Rabbit MQ

Rabbit MQ är ett lightweight bibliotek som använder sig av amqp protokollet för att skicka meddelande på definerade köer.

Vill man inte definera sin egna kö, kan man skicka det på en generisk kö, som alla kan koppla upp sig på.

Meddelandet man vill skicka bör byggas som en egen klass, likt en entity men utan annotationer. Den innhåller variablerna och eventuellt en @JsonProperty annotering i konstruktorns parameter scope.

Sender klassen måste annoteras @Service följt av

```
@Autowired
private RabbitMessagingTemplate rabbitMessagingTemplate;

@Autowired
private MappingJackson2MessageConverter converter;
```

Därefter är det "bara" att skapa metoder som skickar ut data till kanalen.

Mottagnings klassen annoteras @Service och minst en metod ska annoteras med @RabbitListener. Nu kan metoden lyssna på en kö som man definerar i sin konfiguration.

Service klassen måste anropas med @Autowired av en klass som är annoterad med @EnableRabbit. Denna klass måste ha minst två @Bean annoteringar med standard metoder för att tolka meddelanden.

För att hjälpa till med att minska belastningen för lyssnarna kan man använda sig av Routing, Bindings och Exchanges.

Man börjar med att välja typ av Exchange. Direct, Fanout, Topic och Header.

Direct använder sig av en routing key som är deklarerad i binding key. På detta sätt routas ens meddelane till rätt kö och på så sätt kan man undvika att ett meddelande hamnar i fel kö.

Fanout bryr sig inte om eventuella routing keys. Alla kanaler kommer lyssna på detta meddelane.

Topic routar sina meddelane till de köer som matchar helt eller delvis en routing key. Man kan ha flera delar ev en routing key med, sepererade med en punkt.

Header är likadan som Topic men man skriver in routing key i headern men det man skriver in kan även matcha olika kriterier i en kö.

För att integrera JMS meningsfullt i projektet med minsta möjliga jobb, gör ett enkel extra SpringBoot projekt som lyssnar av på en kanal, som tar emot t.ex ett namn på den som loggar in, och sen skriver in en timestamp och skickar det till en databas.

Det behövs således inget större värde i säkerhet, meningen är att ni ska ha en humm om vad JMS är och hur det används. I verkliga livet hade man fått bygga mer säkerhet i det men till projektet är tidsgränsen rätt snäv som den är.