Cart Service { 2 .5 hours }

Problem Statement :

1. As a customer I should be able to create a cart for the shopping
2. As a customer I should be able to add/remove/update items in the shopping cart

CART

Cart_Id
Customer_Id

Customer

Customer_Id
Customer_Name

Item

Item_Id
Name
Description
MfgDate
Category
Cost

Tech Stack

Java
Framework - SpringBoot
Database - MySQL | Any RDBMS - H2

Considerations :

1. Need to create the REST endpoints
   What are those ? ---- (Need to define TBD)
2. We should be able to handle the exceptions well
3. We should be able to do the proper logging , in an external log file
4. We should be writing the Unit Test cases for the code written

Code Implementation

Step1 : Bootstrapping the project - Spring Boot

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-parent</artifactId>
 <version>2.3.4.RELEASE</version>
 <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.unacademy</groupId>
<artifactId>cartService</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>cartService</name>
```

```xml
<description>HandsOn coding for the Cart Service</description>
<properties>
 <java.version>11</java.version>
</properties>
<dependencies>
 <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
 </dependency>

 <dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
 </dependency>

 <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
 </dependency>

 <dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
 </dependency>
</dependencies>

<build>
 <plugins>
  <plugin>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-maven-plugin</artifactId>
  </plugin>
 </plugins>
</build>

</project>
```

Step2 : Start working on the DAL

Configuration

1. Add the configurations for the DB in application.properties
2. Add the dependency of MySQL driver in the
3. POM.xml
4. My application is able to connect to the DB

```
spring.datasource.url=jdbc:mysql://localhost:3306/cartdb
spring.datasource.username=root
spring.datasource.password=Welcome1

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

Entities → creation of the table in the database

```
@Entity
public class Cart {

 @Id
 @GeneratedValue(strategy =
GenerationType.AUTO)
 private int cartId;

 @Column(nullable = false,
unique = true)
 private int customerName;

 @OneToMany(mappedBy =
"cart")
 private List<Item> items ;

 public int getCartId() {
   return cartId;
 }
 public void setCartId(int
cartId) {
    this.cartId = cartId;
  }

 public int getCustomerName()
{
    return customerName;
  }

 public void
setCustomerName(int
customerName) {
    this.customerName =
customerName;
  }
}
```

```
@Entity
public class Item {

 @Id
 @GeneratedValue(strategy =
GenerationType.AUTO)
 private int itemId ;

 @Column(nullable = false)
 private String itemName;

 @Column(length = 512)
 private String
itemDescription;

 private String category ;
 @Column(nullable = false)
 private double cost ;

 @Column(nullable = false)
 private LocalDateTime
mfgDate;

 /**
   * For the relationship
   */
 @ManyToOne
 @JoinColumn(name="cart_id")
// used to customize the FK
column, join column
 private Cart cart;

 public double getCost() {
    return cost;
```

```
  }

  public void setCost(double
cost) {
    this.cost = cost;
  }

  public int getItemId() {
    return itemId;
  }

  public void setItemId(int
itemId) {
    this.itemId = itemId;
  }

  public String getItemName()
{
    return itemName;
  }

 public void
setItemName(String itemName)
{
    this.itemName = itemName;
  }

 public String
getItemDescription() {
    return itemDescription;
  }

 public void
setItemDescription(String
itemDescription) {
    this.itemDescription =
itemDescription;
  }
 public String getCategory()
{
    return category;
  }

 public void
setCategory(String category)
{
    this.category = category;
  }
}
```

Entities are defined -- What next ?

Cart < One to Many > Item

Item < Many to One > Cart

What Next ?

DAL Layer

```
public interface ItemDao extends
JpaRepository<Item, Integer> {

}
```

```
public interface CartDao extends
JpaRepository<Cart, Integer> {
}
```

Testing the DAO Layer code

```
ApplicationContext applicationContext =
SpringApplication.run(CartServiceApplication.c
lass, args);

CartDao cartDao =
applicationContext.getBean(CartDao.class);
ItemDao itemDao =
applicationContext.getBean(ItemDao.class);

System.out.println("cart dao "+ cartDao);
System.out.println("item dao "+ itemDao);

/**
* Create Cart
*/
Cart cart = new Cart();
cart.setCustomerName("Vishwa Mohan");
cartDao.save(cart);

/**
* Items
*/
Item item = new Item();
item.setItemName("Detergent Powder");
item.setCategory("Households");
item.setItemDescription("Great product");
item.setCost(400);
item.setMfgDate(LocalDateTime.of(2021,07,30,12
,40));
item.setCart(cart);
itemDao.save(item);

System.out.println("Hello Students");
```

```
public interface CartService {

 /**
  * This method will add a new
Cart, and persist in the DB
  * @param cart
  * @return
  */
 public Cart createCart(Cart
cart);

 public boolean deleteCart(int
cartId);

 public Cart findByCartId(int
cartId) throws
CartNotFoundException;

 public Cart
findByCustomerName(String
customerName) throws
UserNameNotFoundException;
}
```

```
@Service
public class CartServiceImpl
implements CartService {

 @Autowired
 private CartDao cartDao ;

 @Override
 public Cart createCart(Cart
cart) {
   return cartDao.save(cart);
 }

 @Override
 public boolean deleteCart(int
cartId) {
   cartDao.deleteById(cartId);
   return true;
 }

 @Override
 public Cart findByCartId(int
cartId) throws
CartNotFoundException {
   return
cartDao.findById(cartId).orElseT
hrow(() -> new
CartNotFoundException());
 }

 @Override
 public Cart
findByCustomerName(String
customerName) throws
UserNameNotFoundException {
   Cart cart =
cartDao.findByCustomerName(custo
merName);
   if(cart== null){
     throw new
UserNameNotFoundException();
   }
   return cart ;
 }
}
```

Test Class

```
package
com.unacademy.cartService.services.impl;

import
com.unacademy.cartService.daos.CartDao;
import
```

```
com.unacademy.cartService.entities.Cart;
import
com.unacademy.cartService.exceptions.CartNot
FoundException;
import
com.unacademy.cartService.exceptions.Custome
rNameNotFoundException;
import java.util.Optional;
import org.junit.Assert;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import
org.springframework.boot.test.context.Spring
BootTest;


@SpringBootTest
public class CartServiceImplTest {

 @Mock
 CartDao cartDaoMock ;

 @InjectMocks
 CartServiceImpl cartService ;
 /**
  * Test for createCart
  */
 @Test
 public void testCreateCart(){
   // Create the data
   Cart cart = new Cart();
   cart.setCustomerName("Vishwa Mohan");

   Cart cart1 = new Cart();
   cart1.setCustomerName("Vishwa Mohan");
   cart1.setCartId(1);


   // Give the functionality to the mock

Mockito.when(cartDaoMock.save(cart)).thenRet
urn(cart1);

   //Execute
   Cart savedCart =
cartService.createCart(cart);


   //Testing
   Assert.assertNotNull(savedCart);
   Assert.assertEquals(1,
savedCart.getCartId());
```

```
 }

 /**
  * Test for deleteCart
  */
 @Test
 public void testDeleteCart(){

    //Data



    //Mock   -- When we have to mock a method
that returns nothing

Mockito.doNothing().when(cartDaoMock).delete
ById(1);

    //Execution

    //Assertion

 }
```

```java
 /**
  * Test for findByCartId
  */
 @Test
 public void testFindByCartId() throws
CartNotFoundException {
    //Data
    Cart cart1 = new Cart();
    cart1.setCustomerName("Vishwa Mohan");
    cart1.setCartId(1);
    //Mock

Mockito.when(cartDaoMock.findById(1)).thenRe
turn(Optional.of(cart1));



    //Execution
    Cart cart = cartService.findByCartId(1);


    //Assertion
    Assert.assertNotNull(cart);
    Assert.assertEquals("Vishwa Mohan" ,
cart.getCustomerName());
 }
 @Test
 public void
testFindByCartIdThrowsExceptions() throws
CartNotFoundException {
   //Data
   Cart cart1 = new Cart();
   cart1.setCustomerName("Vishwa Mohan");
   cart1.setCartId(1);

   //Mock

Mockito.when(cartDaoMock.findById(1)).thenRe
turn(Optional.empty());

   //Execution
   try {
     Cart cart =
cartService.findByCartId(1);
   }catch (Exception e){

Assert.assertEquals(CartNotFoundException.cl
ass , e.getClass());
   }
 }
```

```java
 /**
   * Test for findByCustomerName
   */

 @Test
 public void testFindByCustomerName() throws
CustomerNameNotFoundException {
    Cart cart1 = new Cart();
    cart1.setCustomerName("Vishwa Mohan");
    cart1.setCartId(1);


Mockito.when(cartDaoMock.findByCustomerName(
"Vishwa Mohan")).thenReturn(cart1);

    //Execution
    Cart cart =
cartService.findByCustomerName("Vishwa
Mohan");


    //Assertion
    Assert.assertNotNull(cart);
    Assert.assertEquals("Vishwa Mohan" ,
cart.getCustomerName());
```

```
    }
```

```
}
```

Mapper

```
package
com.unacademy.cartService.utils;

import
com.unacademy.cartService.dtos.C
artDTO;
import
com.unacademy.cartService.dtos.I
temDTO;
import
com.unacademy.cartService.entiti
es.Cart;
```

```
import
com.unacademy.cartService.entiti
es.Item;


/**
* This class will be used to map
DTO <--> Entity
*/
public class DTOEntityMapper {

 /**
   * For Mapping the CartDTO to
Cart Entity
   * @param cartDTO
   * @return
   */
 public static Cart
```

```
convertCartDTOToCartEntity(CartD
TO cartDTO){

   Cart cart = new Cart();

cart.setCartId(cartDTO.getCartId
());

cart.setCustomerName(cartDTO.get
CustomerName());

   for(ItemDTO itemDTO :
cartDTO.getItems()){

cart.getItems().add(convertItemD
TOToItemEntity(itemDTO));
   }
   return cart;

 }

 private static Item
convertItemDTOToItemEntity(ItemD
TO itemDTO) {
   Item item = new Item();


item.setItemName(itemDTO.getItem
Name());

item.setItemDescription(itemDTO.
getItemDescription());

item.setCost(itemDTO.getCost());
item.setCategory(itemDTO.getCate
gory());

item.setMfgDate(itemDTO.getMfgDa
te());

item.setItemId(itemDTO.getItemId
());

   return item;

 }

 public static CartDTO
convertCartEntityToCartDTO(Cart
cart){
   CartDTO cartDTO = new
CartDTO();
```

```
cartDTO.setCartId(cart.getCartId
());

cartDTO.setCustomerName(cart.get
CustomerName());

   for(Item item :
cart.getItems()){

cartDTO.getItems().add(convertIt
emEntityToItemDTO(item));
   }
   return cartDTO;

 }

 private static ItemDTO
```

```
convertItemEntityToItemDTO(Item
item) {

   ItemDTO itemDTO = new
ItemDTO();


itemDTO.setItemName(item.getItem
Name());

itemDTO.setItemDescription(item.
getItemDescription());

itemDTO.setCost(item.getCost());

itemDTO.setCategory(item.getCate
gory());
itemDTO.setMfgDate(item.getMfgDa
te());

itemDTO.setItemId(item.getItemId
());

   return itemDTO;
 }
}
```

```
package
com.unacademy.cartService.contro
llers;

import
com.unacademy.cartService.daos.C
artDao;
```

```
import
com.unacademy.cartService.dtos.C
artDTO;
import
com.unacademy.cartService.dtos.I
temDTO;
import
com.unacademy.cartService.entiti
es.Cart;
import
com.unacademy.cartService.entiti
es.Item;
import
com.unacademy.cartService.except
ions.CartNotFoundException;
import
com.unacademy.cartService.except
ions.ItemNotFoundException;
```

```
import
com.unacademy.cartService.servic
es.CartService;
import
com.unacademy.cartService.servic
es.ItemService;
import
com.unacademy.cartService.utils.
DTOEntityMapper;
import
org.apache.coyote.Response;
import
org.springframework.beans.factor
y.annotation.Autowired;
import
org.springframework.http.HttpSta
tus;
import
```

```
org.springframework.http.Respons
eEntity;
import
org.springframework.web.bind.ann
otation.GetMapping;
import
org.springframework.web.bind.ann
otation.PathVariable;
import
org.springframework.web.bind.ann
otation.PostMapping;
import
org.springframework.web.bind.ann
otation.RequestBody;
import
org.springframework.web.bind.ann
otation.RequestMapping;
import
```

```
org.springframework.web.bind.ann
otation.RestController;


@RestController
/**
*
127.0.0.1:8080/cartService/v1/ca
rts
*/
@RequestMapping("/carts")
public class CartController {

 @Autowired
 private CartService
cartService;

 @Autowired
 private ItemService
itemService;

 @GetMapping
 public ResponseEntity
helloStudents(){
   return new
ResponseEntity("Hello Students",
HttpStatus.OK);
 }

 /**
  * create an endpoint to create
the cart
  *
  * POST
127.0.0.1:8080/cartService/v1/ca
rts
```

```
  *   Body request  -- JSON
  */
 @PostMapping
 public ResponseEntity
createCart(@RequestBody CartDTO
cartDTO){
   //Create and save Cart in the
system
   Cart cart
=cartService.createCart(DTOEntit
yMapper.convertCartDTOToCartEnti
ty(cartDTO));

   //Convert the Cart entity
back to the CartDTO
   CartDTO cartResponse =
DTOEntityMapper.convertCartEntit
yToCartDTO(cart);
```

```
    //Return the response
    return new
ResponseEntity(cartResponse,
HttpStatus.CREATED);
 }

 /**
   * Search a cart based on the
cartId
   *
   * GET
127.0.0.1:8080/cartService/v1/ca
rts/{cart_id}
   */
 @GetMapping("/{cart_id}")
 public ResponseEntity
getCart(@PathVariable("cart_id")
int cartId) throws
CartNotFoundException {
   Cart cart =
cartService.findByCartId(cartId)
;

   CartDTO cartResponse =
DTOEntityMapper.convertCartEntit
yToCartDTO(cart);
   return new
ResponseEntity(cartResponse,
HttpStatus.OK);

 }

 /**
  * Add an item in the cart
  */
```

```
@PostMapping("/{cart_id}/items")
 public ResponseEntity
addItemToCart(@RequestBody
ItemDTO itemDTO,
@PathVariable("cart_id") int
cartId)
     throws
CartNotFoundException,
ItemNotFoundException {

   // I need to create item
inside the cart.. so need
ItemService
   Item item =
itemService.addItemToCart(DTOEnt
ityMapper.convertItemDTOToItemEn
tity(itemDTO), cartId);
```

```
   Cart cart =
itemService.getCartOfTheItem(ite
m.getItemId());
   CartDTO cartDTO =
DTOEntityMapper.convertCartEntit
yToCartDTO(cart);

   return new
ResponseEntity(cartDTO,
HttpStatus.CREATED);
 }



}
```