Лабораторная работа №2

Разработка Java-приложения для обмена сообщениями с использованием JMS 1.1

Цель работы

Разработать простое Java-приложение, демонстрирующее процесс обмена сообщениями с использованием технологии JMS 1.1

Общие сведения

JMS – стандарт межпроцессного взаимодействия посредством рассылки сообщений, определенный в спецификации Java EE. История версий насчитывает всего три релиза этого стандарта, последний из которых – 2.0 был добавлен в JEE7.

В версии JMS 1.0 были представлены различные интерфейсы для каждой модели передачи сообщений («Точка-Точка» и «Подписчик-Издатель»).

С выходом версии JMS 1.1 появился общий, унифицированный интерфейс для работы с одновременно двумя доменами.

Рассмотрим основные интерфейсы JMS 1.1 (таблица 1):

Таблица 1 – Интерфейсы JMS

Интерфейс	Описание интерфейса
ConnectionFactory	Объект, создающий Connection. В параметре инициализации нужно передавать данные вашего JMS сервера.
Connection	Соединение с сервером JMS. Создает объект Session.

Завершение таблицы 1

Интерфейс	Описание интерфейса
Session	Контекст, реализующий передачу и получение сообщений. Управляет транзакцией в JMS и его использование разными потоками невозможно или ограничено. Рекомендуется создавать разные объекты Session для каждого потока. Создает объект Destination.
Destination	Объект, хранящий адреса сообщений (имя топика или очереди). С его помощью создаются объекты, отвечающие за отсылку сообщений на конкретно указанный адресат и их получения оттуда.
MessageProducer	Объект для отправки сообщений.
MessageConsumer	Объект для получения сообщений.

Алгоритм создания приложения, работающего с JMS, можно определить, проанализировав таблицу интерфейсов:

1. Подключение к серверу, используя ConnectionFactory.
2. Получение соединения Connection из ConnectionFactory.
3. Создание однопоточного контекста Session из соединения.
4. Получение буфера Destination, привязанного к определенному адресу для создания интерфейсов отправки и получения сообщений.
5. Создание объектов MessageProducer для отправки или MessageConsumer для получения сообщений.

В JMS различают сообщения нескольких типов (таблица 2).

Таблица 2 – Типы JMS сообщений

Тип сообщения	Описание типа сообщения
StreamMessage	Как видно из названия, это поток примитивных типов Java. Считывать можно из стандартных интерфейсов ввода/вывода.
MapMessage	Содержит информацию на подобие коллекций в виде ключ-значение (String, Object).
TextMessage	Обычное, текстовое сообщение.
ObjectMessage	Данный тип предназначен для передачи Serializable-объектов.
BytesMessage	Список не интерпретированных байт. С его помощью можно передавать файлы.

Любое JMS сообщение включает в себя 3 составные части:

1. Заголовок (header). Набор свойств, поставляемый по умолчанию для любого сообщения. В их числе:

– информация об отправителе;

– информация о получателе;

– информация о самом сообщении и др.

Подробнее свойства заголовков рассматриваются ниже.

2. Свойства (properties). Дополнительный набор свойств, определяемый разработчиком.

3. Тело (body). Непосредственно содержимое сообщения.

Все параметры заголовка сообщения (таблица 3) имеют префикс JMS.

Методы установки и считывания их свойств будут выглядеть соответственно setJMS…() и getJMS…().

Таблица 3 – Параметры заголовка сообщения

Наименование	Тип	Описание
JMSMessageID	String	Параметр идентификации сообщения.
JMSDestination	String	Адрес передачи сообщения.
JMSDeliveryMode	int	Может иметь только два значения: DeliveryMode.PERSISTENT и DeliveryMode.NON_PERSISTENT. Персистентное сообщение доставляется один и только один раз; не персистентное сообщение доставляется не более одного раза. «Не более одного раза» подразумевает возможность отсутствия доставки. Для гарантирования отсутствия влияния системных сбоев на доставку персистентных сообщений должны выполняться дополнительные действия. Для передачи персистентных сообщений часто необходимы значительные дополнительные накладные расходы, и нужно тщательно рассматривать баланс между надежностью и производительностью при выборе режима доставки сообщения.
JMSTimestamp	long	Время доставки сообщения JMS серверу для последующей передачи.

Завершение таблицы 3

Наименование	Тип	Описание
JMSExpiration	long	Время жизни сообщения. 0 – означает, что сообщение будет жить, пока оно не будет доставлено.
JMSPriority	int	Приоритет сообщения от 0 (самый ничтожный) до 9 (наивысший приоритет).
JMSCorrelationID	String	Необходимо для связи ответного сообщения с определенным сообщением-запросом. В это поле помещается JMSMessageID сообщения, на которое необходим ответ.
JMSReplyTo	Destination	Указание места, куда должно быть передано ответное сообщение. Например, все вопросы вы помещаете в топик question, а ответы на них хотите видеть в топике response.
JMSType	String	Тип сообщения.
JMSRedelivered	Boolean	Означает, что сообщение было доставлено получателю, но он не подтвердил прием сообщения.

Процесс выполнения работы

В данном лабораторном практикуме, для разработки Java-приложения, реализующего обмен сообщениями с использованием технологии JMS 1.1, будет рассматриваться использование среды разработки Eclipse IDE.

Среду разработки Eclipse IDE можно загрузить с официального сайта по следующей ссылке http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/marsr.

Для создания приложения необходимо выполнить следующие действия:

1. Запустить среду разработки Eclipse IDE, в панели управления выбрать File -> New -> Java Project, в поле Project name ввести название создаваемого проекта и нажать кнопку Finish (рисунок 1).

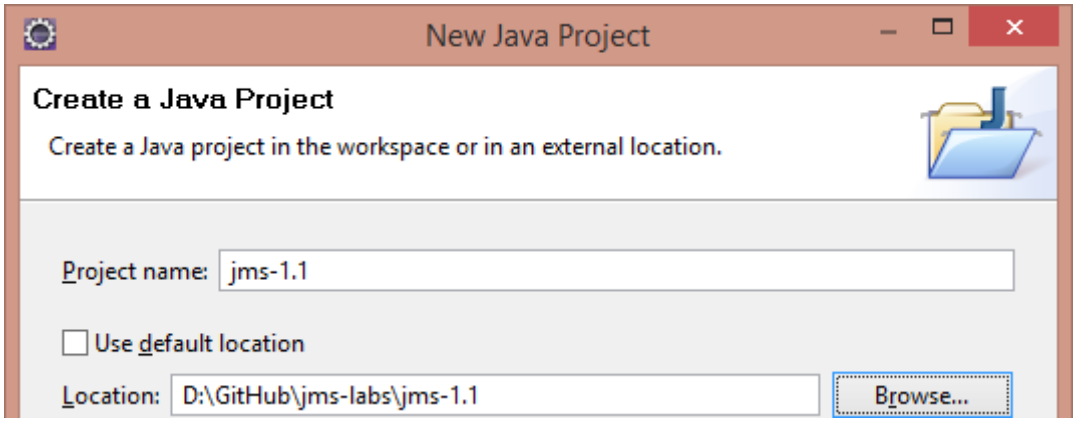

Рисунок 1

2. В созданном проекте щелкнуть правой кнопкой мыши по папке src, выбрать New -> Package, ввести в поле Name название создаваемого пакета и нажать кнопку Finish (рисунок 2).

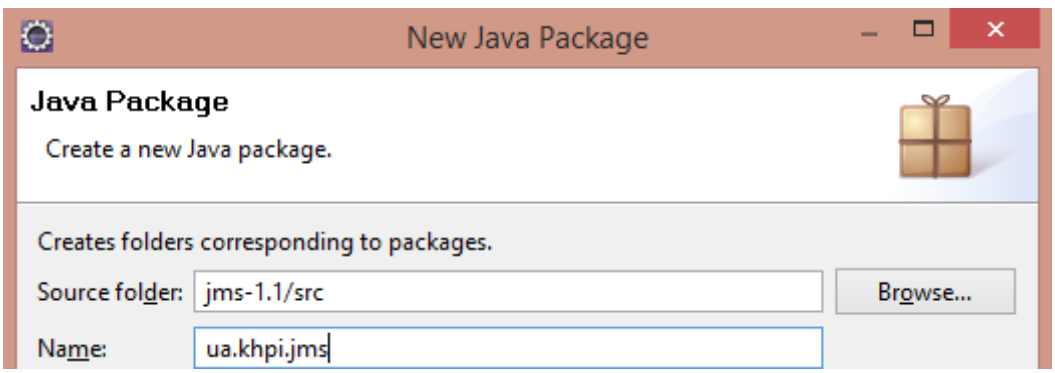

Рисунок 2

3. Аналогично, щелкнуть правой кнопкой мыши по созданному пакету, выбрать New -> Class, ввести в поле Name название создаваемого класса и нажать кнопку Finish (рисунок 3).

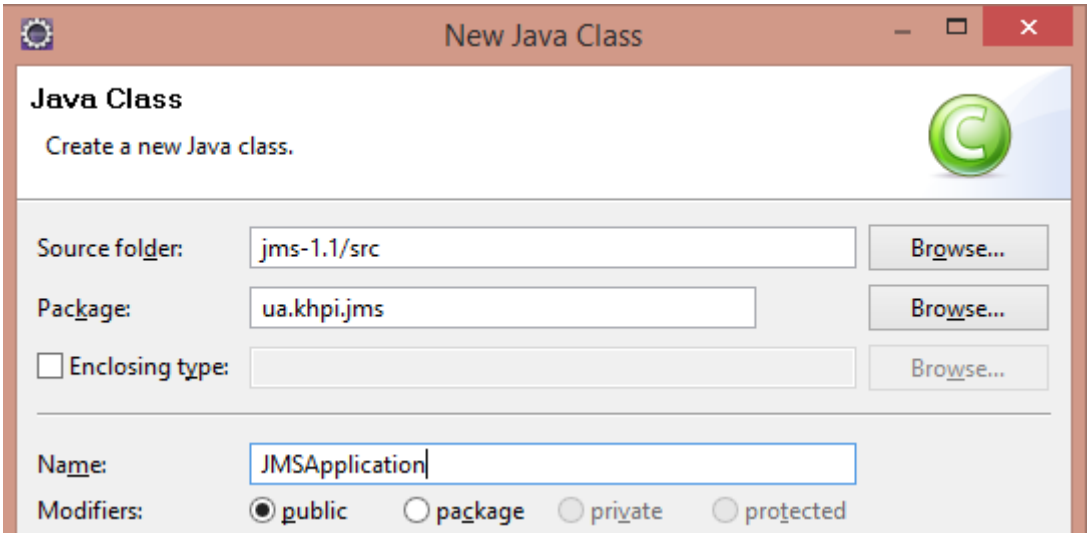

Рисунок 3

4. Для дальнейшей работы с технологией JMS, необходимо подключить библиотеку ActiveMQ. Для этого необходимо перейти в свойства созданного проекта (щелчком правой кнопкой мыши, выбрать пункт Properties), выбрать раздел Java Build Path, перейти во вкладку Libraries и, нажав кнопку Add External JARs (рисунок 4), добавить файл activemq-all.jar (располагается в каталоге с ApacheMQ).

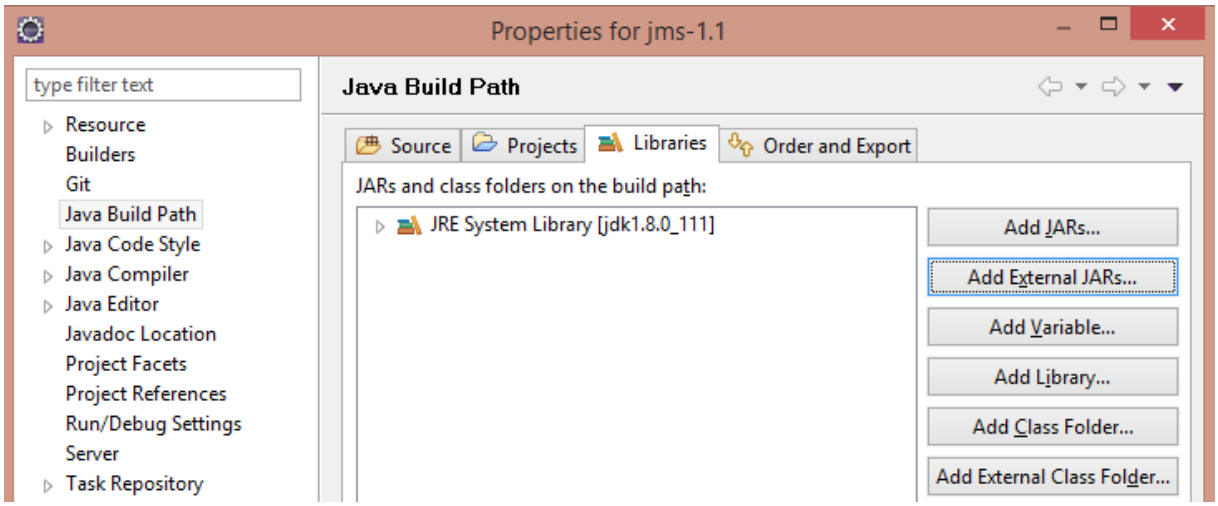

Рисунок 4

5. В созданном классе добавить поля connectionFactory, connection, session и destination (рисунок 5).

```
public class JMSApplication implements AutoCloseable {

    private ActiveMQConnectionFactory connectionFactory;
    private Connection connection;
    private Session session;
    private Destination destination;
```

Рисунок 5

6. Создать поля url и queue (рисунок 6).

```
private String url = "tcp://localhost:61616/";
private String queue = "test.in";
```

Рисунок 6

7. Описать метод, обеспечивающий подключение к серверу, получение соединения, создание контекста и получение буфера (рисунок 7).

```
public void init() throws JMSException {
    connectionFactory = new ActiveMQConnectionFactory(url);

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

    session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
    destination = session.createQueue(queue);
}
```

Рисунок 7

8. Описать метод, обеспечивающий отправку сообщения (рисунок 8).

```
public void send(String text) throws JMSException {
    MessageProducer messageProducer = session.createProducer(destination);

    TextMessage textMessage = session.createTextMessage(text);
    messageProducer.send(textMessage);
}
```

Рисунок 8

9. Описать метод, обеспечивающий получение сообщения (рисунок 9).

```
public String receive() throws JMSException {
    MessageConsumer messageConsumer = session.createConsumer(destination);

    TextMessage textMessage = (TextMessage) messageConsumer.receive();
    return textMessage.getText();
}
```

Рисунок 9

10. В определении класса указать, что он реализует интерфейс AutoCloseable (рисунок 10).

```
public class JMSApplication implements AutoCloseable
```

Рисунок 10

11. Описать автоматически вызываемый метод close() (рисунок 11).

```
@Override
public void close() throws JMSException {
    session.close();
    connection.close();
}
```

Рисунок 11

12. В созданный класс добавить метод main() (рисунок 12).

```
public static void main(String[] args) throws Exception {
    try (JMSApplication app = new JMSApplication()) {
        app.init();
        app.send("Test message");

        System.out.println(app.receive());
    }
}
```

Рисунок 12

13. Запустить разработанное приложение, используя кнопку Run (рисунок 13).

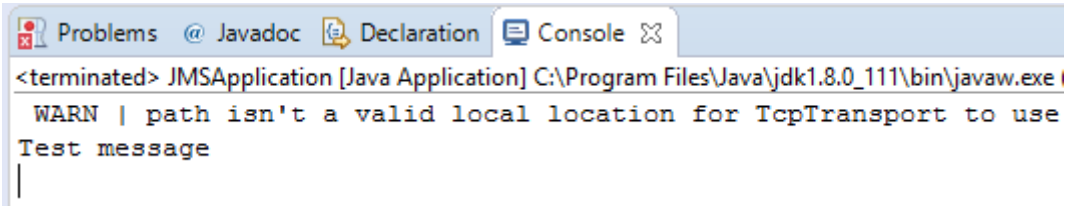

Рисунок 13

Таким образом, исходный код разработанного приложения будет следующим:

```
package ua.khpi.jms;

import javax.jms.Connection;
import javax.jms.Destination;
```

```java
import javax.jms.JMSException;
import javax.jms.MessageConsumer;
import javax.jms.MessageProducer;
import javax.jms.Session;
import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnectionFactory;

public class JMSApplication implements AutoCloseable {

	private ActiveMQConnectionFactory connectionFactory;
	private Connection connection;
	private Session session;
	private Destination destination;

	private String url = "tcp://localhost:61616/";
	private String queue = "test.in";

	public void init() throws JMSException {
		connectionFactory = new ActiveMQConnectionFactory(url);

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

		session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
		destination = session.createQueue(queue);
	}

	public void send(String text) throws JMSException {
		MessageProducer messageProducer =
session.createProducer(destination);

		TextMessage textMessage = session.createTextMessage(text);
		messageProducer.send(textMessage);
	}

	public String receive() throws JMSException {
		MessageConsumer messageConsumer =
session.createConsumer(destination);

		TextMessage textMessage = (TextMessage)
messageConsumer.receive();
		return textMessage.getText();
	}

	@Override
	public void close() throws JMSException {
		session.close();
		connection.close();
	}

	public static void main(String[] args) throws Exception {
		try (JMSApplication app = new JMSApplication()) {
			app.init();
			app.send("Test message");

			System.out.println(app.receive());
		}
	}
}
```

В отчете необходимо кратко описать основные этапы выполнения лабораторной работы, дать ответы на контрольные вопросы.

Контрольные вопросы

1. Сколько существует версий JMS? В чем заключается различие между JMS 1.0 и JMS 1.1?
2. Какие существуют основные интерфейсы JMS? Для чего они предназначены?
3. Какие этапы включает в себя алгоритм создания приложения, работающего с JMS?
4. Какие существуют типы JMS сообщений? В чем их особенности?
5. Из каких частей состоит любое JMS сообщение? Каково их назначение?

Дополнительные источники информации

1. Curry, E. (2004). Message-Oriented Middleware. In Middleware for Communications, ed. Qusay H Mahmoud. Chichester, England: John Wiley and Sons, pp. 1–28.
2. Richards, M., Richard M. H., David A. C. (2009). Java Message Service, Second Edition. O'Reilly.
3. Snyder, B., Bosanac, D., Davies, R. (2010). ActiveMQ in Action (1st ed.). Manning Publications, p. 375.
4. Герберт Шилдт. Java 8. Полное руководство, 9-е издание = Java 8. The Complete Reference, 9th Edition. – М.: «Вильямс», 2015. – 1376 с. – ISBN 978-5-8459-1918-2.
5. Кей С. Хорстманн. Java SE 8. Вводный курс = Java SE 8 for the Really Impatient. – М.: «Вильямс», 2014. – 208 с. – ISBN 978-5-8459-1900-7.