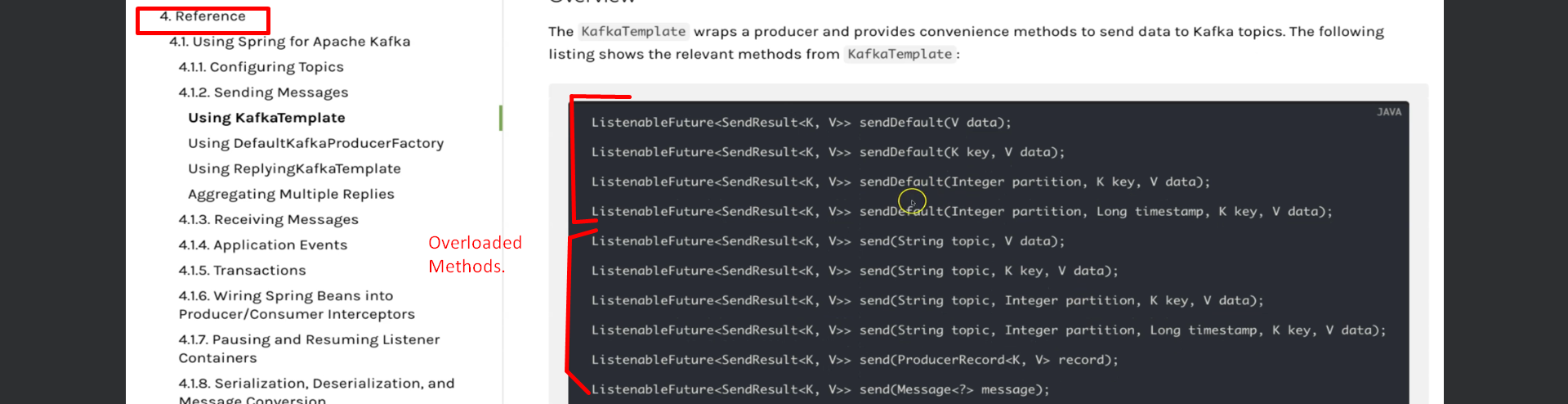
1. **Agenda**:
   1. 
2. **KafkaTemplate**:  
   
3. A class in Spring to produce msgs to a Kafka Topic.
4. A quick **analogy** is to think about **KafkaTemplate** as a **JdbcTemplate** for DB interactions.
5. Let’s see the Doc for **KafkaTemplate**.
6. 
7. **Let’s explore how KafkaTemplate works behind the scene**.
8. **RecordAccumulator**:
   1. Any record sent from the KafkaTemplate will not be sent immediately.
   2. **RecordAccumulator** accumulates the records and once buffer is full, the records are sent to the Kafka Topic.  
      The reason for this approach is to avoid the number of trips from the application to the Kafka Cluster and this eventually avoids the overhead of bombarding Kafka Cluster with numerous requests which helps in improving the overall performance of the system.
   3. **RecordBatch** is the representation of the topic partition combination.   
      If we have 3 partitions for a topic, then 3 RecordBatch.   
      Each and every **RecordBatch** has a batch size represented by the **batch.size property** and value by # of bytes.
   4. **buffer.memory** which represents the overall buffer memory for the **RecordAccumulator** and the value is # of bytes.
   5. **Now under which scenarios, the messages are sent to Kafka Topic?**
      1. Once batch is full, messages are sent.
      2. If batch not full, Producer will not wait for long.   
         If RecordAccumulator meets the **linger.ms** value, then the records are sent.
   6. These are the different layers that all the records go through before getting published.
      1. batch.size, buffer.memory, linger.ms
9. Next is to learn what is needed to configure a **KafkaTemplate**.
10. **What configuration values need to be made**.
11. **How to configure those values?**  
    