# 在Spring中使用Kafka：Producer篇

在某些情况下，我们可能会在Spring中将一些WEB上的信息发送到Kafka中，这时候我们就需要在Spring中编写Producer相关的代码了；不过高兴的是，Spring本身提供了操作Kafka的相关类库，我们可以直接通过xml文件配置然后直接在后端的代码中使用Kafka，非常地方便。本文将介绍如果在Spring中将消息发送到Kafka。在这之前，请将下面的依赖加入到你的 pom.xml 文件中：

```
<dependency>
   <groupId>org.springframework.integration</groupId>
   <artifactId>spring-integration-core</artifactId>
   <version>4.1.0.RELEASE</version>
</dependency>

<dependency>
   <groupId>org.springframework.integration</groupId>
   <artifactId>spring-integration-kafka</artifactId>
   <version>1.0.0.M2</version>
</dependency>
```


在Spring中将消息发送到Kafka需要我们定义一个配置文件，Spring为我们提供了Outbound Channel Adapter，其主要将消息从Spring框架发送到Kafka，我们需要在xml文件中配置 int-kafka:outbound-channel-adapter 标签，这样可以使得Spring可以抽取到消息的Key、目标主题、分区等信息。本文将介绍如何把用户注册的信息发送到Kafka。这里用到的配置文件如下：

```
<int:publish-subscribe-channel id="inputToKafka"/>

<int-kafka:outbound-channel-adapter
      kafka-producer-context-ref="kafkaProducerContext" auto-startup="true"
      channel="inputToKafka" order="1">
</int-kafka:outbound-channel-adapter>
```

```
<int-kafka:producer-context id="kafkaProducerContext"
                    producer-properties="producerProperties">
   <int-kafka:producer-configurations>
      <int-kafka:producer-configuration
           broker-list="www.iteblog.com:9092" key-class-type="java.lang.String"
           value-class-type="com.iteblog.dao.User"
           topic="user" compression-codec="none"/>
   </int-kafka:producer-configurations>
</int-kafka:producer-context>

<bean id="producerProperties"
     class="org.springframework.beans.factory.config.PropertiesFactoryBean">
   <property name="properties">
      <props>
         <prop key="topic.metadata.refresh.interval.ms">3600000</prop>
         <prop key="message.send.max.retries">5</prop>
         <prop key="send.buffer.bytes">5242880</prop>
      </props>
   </property>
</bean>
```

int:publish-subscribe-channel 标签定义了消息发送的通道；int-kafka:producer-context 标签里面可以定义producer 的context，在里面我们可以设置broker的地址，key和value的类型，需要发送消息的目标Topic等相关属性；我们可以在自定义的bean中定义Kafka Producer的相关属性，本文对应的是producerProperties，这里我们定义了 topic.metadata.refresh.interval.ms 等相关属性，更多的属性可以参见Kafka的官方文档。然后可以在 int-int-kafka:producer-context 标签通过 producer-properties 来引用。配置文件设置好之后，我们就可以编写Java代码了：

```
package com.iteblog.controller;

import com.iteblog.dao.User;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.messaging.MessageChannel;
import org.springframework.messaging.support.MessageBuilder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
```

```
/**
 * Created by https://www.iteblog.com on 2016/10/21.
 */
@Controller
public class IteblogController {

    @Autowired
    @Qualifier("inputToKafka")
    MessageChannel channel;

    private static final Log logger = LogFactory.getLog(IteblogController.class);

    @RequestMapping(method = RequestMethod.POST, value = "/register")
    @ResponseBody
    public User registerJson(User user) {
        logger.warn("User: " + user);
        channel.send(MessageBuilder.withPayload(user).build());
        return user;
    }
}
```

withPayload 接受的就是消息的内容；MessageChannel
就是发送消息的通道，所有的消息都是通过这个通道发送到Kafka；User 类的代码如下：

```
package com.iteblog.dao;

/**
 * Created by https://www.iteblog.com on 2016/10/21.
 */
public class User {
    private String userName;
    private String email;

    public User() {
    }

    public User(String userName, String email) {
        this.userName = userName;
        this.email = email;
    }

    public String getUserName() {
```

```
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "User{" +
                "userName='" + userName + '₩'' +
                ", email='" + email + '₩'' +
                '}';
    }
}
```

在发送消息的时候，我们还可以设置消息的Key，以及需要发送的Topic，如下：

```
channel.send(MessageBuilder.withPayload(user)
                .setHeader("topic", "user")
                .setHeader("messageKey", user.getUserName()).build());
```

这里通过设置topic、messageKey等Header信息，来分别制定目标主题的名称（当然，如果只有一个主题我们不需要手动指定，Spring会自定选择配置文件里面指定的Topic；如果有多个需要手动指定）和key的值。我们还可以设置消息的Key和Value编码格式，如下：

```
<bean id="userEncoder"
     class="org.springframework.integration.kafka.serializer.avro.AvroReflectDatumBackedKafkaEncoder">
    <constructor-arg value="com.iteblog.dao.User"></constructor>
</bean>
```

```
<bean id="keyEncoder"
    class="org.springframework.integration.kafka.serializer.avro.AvroReflectDatumBackedKafkaEncoder">
  <constructor-arg value="java.lang.String"></constructor>
</bean>
```

然后在配置文件里面配置：

```
<int-kafka:producer-context id="kafkaProducerContext"
                    producer-properties="producerProperties">
  <int-kafka:producer-configurations>
    <int-kafka:producer-configuration
        broker-list="www.iteblog.com:9092" key-class-type="java.lang.String"
        value-class-type="com.iteblog.dao.User"
        value-encoder="userEncoder"
        key-encoder="keyEncoder"
        topic="user" compression-codec="none"/>
  </int-kafka:producer-configurations>
</int-kafka:producer-context>
```

如果我们得Topic有多个分区，我们还可以指定每条消息的分区ID计算规则，如下：

```
<bean id="iteblogPartitioner" class="org.springframework.integration.kafka.support.DefaultPartitioner"/>
```

我们使用了Spring默认分区类，也就是计算Key的hashCode再对分区数求模(Utils.abs(key.hashCode()) % numPartitions)，然后我们可以在 int-kafka:producer-configuration 里面加上以下配置

```
partitioner="iteblogPartitioner"
```

到这里，我们就可以在Tomcat中启动上面的Web工程，然后访问https://www.iteblog.com/register并传入 userName 和 email 参数即可将消息发送到Kafka，完整的工程目录结构如下：

.

```
├── pom.xml
└── src
    └── main
        ├── java
        │   └── com
        │       └── iteblog
        │           ├── controller
        │           │   └── IteblogController.java
        │           └── dao
        │               └── User.java
        ├── resources
        │   ├── applicationContext.xml
        │   ├── kafka-outbound-context.xml
        │   └── log4j.properties
        └── webapp
            ├── index.jsp
            ├── register.html
            └── WEB-INF
                └── web.xml

10 directories, 9 files
```

我将在后面的文章介绍如何在Spring中接收Kafka的消息，敬请关注。