**ANUBHAW SWARUP**

**8904475828**

TABLES

create table Accounts (

        User\_Name varchar(20) not null,

        Password varchar(20) not null,

        User\_Role varchar(20) not null,

        primary key (User\_Name)

    );

create table Products (

        Code varchar(20) not null,

        Create\_Date datetime not null,

        Name varchar(255) not null,

        Price double precision not null,

        primary key (Code)

    );

create table Orders (

        ID varchar(50) not null,

        Amount double precision not null,

        Delivery\_address varchar(255) not null,

        Email varchar(128) not null,

        Customer\_name varchar(255) not null,

        Phone varchar(128) not null,

        Order\_date datetime not null,

        Order\_Num integer not null,

        primary key (ID)

    );

**JAVA CODE**

API:

1. Add Account
2. Add products
3. Create Order

CONTROLLERS:

package com.ecom.controller;

@Controller

@RequestMapping("sample")

**public** **class** EcomController {

@Autowired

**private** EcomService ecomService;

@RequestMapping(value = "/{id}", produces ="application/json", method = RequestMethod.***GET***)

**public** CompletableFuture<ResponseEntity<String>> getBook(@PathVariable **int** id) {

**return** CompletableFuture.*completedFuture*(**new** ResponseEntity<String>(ecomService.get(id), HttpStatus.***OK***));

}

@RequestMapping(value = "/addProduct", consumes = "application/json", method = RequestMethod.***PUT***)

**public** CompletableFuture<ResponseEntity<Void>> addProduct(@RequestBody Product product) {

**return** CompletableFuture.*completedFuture*(**new** ResponseEntity<>(HttpStatus.***OK***));

}

@RequestMapping(value = "/addAccount", consumes = "application/json", method = RequestMethod.***PUT***)

**public** CompletableFuture<ResponseEntity<Void>> addAccount(@RequestBody Product product) {

**return** CompletableFuture.*completedFuture*(**new** ResponseEntity<>(HttpStatus.***OK***));

}

@RequestMapping(value = "/addOrder", consumes = "application/json", method = RequestMethod.***PUT***)

**public** CompletableFuture<ResponseEntity<Void>> addOrder(@RequestBody Product product) {

**return** CompletableFuture.*completedFuture*(**new** ResponseEntity<>(HttpStatus.***OK***));

}

}

**MODELS:**

package com.ecom.model;

**ACCOUNT**

@JsonIgnoreProperties(ignoreUnknown = true)

public class Account {

@JsonProperty("User Name")

private String userName;

@JsonProperty("Password")

private String password;

@JsonProperty("UserRole")

private String role;

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

}

**PRODUCT**

@JsonIgnoreProperties(ignoreUnknown = true)

public class Product {

@JsonProperty("Code")

private UUID code;

@JsonProperty("Date")

private Date date;

@JsonProperty("Name")

private String name;

@JsonProperty("Price")

private Double price;

public UUID getCode() {

return code;

}

public void setCode(UUID code) {

this.code = code;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

**ORDER**

@JsonIgnoreProperties(ignoreUnknown = true)

public class Order {

private Double orderNumber;

@JsonProperty("Amount")

private Double amount;

@JsonProperty("Delivery\_address")

private String address;

@JsonProperty("Email")

private String email;

@JsonProperty("Phone")

private String phone;

@JsonProperty("Customer\_name")

private String name;

@JsonProperty("Order\_date")

private Date date;

private List<OrderedProduct> orderedProducts;

public List<OrderedProduct> getOrderedProducts() {

return orderedProducts;

}

public void setOrderedProducts(List<OrderedProduct> orderedProducts) {

this.orderedProducts = orderedProducts;

}

public Double getOrderNumber() {

return orderNumber;

}

public void setOrderNumber(Double orderNumber) {

this.orderNumber = orderNumber;

}

public Double getAmount() {

return amount;

}

public void setAmount(Double amount) {

this.amount = amount;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

}

**ORDERED PRODUCT**

public class OrderedProduct {

@JsonProperty("product")

private String productName;

@JsonProperty("quantity")

private Double quantity;

public String getProductName() {

return productName;

}

public void setProductName(String productName) {

this.productName = productName;

}

public Double getQuantity() {

return quantity;

}

public void setQuantity(Double quantity) {

this.quantity = quantity;

}

}

**ACCOUNTDAO**

**package com.ecom.Dao;**

@Repository

public class AccountDao {

@Autowired

private JdbcTemplate jdbctemplate;

public String get(int id) {

return jdbctemplate.queryForObject("select name from sample1 where

quantity=?", String.class,id);

}

public void addAccount() {

//add account and user details along with it;

}

}

**ORDERDAO**

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Repository;

import com.ecom.model.Order;

@Repository

public class OrderDao {

@Autowired

private JdbcTemplate jdbctemplate;

public String get(int id) {

return jdbctemplate.queryForObject("select name from sample1 where quantity=?",

String.class, id);

}

public void addOrder(Order order) {

// adding order with details using insert query in to the ORDER table

}

}

**PRODUCTDAO**

@Repository

public class ProductDao {

@Autowired

private JdbcTemplate jdbctemplate;

@Autowired

private NamedParameterJdbcTemplate namedJdbcTemplate;

public String get(int id) {

return jdbctemplate.queryForObject("select name from sample1 where

quantity=?", String.class,id);

}

public void addProduct() {

MapSqlParameterSource parameters = new MapSqlParameterSource();

// add product to it

namedJdbcTemplate.update("",parameters/\*insert query);\*/);

}

public void buyProduct(String productName, Double quantity) {

// 1 get product name and current quantity from the DB

// if(current quantity is less than what he wants to buy then only give as much left)

// if(current quantity is == 0) return

// 2 update query to change quantity

}

}

**SERVICE**

package com.ecom.service;

@Service

public class EcomService {

@Autowired

private AccountDao accountDao;

@Autowired

private OrderDao orderDao;

@Autowired

private ProductDao productDao;

public String get(int id) {

return accountDao.get(id);

}

public Integer addOrder(Order order) {

synchronized (this) {

boolean validOrder = validate(order);

if (validOrder) {

try {

processOrder(order.getOrderedProducts());

orderDao.addOrder(order);

return 1;

} catch (Exception e) {

// log the error

return 0;

}

} else

return 0;

}

}

private void processOrder(List<OrderedProduct> orderedProducts) {

orderedProducts.stream().forEach(prod->{

productDao.buyProduct(prod.getProductName(),prod.getQuantity());

});

}

private boolean validate(Order order) {

return !(order.getAmount()==0);

}

@Transactional

public void addAccount(Account account) {

accountDao.addAccount();

}

@Transactional

public void addProduct(Product product) {

productDao.addProduct();

}

}