Biblioteka BDL\_DLL

Opis działania

# Wstęp

System bazodanowy SQL serwer pozwala na dołączanie do baz tzw. assembly – wtyczek, bibliotek pisanych na platformę CLR (Common Language Runtime). Mogą być to biblioteki pisane w języku C#, ale również np. Visual Basic bądź też F#.

Niniejsza biblioteka działa właśnie jako assembly. Językiem biblioteki jest C#, użyte weń mechanizmy i składnia języka wymagają korzystania z .net Framework w wersji najwyższej (na czas pisania jest to 4.7).

Metoda wykorzystania takiej wtyczki pozwoli na elastyczne i łatwe połączenie SQL Server z serwisem REST Banku Danych Lokalnych. Biblioteka realizując zadanie łączenia się poprzez udostępnione API działa niezależnie od serwera SQL. Separacja odpowiedzialności jest tu bardzo ważna – każdy fragment projektu wykonuje to co potrafi najlepiej. SQL Server odpowiada za sprawne zarządzanie danymi, natomiast w kwestii połączeń z serwisami REST biblioteki .net są bezkonkurencyjne - w porównaniu z wbudowanymi w SQL Server mechanizmami.

Metoda ta jest również odporna na zmiany. W przypadku zmiany sposobu łączenia się z API Banku Danych Lokalnych wystarczające będzie odpowiednie dostosowanie niniejszej biblioteki, natomiast mechanizmy po stronie SQL Server pozostaną bez zmian.

Biblioteka niniejsza składa się z kilkunastu klas, wszystkie znajdują się w jednej przestrzeni nazw. Biblioteka zawarta jest w ramach jednego projektu Visual Studio. Rolę pomocniczą pełni inny projekt - testów jednostkowych, który jednak nie jest zawarty w wyjściowym artefakcie DLL.

# Diagram klas

Poniższy ogólny diagram przedstawia klasy reprezentujące encje.

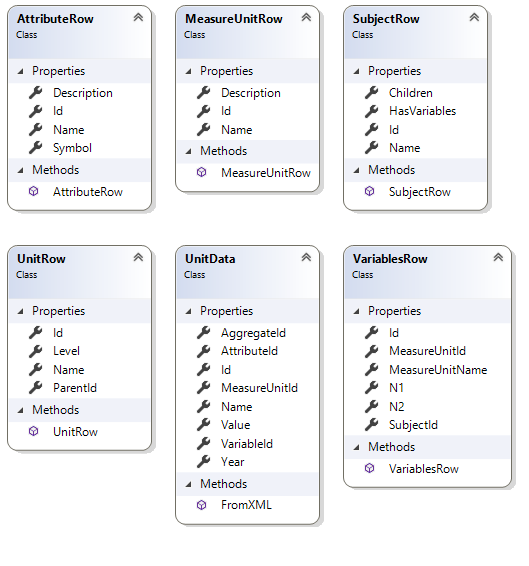
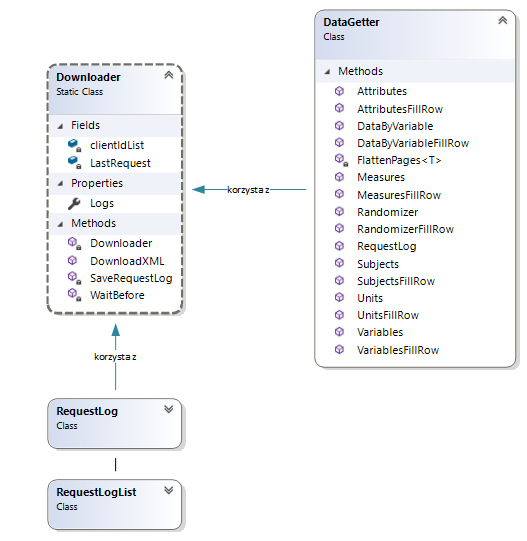


Diagram przedstawiający klasy pomocnicze oraz główną klasę wykorzystywaną po stronie SQL Server.



# Opis klas encji

Są to klasy reprezentujące pojedyncze encje bazodanowe. Każda instancja klasy reprezentuje jeden wiersz, który zwraca funkcja tabelaryczna bazodanowa. Są to proste obiekty DTO (data transfer object). Rola tych obiektów sprowadza się konwersji danych z odpowiedniej metody API REST Banku Danych Lokalnych i przechowywania tych danych w pamięci RAM na okres ich przetwarzania. Dane zapisane w klasie służą więc jako pośredniczące – odbierając z serwisu BDL i przekazując je do SQL Servera.

## AttributeRow

Klasa reprezentuje atrybut. Atrybutem w sensie BDL jest opis specyficznych sytuacji w danych, które są powiązane z każdą wartością liczbową.

|  |  |  |
| --- | --- | --- |
| Właściwość | Typ | Opis |
| Id | Int | Unikalny identyfikator |
| Name | String | Nazwa atrybutu |
| Symbol | String | Symbol atrybutu |
| Description | String | Opis atrybutu |

## MeasureUnitRow

Klasa reprezentuje jednostkę miary. Zwykle jest to jednostka ze standardowego układu jednostek SI, ale poza nimi występują również jednostki walutowe (zł) lub opisujące liczby (promil).

|  |  |  |
| --- | --- | --- |
| Właściwość | Typ | Opis |
| Id | Int | Unikalny identyfikator |
| Name | String | Nazwa jednostki miary |
| Description | String | Opis |

## SubjectRow

Klasa reprezentuje ogólnie pojęty temat. Temat opisuje zgrubnie zbiór danych reprezentowanych – jest to tzw. metadana. Tematy są hierarchiczne, tj. dzielą się na nadrzędne oraz podrzędne. Każdy temat najwyższego poziomu zawiera listę tematów niższego poziomu. Tematy charakteryzują się również tym, że mogą posiadać zmienne – te które mogą posiadać zmienne są tematami „liśćmi” – nie mają tematów podrzędnych, ale można pobierać zmienne, których temat dotyczy – w przeciwieństwie do tematów, które „liścmi” nie są.

|  |  |  |
| --- | --- | --- |
| Właściwość | Typ | Opis |
| Id | Int | Unikalny identyfikator |
| Name | String | Nazwa tematu |
| Children | String | Lista tematów podrzędnych (oddzielona przecinkami) |
| HasVariables | Boolean | Czy temat posiada zmienne |

## UnitRow

Klasa reprezentuję jednostkę terytorialną. Przykładem jednostki jest województwo, gmina, powiat, miejscowość, itp. Jednostki naturalnie układają się w strukturę hierarchiczną – województwa, następnie powiaty, gminy, itd.

|  |  |  |
| --- | --- | --- |
| Właściwość | Typ | Opis |
| Id | Int | Unikalny identyfikator |
| Name | String | Nazwa jednostki (nazwa województwa, powiatu, itd.) |
| ParentId | String | Jednostka terytorialna nadrzędna |
| Level | Int | Poziom zagnieżdżenia jednostki |

## VariablesRow

Klasa reprezentująca zmienną – w sensie Banku Danych Lokalnych. Definicja zmiennej[[1]](#footnote-1)

wielowymiarowe cechy reprezentujące określone zjawisko, z określonym obowiązywaniem w latach i na konkretnym poziomie jednostek, np. liczba pracujących dla wieku '20-26' i płci 'mężczyźni'

Każda zmienna przynależy do jakiegoś tematu. Dany temat może zawierać wiele zmiennych.

|  |  |  |
| --- | --- | --- |
| Właściwość | Typ | Opis |
| Id | Int | Unikalny identyfikator |
| SubjectId | String | Id tematu do którego przynależy zmienna |
| N1 | String | Nazwa ogólna zmiennej |
| N2 | String | Szczegóły zmiennej (np. dodatkowa dookreślająca nazwa) |
| MeasureUnitId | Int | Id jednostki miary |
| MeasureUnitName | String | Nazwa jednostki miary |

## UnitData

Klasa reprezentująca pojedynczy wiersz konkretnych już danych. W terminologii hurtownii danych można poprzez analogię rozumieć tę klasę jako reprezentującą pojedynczy wiersz z tabeli faktów. Mamy zatem odniesienia do wszystkich wyżej wymienionych klas, jak i również konkretne wartości danych. Klasa przyjmuje dowolne wartości z Banku Danych Lokalnych.

|  |  |  |
| --- | --- | --- |
| Właściwość | Typ | Opis |
| Id | Int | Unikalny identyfikator |
| Name | String | Nazwa |
| Year | String | Rok w którym występowała dana |
| Value | string | Wartość |
| VariableId | int | Id zmiennej |
| MeasureUnitId | int | Id jednostki miary |
| AggregateId | int | Id poziomu agregowania danych |
| AttributeId | int | Id atrybutu |

# Opis klas reprezentujących logikę biznesową

Poniżej przedstawiono klasy narzędziowe i klasy realizujące logikę biznesową.

## Downloader

Klasa statyczna której celem jest połączenie się, z użyciem protokołu HTTPS, do serwisu REST Banku Danych Lokalnych. Wymaganiem takiego połączenia jest podanie tokena uwierzytelniającego (w nagłówku HTTP), który klasa dostarcza z listy tokenów. Ze względu na limity połączeń dla pojedynczego tokena, klasa posiada listę kilku tokenów, którymi się uwierzytelniamy (korzystamy tu z klasy NeverendingList). Dodatkowo, ze względu na ograniczenia serwisu API, przed każdym żądaniem HTTP czekamy określony czas – nie przekroczymy w ten sposób limitów.

## RequestLog, RequestLogList

Klasy przechowujące informacje o wykonanych do tej pory żądaniach REST do serwisu BDL. Obsługują serializację i deserializację z pliku XML zapisywanego lokalnie na dysku serwera. Dzięki temu mamy możliwość śledzenia zapytań nawet i po restarcie SQL Servera.

## DataGetter

Centralna klasa, która udostępniana jest na serwerze. Metody tej klasy, które są widoczne po stronie bazy danych opatrzone są adnotacją SqlFunction. Większość z tych funkcji działa po stronie bazodanowej jako funkcja tabelaryczna zwracająca określoną tabelę z jasno określonymi polami.

Lista metod niniejszej klasy:

1. RequestLog. Metoda zwraca listę zapisanych żądań do API Banku Danych Lokalnych.

Po stronie SQL Server widoczna jako procedura, która pokazuje listę w postaci kolejnych wierszy w zakładce Messages.

1. Subjects. Metoda zwraca listę tematów – w zależności od parametru. Jeśli parametr jest null, to metoda zwraca tematy nadrzędne w formie listy encji SubjectRow.
2. Variables. Metoda zwraca listę zmiennych dla danego tematu w postaci listy encji VariablesRow.
3. Measures. Metoda zwraca listę wszelkich jednostek miar używanych w BDL; postać listy encji MeasureUnitRow.
4. Attributes. Metoda zwraca listę wszelkich atrybutów w formie listy encji AttributeRow.
5. DataByVariable. Metoda zwraca listę konkretnych już danych – dla podanego id zmiennej, zakresu lat i zasięgu „w głąb” (poziomu zagnieżdżenia wg jednostek terytorialnych). Forma listy encji UnitData.

Listy encji odczytywane są przez SQL Server jako kolejne wiersze tabeli, którą funkcja zwraca.

# Inne klasy

Wykorzystuję własnoręcznie napisaną klasę NeverendingList, która reprezentuje interfejs „niekończącej się listy”. Pobierając kolejny element listy wewnętrzny wskaźnik przesuwany jest o jeden do przodu. Gdy dojdziemy do końca listy, wskaźnik przesuwany jest na początek. Wykorzystanie tej klasy znajduje się w klasie Downloader.

# Działanie mechanizmów

Klasa DataGetter korzysta z metod klasy Downloader. Klasa Downloader korzysta z kolei z klasy RequestLog, która zależy od RequestLogList. Mamy weń również klasę NeverendingList.

# Przykłady wywołań

Przykłady wywołań znajdują się w projekcie testów jednostkowych.

# Netografia

1. <https://api.stat.gov.pl/Home/BdlApi>, dostęp 24 listopada 2019

1. <https://bdl.stat.gov.pl/api/v1/variables/metadata?format=json>, dostęp 24 XI 2019 [↑](#footnote-ref-1)