Praca domowa 12 – mdb

Szlachetka Miłosz
Nr albumu. 114014

1.Cel zadania

Celem zadania była implementacja komponentu MDB (Message Driven Bean) nasłuchującego komunikatów i realizującego funkcję dwustanowego licznika.

2.Struktura programu

public class MdbBean implements MessageListener - komponent typu MDB (Message Driven Bean). Komponent ten realizuje funkcję dwustanowego licznika(Stan zliczania oraz stop). Przechowuje wartość licznika oraz licznika błędów. Komponent odczytuje dane z zasobu będącego kolejką (Queue) o nazwie “jms/MyQueue”. Wiadomości zwronte wysyłane są do zasobu typu Topic o nazwie jms/MyTopic. Komponent zawiera definicję typu enum reprezentującą dozwolone dla komponentu(licznika) stany. Definicja ta wygląda następująco:

```
enum State {
    COUNTING,
    STOPPED
}
```

Ponadto komponent **MdbBean** zawiera następujące metody:

MdbBean() - konstruktor. Znajduje się w nim inicjalizacja licznika, licznika błędów oraz ustalenie stanu początkowego komponentu.

void start() - ustawia komponent w stan zliczania jeśli poprzednim stanem był stan stop. W przeciwnym wypadku zwiększa licznik błędów o jeden.

void increment(double i) - inkrementuje licznik o wartość “i” jeśli komponent znajduje się w stanie zliczania. W przeciwnym wypadku zwiększa wartość licznika błędów o jeden.

double getDoubleFromString(String str) - usuwa z napisu (będącego argumentem metody) znaki niebędące liczbą, kropką (.) ani myślnikiem (-) a następnie konwertuje wynik do zmiennej double i zwraca ją.

void sendMsg(String message) - nawiązuje połączenie z zasobem o nazwie "jms/MyConnectionFactory" oraz z zasobem typu Topic o nazwie "jms/MyTopic" oraz wysyła do niego wiadomość o tresci, która składa się z napisu (String) będącej argumentem metody poprzedzonym numerem albumu oraz slashem (/).

void onMessage(Message message) - metoda obsługująca wiadomości trafiające do kolejki (Queue). Sprawdza czy otrzymana w argumencie metody wiadomość jest wiadomością tekstową (TextMessage). Jeśli tak to sprawdza jaka jest zawartość wiadomości i w zależności od tego wykonuje określone akcje. Rozpoznawane wiadomości to: start, stop, counter, increment, increment/n gdzie jest liczbą. Jeśli treść wiadomości nie jest zgodna z żadnym z wcześniej wymienionych wzorców inkrementowany jest licznik błędów.

3.Sposób zestawiania połączenia

Połączenie zestawiane jest poprzez użycie kontekstu (InitialContext). Obiekt kontekstu jest następnie wykorzystywany do odnalezienia fabryki typu TopicConnectionFactory o nazwie "jms/MyConnectionFactory". Następnie przy użyciu obiektu typu TopicConnectionFactory do utworzenia sesji typu TopicSession. Następnie przy użyciu mechanizmu lookup nawiązywane jest połączenie z zasobem o nazwie "jms/MyTopic". Aby wysłać wiadomość do zasobu typu Topic wykorzystano metodę send obiektu typi TopicPublisher. Fragment kodu realizujący wyżej opisaną funkcjonalność prezentuje się następująco:

```
Context ctx = new InitialContext();
TopicConnectionFactory factory
      = (TopicConnectionFactory) ctx.lookup(CON_FACTORY);
try (TopicConnection topicConnection = factory.createTopicConnection();
      TopicSession topicSession = topicConnection.createTopicSession(
            false, Session.AUTO_ACKNOWLEDGE)) {
   Topic topic = (Topic) ctx.lookup("jms/MyTopic");
   TopicPublisher publisher = topicSession.createPublisher(topic);
   topicConnection.start();
   message = ALBUM + "/" + message;
   TextMessage textMessage = topicSession.createTextMessage();
   textMessage.setText(message);
   publisher.send(textMessage);
```

Poza tym komponent MDB został opatrzony adnotacją @MessageDriven, która zawiera między innymi nazwę i typ zasobu z którego pobiera komunikaty. Adnotacja ta została użyta w następujący sposób:

```
@MessageDriven(activationConfig = {
   @ActivationConfigProperty(
         propertyName = "destinationLookup",
         propertyValue = "jms/MyQueue") ,
   @ActivationConfigProperty(
         propertyName = "destinationType",
         propertyValue = "javax.jms.Queue"),
   @ActivationConfigProperty(propertyName = "acknowledgeMode",
         propertyValue = "Auto-acknowledge")})
```

4. Algorytm

Zaimplementowany komponent MDB przechowuje licznik oraz licznik błędów. Początkowo liczniki są wyzerowane. Komponent odczytuje z kolejki (Queue) wiadomości. Sprawdza czy wiadomosc jest wiadomością tekstową. Jeśli tak to sprawdza jaka jest treść wiadomości. Dla określonych treści wykonywane są odpowiednio akcje. Dla wiadomości o treści “start” komponent ustawiany jest w stan zliczania, jeśli poprzednim stanem komponentu był stan stop. W przeciwnym wypadku zwiększany jest licznik błędów. Dla wiadomości “stop” komponent MDB ustawiany jest w stan stop, jeśli komponent był w stanie zliczania. W przeciwnym wypadku zwiększany jest licznik błędów. Jeśli treść wiadomości to “counter” do zasobu Topic wysyłana jest wartość licznika błędów poprzedzona numerem albumu i znakiem slash (/). Dla wiadomości “error” wysyłana jest wiadomość o treści będącej wartością licznika błędów (tak samo jak poprzednio wartość ta jest poprzedzona numerem albumu oraz slashem). Dla wiadomości “increment” następuje zwiększenie wartości licznika o jeden. Jeśli wiadomość nie pasuje do żadnej z wyżej wymienionych to treść wiadomości jest sprawdzana ze wzorcem:

"^increment/-?[0-9]+(\\.[0-9]+)?$"

Wzorzec ten dopuszcza ciągi znaków które zaczynają od wyrażenia **inrement/** . Drugim członem wyrażenia może być dodatnia lub ujemna liczba zmiennoprzecinkowa czy też liczba całkowita. Jeśli wyrażenie pasuje do powyższego wzorca to z wiadomości wyłuskiwana jest wartość liczbowa, która następnie jest dodawana do wartości licznika. Jeśli treść wiadomości z kolejki nie pasuje do żadnego, wcześniej wymienionego, przypadku to następuje inkrementacja licznika błędów.