Лабораторная работа №3

Реализация клиент-серверного взаимодействия на основе SOAP поверх JMS

Цель: научиться работать с технологиями MDB, JMS, SOAP

Бины, управляемые сообщениями

Enterprise Bean – это компонент, расположенный на стороне сервера, инкапсулирующий бизнес-логику приложения.

Message-driven bean – это enterprise bean, который позволяет J2EE-приложениям обрабатывать сообщения асинхронно. Обычно он взаимодействует с слушателем сообщений JMS, который похож на слушатель событий с той лишь разницей, что он получает JMS-сообщения вместо событий. Сообщения могут отправлять J2EE-компоненты, JMS-приложение или система, не использующая технологию J2EE. Бин, управляемый сообщениями, может обрабатывать либо JMS-сообщения, либо иные виды сообщений.

Клиентские компоненты не размещают message-driven beans и не вызывают методы прямо на них. Вместо этого, доступ клиента к бину организовывается через JMS посредством отправки сообщений на конечный пункт назначения (destination), для которого класс бина реализует интерфейс MessageListener. Назначение пункта назначения для бина происходит в течение деплоймента с использованием ресурсов сервера приложения.

Message-driven bean'ы имеют следующие характеристики:

- Они исполняются после получения единственного клиентского сообщения
- Они вызываются асинхронно
- Они имеют относительно короткое время жизни
- Они прямо не представляют разделяемые данные в БД, но они могут получить доступ и изменять такие данные
- Они работают с транзакциями
- Они не сохраняют своего состояния

Когда приходит сообщение, контейнер вызывает метод onMessage бина для обработки сообщения. Метод onMessage обычно преобразует сообщение в одному из пяти типов сообщений и обрабатывает его в соответствии с положенной бизнес-логикой. OnMessage может вызывать вспомогательные методы, может вызывать сессионные и сущностные бины.

Сообщение может быть доставлено бину в контексте транзакции, так что все операции в пределах метода onMessage являются частью единой транзакции. Если над обрабатываемым сообщением выполняется откат (roll back), то сообщение будет удалено.

Сессионные и сущностные бины позволяют отправлять JMS-сообщения и получать их синхронно, но не асинхронно. Чтобы избежать чрезмерного использования ресурсов сервера, предпочтительно не использовать блокирующие синхронные получения в компоненте на стороне сервера. Чтобы получать сообщения асинхронно, используются бины, управляемые сообщениями (message-driven beans).

JMS Message-driven beans и длительные/недлительные подписчики

Длительная подписка (durable subscription) на тему означает, что JMS-подписчик получает все сообщения, даже если подписчик не активен. Если сообщение отправляется на топик, имеющий неактивного длительного подписчика, сообщение сохраняется и доставляется, когда тот становится активным.

Недлительная подписка (nondurable subscription) на топик означает, что подписчик получает только те сообщения, которые были опубликованы в то время, когда подписчик был активен. Другие сообщения теряются. Так как message-driven bean является по сути

потребителем, он может регистрировать себя длительным или недлительным подписчиком на сообщения топика.

JMS API

Клиент может посылать сообщения и получать от другого клиента. Каждый клиент соединяется к агенту, который предоставляет такие возможности. Компонент шлет сообщение в destination, а получатель восстановляет его из destination.

JMS API – это API, позволяющее приложениям создавать, посылать, получать и читать сообщения.

JMS и J2EE:

- Ejb-компоненты и web-компоненты могут отправлять или синхронно получать JMS-сообщения. Клиентские приложения вдобавок могут получать JMS-сообщения асинхронно.
- MDB способны асинхронно потреблять сообщения.

JMS-API архитектура:

- JMS-провайдер (включен в J2EE)
- JMS-клиенты (компоненты, которые производят и потребляют сообщения)
- Сообщения (носители информации между клиентами)
- Администрируемые объекты (используются клиентами)

Подходы к *messaging*:

Point-To-Point	**Pub/sub**
Домен предусматривает понятия:	Домен предусматривает понятия:
Queue (очередь)	*Topic* (тема)
Sender (отправитель)	*Publisher* (тот, кто публикует)
Receiver (получатель)	*Subscriber* (подписчик)

Строительные блоки JMS-приложения:

- *Администрируемые объекты.*
  - Destination. Объект, используемый для указания цели для сообщений, что он производит и источник сообщений, которые он потребляет.
  - Connection factory. Объект, используемый клиентом для создания соединения с провайдером.
- *Соединения*
  - Connection. Виртуальное соединение с JMS-провайдером.
- *Сессии*
  - Session. Однопоточный контекст для производства и потребления сообщений.
- *Message producers*
- *Message consumers*
- *Message Listeners*
- Сообщения
  - Messages. Состоят из заголовков (обязательно), свойств и тел (опционально).

Клиентское приложение J2EE

SimpleMessageClient посылает сообщения в очередь, которую прослушивает SimpleMessageBean. Клиент начинает работу с нахождения фабрики соединения и очереди:

```
Context jndiContext= new InitialContext (); //создание контнкста
//извлечение мастера (фабрики) соединения с очередью из контекста
QueueConnectionFactory queueConnectionFactory = (QueueConnectionFactory)
        jndiContext.lookup   ("java:comp/env/jms/MyQueueConnectionFactory");
Queue queue = (Queue)  jndiContext.lookup("java:comp/env/jms/QueueName");
```

// клиент создает соединение очереди, сеанс и отправителя:

```
queueConnection =
   queueConnectionFactory.createQueueConnection();
queueSession =
   queueConnection.createQueueSession(false,
   Session.AUTO_ACKNOWLEDGE);
queueSender = queueSession.createSender(queue);
```

// клиент посылает несколько сообщений в очередь:

```
TextMessage message;
message = queueSession.createTextMessage();
for (int i = 0; i < NUM_MSGS; i++) {
    message.setText("This is message " + (i + 1));
    System.out.println("Sending message: " +
       message.getText());
    queueSender.send(message);
}
```

Метод onMessage

Когда очередь получает сообщение, контейнер EJB вызывает метод onMessage бина, управляемого сообщениями.

В данном классе SimpleMessageBean метод onMessage преобразует сообщение в TextMessage и отображает текст:

```
public void onMessage(Message inMessage) {
   TextMessage msg = null;
   try {
 //если объект inMessage является   экземпляром класса
          if (inMessage instanceof  TextMessage) {
                    msg = (TextMessage) inMessage;
                    System.out.println
                    ("MESSAGE BEAN: Message received: "
                    + msg.getText());
     } else {
        System.out.println
          ("Message of wrong type: "
          + inMessage.getClass().getName());
     }
```

```
    } catch (JMSException e) {
        e.printStackTrace();
        mdc.setRollbackOnly(); // бин предписывает контейнеру откатить транзакцию
    } catch (Throwable te) {
        te.printStackTrace();
    }
}
```

В классе SimpleMessageBean методы ejbCreate и ejbRemove пустые.

Запуск сервера J2EE

Чтобы увидеть вывод бина, управляемого сообщениями, необходимо запустить сервер с опцией -verbose:

j2ee -verbose

Создание очереди

1. Создаем очередь при помощи команды j2eeadmin (j2eeadmin - команда добавление JDBC-драйверов, JMS-назначений и мастеров соединений для различных ресурсов):
   j2eeadmin -addJmsDestination jms/MyQueue queue

2. Проверяем, что очередь создана:
   j2eeadmin -listJmsDestination

Развертывание приложения

1. В deploytool откройте файл j2eetutorial/examples/ears/SimpleMessageApp.ear (File→Open).
2. Разверните приложение SimpleMessageApp (Tools→Deploy). В диалоговом окне Introduction убедитесь, что у вас выбрана радиокнопка Return Client JAR. Подробные инструкции см. ПРИЛОЖЕНИЕ 1.

1. В терминальном окне перейдите в каталог j2eetutorial/examples/ears.
2. Установите переменную окружения APPCPATH в SimpleMessageAppClient.jar.
3. Введите следующую команду (в одной строке):
4. runclient -client SimpleMessageApp.ear -name SimpleMessageClient -textauth
5.
4. В окне входа введите имя пользователя j2ee и пароль j2ee.
5. Клиент выводит такие строки:

Sending message: This is message 1
Sending message: This is message 2
Sending message: This is message 3

6. В терминальном окне, в котором вы стартовали сервер J2EE (в режиме verbose), будут выведены следующие строки:

MESSAGE BEAN: Message received: This is message 1
MESSAGE BEAN: Message received: This is message 2
MESSAGE BEAN: Message received: This is message 3

Индивидуальные задания:

Номер варианта соответствует номеру в журнале. Клиентское приложение отправляет несколько сообщений в QUEUE или TOPIC (нечетный вариант – QUEUE, четный – TOPIC). MDB получает и обрабатывает сообщения следующим образом:

1. Получатель сортирует сообщения в порядке возрастания и таким образом записывает их в текстовый файл.
2. Получатель ищет два одинаковых сообщения и записывает их порядковые номера в файл.
3. Получатель записывает в текстовый файл сообщения с восклицательным знаком.
4. Получатель выводит в консоль сообщения длиной не менее четырех символов отсортированные по алфавиту.
5. Получатель производит поиск сообщения (искать в файле).
6. Получатель записывает приходящие цифры в один файл, буквы – в другой.
7. Получатель записывает полученные сообщения в файл задом наперед.
8. Получатель считает среднюю оценку по трем предметам и записывает её в файл.
9. Получатель сортирует сообщения в порядке возрастания (по длине сообщения) и таким образом записывает их в текстовый файл.
10. Получатель записывает в файл 2 сообщения, которые имеют наименьшее и наибольшее число гласных.

ПРИЛОЖЕНИЕ 1.

Развертывание приложения J2EE

Полный tutorial