# Message Oriented Middleware

Systemy kolejkowania komunikatów

## Plan

1. Koncepcja
   - paradygmat kolejkowania (punkt-punkt)
   - paradygmat publish/subscribe
2. Model systemu
3. Przykłady rozwiązań
4. JMS

## Cechy MOM

- Uniezależnienie funkcjonowania składników aplikacji od dostępności informacji o interfejsach innych składników
- Uniezależnienie funkcjonowania warstwy komunikacyjnej (kanału komunikacyjnego) od działania (obecności) komunikujących się procesów ⇒ komunikacja nieustanna
- Mechanizm komunikacji pośredniej
  - oparty na identyfikacji miejsca pośredniczącego (tzw. skrzynki, a nie adresu nadawcy/odbiorcy)
  - komunikujące się strony nie muszą znać się wzajemnie
- Łatwość wdrożenia komunikacji asynchronicznej

## Koncepcja komunikacji oparta na kolejkowaniu

- Komunikacja *punkt-punkt*
- Organizacja warstwy komunikacyjnej w postaci systemu kolejek (ang. queue) {skrzynka ≡ kolejka} realizowanych w oparciu o zasoby pamięci, w tym pamięci dyskowej (gwarancja trwałości na wypadek awarii ⇒ niezawodność)
- Udostępnienie mechanizmów komunikacji polegających na:
  - umieszczeniu komunikatów w kolejkach,
  - pobieraniu komunikatów z kolejek

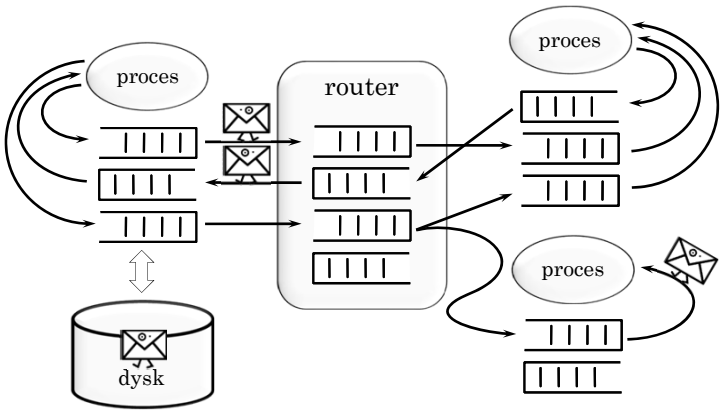
## Paradygmat publish/subscribe

- Komunikacja *jeden do wielu* (potencjalnie *wielu do wielu*)
- Strona publikująca (nadawca) udostępnia treść związaną z określonym tematem (ang. topic) {skrzynka ≡ temat}
- Środowisko komunikacyjne (usługa) przekazuje treść udostępnionych wiadomości odbiorcom (subskrybentom), którzy zarejestrowali (zapisali) się wcześniej na dany temat

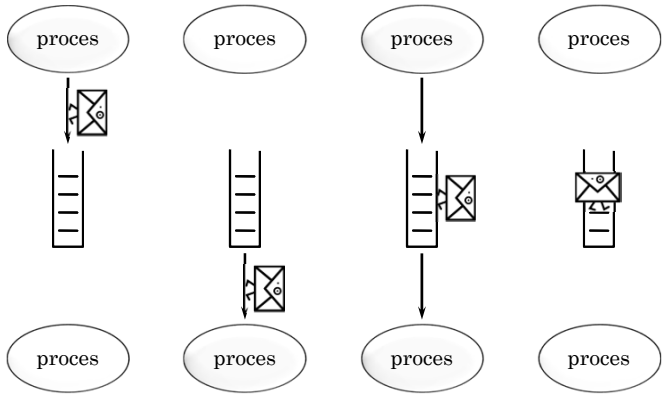
## Model systemu typu MOM — podstawowe pojęcia

- Wiadomość (ang. message) — porcja danych (często z dodatkowymi własnościami) składowana w kolejce
- Kolejka (ang. queue) — miejsce przechowywania wiadomości (komunikatów) {temat ≅ kolejka dla wielu odbiorców, ang. multiconsumer queue}
- Proces (ang. process) — element aplikacji, zlecający operacje dotyczące wiadomości w kolejce
- Zarządca zbioru kolejek — moduł (proces na określonym węźle, dostawca, *broker*), odpowiedzialny za wykonywanie operacji na kolejkach, np. tworzenie, usuwanie, lokalizowanie kolejek, definiowanie atrybutów kolejek itp.)

## Model systemu kolejkowania komunikatów — funkcjonowanie




## Warianty komunikacji



## Przykłady rozwiązań typu MOM

- WebSphere MQ (IBM MQSeries , XMS — Message Service Client)
- Microsoft Message Queuing (MSMQ)
- Usługa IceStorm w systemie ICE
- Java Message Service (JMS) — standard, specyfikacja interfejsu. Implementacje oparte na JMS:
  - Sun Java System Message Queue (OpenMQ — wersja open source) — implementacja JMS
  - Apache ActiveMQ — implementacja JMS
- Oracle Advanced Queueing

# Java Message Service

na podstawie slajdów Cezarego Sobańca

## Historia JMS

- Opracowany w 1998
- Pierwotny cel: dostęp do istniejących systemów kolejkowania wiadomości (tzw. MOM — Message Oriented Middleware, np. IBM MQSeries)
- Integralna część Java EE od wersji 1.3

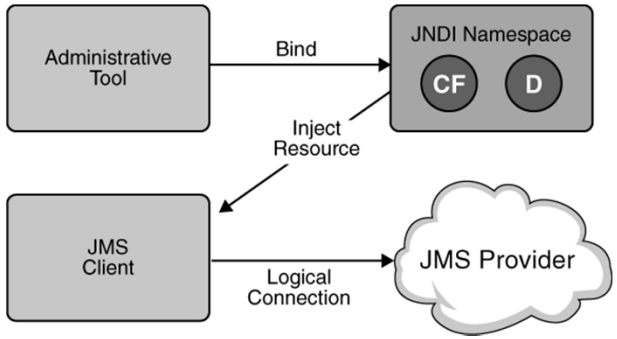
## Komponenty JMS

- Dostawca JMS (ang. JMS provider) — implementacja interfejsów JMS, administracja, sterowanie
- Klienci JMS — aplikacje i komponenty wysyłające i odbierające komunikaty
- Wiadomości — obiekty do przenoszenia informacji
- Obiekty zarządzania (ang. administered objects) – prekonfigurowane obiekty na potrzeby zarządzania:
  - cele (ang. destinations)
  - fabryki połączeń (ang. connection factories)

## Funkcjonalność JMS

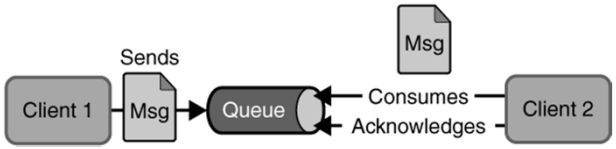
- Nieustanna, niezawodna, asynchroniczna komunikacja międzyprocesowa
- Transakcyjna interakcja z dostawcą JMS
- Modele komunikacji (messaging domains)
  - punkt-punkt (ang. point-to-point)
  - subskrypcji (ang. publish/subscribe)

## Architektura JMS



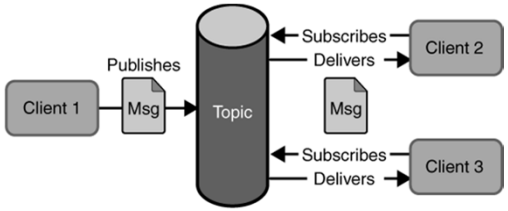

## Model punkt-punkt

- Wysyłanie i odbiór poprzez dostęp do kolejki
- Wiadomości pozostają w kolejce do czasu odbioru lub przedawnienia
- Każda wiadomość ma 1 konsumenta
- Brak ograniczeń czasowych




## Model subskrypcji

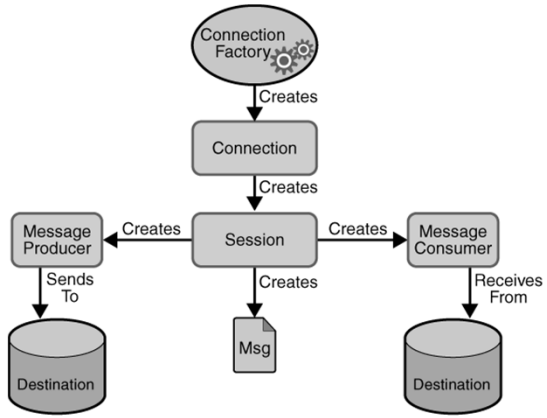
- Każda wiadomość może mieć wielu konsumentów
- Wiadomości są dostarczane do wszystkich aktualnych subskrybentów a następnie są usuwane
- Nie można odczytać wiadomości sprzed subskrypcji
- Trwała subskrypcja (ang. durable subscription) – odbiór wiadomości z okresu nieaktywności klienta



## Model odbioru (konsumpcji) wiadomości

- Konsumpcja synchroniczna – blokująca metoda receive() z (ewentualnym) ograniczeniem czasowym
- Konsumpcja asynchroniczna – odbiornik wiadomości (ang. message listener) – asynchroniczne wywołanie metody onMessage()

## Model programistyczny JMS




## Fabryki połączeń

- Ogólna: ConnectionFactory (interfejs bazowy dla poniższych)
- Dla kolejek: QueueConnectionFactory
- Dla tematów: TopicConnectionFactory

## Destinations

- *queue* w przypadku komunikacji punkt-punkt
- *topic* w przypadku modelu subskrypcji

```
@Resource(mappedName="jms/Queue")
private static Queue queue;

@Resource(mappedName="jms/Topic")
private static Topic topic;
```

## Połączenie (and. connections)

- Reprezentacja wirtualnego połączenia z dostawcą JMS
- Połączenie jest potrzebne do zainicjowania sesji
- Połączenie musi być jawnie zamknięte na końcu aplikacji (zwolnienie zasobów, m.in. otwartych sesji)
- Rozpoczęcie odbioru wiadomości: metoda start()
- Wstrzymanie odbioru wiadomości: metoda stop()

```
Connection connection =
  connFactory.createConnection();
connection.start();
...
connection.close();
```

## Sesje

- Jednowątkowy kontekst do tworzenia i odbioru wiadomości
- Sesje tworzą:
  - wiadomości
  - producentów (nadawców) wiadomości
  - konsumentów (odbiorców) wiadomości

```
Session session =
  connection.createSession(false,
  Session.AUTO_ACKNOWLEDGE);
```

- Pierwszy argument wskazuje czy ma być tworzona transakcja

## Producenci wiadomości

```
@Resource(mappedName="jms/Queue")
private static Queue queue;

MessageProducer producer =
session.createProducer(queue);
producer.send(message);
```

- Wysyłanie do dowolnej kolejki:

```
MessageProducer producer =
session.createProducer(null);
producer.send(queue, message);
```

## Odbiorcy wiadomości

```
@Resource(mappedName="jms/Queue")
private static Queue queue;

MessageConsumer consumer =
session.createConsumer(queue);
Message m1 = consumer.receive();
Message m2 = consumer.receive(1000);
```

- Message listeners:

```
MessageListener myListener = new AListener();
consumer.setMessageListener(myListener);
```

- Listener ma metodę onMessage(Message). Musi obsługiwać wszystkie wyjątki.

## Filtry wiadomości

- Filtr jest wyrażeniem zapisywanym jak warunki w SQL92
  - typ = 'wyniki' and id = '120'
- Wyrażenie odwołuje się do właściwości wiadomości
- Filtr może być parametrem tworzenia konsumenta wiadomości

## Wiadomości

- Wiadomości składają się z
  - nagłówka
  - listy właściwości
  - ciała
- Standardowe właściwości (przechowywane w nagłówku)
  - identyfikator JMSMessageID
  - odbiorca JMSDestination
  - znacznik czasowy JMSTimestamp
  - priorytet JMSPriority
  - typ JMSType

## Zawartość wiadomości (body)

- TextMessage – wiadomość tekstowa (np. dok. XML)
- MapMessage – zbiór par nazwa-wartość (String i typ prymitywny)
- BytesMessage – nieinterpretowany strumień bajtów
- StreamMessage – strumień wartości prymitywnych typów
- ObjectMessage – serializowalny obiekt Javy
- Message – pusta zawartość

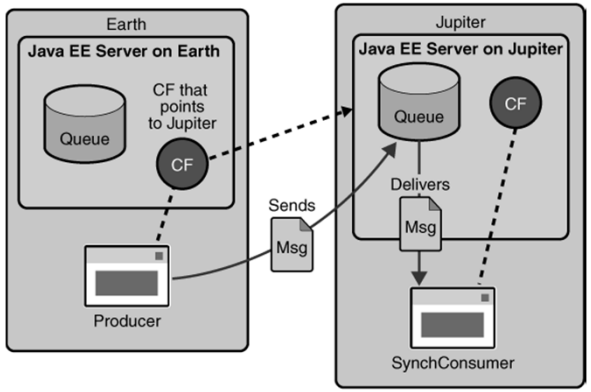
```
TextMessage message = session.createTextMessage();
message.setText(”Ala ma kota”);
producer.send(message)
...
Message m = consumer.receive();
if (m instanceof TextMessage) { ... m.getText(); }
```

## Przeglądanie zawartości kolejki

- Interface QueueBrowser
- Możliwość zastosowania filtru
- Nie można przeglądać topic’ów (wiadomości znikają)

```
QueueBrowser browser =
  session.createBrowser(queue);
Enumeration msgs = browser.getEnumeration();
while (msgs.hasMoreElements()) {
  Message m = (Message)msgs.nextElement();
  ...
}
```

## Przesyłanie wiadomości między systemami




## Niezawodność komunikacji

- Potwierdzenia wiadomości
- Trwałość komunikatów – awarie dostawców JMS
- Priorytety komunikatów
- Czas życia komunikatów
- Tymczasowe kolejki

## Potwierdzenia

- Po odbiorze wiadomości
- Po przetworzeniu wiadomości
- Po odbiorze potwierdzenia
- Wiadomości są automatycznie potwierdzane po zakończeniu transakcji
- Wycofanie transakcji → ponowne dostarczenie

- AUTO_ACKNOWLEDGE – po odbiorze
- CLIENT_ACKNOWLEDGE – jawne potwierdzenie wszystkich odebranych wiadomości w ramach sesji
- DUPS_OK_ACKOWLEDGE – leniwe potwierdzanie z możliwością powstawania duplikatów

## Potwierdzenie

- Wiadomości niepotwierdzone przed końcem sesji są dostarczane ponownie (przy kolejnym połączeniu)
- Przetrzymywanie niepotwierdzonych wiadomości dla trwałych subskrypcji

Potwierdzenie tylko przetworzonych wiadomości:

- asynchroniczny odbiór i tryb AUTO_ACKNOWLEDGE
- odbiór synchroniczny i tryb CLIENT_ACKNOWLEDGE
- odbiór synchroniczny w trybie AUTO_ACKNOWLEDGE powoduje natychmiastowe potwierdzanie (przed przetworzeniem)

## Trwałość wiadomości

- Tryb PERSISTENT – każda wiadomość jest rejestrowana w pamięci trwałej (tryb domyślny)
- Tryb NON_PERSISTENT – wiadomość może zostać utracona w przypadku awarii dostawcy JMS (większa wydajność)

- Własność trwałości może być ustawiana dla producenta wiadomości lub dla pojedynczej wiadomości

```
producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
producer.send(msg, DeliveryMode.NON_PERSISTENT,3,10000);
```

## Priorytety wiadomości

- Priorytet może być ustawiany dla producenta wiadomości lub dla pojedynczej wiadomości
- 0 – najniższy priorytet, 9 – najwyższy, domyślnie 4
- Priorytet określa preferencje i nie decyduje o bezwzględnej kolejności dostarczania

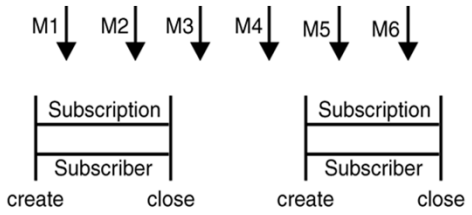
```
producer.setPriority(5);
producer.send(msg,
DeliveryMode.NON_PERSISTENT,5,10000);
```

## Przedawnianie wiadomości

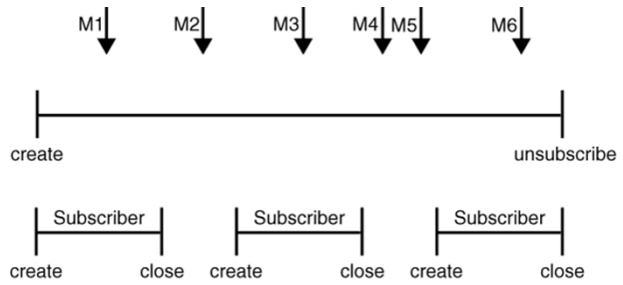
- Czas życia może być ustawiany dla producenta wiadomości lub dla pojedynczej wiadomości
- Domyślnie wiadomości nie ulegają przedawnieniu
- Czas życia 0 oznacza brak przedawniania

```
producer.setTimeToLive(10000);
producer.send(msg,
DeliveryMode.NON_PERSISTENT,5,10000);
```

## Trwałe subskrypcje



## Trwałe subskrypcje (2)




## Trwałe subskrypcje (3)

- Identyfikacja subskrypcji
  - identyfikator klienta
  - topic
  - nazwa subskrypcji

```
MessageConsumer topicSubscriber =

session.createDurableSubscriber(myTopic,
"MySub");
...
topicSubscriber.close();
...
session.unsubscribe("MySub");
```

## Lokalne transakcje

- Grupowanie operacji wysyłania/odbioru w transakcji
- Metody Session.commit() i Session.rollback()
- Zatwierdzenie oznacza wysłanie wyprodukowanych wiadomości i potwierdzenie odebranych
- Wycofanie oznacza usunięcie wyprodukowanych wiadomości i ponowne dostarczenie wiadomości odebranych (z pominięciem przedawnionych)
- Uwaga na zakleszczenia: wysłanie następuje po zatwierdzeniu