Wprowadzenie do systemów rozproszonych

Sprawozdanie z ćwiczenia 4b + 4c

Persystencja (kolejka)

Persystencja (kanał)

Mateusz Jabłoński

1. Wstęp

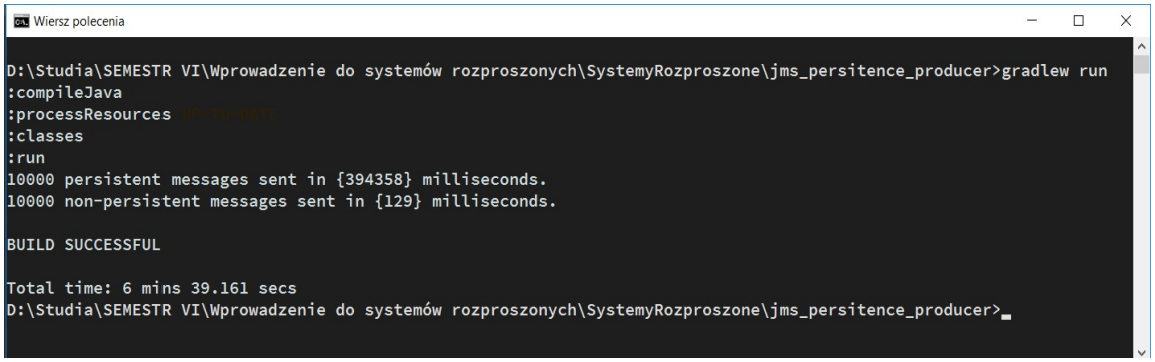
Celem ćwiczenia napisanie dwóch programów. Celem pierwszego jest wysłanie na kolejkę 20 tys. komunikatów w dwóch transzach po 10 tys. każda. Pierwsza część powinna mieć ustawione „delivery mode” na PERSISTENCE, natomiast druga na NON_PERSISTENCE. Dugi program ma za zadanie odebrać z kolejki komunikaty, wysłane na nią wcześniej przez pierwszy program.

W przypadku ćwiczenia 4c podobnie jak w ćwiczeniu 4b, należy napisać dwa programy, jednak zamiast kolejki użyć mechanizmu publish/subscribe .

2. Rozwiązanie

- Pojęcia:
  - PERSISTENCE – parametr określający czy komunikaty maja być trwale przechowywane przez menadżer kolejek i nie mogą zaginąć. Komunikaty dzielimy na persystentne i niepersystentne. Jeśli komunikat jest persystentny oznacza to, że będzie on logowany w pamięci trwalej, co powoduje to, ze nie ma możliwości jego utraty. Komunikat taki przetrwa nawet restart menadżera kolejek. W związku z tym, że komunikaty takie musza być logowane ich procesowanie trwa zdecydowanie dłużej niż komunikatów niepersystentnych. Komunikaty niepersystentne z kolei nie gwarantują dostarczenia do adresata ze względu na swoja ulotna naturę. Nie są one logowane tak jak komunikaty persystentne, ale za to są o wiele szybciej procesowane.
  - Trwała subskrypcja (durable subscription) – umożliwia zbieranie komunikatów przesyłanych na kanał pod nieobecność subskrybenta, a następnie przekazanie mu ich w momencie ponownego połączenia z kanałem. Trwała subskrypcja generuje dodatkowy koszt, oraz może w danym momencie posiadać co najwyżej jednego aktywnego subskrybenta.

- Zrzuty ekranu
  - **Ćwiczenie 4b**



Wysłanie na kolejkę 20 tys. komunikatów. Pierwsze 10 tys. jako PERSISTENT zajęło ok. 3000 razy więcej czasu niż wysłanie takiej samej ilości komunikatów, tyle że jako NON_PERSISTENT.

Queues:

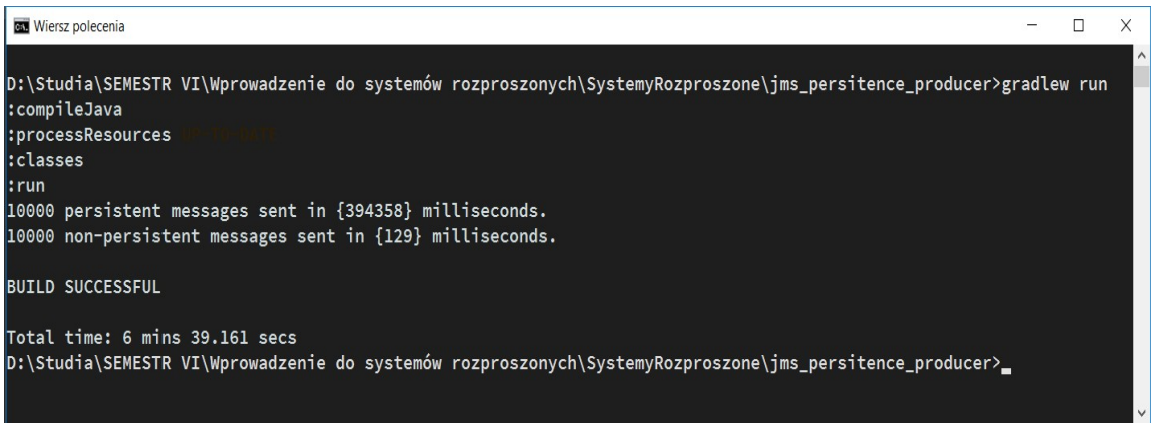
Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
MATEJABLONSKI.QUEUE	20000	0	20000	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete

Stan kolejki po wysłaniu 20 tys. komunikatów. Znajduje się na niej 20 tys. komunikatów czyli tyle samo ile zostało dodanych od chwili uruchomienia brokera.

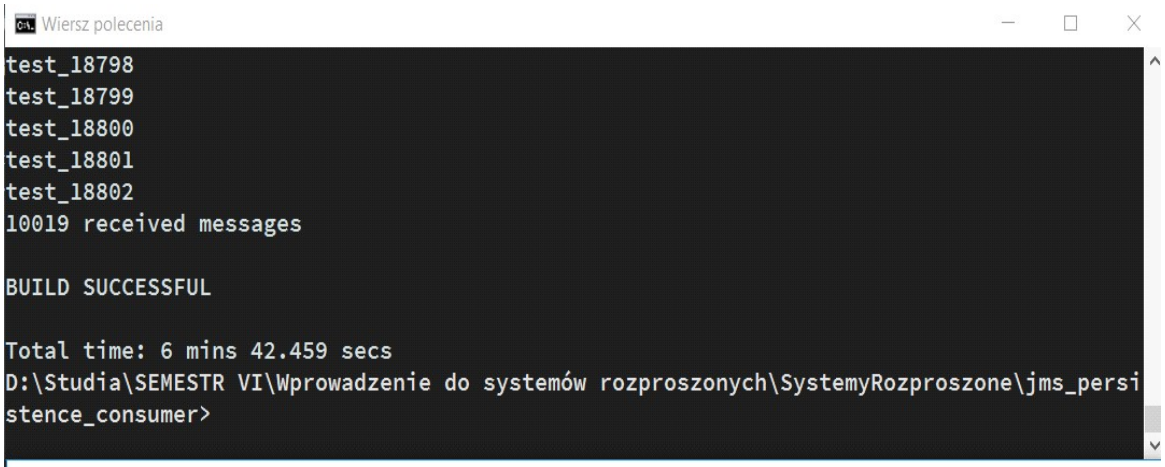
Queues:

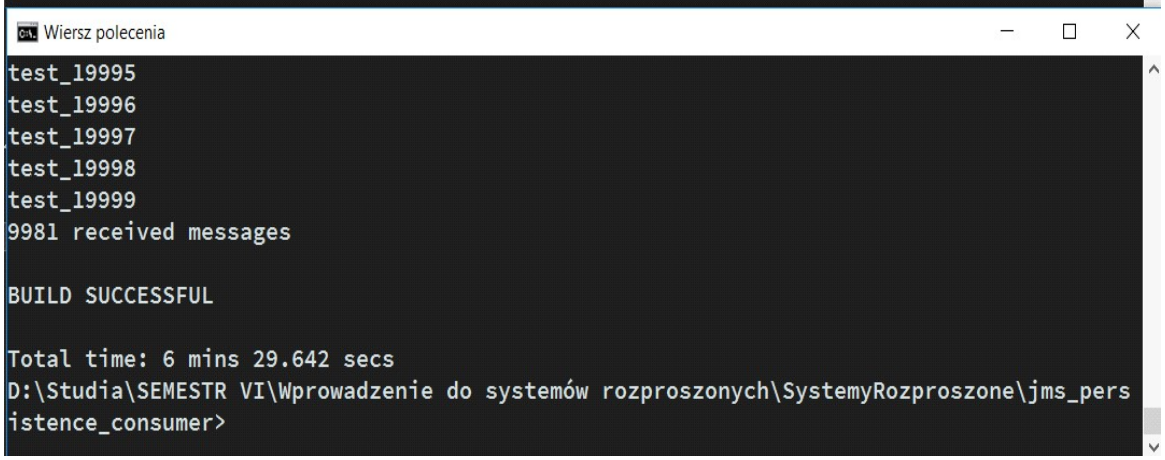
Name	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued	Views	Operations
MATEJABLONSKI.QUEUE	10000	0	0	0	Browse Active Consumers Active Producers atom rss	Send To Purge Delete

Stan kolejki po restarcie brokera. Zostało tylko 10 tys. komunikatów, czyli tylko te które zostały ustawione jako PERSISTENT. Reszta została trwale usunięta.



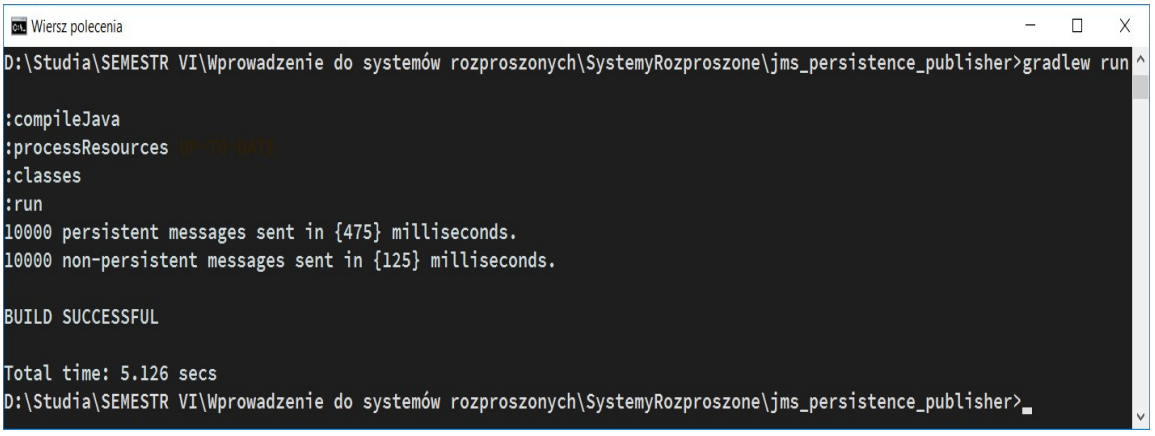
Reszta komunikatów zostaje prawidłowa odebrana przez consumer'a.





W przypadku uruchomienia dwóch consumer'ów jednocześnie część komunikatów została odebrana przez pierwszego, część przez drugiego. Łączna liczba komunikatów jest równa liczbie wysłanych, żadna wiadomość nie została odebrana przez obu jednocześnie, co jest bezpośrednim skutkiem sposobu działania kolejki.

- **Ćwiczenie 4c**



Również i w tym przypadku wysyłanie komunikatów jako NON_PERSISTENT jest znacznie szybsze, jednak tym razem różnica jest 1000 krotnie mniejsza.

Czas wysłania komunikatów bez trwałego zapisania w pamięci jest niemal ten sam, w odróżnieniu do czasu wysłania z ustawionym trybem PERSISTENCE, który w stosunku do kolejki jest 1000 razy krótszy.

Topics

Name ↑	Number Of Consumers	Messages Enqueued	Messages Dequeued	Operations
ActiveMQ.Advisory.Connection	0	2	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Producer.Topic.MATEJABLONSKI....	0	2	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Topic	0	1	0	Send To Active Subscribers Active Producers Delete
MATEJABLONSKI.TOPIC	0	20000	0	Send To Active Subscribers Active Producers Delete

Po uruchomieniu publisher'a na kanale nie jest zarejestrowany żaden odbiorca, do kanału zostało dodanych 20 tys. komunikatów, oraz żaden z komunikatów nie został odebrany z kanału.

Topics

Name ↑	Number Of Consumers	Messages Enqueued	Messages Dequeued	Operations
ActiveMQ.Advisory.Connection	0	5	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Consumer.Topic.MATEJABLONSKI....	0	3	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.MasterBroker	0	1	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Producer.Topic.MATEJABLONSKI....	0	2	0	Send To Active Subscribers Active Producers Delete
MATEJABLONSKI.TOPIC	1	20000	0	Send To Active Subscribers Active Producers Delete

Widok konsoli webowej, podczas gdy uruchomiony jest tylko subscriber. Komunikaty, które zostały wcześniej dodane na kolejkę nie są odbierane przez aktywnego subscriber'a.

Topics

Name ↑	Number Of Consumers	Messages Enqueued	Messages Dequeued	Operations
ActiveMQ.Advisory.Connection	0	9	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Consumer.Topic.MATEJABLONSKI....	0	6	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.MasterBroker	0	1	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Producer.Topic.MATEJABLONSKI....	0	3	0	Send To Active Subscribers Active Producers Delete
MATEJABLONSKI.TOPIC	0	118360	0	Send To Active Subscribers Active Producers Delete

Aktywny jedynie subscriber. Następuje dodawanie komunikatów do kolejki, przy jednoczesnym braku ich

Topics

Name ↑	Number Of Consumers	Messages Enqueued	Messages Dequeued	Operations
ActiveMQ.Advisory.Connection	0	2	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Consumer.Topic.MATEJABLONSKI....	0	1	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Producer.Topic.MATEJABLONSKI....	0	1	0	Send To Active Subscribers Active Producers Delete
ActiveMQ.Advisory.Topic	0	1	0	Send To Active Subscribers Active Producers Delete
MATEJABLONSKI.TOPIC	1	51313	30186	Send To Active Subscribers Active Producers Delete

Widok konsoli webowej w przypadku, kiedy uruchomione są obie aplikacje, publisher oraz subscriber. Dostarczane na kanał komunikaty są odbierane przez consumer'a.

3. Wnioski

Jeżeli tryb dostarczania wiadomości NON_PERSISTENT powoduje że komunikaty nie są zapisywane na stałe w pamięci i zostają utracone w przypadku awarii brokera lub jego wyłączenia. Można temu zapobiec zmieniając tryb na PERSISTENT, jednak niesie to ze sobą znacznie wydłużenie czasu dostarczanie wiadomości. Przy małej liczbie komunikatów będzie to bez znaczenia, a zapewnia że żaden komunikat nie zostanie utracony, jednak przy bardzo dużej liczbie wysyłanych wiadomości może przynieść znaczny narzut czasowy.

W przypadku odbierania komunikatów z kolejki przez więcej niż jednego konsumenta, każdy z nich otrzyma tylko część wysłanych komunikatów, przy czym ich łączna liczba będzie równa liczby wysłanych. Jest to spowodowane tym, że każdy komunikat w kolejce może zostać odebrany tylko przez jednego konsumenta.

Jeżeli wykorzystujemy mechanizm kanału i chcemy, aby subscriber mógł otrzymywać również te komunikaty, które zostały wysłane na kanał, gdy subscriber był wyłączony należy wykorzystać mechanizm durable subscription. Jeżeli, w takim wypadku w temacie będą publikowane komunikaty, a subscriber będzie nieaktywny, będą one zbierane a następnie przesłane do subscriber'a w momencie jego ponownego połączenia.

4. Literatura

1. Java EE 6 Zaawansowany przewodnik, praca zbiorowa:
   - Zagadnienia technologii JMS, Tworzenie wydajnych aplikacji JMS – Określanie trwałości komunikatu: s. 356-357.
   - Zagadnienia technologii JMS, Tworzenie wydajnych aplikacji JMS – Użycie zaawansowanych mechanizmów niezawodności, Tworzenie trwałych subskrypcji: s. 359-361.
2. https://docs.oracle.com/cd/E19798-01/821-1841/bncfy/index.html
3. https://docs.oracle.com/cd/E19798-01/821-1841/bncgd/index.html