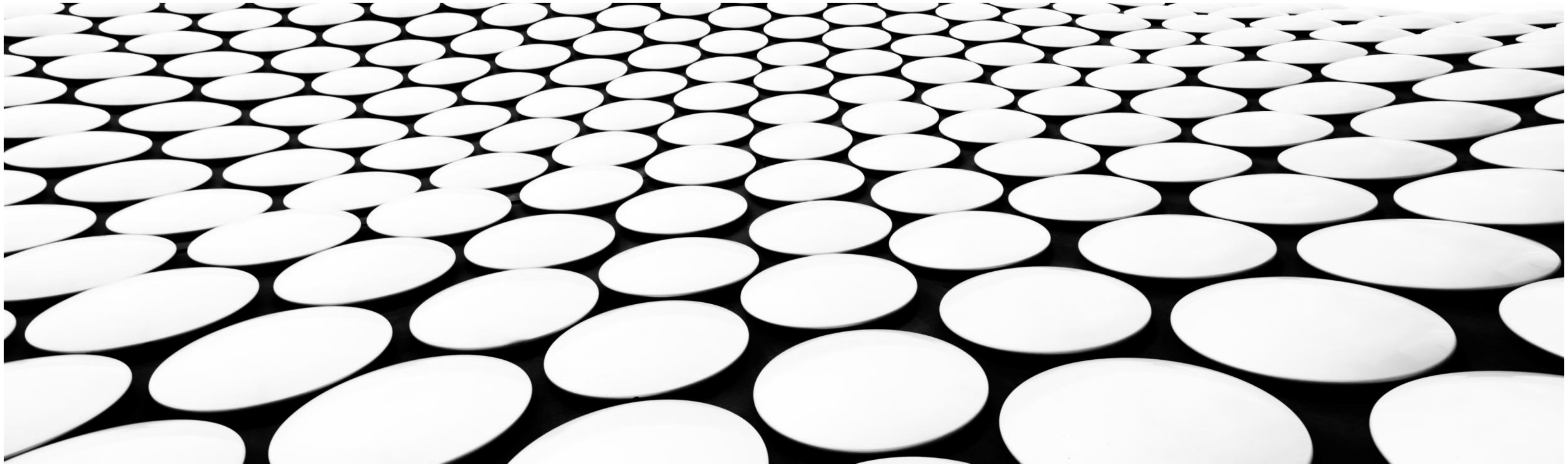
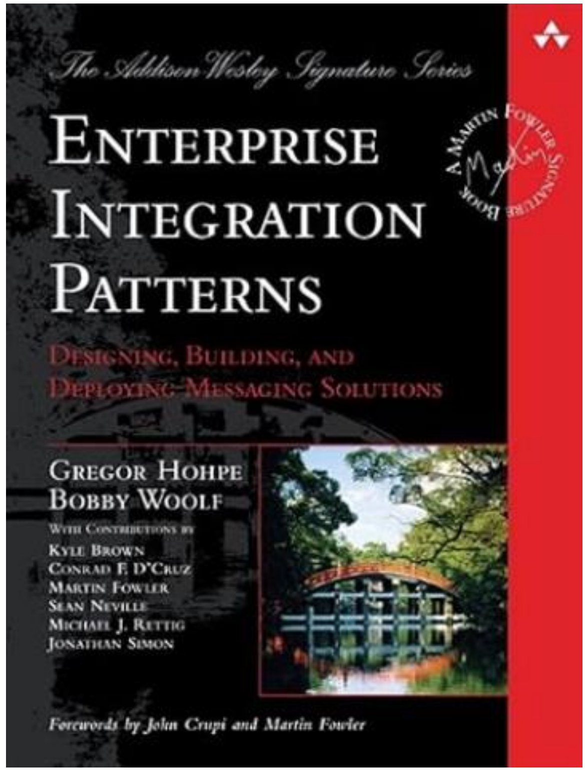


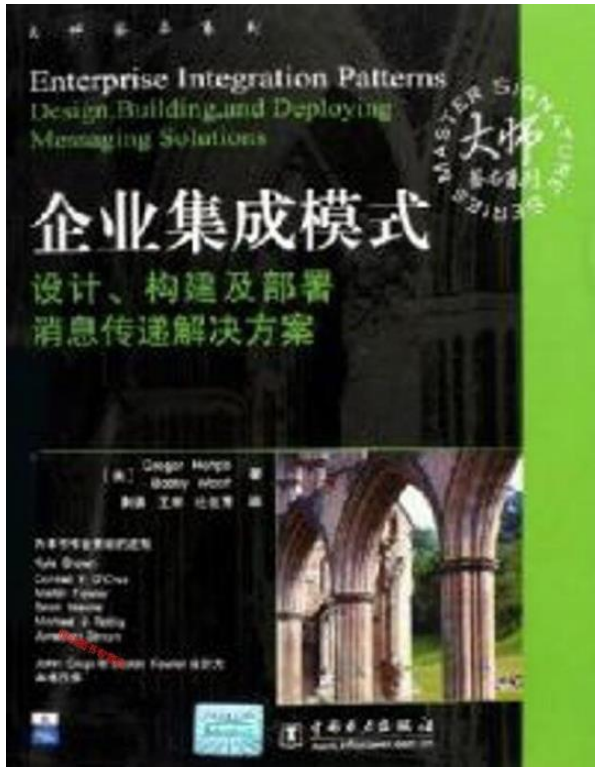
服务端开发-Spring Integration

陶召胜

EIP（Enterprise Integration Patterns，企业集成模式）

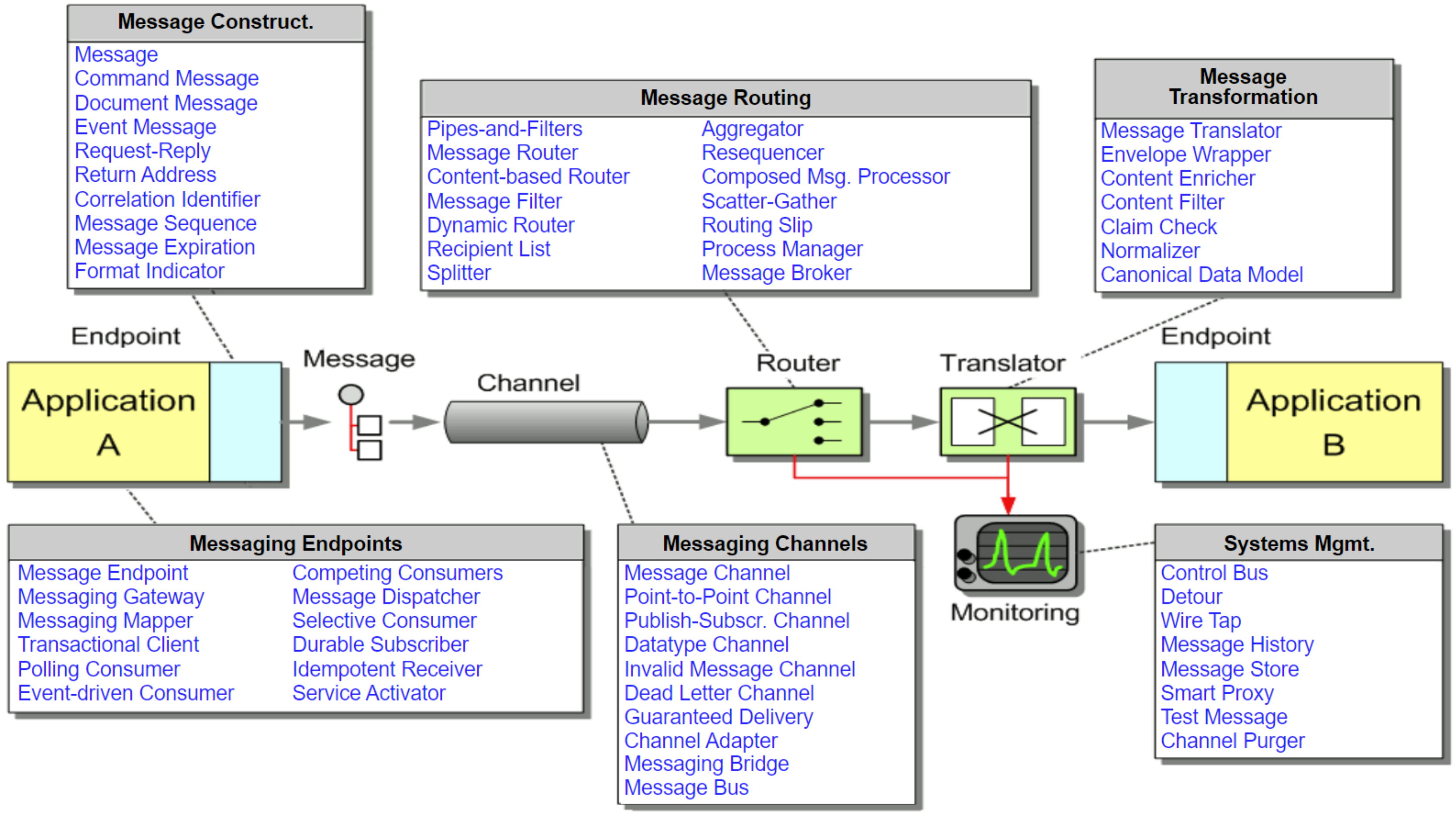
- Enterprise Integration Pattern - 组成简介：https://www.cnblogs.com/loveis715/p/5185332.html


The Addison-Wesley Signature Series
Enterprise Integration Patterns
Designing, Building, and Deploying Messaging Solutions
A Martin Fowler Signature Book
Gregor Hohpe
Bobby Woolf
With Contributions by
Kyle Brown
Conrad F. D'Cruz
Martin Fowler
Sean Neville
Michael J. Rettig
Jonathan Simon
Forewords by John Crupi and Martin Fowler


Enterprise Integration Patterns
Design, Building and Deploying Messaging Solutions
企业集成模式
设计、构建及部署消息传递解决方案
大师

Integration Pattern Language

- https://www.enterpriseintegrationpatterns.com/patterns/messaging/

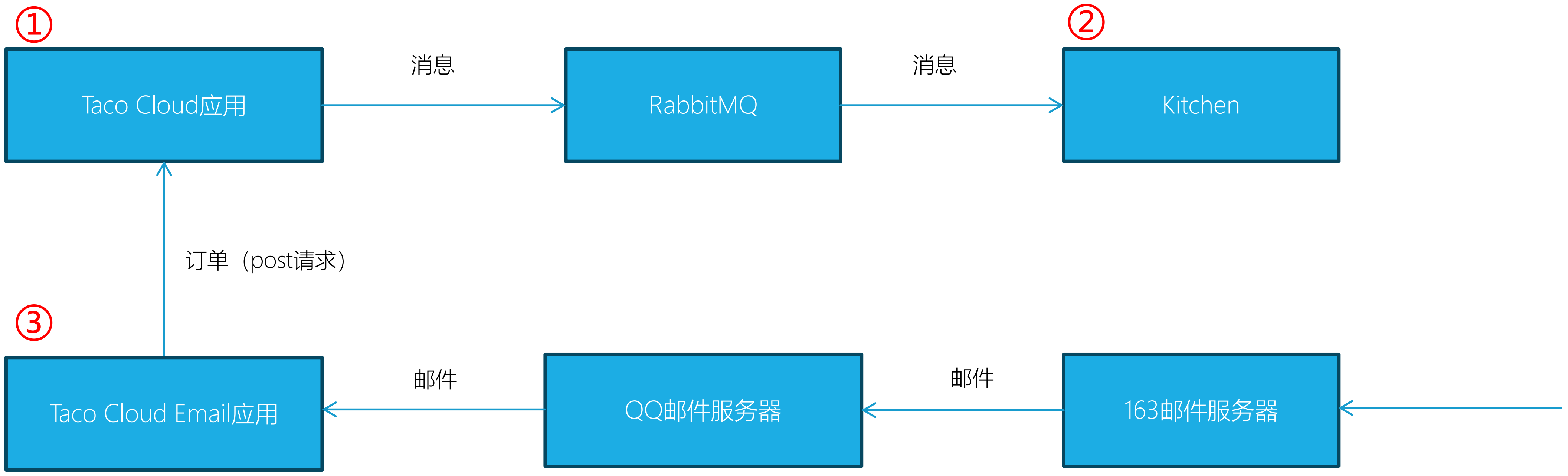


spring integration

- https://spring.io/projects/spring-integration

内容

本节课目标



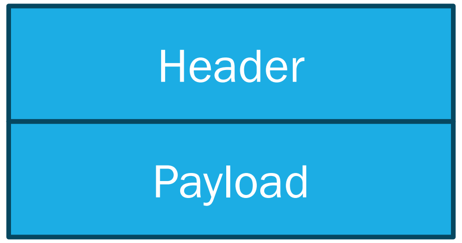
An IMAP email inbound channel adapter 轮询邮件服务器，触发订单生成

一个简单的集成流

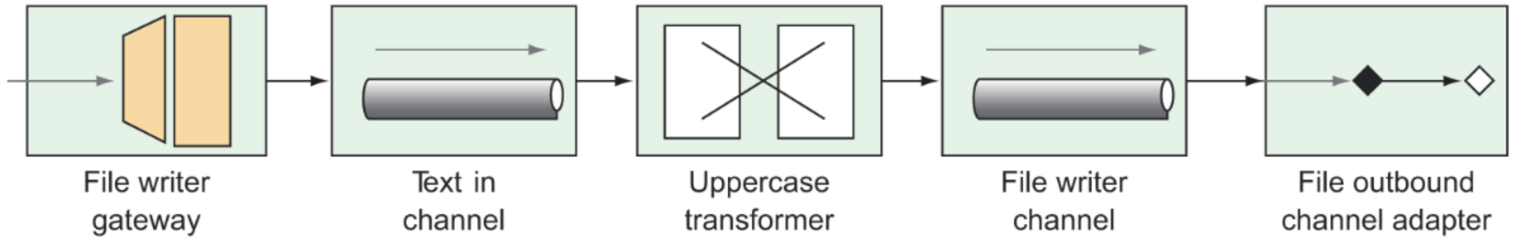
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-integration</artifactId>
</dependency>

<dependency>
  <groupId>org.springframework.integration</groupId>
  <artifactId>spring-integration-file</artifactId>
</dependency>
```

- 集成流配置
  - ✓ XML配置
  - ✓ Java配置
  - ✓ 使用DSL的Java配置

消息



集成流（integration flow)



内容

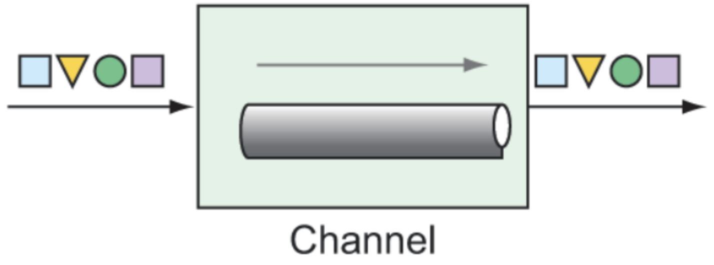
集成流的组件

- Channels（通道） —Pass messages from one element to another.
- Filters（过滤器） —Conditionally allow messages to pass through the flow based on some criteria.
- Transformers（转换器） —Change message values and/or convert message payloads from one type to another.
- Routers（路由器） —Direct messages to one of several channels, typically based on message headers.
- Splitters（切分器） —Split incoming messages into two or more messages, each sent to different channels.
- Aggregators（聚合器） —The opposite of splitters, combining multiple messages coming in from separate channels into a single message.
- Service activators（服务激活器） —Hand a message off to some Java method for processing, and then publish the return value on an output channel.
- Channel adapters（通道适配器） —Connect a channel to some external system or transport. Can either accept input or write to the external system.
- Gateways（网关） —Pass data into an integration flow via an interface.

消息通道（Message channels）

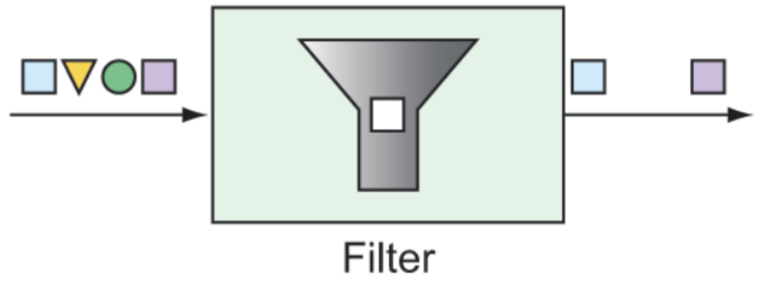
- PublishSubscribeChannel
- QueueChannel
- PriorityChannel
- RendezvousChannel
- DirectChannel（缺省）
- ExecutorChannel
- FluxMessageChannel

```
@Bean
public MessageChannel orderChannel() {
  return new PublishSubscribeChannel();
}
```



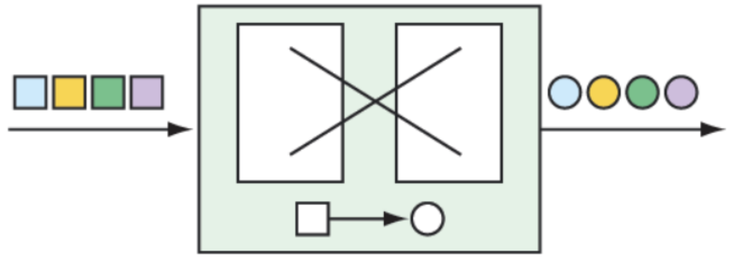
过滤器（Filters）

```
@Filter(inputChannel="numberChannel",
        outputChannel="evenNumberChannel")
public boolean evenNumberFilter(Integer number) {
  return number % 2 == 0;
}
```



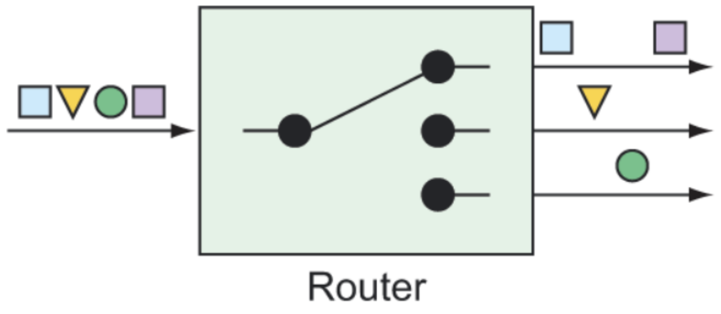
转换器（Transformers）

```
@Bean
@Transformer(inputChannel="numberChannel",
          outputChannel="romanNumberChannel")
public GenericTransformer<Integer, String> romanNumTransformer()
{
  return RomanNumbers::toRoman;
}
```



路由器（Routers）

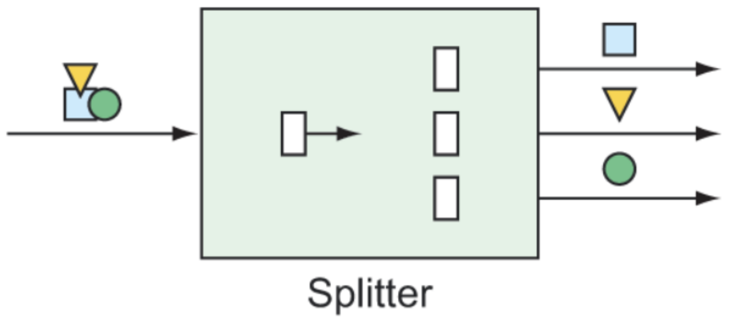
```
@Bean
@Router(inputChannel="numberChannel")
public AbstractMessageRouter evenOddRouter() {
  return new AbstractMessageRouter() {
    @Override
    protected Collection<MessageChannel>
              determineTargetChannels(Message<?> message) {
      Integer number = (Integer) message.getPayload();
      if (number % 2 == 0) {
        return Collections.singleton(evenChannel());
      }
      return Collections.singleton(oddChannel());
    }
  };
}

@Bean
public MessageChannel evenChannel() {
  return new DirectChannel();
}

@Bean
public MessageChannel oddChannel() {
  return new DirectChannel();
}
```



切分器（Splitters）

```
public class OrderSplitter {
  public Collection<Object> splitOrderIntoParts(PurchaseOrder po) {
    ArrayList<Object> parts = new ArrayList<>();
    parts.add(po.getBillingInfo());
    parts.add(po.getLineItems());
    return parts;
  }
}

@Bean
@Splitter(inputChannel="poChannel",
          outputChannel="splitOrderChannel")
public OrderSplitter orderSplitter() {
  return new OrderSplitter();
}

@Bean
@Router(inputChannel="splitOrderChannel")
public MessageRouter splitOrderRouter() {
  PayloadTypeRouter router = new PayloadTypeRouter();
  router.setChannelMapping(
      BillingInfo.class.getName(), "billingInfoChannel");
  router.setChannelMapping(
      List.class.getName(), "lineItemsChannel");
  return router;
}
```

```
@Splitter(inputChannel="lineItemsChannel", outputChannel="lineItemChannel")
public List<LineItem> lineItemSplitter(List<LineItem> lineItems) {
  return lineItems;
}
```



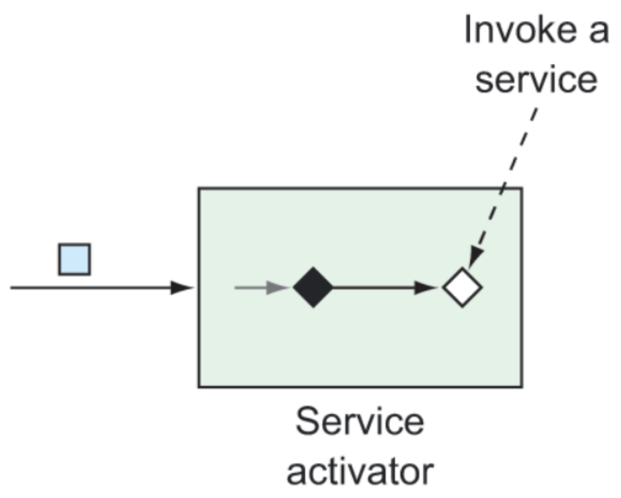
服务激活器（Service activators）

- MessageHandler实现

```
@Bean
@ServiceActivator(inputChannel="someChannel")
public MessageHandler sysoutHandler() {
  return message -> {
    System.out.println("Message payload:  " + message.getPayload());
  };
}
```

- GenericHandler实现

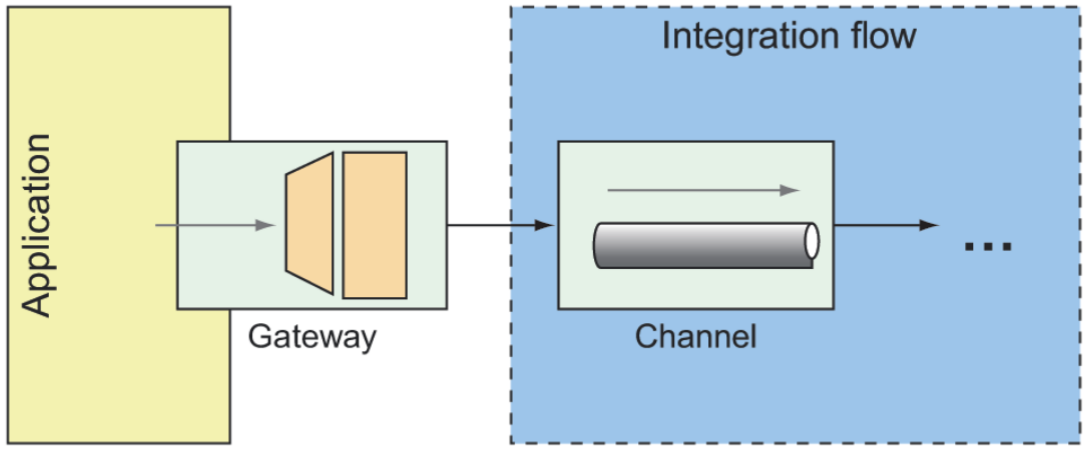
```
@Bean
@ServiceActivator(inputChannel="orderChannel",
                  outputChannel="completeOrder")
public GenericHandler<TacoOrder> orderHandler(
                              OrderRepository orderRepo) {
  return (payload, headers) -> {
    return orderRepo.save(payload);
  };
}
```



网关（Gateways）

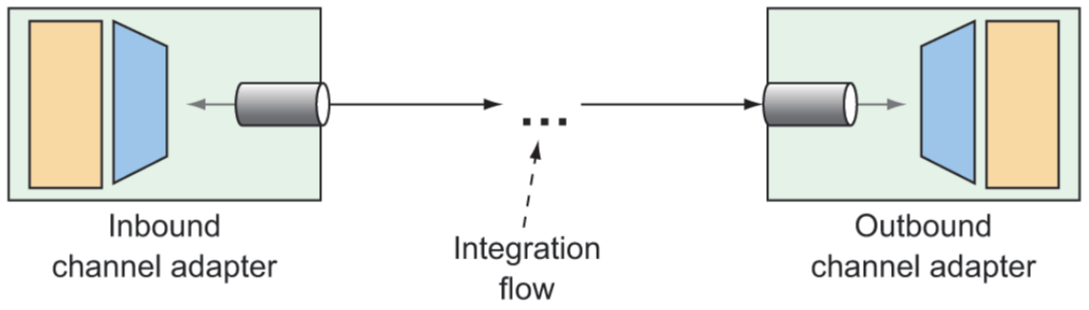
- 双向网关

```
import org.springframework.integration.annotation.MessagingGateway;
import org.springframework.stereotype.Component;

@Component
@MessagingGateway(defaultRequestChannel="inChannel",
                  defaultReplyChannel="outChannel")
public interface UpperCaseGateway {
  String uppercase(String in);
}

@Bean
public IntegrationFlow uppercaseFlow() {
  return IntegrationFlows
    .from("inChannel")
    .<String, String> transform(s -> s.toUpperCase())
    .channel("outChannel")
    .get();
}
```



通道适配器（Channel adapters）

```
@Bean
@InboundChannelAdapter(
    poller=@Poller(fixedRate="1000"), channel="numberChannel")
public MessageSource<Integer> numberSource(AtomicInteger source) {
  return () -> {
    return new GenericMessage<>(source.getAndIncrement());
  };
}

@Bean
@InboundChannelAdapter(channel="file-channel",
                                          poller=@Poller(fixedDelay="1000"))
public MessageSource<File> fileReadingMessageSource() {
  FileReadingMessageSource sourceReader = new FileReadingMessageSource();
  sourceReader.setDirectory(new File(INPUT_DIR));
  sourceReader.setFilter(new SimplePatternFileListFilter(FILE_PATTERN));
  return sourceReader;
}
```

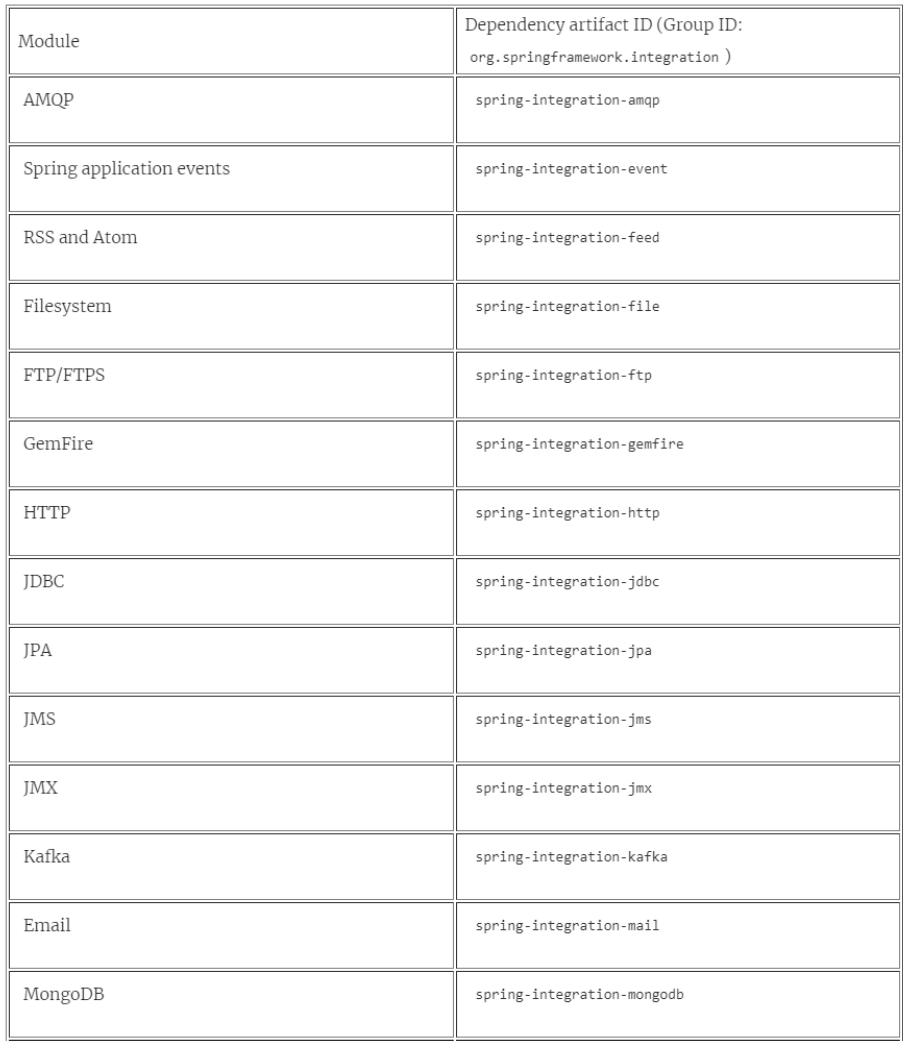
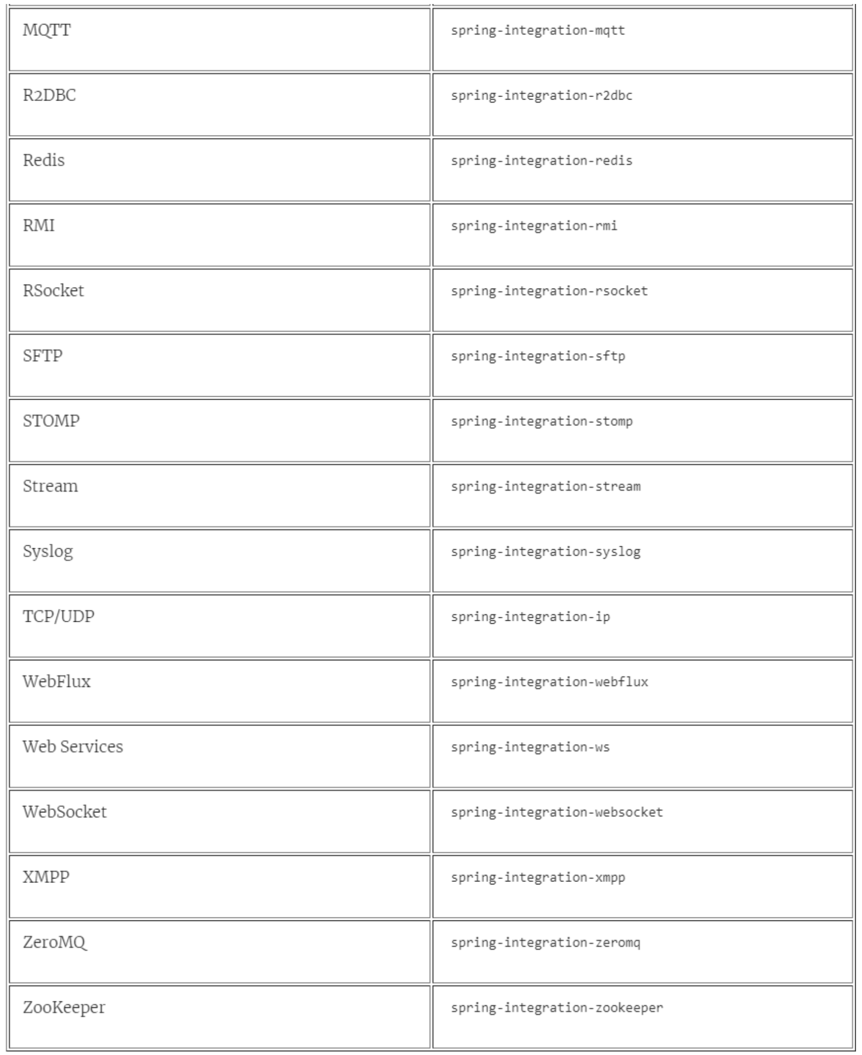


通道适配器（DSL定义）

```
@Bean
public IntegrationFlow someFlow(AtomicInteger integerSource) {
  return IntegrationFlows
      .from(integerSource, "getAndIncrement",
          c -> c.poller(Pollers.fixedRate(1000)))
    ...
      .get();
}
```

```
@Bean
public IntegrationFlow fileReaderFlow() {
  return IntegrationFlows
      .from(Files.inboundAdapter(new File(INPUT_DIR))
          .patternFilter(FILE_PATTERN))
      .get();
}
```

端点模块（Endpoint modules）

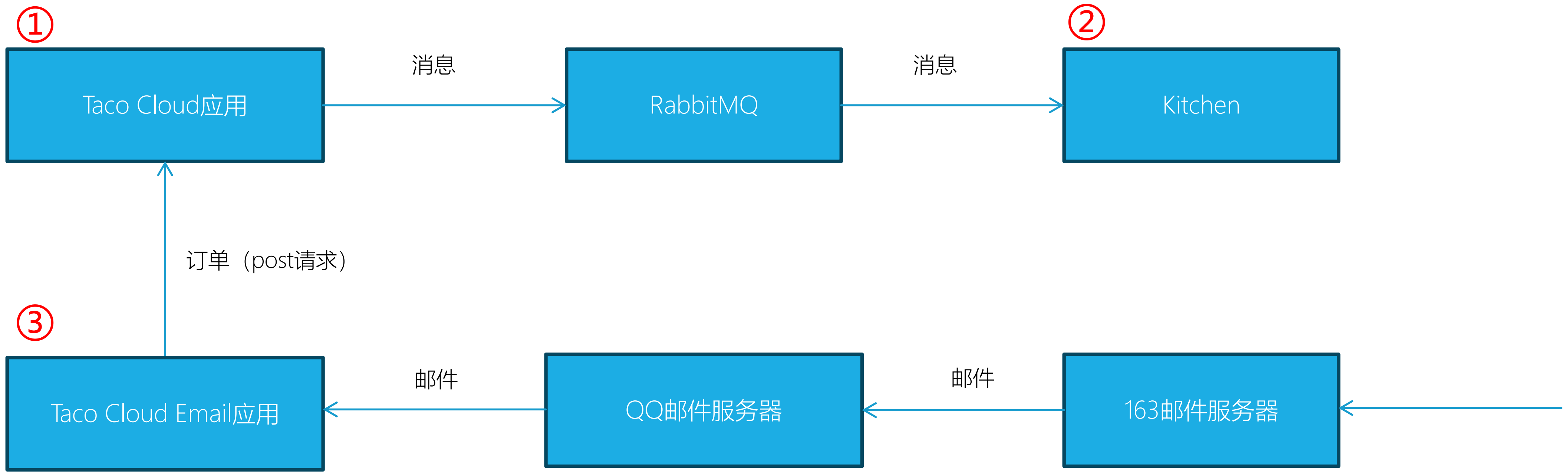



Module	Dependency artifact ID (Group ID: `org.springframework.integration`)
AMQP	`spring-integration-amqp`
Spring application events	`spring-integration-event`
RSS and Atom	`spring-integration-feed`
Filesystem	`spring-integration-file`
FTP/FTPS	`spring-integration-ftp`
GemFire	`spring-integration-gemfire`
HTTP	`spring-integration-http`
JDBC	`spring-integration-jdbc`
JPA	`spring-integration-jpa`
JMS	`spring-integration-jms`
JMX	`spring-integration-jmx`
Kafka	`spring-integration-kafka`
Email	`spring-integration-mail`
MongoDB	`spring-integration-mongodb`
MQTT	`spring-integration-mqtt`
R2DBC	`spring-integration-r2dbc`
Redis	`spring-integration-redis`
RMI	`spring-integration-rmi`
RSocket	`spring-integration-rsocket`
SFTP	`spring-integration-sftp`
STOMP	`spring-integration-stomp`
Stream	`spring-integration-stream`
Syslog	`spring-integration-syslog`
TCP/UDP	`spring-integration-ip`
WebFlux	`spring-integration-webflux`
Web Services	`spring-integration-ws`
WebSocket	`spring-integration-websocket`
XMPP	`spring-integration-xmpp`
ZeroMQ	`spring-integration-zeromq`
ZooKeeper	`spring-integration-zookeeper`

内容

1. 一个简单的集成流（例子）
2. 集成流的组件介绍
3. 电子邮件集成流（本节课目标）

本节课目标



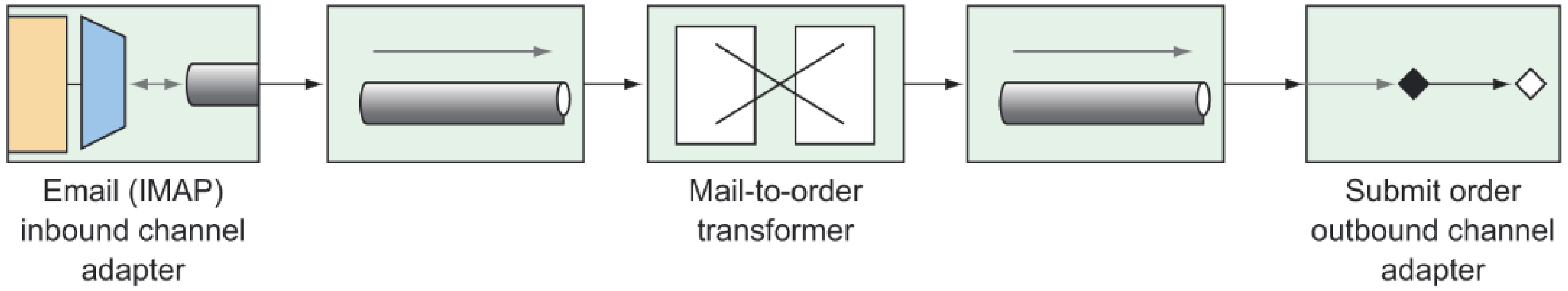
An IMAP email inbound channel adapter 轮询邮件服务器，触发订单生成

电子邮件端点模块

```
<dependency>
    <groupId>org.springframework.integration</groupId>
    <artifactId>spring-integration-mail</artifactId>
  </dependency>
```

构建集成流

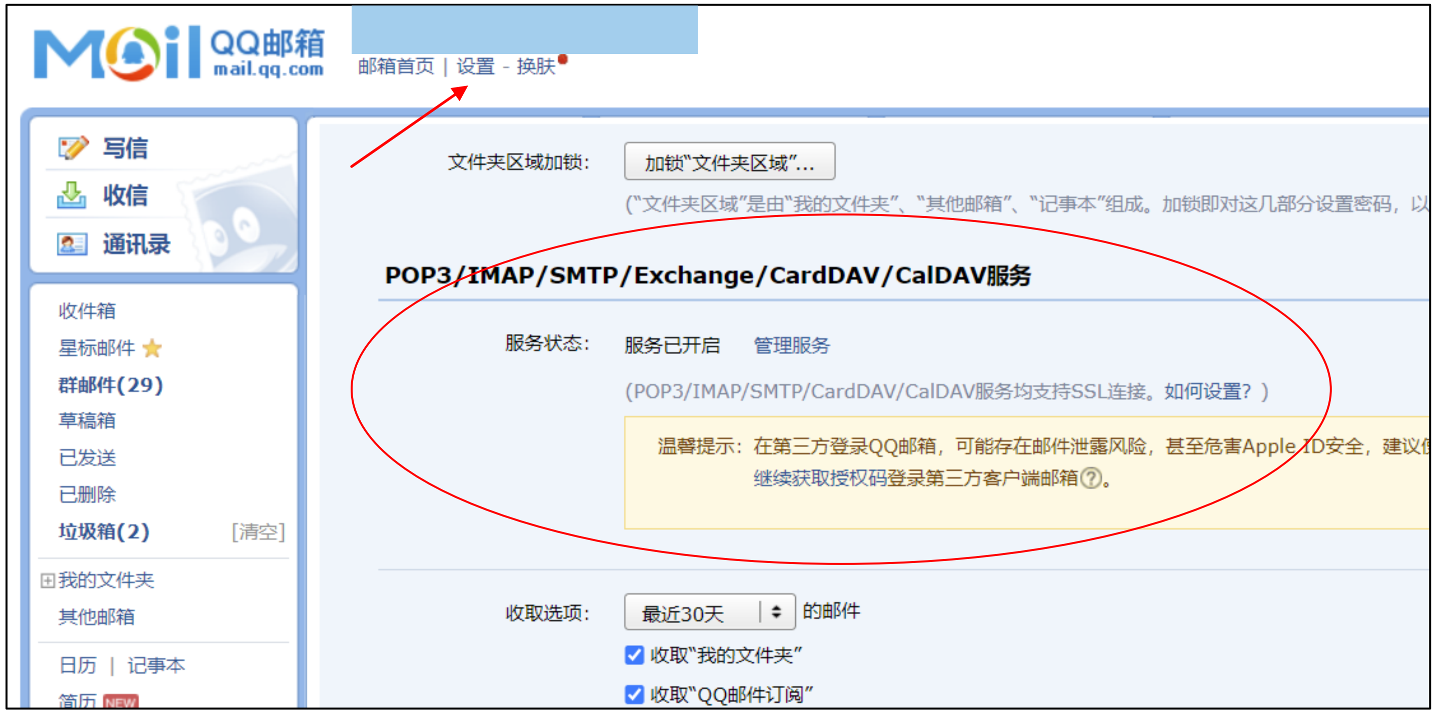
- TacoOrderEmailIntegrationConfig.java (DSL方式)



设置QQ邮箱

- https://wx.mail.qq.com/
- 开启IMAP服务
- 获取授权码
- 配置：

```
tacocloud:
  email:
    host: imap.qq.com
    mailbox: INBOX
    username: qq账号
    password: 授权码
    poll-rate: 10000
```



QQ邮箱 mail.qq.com

邮箱首页 | 设置 - 换肤

写信
收信
通讯录

收件箱
星标邮件
群邮件(29)
草稿箱
已发送
已删除
垃圾箱(2) [清空]
⊞我的文件夹
其他邮箱
日历 | 记事本
简历 NEW

文件夹区域加锁： 加锁"文件夹区域"...

("文件夹区域"是由"我的文件夹"、"其他邮箱"、"记事本"组成。加锁即对这几部分设置密码，以

POP3/IMAP/SMTP/Exchange/CardDAV/CalDAV服务

服务状态： 服务已开启 管理服务

(POP3/IMAP/SMTP/CardDAV/CalDAV服务均支持SSL连接。如何设置？)

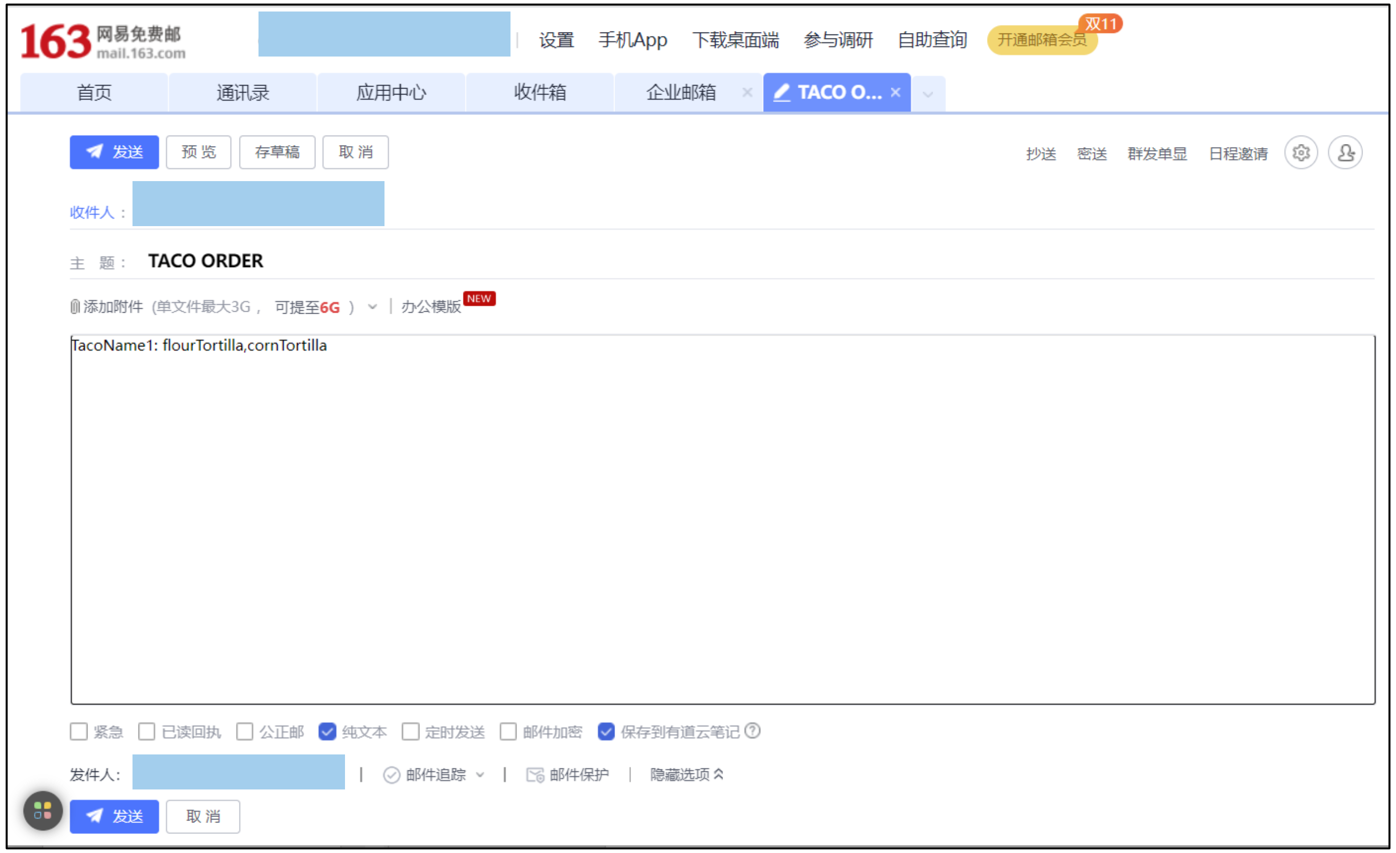
温馨提示：在第三方登录QQ邮箱，可能存在邮件泄露风险，甚至危害Apple ID安全，建议
继续获取授权码登录第三方客户端邮箱?。

收取选项： 最近30天 的邮件

✓ 收取"我的文件夹"
✓ 收取"QQ邮件订阅"

发送邮件

- 注意使用纯文本方式
  - ✓ 邮件主题：TACO ORDER
  - ✓ 邮件内容：TacoName1: flourTortilla,cornTortilla



收件人：[redacted]

主　题：TACO ORDER

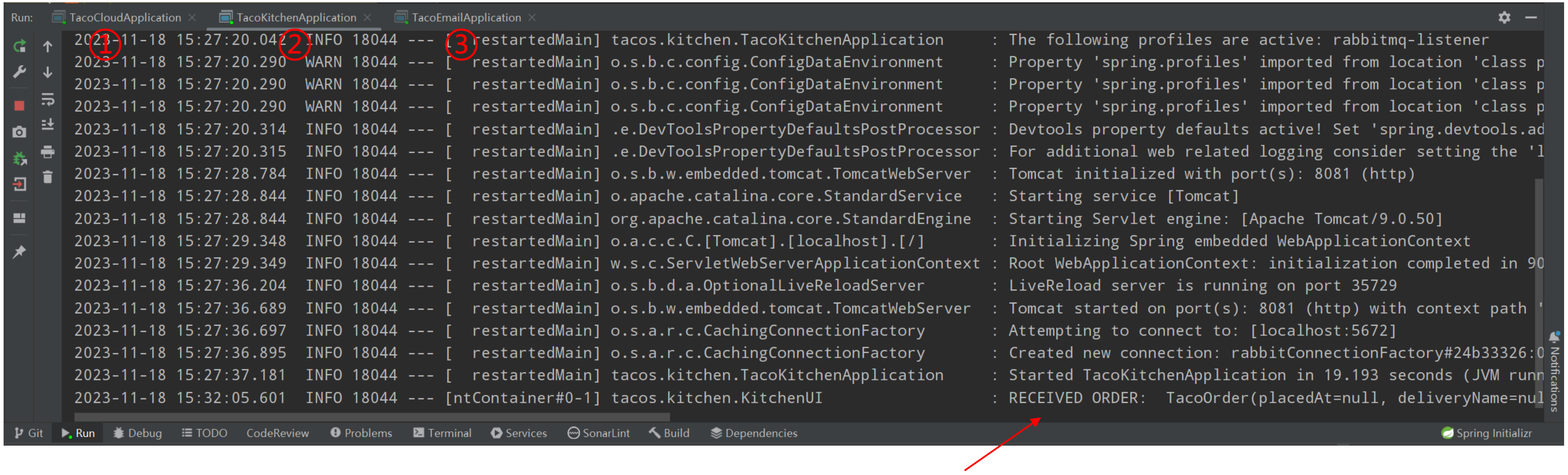
TacoName1: flourTortilla,cornTortilla

☐ 紧急 ☐ 已读回执 ☐ 公正邮 ☑ 纯文本 ☐ 定时发送 ☐ 邮件加密 ☑ 保存到有道云笔记

发件人：[redacted]

作业

- 配置自己的邮件服务器，使用邮件触发订单
- 提交以下截图，含①②③三个应用的启动，以及获取到订单的日志



```
2023-11-18 15:27:20.042  INFO 18044 --- [  restartedMain] tacos.kitchen.TacoKitchenApplication     : The following profiles are active: rabbitmq-listener
2023-11-18 15:27:20.290  WARN 18044 --- [  restartedMain] o.s.b.c.config.ConfigDataEnvironment     : Property 'spring.profiles' imported from location 'class p
2023-11-18 15:27:20.290  WARN 18044 --- [  restartedMain] o.s.b.c.config.ConfigDataEnvironment     : Property 'spring.profiles' imported from location 'class p
2023-11-18 15:27:20.290  WARN 18044 --- [  restartedMain] o.s.b.c.config.ConfigDataEnvironment     : Property 'spring.profiles' imported from location 'class p
2023-11-18 15:27:20.314  INFO 18044 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.ad
2023-11-18 15:27:20.315  INFO 18044 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'l
2023-11-18 15:27:28.784  INFO 18044 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8081 (http)
2023-11-18 15:27:28.844  INFO 18044 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-11-18 15:27:28.844  INFO 18044 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.50]
2023-11-18 15:27:29.348  INFO 18044 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2023-11-18 15:27:29.349  INFO 18044 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 90
2023-11-18 15:27:36.204  INFO 18044 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2023-11-18 15:27:36.689  INFO 18044 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8081 (http) with context path '
2023-11-18 15:27:36.697  INFO 18044 --- [  restartedMain] o.s.a.r.c.CachingConnectionFactory       : Attempting to connect to: [localhost:5672]
2023-11-18 15:27:36.895  INFO 18044 --- [  restartedMain] o.s.a.r.c.CachingConnectionFactory       : Created new connection: rabbitConnectionFactory#24b33326:0
2023-11-18 15:27:37.181  INFO 18044 --- [  restartedMain] tacos.kitchen.TacoKitchenApplication     : Started TacoKitchenApplication in 19.193 seconds (JVM runn
2023-11-18 15:32:05.601  INFO 18044 --- [ntContainer#0-1] tacos.kitchen.KitchenUI                  : RECEIVED ORDER:  TacoOrder(placedAt=null, deliveryName=nul
```

谢谢观看！