# How to start Spring Kafka Application with Spring Boot

The **Spring Apache Kafka** (spring-kafka) provides a high-level abstraction for Kafka-based messaging solutions. And in the previous post, We had developed a **Spring Kafka Application** with the **auto-configuration** supported by SpringBoot (from version **1.5**). But when we need explicitly configure **Kafka factories** (**Kafka Producer** and **Kafka Consumer**) for development, how to do it? So in the tutorial, **JavaSampleApproach** will introduce an alternative solution by manually configure **Kafka factories** to build a **Spring Kafka Application**.

Related Articles:
– How to start Apache Kafka
– How to start Spring Apache Kafka Application with SpringBoot Auto-Configuration
– How to use Spring Kafka JsonSerializer(JsonDeserializer) to produce/consume Java Object messages
– How to create Spring RabbitMQ Producer/Consumer application with SpringBoot

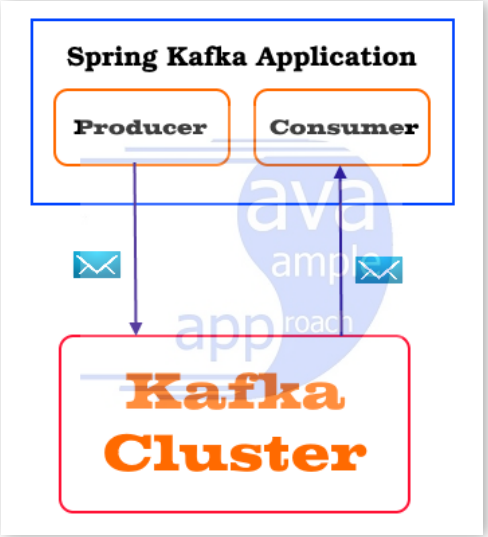
**Contents** [hide]



## I. Technologies

– Java 8
– Maven build
– Spring Boot
– Spring Kafka
– Apache Kafka
– Spring Tool Suite editor

## II. Overview



We will explicitly implement a **ProducerFactory** and **ConsumerFactory** with customized properties:

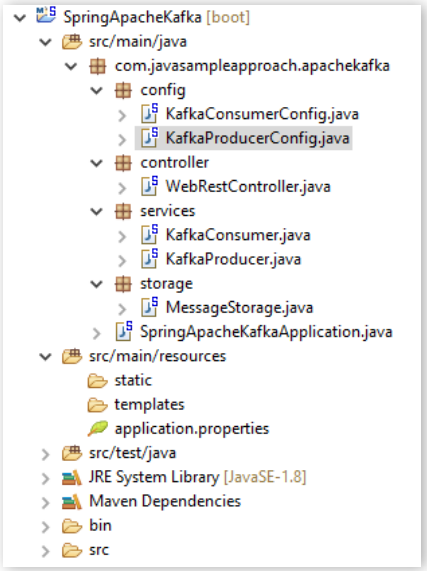
```
@Bean

public ProducerFactory<String, String> producerFactory() {
    Map<String, Object> configProps = new HashMap<>();
    ...
    return new DefaultKafkaProducerFactory<>(configProps);
}

@Bean
public ConsumerFactory<String, String> consumerFactory() {
    Map<String, Object> props = new HashMap<>();
    ...
    return new DefaultKafkaConsumerFactory<>(props);
}
```

Then use **ProducerFactory** to build **KafkaTemplate** and use **ConsumerFactory** to build **ConcurrentKafkaListenerContainerFactory** which will handle `@KafkaListener` later:

```
@Bean
public KafkaTemplate<String, String> kafkaTemplate() {
    return new KafkaTemplate<>(producerFactory());
}

...
@Bean
public ConcurrentKafkaListenerContainerFactory<String, String> kafkaListenerContainerFactory() {
    ConcurrentKafkaListenerContainerFactory<String, String> factory = new ConcurrentKafkaListenerContainerFactory<>();
    factory.setConsumerFactory(consumerFactory());
    return factory;
}
```

### III. Practice

We create a **SpringBoot** project with 2 main services: **KafkaProducer** and **KafkaConsumer** for sending and receiving messages from **Apache Kafka** cluster. And export 2 RestAPIs *{'/producer', '/consumer'}* for interaction.




**Step to do**:
– Create a SpringBoot project
– Create Kafa Factories (ProducerFactory & ConsumerFactory)
– Create Services (Producer and Consumer)
– Export some RestAPIs
– Deployment

#### 1. Create a SpringBoot project

Use **SpringToolSuite** to create a **SpringBoot** project, then add dependencies *{spring-kafka, spring-boot-starter-web}*:

```
<dependency>
  <groupId>org.springframework.kafka</groupId>
  <artifactId>spring-kafka</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

#### 2. Create Kafa Factories (ProducerFactory & ConsumerFactory)

Open **application.properties**, add kafka configuration:

```
jsa.kafka.bootstrap-servers=localhost:9092
jsa.kafka.consumer.group-id=jsa-group
jsa.kafka.topic=jsa-kafka-topic
```

– `jsa.kafka.bootstrap-servers` is used to indicate the **Kafka Cluster** address.
– `jsa.kafka.consumer.group-id` is used to indicate the **consumer-group-id**.

– `jsa.kafka.topic` is used to define a Kafka topic name to produce and receive messages.

### 2.1 Create ProducerFactory and KafkaTemplate

```
package com.javasampleapproach.apachekafka.config;

import java.util.HashMap;
import java.util.Map;

import org.apache.kafka.clients.producer.ProducerConfig;
import org.apache.kafka.common.serialization.StringSerializer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.core.DefaultKafkaProducerFactory;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.kafka.core.ProducerFactory;

@Configuration
public class KafkaProducerConfig {

  @Value("${jsa.kafka.bootstrap-servers}")
  private String bootstrapServer;

  @Bean
  public ProducerFactory<String, String> producerFactory() {
      Map<String, Object> configProps = new HashMap<>();
      configProps.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServer);
      configProps.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
      configProps.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
      return new DefaultKafkaProducerFactory<>(configProps);
  }

  @Bean
  public KafkaTemplate<String, String> kafkaTemplate() {
      return new KafkaTemplate<>(producerFactory());
  }
}
```

#### 2.2 Create ConsumerFactory and KafkaListenerContainerFactory

```
package com.javasampleapproach.apachekafka.config;

import java.util.HashMap;
import java.util.Map;

import org.apache.kafka.clients.consumer.ConsumerConfig;
import org.apache.kafka.common.serialization.StringDeserializer;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.kafka.annotation.EnableKafka;
import org.springframework.kafka.config.ConcurrentKafkaListenerContainerFactory;
import org.springframework.kafka.core.ConsumerFactory;
import org.springframework.kafka.core.DefaultKafkaConsumerFactory;

@EnableKafka
@Configuration
public class KafkaConsumerConfig {

  @Value("${jsa.kafka.bootstrap-servers}")
  private String bootstrapServer;

  @Value("${jsa.kafka.consumer.group-id}")
  private String groupId;

  @Bean
  public ConsumerFactory<String, String> consumerFactory() {
      Map<String, Object> props = new HashMap<>();
      props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG, bootstrapServer);
      props.put(ConsumerConfig.GROUP_ID_CONFIG, groupId);
      props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
      props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG, StringDeserializer.class);
      return new DefaultKafkaConsumerFactory<>(props);
  }

  @Bean
  public ConcurrentKafkaListenerContainerFactory<String, String> kafkaListenerContainerFactory() {
      ConcurrentKafkaListenerContainerFactory<String, String> factory = new ConcurrentKafkaListenerContainerFactory<>();
      factory.setConsumerFactory(consumerFactory());
      return factory;
  }
}
```

`@EnableKafka` is used to enable detection of `@KafkaListener` annotation.

#### 3. Create Services (Producer and Consumer)

– Create a **KafkaProducer** service:

```
package com.javasampleapproach.apachekafka.services;

import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.stereotype.Service;

@Service
public class KafkaProducer {
  private static final Logger log = LoggerFactory.getLogger(KafkaProducer.class);

  @Autowired
  private KafkaTemplate<String, String> kafkaTemplate;

  @Value("${jsa.kafka.topic}")
  String kafkaTopic = "jsa-test";

  public void send(String data) {
      log.info("sending data='{}'", data);

      kafkaTemplate.send(kafkaTopic, data);
  }
}
```

– Create a **KafkaConsumer** service:

```
package com.javasampleapproach.apachekafka.services;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.kafka.annotation.KafkaListener;
import org.springframework.stereotype.Component;

import com.javasampleapproach.apachekafka.storage.MessageStorage;

@Component
public class KafkaConsumer {
  private static final Logger log = LoggerFactory.getLogger(KafkaProducer.class);

  @Autowired
  MessageStorage storage;

  @KafkaListener(topics="${jsa.kafka.topic}")
    public void processMessage(String content) {
    log.info("received content = '{}'", content);
    storage.put(content);
    }
}
```

About **MessageStorage**, it is an additional implement to store Kafka-based messages after received. See details the implementation of **MessageStorage**:

```
package com.javasampleapproach.apachekafka.storage;
```

```
import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Component;

@Component
public class MessageStorage {

  private List<String> storage = new ArrayList<String>();

  public void put(String message){
    storage.add(message);
  }

  public String toString(){
    StringBuffer info = new StringBuffer();
    storage.forEach(msg->info.append(msg).append("<br/>"));
    return info.toString();
  }

  public void clear(){
    storage.clear();
  }
}
```

**4. Export some RestAPIs**

Create a Web Controller to export 2 RestAPIs *{'/producer', '/consumer'}*

```
package com.javasampleapproach.apachekafka.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.javasampleapproach.apachekafka.services.KafkaProducer;
import com.javasampleapproach.apachekafka.storage.MessageStorage;

@RestController
@RequestMapping(value="/jsa/kafka")
public class WebRestController {

  @Autowired
  KafkaProducer producer;

  @Autowired
  MessageStorage storage;

  @GetMapping(value="/producer")
  public String producer(@RequestParam("data")String data){
    producer.send(data);
```

```
    return "Done";
  }

  @GetMapping(value="/consumer")
  public String getAllRecievedMessage(){
    String messages = storage.toString();
    storage.clear();

    return messages;
  }
}
```

- `/producer` is used to send messages from browser to **KafkaProducer** service.
- `/consumer` is used to get all recieved messages that are buffered in **MessageStorage**.

#### 5. Deployment

Start **Apache Kafka Cluster:**

- Start a ZooKeeper:

```
C:\kafka_2.12-0.10.2.1>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
```

- Start the **Apache Kafka server**:

```
.\bin\windows\kafka-server-start.bat .\config\server.properties
```

>>> More details at: How to start Apache Kafka

**Build** and **Install** the SpringBoot project with commandlines: `mvn clean install` and `mvn spring-boot:run`

- Make a producer request: `http://localhost:8080/jsa/kafka/producer?data=Hello World`

-> Logs:

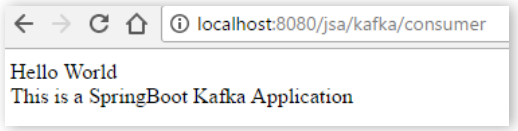
```
...

2017-06-08 13:49:47.111  INFO 12240 --- [io-8080-exec-10] c.j.apachekafka.services.KafkaProducer   : sending data='Hello World'
...
2017-06-08 13:49:47.248  INFO 12240 --- [ntainer#0-0-L-1] c.j.apachekafka.services.KafkaProducer   : received content = 'Hello World'
```

- Make another producer request: `http://localhost:8080/jsa/kafka/producer?data=This is a SpringBoot Kafka Application`

-> Logs:

```
2017-06-08 13:51:34.909  INFO 12240 --- [nio-8080-exec-7] c.j.apachekafka.services.KafkaProducer   : sending data='This is a SpringBoot
Kafka Application'
2017-06-08 13:51:34.913  INFO 12240 --- [ntainer#0-0-L-1] c.j.apachekafka.services.KafkaProducer   : received content = 'This is a Sprin
gBoot Kafka Application'
```

- Make a consumer request: `http://localhost:8080/jsa/kafka/consumer`, result:




### IV. Sourcecode

SpringApacheKafka



By grokonez | June 10, 2017.

**Related Posts**

- How to use Spring Kafka JsonSerializer (JsonDeserializer) to produce/consume Java Object messages
- How to start Spring Apache Kafka Application with SpringBoot Auto-Configuration
- How to start Apache Kafka
- RabbitMq Queue Durability and Persistent MessageDelivery | SpringBoot
- SpringBoot RabbitMq Exchange to Exchange Topology
- SpringBoot RabbitMq Headers Exchange
- SpringBoot RabbitMQ Topic Exchange
- Apache Artemis – How to produce/consume JMS messages with SpringBoot Artemis applications.
- Spring JMS ActiveMq – How to implement a runtime SpringBoot ActiveMQ JmsResponse application
- Spring Jms ActiveMQ – How to create a SpringBoot ActiveMQ Response Management application by @SendTo annotation

**Post Tags**

Apache Kafka | messaging system | spring boot | spring kafka

## 9 thoughts on “How to start Spring Kafka Application with Spring Boot”

**Venkat**

October 13, 2017 at 2:24 pm

Thanks , well explained. But i was facing an error while sending message. Its timed out after long waiting to send message. Any thoughts?

My propeties here:

```
jsa.kafka.bootstrap-servers=localhost:9092
jsa.kafka.consumer.group-id=test-consumer-group
jsa.kafka.topic=test-topic
```

```
2017-10-13 09:18:05.603 ERROR 48523 --- [nio-8080-exec-2] o.s.k.support.LoggingProducerListener    : Exception thrown when sending a message with key='null' and payload='t

org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 60000 ms.
```

**JavaSampleApproach**

October 14, 2017 at 4:12 am

Hi Venkat,

I had tried to re-produce your case. But can NOT in my environment.
Do you try the tutorial with your local enviroment or any cloud or docker?

The time-out exception may be related with your server Kafka setup. I had the experience when done with other tech-stacks.
May be, your **cursor** in **cmd** stops the excution of Kafka engine -> So please check it by do an `enter` keyboard on Kafka starting server **cmd**.

If having any difficult for setup and running, you can follow the video guide: https://youtu.be/A2FXupo7FLs

Regards,
JSA

---

**Talha**

December 6, 2017 at 4:42 pm

Hi ! I am not receiving any messages when i make request to http://localhost:8080/jsa/kafka/consumer what should i do ? I debug the code too but no success.

Thanks

---

**JavaSampleApproach**

December 7, 2017 at 9:20 am

Hello Talha,

You can follow 2 steps for working with the tutorial:
– Start Apache Kafka Cluster, follow at Deployment session.
More practice with Kafka startup, follow link: How to start Apache Kafka
– Then download sourcecode, build and run it.

More details, See guide video for practicing: https://youtu.be/A2FXupo7FLs

Regards,
JSA

---

**Joe Ann Midhun**

February 16, 2018 at 9:17 am

I am getting the following error while doing mvn spring-boot:run.Any help??

java.net.BindException: Address already in use: bind
at sun.nio.ch.Net.bind0(Native Method) ~[na:1.8.0_111]
at sun.nio.ch.Net.bind(Net.java:433) ~[na:1.8.0_111]
at sun.nio.ch.Net.bind(Net.java:425) ~[na:1.8.0_111]

```
at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:223) ~[na:1.8.0_111]
at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:74) ~[na:1.8.0_111]
at org.apache.tomcat.util.net.NioEndpoint.bind(NioEndpoint.java:210) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.tomcat.util.net.AbstractEndpoint.start(AbstractEndpoint.java:978) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.coyote.AbstractProtocol.start(AbstractProtocol.java:628) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.catalina.connector.Connector.startInternal(Connector.java:993) [tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:150) [tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.catalina.core.StandardService.addConnector(StandardService.java:225) [tomcat-embed-core-8.5.14.jar:8.5.14]
at
org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer.addPreviouslyRemovedConnectors(TomcatEmbeddedServletContainer.java:247
) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer.start(TomcatEmbeddedServletContainer.java:190) [spring-boot-
1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.startEmbeddedServletContainer(EmbeddedWebApplicationContext.java:297)
[spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.finishRefresh(EmbeddedWebApplicationContext.java:145) [spring-boot-
1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:545) [spring-context-4.3.8.RELEASE.jar:4.3.8.RELEASE]
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.refresh(EmbeddedWebApplicationContext.java:122) [spring-boot-
1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:737) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:370) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.run(SpringApplication.java:314) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.run(SpringApplication.java:1162) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.run(SpringApplication.java:1151) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at com.javasampleapproach.apachekafka.SpringBootApacheKafkaApplication.main(SpringBootApacheKafkaApplication.java:10) [classes/:na]
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:1.8.0_111]
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) ~[na:1.8.0_111]
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:1.8.0_111]
at java.lang.reflect.Method.invoke(Method.java:498) ~[na:1.8.0_111]
at org.springframework.boot.maven.AbstractRunMojo$LaunchRunner.run(AbstractRunMojo.java:527) [spring-boot-maven-plugin-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at java.lang.Thread.run(Thread.java:745) [na:1.8.0_111]

2018-02-16 14:45:25.381 ERROR 9564 — [ main] o.apache.catalina.core.StandardService : Failed to start connector [Connector[HTTP/1.1-8080]]

org.apache.catalina.LifecycleException: Failed to start component [Connector[HTTP/1.1-8080]]
at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:167) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.catalina.core.StandardService.addConnector(StandardService.java:225) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at
org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer.addPreviouslyRemovedConnectors(TomcatEmbeddedServletContainer.java:247
) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer.start(TomcatEmbeddedServletContainer.java:190) [spring-boot-
1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.startEmbeddedServletContainer(EmbeddedWebApplicationContext.java:297)
[spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.finishRefresh(EmbeddedWebApplicationContext.java:145) [spring-boot-
```

```
1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:545) [spring-context-4.3.8.RELEASE.jar:4.3.8.RELEASE]
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.refresh(EmbeddedWebApplicationContext.java:122) [spring-boot-
1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:737) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:370) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.run(SpringApplication.java:314) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.run(SpringApplication.java:1162) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at org.springframework.boot.SpringApplication.run(SpringApplication.java:1151) [spring-boot-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at com.javasampleapproach.apachekafka.SpringBootApacheKafkaApplication.main(SpringBootApacheKafkaApplication.java:10) [classes/:na]
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:1.8.0_111]
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62) ~[na:1.8.0_111]
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:1.8.0_111]
at java.lang.reflect.Method.invoke(Method.java:498) ~[na:1.8.0_111]
at org.springframework.boot.maven.AbstractRunMojo$LaunchRunner.run(AbstractRunMojo.java:527) [spring-boot-maven-plugin-1.5.3.RELEASE.jar:1.5.3.RELEASE]
at java.lang.Thread.run(Thread.java:745) [na:1.8.0_111]
Caused by: org.apache.catalina.LifecycleException: service.getName(): “Tomcat”; Protocol handler start failed
at org.apache.catalina.connector.Connector.startInternal(Connector.java:1000) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:150) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
… 19 common frames omitted
Caused by: java.net.BindException: Address already in use: bind
at sun.nio.ch.Net.bind0(Native Method) ~[na:1.8.0_111]
at sun.nio.ch.Net.bind(Net.java:433) ~[na:1.8.0_111]
at sun.nio.ch.Net.bind(Net.java:425) ~[na:1.8.0_111]
at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:223) ~[na:1.8.0_111]
at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:74) ~[na:1.8.0_111]
at org.apache.tomcat.util.net.NioEndpoint.bind(NioEndpoint.java:210) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.tomcat.util.net.AbstractEndpoint.start(AbstractEndpoint.java:978) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.coyote.AbstractProtocol.start(AbstractProtocol.java:628) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
at org.apache.catalina.connector.Connector.startInternal(Connector.java:993) ~[tomcat-embed-core-8.5.14.jar:8.5.14]
… 20 common frames omitted

2018-02-16 14:45:25.431 INFO 9564 — [ main] o.apache.catalina.core.StandardService : Stopping service Tomcat
2018-02-16 14:45:25.449 INFO 9564 — [ntainer#0-0-C-1] o.a.k.c.c.internals.AbstractCoordinator : Discovered coordinator D-113060917.wipro.com:9092 (id:
2147483647 rack: null) for group jsa-group.
2018-02-16 14:45:25.462 INFO 9564 — [ntainer#0-0-C-1] o.a.k.c.c.internals.ConsumerCoordinator : Revoking previously assigned partitions [] for group jsa-group
2018-02-16 14:45:25.464 INFO 9564 — [ main] utoConfigurationReportLoggingInitializer :

Error starting ApplicationContext. To display the auto-configuration report re-run your application with ‘debug’ enabled.
2018-02-16 14:45:25.469 INFO 9564 — [ntainer#0-0-C-1] o.s.k.l.KafkaMessageListenerContainer : partitions revoked:[]
2018-02-16 14:45:25.472 INFO 9564 — [ntainer#0-0-C-1] o.a.k.c.c.internals.AbstractCoordinator : (Re-)joining group jsa-group
2018-02-16 14:45:25.476 ERROR 9564 — [ main] o.s.b.d.LoggingFailureAnalysisReporter :

***************************
APPLICATION FAILED TO START
***************************
```

Description:

The Tomcat connector configured to listen on port 8080 failed to start. The port may already be in use or the connector may be misconfigured.

Action:

Verify the connector’s configuration, identify and stop any process that’s listening on port 8080, or configure this application to listen on another port.

2018-02-16 14:45:25.498 INFO 9564 — [ main] ationConfigEmbeddedWebApplicationContext : Closing
org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@4f055159: startup date [Fri Feb 16 14:45:22 IST 2018]; root of
context hierarchy
2018-02-16 14:45:25.503 INFO 9564 — [ main] o.s.c.support.DefaultLifecycleProcessor : Stopping beans in phase 0
2018-02-16 14:45:25.508 INFO 9564 — [ntainer#0-0-C-1] essageListenerContainer$ListenerConsumer : Consumer stopped
2018-02-16 14:45:25.508 INFO 9564 — [ main] o.s.j.e.a.AnnotationMBeanExporter : Unregistering JMX-exposed beans on shutdown
[WARNING]
java.lang.reflect.InvocationTargetException
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at org.springframework.boot.maven.AbstractRunMojo$LaunchRunner.run(AbstractRunMojo.java:527)
at java.lang.Thread.run(Thread.java:745)
Caused by: org.springframework.boot.context.embedded.tomcat.ConnectorStartFailedException: Connector configured to listen on port 8080 failed to start
at
org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer.checkThatConnectorsHaveStarted(TomcatEmbeddedServletContainer.java:219
)
at org.springframework.boot.context.embedded.tomcat.TomcatEmbeddedServletContainer.start(TomcatEmbeddedServletContainer.java:195)
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.startEmbeddedServletContainer(EmbeddedWebApplicationContext.java:297)
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.finishRefresh(EmbeddedWebApplicationContext.java:145)
at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.java:545)
at org.springframework.boot.context.embedded.EmbeddedWebApplicationContext.refresh(EmbeddedWebApplicationContext.java:122)
at org.springframework.boot.SpringApplication.refresh(SpringApplication.java:737)
at org.springframework.boot.SpringApplication.refreshContext(SpringApplication.java:370)
at org.springframework.boot.SpringApplication.run(SpringApplication.java:314)
at org.springframework.boot.SpringApplication.run(SpringApplication.java:1162)
at org.springframework.boot.SpringApplication.run(SpringApplication.java:1151)
at com.javasampleapproach.apachekafka.SpringBootApacheKafkaApplication.main(SpringBootApacheKafkaApplication.java:10)
… 6 more
[INFO] ————————————————————————
[INFO] BUILD FAILURE
[INFO] ————————————————————————
[INFO] Total time: 8.289 s
[INFO] Finished at: 2018-02-16T14:45:25+05:30
[INFO] Final Memory: 36M/274M
[INFO] ————————————————————————
[ERROR] Failed to execute goal org.springframework.boot:spring-boot-maven-plugin:1.5.3.RELEASE:run (default-cli) on project SpringBootApacheKafka: An exception

occurred while running. null: InvocationTargetException: Connector configured to listen on port 8080 failed to start -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoExecutionException

---

**Joe Ann Midhun**

February 16, 2018 at 9:25 am

I sorted out!.

---

**Andy Harris**

March 23, 2018 at 7:25 am

The error messages are clearly telling you that you have a port conflict as something else on your machine is using the same port (8080). Stop that process and try again.

---

**Andy Harris**

March 23, 2018 at 7:24 am

Awesome quick tutorial and it worked perfectly. Huge thanks for that.

---

**dhakshina**

April 26, 2018 at 3:07 pm

wonderful tutorial!!!!!:)

FOLLOW US

ABOUT US

We are passionate engineers in software development by Java Technology & Spring Framework. We believe that creating little good thing with specific orientation everyday can make great influence on the world someday.