EJB 2.1 Exercises

Exercise Book

Version 1.0

EJB Exercises

Labs Setup:

Terminology:
<labs home> - Lab files installation root.

- **Database setup:**
  - Execute the SQL script tables.sql in your database.
  - Make sure that the following tables were created:
    BOOKS, CUSTOMERS, ORDERS, ORDER_ITEMS.

- **Application server setup:**

  Create JDBC Resources
  - Make sure you have a data source configured to work with your database with the following JNDI name: **jdbc/BookStoreDS**.

  Create JMS Resources
  Make sure you have configured the following JMS resources:
  - A queue connection factory under the JNDI name jms/qCon1.
  - A queue under the JNDI name jms/q1.

General:

Following exercises shall provide a partial functionality for an on-line bookstore.
Minimal requirements should include:

- Administration:
  - Add books to inventory
  - Register new customers
  - View all books / customers currently in DB

- Customers interface:
  - Purchase books using a shopping cart
  - Canceling a previously-placed order

Provided classes:

We have provided the following auxiliary classes.
Please feel free to include them in your project & use them when necessary.

- **interface BookstoreDAO**
  class SQL92BookstoreDAO
  class BookstoreDaoFactory

  These classes provide simple persistence capabilities using SQL statements.
  Before using them, please make sure that
  - All required tables have been generated (DDL are supplied)
  - An appropriate *DataSource* has been registered under ***"jdbc/ds1"***

  Usage example:

```
// Obtain DAO:
InitialConext ictx=new InitialContext();
DataSource  ds = (DataSource)ictx.lookup("jdbc/ds1");
BookstoreDAO  dao = BookstoreDaoFactory.getDAO(ds);
//  Insert book into DB:
dao.insertBook("The truth", "T.Pratchett",  100 );
```

- **BookDTO**
  CustomerDTO
  OrderDetailsDTO (not used)

  Simple serializable classes for transferring Book/Customer information between client & server.

Exercise 0 : simple session bean deployment

For this simple exercise, we shall create, deploy & test a very simple stateless session bean. Please follow the instructor's demonstration for deployment procedure.

- Define a bean using the 3 provided bean files (interface, implementation & home) And the provided XML files
- Compile, deploy & run on server
- Use the provided client for testing your bean

Provided files:

- **interface hello.Greet**

  Defines the methods exposed by our bean.
  Includes a single business method:

String getHelloMessage() throws RemoteException

- **class hello.GreetBean**

  Actual implementation for the Greet interface (add your implementation of our simple business method).

- **class hello.GreetHome**

  Defines the home , capable of creating instances of the Greet bean.

- **Required deployment files**

- **GreetClient**

  Looks up the bean's home, obtains a stub & invokes its business method.

Exercise 1 – Stateless Session Bean

This exercise involves the **Administrator functionality only.**
Please create a ***stateless session bean*** which exposes the following interface:

- ```
  void addNewBook(BookDTO book)
    throws InvalidBookDataException, RemoteException
  ```

  Adds a new book to inventory.
  BookDTO is a simple serializable class holding book details (provided).
  InvalidBookDataException is thrown if book data is invalid (e.g. negative price, empty title). It is sufficient to provide partial, example validity checks.

- ```
  void addNewCustomer(CustomerDTO customer)
    throws RemoteException
  ```

  Registers a new customer.
  The provided CustomerDTO is a serializable class holding customer details.

- ```
  List showBooks() throws RemoteException
  ```

  Allows the administrator to view all books in inventory.
  The returned list consists of BookDTO's.

- ```
  List showCustomers() throws RemoteException
  ```

  Presents a list of all registered customers (list of CustomerDTO's).

Note:

Please feel free to rely on provided classes:

- BookstoreDAO for DB access
- BookDTO, CustomerDTO

For additional information, please consult:

- The first page of this exercises book
- The provided classes' documentation

Testing:

Write a simple client to test your admin bean, by obtaining an Admin bean stub and invoking some methods (it is sufficient to use some simple, hard-coded data).

Exercise 2 – Stateful Session Bean

This exercise involves the **ShoppingCart factionality.**
Please create a ***stateful session bean*** .

This stateful session bean would hold the following data in memory:

- Customer's ID (provided during the bean's creation. Please make sure this is properly handled in both the bean's *ejbCreate* and in its home's *create* method)
- List of book titles to purchase (initialized to an empty LinkedList).

And would expose the following interface:

- ```
  String getCustomerId() throws
  RemoteException
  ```

  Returns the customer id, which was supplied during the bean's creation.

- ```
  void addToCart(String title)
    throws NoSuchBookException, RemoteException
  ```

  Adds a given title to the shopping cart.
  Note this manipulation is done in memory only. We shall write to the DB only when the user actually calls *placeOrder*.
  NoSuchBookException should be thrown when attempting to add an unknown book.

- ```
  List getTitlesInCart() throws
  RemoteException
  ```

  Returns the list of titles currently in the shopping cart.

- ```
  String placeOrder()
    throws EmptyOrderException, RemoteException
  ```

  Actually places the order by saving it into the DB.
  An exception is thrown if the purchase list is currently empty.

Note:

Again, you may rely on BookstoreDAO for DB access.
It already provides a method for placing an order with a list of items.

Testing:

Write a simple client to test your ShoppingCart bean, by obtaining a cart and adding some items to it.

Exercise 3 – Message Driven Beans

This exercise involves ***Order Cancellations*.**

Design considerations

For the sake of this exercise, we shall decide that clients, who wish to cancel previously-placed orders, would have to post asynchronous JMS messages into a dedicated queue.

Advantages of this approach:
- Non-blocking
- Reliability: cancellation requests may be placed by a JMS server even which the application server is down.

Important disadvantage:
- No easy way to notify customer if cancellation fails (e.g. due to an unknown order or DB failure).

Message format

Cancellation requests would arrive as *TextMessage*s , of the format *"cancel <order-id>"*
For instance:
"cancel 122333"

Each such message should include *a single order*.

Coding & deployment

- Define a dedicated queue (and related resources, such as queue connection factory & port)
- Create & deploy a message-driven bean with an appropriate *onMessage()* method.
  Implementation may rely on BookstoreDAO's method for removing an order (this method already performs a cascade-delete)
- Create a test client which would post some simple cancellation order.