Java Messaging Service

Katarzyna Herman, Rafał Szęszoł

1. Wprowadzenie

JMS – Java messaging service – jak wskazuje nazwa jest to API służące do wymieniania komunikatów, a więc działające w peer-to-peer.

JMS umożliwia 2 sposoby komunikacji:

- asynchroniczną
- synchroniczną

Aplikacje mogą współdziałać poprzez wymianę komunikatów, przy czym:

- aplikacja-producent wysyła komunikat,
- aplikacja-konsument odczytuje komunikat,
- obie aplikacje nie muszą działać równocześnie,
- żadna z nich nie musi nic "wiedzieć" o budowie, kodzie itp. drugiej.

Współdziałanie aplikacji poprzez wymianę komunikatów ma tę zaletę, że uniezależnia kody programów od siebie ("loose coupling").

Ważną cechą JMS jest niezawodność – API gwarantuje, że każdy komunikat zostanie dostarczony i to dokładnie raz.

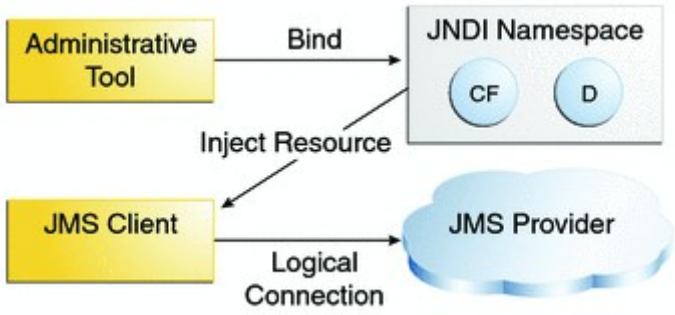
We współpracy z Java EE daje nam to następujące możliwości:

- korzystając z komponentów EJB aplikacja klienta może wysyłać i otrzymywać synchronicznie lub asynchronicznie wiadomości,
- Message-driven beany pozwalają na asynchroniczne odbieranie wiadomości, a także na równoległe przetwarzanie w nich wiadomości,
- operacje otrzymywania i wysyłania wiadomości mogą brać udział w rozproszonych transakcjach, co pozwala na przeprowadzeniu operacji na bazie danych i wykorzystaniu JMS w jednej transakcji.

2. Architektura JMS

Składowe aplikacji JMS

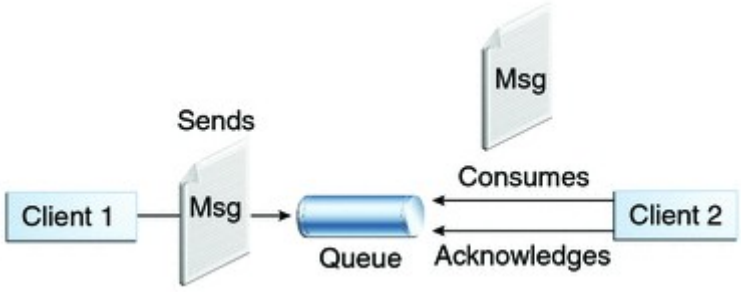
- Serwis (JMS provider) implementuje interfejsy i dostarcza narzędzi administrowania,
- Klienci - tworzą, wysyłają i odbierają komunikaty,
- Komunikaty – obiekty przesyłane między klientami
- Obiekty administrowane – miejsca docelowe (destinations) oraz fabryki połączeń (są zwykle tworzone i konfigurowane przez narzędzia administracyjne, ale również istnieje programistyczne API do tych celów)



Miejsca docelowe i domeny

Miejsce docelowe JMS jest miejscem, w którym nadawca umieszcza wiadomości, a odbiorca – odczytuje. Wyróżniamy dwie możliwe domeny (rodzaje miejsc docelowych):

Kolejka (queue) - za pośrednictwem kolejki komunikatów, nadawca wysyła do określonej, nazwanej kolejki (miejsca docelowego), a odbiorca - odbiera z niej komunikaty (jest to sposób komunikacji P2P – point to point).



Kolejkę charakteryzują następujące własności:

- każdy komunikat może być odebrany tylko przez jednego odbiorcę,

- odbiorca może odebrać wiadomość niezależnie od tego czy nadawca działa czy też już zakończył działanie,
- zasadą jest, że odbiorca potwierdza odebranie wiadomości, co zapewnia, że nie będzie mu ona przysłana po raz kolejny.

Temat (topic) - mechanizm publikacji i subskrypcji.

Temat charakteryzują następujące własności:

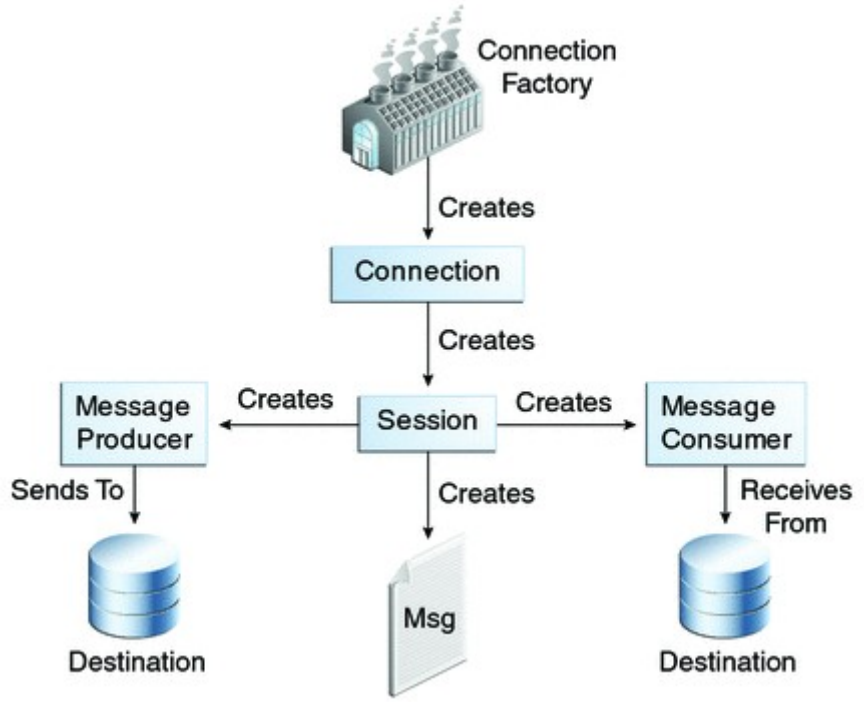
- każda wiadomość może mieć wielu odbiorców (subskrybentów tematu),
- w danym temacie może publikować wielu nadawców,
- odbiorca może odbierać tylko te wiadomości z danego tematu, które zostały opublikowane po zapisaniu się przez niego do subskrypcji,
- w zależności od typu subskrybenta możliwe jest również otrzymywanie wiadomości które zostały przysłane, w czasie gdy był on nieaktywny („durable subscriptions”).

Wersja 1.1 JMS pozwala na oprogramowanie obu typów komunikacji za pomocą tych samych interfejsów.

Odbiór komunikatów:

- asynchronicznie – wymaga zarejestrowania MessageListnera, który odbierze komunikaty (odbywa się to przez zaimplementowanie interfejsu o tej nazwie i metody onMessage w nim zawartej)
- synchronicznie – poprzez wywołanie metody receive w subskrybencie lub odbiorcy.

Idea aplikacji JMS



ConnectionFactory - tworzy połączenie do dostawcy serwisu JMS.
Destination - określa destynację nadawanych i odbieranych wiadomości.
Session – odpowiada za tworzenie wiadomości oraz obiektów które odbierają i wysyłają komunikaty.

3. Najważniejsze klasy/interfejsy:

Message – interfejs po którym dziedziczą wszystkie możliwe wiadomości wysyłane w JMS

InitialContext + lookup(„”) - klasa pozwalająca na rozwiązanie nazw elementów JMS. W funkcji lookup jako argument podaje się referencję obiektu do którego chcemy się odnosić

Queue – obiekt wyżej opisanej kolejki

TopicSession, QueueSession – obiekty sesji, w zależności od wybranej domeny.

Topic – obiekt wyżej opisanego tematu.

MessageListner – interfejs służący do asynchronicznego odbierania wiadomości

QueueSender, QueueReceiver – nadawca i odbiorca dla kolejki. Tworzy się go dla danej sesji, jak parametr konstruktora podając kolejkę do której ma wysyłać/z której ma odbierać/

TopicPublisher, TopicSubscriber – analogicznie jak dla kolejki. Obiekty do publikacji i pobierania komunikatu dla tematu.

MessageDriven – jest to Enterprise Java Bean który pozwala aplikacjom Java EE na asynchroniczne przetwarzanie wiadomości