Spring Integration

Agenda

- Spring Beans, Messaging
- Message
- Channel
- Message Endpoints

Pause med sandwich

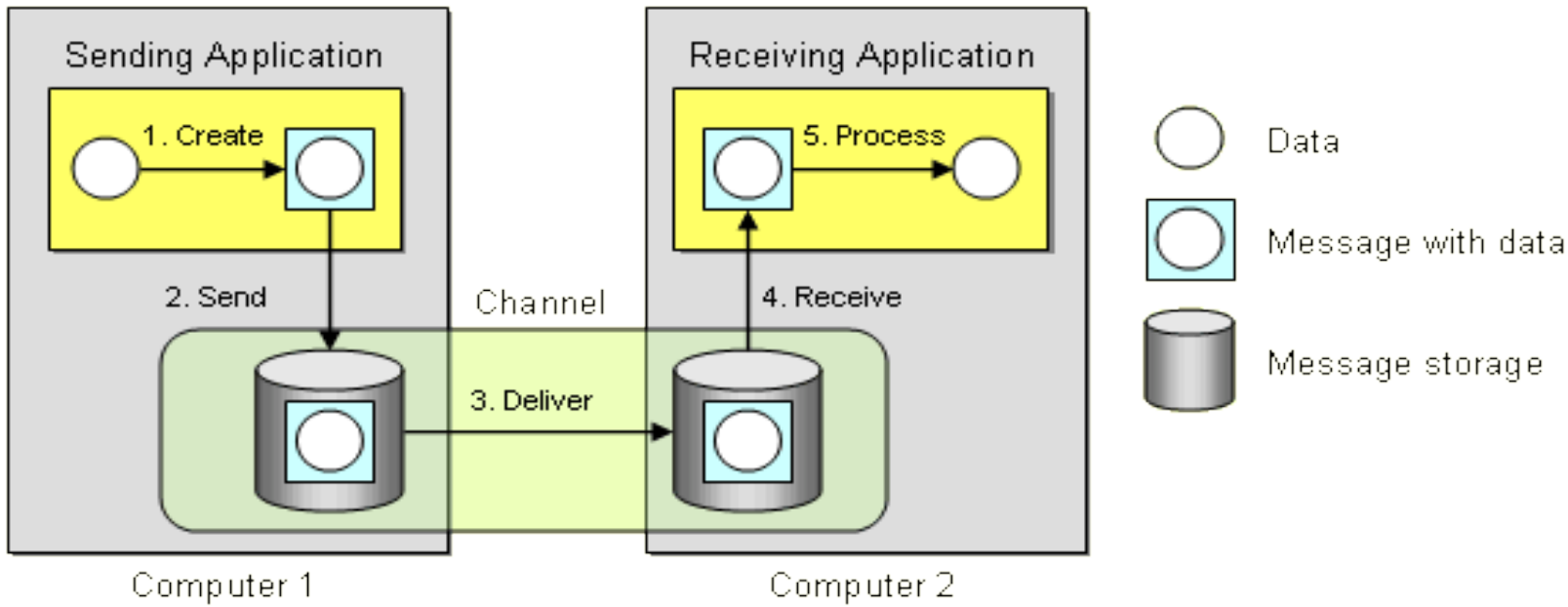
- Channel Adaptors
- Leveringssikerhed og transaktioner
- Test
- Tooling
- Afrunding

Spring Beans

- Traditionelle Spring Beans
  - <bean id="espressoService" class="it.bezzera.Bz07P"/>
- Spring custom beans
  - <si:channel id="orders"/>

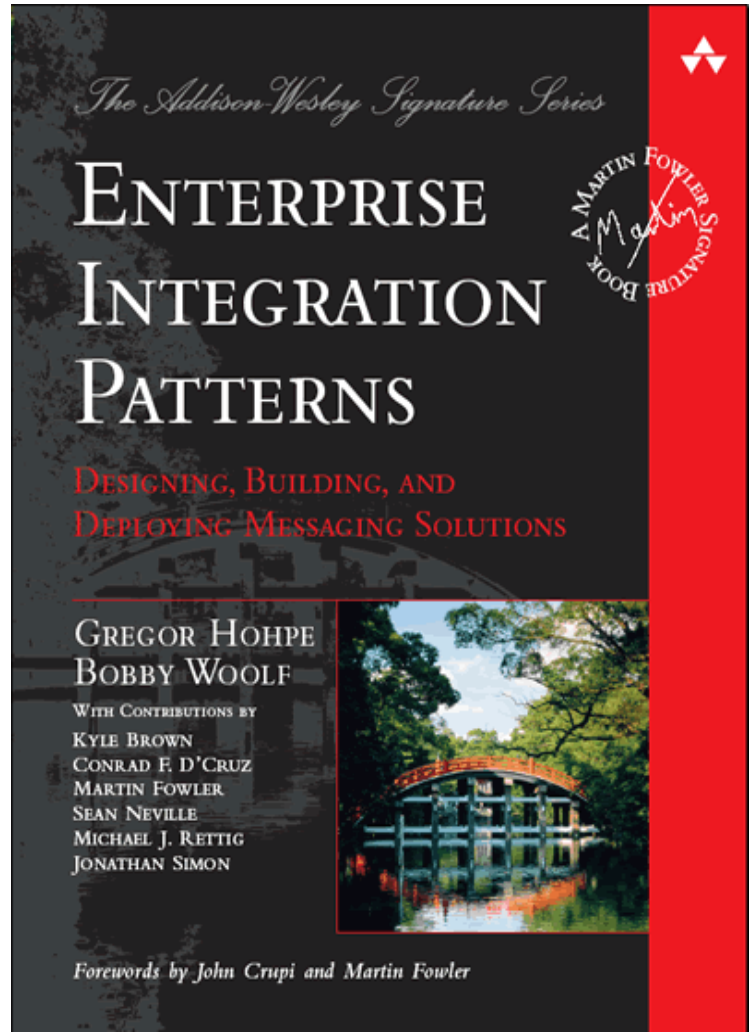
Messaging

- Løs kobling af systemer
- Robust arkitektur
- Orkestrering af beskeder og kanaler
- Bankerne elsker det

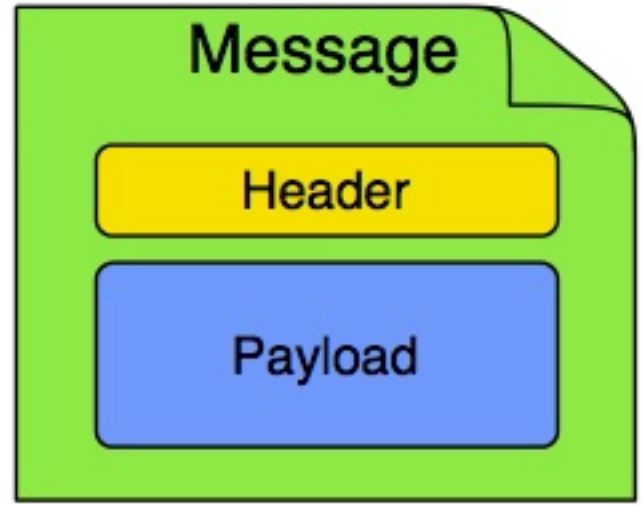


Spring Integration

- Bygger på ideer af EAIP
- Letvægt messaging i Spring applikationer
- En håndfuld adaptere giver "...higher-level of abstraction for remoting, messaging..."
- Simpel model for opbygning af EAI



Message




```
package org.springframework.integration;

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

Metadata til procesering

Kan være ethvert java objekt

class MessageHeaders implements Map<String, Object>, Serializable

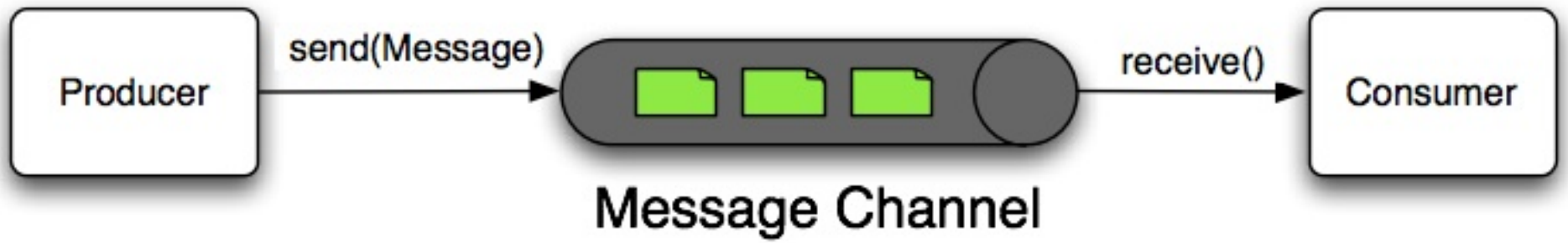
Eksemplerpå Message Headers anvendt af SI-komponenter:

id	timestamp
correlationId	replyChannel
errorChannel	expirationDate
priority	sequenceNumber
sequenceSize	sequenceDetails
file_name	file_originalFile

DEMO 1!

Message payload og headere

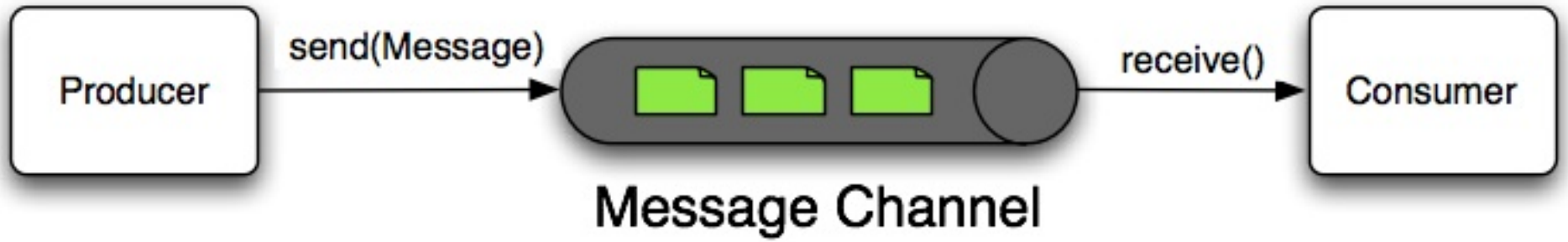
Spring Integration Channel



Medie til at overføre messages fra producer til consumer.

- Point-to-point channel
  - Beskeden sendes kun til en consumer. Producer og consumer kører i samme tråd.
- Publish-subscribe channel
  - Flere modtagere. Consumere skal kunne modtage message når den sendes af producer.
- Queue Channel
  - Consumere poller for besked når de er klar.

Spring Integration Channel - fortsat



- Persistens
  - Channels er default in-memory!
  - messageStore kan angives på channel
- Andre channel-typer:
  - *PriorityChannel:* Messages prioriteres efter priority header eller prioriteringsregel.
  - *RendevousChannel:* Som queue med kapacitet 0. Blocker producer indtil consumer har taget beskeden.

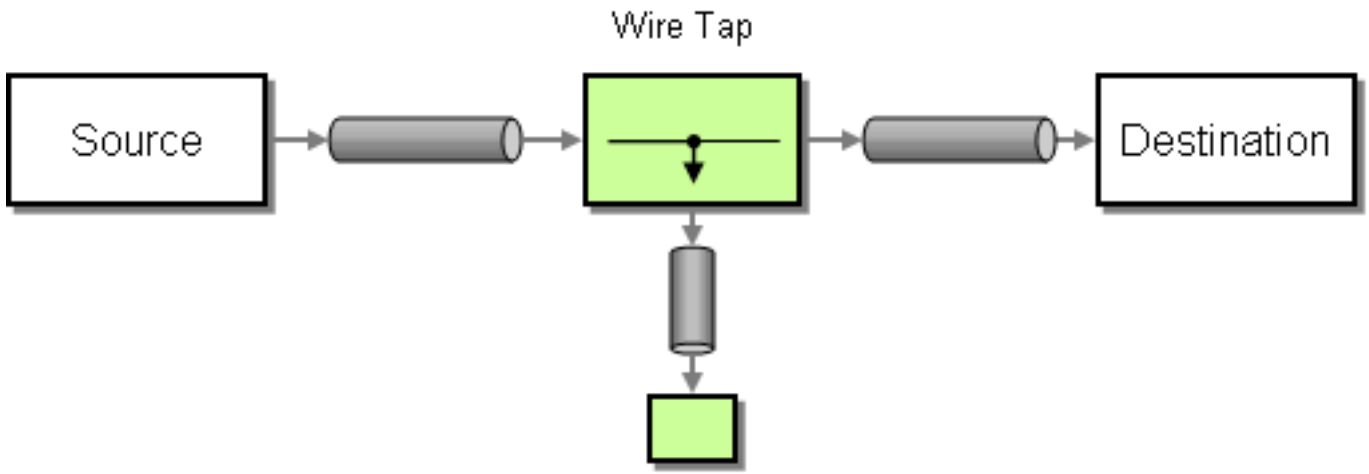
DEMO 2!

Queue channel
Publish-subscriber channel

Medfølgende kanaler

- errorChannel
  - modtager for alle ErrorMessages
  - point-to-point
- nullChannel
  - ækvivalent til /dev/null

Wiretap



- Mulighed for en ekstra modtager på en channel
- Godt til eksempelvis
  - logning og monitorering
  - Arkivering

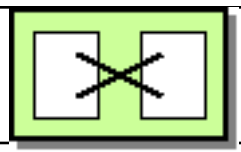
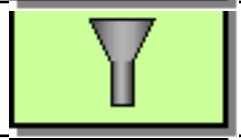
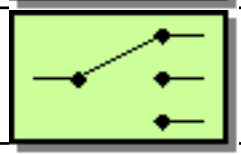
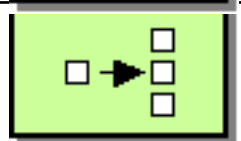
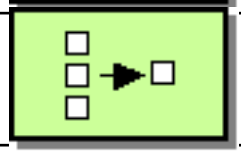
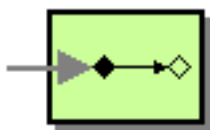
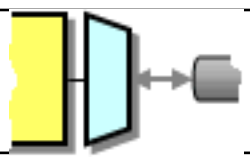
Chains

- Når point-to-point kanaler er overflødige

```
<chain input-channel="..." [output-channel="..."]>
    <header-enricher .../>
    <transformer .../>
    <router .../>
</chain>
```

Message Endpoints

- Flyt ansvar for integration ud af forretnings-koden
- Deklarativ komponent-konfiguration
- Inversion of Control

	Transformer	Får besked fra inputChannel, ændrer den og sender videre på outputChannel.
	Filter	Afgør om en besked skal sendes videre. Regel udtrykkes i enten SPeL eller kodes i java.
	Router	Vælger hvilken kanal en besked skal sendes videre ad.
	Splitter	Splitter en besked på i flere, som sendes videre ad samme channel.
	Aggregator	Samler flere beskeder fra en channel baseret på regler for hvor mange og hvilke der skal til.
	Service Activator	Kalder en synkron service med beskedens payload som input og sender servicens output videre som payload.
	Channel Adaptor	Forbinder kanalen med et andet system eller transport-mekanisme.

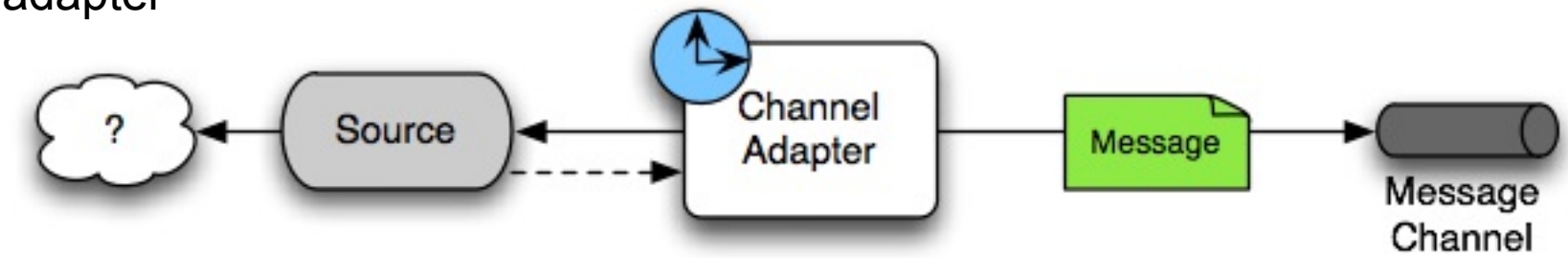
DEMO 3!

Transformers og
Adaptors

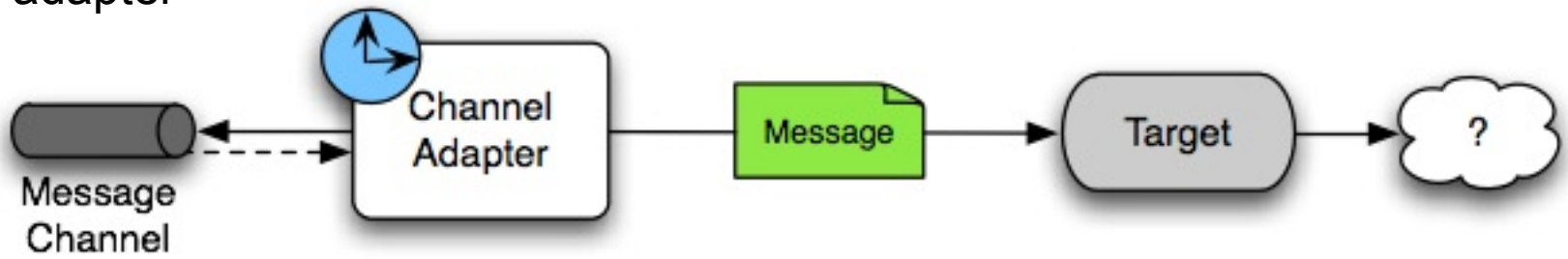
Pause

Channel Adapters

Inbound Channel adapter



Outbound channel adapter



Channel adapters/Gateways inkluderet i Spring Integration:

- Feed RSS/ATOM
- Fil
- FTP/FTPS/SFTP
- HTTP
- Mail
- TCP/UDP
- JDBC
- JMS
- RMI
- Stream
- Twitter
- Web Service
- XMPP

Splitter og aggregator

- Splitter en List til et antal beskeder
- Tildeler header på hver besked
  - correlation_id
  - sequence_number
  - sequence_size

- Aggregator samler et antal af beskeder til en List efter samme regler

Router

- Sender beskeder på bestemte channels
- Regler kan være baseret på
  - Indhold
  - Type
  - Headere
  - Og andet

DEMO 4!

Splitter, aggregator og router

Leveringssikkerhed og transaktioner

- Spring Transaction Manager
  - Transaktion er bundet på tråden
- Transaktionelle channels
  - Queue-backed channel på persistent message store
  - JMS-backed channel
- Initialisering af transaktion
  - Bruger-drevet
    - En tråd startes udefra og rammer SI. Tx skal være sat op på forhånd.
  - Daemon-drevet
    - Poller/Scheduler starter en tråd.
    - <poller max-messages-per-poll="1" fixed-rate="1000"> <transactional transaction-manager="txManager" isolation="DEFAULT" propagation="REQUIRED" read-only="true" timeout="1000"/> </poller>
- Semi-transaktionelle channel adapters
  - Alle pånær JDBC og JMS!
  - DIY: Spring Transaction postCommit() hook

Unit test

- Brug mock værktøj
  - jMock (kan klare asynkrone tilstande fra version 2.6)
  - Mockito
- Del AppContext op i hvad der kun har med integrationslogik at gøre
  - Udskil Adaptorer, egne beans, messagestore osv.

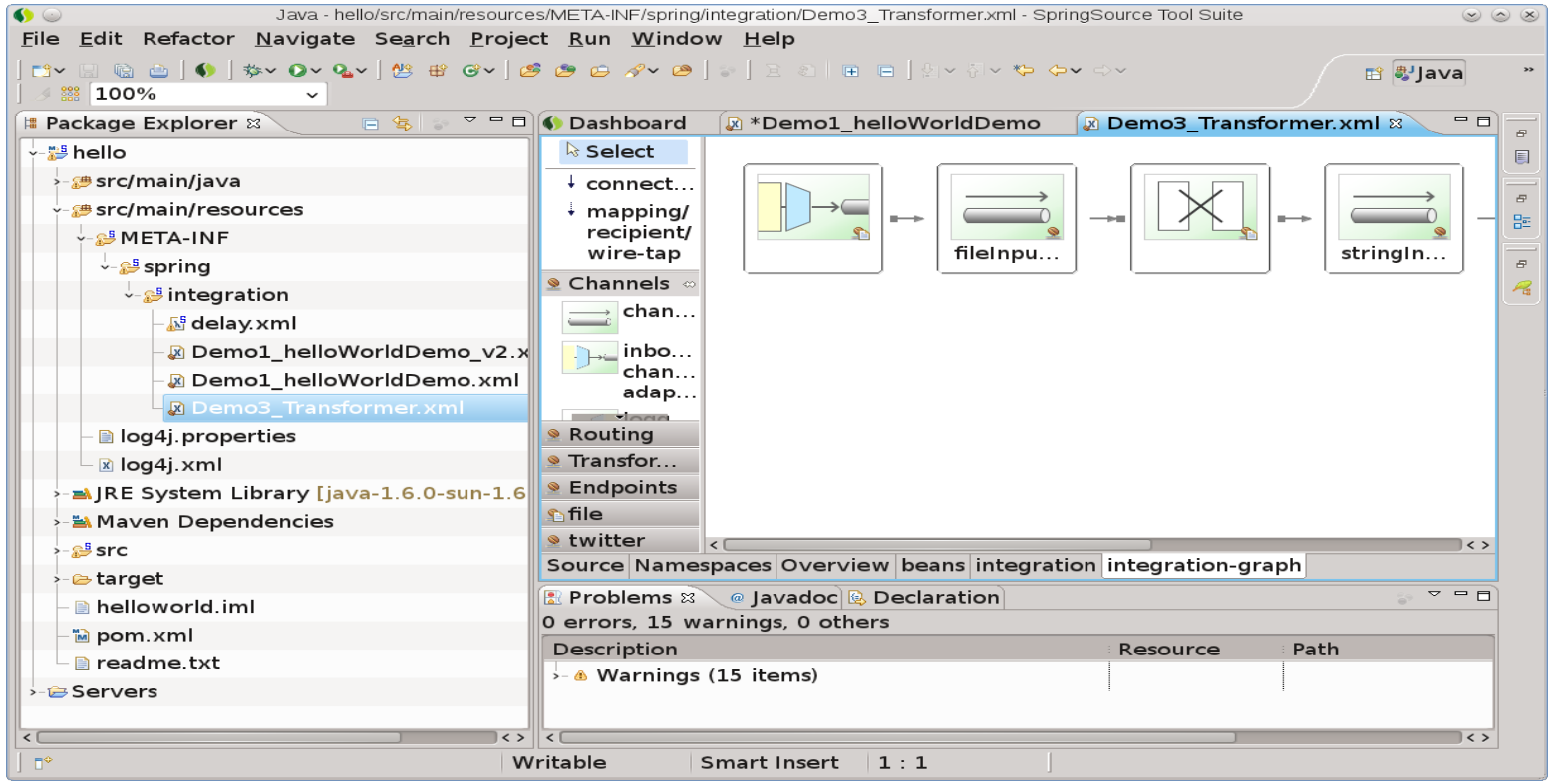
Demo 5

Unit testing

Tooling

IDE

- IntelliJ IDEA 10.5
  - Visuel præsentation af dependencies mellem spring beans
- SpringSource Tool Suite (STS)
  - Visuel præsentation af flow (men grim og ubrugelig)



RTE

Standard Java/Spring (JVM v5+, JMX, JDBC osv.)

Afrunding

Andre interessante aspekter ved Spring Integration:

- Gateways versus channels
- Understøttelse af XML payloads
  - Validering
  - XSLT
  - Xpath
- Security på channels
- Java annotation API til Message Publishers
- Groovy support

Kom godt igang

- http://www.springsource.org/spring-integration
- Masser af samples

Apache Camel

- Alternativ til SI