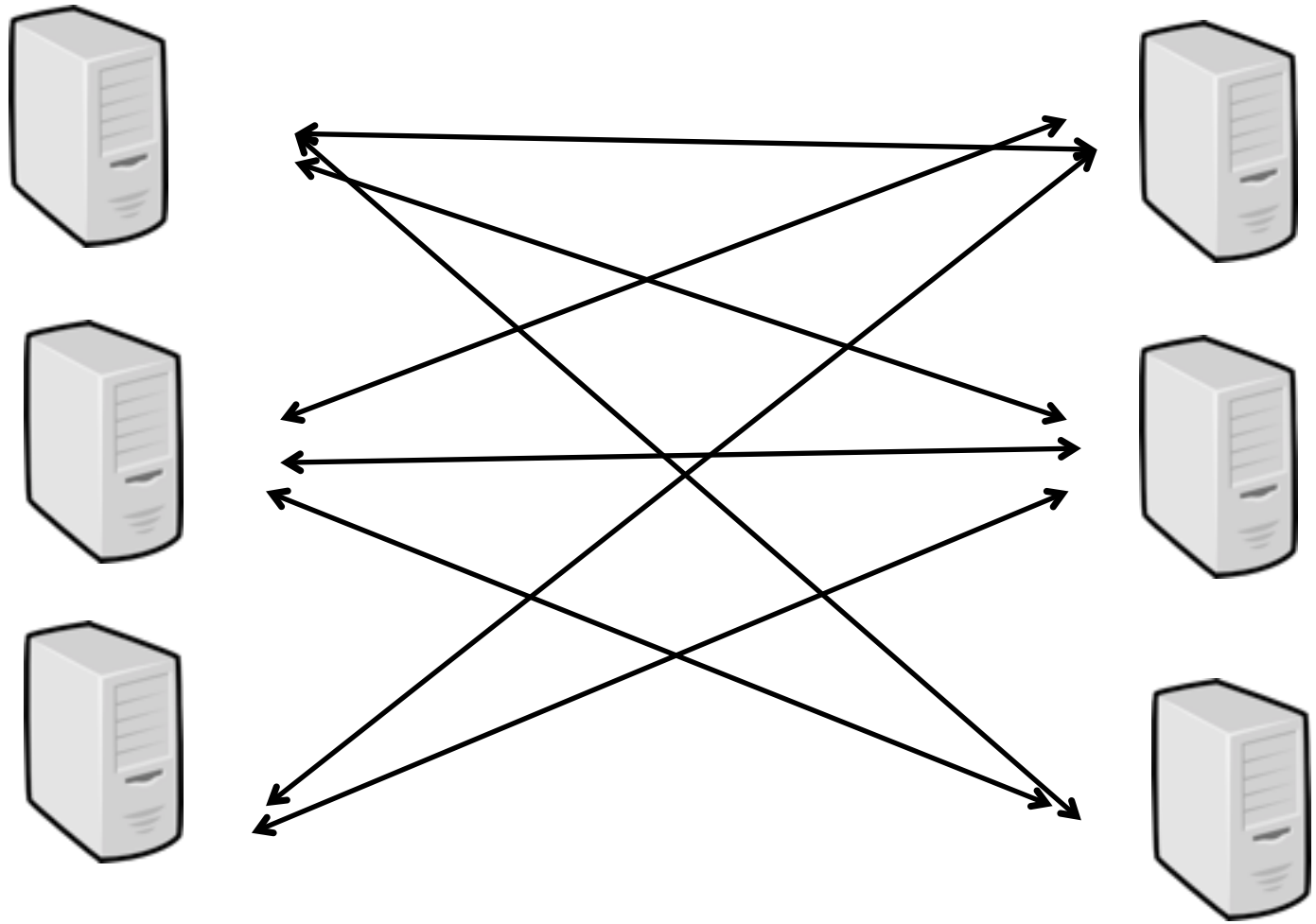
Enterprise Integration Patterns

19/03/2013

C.Exbrayat

Cédric Exbrayat
Ninja Squad

Intégration



Comment faire communiquer différentes applications ?

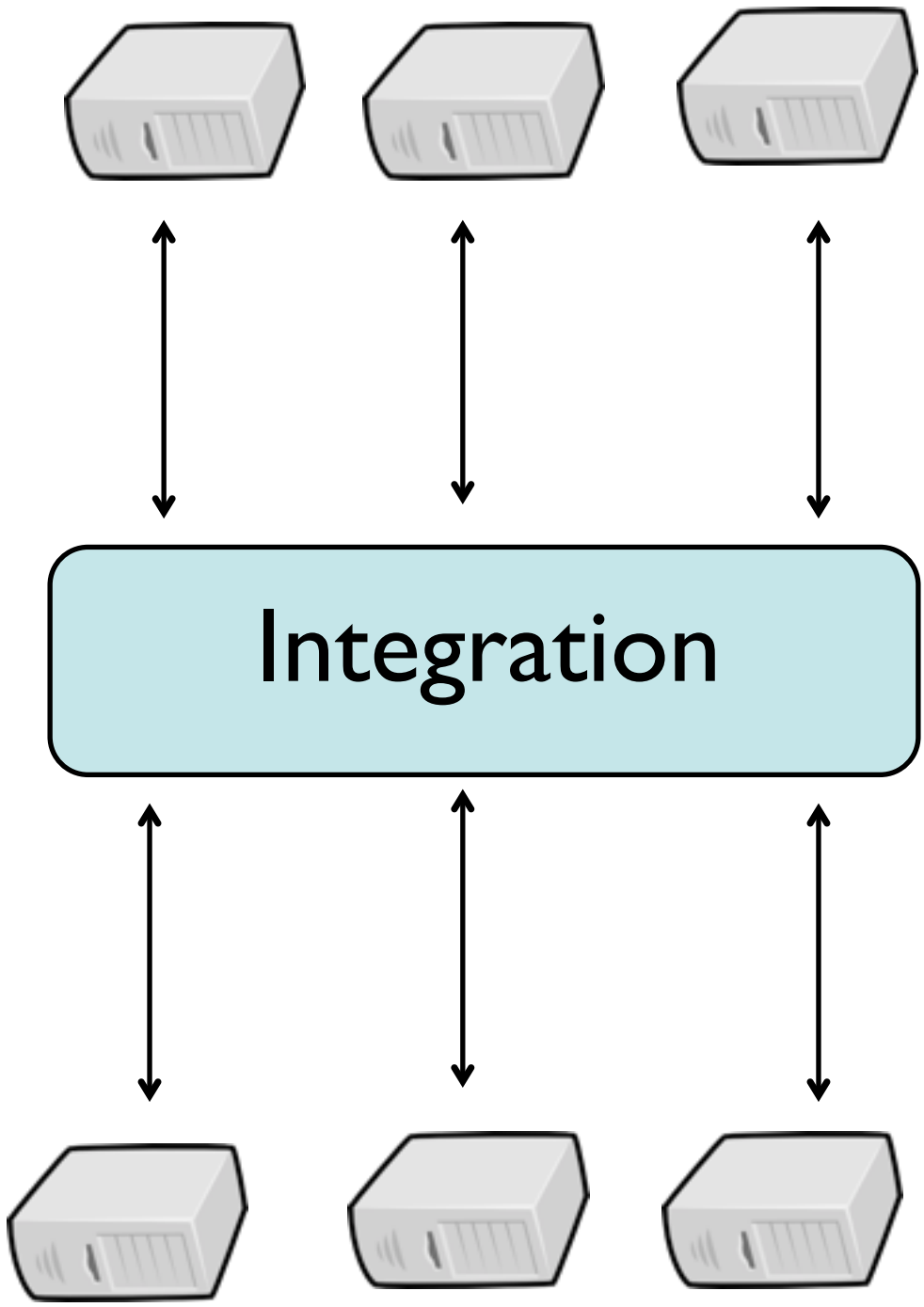
- échanges de fichiers
- web services
- base de données partagées
- messages
- ...

Cas pratique

Je reçois des commandes dans un fichier.

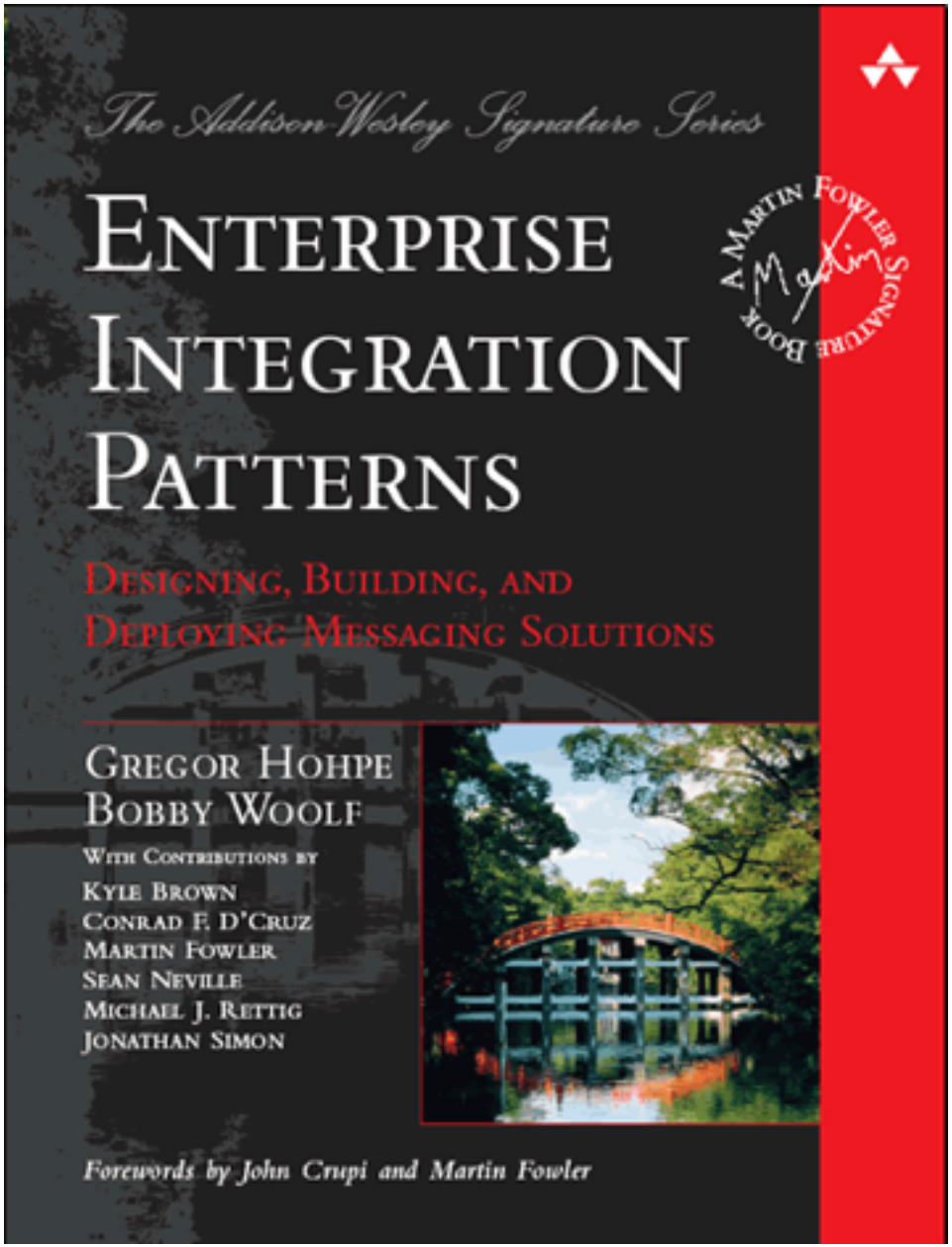
Je veux envoyer un mail de notification au client et écrire cette commande dans une base de données.

Je veux prévenir la facturation de cette nouvelle commande.


Integration

Enterprise Integration Patterns

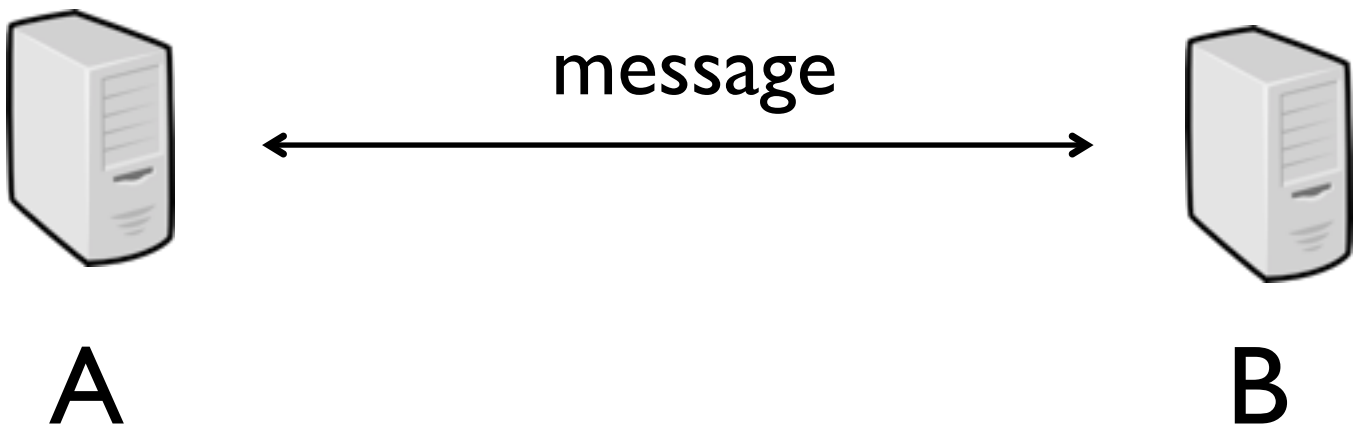
Ensemble de pattern qui répondent à des besoins courants

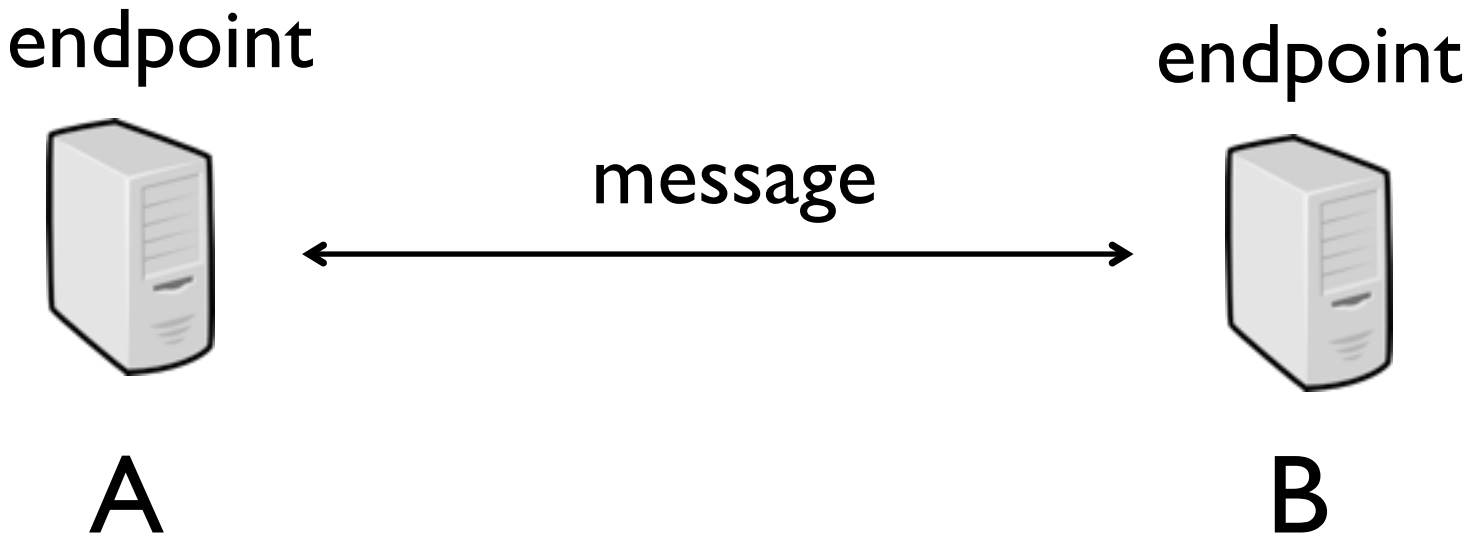

The Addison-Wesley Signature Series
A Martin Fowler Signature Book
Enterprise Integration Patterns
Designing, Building, and Deploying Messaging Solutions
Gregor Hohpe
Bobby Woolf
With Contributions by
Kyle Brown
Conrad F. D'Cruz
Martin Fowler
Sean Neville
Michael J. Rettig
Jonathan Simon
Forewords by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

A

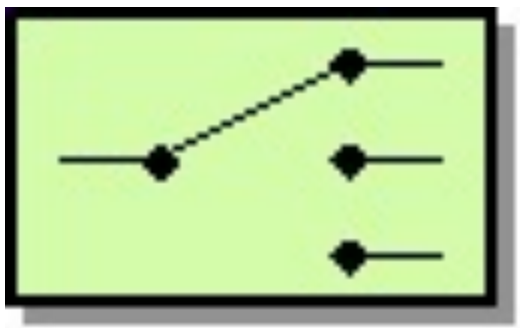
B

route

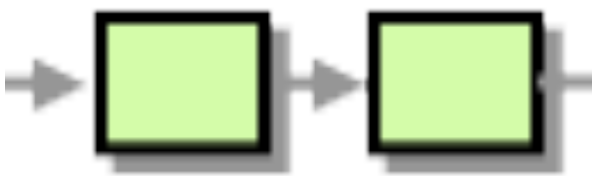



endpoint
endpoint
message
A
B

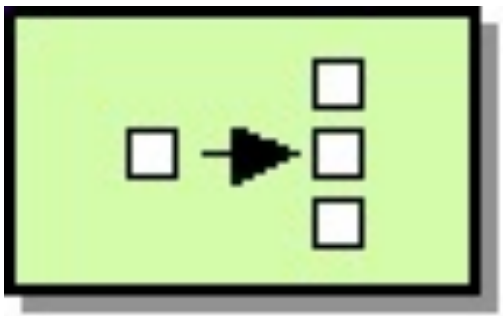
Routeur



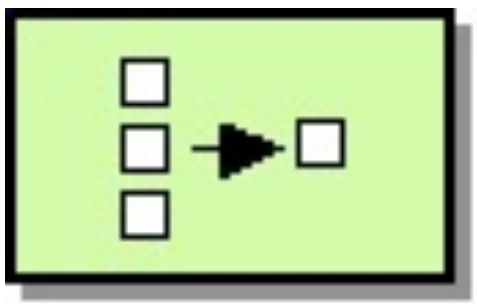
Filtre



Splitter

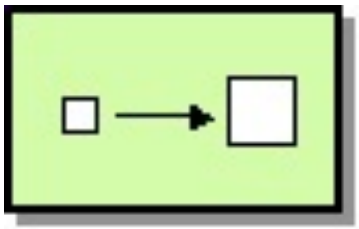


Aggregateur

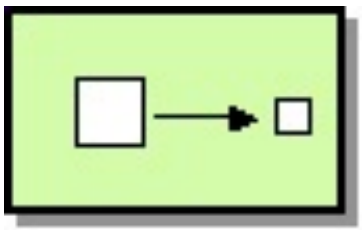


Transformateur

- Enrichissement

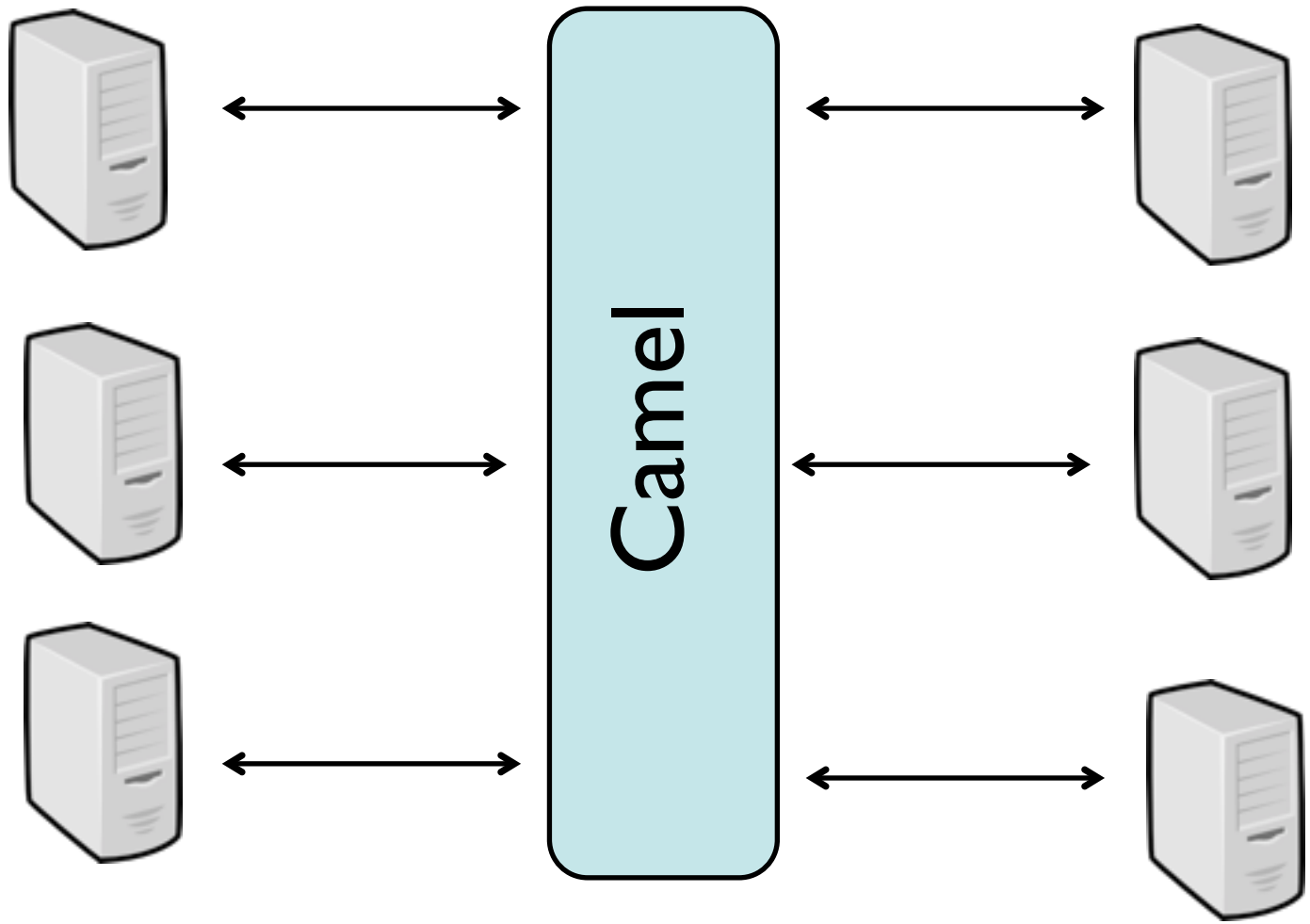


- Filtre



sqli GROUP

- Framework Java
- Implémente les EIPs
- Facile à mettre en place


Camel

Domain Specific Language

Methodes « chainables » pour simplifier l'écriture d'un code

-> code simple à lire

-> autocompletion dans l'IDE

```
public void configure() throws Exception {

    Endpoint jms = endpoint(« jms:commandes »);
    Endpoint file = endpoint(« file:commandes »);
    Endpoint mail = endpoint(« mail:cedric@smtp »);

    //Camel DSL
    from(jms).to(file).to(mail);

}
```

Structure d'un message

Message

ExchangeIn / ExchangeOut

Headers / Body

Composants

- Atom/RSS
- AMQP
- Cache
- CXF
- File
- FTP
- HTTP
- IRC
- JDBC
- Jetty
- JMS
- JPA
- LDAP
- Log
- Lucene
- Mail
- MongoDB
- Netty
- SQL
- Websocket

Exemple de composant

```
from("seda:a")
   .choice()
     .when(header("foo").isEqualTo("bar"))
          .to("seda:b")
     .when(header("foo").isEqualTo("cheese"))
          .to("seda:c")
     .otherwise()
          .to("seda:d");
```

Logger

```
from("file:orders")
        .log(LoggingLevel.INFO, "commande ${body}")
```

Processor

```
from(...)
.process(new Processor() {
        public void process(Exchange exchange) throws
    Exception {
            exchange.getOut().setBody(«hello»);
        }
})
```

Sous route

```
from("file:orders")
        .log(LoggingLevel.INFO, "commande ${body}")
        .to(«direct:route2»);

from(«direct:route2»)
        .to(«direct:route3»);
```

Beaucoup d'autres avantages!

- Politique d'erreur
  - Nombre d'essais
  - Selon le type d'exception
  - Endpoint d'erreur
- Conversion de type automatique