Beaconfire Inc, Home Work, Week6 Day25.
By (Ping) Nalongsone Danddank.
ndanddank@gmail.com
wechatID: ndanddank
Short Answer:

1. What is “Offline Transaction”?

-> After user submit an order, instead of asking user to wait online until all process, such as validation and saving to database, completed, we would like to process the transaction offline and after all process have completed we send an email to inform customer.

2. What is RabbitMQ and what can it help us to achieve in a web application?

-> is one of such message broker. if we want to trade messages between two distributed software components, we need a mediator.

• This mediator is known as the message broker.
• It receives incoming messages from a sender and sends them to a recipient. This way the sender and receiver can be totally isolated

3. What are the component of RabbitMQ?

-> The main components of RabbitMQ are Producer, Exchange, Queue and Consumer.
It has three entities
• Exchange
• Queue
• Binding
• When a publisher pushes a message to RabbitMQ
• It first arrives at an exchange.
• The exchange then distributes copies of these messages to variously connected queues.
• Finally, consumers receive these messages

4. What are different types of Exchange that exist in RabbitMQ?

-> Primarily there are three types.
• Direct — A direct exchange delivers a message directly to the queues that satisfy the below condition:
Routing key == Binding key
  • A routing key is an attribute of the message.
  • On the other hand, a binding key is something you specify while creating a binding between a queue and an exchange.
• Fanout: — A Fanout exchange ignores routing keys and distributes a message to all the connected queues.
• Topic: A topic exchange routes a message by matching routing key with pattern in the binding key like below:
Routing key == Pattern in binding key.
  • RabbitMQ uses two wild card characters for pattern matching * and #.
  • Use a * to match 1 word and a # to match 0 or more words
  • This type of exchange has a vast range of use cases.
  • It can be used in the publish-subscribe pattern, distributing relevant data to desiring workers processes and many more.
  • In most case, we will have the topic exchange in enterprise application.
• Header — A header is a particular type of exchange that routes messages based on keys present in the

message header. It overlooks routing key attribute of the message.

- When creating bindings for a header exchange, it is possible to bind a queue to match more than one header.
- In such a case, RabbitMQ should know from the producer if it should match all or any of these keys.
- It is rarely used exchange types.

5. What is Scheduler and what can it help us to achieve in a web application?

-> Scheduling is to execute tasks for specific time period.

- Fixed Rate
  - The method is invoked for every specified time.
  - Note that the time measured between successive start times of each invocation.
  - In other words, the task is invoked again even if the previous invocation of the task is not finished.

```
@Scheduled(fixedRate = 1000)
public void scheduledTaskWithFixedRate() {
    Logger.debug("scheduledTaskWithFixedRate: current time: " + new Date());
}
```

- Fixed Delay
  - The method is invoked for every specified time like for fixedRate but the time is measured from completion time of each preceding invocation.

```
@Scheduled(fixedDelay = 3000)
public void scheduledTaskWithFixedDelayAndProcessingTime() throws InterruptedException {
    Logger.debug("scheduledTaskWithFixedDelayAndProcessingTime: started: " + new Date());
    TimeUnit.SECONDS.sleep(2);
    Logger.debug("scheduledTaskWithFixedDelayAndProcessingTime: finished: " + new Date());
}
```

- Initial Delay
  - We can specify the milliseconds to wait before first execution of task.
  - It could be used with both fixedRate and fixedDelay

```
@Scheduled(initialDelay = 5000, fixedRate = 1000)
public void scheduledTaskWithInitialDelay() {
    Logger.debug("scheduledTaskWithInitialDelay: current time: " + new Date());
}
```

- Cron
  - cron expressions could also be used with Scheduled annotations
  - it consists following fields

```
<second> <minute> <hour> <day-of-month> <month> <day-of-week> <year>
```

  - From these, <year> field is optional.

```
0 0 12 * * ? 2017
```

    - At 12:00 pm (noon) every day during the year 2017
- Cron Specials Characters In Expression
    - * (all) – it is used to specify that event should happen for every time unit. For example, “*” in the <minute> field – means “for every minute”
    - ? (any) – it is utilized in the <day-of-month> and <day-of -week> fields to denote the arbitrary value – neglect the field value. For example, if we want to fire a script at “5th of every month” irrespective of what the day of the week falls on that date, then we specify a “?” in the <day-of-week> field
    - – (range) – it is used to determine the value range. For example, “10-11” in <hour> field means “10th and 11th hours”
    - , (values) – it is used to specify multiple values. For example, “MON, WED, FRI” in <day-of-week> field means on the days “Monday, Wednesday, and Friday”
    - / (increments) – it is used to specify the incremental values. For example, a “5/15” in the <minute> field, means at “5, 20, 35 and 50 minutes of an hour”
    - # – it is used to specify the “N-th” occurrence of a weekday of the month, for example, “3rd Friday of the month” can be indicated as “6#3“