Spring, JavaConfig (i.e @Configuration) & JMS by example

Posted on November 3, 2015 by Arulkumaran Kumaraswamipillai

Step 1: Define the messaging system connectivity, destination (e.g. topic or queue) and message selector properties via a YAML or .properties file. Here is an example of **application.yml** file.

```
1
2  ##############################
3  # MQ Settings                #
4  ##############################
5  spring:
6    mq:
7      hostname: my-mq-host
8      channel: mychannel
9      port: 2500
10     queue.manager: GHTFDS
11     environment: EN7
12
13   # MQ SSL settings -- uncomment the below secti
14     # ssl:
15       # enabled:
16       # cipher.suite:
17       # keyStore:
18       # keyStorePassword:
19
```

600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs

open all | close all

- Ice Breaker Interview
- Core Java Interview C
- JEE Interview Q&A (3
- Pressed for time? Jav
- SQL, XML, UML, JSC
- Hadoop & BigData Int
- Java Architecture Inte
- Scala Interview Q&As
- Spring, Hibernate, & M
  - Spring (18)
    - Spring boot (4)
    - Spring IO (1)
    - Spring JavaConf
      - 10: Spring, Ja
      - Spring, JavaC
      - Spring, JavaC
      - Spring, JavaC
    - 01: ♥♦ 13 Spring
    - 01b: ♦ 13 Spring
    - 02: ▶ Spring DII
    - 03: ♥♦ Spring DI
    - 04 ♦ 17 Spring b

```
  # MQ misc settings, if not set, will set with
    # transport.type: 1
    # client.id: YOUR_CLIENT_NAME
    # delivery.mode: 2

##############################################
# topic selector                             #
##############################################
myapp:
  app:
    title: my app
    systemname: send-prices
    PriceSubscriptionName: price_event

    selector:
      PriceEventTopic: TOPIC/PRICE
      PriceEventMessageType: event
      PriceEventOriginator: PRICE
      PriceEventCategory: business/price

```

Step 2: MQTopicConnectionFactory The Java based Spring configuration defining the connection factory.

```
import javax.jms.ConnectionFactory;
import javax.jms.Destination;
import javax.jms.JMSException;
import javax.jms.Session;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotat
import org.springframework.beans.factory.annotat
import org.springframework.context.annotation.Be
import org.springframework.context.annotation.Co
import org.springframework.core.env.Environment;
import org.springframework.jdbc.core.JdbcTemplat
import org.springframework.jms.core.JmsTemplate;
import org.springframework.jms.listener.DefaultM

//......

@Configuration
public class AppConfig {

   private static final Logger LOG = LoggerFacto
   public static final String DEFAULT_MQ_CLIENT_
   public static final Boolean DEFAULT_MQ_SSL_EN

   @Autowired
   private Environment environment;

   //.....

   @Bean
   public ConnectionFactory connectionFactory()
        MQTopicConnectionFactory connectionFacto

        connectionFactory.setTransportType(envir
                DEFAULT_MQ_TRANSPORT_TYPE));
        connectionFactory.setHostName(environmen
        connectionFactory.setChannel(environment
```


- 05: ♦ 9 Spring B
- 06: ♥ Debugging
- 07: Debugging S
- Spring loading p
- Hibernate (13)
- AngularJS (2)
- Git & SVN (6)
- JMeter (2)
- JSF (2)
- Maven (3)
- Testing & Profiling/Sa
- Other Interview Q&A f
- ▶ Free Java Interviev

16 Technical Key Areas

open all | close all

- Best Practice (6)
- Coding (26)
- Concurrency (6)
- Design Concepts (7)
- Design Patterns (11)
- Exception Handling (3
- Java Debugging (21)
- Judging Experience Ir
- Low Latency (7)
- Memory Managemen
- Performance (13)
- QoS (8)
- Scalability (4)
- SDLC (6)
- Security (13)
- Transaction Managen

80+ step by step Java Tutorials

```
39          connectionFactory.setPort(environment.ge
40          connectionFactory.setQueueManager(enviro
41          connectionFactory.setClientID(environmen
42                  + environment.getRequiredPropert
43
44          LOG.debug(
45                  "Creating MQTopicConnectionFacto
46                  connectionFactory.getTransportTy
47                  connectionFactory.getPort(), con
48
49          if (environment.getProperty("spring.mq.s
50              connectionFactory.setSSLCipherSuite(
51              System.setProperty("javax.net.ssl.ke
52              System.setProperty("javax.net.ssl.ke
53                      environment.getRequiredPrope
54
55              LOG.info("SSL is enabled for JMS con
56              LOG.info("MQTopicConnectionFactory:
57
58          }
59
60          return connectionFactory;
61      }
62
63
64      //....
65
66  }
```

Step 3: Destination The Java based Spring configuration defining the JMS destination.

```
1
2   //......
3
4   @Configuration
5   public class AppConfig {
6
7       private static final Logger LOG = LoggerFacto
8       public static final String DEFAULT_MQ_CLIENT_
9       public static final Boolean DEFAULT_MQ_SSL_EN
10      public static final int DEFAULT_MQ_DELIVERY_M
11      public static final String DEFAULT_PRICE_EVEN
12
13      @Autowired
14      private Environment environment;
15
16      //.....
17
18       @Bean
19       public Destination destination() throws JMSE
20           MQTopic topic = new MQTopic();
21
22           topic.setBaseTopicName(environment.getPr
23           topic.setPersistence(environment.getProp
24           LOG.debug("Listening message on : topicN
25           return topic;
26       }
27
28      //....
29  }
```

open all | close all
Setting up Tutorial (6)
Tutorial - Diagnosis (2
Akka Tutorial (9)
Core Java Tutorials (2
Hadoop & Spark Tuto
JEE Tutorials (19)
Scala Tutorials (1)
Spring & HIbernate Tu
Tools Tutorials (19)
Other Tutorials (45)

Step 4: JMSTemplate to send messages and makes use of the "**connectionFactory()**" and "**destination()**" defined above.

```
1
2  //......
3
4  @Configuration
5  public class AppConfig {
6
7     //.....
8
9     @Autowired
10    private Environment environment;
11
12    //.....
13
14     @Bean
15     public JmsTemplate jmsTemplate() throws JMSE
16         JmsTemplate jmsTemplate = new JmsTemplat
17
18         jmsTemplate.setSessionTransacted(false);
19         jmsTemplate.setReceiveTimeout(5000);
20         jmsTemplate.setDefaultDestination(destin
21         return jmsTemplate;
22     }
23
24     //....
25 }
```

Step 5: MessageListenerContainer for the Spring JMS listener to use to receive messages.

```
1
2  //......
3
4  @Configuration
5  public class AppConfig {
6
7     private static final Logger LOG = LoggerFacto
8
9     //....
10    public static final String DEFAULT_PRICE_EVEN
11    public static final String DEFAULT_PRICE_EVEN
12    public static final String DEFAULT_PRICE_EVEN
13    public static final String DEFAULT_PRICE_EVEN
14
15    @Autowired
16    private Environment environment;
17
18     @Bean
19     @Qualifier("price_event")
20     public DefaultMessageListenerContainer messa
21
22         DefaultMessageListenerContainer containe
23
24         container.setSubscriptionDurable(true);
25         container.setDurableSubscriptionName(env
```

```
26                  + environment.getRequiredPropert
27
28          // Single thread mode
29          container.setConcurrentConsumers(1);
30          container.setMaxConcurrentConsumers(1);
31
32          container.setConnectionFactory(connectio
33          container.setDestination(destination());
34
35          container.setSessionAcknowledgeMode(Sess
36          container.setSessionTransacted(true);
37
38          container.setMessageSelector(priceMessag
39
40          // message listener
41          container.setMessageListener(priceEventL
42
43          return container;
44      }
45
46      private String priceMessageSelector() {
47          String messageType = environment.getProp
48          String originator = environment.getPrope
49          String category = environment.getPropert
50          String messageEnv = environment.getRequi
51          String selector = String.format("message
52          return selector;
53      }
54
55      @Bean
56      public PriceEventListener priceEventListener
57          return new PriceEventListener();
58      }
59
60      //....
61  }
```

Step 6: JMS listener that is attached to the default container defined above.

```
1
2  @Component
3  public class PriceEventListener implements Messa
4
5      @Override
6      public void onMessage(Message message) {
7          LOG.info("PriceEvent Message received...
8          //.....
9      }
10 }
```

Popular Member Posts

♦ 11 Spring boot interview questions & answers

904 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

816 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

427 views

18 Java scenarios based interview Questions and Answers

407 views

♦ 7 Java debugging interview questions & answers

322 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

310 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

303 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

001B: ♦ Java architecture & design concepts interview questions & answers

209 views

Bio | Latest Posts

Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription

based site.**945+** paid members. join my LinkedIn Group. **Reviews**

About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via Amazon.com in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site.**945+** paid members. join my LinkedIn Group. **Reviews**

‹ Integration Unit testing Spring Transaction Management

Convert Arrays to Lists and lists to array with real life example(s) ›

Posted in member-paid, Spring JavaConfig

Empowers you to open more doors, and fast-track

Technical Know Hows

☀ Java generics in no time ☀ Top 6 tips to transforming your thinking from OOP to FP ☀ How does a HashMap internally work? What is a hashing function?
☀ 10+ Java String class interview Q&As ☀ Java auto un/boxing benefits & caveats ☀ Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect

Non-Technical Know Hows

☀ 6 Aspects that can motivate you to fast-track your career & go places ☀ Are you reinventing yourself as a Java developer? ☀ 8 tips to safeguard your Java career against offshoring ☀ My top 5 career mistakes

Prepare to succeed

☀ Turn readers of your Java CV go from “Blah blah” to “Wow”? ☀ How to prepare for Java job interviews? ☀ 16 Technical Key Areas ☀ How to choose from multiple Java job offers?

Select Category

© Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.

↑