        string query = "SELECT * FROM Students";
        dataAdapter = new SqlDataAdapter(query, con);

        studentDataSet.Clear(); // Clear previous data if any
        dataAdapter.Fill(studentDataSet, "Students"); // Fill dataset
    }
}
```

2. Create(Add New Rows)

```
static void AddStudent()
    {
        Console.Write("Enter Student Name: ");
        string name = Console.ReadLine();
        Console.Write("Enter Age: ");
        int age = Convert.ToInt32(Console.ReadLine());
        Console.Write("Enter Course: ");
        string course = Console.ReadLine();

        DataRow newRow = studentDataSet.Tables["Students"].NewRow();
        newRow["Name"] = name;
        newRow["Age"] = age;
        newRow["Course"] = course;
        studentDataSet.Tables["Students"].Rows.Add(newRow);

        Console.WriteLine("Student Added Locally. Save changes to update
database.");
    }
```

3. Read Data

```
 foreach (DataRow row in studentDataSet.Tables["Students"].Rows)
 {
     Console.WriteLine($"ID: {row["StudentID"]}, Name: {row["Name"]}, Age:
{row["Age"]}, Course: {row["Course"]}");
 }
```

4. Update Data

```
 static void UpdateStudent()
 {
     Console.Write("Enter Student ID to Update: ");
     int id = Convert.ToInt32(Console.ReadLine());

     DataRow[] rows = studentDataSet.Tables["Students"].Select($"StudentID =
{id}");
     if (rows.Length > 0)
     {
         Console.Write("Enter New Age: ");
         int newAge = Convert.ToInt32(Console.ReadLine());

         rows[0]["Age"] = newAge; // Update in memory

         Console.WriteLine("Student Updated Locally. Save changes to update
database.");
     }
     else
     {
         Console.WriteLine("Student not found!");
     }
 }
```

5. Delete Data

```
 static void DeleteStudent()
 {
     Console.Write("Enter Student ID to Delete: ");
     int id = Convert.ToInt32(Console.ReadLine());

     DataRow[] rows = studentDataSet.Tables["Students"].Select($"StudentID =
{id}");
     if (rows.Length > 0)
     {
         rows[0].Delete(); // Mark row for deletion in DataSet

         Console.WriteLine("Student Marked for Deletion. Save changes to update
database.");
     }
     else
     {
         Console.WriteLine("Student not found!");
     }
 }
```

6. Save Changes to Database

```
static void SaveChanges()
{
    using (SqlConnection con = new SqlConnection(connStr))
    {
        string query = "SELECT * FROM Students";
        dataAdapter = new SqlDataAdapter(query, con);

        // CommandBuilder automatically generates INSERT, UPDATE, DELETE commands
        SqlCommandBuilder commandBuilder = new SqlCommandBuilder(dataAdapter);

        dataAdapter.Update(studentDataSet, "Students"); // Update database
    }
}
```