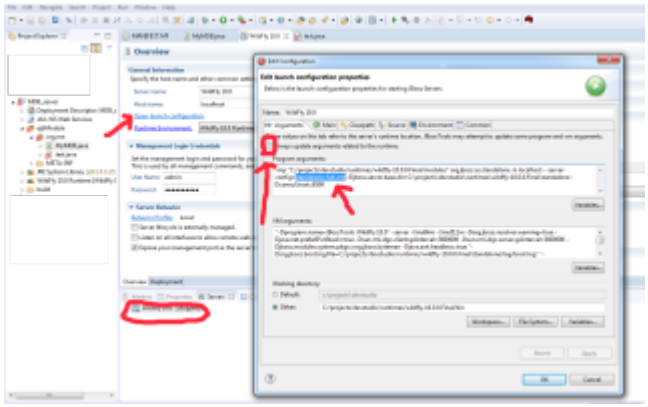
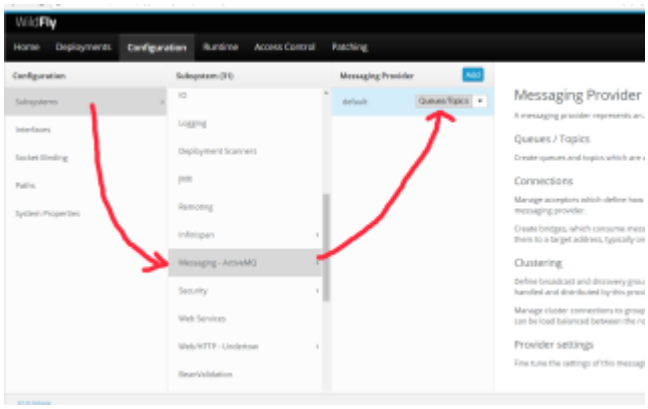
Pont-Pont példa

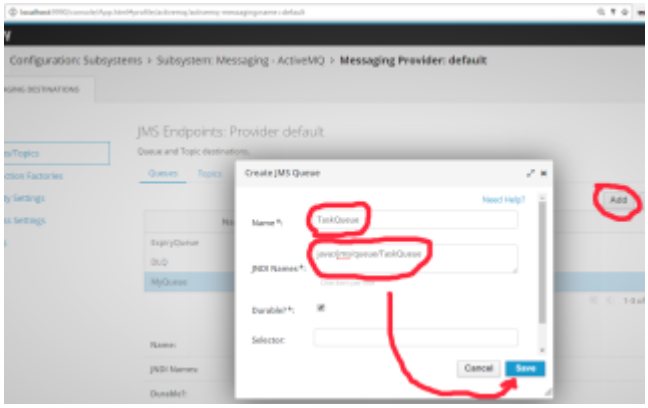
A JMS modul bekapcsolása a Wildfly alkalmazás szerverben. A JMS komponens, a standalone-full.xml konfigurációban szerepel, az alap konfiguráció (standalone.xml) nem tartalmazza. Az Jboss Dev. Studio “Launch configurations”-nál be kell állítani a ”–server-config=standalone-full.xml” és újraindítani a wildfly-t. A pirossal jelzett checkboxot is ki kell kapcsolni.



Az alkalmazásszerver újraindítása után, az admin felületen a Subsystems/Messaging/Queues Topics alatt létrehozhatjuk az üzenetsorokat:



Majd az alábbi kép alapján hozzunk létre egy üzenetsort a megadott néven és JNDI névvel:



Az üzenetsor már a java:/jms/queue/TaskQueue JNDI néven elérhető erőforrás. Hozzunk létre egy projekt-et: dinamikus web vagy ejb projekt is lehet, és az alábbi osztályt adjuk meg:

```
package org.me;
```

```
 import java.util.Date;

 import javax.ejb.ActivationConfigProperty;
 import javax.ejb.MessageDriven;
 import javax.jms.JMSException;
 import javax.jms.Message;
 import javax.jms.MessageListener;
 import javax.jms.TextMessage;

 @MessageDriven(activationConfig = {
        @ActivationConfigProperty(propertyName = "destinationType",
propertyValue = "javax.jms.Queue"),
        @ActivationConfigProperty(propertyName = "destination",
propertyValue = "java:/jms/queue/TaskQueue") })
 public class TaskGenerator implements MessageListener {

    @Override
    public void onMessage(Message message) {
        try {
            if (message instanceof TextMessage) {
                System.out.println("Az üzenet ekkor megérkezett: " + new
Date());
                TextMessage msg = (TextMessage) message;
                System.out.println("Az üzenet: " + msg.getText());
            } else {
                System.out.println("hibás üzenet");
            }
        } catch (JMSException e) {
            e.printStackTrace();
        }

    }

 }
```

Távoli elérés az üzenetsorhoz

Ahhoz, hogy távolról is el lehessen érni egy üzenetsort, a JNDI nevének “java:jboss/exported/” kell kezdődnie. A wildfly admin oldalon ez megtehető, hiszen egy üzenetsor több jndi néven is elérhető. Hozzuk létre újra a TaskQueue sort két JNDI névvel: “java:/jms/queue/TaskQueue” és “java:jboss/exported/TaskQueue”

A MDB-ben az annotációt értelem szerűen változtatni kell:

```
  @ActivationConfigProperty(propertyName = "destination", propertyValue =
"java:/jms/queue/TaskQueue, java:jboss/exported/TaskQueue") })
```

A wildfly /bin/add-user.bat futtatásával létrehozhatunk egy felhasználót quser/Password_1 belépéssel

a guest csoportban:

```
  What type of user do you wish to add?
   a) Management User (mgmt-users.properties)
   b) Application User (application-users.properties)
  (a): b
  Enter the details of the new user to add.
  Using realm 'ApplicationRealm' as discovered from the existing property
files.
  Username : quser
  Password recommendations are listed below. To modify these restrictions
edit the add-user.properties configuration file.
   - The password should be different from the username
   - The password should not be one of the following restricted values
{root, admin, administrator}
   - The password should contain at least 8 characters, 1 alphabetic
character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
  Password :
  Re-enter Password :
  What groups do you want this user to belong to? (Please enter a comma
separated list, or leave blank for none)[  ]: guest
  About to add user 'quser' for realm 'ApplicationRealm'
  Is this correct yes/no? yes
  Added user 'quser' to file
'C:\projects\devstudio\runtimes\wildfly-10.0.0.Final\standalone\configuratio
n\application-users.properties'
  Added user 'quser' to file
'C:\projects\devstudio\runtimes\wildfly-10.0.0.Final\domain\configuration\ap
plication-users.properties'
  Added user 'quser' with groups guest to file
'C:\projects\devstudio\runtimes\wildfly-10.0.0.Final\standalone\configuratio
n\application-roles.properties'
  Added user 'quser' with groups guest to file
'C:\projects\devstudio\runtimes\wildfly-10.0.0.Final\domain\configuration\ap
plication-roles.properties'
  Is this new user going to be used for one AS process to connect to another
AS process?
  e.g. for a slave host controller connecting to the master or for a
Remoting connection for server to server EJB calls.
  yes/no? yes
  To represent the user add the following to the server-identities
definition <secret value="UGFzc3dvcmRfMQ==" />
  Press any key to continue . . .
```

Az MDB klienst most ne a dinamikus web projektben, hanem egy külön java projektben hozzuk létre. A projektre jobb egérgombbal kattintva: Properties/Java Build Path/Add External Jar - segítségével adjuk hozzá a jboss-client.jar-t a [wildfly]\bin\client\ könyvtárból.

A kliens kódja legyen a következő:

```
  import java.util.Hashtable;
```

```
  import javax.jms.Connection;
  import javax.jms.Queue;
  import javax.jms.QueueConnection;
  import javax.jms.QueueConnectionFactory;
  import javax.jms.QueueSender;
  import javax.jms.QueueSession;
  import javax.jms.Session;
  import javax.jms.TextMessage;
  import javax.naming.Context;
  import javax.naming.InitialContext;
  import javax.naming.NamingException;
  public class TestRemoteQueue {
    public static void main(String[] args) throws Exception {
        Connection connection = null;
        InitialContext initialContext = null;
        try {
            initialContext = getInitialContext();
            QueueConnectionFactory qconFactory = (QueueConnectionFactory)
initialContext
                    .lookup("jms/RemoteConnectionFactory");
            QueueConnection qcon =
qconFactory.createQueueConnection("quser", "Password_1");
            QueueSession qsession = qcon.createQueueSession(false,
Session.AUTO_ACKNOWLEDGE);
            Queue queue = (Queue) initialContext.lookup("TaskQueue");
            QueueSender qsender = qsession.createSender(queue);
            qcon.start();
            TextMessage msg = qsession.createTextMessage();
            msg.setText("Hello world");
            qsender.send(msg);
            qsender.close();
            qsession.close();
            qcon.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    private static InitialContext getInitialContext() throws NamingException
{
        Hashtable env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY,
"org.jboss.naming.remote.client.InitialContextFactory");
        env.put(Context.PROVIDER_URL, "http-remoting://127.0.0.1:8080");
        env.put("jboss.naming.client.ejb.context", false);
        env.put(Context.URL_PKG_PREFIXES, "org.jboss.ejb.client.naming");
        return new InitialContext(env);
    }
  }
```

Indítás után a szerver consol-ban megjelenik a “Hello world” szöveg.

Tranzakció visszavonás

A következő MDB megállapítja, hogy újra lett-e küldve az üzenet és ha igen, akkor visszavonja a tranzakciót. A visszavont tranzakció miatt az üzenetsor újra megpróbálja elküldeni az üzenetet, mivel ez már egyszer el lett küldve, ezért fogadjuk. A getJMSRedelivered() true értéket fog adni.

```
@MessageDriven(name = "RollbackOnly", activationConfig = {
        @ActivationConfigProperty(propertyName = "destinationType",
propertyValue = "javax.jms.Queue"),
        @ActivationConfigProperty(propertyName = "destination",
propertyValue = "queue/RemoteQueue") })
@TransactionManagement(value = TransactionManagementType.CONTAINER)
@TransactionAttribute(value = TransactionAttributeType.REQUIRED)

public class RollbackOnly implements MessageListener {
@Resource
MessageDrivenContext ctx;

public void onMessage(final Message message) {
    try {
        TextMessage textMessage = (TextMessage) message;

        String text = textMessage.getText();

        if (!textMessage.getJMSRedelivered()) {
            System.out.println("message " + text
                    + " received for the first time");
            ctx.setRollbackOnly();
        } else {
            System.out.println("message " + text
                    + " received for the second time");
        }

    } catch (JMSException e) {
        e.printStackTrace();
    }
}
}
```

A hozzá tartozó kliens kódja ugyanaz is lehet mint az előző példában.

Halott levél csatorna (DEAD Letter QUEUE)

Állítsuk be az előző példánál, hogy csak 1x próbálja újra az üzenet kézbesítést: [jbdevstudio]\runtimes\jboss-eap\standalone\configuration\standalone-full.xml -ban keressük meg a következő részt:

```
<address-settings>
  <address-setting match="#">
```

És illesszük be a következő sort, majd indítsuk újra a JBoss szervert.

```
<max-delivery-attempts>1</max-delivery-attempts>
```

Ilyenkor az üzenet a dead letter queue-ra kerül 1 próbálkozás után.

A Dead Letter Queue-ról a következő osztály leveszi az üzeneteket:

```
package org.ait;

import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.TextMessage;

@MessageDriven(name = "DeadLetter", activationConfig = {
        @ActivationConfigProperty(propertyName = "destinationType",
propertyValue = "javax.jms.Queue"),
        @ActivationConfigProperty(propertyName = "destination",
propertyValue = "queue/DLQ") })
public class DeadLetterQueue implements MessageListener {
    public void onMessage(final Message message) {
        try {
            TextMessage textMessage = (TextMessage) message;
            String text = textMessage.getText();
            System.out.println("DEAD Letter QUEUE got a message:" + text);
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```