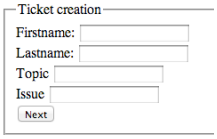
- TD 3 -
The Beans send and receive messages

The application that we have developed allows us to save tickets submitted by customers. We should now adapt the application to enable the transmission of (some) tickets not only to the local HelpDesk but also to one (or more) remote HelpDesk(s). We will also develop a new application that allows the outsourced HelpDesks to handle the tickets they receive. HelpDesk technicians do not work 24h/24h and therefore we should propose a mechanism for the transmission/reception of tickets outside service hours.

1. Resource management

The application uses messages (JMS) to deliver the tickets - we should thus declare the corresponding (JMS) resources using the GlassFish administration console. We could first create a `Connection Factory` of type `javax.jms.QueueConnectionFactory` (if we don't use the default one in JMS 2) that we can call `jms/TicketQueueConnectionFactory` and then we should create a destination of type `javas.jms.Queue` called `TicketRequestQueue`.

2. A simple solution



To simplify the development we assume that all the information needed to create a ticket sent to a relocated HelpDesk are independent and therefore can be specified simultaneously by using, for example, an interface of the form:
As HelpDesk technicians do not work 24h/24h we have to adapt our application to send a message to a message queue; this queue is accessed by the application HelpDesk.

First, the application HelpDesk retrieves the tickets sent in the queue and displays them in a console (which will be read by technicians). Eventually, the tickets will be stored in a DB and displayed (on demand) by the (web interface of the) application HelpDesk. Do not forget to include any libraries needed by the application.

3. Several HelpDesk centers

The tickets concern different areas of expertise (IT, equipment management, etc...) and therefore they are potentially treated by different centers. We should modify the application to enable a processing of tickets according to their characteristics (e.g. their topic) - each HelpDesk will have the opportunity to retrieve only the tickets directly concerning it. Develop and deploy several applications HelpDesk dealing with different types of tickets. Be careful, each (MDB) should see another (client)Id.

There may be some relationships between tickets (for example, a hardware problem reported in a ticket is the cause of software problems reported in another ticket) and therefore it would be interesting to have all tickets in all centers even if some of them are not treated in each center. Modify your application to distribute tickets to all centers - each center displays all tickets but handles only tickets corresponding to their skills. You can provide a mechanism for filtering messages read/processed by the centers.

4. Confirm the treatment of tickets

It will now provide a mechanism for notifying customers about the reception of their tickets (and then about their treatment). Once a ticket is received by a HelpDesk, a confirmation message is sent to the client - since the ticket submission application is not available permanently, the confirmations should be done asynchronously (like the submission of tickets). You can use a solution based on temporary queues (use `JMSReplyTo`) and/or based on (filtered) `Topic`s.

Eventually, we should of course update the tickets (their state and possibly the corresponding solution) based on the responses received.