

SPRING INTEGRATION | JAVA DSL

Владимир Плизга
ЦФТ

ТЫ ВООБЩЕ КТО?

- Владимир Плизга
- Backend разработчик
 - **Позиция:** главный инженер-программист, ЦФТ (Центр Финансовых Технологий)
 - **Область:** разработка Интернет-банков и сопутствующих сервисов
 - **Опыт** в области: 6+ лет



<https://oplata.kykyryza.ru>

- 20+ партнёров по Интернет-банкам
- 20+ партнёров по сайтам подарочных карт

ЧТО ТАКОЕ SPRING INTEGRATION?

- Проект в экосистеме Spring

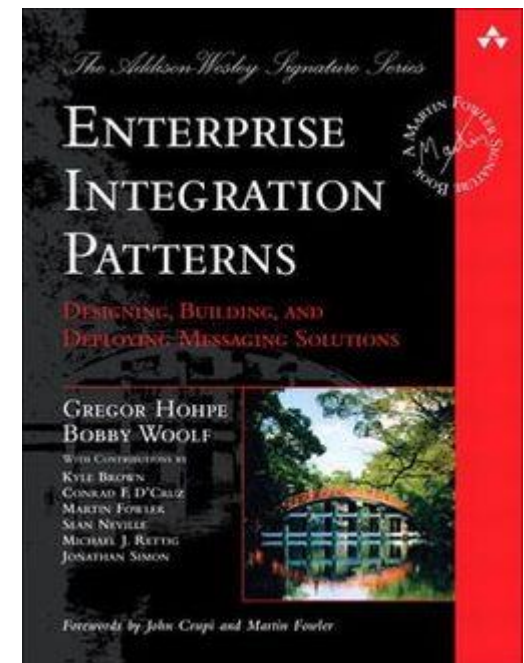
Spring Integration = Spring IoC + EIP

- * **IoC** – Inversion of Control
- * **EIP** – Enterprise Integration Patterns



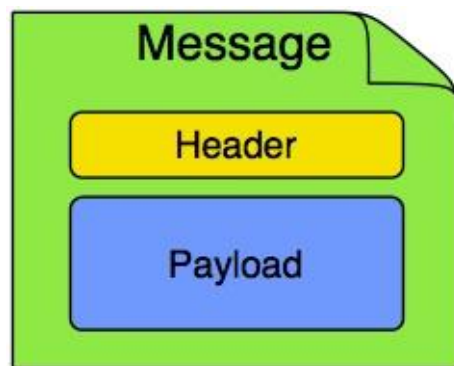
ВКРАТЦЕ О EIP

- Высокоуровневая программная модель для построения интеграций
 - Как внутри приложения
 - Так и между приложениями
- 65 паттернов на все случаи жизни
- Обеспечивает слабую связность компонентов
- Не зависит от стека технологий



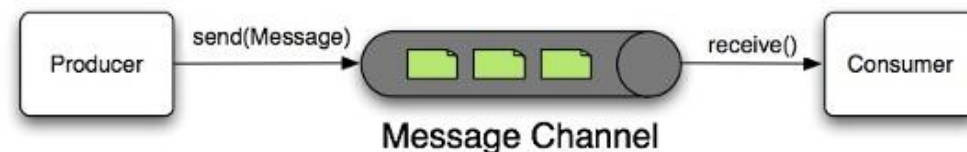
КОНЦЕПЦИИ EIP: MESSAGE & CHANNEL

Сообщение



```
public interface Message<T> {  
    T getPayload();  
    MessageHeaders getHeaders();  
}
```

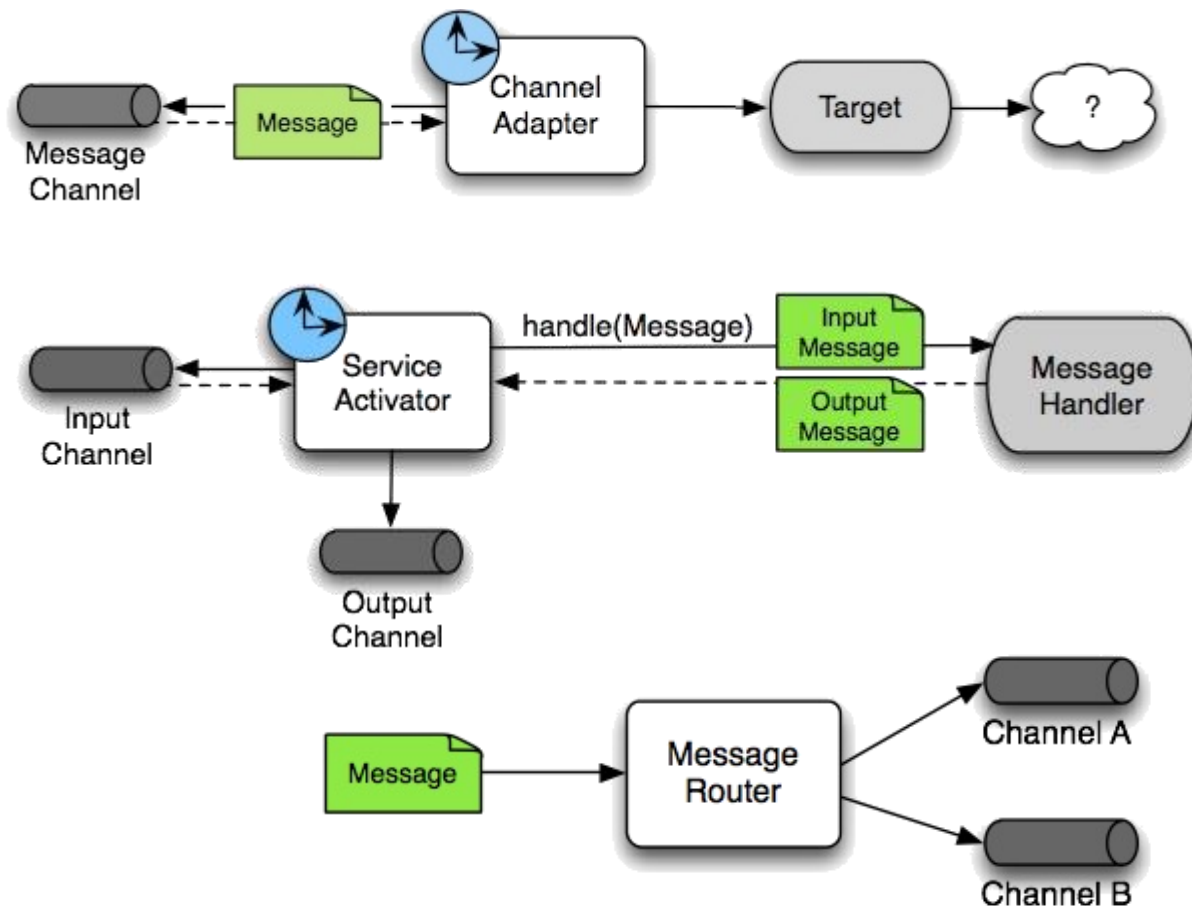
Канал



```
@FunctionalInterface  
public interface MessageChannel {  
    boolean send(Message<?> message);  
}
```

КОНЦЕПЦИИ EIP: ENDPOINT

- Transform
- Filter
- Handle
- Split
- Aggregate
- Route
- Bridge
- ...



КОНЦЕПЦИИ EIP: ADAPTER

- AMQP/JMS
- JDBC/JPA/NoSQL
- File
- Mail
- Social
- FTP[S]/SFTP
- HTTP/WebFlux/WebSocket
- RMI
- ...



ПРИЧЁМ ТУТ SPRING INTEGRATION?

«Спецификация»	«Имплементация»
JVMS	HotSpot, JRockit, Excelsior JET
JPA	Hibernate
AMQP	RabbitMQ, ActiveMQ
EIP	Apache Camel, Spring Integration

Набор реализующих EIP компонентов, позволяющих разработчику концентрироваться на бизнес-логике и минимально заниматься инфраструктурой обмена сообщениями

КАК РАБОТАЕТ SPRING INTEGRATION

- Комбинация каналов и точек обработки (endpoints) – конвейер (flow)
- Все элементы конвейера – бины в Spring-контексте
- Конвейером может быть как всё приложение, так и любые участки
- 3 способа задания конфигурации конвейеров (не исключаящие):
 1. XML
 2. Java Annotations
 3. DSLs: Java, Groovy, Scala

CONFIGURATION: XML

```
<int:gateway id="cafe" service-interface="org.springframework.integration.samples.cafe.Cafe"/>

<!-- each order has a collection of order items that is split apart to be processed -->
<int:channel id="orders"/>
<int:splitter input-channel="orders" expression="payload.items" output-channel="drinks"/>

<!-- drink order items are aggregated in a call to the waiter -->
<int:channel id="preparedDrinks"/>
<int:aggregator input-channel="preparedDrinks" method="prepareDelivery"
    output-channel="deliveries">
    <beans:bean class="org.springframework.integration.samples.cafe.xml.Waiter"/>
</int:aggregator>

<int-stream:stdout-channel-adapter id="deliveries"/>
<beans:bean id="barista" class="org.springframework.integration.samples.cafe.xml.Barista"/>
<int:poller id="poller" default="true" fixed-delay="1000"/>
```

CONFIGURATION: JAVA ANNOTATION

@Bean

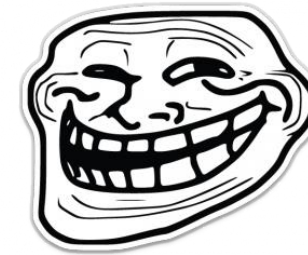
```
@Transformer(inputChannel = "enrichHeadersChannel", outputChannel = "emailChannel")
public HeaderEnricher enrichHeaders() {
    Map<String, ? extends HeaderValueMessageProcessor<?>> headersToAdd =
        Collections.singletonMap("emailUrl",
            new StaticHeaderValueMessageProcessor<>(this.imapUrl));
    HeaderEnricher enricher = new HeaderEnricher(headersToAdd);
    return enricher;
}
```

@Bean

```
@Transformer(inputChannel="enrichHeadersChannel", outputChannel="emailChannel")
public HeaderEnricher enrichHeaders() {
    Map<String, HeaderValueMessageProcessor<?>> headersToAdd = new HashMap<>();
    headersToAdd.put("emailUrl", new StaticHeaderValueMessageProcessor<String>(this.imapUrl));
    Expression expression = new SpelExpressionParser().parseExpression("payload.from[0].toString()");
    headersToAdd.put("from",
        new ExpressionEvaluatingHeaderValueMessageProcessor<>(expression, String.class));
    HeaderEnricher enricher = new HeaderEnricher(headersToAdd);
    return enricher;
}
```

CONFIGURATION: JAVA DSL

- Основан на паттерне Builder и Java Annotation Configuration
- «Another useful tool to simplify configuration is Java 8 Lambdas.»
- Всем видам endpoint'ов соответствуют методы builder'a
- Каждый конвейер – экземпляр класса **IntegrationFlow**, регистрируемый в виде Spring бина:



```
@Bean
public IntegrationFlow integerFlow() {
    return IntegrationFlows.from("input") // input - имя канала
        .<String, Integer> transform(Integer::parseInt)
        .log()
        .get();
}
```

→ implicit direct channel



БЛИЖЕ К ДЕЛУ

Практическая часть

БИЗНЕС ТРЕБОВАНИЕ

Требуется:

- Определять самый упоминаемый проект экосистемы **Spring** в каждой новой публикации блога **Spring Engineering**, а затем выяснять, что говорили об этом проекте в **Twitter**'е за последнюю неделю, и сохранять к себе в БД выжимку этих сведений.

ДЕКОМПОЗИЦИЯ ЗАДАЧИ

1. Поход в Atom Feed по расписанию
2. Определение самого упоминаемого проекта Spring в каждой публикации
3. Запрос твитов по хэш-тегу в Twitter
4. Преобразование к сжатому виду
5. Сохранение в БД

Инструментарий:
Spring Boot + Spring Integration

ПРОТИВОПОКАЗАНИЯ

- Когда легко обойтись обычным Java Stream API
- Когда в команде мало носителей знаний по Spring Integration
- Когда нужен конвейер для массивной транзакционной обработки (для этого есть Spring Batch)

ОСТАЛОСЬ ЗА КАДРОМ

- Каналы: Queue, Executor, Priority, ...
- Endpoints: Filter, Aggregator, Gateway, ...
- Message & Metadata Store
- Мониторинг (в т.ч. JMX)
- Управление транзакциями
- Обработка ошибок
- Тестирование



<https://github.com/Toparvion/sprintegration>

HAPPY INTEGRATION!

Владимир Плизгá (ЦФТ)
toparvion@gmx.com