How to use @Transactional(propagation = Propagation.***REQUIRES\_NEW***)

A user is trying to buy a flight ticket and while booking the ticket, the user does not receive the email notification that ticket has been booked because SMTP server was down in the booking site or application. In this case, ticket details should be saved but notification details may not be saved so that user can request for email notification for some other day. Code is given below.

@Autowired

**private** BookingService bookingService; 🡸 Code inside AutoRun class

**private** **void** trial3() {

Customer user = **new** Customer("Hari", "Hari@ddlab.com");

Ticket ticket = **new** Ticket(1000, "Hari");

**bookingService.book(ticket, user);**

}

@EventListener(ApplicationReadyEvent.**class**)

**public** **void** run() {

trial3();

System.***out***.println("Ticket booked successfully ...");

}

Main Service Class and it internally calls two other service classes with propagation as Requires New.

@Service

**public** **class** BookingService {

**@Autowired**

**private TicketBookingServiceImpl tktService;**

**@Autowired**

**private EmailNotificationServiceImpl notificationService;**

@Transactional

**public** **void** book(Ticket tkt, Customer cust) {

tktService.book(tkt, cust); // Ticket will be booked

notificationService.saveAndNotify(cust);

}

}

@Service

**public** **class** TicketBookingServiceImpl {

**@Autowired**

**private TicketRepository ticketRepo;**

@Transactional(propagation = Propagation.***REQUIRES\_NEW***)

**public** **void** book(Ticket ticket, Customer user) {

ticketRepo.save(ticket); // Save only ticket information

}

}

@Service

**public** **class** EmailNotificationServiceImpl {

@Autowired

**private** CustomerRepository customerRepo;

@Transactional(propagation = Propagation.***REQUIRES\_NEW***)

**public** **void** saveAndNotify(Customer customer) {

customerRepo.save(customer);

sendMail();

}

**public** **void** sendMail() {

**throw** **new** NullPointerException("SMTP server down...");

}

}

In the above case, ticket table will be populated with data and customer table will not be populated.

@Entity(name="Ticket") @Table(name="ticket")

@Getter @Setter @NoArgsConstructor

**public** **class** Ticket {

@Id @GeneratedValue

**private** Long id;

**private** **int** price;

**private** String name;

**public** Ticket(**int** price, String name) {

**// Constructor**

}

}

@Entity(name="Customer") @Table(name="customer")

@Getter @Setter @NoArgsConstructor

**public** **class** Customer {

@Id @GeneratedValue

**private** Long id;

**private** String name;

**private** String email;

**public** Customer(String name, String email) {

**//Constructor**

}

}