

---

# СУПЕРКОМПЬЮТЕРНЫЙ КОНСОРЦИУМ УНИВЕРСИТЕТОВ РОССИИ

## Проект

*Создание системы подготовки  
высококвалифицированных кадров  
в области суперкомпьютерных  
технологий и специализированного  
программного обеспечения*



Московский государственный университет  
им. М.В. Ломоносова

---

# Методы и средства построения распределенных программных систем с использованием технологии Java

Лекция 15.

JMS

# Лекция 15

- Другой подход к построению распределенных приложений: системы построенные на обмене сообщениями
- Что такое JMS?
  - История
  - Область применимости
  - Элементы архитектуры
- Пример использования

# Модель, основанная на событиях

- Объекты реагируют на события (например, вызовом методов)
- События могут изменять состояние объектов
- Множество объектов, расположенных на различных узлах могут извещаться о наступлении событий определенного типа
- Используется модель «издатель-подписчик»
  - Объект, генерирующий события, называется издателем - он публикует список типов событий, на которые остальные объекты могут подписаться
  - Объекты, подписывающиеся на события определенного типа называются подписчиками. Они получают уведомление о наступлении события, на которое они подписались

# Модель, основанная на событиях

- Связь между компонентами – слабая
  - Компонент может получить событие и преобразовать его к нужному виду
- Не требуется одновременного наличия «на связи» обменивающихся сообщениями КОМПОНЕНТОВ
  - Синхронные и асинхронные механизмы приема и обработки сообщений
  - Сохранение сообщений в буфере, в случае отсутствия получателя
- Возможность как широковещательной рассылки сообщений, так и избирательной

# JMS

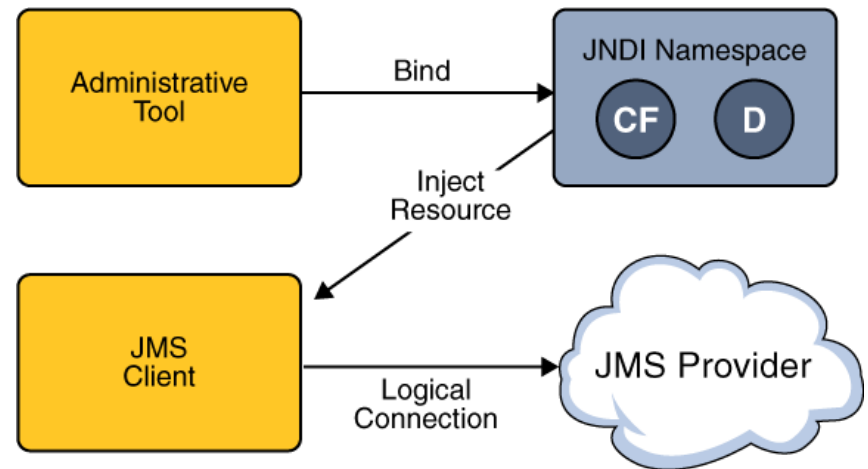
- Реализация событийной модели от Sun
- Входит в состав J2EE
- Обеспечивает механизм обмена сообщениями:
  - Асинхронный
  - Надежный
- Проверенная временем технология
  - Первая спецификация опубликована в 1998 г.
- Реализуется во многих продуктах сторонних разработчиков

# Рекомендуется использовать, если...

- необходимо обеспечить независимость модуля от интерфейса взаимодействия с другим модулем (grc и rti требуют знания интерфейса удаленного модуля)
- компоненты должны работать независимо и могут выполняться в разное (!) время
- структура приложения такова, что информацию можно передать в другой модуль, не дожидаясь ответа и каких-либо результатов обработки

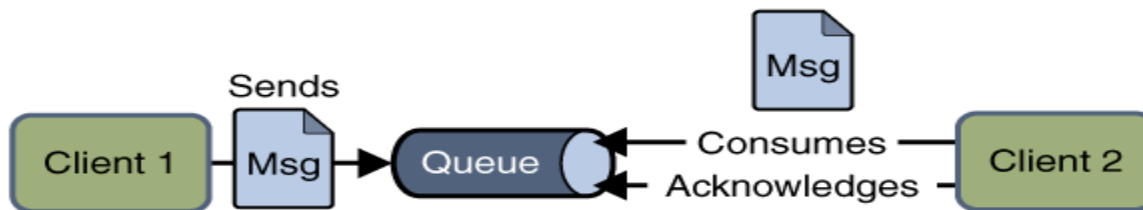
# Архитектура

- JMS Provider – система, реализующая интерфейс JMS и предоставляющая средства администрирования
- JMS Clients – компоненты, посылающие и принимающие сообщения
- Messages – сообщения (объекты, передающиеся и принимающиеся компонентами)
- Administrative tools – средства администрирования

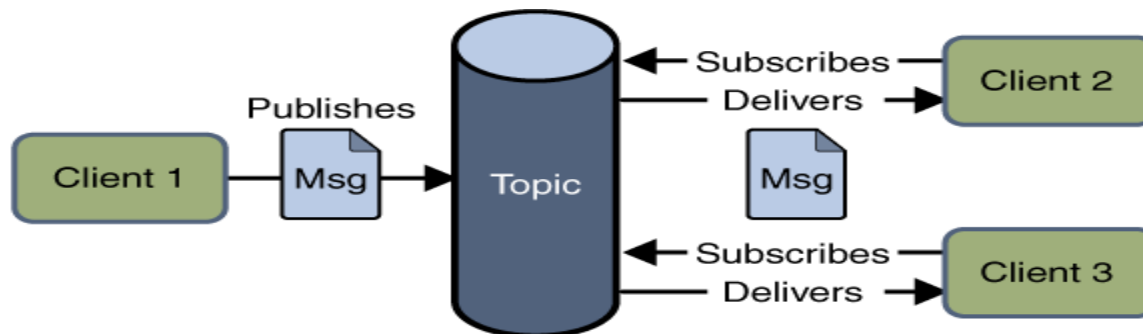




# JMS – два подхода



- Модель точка-точка для обмена сообщениями между двумя компонентами



- Модель «издатель-подписчик» для обмена сообщениями между многими компонентами

# Модель точка-точка

- Область применимости:
  - Взаимодействие двух компонентов
- Элементы модели:
  - Очередь
  - Отправитель
  - Получатель
- Каждое сообщение обрабатывается только одним клиентом
- Зависимости по времени между отправителем и получателем нет
  - Получатель может запуститься и обработать сообщения, когда отправитель работу уже закончил (или нет)

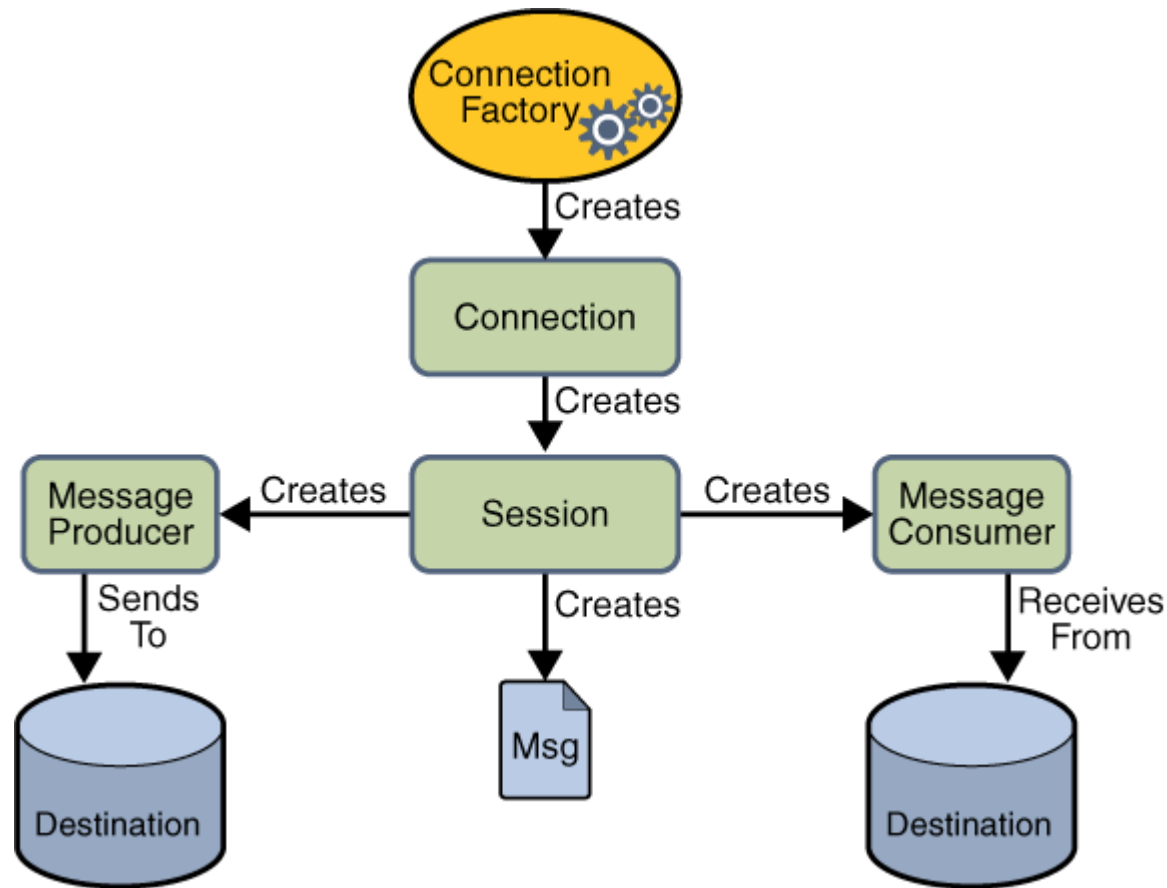
# Модель «издатель - подписчики»

- Область применимости:
  - Взаимодействие нескольких компонентов
- Элементы модели:
  - «тема» (topic)
  - Издатель
  - Подписчик
- Сообщение может быть получено несколькими подписчиками
- Подписчик получает только те сообщения, что были отправлены после его подписки на «тему»

# Обработка сообщений

- Синхронная
  - Клиент вызывает блокирующий метод receive
- Асинхронная
  - Клиент регистрирует «слушателя» (listener), метод которого вызывается при поступлении сообщения

# Схема работы с API JMS



# Connection Factory

- Используется для создания соединения с провайдером
- Как правило связывается с определенным ресурсом посредством механизма аннотаций

```
@Resource(mappedName="jms/ConnectionFactory")  
private static ConnectionFactory connectionFactory;
```

# Destination

- Используется для указания объекта назначения для сообщения
- Поддерживается два типа:
  - Очередь (Queue)
  - Тема (Topic)
- Как правило связывается с определенным ресурсом посредством механизма аннотаций

```
@Resource(mappedName="jms/Queue")  
private static Queue queue;
```

```
@Resource(mappedName="jms/Topic")  
private static Topic topic;
```

# Connection

- Инкапсулирует соединение с провайдером
- Используется для создания сессий

```
Connection connection = connectionFactory.createConnection();
```

- После использования соединение должно быть закрыто

```
connection.close();
```



# Session

- Сессия представляет собой однопоточный контекст для создания следующих объектов:
  - Отправителей сообщений (Message Producers)
  - Приемщиков сообщений (Message Consumers)
  - Сообщений (Messages)
  - Просмотрщиков очередей (Queue Browsers)
  - Вспомогательных очередей и тем (Temporary queues and topics)

- Создание сессии:

```
Session session = connection.createSession(false,  
Session.AUTO_ACKNOWLEDGE);
```

- Первый аргумент – поддерживаются транзакции или нет
- Оповещение об успешной обработке сообщения

---

# Типы сообщений

- `TextMessage` – текстовое сообщение
- `MapMessage` – пары «ключ-значение»
- `BytesMessage` – поток байт
- `StreamMessage` – поток примитивных типов Java
- `ObjectMessage` – сериализуемый объект
- `Message` – пустое сообщение

# Схема программы – отправителя сообщений

```
@Resource(mappedName="jms/ConnectionFactory")
private static ConnectionFactory connectionFactory;

@Resource(mappedName="jms/Queue")
private static Queue queue;

//@Resource(mappedName="jms/Topic")
//private static Topic topic;

Destination dest = queue;
Connection connection =
    connectionFactory.createConnection();
Session session = connection.createSession(false,
    Session.AUTO_ACKNOWLEDGE);
MessageProducer producer =
    session.createProducer(dest);
TextMessage message = session.createTextMessage();

message.setText("This is message " + (i + 1));
producer.send(message);

connection.close();
```

- Создается объект Connection, с которым связывается ресурс (механизм аннотаций)
- Создается очередь (тема)
- Определяется Destination
- Создается соединение и сессия
- Создается MessageProducer и сообщение (в данном случае - текстовое)
- В тело сообщения записываются текстовые данные и сообщение посылается
- Соединение закрывается

# Схема программы – читателя сообщений (синхронный прием)

```
@Resource(mappedName="jms/ConnectionFactory")  
private static ConnectionFactory connectionFactory;
```

```
@Resource(mappedName="jms/Queue")  
private static Queue queue;
```

```
//@Resource(mappedName="jms/Topic")  
//private static Topic topic;
```

```
Destination dest = queue;  
Connection connection =  
    connectionFactory.createConnection();  
Session session = connection.createSession(false,  
    Session.AUTO_ACKNOWLEDGE);  
consumer = session.createConsumer(dest);  
connection.start();
```

```
Message m = consumer.receive();  
message = (TextMessage) m;  
    System.out.println(message.getText());
```

```
connection.close();
```

- Создается объект Connection, с которым связывается ресурс (механизм аннотаций)
- Создается очередь (тема)
- Определяется Destination
- Создается соединение и сессия
- Создается MessageConsumer и запускается обработка сообщений
- Читается и печатается сообщение (receive - блокирующий)
- Соединение закрывается

# Схема программы – читателя сообщений (асинхронный прием)

```
@Resource(mappedName="jms/ConnectionFactory")  
private static ConnectionFactory connectionFactory;
```

```
@Resource(mappedName="jms/Queue")  
private static Queue queue;
```

```
//@Resource(mappedName="jms/Topic")  
//private static Topic topic;
```

```
Destination dest = queue;  
Connection connection =  
    connectionFactory.createConnection();  
Session session = connection.createSession(false,  
    Session.AUTO_ACKNOWLEDGE);  
consumer = session.createConsumer(dest);
```

```
listener = new XXXListener();  
consumer.setMessageListener(listener);  
connection.start();
```

- Создается объект Connection, с которым связывается ресурс (механизм аннотаций)
- Создается очередь (тема)
- Определяется Destination
- Создается соединение и сессия
- Создается MessageConsumer
- Создается листенер
- Листенер связывается с MessageConsumer и запускается обработка сообщений
- При возникновении сообщения будет вызван метод листенера `public void onMessage(Message message)`

# Схема компиляции и запуска

- Создание ресурсов (очередей или тем)
- Компиляция
  - Программы отправляющей сообщений
  - Программы принимающей сообщения
- Запуск
  - Программы отправляющей сообщений
  - Программы принимающей сообщения

# Создание ресурсов и компиляция

- Создание ConnectionFactory

```
asadmin.bat create-jms-resource --user admin --passwordfile  
admin-password.txt --host localhost --port 4848 --restype  
    javax.jms.ConnectionFactory --enabled=true jms/ConnectionFactory
```

- Создание очереди

```
asadmin.bat create-jms-resource --user admin --passwordfile  
admin-password.txt --host localhost --port 4848 --restype javax.jms.Queue --  
    enabled=true --property Name=PhysicalQueue jms/Queue
```

- Компиляция

```
javac -classpath .; activation.jar; appserv-ws.jar; javaee.jar; jsf-impl.jar;  
    appserv-jstl.jar; -d build *.java
```

- Создание архива

```
jar -cvfm cl.jar manifest.mf -C ./build
```

- Выполнение

```
appclient -client cl.jar
```

# Просмотр ресурсов в административной консоли

The screenshot shows the Sun Java System Application Server Admin Console in Microsoft Internet Explorer. The browser address bar shows `http://localhost:4848/asadmin/amingui/TopFrameset`. The page title is "Sun Java™ System Application Server Admin Console". The user is logged in as "admin" on "localhost" in the "domain1" domain. The left navigation pane shows a tree view with "JMS Resources" expanded to "Destination Resources". The main content area displays "JMS Destination Resources" with a description: "JMS destinations serve as the repositories for messages. Click New to create a new destination resource. Click the name of a destination resource to modify its properties." Below the description is a table titled "Destination Resources (1)". The table has columns for "JNDI Name", "Enabled", and "Description". The first row contains "jms/Queue", "true", and an empty description. A red circle highlights the "New" and "Delete" buttons above the table.

JNDI Name	Enabled	Description
jms/Queue	true	

The screenshot shows the Sun Java System Application Server Admin Console in Microsoft Internet Explorer. The browser address bar shows `localhost:4848/asadmin/amingui/TopFrameset`. The page title is "Sun Java™ System Application Server Admin Console". The user is logged in as "admin" on "localhost" in the "domain1" domain. The left navigation pane shows a tree view with "JMS Resources" expanded to "Connection Factories". The main content area displays "JMS Connection Factories" with a description: "Java Message Service (JMS) connection factories are objects that allow an application to create other JMS objects programmatically. Click New to create a new connection factory. Click the name of a connection factory to modify its properties." Below the description is a table titled "Connection Factories (1)". The table has columns for "JNDI Name", "Enabled", and "Description". The first row contains "jms/ConnectionFactory", "true", and an empty description. A red circle highlights the "New" and "Delete" buttons above the table.

JNDI Name	Enabled	Description
jms/ConnectionFactory	true	



# Итоги

- JMS – реализация концепции построения приложений, основанных на обмене событиями
- Существуют различные реализации JMS
- Поддерживает две концепции обмена:
  - Точка-точка
  - Издатель–подписчик
- Поддерживает как синхронный, так и асинхронный обмен сообщениями
- Является удобным механизмом для создания класса систем, нуждающихся в обмене сообщениями

# О проекте

**Целью проекта** является создание национальной системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения.

**Задачами** по проекту являются:

**Задача 1.** Создание сети научно-образовательных центров суперкомпьютерных технологий (НОЦ СКТ).

**Задача 2.** Разработка учебно-методического обеспечения системы подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий.

**Задача 3.** Реализация образовательных программ подготовки, переподготовки и повышения квалификации кадров в области суперкомпьютерных технологий.

**Задача 4.** Развитие интеграции фундаментальных и прикладных исследований и образования в области суперкомпьютерных технологий. Обеспечение взаимодействия с РАН, промышленностью, бизнесом.

**Задача 5.** Расширение международного сотрудничества в создании системы суперкомпьютерного образования.

**Задача 6.** Разработка и реализации системы информационного обеспечения общества о достижениях в области суперкомпьютерных технологий.

См. <http://www/hpc-education.ru>