nAxiom

Akcja Algorytm C#, klasa SQLHelper

Wydanie: 1.0, wersja nAxiom: 1.9.1

Spis treści

1. Wstęp

Klasa `SqlHelper` pozwala na wykonanie typowych operacji na bazach danych. Wspierane są połączenia z bazą systemową nAxiom oraz źródłami danych w bazach MS SQL i Oracle. W miarę rozwoju platformy możliwe, że będą dostępne inne typy źródeł danych, które powinny wspierać metody dostępne w

udostępnionej klasie. `SqlHelper` to klasa statyczna dostępna w przestrzeni nazw `CSharpScript.SqlRunner`. Dostępne metody to:

- `ExecuteSqlQueryRowsToDictionary`: zwraca wynik zapytania SQL jako *List<Dictionary<string, object>>*,
- `ExecuteSqlScalar`: zwraca wynik zapytania jako pojedynczy rezultat typu *object*,
- `ExecuteSqlNonQuery`: zwraca w wyniku *true* lub *false*, zależnie od rezultatu zapytania modyfikującego dane (np. czy instrukcja UPDATE dokonała aktualizacji rekordu); zwracany typ to *bool*,
- *ExecuteSqlNonQueryWithOutputParameters*: metoda wykorzystywana przede wszystkim do wykonywania procedur, gdzie wynik zwracany jest za pomocą parametrów, wynik zwracany jako *Dictionary<string, object>*.

2. Metoda `ExecuteSqlQueryRowsToDictionary`

2.1. Opis

Asynchroniczna metoda statyczna służąca do pobierania wyniku zapytania SQL w formie *List<Dictionary<string, object>>*. Zwracana lista to lista rekordów, a słownik opisuje wartości kolumn w rekordzie: klucz to nazwa kolumny, a wartość to wartość z tej kolumny zwrócona przez SQL.

2.2. Parametry wejściowe

- *query* (*string*): zapytanie SQL do wykonania na źródle danych,
- *parameters* (*Dictionary<string, object>*): słownik parametrów wejściowych zapytania SQL,
- *dataSourceId* (*int*): identyfikator źródła danych

2.3. Rezultat

List<Dictionary<string, object>>

lista reprezentująca rekordy zwrócone przez zapytanie SQL, słownik reprezentuje wartości w poszczególnych kolumnach rekordu: klucz to nazwa kolumny, wartość klucza to wartość z kolumny

2.4. Przykład użycia

```
var result =
await SqlHelper.ExecuteSqlQueryRowsToDictionary("SELECT * FROM [core].[UserProfiles]");
File.WriteAllText(@"C:\nAxiom\Akcje C#\1 Profile.txt",
    JsonConvert.SerializeObject(result, Formatting.Indented));
```

3. Metoda `ExecuteSqlScalar`

3.1. Opis

Asynchroniczna metoda statyczna służąca do pobierania wyniku zapytania SQL w formie pojedynczej wartości typu *object*.

3.2. Parametry wejściowe

- *query* (*string*): zapytanie SQL do wykonania na źródle danych,
- *parameters* (*Dictionary<string, object>*): słownik parametrów wejściowych zapytania SQL,
- *dataSourceId* (*int*): identyfikator źródła danych

3.3. Rezultat

object
: pojedyncza wartość zwracana przez zapytanie SQL

3.3.1. Przykład użycia

```
File.WriteAllText(@"C:\nAxiom\Akcje C#\4 ScalarLocal.txt",
(await SqlHelper.ExecuteSqlScalar(
    "SELECT TOP 1 [Id] FROM [core].[UserProfiles]")).ToString());
```

4. Metoda `ExecuteSqlNonQuery`

4.1. Opis

Asynchroniczna metoda statyczna służąca do wywołania polecenia SQL do manipulowania danymi (*NonQuery*, czyli nie zwracającego żadnego wyniku) na przykład *INSERT*, *UPDATE*, *DELETE*. Rezultat to wartość *true* lub *false*, zależnie od tego, czy zapytanie dokonało modyfikacji danych.

4.2. Parametry wejściowe

- *query* (*string*): zapytanie SQL do wykonania na źródle danych,
- *parameters* (*Dictionary<string, object>*): słownik parametrów wejściowych zapytania SQL,
- *dataSourceId* (*int*): identyfikator źródła danych

4.3. Rezultat

true/false (bool)
: wartość zależna od tego, czy polecenie zmieniło dane w źródle

4.4. Przykład użycia

```
var result =
await SqlHelper.ExecuteSqlNonQuery(
    "INSERT INTO [dbo].[Names] ([Name]) VALUES ('Michał')");
```

5. Metoda `ExecuteSqlNonQueryWithOutputParameters`

5.1. Opis

Asynchroniczna metoda statyczna służąca do wywołania polecenia SQL, którego wynik jest zwracany za pomocą parametrów wyjściowych, na przykład procedury. Wynik jest zwracany w formie słownika, w

którym klucz to nazwa parametru wyjściowego, a wartość to wynik dla danego parametru zwrócony przez zapytanie.

5.2. Parametry wejściowe

- *query* (*string*): zapytanie SQL do wykonania na źródle danych,
- *inputParameters* (*Dictionary<string, object>*): słownik parametrów wejściowych zapytania SQL,
- *outputParameters* (*Dictionary<string, object>*): słownik parametrów wyjściowych zapytania SQL,
- *dataSourceId* (*int*): identyfikator źródła danych

5.3. Rezultat

Dictionary<string, object>

słownik wartości wyjściowych zapytania SQL, gdzie klucz to nazwa parametru przekazana w danych wyjściowych (*outputParameters*), a wartość to dane zwrócone przez zapytanie

5.4. Przykład użycia

```
var result =
await SqlHelper.ExecuteSqlNonQueryWithOutputParameters(
"EXEC [dbo].[InsertNewNameWithOutputParam] @Name = {@Name}, @Id = {@Id} OUTPUT",
 new Dictionary<string, object>() { { "Name", "Protazy"} },
 new Dictionary<string, object>() { { "Id", 0} });
```

6. Inne źródła danych

Metod klasy *SqlHelper* można używać do wykonywania zapytań na bazach innych niż systemowa, dzięki użyciu parametru *dataSourceId*. Podczas definiowania akcji wartość parametru można określić przy użyciu mechanizmu *SmartNumbers*.
Przykład użycia:

```
var resultDevLocal =
await SqlHelper.ExecuteSqlQueryRowsToDictionary("SELECT * FROM [core].[UserProfiles]",
null, {&BaseApp.DataSources.BaseModule.My_DS});
File.WriteAllText(@"C:\nAxiom\Akcje C#\2 ProfileDevLocal.txt",
    JsonConvert.SerializeObject(resultDevLocal, Formatting.Indented));
```

7. Obsługa błędów

Mechanizm wspiera standardową obsługę błędów. Błędy specyficzne dla klasy *SqlHelper* to:

- `Api.ActionModule.CSharpScriptDataSourceNotExist`: brak źródła danych dla wskazanego ID,
- `Api.ActionModule. CSharpScriptNonSqlDataSource`: wskazane źródło danych nie jest źródłem SQL lub źródło nie zostało zaimplementowane.

8. Przekazywanie parametrów

W poleceniu SQL można używać zmiennych z platformy nAxiom, to znaczy *{@Parametr}*. Wartości tych zmiennych należy przekazać na przykład za pomocą słownika *Dictionary<string, object>*.
Taki słownik można zbudować ręcznie lub przekazać zdeserializowany model formularza ze zmiennej globalnej *Model*.
Przykład:

```
JsonConvert.DeserializeObject<Dictionary<string, object>>(Model)
```

Pamiętać należy o dołączeniu biblioteki *Newtonsoft.Json.dll* oraz dodaniu przestrzeni nazw na początku skryptu:

```
using Newtonsoft.Json;
```

9. Przykłady kodu

1. Pobranie profili użytkowników z bazy systemowej w formacie JSON i zapisanie do pliku

```
var result =
await SqlHelper.ExecuteSqlQueryRowsToDictionary(
 "SELECT * FROM [core].[UserProfiles]");
File.WriteAllText(@"C:\nAxiom\Akcje C#\1 Profile.txt",
 JsonConvert.SerializeObject(result, Formatting.Indented));
```

2. Pobranie profili użytkowników z innego źródła w formacie JSON i zapisanie do pliku

```
var resultDevLocal =
await SqlHelper.ExecuteSqlQueryRowsToDictionary(
 "SELECT * FROM [core].[UserProfiles]",
null, {&BaseApp.DataSources.BaseModule.My_DS});
File.WriteAllText(@"C:\nAxiom\Akcje C#\2 ProfileDevLocal.txt",
 JsonConvert.SerializeObject(resultDevLocal, Formatting.Indented));
```

3. Pobranie pierwszego ID profilu użytkownika

```
File.WriteAllText(@"C:\nAxiom\Akcje C#\4 ScalarLocal.txt",
(await SqlHelper.ExecuteSqlScalar(
 "SELECT TOP 1 [Id] FROM [core].[UserProfiles]")).ToString());
```

4. Wykonanie procedury jako polecenie skalarne, gdzie wynik to *ID* wstawionego wpisu *(SCOPE_IDENTITY())*, gdzie przekazano dane z formularza

```
File.WriteAllText(@"C:\nAxiom\Akcje C#\9 ScalarMSInsertProcedureFromModel.txt",
(await SqlHelper.ExecuteSqlScalar(
 "EXEC [dbo].[InsertNewName] {@Imie}",
 JsonConvert.DeserializeObject<Dictionary<string, object>>(Model))).ToString());
```

5. Wykonanie procedury na bazie Oracle z danymi wyjściowymi w formie parametrów, procedura zwraca imię dla wskazanego *ID* oraz datę wykonania

```
var resultParams =
await SqlHelper.ExecuteSqlNonQueryWithOutputParameters(
 "BEGIN GETNAMEBYID({@Id}, {@Imie}, {@Date}); END;",
new Dictionary<string, object>() { { "Id", 10} },
new Dictionary<string, object>() { { "Imie", ""}, { "Date", ""} }, 3);
```