Spring Integration

Introduction (SI)

Messaging Infra

Today - Theory. (Concepts)

Monday - Hands on

Enterprise Messaging. JMS / JCA

this story

Quickly

SI → EAI

↓

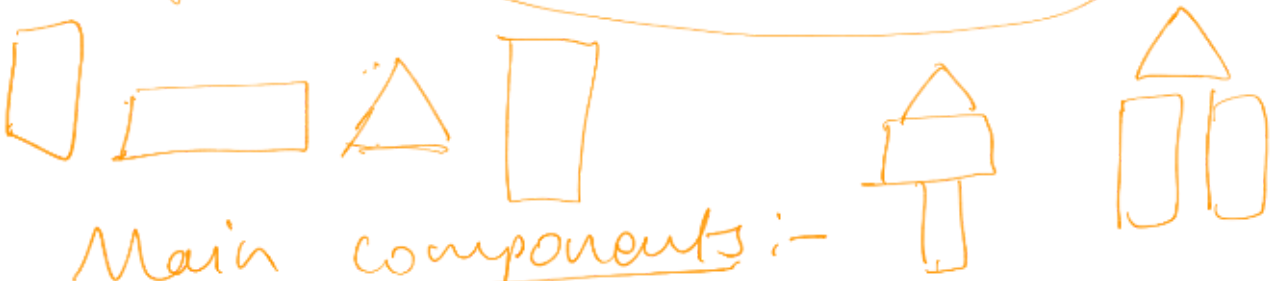
Enterprise App Integration

S/W Architecture capable of integrating Heterogeneous systems applications

SI

→ messaging architecture, built over and above Spring Framework

series of components that can be combined to create a typical EAI application

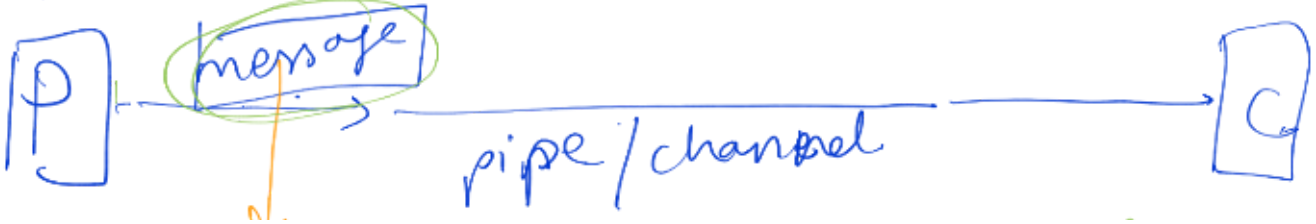


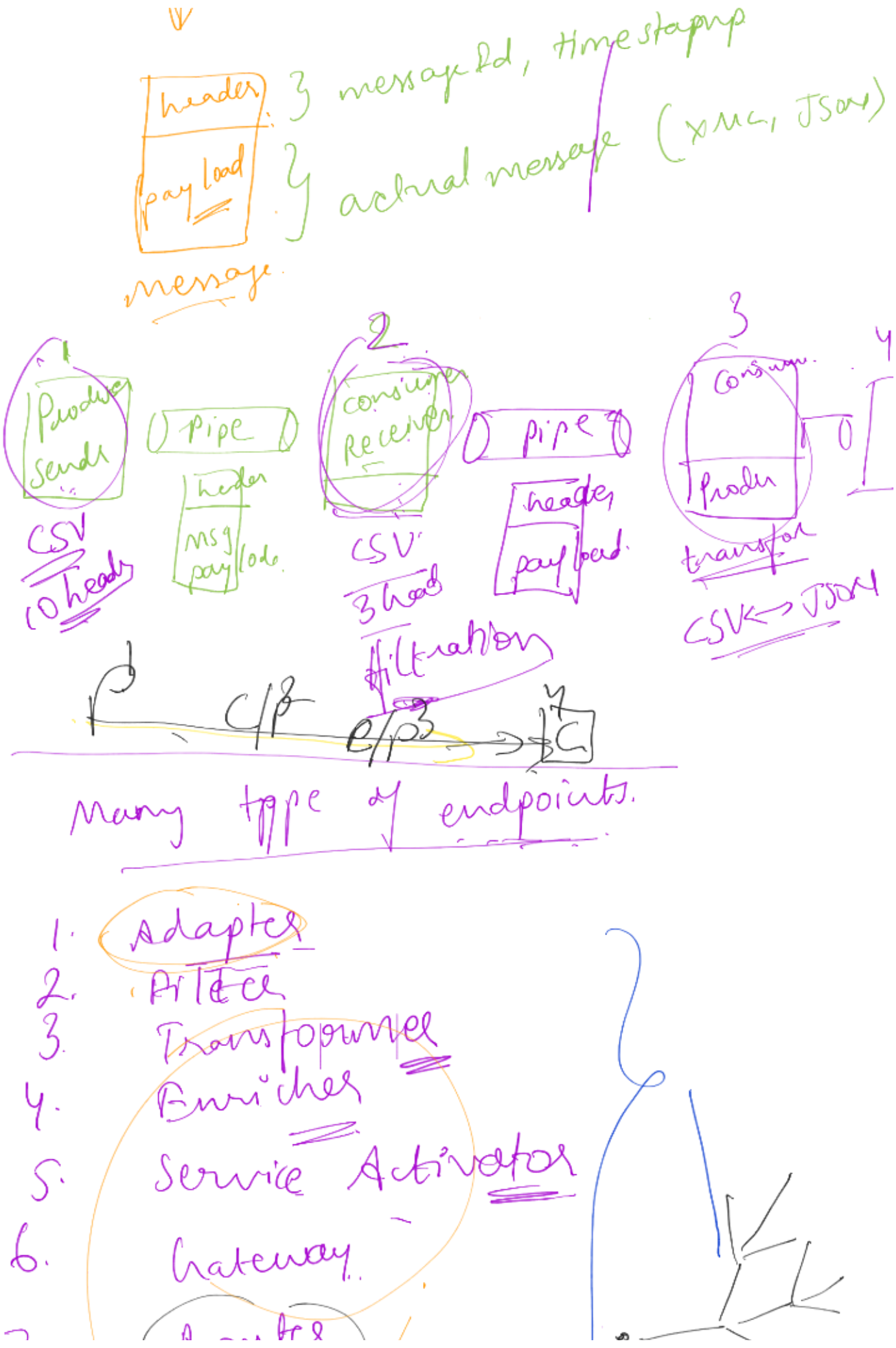
Main components :-

- Producer (sender) } endpoints
- Consumer (Receiver) }
- Pipe / Channel (helping communication btw producer and consumer)

endpoint

endpoint

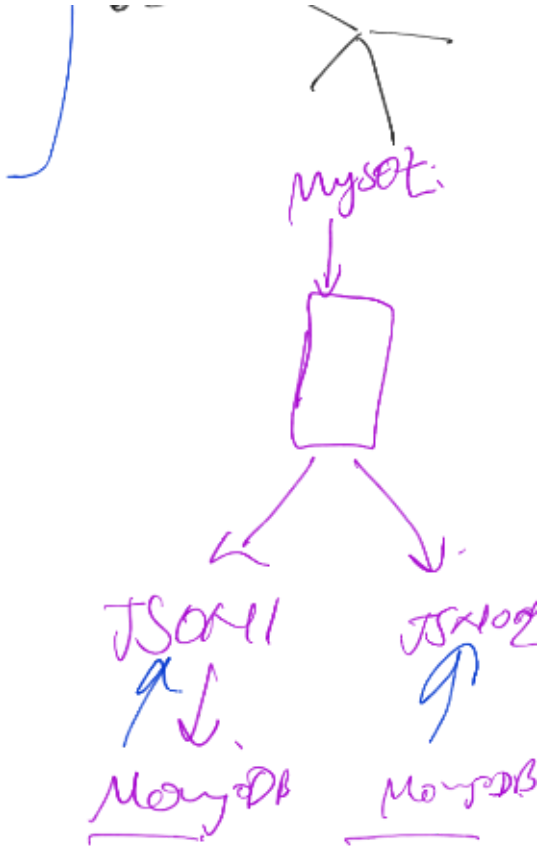



header
payload
message
messageId, timestamp
actual message (XML, JSON)
Produce sends
CSV
10 heads
Pipe
header
msg payload
consume Receive
CSV
3 head
pipe
header
payload
Consum
Produ
transfor
CSV ↔ JSON
filtration
Many type of endpoints.
1. Adapter
2. Filter
3. Transformer
4. Enricher
5. Service Activator
6. Gateway

7. Router

8. Splitter

9. Aggregator.

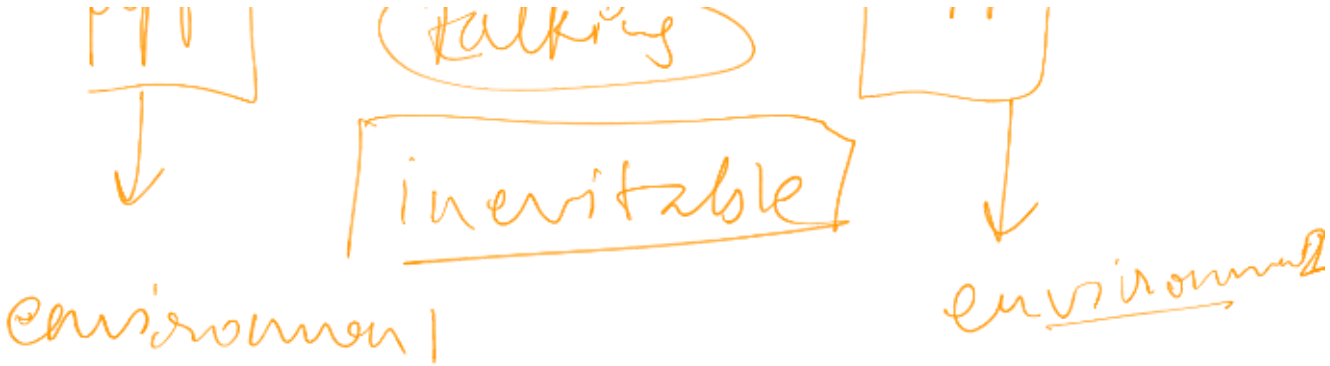


Quick bootup for SI → Done.

Hello world Handson SI → Done

Next → Basic building Thoughts

1. Complex message architecture.
   - In any enterprise project
   - Integration is always a challenge



- Before messaging way of integration was made standard.

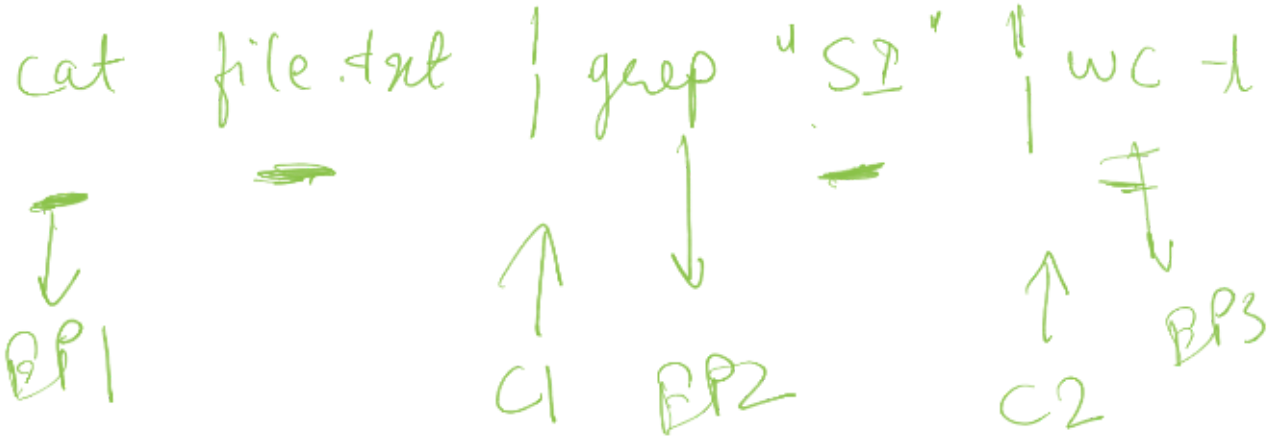
1. Shared file system
2. Single / common Database
3. Messaging

Messaging Strategy

Messaging Patterns

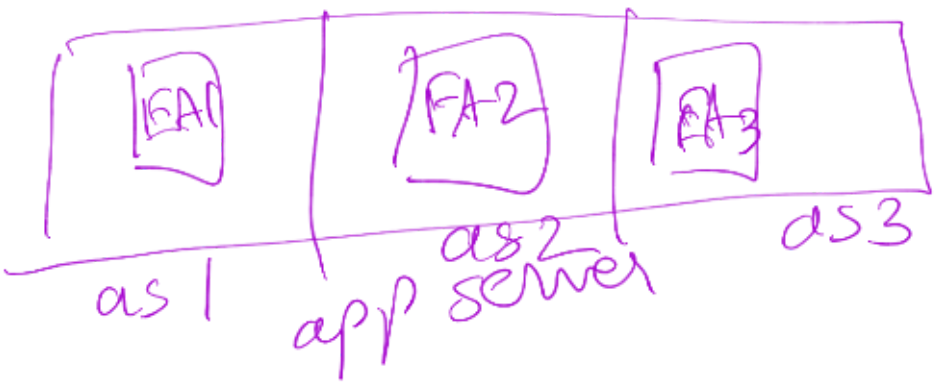
Pipe & filter

Unix pipe symbol |



- These commands do not know anything about each other.
- Small
- narrow focus → take input message
  → publish message output

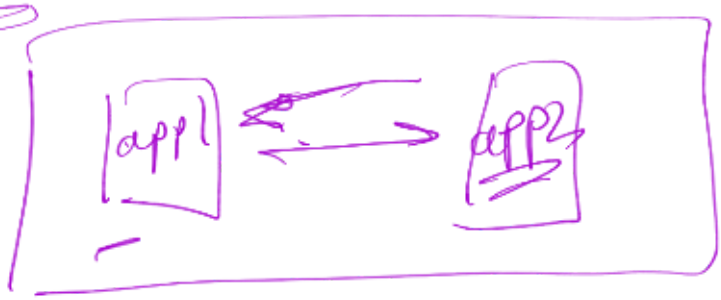
- FA → running on the network.
  ↓
  use application server to host them



- Talk to each other using exposed services

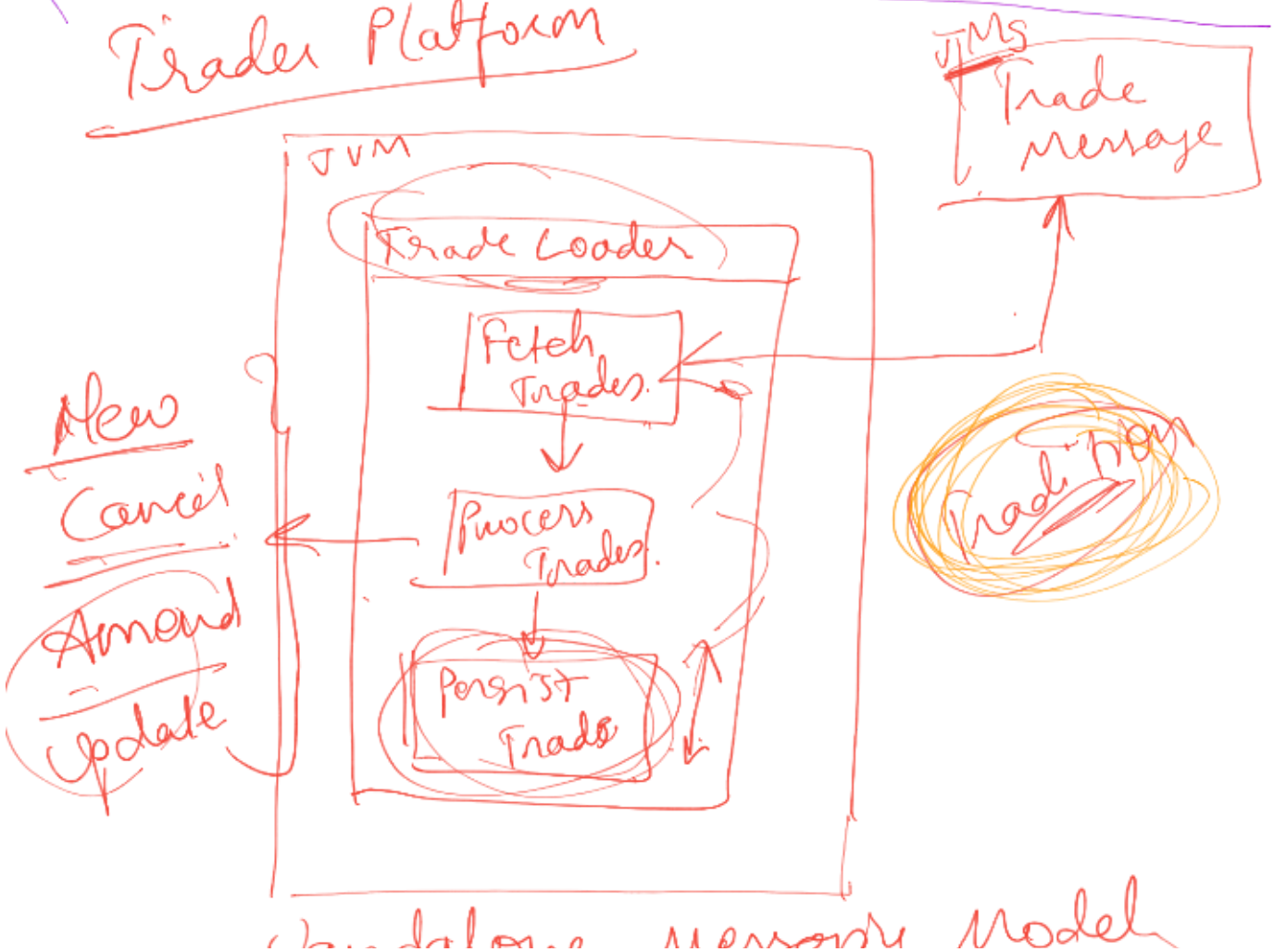
- call eachother exposed source as per the needs.

- One standalone program in JVM another

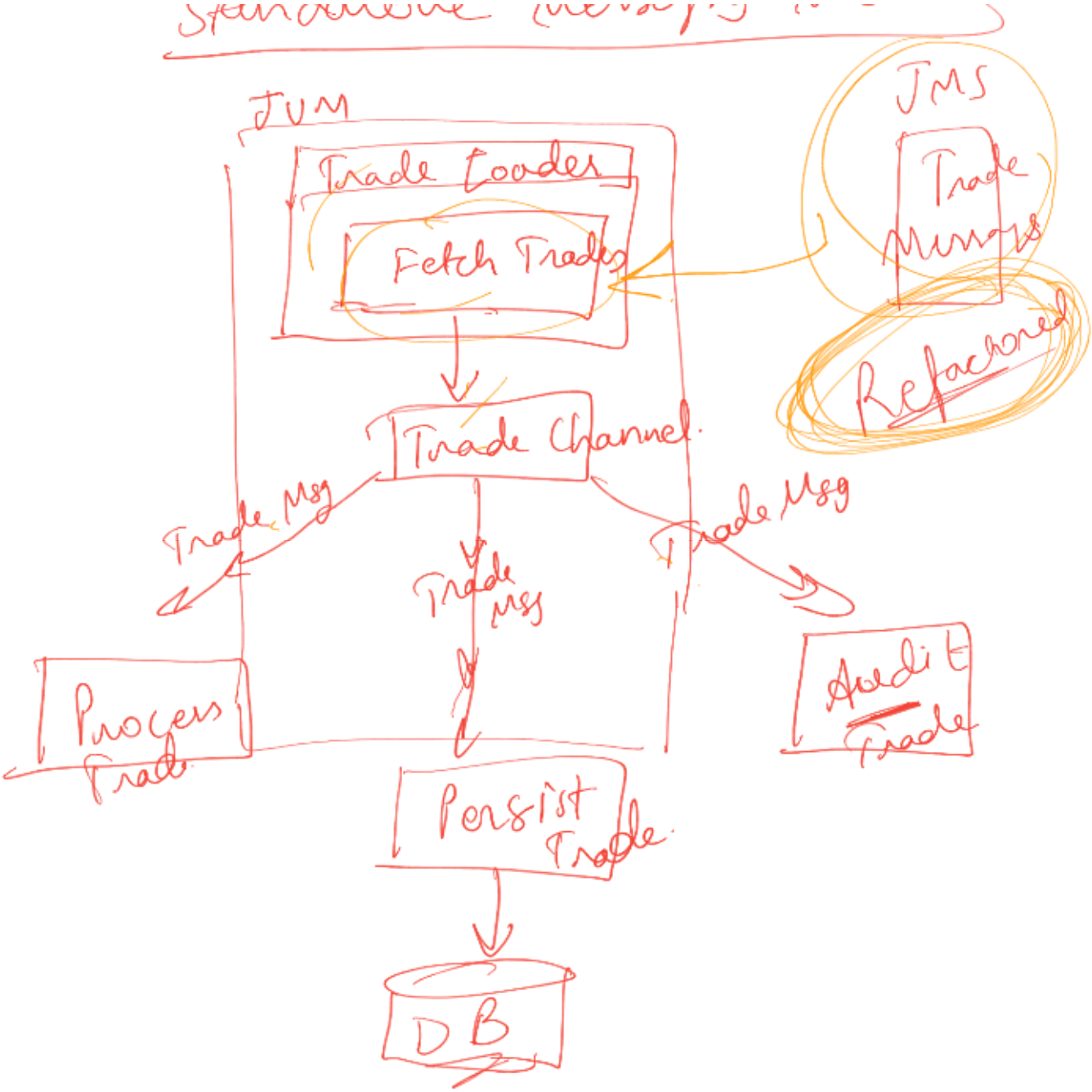


JVM

Intra-JVM messaging



Standalone Messaging Model



- EIP → building blocks

1. Message → header + payload
2. channels
3. Endpoints

Message Interface

Message Builder class

Channels → location where message has to be sent

Message Channel Interface

Endpoints → components

- consume message from input channel
- deliver message to output channel
- combining our business logic with integration spec/Impl.

created declaratively

Part of endpoint

Adapters
Activator
Gateways
Transformers
Filters
Routers

Service Activator → generic EP
who invokes a method on a bean
whenever a message arrives
on a channel

<int: service-activator
input-channel = "pc"
ref = "newMethodCall"
method = "newMethod"

</int: service-activator>

- Message Bridge
  → it couples / connects diff. messaging modes.

- Message Enricher
  → enrich incoming msg with additional information

- Gateway
  → two messaging systems to talk / connect to each other

- Delayer → introduce the delay btw" sender & receiver

- Transformers
  JSON ↔ XML

- Routers ⟷ Flow Components
  - Filters
- Splitters → split the message
- Aggregators → assemble the message

Next → Good Hands One
- Monday