Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение наукии высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

**Институт информационных технологий и радиотехники**

**(ИИТР)**

Кафедра информационных систем и программной инженерии

**Лабораторная работа № 05**

**по дисциплине**

**«Распределенные программные системы»**

Выполнил:

ст. гр. ПРИ-117

Хлызова В.Г.

Принял:

Трифонов Д.А.

Владимир, 2020

ЦЕЛЬ РАБОТЫ

Получить навыки работы с системами обмена сообщениями и реализации приложений на платформе Java, взаимодействующих путем обмена сообщениями.

ВЫПОЛНЕНИЕ РАБОТЫ

Лабораторная работа была выполнена с использованием сервера Glassfish 4.1.2 и ActiveMQ 5.5.0.

Открываем консоль администрирования Glassfish и установлеваем адаптер ресурсов activemq-rar-5.5.0.rar.

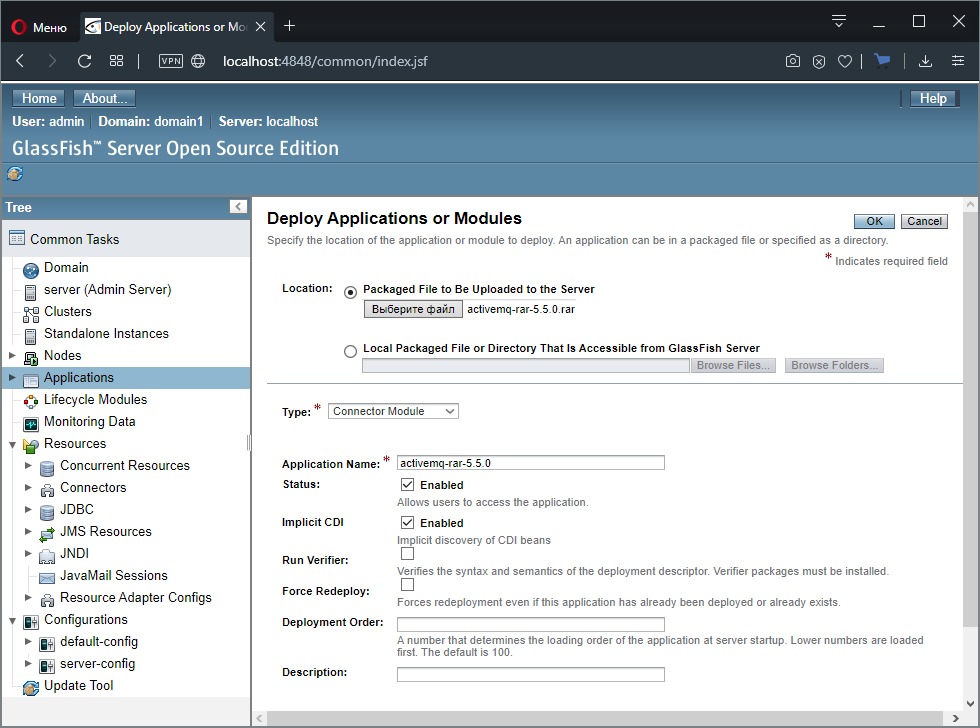


Рисунок 1. Установка адаптера ресурсов.

Создаем конфигурацию адаптера ресурсов (Resource Adapter Config), выбираем пул потоков и устанавливаем следующие значения свойств: ServerUrl: vm://localhost:61616 и BrokerXmlConfig: broker:(tcp://localhost:61616)

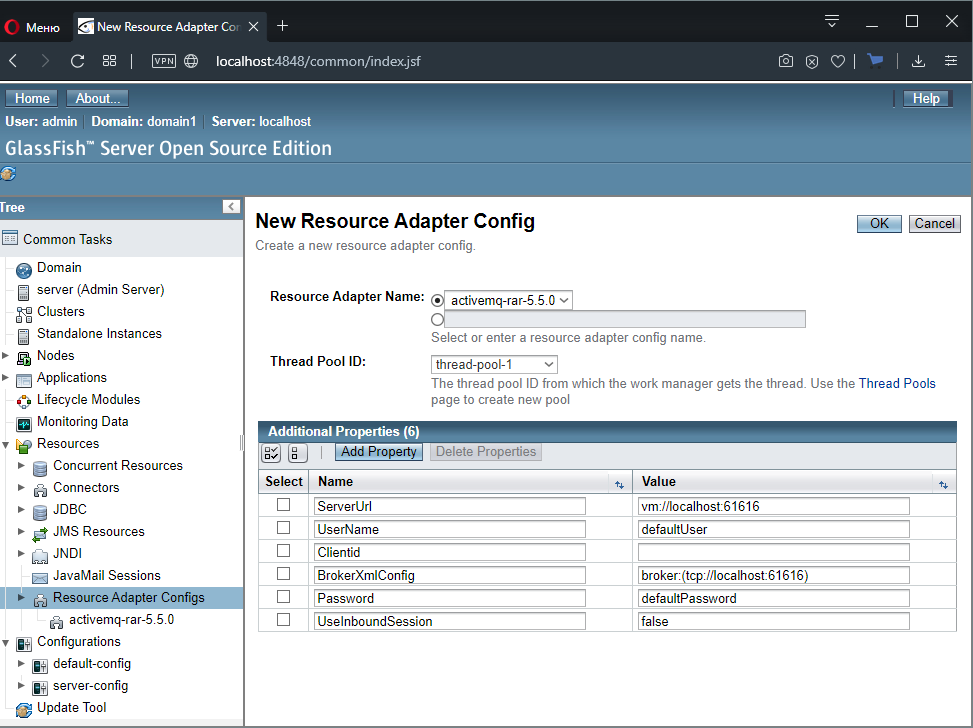


Рисунок 2. Создаем конфигурацию адаптера ресурсов.

Добавляем пул соединений с коннектором (Connector Connection Pool), выбираем адаптер ресурсов для ActiveMQ, на второй странице мастера оставляем значения свойств по умолчанию.

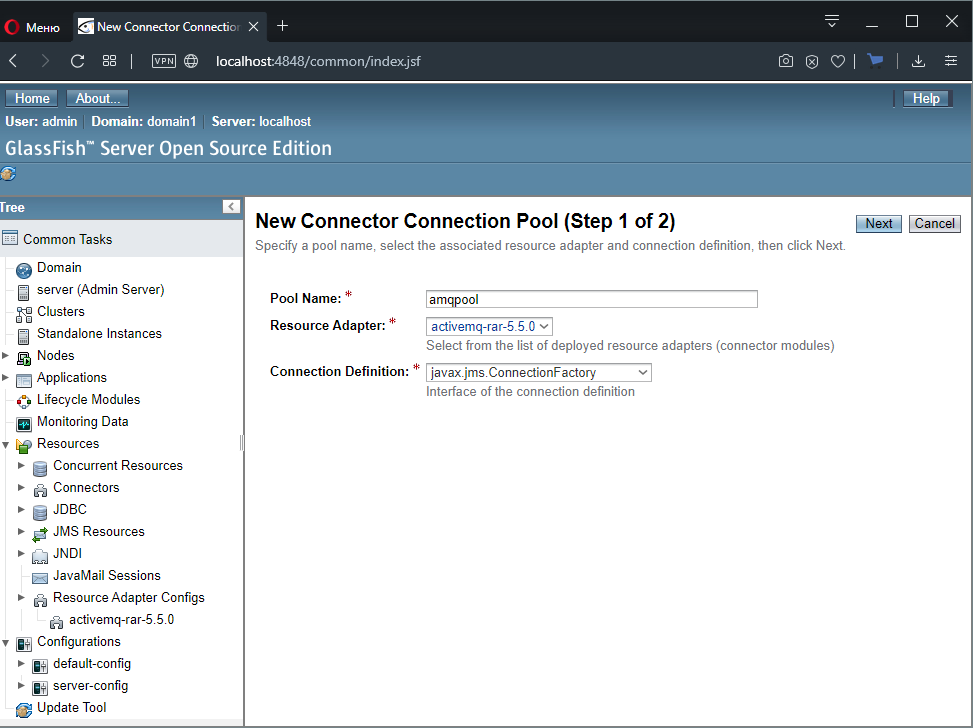


Рисунок 3. Создаем пул соединений.

Добавляем ресурс коннектора (Connector Resource), указываем в нем созданный на предыдущем шаге пул соединений с коннектором.

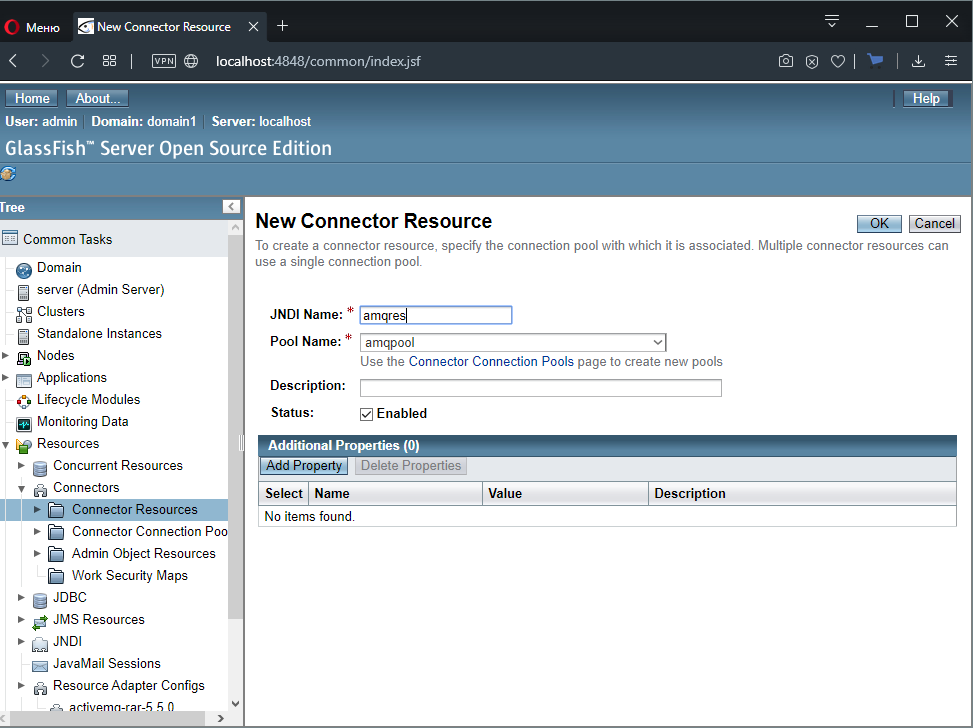


Рисунок 4. Добавляем ресурс коннектора.

Добавляем ресурс администрируемого объекта (Admin Object Resource), выбираем для него адаптер ресурса ActiveMQ и тип ресурса «очередь» (Queue).

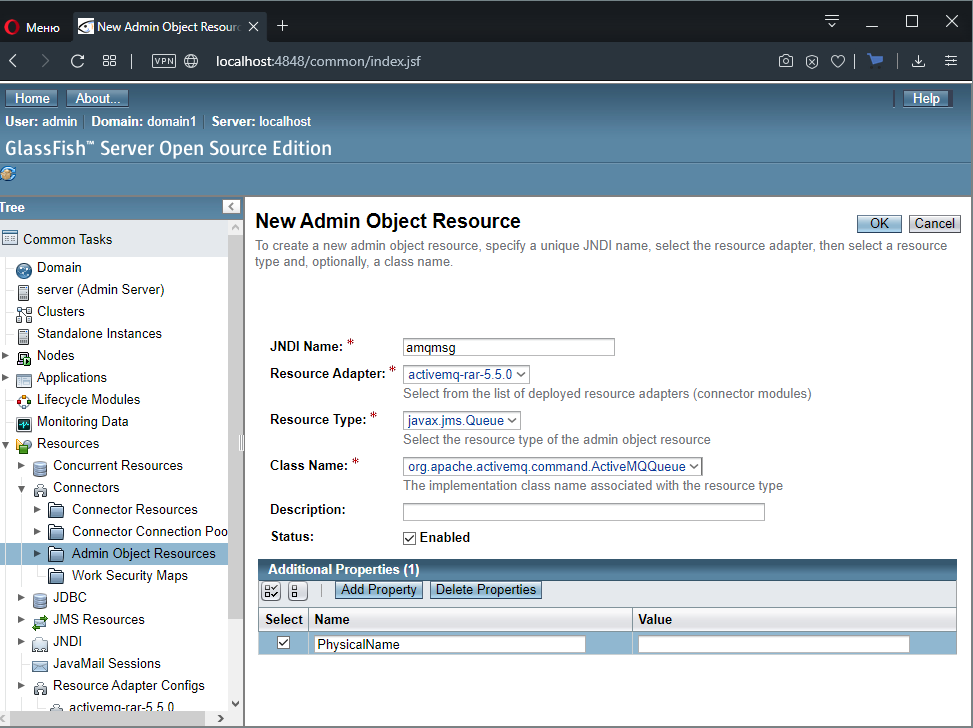


Рисунок 5. Добавляем ресурс администрируемого объекта.

Добавляем следующие настройки JVM для запуска сервера приложений (и для default-config, и для server-config) и перезапускаем Glassfish:

-Dwebconsole.type=properties

-Dwebconsole.jms.url=tcp://localhost:61616

-Dwebconsole.jmx.url=service:jmx:rmi:///jndi/rmi://localhost:1099/jmxrmi

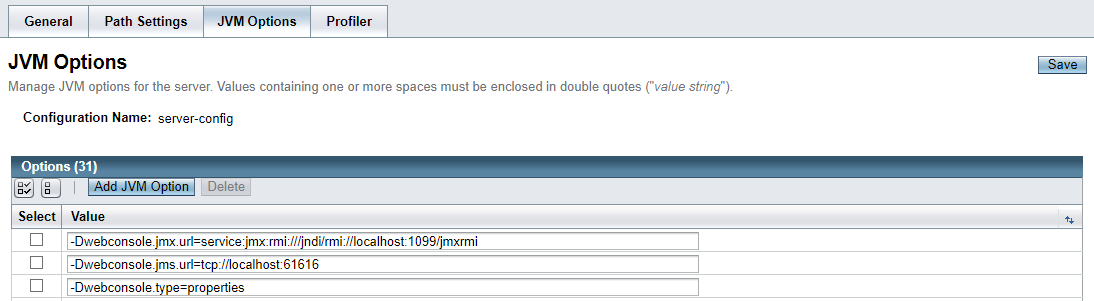


Рисунок 6. Добавление настроек JVM.

Устанавливаем веб-приложение консоли администрирования ActiveMQ.

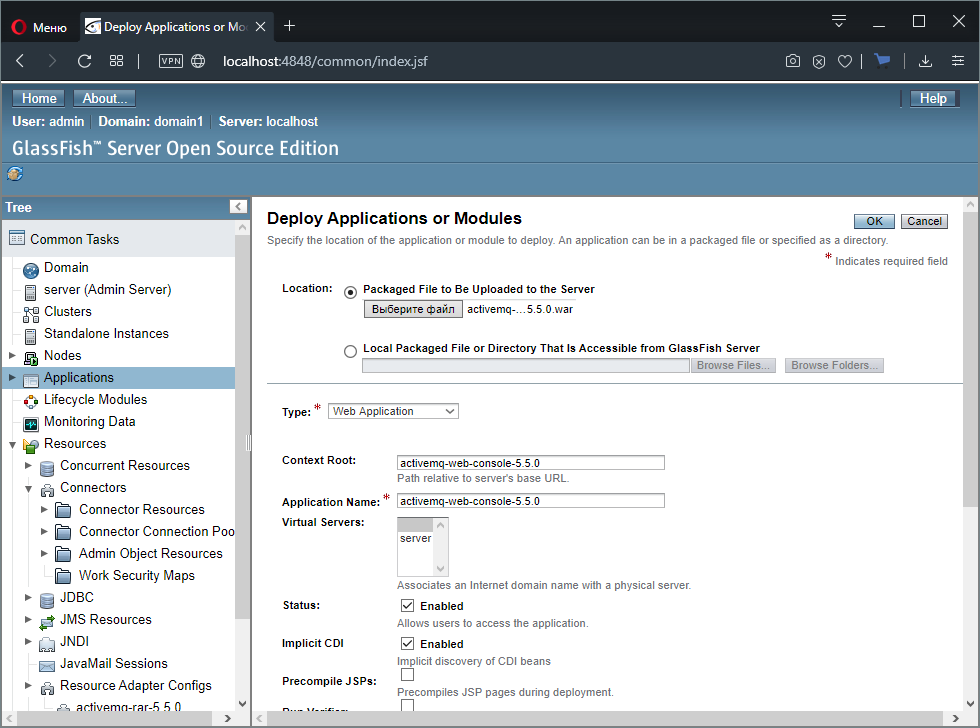


Рисунок 7. Установка консоли администрирования.

Запускаем консоль администрирования и создаём очередь с таким же названием, что и созданный ранее ресурс администрируемого объекта(amqmsg).

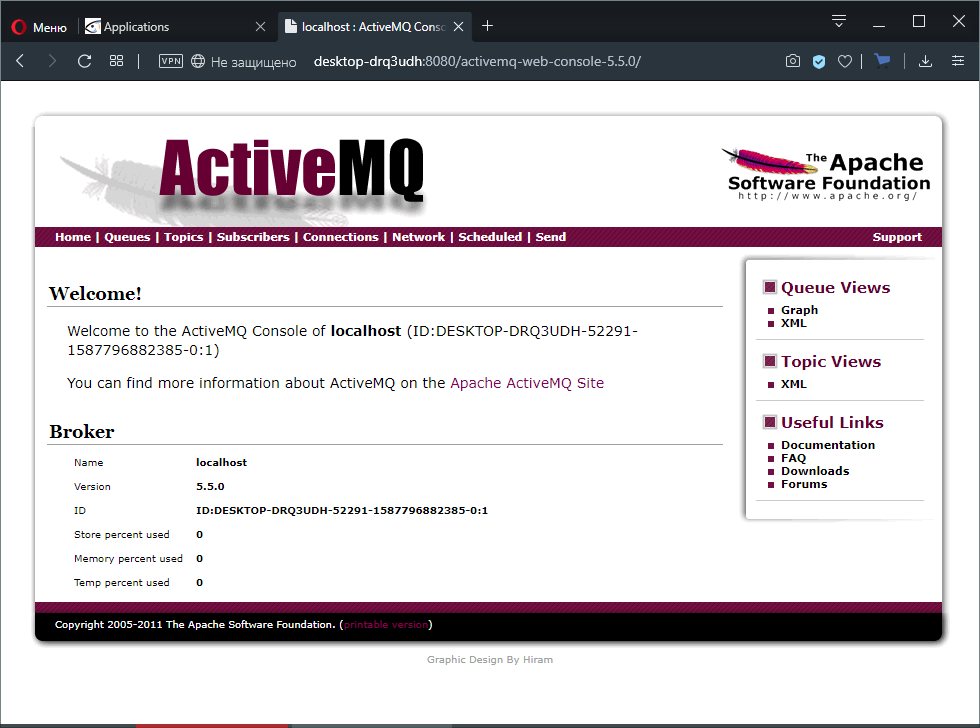


Рисунок 8. Запуск консоли администрирования.

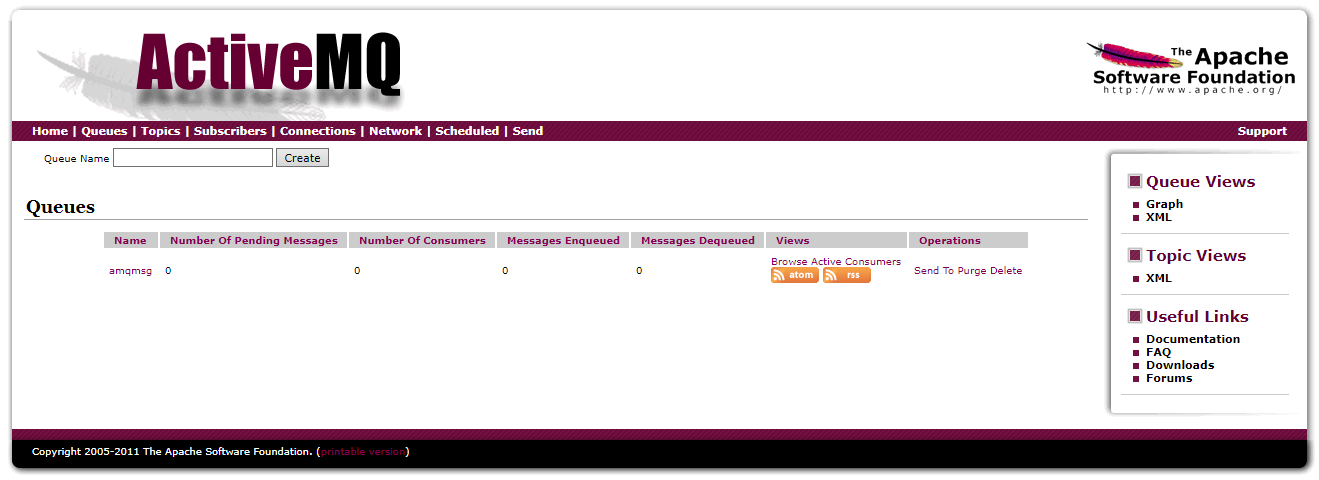


Рисунок 9. Создание очереди.

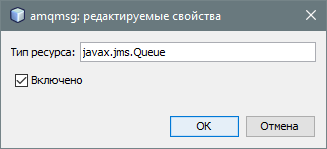


Рисунок 10 Проверяем состояние очереди в NetBeans/ Очередь включена.

Далее создаем проект клиентского приложения, которое позволит убедиться в правильной настройке очереди. Код главного класса приложения:

package simplsendamq;

import javax.jms.Connection;

import javax.jms.DeliveryMode;

import javax.jms.Destination;

import javax.jms.MessageProducer;

import javax.jms.Session;

import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnection;

import org.apache.activemq.ActiveMQConnectionFactory;

public class SimpleSendAMQ {

private String user = ActiveMQConnection.DEFAULT\_USER;

private String password = ActiveMQConnection.DEFAULT\_PASSWORD;

private String url = ActiveMQConnection.DEFAULT\_BROKER\_URL;

private Destination destination;

public static void main(String[] args) {

SimpleSendAMQ mySend = new SimpleSendAMQ();

mySend.sendMessage();

}

void sendMessage() {

try {

Connection connection = null;

ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory(user, password, url);

connection = connectionFactory.createConnection();

connection.start();

Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

destination = session.createQueue("amqmsg");

MessageProducer producer = session.createProducer(destination);

producer.setDeliveryMode(DeliveryMode.NON\_PERSISTENT);

TextMessage message = session.createTextMessage("THIS IS THE TEST MESSAGE !");

producer.send(message);

connection.close();

} catch (Exception ex) {

System.out.println(ex.getMessage());

}

}

}



Рисунок 11. После запуска приложения в очереди появилось одно сообщение.

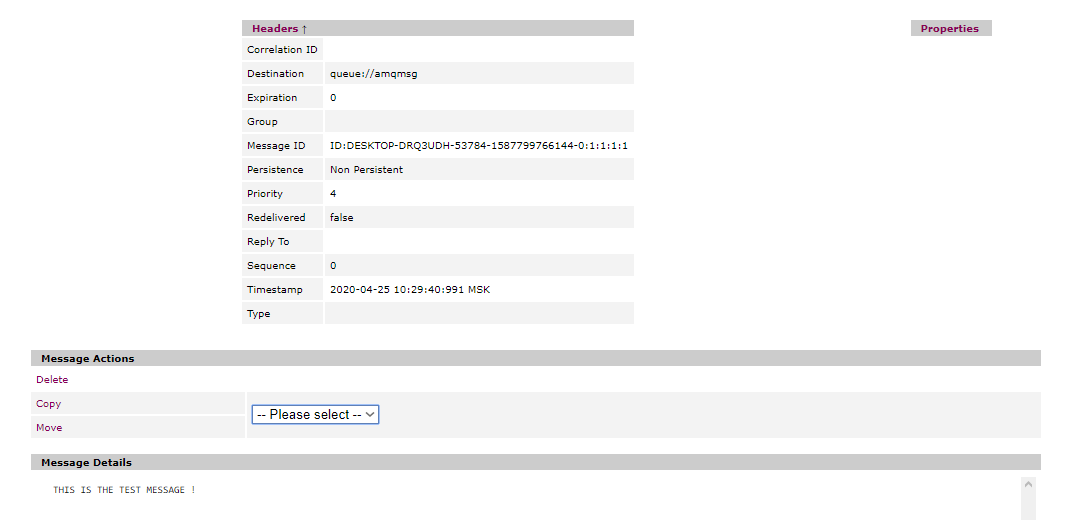


Рисунок 12. Содержание сообщения.

Далее создаем проект EJB-модуля с компонентом, управляемым сообщениями. При создании компонента выбираем в списке адресатов сервера созданную ранее очередь. Сгенерированный код представлен ниже.

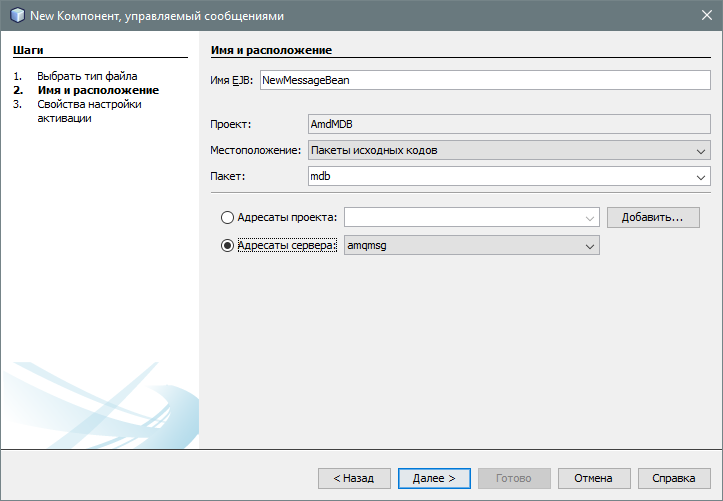


Рисунок 13. Создание компонента.

package mdb;

import javax.ejb.ActivationConfigProperty;

import javax.ejb.MessageDriven;

import javax.jms.Message;

import javax.jms.MessageListener;

/\*\*

 \*

 \* @author Лера

 \*/

@MessageDriven(activationConfig = {

    @ActivationConfigProperty(propertyName = "destinationLookup", propertyValue = "amqmsg")

    ,

        @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue")

})

public class NewMessageBean implements MessageListener {

    public NewMessageBean() {

    }

    @Override

    public void onMessage(Message message) {

    }

}

Для взаимодействия с ActiveMQ требуется дополнительная конфигурация компонента, управляемого сообщениями. Создаем дескриптор для Glassfish:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE glassfish-ejb-jar PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 EJB 3.1//EN" "http://glassfish.org/dtds/glassfish-ejb-jar\_3\_1-1.dtd">

<glassfish-ejb-jar>

 <enterprise-beans>

  <ejb>

   <ejb-name>NewMessageBean</ejb-name>

    <mdb-resource-adapter>

     <resource-adapter-mid>activemq-rar-5.5.0</resource-adapter-mid>

    </mdb-resource-adapter>

      <mdb-connection-factory>

          <jndi-name>amqpool</jndi-name>

    </mdb-connection-factory>

   </ejb>

 </enterprise-beans>

</glassfish-ejb-jar>

Далее создаем стандартный дескриптор (файл типа «Enterprise JavaBeans /Стандартный дескриптор развертывания»):

<?xml version="1.0" encoding="UTF-8"?>

<ejb-jar xmlns="http://xmlns.jcp.org/xml/ns/javaee"

        version="3.2"

        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/ejb-jar\_3\_2.xsd">

<display-name>Ejb1</display-name>

    <enterprise-beans>

        <message-driven>

            <display-name>NewMessageBean</display-name>

            <ejb-name>NewMessageBean</ejb-name>

            <ejb-class>mdb.NewMessageBean</ejb-class>

            <messaging-type>javax.jms.MessageListener</messaging-type>

            <transaction-type>Container</transaction-type>

            <activation-config>

                <activation-config-property>

                    <activation-config-property-name>DestinationType</activation-config-property-name>

                    <activation-config-property-value>javax.jms.Queue</activation-config-property-value>

                </activation-config-property>

                <activation-config-property>

                    <activation-config-property-name>Destination</activation-config-property-name>

                    <activation-config-property-value>amqmsg</activation-config-property-value>

                </activation-config-property>

            </activation-config>

        </message-driven>

    </enterprise-beans>

    <assembly-descriptor>

        <container-transaction>

            <method>

                <ejb-name>NewMessageBean</ejb-name>

                <method-name>onMessage</method-name>

                <method-params>

                    <method-param>java.lang.String</method-param>

                </method-params>

            </method>

            <trans-attribute>Required</trans-attribute>

        </container-transaction>

    </assembly-descriptor>

</ejb-jar>

Устанавливаем проект EJB-модуля на сервере приложений. Заходим в консоль администрирования ActiveMQ, и проверяем обработку сообщения. (в столбце Number of Pending Messages стоит 0).

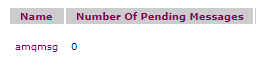


Рисунок 14. Успешная обработка сообщения компонентом.

Сообщения можно отправлять используя консоль администрирования ActiveMQ:

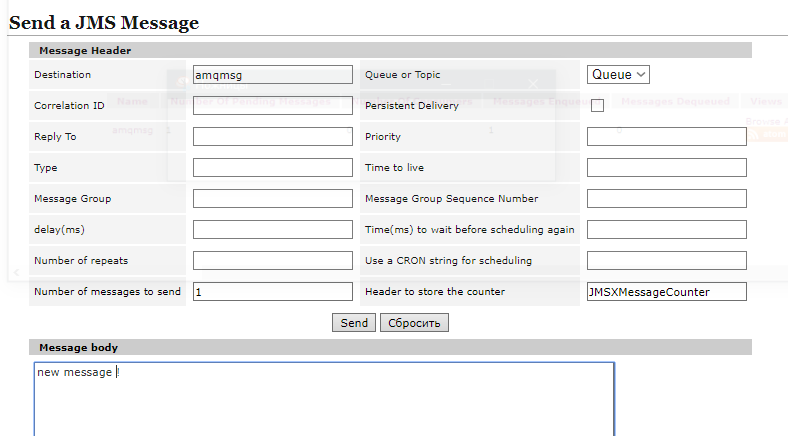


Рисунок 15. Отправка сообщения.

Создание веб-приложения для отправки сообщений и получения сообщений из очереди. Класс управляемого бина с методами отправки и получения сообщений из очереди.

package bean;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.inject.Named;

import javax.enterprise.context.ApplicationScoped;

import javax.jms.Connection;

import javax.jms.DeliveryMode;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.Message;

import javax.jms.MessageConsumer;

import javax.jms.MessageProducer;

import javax.jms.Session;

import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnection;

import org.apache.activemq.ActiveMQConnectionFactory;

/\*\*

 \*

 \* @author Лера

 \*/

@Named(value = "newManagedBeanJMS")

@ApplicationScoped

public class NewManagedBeanJMS {

    private final String user = ActiveMQConnection.DEFAULT\_USER;

    private final String password = ActiveMQConnection.DEFAULT\_PASSWORD;

    private final String url = ActiveMQConnection.DEFAULT\_BROKER\_URL;

    private Destination destination;

    public void sendMessage(String textMsg) {

        try {

            Connection connection = null;

            ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory(user, password, url);

            connection = connectionFactory.createConnection();

            connection.start();

            Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

            destination = session.createQueue("amqmsg");

            MessageProducer producer = session.createProducer(destination);

            producer.setDeliveryMode(DeliveryMode.NON\_PERSISTENT);

            TextMessage message = session.createTextMessage(textMsg);

            producer.send(message);

            connection.close();

        } catch (JMSException ex) {

           Logger.getLogger(this.getClass().getName()).log(Level.WARNING, "Cannot close session", ex);

        }

    }

    public String receiveMessage(){

        Connection connection = null;

        String msgStr = "";

        try {

            ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory(user, password, url);

            connection = connectionFactory.createConnection();

            Session session = connection.createSession(false, Session.AUTO\_ACKNOWLEDGE);

            destination = session.createQueue("amqmsg");

            MessageConsumer consumer = session.createConsumer(destination);

            connection.start();

            Message msg;

            msg = consumer.receive(60);

            if (msg==null) return "В очереди нет сообщений";

            TextMessage textMsg = (TextMessage)msg;

            msgStr = msg.getJMSMessageID() + " ТЕКСТ: " + textMsg.getText();

            connection.close();

        }catch (JMSException ex) {

            Logger.getLogger(this.getClass().getName()).log(Level.WARNING, "Cannot close session", ex);

        }

        return msgStr;

    }

    public NewManagedBeanJMS() {

    }

}

Контроллер:

package controller;

import bean.NewManagedBeanJMS;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

/\*\*

 \*

 \* @author Лера

 \*/

public class JMSServlet extends HttpServlet {

    @Override

    protected void doGet(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        NewManagedBeanJMS jms = new NewManagedBeanJMS();

        String receiveMessage = jms.receiveMessage();

        request.setAttribute("msg", receiveMessage);

        RequestDispatcher requestDispatcher = request.getRequestDispatcher("index.jsp");

        requestDispatcher.forward(request, response);

    }

    @Override

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

            throws ServletException, IOException {

        String text = request.getParameter("text");

        NewManagedBeanJMS jms = new NewManagedBeanJMS();

        jms.sendMessage(text);

        RequestDispatcher requestDispatcher = request.getRequestDispatcher("index.jsp");

        requestDispatcher.forward(request, response);

    }

}

Jsp-cтраница:

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE html>

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

        <title>JSP Page</title>

    </head>

    <body>

       <form action="JMSServlet" method="POST">

            Текст сообщения:

            <textarea name="text"></textarea>

            <button>Отправить</button>

        </form>

        <form action="JMSServlet" method="GET">

            <button>Получить сообщение из очереди</button>

        </form><br>

        <c:out value="${msg}"/>

    </body>

</html>

Запускаем приложение и отправляем три сообщения в очередь, заходим в консоль администрирования и проверяем отправку сообщений (Рисунок 16) Все три сообщения были успешно отправлены. Нажимаем на кнопку «Получить сообщение из очереди», получаемое сообщение содержит ID сообщения и текст. (Рисунки 17-19)

Ссылка на гит с приложениями, созданными в данной лабораторной: <https://github.com/arranay/LAB-RPS/tree/master/lab11%20Обмен%20сообщениями%20и%20Java%20Message%20Service>

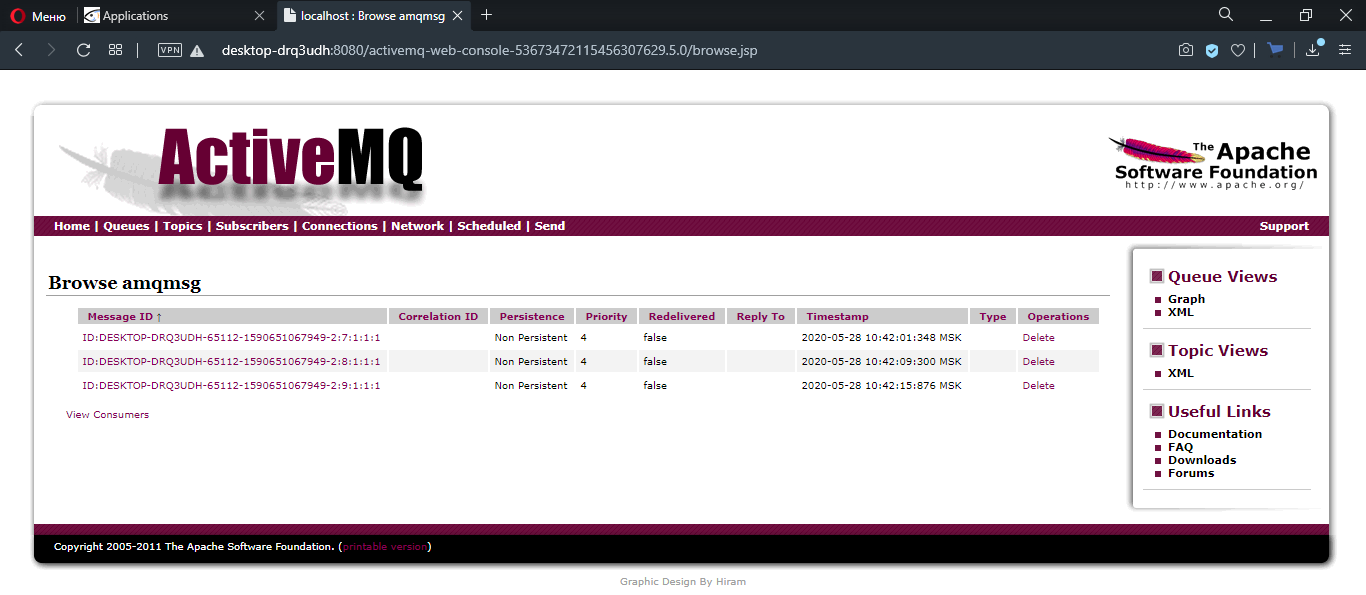


Рисунок 16. Отправленные сообщения.

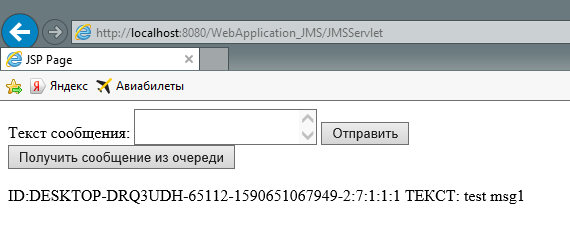


Рисунок 17. Первое сообщение из очереди.

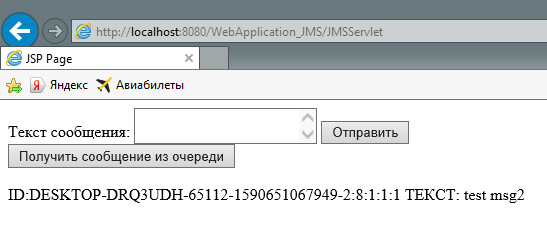


Рисунок 18. Второе сообщение из очереди.

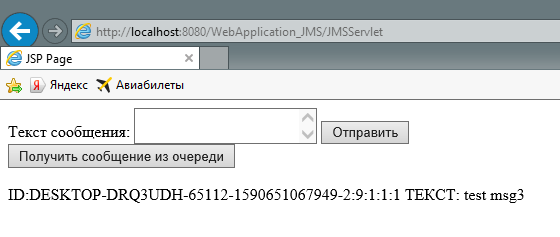


Рисунок 19. Последнее сообщение.

ВЫВОД

В ходе выполнения лабораторной работы были получены навыки работы с системами обмена сообщениями и реализации приложений на платформе Java, взаимодействующих путем обмена сообщениями.