***Podstawy, nawiązywanie połączenia z bazą danych, zapisywanie rekordów do bazy danych***

**Instrukcja**

1. **Cel ćwiczenia**

Zapoznanie się z podstawowymi zagadnieniami dotyczącymi łączenia się z bazą danych (Microsoft SQL Server) z poziomu aplikacji napisanej w C# (WinForms – Net.Framework, WPF, WinFormsc- .Net Core lub też winforms .Net 6) – wedle indywidualnych preferencji. Operacje Wykorzystanie dostawcy danych ADO.NET – SqlClient ( **System.Data.SqlClient** ) oraz ORM np. EF6.

Główne zagadnienia realizowane w części teoretycznej ćwiczeń to:

* Omówienie podstawowych klas i metod niezbędnych do prawidłowego zainicjalizowania połączenia z bazą danych
* Ustanowienie połączenia
* Przechwytywanie błędów (**SqlException**)
* Wykonywanie podstawowych operacji z grupy **DQL** – Data Query Language.
* Zamykanie połączenia
* Wykorzystywanie bloku ***using.***

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/language-reference/keywords/try-catch-finally>
* <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>

1. **Wiadomości teoretyczne**

W celu przygotowania się do części praktycznej ćwiczenia, należy odnaleźć w udostępnionej dokumentacji (linki podane w punkcie 3) oraz innych źródłach definicję wybranych pojęć. Wybrać te, które w ocenie ucznia są najbardziej przystępne oraz najlepiej wyjaśniają wybrane zagadnienie.

* Blok **try-catch-finally** przyczyny stosowania, jakie są korzyści oraz przykłady użycia
* Klasa SqlCommandi jej główne składowe
  + W szczególności zwrócić uwagę na składowe takie jak:
  + Transaction
  + Connection
  + CommandText
  + Parameters
  + ExecuteNonQuery()
  + ExecuteReader()
  + ExecuteScalar()
* Omówić connection stringjakie składowe mogą się w nim zawierać.
* GłównepoleceniadlaSqlConnection:
  + Open()
  + BeginTransaction()
  + Close()
* Klasa SqlDataReaderi jej główne składowe
* Klasa SqlException
* **Blokusing** przyczyny stosowania, jakie są korzyści oraz przykłady użycia, nie mylić z dyrektywą using stosowaną w części deklaracji pliku. Chodzi tutaj o blok kodu (tak samo jak na przykład **try-catch-finally**.

1. **Przebieg ćwiczenia**

Wykorzystując powyższa bazę teoretyczną, oraz wiedzę bazując na informacjach odnalezionych w Internecie, należy utworzyć nową bazę danych a w niej tabele „Kody\_Pocztowe” zawierającą następujące kolumny: **Kod\_Pocztow**y, **Adres**, **Miejscowosc**, **Wojewodztwo**, **Powiat**.

Zadaniem jest napisanie prostego programu (może to być aplikacja konsolowa) który to wczyta zawartość pliku csv (kody.csv – dołączony w osobnym pliku) do pamięci a następnie zapisze go w bazie danych, równocześnie dokonując pomiaru całkowitego czasu zapisywania danych w bazie, przy czym przyjąć należy, że początek pomiaru czasu jest w momencie, gdy cały plik jest już wstępnie wczytany do pamięci i rozpoczyna się procedura zapisu (przed dokonaniem pierwszego wpisu oraz przed ustanowieniem połączenia z bazą danych), a za koniec pomiaru przyjąć moment, gdy zostanie zapisany ostatni rekord z pliku oraz połączenie do bazy zostanie zamknięte.

Ćwiczenie należy wykonać kilkukrotnie, każdorazowo zmieniając sposób zapisywania danych w bazie\*. Za każdym razem do ćwiczenia wykorzystywany ma być dokładnie ten sam plik oraz ta sama tabela docelowa w bazie danych, z której też każdorazowo przed przystąpieniem do testów należy usunąć wszystkie dane. Opracować wyniki, w raz z omówieniem zależności czasu realizacji w zależności od użytej metody. Dla każdej z wykorzystanych metod przedstawić takie dane jak całkowity czas zapisu danych, wyliczony średni czas zapisu na pojedynczy rekord. Opcjonalnie można też przeprowadzić testy w trybie debbugowania w Visual Studio i uwzględnić zużycie CPU oraz RAM. Wszystkie dane zestawić w jednej tabeli porównawczej a w oparciu o uzyskane wyniki wyprowadzić stosowne wnioski wraz z analizą odnotowanych różnic zmierzonych wartość starając się oczywiście wyjaśnić z czego różnice te wynikają.

\*W ramach testowanych metod należy **obowiązkowo** uwzględnić 2 poniższe scenariusze:

* Metoda zapisu dotyczy pojedynczego rekordu (jako parametr metody przekazujemy jeden rekord) a wewnątrz metody otwierane jest połączenie do bazy danych, wykonywany jest zapis rekordu a następnie połączenie to jest zamykane. Tak więc następuje tyle wywołań metody ile jest rekordów w bazie danych.
* Metoda zapisu dotyczy całej kolekcji, a więc przekazywane są do niej wszystkie rekordy, wewnątrz metody na samym początku ustanawiane jest połączenie z bazą danych, następuje przesłanie wszystkich rekordów, a następnie rozłączenie i wyjście z metody.

Dodatkowo należy wykorzystać co najmniej 2 inne wybrane przez ucznia sposoby zrealizowania zapisu danych do bazy, np.: **SqlBulkCopy,** lub teżWykorzystując **ORM**np. **EF\*\*.** Można wykorzystać inny ORM lub też zaproponować inne rozwiązanie, które w ocenie ucznia mogłoby być wydajniejsze czasowo.

\*\*Przy czym w ramachw EF zadanie to też może zostać wykonane na kilka sposobów:

* **DbContext.SaveChanges()** wywoływane po każdej operacji dodania obiektu do kontekstu
* **DbContext.SaveChanges()** wywoływane na końcu (po dodaniu do kontekstu wszystkich obiektów)
* **DbContext.SaveChanges()**wywoływane co określoną liczbę rekordów (paczkowanie)

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