Objective: Implement a secure ADO.NET solution with CRUD operations, stored procedures, SqlDataReader/SqlDataAdapter, and disconnected DataSet/DataTable pattern.

# 1) Database Design (SQL)

Create database, table, indexes, and stored procedures. Run this in SQL Server (SSMS).

-- 1. Create Database  
IF DB\_ID('BookstoreDb') IS NULL  
BEGIN  
 CREATE DATABASE BookstoreDb;  
END  
GO  
  
USE BookstoreDb;  
GO  
  
-- 2. Books Table  
IF OBJECT\_ID('dbo.Books', 'U') IS NULL  
BEGIN  
 CREATE TABLE dbo.Books  
 (  
 BookId INT IDENTITY(1,1) PRIMARY KEY,  
 Title NVARCHAR(200) NOT NULL,  
 Author NVARCHAR(200) NOT NULL,  
 Price DECIMAL(10,2) NOT NULL CHECK (Price >= 0),  
 PublishedOn DATE NULL,  
 Genre NVARCHAR(100) NULL,  
 CreatedAt DATETIME2 NOT NULL DEFAULT SYSUTCDATETIME()  
 );  
 CREATE INDEX IX\_Books\_Title ON dbo.Books(Title);  
END  
GO  
  
-- 3. Stored Procedures (Add / Update / Delete / Get)  
IF OBJECT\_ID('dbo.spAddBook', 'P') IS NOT NULL DROP PROCEDURE dbo.spAddBook;  
GO  
CREATE PROCEDURE dbo.spAddBook  
 @Title NVARCHAR(200),  
 @Author NVARCHAR(200),  
 @Price DECIMAL(10,2),  
 @PublishedOn DATE = NULL,  
 @Genre NVARCHAR(100) = NULL  
AS  
BEGIN  
 SET NOCOUNT ON;  
 INSERT INTO dbo.Books(Title, Author, Price, PublishedOn, Genre)  
 VALUES(@Title, @Author, @Price, @PublishedOn, @Genre);  
  
 SELECT SCOPE\_IDENTITY() AS NewBookId;  
END  
GO  
  
IF OBJECT\_ID('dbo.spUpdateBook', 'P') IS NOT NULL DROP PROCEDURE dbo.spUpdateBook;  
GO  
CREATE PROCEDURE dbo.spUpdateBook  
 @BookId INT,  
 @Title NVARCHAR(200),  
 @Author NVARCHAR(200),  
 @Price DECIMAL(10,2),  
 @PublishedOn DATE = NULL,  
 @Genre NVARCHAR(100) = NULL  
AS  
BEGIN  
 SET NOCOUNT ON;  
 UPDATE dbo.Books  
 SET Title = @Title,  
 Author = @Author,  
 Price = @Price,  
 PublishedOn = @PublishedOn,  
 Genre = @Genre  
 WHERE BookId = @BookId;  
  
 SELECT @@ROWCOUNT AS RowsAffected;  
END  
GO  
  
IF OBJECT\_ID('dbo.spDeleteBook', 'P') IS NOT NULL DROP PROCEDURE dbo.spDeleteBook;  
GO  
CREATE PROCEDURE dbo.spDeleteBook  
 @BookId INT  
AS  
BEGIN  
 SET NOCOUNT ON;  
 DELETE FROM dbo.Books WHERE BookId = @BookId;  
 SELECT @@ROWCOUNT AS RowsAffected;  
END  
GO  
  
IF OBJECT\_ID('dbo.spGetBooks', 'P') IS NOT NULL DROP PROCEDURE dbo.spGetBooks;  
GO  
CREATE PROCEDURE dbo.spGetBooks  
AS  
BEGIN  
 SET NOCOUNT ON;  
 SELECT BookId, Title, Author, Price, PublishedOn, Genre, CreatedAt  
 FROM dbo.Books  
 ORDER BY BookId DESC;  
END  
GO

# 2) Project Structure

Create a Console project and add a class library structure inside the same project for clarity.

Bookstore.AdoConsole\  
 Program.cs  
 Models\  
 Book.cs  
 Data\  
 DbConfig.cs  
 BookRepository.cs  
 Services\  
 DataSetDemo.cs

## Book.cs

using System;  
  
namespace Bookstore.AdoConsole.Models  
{  
 public class Book  
 {  
 public int BookId { get; set; }  
 public string Title { get; set; } = string.Empty;  
 public string Author { get; set; } = string.Empty;  
 public decimal Price { get; set; }  
 public DateTime? PublishedOn { get; set; }  
 public string? Genre { get; set; }  
  
 public override string ToString()  
 {  
 return $"#{BookId} | {Title} by {Author} | {Price:C} | {PublishedOn:yyyy-MM-dd} | {Genre}";  
 }  
 }  
}

## DbConfig.cs

namespace Bookstore.AdoConsole.Data  
{  
 public static class DbConfig  
 {  
 Server=.;Database=BookstoreDb;Trusted\_Connection=True;TrustServerCertificate=True;  
 Server=.;Database=BookstoreDb;User Id=sa;Password=YourStrong!Passw0rd;TrustServerCertificate=True;  
 public static string ConnectionString =>  
 "Server=.;Database=BookstoreDb;Trusted\_Connection=True;TrustServerCertificate=True;";  
 }  
}

## BookRepository.cs

using System;  
using System.Collections.Generic;  
using System.Data;  
using System.Data.SqlClient;  
using Bookstore.AdoConsole.Models;  
  
namespace Bookstore.AdoConsole.Data  
{  
 public class BookRepository  
 {  
 private readonly string \_connString;  
  
 public BookRepository(string connString)  
 {  
 \_connString = connString;  
 }

public int AddBook(Book book)  
 {  
 using var conn = new SqlConnection(\_connString);  
 using var cmd = new SqlCommand(@"INSERT INTO dbo.Books(Title, Author, Price, PublishedOn, Genre)  
 VALUES(@Title, @Author, @Price, @PublishedOn, @Genre);  
 SELECT CAST(SCOPE\_IDENTITY() AS INT);", conn);  
  
 cmd.Parameters.AddWithValue("@Title", book.Title);  
 cmd.Parameters.AddWithValue("@Author", book.Author);  
 cmd.Parameters.AddWithValue("@Price", book.Price);  
 cmd.Parameters.AddWithValue("@PublishedOn", (object?)book.PublishedOn ?? DBNull.Value);  
 cmd.Parameters.AddWithValue("@Genre", (object?)book.Genre ?? DBNull.Value);  
  
 conn.Open();  
 int newId = (int)cmd.ExecuteScalar();  
 return newId;  
 }  
  
 public int UpdateBook(Book book)  
 {  
 using var conn = new SqlConnection(\_connString);  
 using var cmd = new SqlCommand(@"UPDATE dbo.Books SET  
 Title=@Title,  
 Author=@Author,  
 Price=@Price,  
 PublishedOn=@PublishedOn,  
 Genre=@Genre  
 WHERE BookId=@BookId", conn);  
  
 cmd.Parameters.AddWithValue("@Title", book.Title);  
 cmd.Parameters.AddWithValue("@Author", book.Author);  
 cmd.Parameters.AddWithValue("@Price", book.Price);  
 cmd.Parameters.AddWithValue("@PublishedOn", (object?)book.PublishedOn ?? DBNull.Value);  
 cmd.Parameters.AddWithValue("@Genre", (object?)book.Genre ?? DBNull.Value);  
 cmd.Parameters.AddWithValue("@BookId", book.BookId);  
  
 conn.Open();  
 return cmd.ExecuteNonQuery();  
 }  
  
 public int DeleteBook(int bookId)  
 {  
 using var conn = new SqlConnection(\_connString);  
 using var cmd = new SqlCommand("DELETE FROM dbo.Books WHERE BookId=@BookId", conn);  
 cmd.Parameters.AddWithValue("@BookId", bookId);  
  
 conn.Open();  
 return cmd.ExecuteNonQuery();  
 }  
  
   
 public List<Book> GetAllBooksWithReader()  
 {  
 var list = new List<Book>();  
  
 using var conn = new SqlConnection(\_connString);  
 using var cmd = new SqlCommand("SELECT BookId, Title, Author, Price, PublishedOn, Genre FROM dbo.Books ORDER BY BookId DESC", conn);  
  
 conn.Open();  
 using var reader = cmd.ExecuteReader(CommandBehavior.CloseConnection);  
 while (reader.Read())  
 {  
 var book = new Book  
 {  
 BookId = reader.GetInt32(0),  
 Title = reader.GetString(1),  
 Author = reader.GetString(2),  
 Price = reader.GetDecimal(3),  
 PublishedOn = reader.IsDBNull(4) ? (DateTime?)null : reader.GetDateTime(4),  
 Genre = reader.IsDBNull(5) ? null : reader.GetString(5)  
 };  
 list.Add(book);  
 }  
  
 return list;  
 }  
  
   
 public int AddBookUsingSp(Book book)  
 {  
 using var conn = new SqlConnection(\_connString);  
 using var cmd = new SqlCommand("dbo.spAddBook", conn);  
 cmd.CommandType = CommandType.StoredProcedure;  
  
 cmd.Parameters.AddWithValue("@Title", book.Title);  
 cmd.Parameters.AddWithValue("@Author", book.Author);  
 cmd.Parameters.AddWithValue("@Price", book.Price);  
 cmd.Parameters.AddWithValue("@PublishedOn", (object?)book.PublishedOn ?? DBNull.Value);  
 cmd.Parameters.AddWithValue("@Genre", (object?)book.Genre ?? DBNull.Value);  
  
 conn.Open();  
 var result = cmd.ExecuteScalar();  
 return Convert.ToInt32(result);  
 }  
  
 public int UpdateBookUsingSp(Book book)  
 {  
 using var conn = new SqlConnection(\_connString);  
 using var cmd = new SqlCommand("dbo.spUpdateBook", conn);  
 cmd.CommandType = CommandType.StoredProcedure;  
  
 cmd.Parameters.AddWithValue("@BookId", book.BookId);  
 cmd.Parameters.AddWithValue("@Title", book.Title);  
 cmd.Parameters.AddWithValue("@Author", book.Author);  
 cmd.Parameters.AddWithValue("@Price", book.Price);  
 cmd.Parameters.AddWithValue("@PublishedOn", (object?)book.PublishedOn ?? DBNull.Value);  
 cmd.Parameters.AddWithValue("@Genre", (object?)book.Genre ?? DBNull.Value);  
  
 conn.Open();  
 var result = cmd.ExecuteScalar();  
 return Convert.ToInt32(result); // RowsAffected  
 }  
  
 public int DeleteBookUsingSp(int bookId)  
 {  
 using var conn = new SqlConnection(\_connString);  
 using var cmd = new SqlCommand("dbo.spDeleteBook", conn);  
 cmd.CommandType = CommandType.StoredProcedure;  
  
 cmd.Parameters.AddWithValue("@BookId", bookId);  
  
 conn.Open();  
 var result = cmd.ExecuteScalar();  
 return Convert.ToInt32(result); // RowsAffected  
 }  
  
 public DataSet GetBooksWithAdapter(out SqlDataAdapter adapter)  
 {  
 var ds = new DataSet();  
 var conn = new SqlConnection(\_connString);  
 adapter = new SqlDataAdapter("SELECT BookId, Title, Author, Price, PublishedOn, Genre FROM dbo.Books ORDER BY BookId DESC", conn);  
  
   
 var builder = new SqlCommandBuilder(adapter);  
 adapter.Fill(ds, "Books");  
  
 return ds;  
 }  
  
 public int UpdateBooksFromDataSet(SqlDataAdapter adapter, DataSet ds)  
 {  
   
 return adapter.Update(ds, "Books");  
 }  
 }  
}

## DataSetDemo.cs

using System;  
using System.Data;  
using System.Data.SqlClient;  
using Bookstore.AdoConsole.Data;  
  
namespace Bookstore.AdoConsole.Services  
{  
 public static class DataSetDemo  
 {  
 public static void RunDisconnectedDemo(BookRepository repo)  
 {  
 Console.WriteLine("=== Disconnected DataSet/DataTable Demo ===");  
  
 var ds = repo.GetBooksWithAdapter(out SqlDataAdapter adapter);  
 var table = ds.Tables["Books"]!;  
  
   
 var newRow = table.NewRow();  
 newRow["Title"] = "In-Memory Book";  
 newRow["Author"] = "DataSet Author";  
 newRow["Price"] = 199.99m;  
 newRow["PublishedOn"] = DBNull.Value;  
 newRow["Genre"] = "Tech";  
 table.Rows.Add(newRow);  
  
   
 if (table.Rows.Count > 0)  
 {  
 var first = table.Rows[0];  
 first["Price"] = Convert.ToDecimal(first["Price"]) + 10m  
 }  
  
   
 if (table.Rows.Count > 1)  
 {  
 table.Rows[table.Rows.Count - 1].Delete();  
 }  
  
   
 int affected = repo.UpdateBooksFromDataSet(adapter, ds);  
 Console.WriteLine($"DataSet changes pushed to DB. Rows affected: {affected}");  
 }  
 }  
}

## Program.cs

using System;  
using Bookstore.AdoConsole.Data;  
using Bookstore.AdoConsole.Models;  
using Bookstore.AdoConsole.Services;  
  
namespace Bookstore.AdoConsole  
{  
 class Program  
 {  
 static void Main(string[] args)  
 {  
 Console.WriteLine("ADO.NET Bookstore – Console Demo\n");  
 var repo = new BookRepository(DbConfig.ConnectionString);  
  
 // 1) Add   
 var newBook = new Book  
 {  
 Title = "Clean Code",  
 Author = "Robert C. Martin",  
 Price = 45.50m,  
 PublishedOn = new DateTime(2008, 8, 1),  
 Genre = "Programming"  
 };  
 int newId = repo.AddBook(newBook);  
 Console.WriteLine($"Added book (param query). New ID: {newId}");  
  
 // 2) Add using Stored Procedure  
 var newBook2 = new Book  
 {  
 Title = "The Pragmatic Programmer",  
 Author = "Andrew Hunt & David Thomas",  
 Price = 55.00m,  
 PublishedOn = new DateTime(1999, 10, 30),  
 Genre = "Programming"  
 };  
 int spId = repo.AddBookUsingSp(newBook2);  
 Console.WriteLine($"Added book (stored proc). New ID: {spId}");  
  
 // 3) Read with SqlDataReader  
 Console.WriteLine("\nAll books (SqlDataReader):");  
 foreach (var b in repo.GetAllBooksWithReader())  
 {  
 Console.WriteLine(b);  
 }  
  
 // 4) Update using SP  
 var toUpdate = new Book  
 {  
 BookId = spId,  
 Title = "The Pragmatic Programmer (20th Anniversary)",  
 Author = "Andrew Hunt & David Thomas",  
 Price = 60.00m,  
 PublishedOn = new DateTime(2019, 9, 13),  
 Genre = "Programming"  
 };  
 int upd = repo.UpdateBookUsingSp(toUpdate);  
 Console.WriteLine($"\nUpdated via SP. Rows affected: {upd}");  
  
 // 5) Disconnected DataSet/DataTable + SqlDataAdapter  
 DataSetDemo.RunDisconnectedDemo(repo);  
  
 // 6) Delete via SP (cleanup demo)  
 int del = repo.DeleteBookUsingSp(spId);  
 Console.WriteLine($"\nDeleted via SP (bookId={spId}). Rows affected: {del}");  
  
 Console.WriteLine("\nDone.");  
 }  
 }  
}

# 3) SQL Injection Prevention

// WRONG  
var title = userInput; // e.g., "X'; DROP TABLE dbo.Books; --"  
var cmdText = "SELECT \* FROM dbo.Books WHERE Title = '" + title + "'";  
  
// RIGHT  
using var cmd = new SqlCommand("SELECT \* FROM dbo.Books WHERE Title = @Title", conn);  
cmd.Parameters.AddWithValue("@Title", title);

# 4) Sample Console Output

ADO.NET Bookstore – Console Demo  
  
Added book (param query). New ID: 1  
Added book (stored proc). New ID: 2  
  
All books (SqlDataReader):  
#2 | The Pragmatic Programmer by Andrew Hunt & David Thomas | $55.00 | 1999-10-30 | Programming  
#1 | Clean Code by Robert C. Martin | $45.50 | 2008-08-01 | Programming  
  
Updated via SP. Rows affected: 1  
=== Disconnected DataSet/DataTable Demo ===  
DataSet changes pushed to DB. Rows affected: 2  
  
Deleted via SP (bookId=2). Rows affected: 1  
  
Done.