COS30041 Creating Secure and Scalable Software [Java EE]

06 Pass Task 6.1 Stateful Business Logic

Time Frame: Weeks 6 – 8

Suggested to start in Week 6 and complete in Week 8
Submission Due: Week 8, Fri, 6:30pm

Overview

In this task, you are required to program business layer objects that has stateful behaviour and can handle business logics on the server using related technologies. You are also required to demonstrate your work is of good quality.

Purpose	To demonstrate your ability to use relevant technologies to develop quality application that handles the "stateful" business logics on the server
Tasks	1. Learn to develop an enterprise application that uses stateful session bean to emulate a shopping cart behaviour, usually found on e-Commerce Web site 2. Extend the stateful session bean so that it can handle CRUD operations 3. Extend the application client developed in 1 above to act as a test harness to perform testing 4. Prepare your test cases and test your application thoroughly by using appropriate input values and database contents 5. Describe (with justification) and document your design
Pre-req Task[1]	
Follow-up Task[2]	
Suggested Time	1 – 2 hour if you know the stuff well 4 – 5 hours if you need to read the concepts and know how to establish database connections
Resources	Lecture 04a Business Logics 1 Lecture 04b EJB Lecture 04c Stateless Session Bean Lecture 06 Stateful Session Bean Java EE – Stateful Session EJBs
Feedback	Ask your tutor for feedback
Next task	Pass Task 7.1

Pass Task 6.1 Submission Details and Assessment Criteria

You must create your own document (pdf) in **portrait** mode[3], which you will upload to Doubtfire, with the following details:

- Your name and student id
- Your tutor's name
- Your own responses to the tasks according to the corresponding instructions (see below)

Tasks and Instructions

Task 1. Complete Lab_06_SFSB_ShopCart

[1]You need to complete the pre-requisite task before doing this task.
[2]You need to complete this task in order to do the follow-up task because the follow-up task depends on your answer in this one.
[3]Landscape mode pdf does not work properly in Doubtfire.

Task 2. Programming [Assume you have completed Task 1 above]
Add the following methods in the "ShopCartBean.java" class (Lab_06_SFSB_ShopCart) that supports the remaining CRUD operations of CartItem

1. "`public boolean addCartItem(CartItem cartItem)`" – accepts a CartItem object and checks whether there is a CartItem object in the Shopping Cart having the same itemId as that of cartItem. If there is, the method updates the quantity of the relevant CartItem object by adding the two quantities together. Otherwise, it just adds the cartItem to the Shopping Cart. Either case, the method returns `true`. In case, there are any exception being thrown, catch it and return `false`. After successfully implement this method, change the signature of the "add(CartItem cartItem)" method to "private" and remember to change the relevant method calls to "addCartItem" in your ShopCartAppClient.java. Otherwise, you will get complaints from NetBeans. Also remember to change the signature in the Remote Interface class as well.
   Note: Here, we have not implement the PRODUCT database (or, more precisely the inventory management system). As a result, we could not check whether we have the actual product item for sale. We just assume that we have the item and have enough stock to sale. In real business, the situation will be quite different, depending on the actual business process the company adopts.

Note: After finishing the "addCartItem()" method in 1 above and always calling this method to add CartItem into the Shopping Cart, all "itemId"s in the CartItems in the Shopping Cart are different. Hence, we can safely assume, for a particular itemId, there will only be at most one such CartItem in the Shopping Cart.

2. "`public boolean deleteCartItem(String itemId)`" – accepts a String object whose value is supposed to be the itemId of a product in one of the CartItems in the Shopping Cart. It then checks whether there is a CartItem object in the Shopping Cart having the same itemId. If there is, it then removes the CartItem object from the Shopping Cart and returns `true`. Otherwise, it returns `false`.

3. "`public boolean updateCartItem(CartItem cartItem)`" – accepts a CartItem object and checks whether there is a CartItem object in the Shopping Cart having the same itemId as that of cartItem. If there is, it replaces that CartItem object with the current cartItem and returns `true`. Otherwise, it returns `false`.

Task 3. Programming [Assume you have completed Task 2 above]
Modify the ShopCartAppClient.java class so that it acts as a test harness of the methods you implemented in Task 2 above.

Note: Although this is a test harness, there is no need to use JUnit Tests.

Task 4. Testing
Write your test cases (including the database content and input values) and test your work thoroughly. Remember to collect the relevant screen dump. Console output in the Output tab of NetBeans is fine.

Note: For this task, organize the test cases in "ShopCartAppClient.java" properly will do. For example, when testing whether a CartItem could be added or not, you need to show the evidence that your method can (1) "add a CartItem to the Shopping Cart where the Shopping Cart does not have a CartItem having the same "itemId"; (2) "sum the two quantities together if the Shopping Cart does have a CartItem having the same "itemId"; and (3) "catch an exception and return false (e.g. pass cartItem as "null" to addCartItem()).

Task 5. Design Justification

In a real e-commerce web site of "programming a shopping cart", we need at least the following tables on the database server and at least several session beans in the "Business (EIS) server" other than the software components on the Web server.

Database Table [for storing relevant data]

1. PRODUCT – records all product information including quantity_on_hand (a number showing how many the company has that item on stock). The "productId" is likely to be the primary key)
2. ORDER – records all order information (orderId is likely to be the primary key)
3. CUSTOMER – records all information about a customer
4. PAYMENT – records all payment information related to each order (e.g. Credit Card payment only for simplicity)

Entity Class [for ORM purposes]

1. Product
2. …

Session Bean [for implementing business logics]

1. ShoppingCartBean – Stateful
2. ProductFacade – Stateless
3. …

Assume that you are about to build such a basic e-commerce web site that allows customers to (1) add / update / remove items in the shopping cart and (2) perform check out and payment.

5.1. Draw an architecture diagram, using UML notations, of your entire set of programs in building this web site. In your diagram, you need to annotate the names and types of each software component, put them in the right tiers, and indicate how they interact with each other.
Note: My diagram in the Lab 06 is not in UML notation.

5.2. Describe and discuss the role of each of the components in your diagram. Justify your decision, especially on the roles of each session bean.

Submission Task

Once completed, you need to submit a pdf file that contains all your work (e.g. selected code segments – show me the key stuff and some screen dumps of your testing)

Demonstration

You may be asked to demonstrate your assignment in the lab. You should be able to do this and explain your code when asked in the practical session.