
TTMS
TRANSITION
TECHNOLOGIES
MANAGED
SERVICES
GROUP
TRANSITION
TECHNOLOGIES
Java Message Service
KRZYSZTOF ŁOPUCKI • 2018
TECHNOLOGIES THAT GUIDE INDUSTRY

Java Message Service

Java Message Service – (JMS) – zestaw mechaniuzmów do asynchronicznego przesyłania komunikatów w języku programowania Java.

Java Message Service

- Darmowe
- Jest częścią JavaEE
- Komunikuje się między komponentami
- Asynchronicznie wysyła komunikaty które otrzyma
- Powstało wiele implementacji

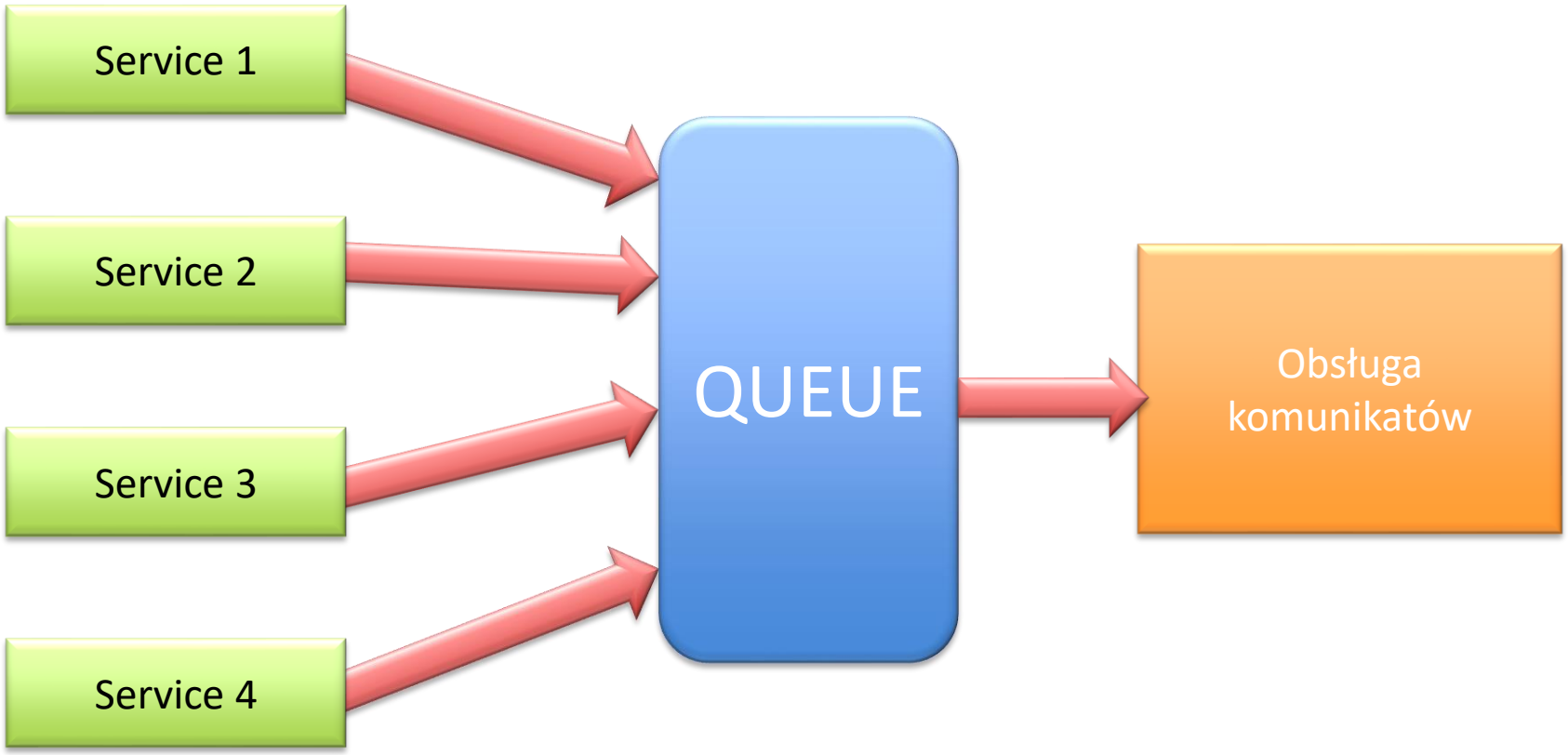
Implementacje

- **Amazon** SQS's Java Messaging Library. Apache Qpid, using AMQP.
- **IBM** MQ (formerly MQSeries, then WebSphere MQ)
- **IBM** WebSphere Application Server's Service Integration Bus (SIBus)
- JBoss Messaging and HornetQ from JBoss.
- RabbitMQ.
- Apache ActiveMQ.

Elementy

- Provider – implementacja systemu zarządzania aplikacją
- Klient – część kodu odpowiedzialna za wysyłanie i ewentualnie odbieranie informacji.
- Producent – klient który tworzy i wysyła wiadomości
- Konsument – klient który odbiera wiadomości i przetwarza je.
- Wiadomość – obiekt zawierajacy dane przenoszone pomiędzy klientami.
- Kolejka – miejsce w którym odkładane są kolejne rządania do obsłużenia.

Java Message Service



Spring boot - ActiveMQ

- Automatyczna konfiguracja
- Duże możliwości konfiguracyjne
- Wbudowany mechanizm

Spring boot – Zależności projektowe

- spring-boot-starter-activemq
- activemq-broker

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>
          spring-boot-starter-activemq
    </artifactId>
</dependency>
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-broker</artifactId>
</dependency>
```

Konfiguracja

```
@SpringBootApplication
@EnableJms
public class MainBrain {
    @Bean
    public JmsListenerContainerFactory<?> myFactory(ConnectionFactory connectionFactory,
                                DefaultJmsListenerContainerFactoryConfigurer configurer){
        DefaultJmsListenerContainerFactory factory =
                                        new DefaultJmsListenerContainerFactory();
          configurer.configure(factory, connectionFactory);
        return factory;
    }
    @Bean
    public MessageConverter jacksonJmsMessageConverter() {
        MappingJackson2MessageConverter converter = new MappingJackson2MessageConverter();
        converter.setTargetType(MessageType.TEXT);
        converter.setTypeIdPropertyName("_type");
        return converter;
    }
}
```

Wiadomość

```
public class EventObject {

    private Long objectId;

    private String message;

    private EventType eventType;

       // GETTER & SETTER

}
```

Listener

```
@Component
public class EventReceiver {
    private final EventRepository eventRepository;

    @Autowired
    public EventReceiver(EventRepository eventRepository) {
        this.eventRepository = eventRepository;
    }

    @JmsListener(destination = "events",
                 containerFactory = "myFactory")
    public void receiveMessage(EventObject eventObject) {
        Event event = prepareMessage(eventObject);
        eventRepository.save(event);
    }
}
```

Wysyłanie wiadomości

```
private final JmsTemplate jmsTemplate;

@Autowired
public UserService(UserRepository userRepository,
                   JmsTemplate jmsTemplate) {
    this.userRepository = userRepository;
    this.jmsTemplate = jmsTemplate;
}
```

```
public User save(User user) {
    User savedUser = userRepository.save(user);
    runEvent(CREATE,
             savedUser.getId(),
             "Successfull User Saved„
        );
    return savedUser;
}
```

Zadanie

Zaprogramuj obsługę kolejki w sklepie. Każda sprzedaż powinna być odnotowana w bazie. Pamiętaj, że nie można sprzedawać alkoholu osobom nieletnim, a niektórych produktów może być ograniczona ilość. Zakupów można dokonywać przez aplikację Victor – to tam będzie producent wiadomości.

TRANSITION
TECHNOLOGIES
MANAGED
SERVICES

GROUP
TRANSITION
TECHNOLOGIES

DZIĘKUJĘ

Krzysztof Łopucki

Krzysztof.Lopucki@ttms.pl