

# TP 4 des architectures logicielles

## Séance 4 : Architecture n-tiers distribuée à base de JMS et Message Driven Bean

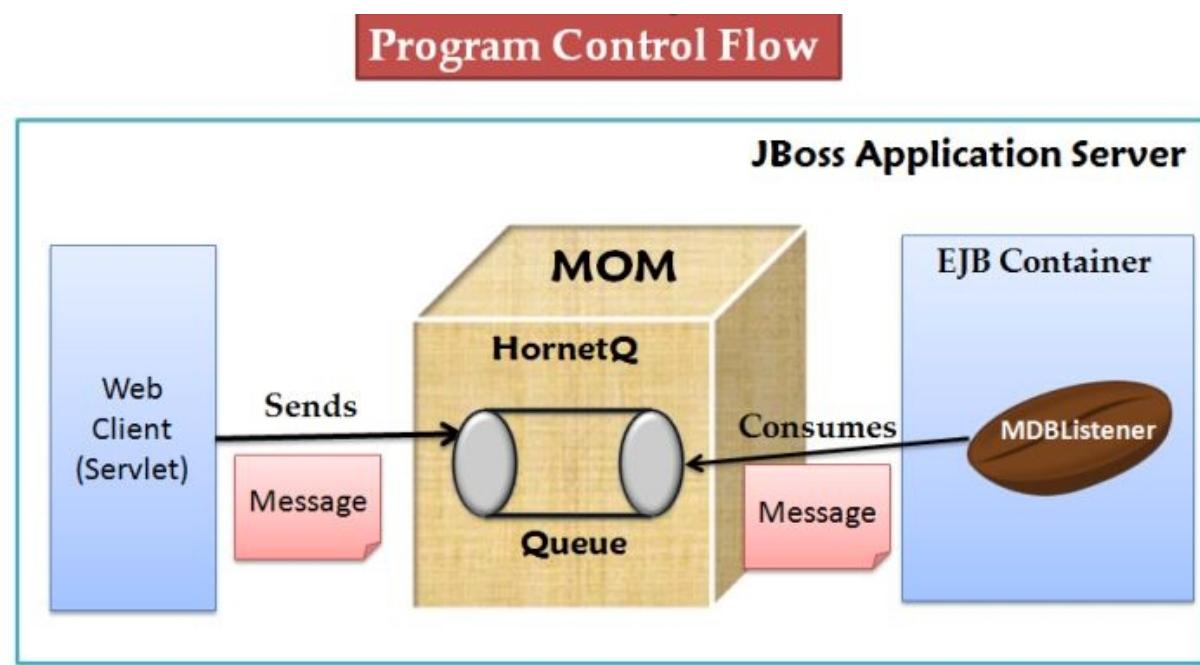
### 1 Préparation de l'environnement Eclipse

#### 1. Environment Used

- JDK 7 (Java SE 7)
- **JMS Sender/Client** – Servlet API
- **JMS Consumer** – EJB 3.0 Message Driven Bean (MDB)
- Eclipse
- JBoss Tools – Core 4.4.1 M5 for Eclipse
- JBoss Application Server (AS) 7.1.0 Final

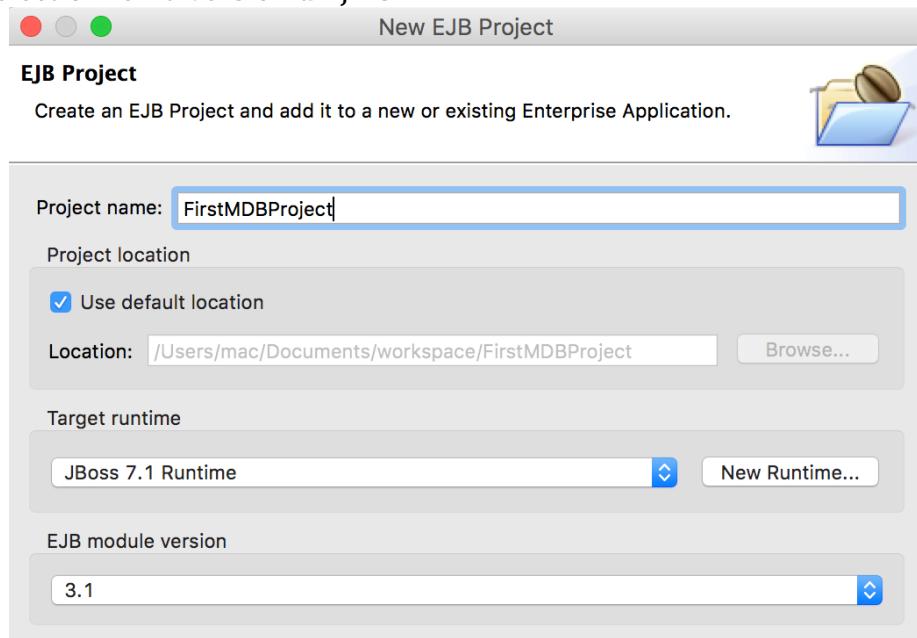
#### 2. Description du projet

- Nous allons créer un Message Driven Bean EJB 3 avec une application Client Web (Servlet) permettant d'envoyer des messages vers une destination de type Queue.
- Ce TP explique comment développer, déployer et exécuter un EJB3 MDB sous forme d'un « message consumer » au niveau du serveur d'application JBoss.
- Pour tester le MDB Listener nous implémentant une application Client Web (Servlet) sous forme d'un « message producer » qui envoie un texte simple ou bien un objet.
- Les deux projets, à savoir le message driven bean (message consumer) et la Servlet (message producer) seront déployés sur la même instance du serveur d'application (JBoss AS).

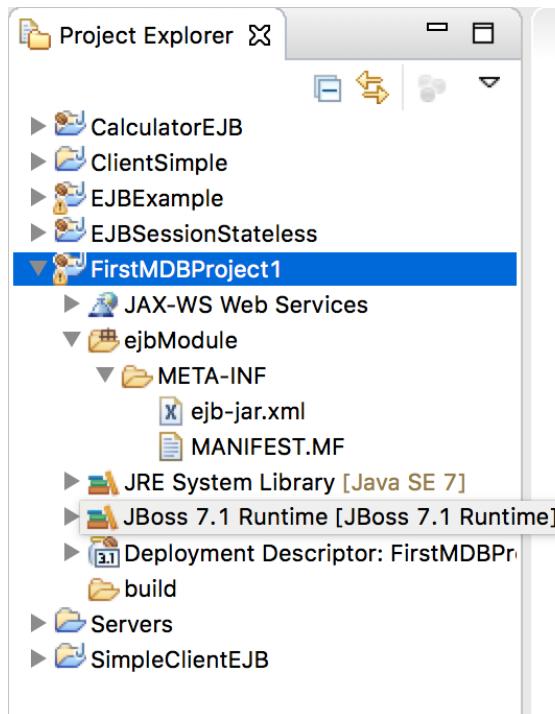


## 2 Création du Message Consumer « MDB »

- a) Céer un nouveau projet EJB « FirstMDBProject »
- b) Spécifier le target run time
- c) Sélectionner la version d'EJB 3.1



- d) A la fin vous allez voir le structure suivante :



Créer un Object Message class :

- a. Java package **ma.ac.bean**
- b. Class name **Employee**

```
package ma.ac.bean;
import java.io.Serializable;
```

```

public class Employee implements Serializable {
    private int id;
    private String name;
    private String designation;
    private double salary;

    public Employee() { }

    public Employee(int id, String name, String designation,
                   double salary)
    {
        this.id = id;
        this.name = name;
        this.designation = designation;
        this.salary = salary;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDesignation() {
        return designation;
    }

    public void setDesignation(String designation) {
        this.designation = designation;
    }

    public double getSalary() {
        return salary;
    }

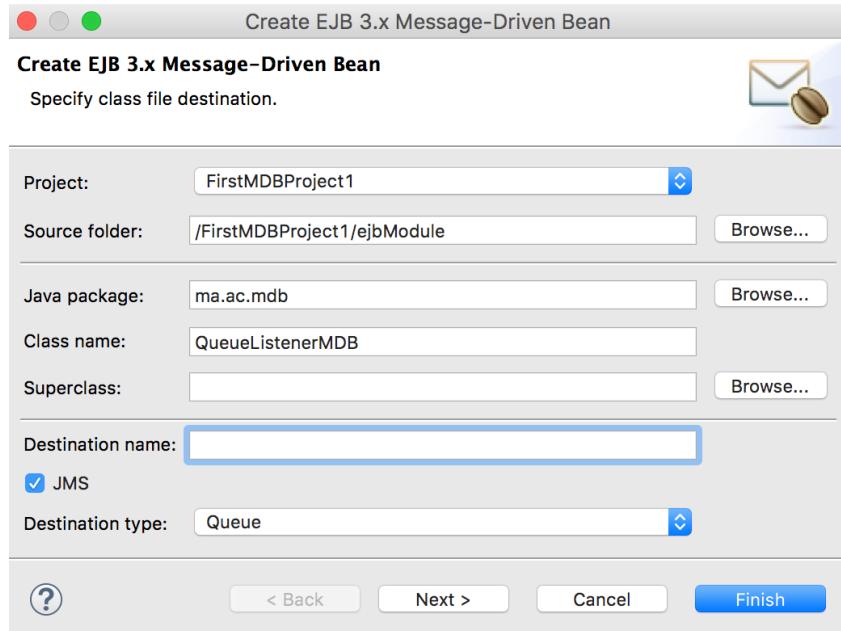
    public void setSalary(double salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ",
designat=on="
                           + designation + ", salary=" + salary + "]";
    }
}

```

- e) Créer le Message Driven Bean Consumer

- Java package name **ma.ac.mdb**
- Class name **QueueListenerMDB**
- Destination type **Queue**



- Planter le code suivant :

```

package ma.ac.mdb;

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.ObjectMessage;
import javax.jms.TextMessage;
import ma.ac.bean.Employee;

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

    public void onMessage(Message message) {
        try {
            if (message instanceof TextMessage) {
                System.out.println("Queue: I received a TextMessage at "
                    + new Date());
                TextMessage msg = (TextMessage) message;
                System.out.println("Message is : " + msg.getText());
            } else if (message instanceof ObjectMessage) {
                System.out.println("Queue: I received an ObjectMessage
at ")
            }
        }
    }
}

```

```

        + new Date());
ObjectMessage msg = (ObjectMessage) message;
Employee employee = (Employee) msg.getObject();
System.out.println("Employee Details: ");
System.out.println(employee.getId());
System.out.println(employee.getName());
System.out.println(employee.getDesignation());
System.out.println(employee.getSalary());
} else {
    System.out.println("Not a valid message for this Queue
MDB");
}
}

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

```

- The activationConfig property of @MessageDriven is an array of ActivationConfigProperty and it should specify the destinationType (Queue or Topic) and destination (Queue/Topic's JNDI name).
- In the onMessage() we are receiving two types of message, TextMessage and ObjectMessage.

#### f) Configurer le messaging services on JBoss AS 7

Dans **JBoss AS 7**, le « message service » est configuré à partir du fichier de configuration xml (standalone.xml) JBoss/Home/standalone/configuration.

a. Dans l‘élément <extensions>, ajouter entension element :

```
<extension module="org.jboss.as.messaging"/>
```

b. Dans l‘élément <profile>, ajouter l‘élément <subsystem> :

```
<subsystem xmlns="urn:jboss:domain:messaging:1.1">
    <hornetq-server>
        <persistence-enabled>true</persistence-enabled>
        <journal-file-size>102400</journal-file-size>
        <journal-min-files>2</journal-min-files>

        <connectors>
            <netty-connector name="netty" socket-
binding="messaging"/>
            <netty-connector name="netty-throughput" socket-
binding="messaging-throughput">
                <param key="batch-delay" value="50"/>
            </netty-connector>
            <in-vm-connector name="in-vm" server-id="0"/>
        </connectors>

        <acceptors>
            <netty-acceptor name="netty" socket-
binding="messaging"/>
            <netty-acceptor name="netty-throughput" socket-
binding="messaging-throughput">
                <param key="batch-delay" value="50"/>
                <param key="direct-deliver" value="false"/>
            </netty-acceptor>
            <in-vm-acceptor name="in-vm" server-id="0"/>
        </acceptors>
    </hornetq-server>
</subsystem>
```

```

<security-settings>
    <security-setting match="#">
        <permission type="send" roles="guest"/>
        <permission type="consume" roles="guest"/>
        <permission type="createNonDurableQueue"
roles="guest"/>
        <permission type="deleteNonDurableQueue"
roles="guest"/>
    </security-setting>
</security-settings>

<address-settings>
    <address-setting match="#">
        <dead-letter-address>jms.queue.DLQ</dead-letter-
address>
        <expiry-address>jms.queue.ExpiryQueue</expiry-
address>
        <redelivery-delay>0</redelivery-delay>
        <max-size-bytes>10485760</max-size-bytes>
        <address-full-policy>BLOCK</address-full-policy>
        <message-counter-history-day-limit>10</message-
counter-history-day-limit>
    </address-setting>
</address-settings>
<jms-connection-factories>
    <connection-factory name="InVmConnectionFactory">
        <connectors>
            <connector-ref connector-name="in-vm"/>
        </connectors>
        <entries>
            <entry name="java:/ConnectionFactory"/>
        </entries>
    </connection-factory>
    <connection-factory name="RemoteConnectionFactory">
        <connectors>
            <connector-ref connector-name="netty"/>
        </connectors>
        <entries>
            <entry name="RemoteConnectionFactory"/>
            <entry
name="java:jboss/exported/jms/RemoteConnectionFactory"/>
        </entries>
    </connection-factory>
    <pooled-connection-factory name="hornetq-ra">
        <transaction mode="xa"/>
        <connectors>
            <connector-ref connector-name="in-vm"/>
        </connectors>
        <entries>
            <entry name="java:/JmsXA"/>
        </entries>
    </pooled-connection-factory>
</jms-connection-factories>

<jms-destinations>
    <jms-queue name="testQueue">
        <entry name="queue/MyQueue"/>
    </jms-queue>
    <jms-topic name="testTopic">
        <entry name="topic/MyTopic"/>
    </jms-topic>
</jms-destinations>
</hornetq-server>

```

```
</subsystem>
```

The entry name for jms-queue should match the destination activation config property in @MessageDriven.

g) Ajouter le « messaging port » dans l’élément **<socket-binding-group>**

```
<socket-binding name="messaging" port="5445"/>
<socket-binding name="messaging-throughput" port="5455"/>
```

h) Ajouter le MDB Resource Adapter Reference dans l’élément **<subsystem xmlns="urn:jboss:domain:ejb3:1.2">**

```
<mdb>
  <resource-adapter-ref resource-adapter-name="hornetq-ra"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

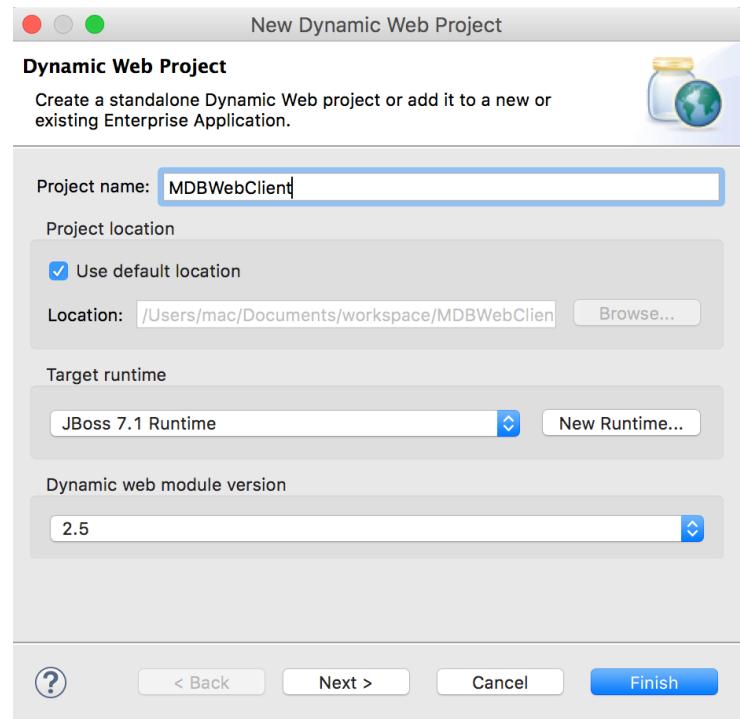
Redémarrer le serveur JBoss et déployer ton projet EJB, Si le projet est déployé correctement avec un global JNDI mapping, alors vous allez voir les informations suivantes sur le console :

- 13:21:50,174 INFO [org.jboss.as.ejb3] (MSC service thread 1-3) JBAS014142: Started message driven bean 'QueueListenerMDB' with 'hornetq-ra' resource adapter
- 13:21:50,244 INFO [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS018559: Deployed "FirstMDBProject1.jar"

### 3. Création d'un client JMS (Dynamic Web Project)

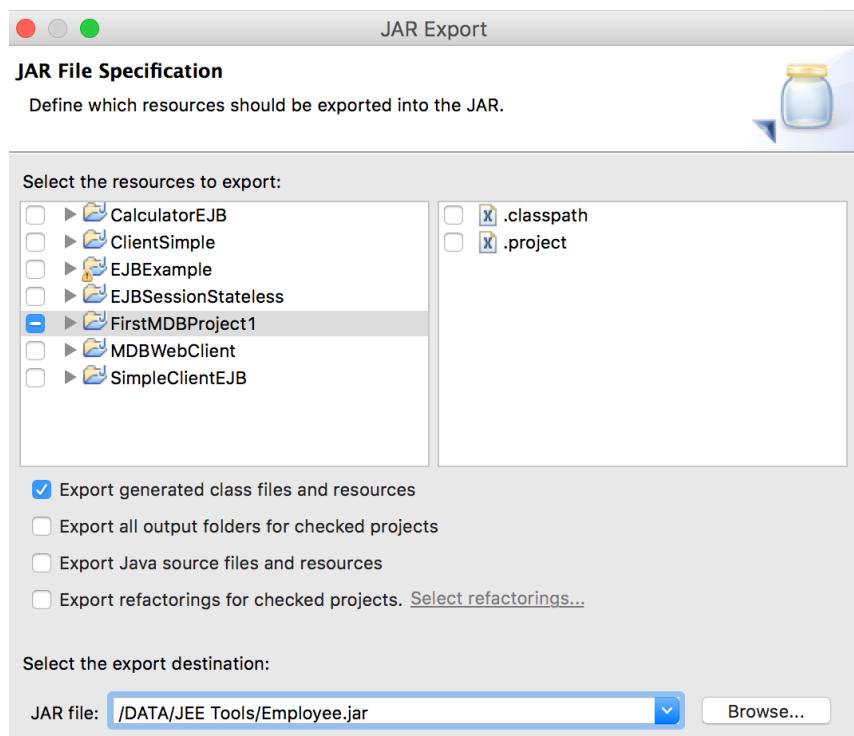
A ce niveau nous allons créer une application web (Servlet) qui va produire et envoyer des messages à MyQueue. Le servlet va utiliser le JNDI pour faire un LookUp sur le ConnectionFactory et la Queue.

- a) Créer un nouveau projet ‘Dynamic Web project’ :
  - a. Project name “MDBWebClient”
  - b. Sélectionner JBoss 7.1 Runtime
  - c. Dynamic web module version 2.5

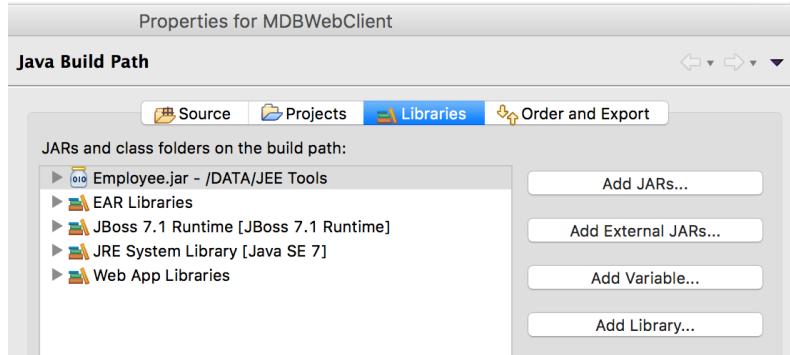


b) Création de jar de la class Employee.class

- Exporter le Employee.java sous la forme d'un .jar et enregistrer le dans un emplacement déterminé.



- Ajouter le jar dans le lib du projet web



c) Créer une Servlet

- a. Java package name **ma.ac.servlets**
- b. Class name **ServletMessageProducer**

```

package ma.ac.servlets;

import java.io.IOException;
import java.io.PrintWriter;
import javax.jms.ObjectMessage;
import javax.jms.Queue;
import javax.jms.QueueConnection;
import javax.jms.QueueConnectionFactory;
import javax.jms.QueueSender;
import javax.jms.QueueSession;
import javax.jms.TextMessage;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import ma.ac.bean.*;

@WebServlet("/ServletMessageProducer")
public class ServletMessageProducer extends HttpServlet {

    /**
     *
     */
    private static final long serialVersionUID = 5467957317519593989L;

    public ServletMessageProducer() {
        super();
    }

    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
            throws ServletException, IOException {
        final String QUEUE_LOOKUP = "queue/MyQueue";
        final String CONNECTION_FACTORY = "ConnectionFactory";

        PrintWriter out = response.getWriter();
        try{
            Context context = new InitialContext();

```

```

QueueConnectionFactory factory =
(QueueConnectionFactory)context.lookup(CONNECTION_FACTORY);
QueueConnection connection = factory.createQueueConnection();
QueueSession session =
connection.createQueueSession(false,
QueueSession.AUTO_ACKNOWLEDGE);

Queue queue = (Queue)context.lookup(QUEUE_LOOKUP);
QueueSender sender = session.createSender(queue);

//1. Sending TextMessage to the Queue
TextMessage message = session.createTextMessage();
message.setText("Hello EJB3 MDB Queue!!!");
sender.send(message);
out.println("1. Sent TextMessage to the Queue");

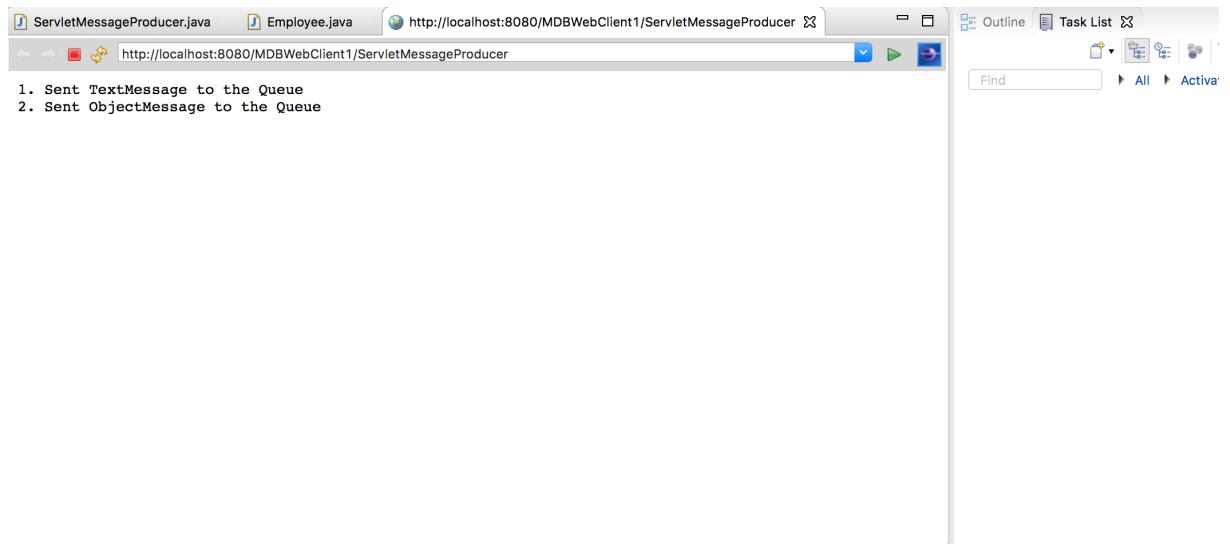
//2. Sending ObjectMessage to the Queue
ObjectMessage objMsg = session.createObjectMessage();

Employee employee = new Employee();
employee.setId(2163);
employee.setName("Imade");
employee.setDesignation("Prof");
employee.setSalary(100000);
objMsg.setObject(employee);
sender.send(objMsg);
out.println("2. Sent ObjectMessage to the Queue");

session.close();
}catch(Exception e){e.printStackTrace();}
}
}

```

- 
- d) Redémarrer le serveur JBoss et déployer ton projet Web, si le projet est déployé correctement avec un JNDI mapping, alors vous allez voir les informations suivantes sur la console :



```
JBoss AS 7.1 [JBoss Application Server Startup Configuration] /Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home/bin/java (26 oct. 2016 à 15:43:25)
15:43:30,413 INFO  [org.jboss.as] (MSC service thread 1-2) JBAS015874: JBoss AS 7.1.1.Final "Brontes" started in 4552ms - Started 335 of 420 services (15:43:30,470 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS018559: Deployed "MDBWebClient1.war")
15:43:30,471 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS018559: Deployed "FirstMDBProject1.jar"
15:43:30,472 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS018559: Deployed "EJBSessionStateless.jar"
15:43:30,473 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS018559: Deployed "CalculatorEJB.jar"
15:43:40,867 INFO  [stdout] (Thread-2 (HornetQ-client-global-threads-1952959149)) Queue: I received an ObjectMessage at Wed Oct 26 15:43:40 WEST 2016
15:43:40,869 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1952959149)) Queue: I received a TextMessage at Wed Oct 26 15:43:40 WEST 2016
15:43:40,870 INFO  [stdout] (Thread-1 (HornetQ-client-global-threads-1952959149)) Message is : Hello EJB3 MDB Queue!!!
15:43:40,872 INFO  [stdout] (Thread-2 (HornetQ-client-global-threads-1952959149)) Employee Details:
15:43:40,872 INFO  [stdout] (Thread-2 (HornetQ-client-global-threads-1952959149)) 2163
15:43:40,872 INFO  [stdout] (Thread-2 (HornetQ-client-global-threads-1952959149)) Imade
15:43:40,873 INFO  [stdout] (Thread-2 (HornetQ-client-global-threads-1952959149)) Prof
15:43:40,873 INFO  [stdout] (Thread-2 (HornetQ-client-global-threads-1952959149)) Prof
```

## 4. Reprendre les mêmes étapes pour développer un MDB de type « Topic »