Scenario

by Simon Zambrovski

Table of contents

1. Sample Scenario Overview

The following scenario has been implemented as a sample application in order to demonstrate the framework functionality.

A sample travel agency has access to a sample airline Web Service and to a sample hotel Web Service. The agency offers a travel to a given destination by plane which lasts over several days, thereby a booking of a hotel is needed.

The scenario uses two domain-specific entities:

1.1. Airline Service

An airline service is implemented as a simple J2EE application, exposing its functionality over a Web Service. The business logic is implemented inside a Session EJB with the following semantic.

The AirlineManagerBean offers three methods to the client:

```
public <FlightInfo>Collection getFlights(String cityNameStart, String
cityNameFinish, Date dateOfFlight)
public FlightReservationInfo bookFlight(Integer flightNumber, String
travelAgencyName)
public boolean cancelReservation(FlightReservationInfo reservationInfo)
```

The relation between the AirlineManagerBean and other data types is shown in the following figure:

Sample Airline Logic

The typical invocation looks like:

1. Call getFlights() for given start, destination and date, retrieving the collection of FlightInfo entities
2. Select a flight and call bookFlight() providing a flight number and the travel agency name, retrieving a FlightReservationInfo with a valid seat number or an error (in form of NoSeatAvailableException)
3. If not needed anymore cancel the reservation calling the cancelReservation() and providing the FlightReservation object

The sample application provides some sample data that can be used during the sample runs. This data is stored in the database and can be adjusted as needed. After the call of bookFlight() the number of seats avaliable is decremented by one. The call of

cancelReservation() increments the number of seat by one.

1.2. Hotel Service

Similiar to airline, the hotel service is implemented as a simple J2EE application, exposing its functionality over a Web Service. The business logic is implemented inside a Session EJB with the following semantic.

The HotelManagerBean offers four methods to the client:

```
public int getAvailiableRooms(String hotelName, Date begin, Date end)
public ReservationInfo bookRoom(String hotelName, String agencyName, Date
begin, Date end)
public boolean cancelReservation(String hotelName, Integer roomNumber, Date
beginDate, Date endDate)
public String getHotelName(String cityName)
```

The relation between the HotelManagerBean and other entities is depicted in the next figure:

Sample Hotel Logic

Typical invocation looks like:

1. Call getHotelName() providing the city name to retrieve the name of the hotel in the city. (For simplicity, the example assumes, that only one hotel exists)
2. Retrieve the number of rooms avaliable in the hotel in a given period of time calling getAvaliableRooms().
3. Call bookRoom() in order to book a room and provide additionally the name of travel agency
4. If the order should be canceled call cancelReservation()

The sample application provides sample data for experiments. The sample hotel has a given number of rooms which can be booked. The startDate-endDate interval is evaluated in order to provide the number of avaliable rooms. A particular room is free if no reservation with overlapping interval is present in the system.