CBSD - Component Based Software Development

Programming Coursework Two

Sachith Dassanayake (12712980)

All the projects are completed in NetBeans.

The folders are structured according to the questions on the course work.

I have used Glass fish server and Derby as the database for all the parts in this course work.

Before running the projects setup the databases cusdb and the StratfordDB

Use the databases in database directory

Or

Create two new databases cusdb and StratfordDB and run the sql in “db table setup.txt” on each database.

There are no validation done on these applications so invalid data may results errors and exceptions.

Java doc is provided with each and every project (Project Folder -> dist - > javadoc)

**Part 1**

1, 2, 3

Project Name: CusManagement

Change the values accordingly and run CusManagement.java

4

Can you identify possible bottlenecks on the client side in this scenario?

Security is very important in a client server application. So firstly it’s not a good idea to expose the database access directly over the internet.

The database operation functionalities are implemented on client application. So if there are any modifications to the database all the client applications must be updated.

Ability for fault tolerance is very difficult in this scenario.

If many clients connect to the database directly it can cause a high number of connections between the client and remote databases.

Managing large number of clients is extremely difficult as all the business logics are implemented on client side. So a simple modification made to the database has to be implemented on the client application.

Can you easily maintain and update a large number of clients?

If a client is connected to a database directly you also need to take care of concurrency issues and also when dealing with large number of records it is quite expensive to process over the internet.

Can you identify possible problems on the server side?

When there are many database connections and query requests, the database servers can be over loaded so it can cause performance issues or may be crashes if unable to handle the requests.

5

This part is also implemented under the same project.

Execute RMIServer.java first and then RMIClient.java individually (Do not run the project)

**Part 2**

1

Please see part 2.1

Run the project CusManagement. (This will show the index.jsp on the browser. Select the CRUD operation and the database to perform the operation. In this part the CRUD operations on address table are not implemented to move quickly to the next part)

2

Please see part 2.2

Run the project CustomerManagement. (This will show the index.jsp on the browser. Select the CRUD operation and the database to perform the operation. In this part the CRUD operations on all the tables of both the databases are implemented)

3

Advantages

JPA ensures the management of persistence and object / relational mapping. In earlier approach we had to hard code SQL in the application. By using JPA we did not have to write the vendor specific SQL so this increases the portability of the application. JPA uses annotations which replaces the lots of cumbersome code. JPA supports caching so it can be faster compared to the JDBC approach in the previous task. It also supports polymorphism and inheritance.

Disadvantages

For an application which has a database of only 3 tables, JPA can be overkill. JPA will require additional jar files which adds to the size of the application.

4

Disadvantages

The main disadvantage of adding layers is the speed. Adding layers means basically executing more codes so it reduces the performance of the application. The abstraction layers may limit the available operations such as backend-specific optimizations or debugging features on the databases.

Advantages

Increases the scalability, portability of the application

**Part 3**

1.

Please see part 3.1

Deploy the project CusManagementWS

2.

Please see part 3.2

Change the values accordingly on CMClientWS.java and run the project CMClientWS

3.

Advantages

Web services are platform and language independent. Means the client application can be programmed in C# and running under Linux whilst the web service can be programmed in Java and running under Windows.

As our web services use HTTP for transmitting service requests and responses, it’s a major advantage for internet scale applications because of the cost and less problems with HTTP traffic from firewalls and internet proxies.

Disadvantages

Transmitting the data in XML is not as efficient as transmitting in binary code. So it can slow down the performance of the application.

As the web services are exposed to the public using the HTTP protocol security is another concern.

**Part 5**

1, 2

JMSTopicSenderClient – Acts as a client pc

JMSTopicReceiver – Acts as a local office

JMSQueueReceiver – Central Office in Dagenham

**Glass fish JMS Resources names**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  | |
|  | |  | |  | |  | |
|  |  | |  | |  | |
|  |  | |  | |  | |

* Connection Factories

Pool Name: jms/CusManagementConnectionFactory [jms/CusManagementConnectionFactory](http://localhost:4848/jms/jmsConnectionEdit.jsf?name=jms/CusManagementConnectionFactory&jndiName=jms/CusManagementConnectionFactory)

JNDI Name: jms/CusManagementConnectionFactory

Resource Type: javax.jms.ConnectionFactory javax.jms.ConnectionFactory

* Destination Resources

JNDI Name: jms/CusManagementQueue

jms/CusManagementQueue

Resource Type: javax.jms.Queue javax.jms.Queue

JNDI Name: jms/CusManagementQueue

jms/CusManagementTopic

Resource Type: javax.jms.Queue javax.jms.Topic

Create the above JMS resources

Deploy the projects JMSQueueReceiver and JMSTopicReceiver

Run the project JMSTopicSenderClient (Please use a valid id to delete a customer and delete a country from Dagenham database)

3.

From client to local offices I have used Publish/Subscribe Messaging

And from local offices to the central office I have used Point-to-Point Messaging

4.

The client pc acts as the publisher and the local offices act as subscribers. Subscribers are registered on a topic to receive the messages. When the client send a message to the channel and each subscribers receive this message.

Once the message is received by the subscribers (local offices) it then acts as a client and forwards the message to a queue. Each message has only one consumer (central office). If there is no consumer available to receive the message it will be kept until a consumer is available that can process the message.

In this application the central office reads the message and removes the customer and the country record by their ids sent from the client.

In my application I have used message driven beans to act as a listener for the JMS and to process the messages asynchronously.

Advantages

Due to asynchronous messaging of JMS, not all the pieces need to be up and running all the time for the application to function as a whole. If the receiver / receivers are down the message-oriented middleware can store the messages until they come back.

Loose coupling" between message consumers and message producers

High degree of anonymity between producer and consumer

High scalability

Disadvantages

As JMS only supports asynchronous messaging, it will be difficult when addressing response correlation, error handling, and data synchronization.