Servicios de Mensajes en JEE

Sesión1: JMS: Java Message Service (2)

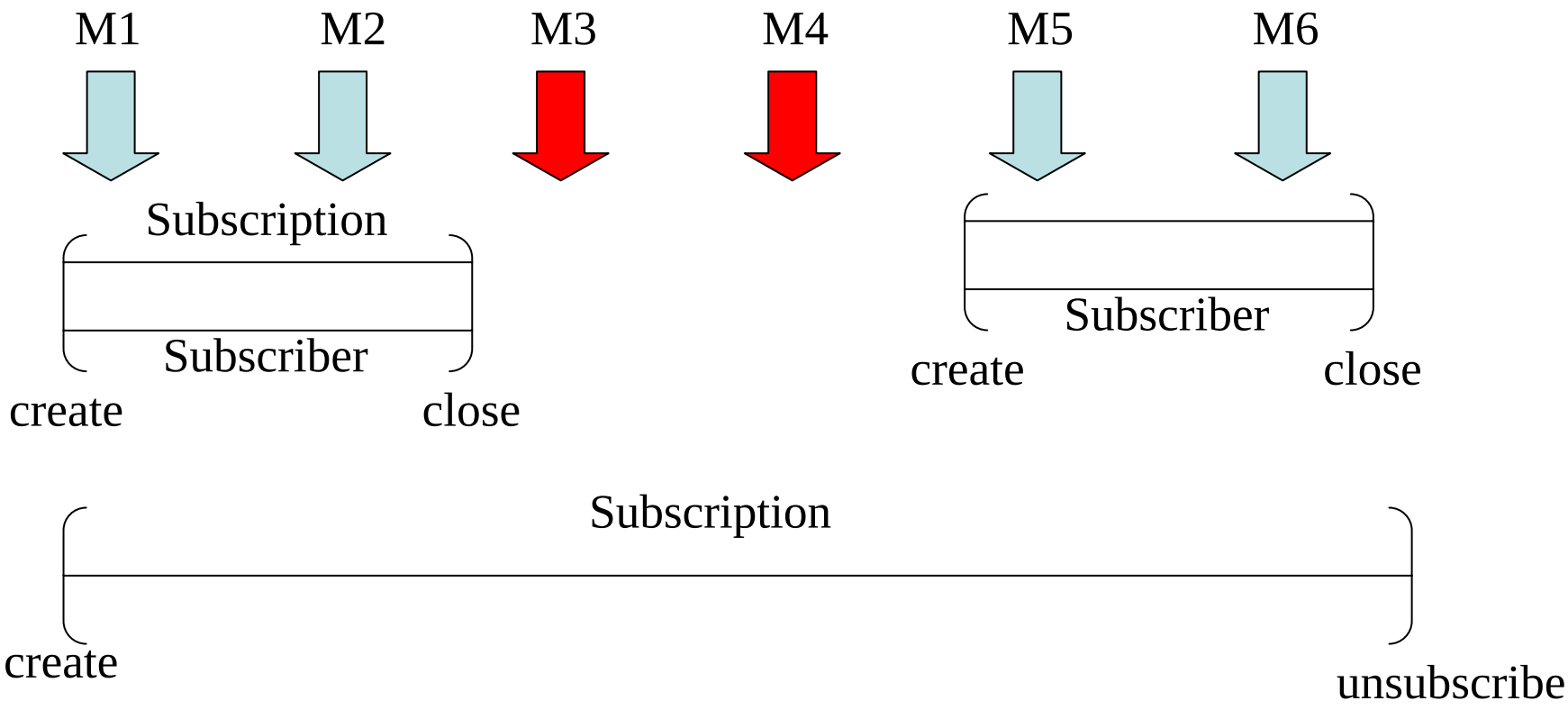
Indice

- Durable subscribers
- Browsing de mensajes
- Interación servlets y JMS
- Transacciones JMS

Indice

- Durable Subscribers
- Browsing de mensajes
- Interacción servlets y JMS
- Transacciones JMS

Normal vs Durable Subscribers



Durable subscribers

- Pasos a seguir:
  1. Si no existe un Persistent Store Crearlo.
     - Antes hay que haber creado un servidor JMS.
  2. Después de la conexión crear inmediatamente un identificador de cliente:

```
topicConnection = topicConnectionFactory.createTopicConnection();
topicConnection.setClientID("sco");
```

  3. Suscribirse a la conexión:

```
String nombre = "mySub";
TopicSubscriber topicSubscriber =
    createDurableSubscriber(mytopic, nombre);
```

Durable subscribers

- Pasos a seguir:

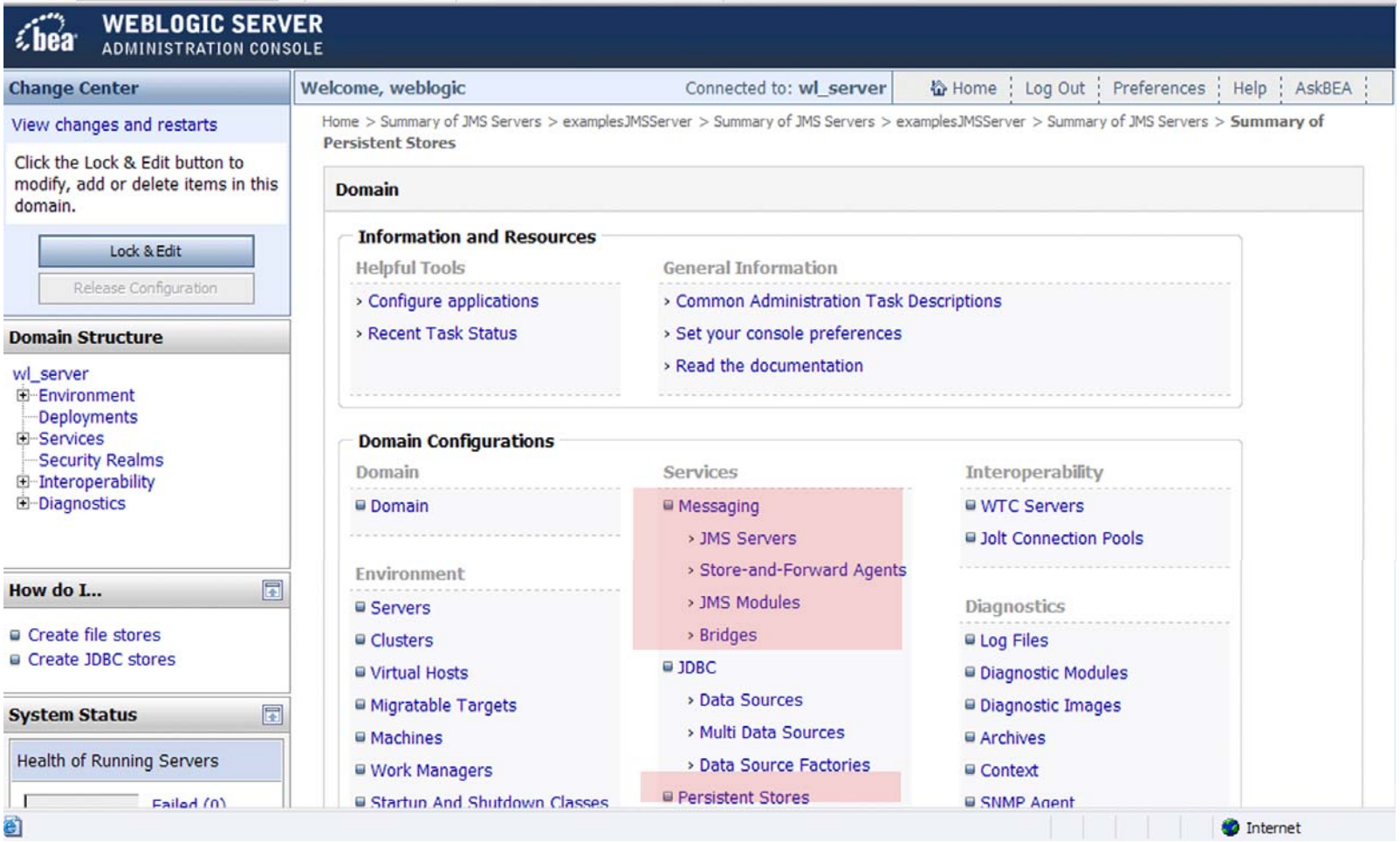
4. Más tarde puede cerrarse el TopicSubscriber:

```
topicSubscriber.close()
```

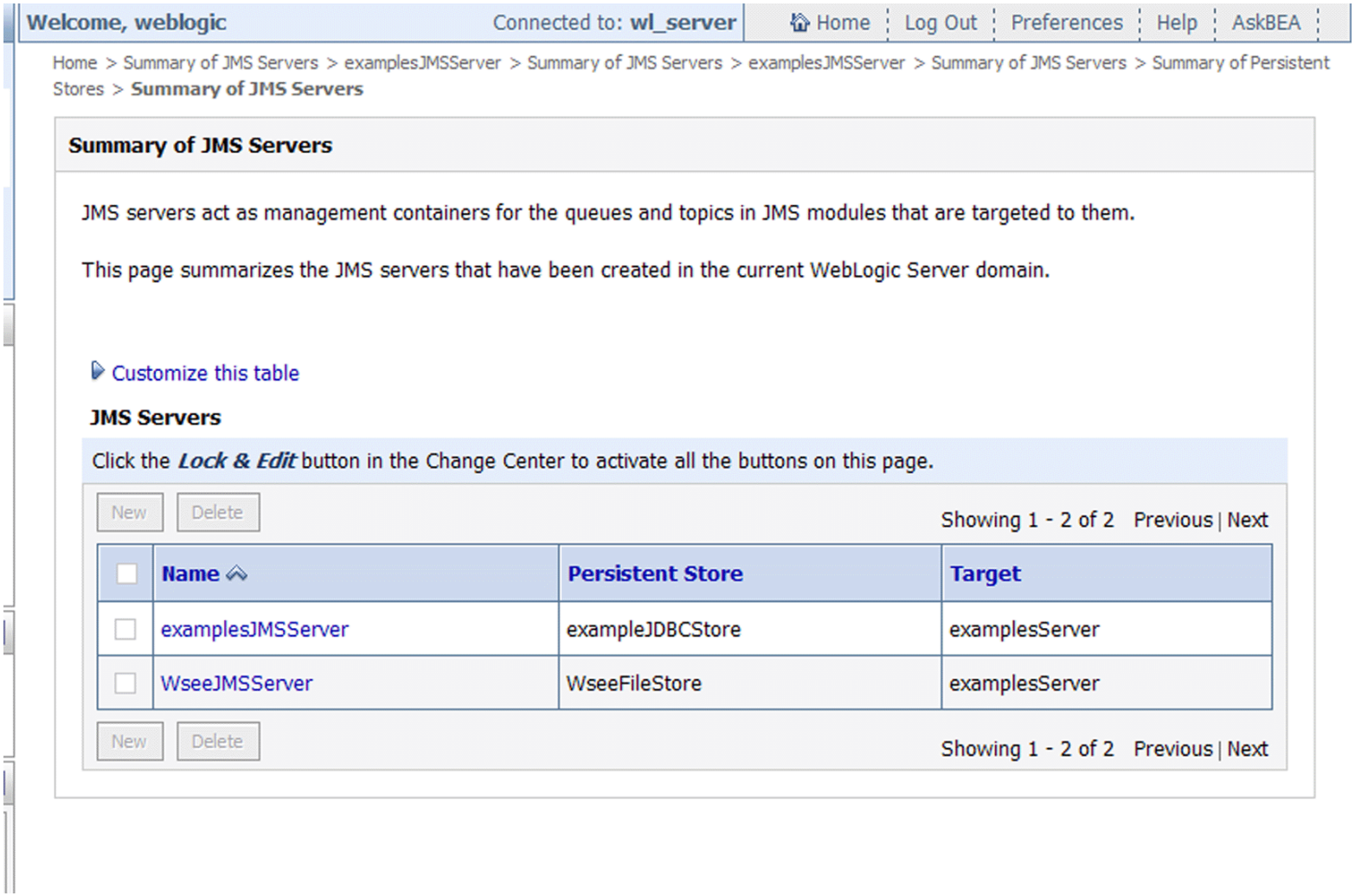
5. Los mensajes estarán disponibles mientras no se desactive la subscripción.
   - Si el programa u otra aplicación llama a createDurableSubscriber con el mismo id de cliente y al mismo tópico entonces la subscripción se reactiva y se recuperan los mensajes del almacén.

```
topicSubscriber.close();
topicSession.unsubscribe("MySub");
```

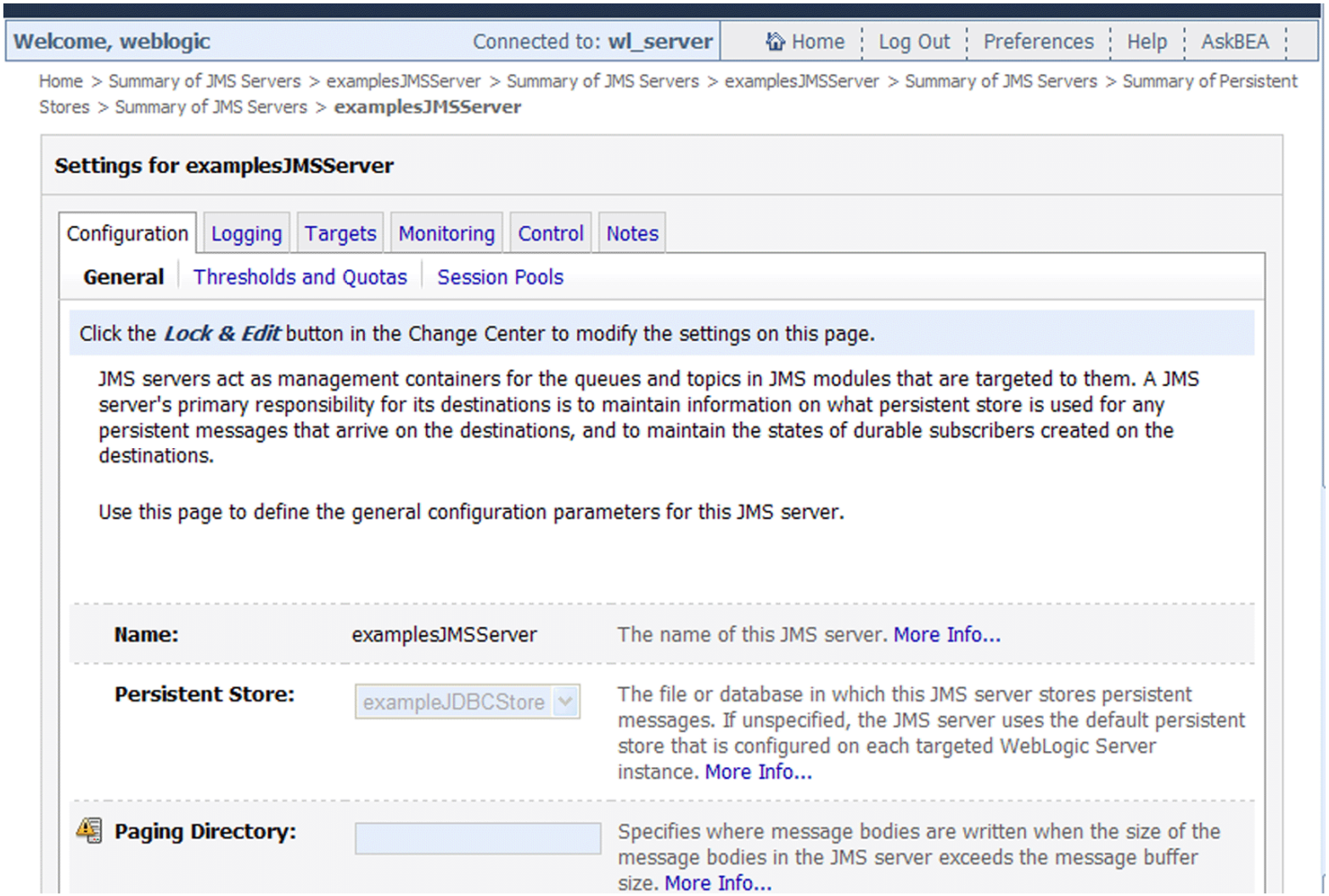
Durable Subscribers



Durable Subscribers



Durable Subscribers



Indice

- Durable Subscribers
- Browsing de mensajes
- Interacción servlets y JMS
- Transacciones JMS

Browser de mensajes

- Ejemplo: `QueueBrowse.java`
  1. Establecer un contexto JNDI
  2. Configuración de la comunicación
  3. Crear `QueueBrowser:createBrowser()`
  4. Mostrar la cola: `displayQueue()`
     1. Obtener mensajes con `getEnumeration()`
     2. Recorrer mensajes mostrando sus propiedades:

        `JMSMessageID, JMSTimestamp, JMSDestination, JMSExpiration, JMSPriority, JMSDeliveryMode, JMSCorrelationID, JMSReplyTo, JMSType`
  5. Cerrar la comunicación

Browser de mensajes

```
Enumeration e = qbrowser.getEnumeration();
...
if (! e.hasMoreElements()) {
      System.out.println("There are no messages on this queue.");
    } else {
      System.out.println("Queued JMS Messages: ");
      while (e.hasMoreElements()) {
        m = (Message) e.nextElement();
        System.out.println("Message ID " + m.getJMSMessageID() +
                           " delivered " +
                           new Date(m.getJMSTimestamp()) +
                           " to " + m.getJMSDestination());
        System.out.print("\tExpires        ");
        if (m.getJMSExpiration() > 0) {
          System.out.println( new Date( m.getJMSExpiration()));
            }
        else
          System.out.println("never");
```

Browser de mensajes

```
System.out.println("\tPriority       " + m.getJMSPriority());
        System.out.println("\tMode           " + (
                    m.getJMSDeliveryMode() == DeliveryMode.PERSISTENT ?
                                        "PERSISTENT" : "NON_PERSISTENT"));
        System.out.println("\tCorrelation ID " + m.getJMSCorrelationID());
        System.out.println("\tReply to       " + m.getJMSReplyTo());
        System.out.println("\tMessage type   " + m.getJMSType());
        if (m instanceof TextMessage) {
          System.out.println("\tTextMessage    \"" +
((TextMessage)m).getText() + "\"");
        }
      }
    }
```

Indice

- Durable Subscribers
- Browsing de mensajes
- Interacción servlets y JMS
- Transacciones JMS

Interacción de Servlets y JMS

- Ejemplo: `SenderServlet.java`

1. Preparar variables

```
boolean persistent;
String topicMsg="";
int priority;
long ttl; // time to live
...
```

2. Display la página web para recoger los parámetros necesarios.
3. Enviar mensaje a un Topic o a una Queue:

```
sendTopicMessage()
sendQueueMessage()
```

Transacciones en JMS

- Ejemplo: `JMSDrawDemo.java` (en WL 7.0)
  - Funcionamiento:
    - Cada cliente accede en modo Pub/sub a un mismo `Topic` para publicar o recibir (en modo asíncrono) mensajes.
    - Cada uno crea un panel en donde se puede seleccionar si la sesión va a ser transaccional o no.
    - Para publicar un mensaje hay que pinchar con el ratón en el canvas, por lo que un mensaje se construye a partir de las coordenadas seleccionadas.
    - Normalmente los mensajes consisten en óvalos sólidos, pero cuando se aplica un `commit()` en una sesión transaccional entonces estos se convierten en imágenes con un icono asociado
    - En caso de `rollback()` estos mensajes (aunque estén en la ventana de otro cliente) son eliminados del sistema.

Transacciones en JMS

- Ejemplo: `JMSDrawDemo.java`
  1. Inicialización de sesión: `initJMS()` (desde constructor)
     - Se crea una sesión transaccional y otra no transaccional al aplicar el `createTopicSession()`.

```
session = connection.createTopicSession(false,
Session.AUTO_ACKNOWLEDGE);
sessionTX = connection.createTopicSession(true,
Session.AUTO_ACKNOWLEDGE);
```

     - A partir de este punto, los `TopicPublisher` que se creen

```
topic = (Topic) ctx.lookup(TOPIC);
publisher = session.createPublisher(topic);
publisherTX = sessionTX.createPublisher(topic);
```

Transacciones en JMS

- Ejemplo: `JMSDrawDemo.java`
  1. Inicialización de sesión: `initJMS()`
     - En cuanto al `TopicSubscriber` que se crea para cada cliente, solo interesa especificar si su tipo es “durable” o no (`String`).
     - En el primer caso, forzamos a JMS a mantener una capa de persistencia para asegurarse de que o bien el mensaje ha sido entregado o ha expirado (incluso si el “subscriber” no estaba activo en el momento en que se produjo el mensaje).
       - `createSubscriber()`
       - `createDurableSubscriber()`
     - A continuación, el registro del “listener” con el método `setMessageListener()`
     - Finalmente, se crea un mensaje vacío y se inicia la conexión.

Transacciones en JMS

```
if (durableSubscriberID != null) {
      connection.setClientID("JMSDrawDemo" + durableSubscriberID);
    }
    session = connection.createTopicSession(false, Session.AUTO_ACKNOWLEDGE);
    sessionTX = connection.createTopicSession(true, Session.AUTO_ACKNOWLEDGE);
    topic = (Topic) ctx.lookup(TOPIC);
    publisher = session.createPublisher(topic);
    publisherTX = sessionTX.createPublisher(topic);

    if (durableSubscriberID == null) {
      subscriber = session.createSubscriber(topic, "TRUE", noLocal);
    } else {
      subscriber = session.createDurableSubscriber(topic,
durableSubscriberID);
    }
    subscriber.setMessageListener(this);
    msg = session.createMessage();
    connection.start();
  }
```

Transacciones en JMS

- Ejemplo: `JMSDrawDemo.java`

2. Publicar: `publishPoint()`
   - Activado por el manejador de eventos de ratón `mouseReleased()` pasando las coordenadas del punto.
   - Cada punto se añade a un vector de puntos `pointsSet`.
   - El mensaje es una variable global `msg` que se inicializó vació en initJMS(). Ahora se especifican dos propiedades enteras con el método `setIntProperty()` (tipo MAP).
   - El “delivery mode” se obtiene aparte con `deliveryMode()`
   - A continuación se procede a publicar el mensaje, de una forma o de otra dependiendo si la sesión es transaccional o no. Se llama al método `publish()`

Transacciones en JMS

```
private void publishPoint(int x, int y)
  {
    try {
      msg.setIntProperty("x", x);
      msg.setIntProperty("y", y);
      pointsSent.addElement(new Point(x, y));
      if (isTransacted) {
        publisherTX.publish(msg, deliveryMode(), 5, 0);
      } else {
        publisher.publish(msg, deliveryMode(), 5, 0);
      }
    } catch (JMSException jmse) {
      jmse.printStackTrace();
    }
  }
```

Transacciones en JMS

- Ejemplo: `JMSDrawDemo.java`

3. Recibir: `onMessage()`
   - Primero se controla si hay que borrar la pantalla. Cuando el manejador del botón de borrado es `publishClear()` que se encarga de publicar un mensaje MAP “clearScreen”:

```
private void publishClear()
  {
    try {
      Message clearMsg = session.createMessage();
      clearMsg.setStringProperty("command", "clearScreen");
      publisher.publish(clearMsg, deliveryMode(), 5, 0);
    } catch (JMSException jmse) {
      jmse.printStackTrace();
    }
  }
```


- El método construye y envía un “mensaje de borrado”

Transacciones en JMS

- Ejemplo: `JMSDrawDemo.java`

3. Recibir: `onMessage()`
   - Entonces desde `onMessage()` leeríamos un posible mensaje de borrado de pantalla y en ese caso deberíamos llamar al método `clearScreen()` para proceder al borrado efectivo, lo cual conlleva hacer un “rollback” en el caso transaccional:

```
private void clearScreen()
  {
    if (isTransacted) {
      doRollback();
    }
    pointsSent.removeAllElements();
    points.removeAllElements();
    offScreenImage = null;
    repaint();
  }
```

Transacciones en JMS

```
 public void onMessage(Message msg)
{
  try {
    String command = msg.getStringProperty("command");
    if (command != null && command.equals("clearScreen")) {
      clearScreen();
    } else {
      int x = msg.getIntProperty("x");
      int y = msg.getIntProperty("y");
      Point p = new Point(x,y);
      updateImage(p);
    }
  } catch (JMSException e) {
    e.printStackTrace();
  }
}
```

Transacciones en JMS

- Ejemplo: `JMSDrawDemo.java`

3. Recibir: `onMessage()`(si no “clearScreen”)
   - Dado que los mensajes son MAP, la forma de obtener los valores de las propiedades es llamar a `getIntPropery()` especificando el nombre de la propiedad (“x” o “y”).
   - Se construye un nuevo punto con las coordenadas.
   - Se “actualiza” la imagen asociada al punto con `updateImage()` Dicha actualización consiste en pintar un asterisco en el caso transaccional o un icono en el caso no-transaccional.
   - En una sesión transaccional, los asteriscos no se pintan hasta que se hace un “commit” y se borran del panel si se decide hacer un “rollback”.
   - Estos métodos se llaman desde `doCommit()` y `doRollback()`

Transacciones en JMS

```
 private void doCommit()
{
  try {
    sessionTX.commit();
    commitButton.setEnabled(false);
    rollbackButton.setEnabled(false);
    repaint();
  } catch (JMSException jmse) {
    jmse.printStackTrace();
  }
}
```

```
private void doRollback()
  {
    try {
      sessionTX.rollback();
      pointsSent.removeAllElements();
      commitButton.setEnabled(false);
      rollbackButton.setEnabled(false);
      offScreenImage = null;
      repaint();
    } catch (JMSException jmse) {
      jmse.printStackTrace();
    }
  }
```

Ejercicios…

- Durable Subscriber
  - Realizad un cliente **DurableSubscriberExample** que demuestre el funcionamiento de la subscripción duradera.

1. Se declara un DurableSubscriber asociado a un tópico
2. Un cliente publica los mensajes 1,2 y 3 en ese topicoç
3. El suscriptor los lee y después se cierra.
4. El cliente publica los mensajes 4, 5 y 6 en ese tópico.
5. El subscriptor de crea de nuevo. Leerá los mensajes correspondientes.
6. El subsciptor se cierra.
7. La subscripción duradera se cancela.

Ejercicios…

- Ejemplos de transacciones (1)
  - Probad el ejemplo `examples.jms.queue.QueueReceiveInTx` para para mostrar el funcionamiento de un cliente transaccional.
  - Interaccionad mediante cliente que envíen mensajes.
  - Simulad un rollback y estudiar los efectos.
- Ejemplo transacciones (2) (Optativo)
  - Adaptad el ejemplo JMSDrawDemo de la version 7.0
  - Mostrad el funcionamiento de las transacciones tanto JMS como JTA.