The report of Micro project Mailbox Management

1. Create the server

1.1. Create the entity classes

There are 4 basic entity classes : **Message**, **Box**, **FinalUser** and **NewsGroupRight**. And the other two classes **Mailbox** and **NewsBox** are created by inheriting the class **Box.**

For every class, we list their attributes below (the attribute which has a bold header is the primary key for each class):

Message (**messageId**, senderName, receiverName, sendingDate, subject, body, alreadyRead, box)

Box (**boxId**, type, boxName, finalUser, messages)

FinalUser (**userId**, username, password, newsGroupRight, mailbox)

1.2. Create the relation between the classes

We can annotate the phrase below in every basic class to create the relation between them:

Message and **Box**

Message: @ManyToOne(cascade=ALL)
@JoinColumn(name=”boxId”)

Box: @OneToMany(cascade = ALL, mappedBy = "box")

Box and **FinalUser**

Box: @OneToOne()
@JoinColumn(name = "userId")

FinalUser: @OneToOne
@JoinColumn(name = "boxId")

FinalUser and **NewsGroupRight**

FinalUser: @OneToOne(cascade = ALL, mappedBy = "finalUser")

NewsGroupRight: @OneToOne(cascade=ALL)
@JoinColumn(name = "userId")

1.3. Create the basic ejb classes

We need to create two interface **IMailBoxManager** and **IUserDirectory** which represent the **remote business interface**. In order to implement these two interfaces, we create the two corresponding **stateless session bean** class **MailBoxManagerBean** and **UserDirectory.**

To implement the methods in the interface, we can use simply the method of the **EntityManager** object by call the function **entityManager.createQuery(query)** to retrieve the results from our sql exections and put them into our **entity objects**.

For example, you can get the **user** entity by retrieve the result of query in this way below:

```
public FinalUser getUserByName(String userName) {
    try {
        Query q = em.createQuery("select user from FinalUser
user where user.userName =:userName", FinalUser.class);
        q.setParameter("userName", userName);
        FinalUser user = (FinalUser) q.getSingleResult();
        return user;
    }
    catch (NoResultException e) {
        return null;
    }
}
```

We list all the method for each **remote business interface** below:

```
@Remote
public interface IUserDirectory {
    public boolean addUser(String userName, String password);
    public boolean removeUser(String userName);
    public Collection<FinalUser> lookupAllUsers();
    public NewsGroupRight lookupAUserRights(String userName);
    public FinalUser getUserByName(String userName);
```

```
    public boolean updateAUserRights(String userName, boolean
readRight, boolean writeRight);
    public boolean checkPassword(String userName, String
password);
}

@Remote
public interface IMailBoxManager {
    public Collection<Message> readAUserNewMessages(String
userName);
    public Collection<Message> readAUserAllMessages(String
receiverName);
    public boolean deleteAUserReadMessages(String userName);
    public boolean sendAMessageToABox(String subject, String
body, String receiverName, String senderName);
    public Message getMessageById(int messageId);
    public Box getBoxByUserId(int userId);
}
```

2. Create the client

We create the client interface by calling back the method above to test our application including the method below:

addUser
removeUser
lookUserRight
updateUserRight
sendMessage
readNewMessage
readAllMessage
deleteReadMessages
listAllUsers

3. Problems and bugs

The operation of **addUser**, **removeUser**, **LookUserRight**, **updateUserRight**, **listAllUsers** are working really well but the operation of **sendMessage**, **readAllMessage** and **deleteReadMessages** are not working at the moment.