JMS
Java Message Service

Hello

Maciej Kosiedowski

Software Engineer @ Schibsted Tech Polska

JMS - agenda

- Messaging
- JMS
- Queue
- Topic
- Ćwiczenie 1
- Message Driven Bean
- Ćwiczenie 2

Messaging

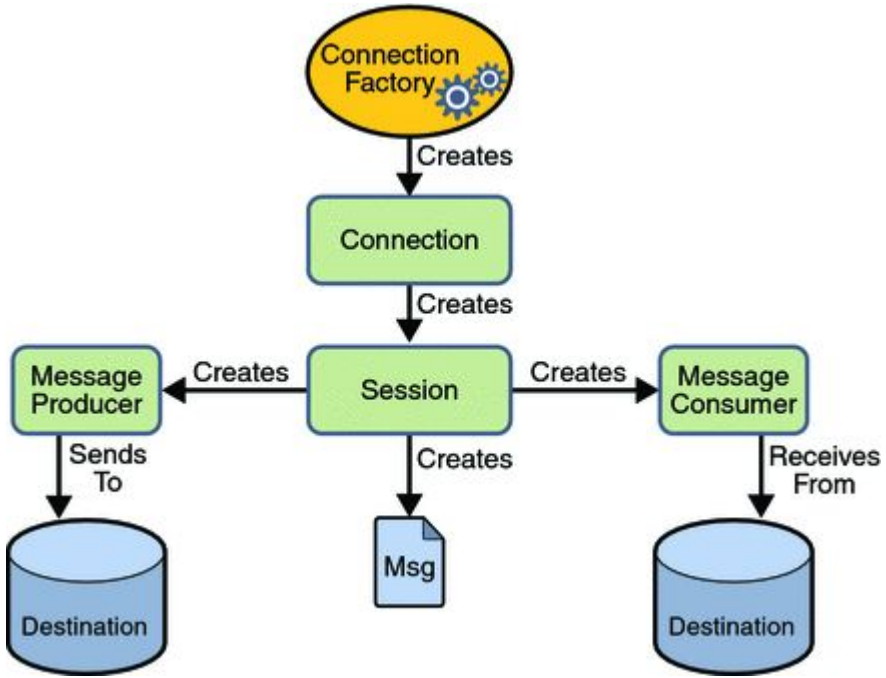
- To sposób komunikacji między systemami (lub ich częściami) oparty o wysyłanie wiadomości
- Nadawca i odbiorca nie muszą być ściśle powiązane
  - mogą nie być osiągalne w tym samym momencie
  - mogą być stworzone w różnych technologiach
  - nie muszą wiedzieć o sobie czegokolwiek
- Ważny jest adres serwera wiadomości i format komunikatu

JMS - Java Message Service

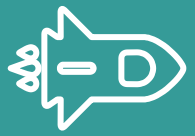
- Standardowy zestaw interfejsów i modeli przesyłania wiadomości w języku Java
- Umożliwia rozproszoną komunikację między wieloma (>=2) klientami
- Komunikacja jest asynchroniczna i niezawodna
- Istnieje wiele implementacji, m.in. ActiveMQ, RabbitMQ, Oracle Weblogic, WebSphere MQ

JMS - schemat

- Używamy ConnectionFactory do utworzenia połączenia (Connection)
- Używamy połączenia do utworzenia sesji (Session)
- Używamy sesji do utworzenia producenta (Producer), konsumenta (Consumer) i wiadomości (Message)
- Za ich pomocą wysyłamy i odbieramy wiadomości do/z miejsca docelowego (Destination)



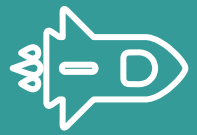
ConnectionFactory



"Connection factory" jest to obiekt służący klientowi do ustanawiania połączeń z dostawcą usługi JMS.
Tutaj odbywa się wybór konkretnej implementacji, np. ActiveMQ, i adresu

```
ConnectionFactory connectionFactory =
    new ActiveMQConnectionFactory("tcp://localhost:61616");
```

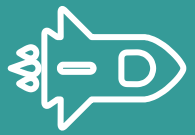
Connection



Jest to wirtualne połączenie do dostawcy JMS, służy m.in. do utworzenia jednej, lub większej liczby sesji

```
Connection connection = connectionFactory.createConnection();
connection.start();
```

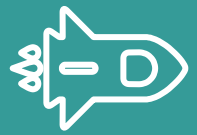
Session



Jest to jednowątkowy kontekst dla produkcji i konsumpcji wiadomości
Używany do tworzenia producentów, konsumentów i wiadomości

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

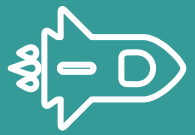
Destination



Jest to miejsce, do którego piszemy (wysyłamy) wiadomości
Jest to też miejsce, z którego czytamy (odbieramy) wiadomości

```
Destination destination = session.createQueue("TEST.MESSAGES.QUEUE");
```

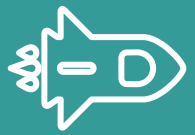
Producer



Jest to obiekt służący do wysyłania wiadomości
Tworzony jest na podstawie Session

```
MessageProducer producer = session.createProducer(destination);
producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);
```

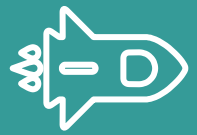
Consumer



Jest to obiekt służący do odbierania wiadomości
Tworzony jest na podstawie Session

```
MessageConsumer consumer = session.createConsumer(destination);
```

Message



Jest to obiekt reprezentujący wysyłaną lub odbieraną wiadomość

Istnieją różne rodzaje: TextMessage, ObjectMessage, BytesMessage, StreamMessage, MapMessage

```
Message message = consumer.receive();
if (message instanceof TextMessage) {
    TextMessage textMessage = (TextMessage) message;
    String text = textMessage.getText();
    System.out.println("Received: " + text);
}
```

JMS - queue

- Model point-to-point (PTP)
- Wiadomości wysyłane są do kolejek
  - wiadomości mogą mieć priorytety, są one różnie obsługiwane przez różnych dostawców JMS
- Jedna wiadomość jest obsługiwana przez jednego konsumenta
- Jeśli nie ma aktywnego konsumenta, wiadomość będzie na niego czekać
- Klient musi potwierdzić odbiór wiadomości, w przeciwnym wypadku wiadomość zostanie dostarczona ponownie (potencjalnie do innego odbiorcy)

JMS - topic

- Model publish/subscribe (pub/sub)
- Wiadomości wysyłane są do tematów (topic)
- Dla jednego tematu może istnieć wielu odbiorców
- Odbiorca musi być zarejestrowany w chwili wysłania wiadomości, by móc ją odebrać
- Podobne do rozgłośni radiowej - wiadomość może być odebrana przez dowolną liczbę klientów
- Przykład zastosowania - powiadamianie o zdarzeniach

Ćwiczenie 1

Użyjemy serwera ActiveMQ do wysłania i odebrania wiadomości

Ćwiczenie 1

1. Uruchom ActiveMQ za pomocą dockera:
2. `docker run --name='activemq' -p 8161:8161 -p 61616:61616 -p 61613:61613 webcenter/activemq`
3. Konsola administracyjna: http://localhost:8161/admin/index.jsp (admin/admin)
4. Broker URL: tcp://localhost:61616
5. git clone https://github.com/infoshareacademy/jjdd4-materialy-jms
6. Zaimportuj projekt **activemq-demo** do IDE
7. Zmodyfikuj kod w klasach Consumer/Producer tak, aby wiadomości były wysyłane z Producenta do Konsumenta przy użyciu kolejki ISA.JJDD4.MSG.QUEUE
8. Podejrzyj połączenia i wiadomości w konsoli administracyjnej http://localhost:8161/admin/index.jsp

Message Driven Bean

Enterprise bean który pozwala na asynchroniczne przetwarzanie wiadomości (produkowanych przez dowolnego producenta JMS)

```
@MessageDriven(activationConfig = {
   @ActivationConfigProperty(
      propertyName = "destinationLookup",
      propertyValue = "java:/jms/topic/ISA.TOPIC"),
   @ActivationConfigProperty(
      propertyValue = "destinationType",
      propertyName = "javax.jms.Topic")
})
public class SubscriberBean implements MessageListener {
   @Override
   public void onMessage(Message message) {
   }
}
```

Ćwiczenie 2

Użyjemy Message Driven Bean i mechanizmu topic

Ćwiczenie 2 - Message Driven Bean

1. Pobierz i uruchom WildFly (pełna dystrybucja)
   a. http://wildfly.org/downloads/
   b. ./bin/standalone.sh -c standalone-full.xml
   c. ./bin/add-user.sh (dwa razy - raz by aktywować usera admin, drugi raz by ustawić hasło)
2. http://localhost:9990 - konsola administracyjna
3. Configuration -> Subsystems -> Messaging -> Server -> defalt -> Destinations -> View -> JMS Topic -> Add
   a. Name: ISA.TOPIC
   b. Entries: java:/jms/topic/ISA.TOPIC (po wpisaniu wciśnij Enter)
4. Zaimportuj projekt subscriber-demo

Ćwiczenie 2 - Message Driven Bean

1. Uzupełnij kod tak, aby:
   a. PublisherServlet wysyłał wiadomości do tematu
   b. RepositoryWriterSubscriberBean odbierał wiadomości z tematu (@MessageDriven) i dodawał je do MessageRepository
2. Uruchom projekt: mvn clean deploy
3. Wiadomości powinny być wysyłane po wywołaniu metody GET na PublisherServlet (np. /publisher?msg=MyMessage)
4. Odebrane wiadomości powinny być wyświetlane przez MessagesServlet (/messages)

Dzieki

You can find me at
mkosied@gmail.com