Design Document

My Hotel Booking System

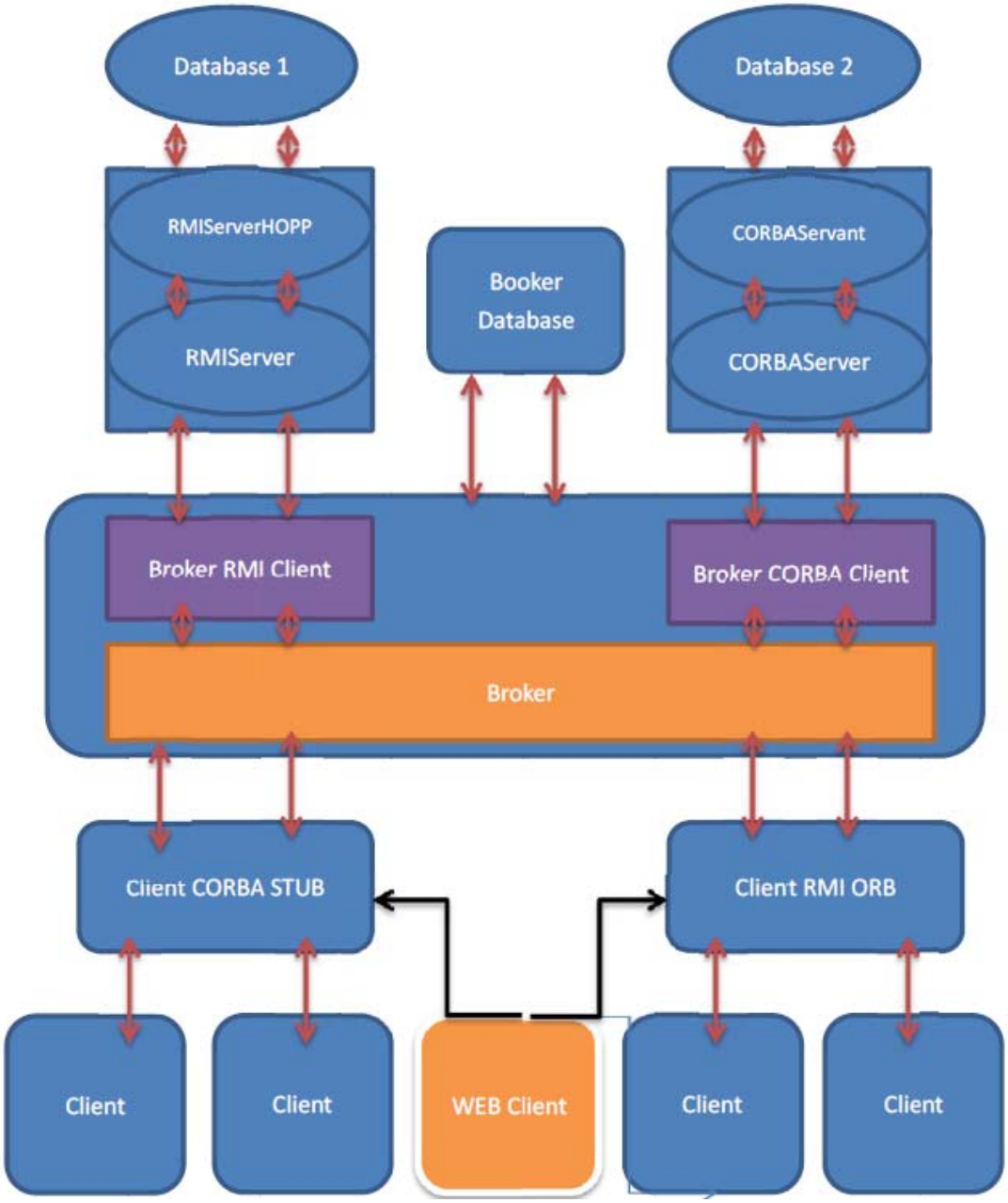
2015/5/19

Yu Jun 26346702

Design Document for My Hotel Booking System

Content

1. The Model of the Hotel System



Comments: you can see from this above model, and this system has two Databases and corresponding two Servers: RMI Server, CORBA Server. Additionally, we have a booker database to store information of the bookers. The Servers can get the records of the Database by invoking the RMIServerHopp or CORBAServant Object. For the RMI Server, it will send marshaled results or exception to the broker. Similarly, the broker will then in the same way send the marshaled results or exception to the client. For the CORBA Server, it will get the preprocessed information from the CORBA Servant, and then using the ORB (Object Request Broker) it will transfer the data, looks after marshalling parameters, make the remote call and return the results. On the

Broker side, in the same way the broker will get then get the information from the Server. In my project, my broker can both handle the CORBA and RMI requests by changing parameters of client between RMI and CORBA. And this project also supports multiple clients to communicate with one broker and server, including web client. And the information between server and broker, also between broker and client is encrypted using RSA algorithm.

2. The deployment of the keytool for Security issues:

1) I have already generated three keystore files: server.keystore, client.keystore, and rsa.keystore. Both server.keystore and client.keystore are generated with DSA algorithm. While the rsa.keystore file is generated using the RSA algorithm. The steps of generating these files are as following, I would not go into detail for this.

```
hugh@hugh-virtual-machine:~/Javahwk/src2$ keytool -genkey -keystore server.keyst
ore -keyalg DSA -alias mykey -keypass yujun123
```

2) If you want to check the basic information of these files, you should use the command as following (the password is 'yujun123'):

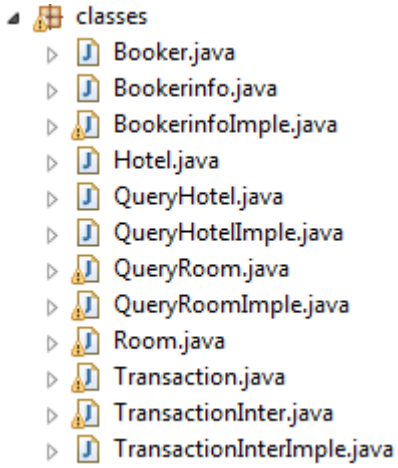
```
hugh@hugh-virtual-machine:~/Javahwk/src2$ keytool -list -v -keystore ./server.ke
ystore
输入密钥库口令:
```

3) You will see the information of server.keystore like this:

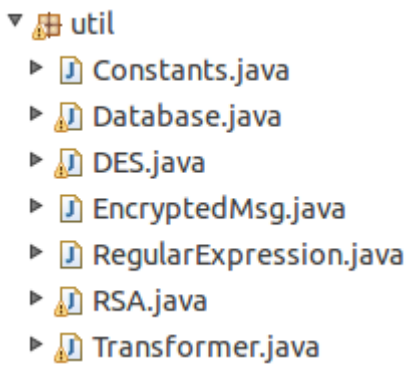
```
密钥库类型: JKS
密钥库提供方: SUN

您的密钥库包含 1 个条目

别名: mykey
创建日期: 2015-5-20
条目类型: PrivateKeyEntry
证书链长度: 1
证书[1]:
所有者: CN=Jun Yu, OU=FIT, O=Monash University, L=Suzhou, ST=Jiangsu, C=CN
发布者: CN=Jun Yu, OU=FIT, O=Monash University, L=Suzhou, ST=Jiangsu, C=CN
序列号: 6846d16d
有效期开始日期: Wed May 20 15:18:33 CST 2015, 截止日期: Tue Aug 18 15:18:33 CST
2015
证书指纹:
         MD5: FE:30:E5:83:79:97:03:18:83:D0:B9:04:50:7E:82:C2
         SHA1: 28:2E:96:5B:D5:16:0E:57:1A:1D:85:60:75:1D:C7:1A:2A:F1:75:16
         SHA256: 16:D4:E3:0A:2E:C9:B9:B7:05:5C:95:F5:C6:40:CE:67:16:D7:3F:09:0A:
88:A4:FA:EA:8F:E1:DA:EF:29:0E:A6
         签名算法名称: SHA1withDSA
         版本: 3

扩展:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 0A 77 99 9E B0 9E 97 D9   12 3F D8 2C 68 5B B2 21  .w.......?.,h[.!
0010: 82 C7 10 F1                                        ....
]
]



*******************************************
*******************************************
```

You can see from the picture that this file contains the basic information of this file, like CN, OU, and the algorithm I have used. You can also check other two files using this command.

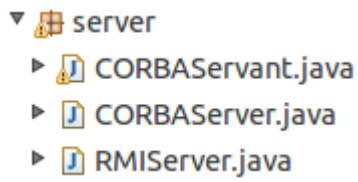
3. The main Class of my Java Program:



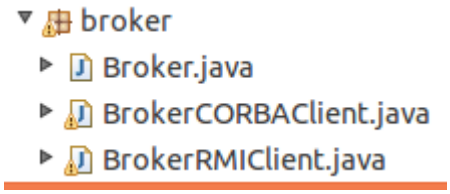
1) Hotel, Room and Transaction are used to define the properties of the hotel, room and transaction and are also used to maintain the information from Database. What's more, QueryRoom, QueryHotel, TransactionInter, Bookerinfo are three interfaces to define the functions I may use later. These three interfaces are implemented by QueryRoomImple, QueryHotelImple, TransactionInterImple and Bookerinfoimple respectively. These three classes are used to query or update the tables in the database.



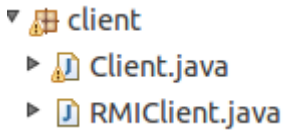
2) In this package, I define the Constants Class to store the basic constants I would use very often. Database Class is used to initialize the records in the Database. RegularExpression Class is used to normalize the data the client inputs in. DES class is used to encrypt the data transferred from the server to broker, as does the RSA class. Transformer class is used to transformer the format of the data, like change bytes to objects, string to objects or bytes.



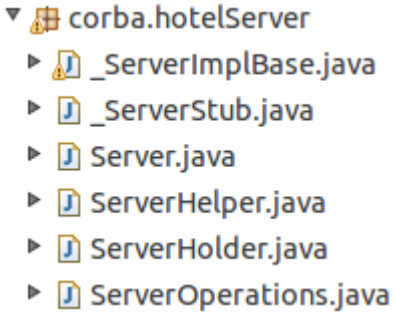
3) In this package, I have written two servers to support the system. One is CORBA Server, and RMI Server. In order to successfully launch the CORBA Server, the Servant class is also needed to provide the basic functions to obtain the data from the database, and also encrypted the data by invoking the DES class in packet util. In the CORBA Server, I have also written several lines of codes to simplify the deployment of this project. For this, I don't need to open another command window to start up the naming service with port number 1235 similar to the slides of week 8. For the RMI Server, I have created a registry with port number 1099, using LocateRegistry, and also created a stub of type RMIInterface.



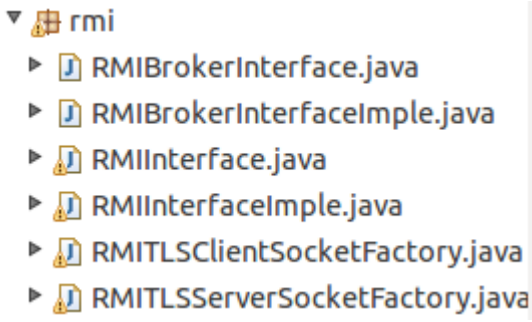
4) In this package, I have written a RMI type Broker to be communicated with the client with port number 1100, and all the information from client to broker is also encrypted in order to maintain the security of this project, and also can decrypt the information from server to broker for showing the integrity of the information. The BrokerCORBAClient class and BrokerRMIClient class are used to handle the information from CORBA Server and RMI Server.



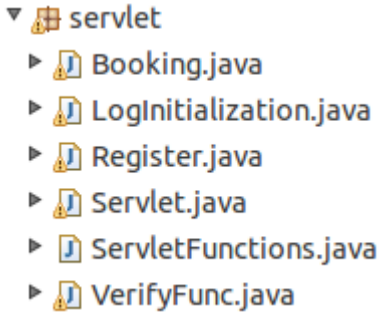
5) In this packet, the RMIClient class is used to process the information from the broker side. And the client is a text – based client for the users to communicate with the broker side.



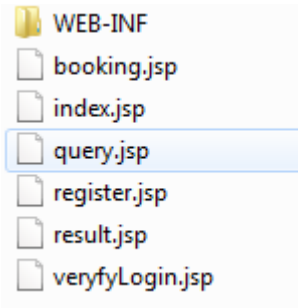
6) This CORBA packet is generated by the HotelServant.idl using the command: idlj –f all –oldImplBase HotelServant.idl. I will not go into detail for this.



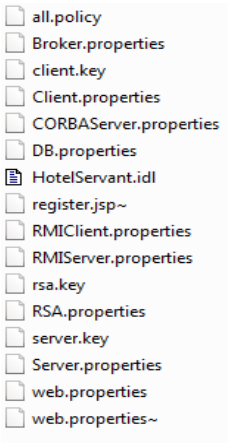
7) In this packet, I have defined several interfaces and its implementations to be used by both server and broker. The RMIBrokerInterface and its implementations are used by the broker to communicate with the server, similarly the RMIInterface and its implementation are used by the Server, the rest two classes is used to make the information encrypted. In the book of Java core java volume 2, you will see how it works.



8) For this packet, I generally use these classes along with 6 jsp files to display the information on the browser, Booking is used to book the room, and verifyFunc is used to verify the information of the booker, register class is used to register an account in this hotel system.



9) In this folder, you can see that I have written six jsp files. In these files, I tend to work with the servlets which I have introduced before, so that we can access the hotel booking system on the browser. In the WEB-INF folder, I have already put created classes folder and put the class files of this project into Classes folder. In the Classes folder, you will also see these following files which are the deployment files which I use to generate the log files. And the ras.key, server.key and client.key files are used to encrypt and decrypt the information.



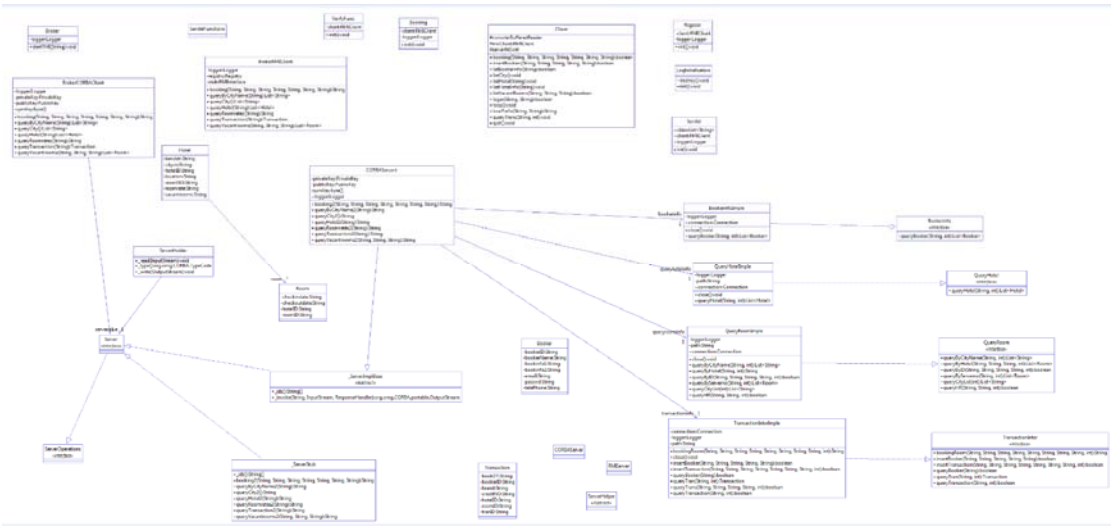
4. The message Format for transmission

Between the RMIServer and BrokerRMIClient, also between CORBAServer and BrokerCORBAClient, these two Class as processes when running communicate with each other using RMI or CORBA respectively. Since you already have known the concept of RMI or CORBA and its implementation behind these two techniques, I will instead introduce the basic functions both the broker and server use for the communication.

List<String> queryCity (int serverNO); Parameters: **serverNO** (1 for RMI Server, 2 CORBA Server) Return: The cities supported by this Hotel Booking System
List<String> queryByCityName (String cityname, int serverNO); Parameters: **serverNO** (1 for RMI Server, 2 CORBA Server) **cityname (input the city name like Suzhou, Nanjing)** Return: Cities supported by this Hotel Booking System
String booking (String checkinDate, String checkoutDate, String hotelID, String roomID, String bookerID, String creditNO, String brand, int serverno) Parameters: **serverNO** (1 for RMI Server, 2 CORBA Server) **checkinDate** (the day of enter the hotel, and the format is '2015-05-19') **checkoutDate** (the day of leaving the hotel, and the format is '2015-05-19') **hotelID** (The number of you hotel) **roomID** (the room you want to book) **creditNO** (the credit card number) **serverNO** (1 for RMI Server, 2 CORBA Server) Return: **Booking conditions** ("BOOKSUCCESSFUL" for success, "FAILED" for the failure of the booking,

"DUPLICATEDBOOKING" means you have already booked this room)
String queryRoomrates (String hotelID, int serverNO); Parameters: **serverNO** (1 for RMI Server, 2 CORBA Server) **hotelID** (The number of you hotel) Return: The room rate of this hotel (String)
List<Room> queryVacantrooms (String hotelID, String checkindate, String checkoutdate, int serverNO); Parameters: **serverNO** (1 for RMI Server, 2 CORBA Server) **checkinDate** (the day of enter the hotel, and the format is '2015-05-19') **checkoutDate** (the day of leaving the hotel, and the format is '2015-05-19') Return: The qualified rooms supported by this conditions which you would then book. The type is a list of class Room

5. The UML diagram



Description: if you can see this UML diagram clearly, you can open the picture at this directory. This UML diagram shows the relationship between the parent class and the extended classes. The Booker, Hotel, Transaction and the Room are the Classes needed to describe these particular objects and also store the information from the Database, and then Bookerinfo, QueryHotel, QueryRoom and TransactionInter are the interfaces we would invoke later, in the CORBAServant and RMIInterfaceImple Class, this class would invoke the different functions to get the information from the database, and then pass the information to the Server, the Server would directly send this information to the BrokerCORBAClient and BrokerRMIClient by using CORBA or RMI respectively to get connection. Similarly, the BrokerCORBAClient and BrokerRMIClient would

then handle the request from the broker side and send the results or exception the broker. The broker is like a RMI Server to handle the parameters or send the results from or to the client. The text-based client is passing the parameter or commands to the Broker side, and the Broker side would then parse the parameters sent from the client. This system also has the web –based client to send request or get the results through the Web Browser like 360, chrome, IE.