***Praca z bazą danych SqlClient***

**Instrukcja**

1. **Cel ćwiczenia**

Praktyczne wprowadzanie do Przetwarzania transakcyjnego. Wszystkie zadania wykonywane będą z wykorzystaniem dostawcy danych ADO.NET – SqlClient ( **System.Data.SqlClient** ). Na tym etapie nie powinno się wykorzystywać do tego celu żadnego ORM-a.

1. **Wymagana znajomość zagadnień**

* Pisanie prostych aplikacji w C# lub innym obiektowym języku wysokiego poziomu
* Podstawowa znajomość SQL, umiejętność pisania zapytań do bazy danych
* Wskazana podstawowa znajomość języka angielskiego lub też umiejętność korzystania z narzędzi tłumaczenia on-line. Wynika to z faktu, że większość użytecznej i najbardziej aktualnej dokumentacji jest publikowana właśnie w języku angielskim.

1. **Literatura, materiały dydaktyczne**

* <https://docs.microsoft.com/pl-pl/dotnet/csharp/>
* <https://www.sqlpedia.pl/>
* <https://www.mssqltips.com/sqlservertip/5771/querying-sql-server-tables-from-net/>
* <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/>
* <https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient?view=dotnet-plat-ext-5.0>
* <https://docs.microsoft.com/pl-pl/dotnet/api/system.data.sqlclient.sqltransaction?view=dotnet-plat-ext-5.0>
* <https://www.webtrainingroom.com/adonet/transaction>
* <https://www.plukasiewicz.net/Artykuly/SQL_Transactions>

1. **Wiadomości teoretyczne**

Przydatne wskazówki:

* **Pobranie ID ostatnio dodanego do bazy danych rekordu**

W przypadku gdy mamy w bazie danych tabele dla której klucz główny (id) nadawane są przez bazę automatycznie i chcemy uzyskać dostęp do przydzielonego id czyli mamy włączone IDENTITY z opcją auto-increment, to możemy to zrealizować na przykład poprzez:

Na końcu CommandText w SqlCommand dopisujemy:

cmd.CommandText += " SELECT SCOPE\_IDENTITY();";

teraz możemy wysłać polecenie do bazy ale wykorzystując do tego ExecuteScalar():

**object** value = cmd.ExecuteScalar();

if(value != DBNull.Value)

{

**string**ResultId = value.ToString();

}

Oczywiście jeżeli chcemy pobrać wartość jako na przykład **long** to musimy zastosować odpowiednie rzutowanie.

Wywołując **ExecuteScalar** najbezpieczniej jest zastosowanie przypisania do **object**, wynika to z faktu, że zwrócone może być DBNullprzy czym ważna uwaga to nie jest to samo co **null** w C#. Nie możemy bezpośrednio rzutować się pomiędzy tymi typami.

* **Zapisywanie do bazy danych rekordu zawierającego w danych pola null**

W sytuacji gdy chcemy zapisać do bazy danych jakikolwiek obiekt, który zawiera atrybuty nullable (dopuszczające null) lub też typu string musimy uwzględnić to przy budowaniu polecenia Insert (lub Update). Ponieważ w przypadku SqlClient zaleca się wykorzystanie parametryzowanych kwerend, to nie możemy przypisywać do parametru wartości null (jak już wspomniano nie jest to tym samym **null** z bazy danych). Przykład w jaki sposób można wykonać właściwe rzutowanie typów.

**SqlCommand**myCommand = newSqlCommand();

myCommand.CommandText = "INSERT INTO [InitialCatalog].[dbo].[Table] ( "

+ "[ColumnName1], "

+ "[ColumnName2], "

+ "[ColumnName3] "

+ " ) "

+ "VALUES ( @val1, @val2, @val3 ) ; ";

myCommand.Parameters.AddWithValue("@val1", var1);

if (var2 == null) myCommand.Parameters.AddWithValue("@val2",DBNull.Value);

elsemyCommand.Parameters.AddWithValue("@val2", var2);

myCommand.Parameters.AddWithValue("@val3", var3);

Jak widzimy w przytoczonym przykładzie, zmienne var1 oraz var3 są zmiennymi niedopuszczającymi wartości **null** (wiemy że takowa nie wystąpi) więc przy bindowaniu parametrów można swobodnie wykorzystać Parameters.AddWithValue przy czym co istotne nie musimy wnikać jakiego dokładnie typu to będą dane (int, string, DateTime, decimal …. ), nie ma znaczenia – środowisko sobie poradzi o ile nie będziemy mieli do czynienia z null. Dla tej sytuacji (zmienna var2) z przykładu widzimy uwarunkowanie bindowanej wartości: w przypadku null przekazujemy DBNull.Value, w pozostałych zaś przypadkach postępujemy tak samo jak w przypadku innych zmiennych z przykładu.

* **Odczytanie z bazy danych rekordu zawierającego w danych pola null**

W przypadku pobierania danych z obiektu Reader musimy postępować analogicznie, tzn. przed mapowaniem należy się upewnić czy zwracana dana ma wartość czy też jest to DBNulli w zależności od tego postępować różnie w trakcie odczytu. Przykład dla tego samego obiektu co powyżej:

List<NaszTyp>results = new List<NaszTyp>();

myCommand.CommandText = "SELECT "

+ "[ColumnName1], "

+ "[ColumnName2], "

+ "[ColumnName3] "

+ "FROM [InitialCatalog].[dbo].[Table] "

+ "WHERE …………… ";

try

{

myCommand.Connection.Open();

SqlDataReader reader = myCommand.ExecuteReader();

if (reader.HasRows)

{

while (reader.Read())

{

int var1 = (int)reader["ColumnName1"];

DateTime? Var2 = null;

if (!reader.IsDBNull(reader.GetOrdinal("ColumnName2")))

{

var2 = (DateTime)reader["ColumnName2"];

}

string var3 = ((string)reader["ColumnName3"]).Trim();

result.Results.Add(new NaszTyp(var1, var2, var3));

}

}

reader.Close();

}

catch (SqlException e)

{

… (Jakaś obsługa wyjątku, logowanie etc.)

}

finally

{

if (myCommand.Connection != null) myCommand.Connection.Close();

}

* **Przykład użycia transakcji w kodzie**

using (SqlConnection connection = newSqlConnection("NaszConnectionString"))

{

try

{

connection.Open();

using (SqlTransaction trans = connection.BeginTransaction())

{

Tutaj nasze operacje na bazie danych (Insert, Update, Delete);

…

…

trans.Commit();

}

}

catch (SqlException ex)

{

(Jakaś obsługa wyjątku, logowanie etc.)

}

}

1. **Przebieg ćwiczenia**

Stworzyć w bazie danych co najmniej 2 tabele (można sobie tę bazę przygotować z wykorzystaniem dowolnych narzędzi), dla przebiegu ćwiczenia dobrze by było aby te 2 tabele połączone były jakąkolwiek relacją, np. zamówienie z danymi nagłówkowymi oraz pozycje na zamówieniu przypisane do tegoż zamówienia. Następnie przygotować prosty program pobierający dane od użytkownika (z jakiegoś formularza) lub też pliku CSV cały zestaw danych tzn. wszystkie dane dla tabeli 1 oraz tabeli 2. A następnie wykonujący operacje zapisu tych danych do wskazanych tabel, **jako dwie kolejne niepowiązane ze sobą operacje**. Zadbać o to, by mieć przygotowane 2 zestawy danych testowych, 1 zawierający kompletny zestaw danych prawidłowych, typy danych i ich rozmiar zgodne ze zdefiniowaną strukturą bazy danych oraz 2 zestaw danych dla którego dane zależne (w tabeli podrzędnej) zawierać będą jakieś błędy, przykładowo pole tekstowe o rozmiarze większym niż zdefiniowana maksymalna liczba znaków w tabeli dla tego pola. W trakcie testów wprowadzić kolejno, zestaw danych zawierający prawidłowe wartości dla obu tabel, sprawdzić poprawność wykonania programu. Następnie podjąć próbę zapisania danych z zestawu zawierającego błędne dane. Po wykonana się wszystkich instrukcji i komunikacie błędu przy próbie zapisu nieprawidłowych danych do tabeli 2, sprawdzić rezultat wykonanych operacji w bazie danych (jakie dane zostały zapisane), zanotować uwagi.

Po ukończeniu pierwszej części, należy zmodyfikować kod programu tak, aby operacje dodania wpisów do tych tabel zrealizowane zostały w obrębie jednej transakcji. Powtórzyć próby dodania danych analogicznie jak w poprzednim przypadku: zestaw prawidłowych danych, oraz zestaw zawierający błędy. Ponownie zweryfikować rezultat wykonywanych operacji w bazie danych, wyprowadzić i zanotować wnioski.

Dodatkowo przygotować program zapisujący i odczytujący rekordy które zawierające wartości liczbowe (integer) z dopuszczeniem **null**,czyli po stronie aplikacji są to typu nullable. Zaprezentować wyniki działania aplikacji, i opisać jej działanie.

.

1. **Opracowanie wyników, sprawozdanie.**
2. **Zadania dodatkowe / Uwagi**