ARAMIS : Apache Camel

Stéphane Deraco stephane.deraco@dsi.cnrs.fr

Agenda

Présentation 15 min

Démo 10 min

Questions 5 min

Présentation

DSI CNRS

- Intégration de données
- > 80 applications inter-connectées
- > 300 flux de données différents

Conception & développement

- Forge logicielle
- Développement interne
- Expertise

Integration de données

Récupérer un fichier du serveur serv1

Récupérer un fichier du serveur serv1

```
scp user@serv1:/path/to/file .
```

Récupérer un fichier du serveur serv1 toutes les 10 minutes, le supprimer de la source, le nommer en local avec un timestamp, et l'envoyer par mail

Récupérer un fichier du serveur serv1 toutes les 10 minutes, le supprimer de la source, le nommer en local avec un timestamp, et l'envoyer par mail

```
# script.sh à mettre en crontab avec */10 * * * *
sftp user@serv1 <<<"
  get /path/to/file localfile
  rm /path/to/file
"
tstamp=$(date +%Y-%m-%d)
mv localfile localfile_$tstamp
mail -s "Fichier important" -a localfile_$tstamp steph@cnrs.fr
```

Récupérer le même fichier, mais :

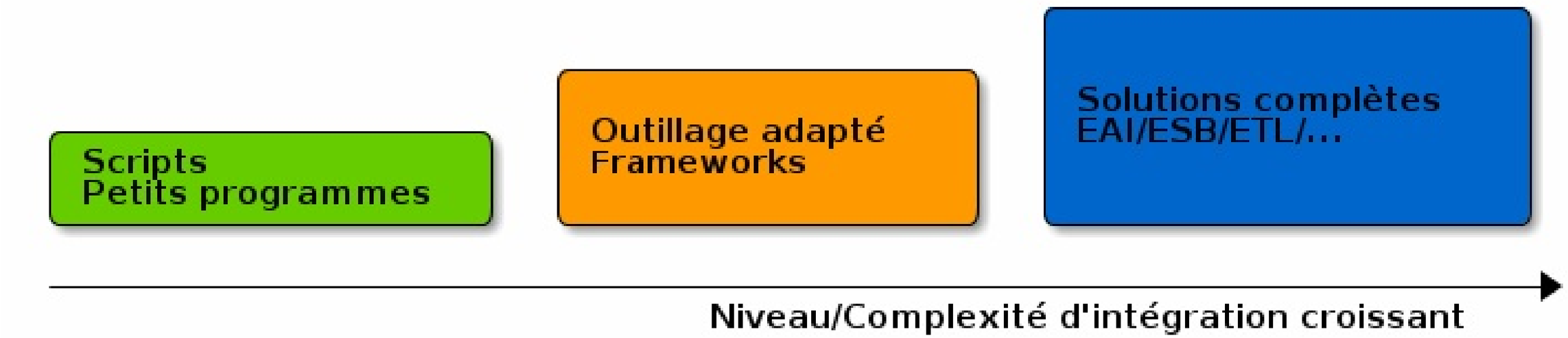
- faire attention à ne pas le traiter s'il est en cours d'écriture sur la source
- le découper sur la balise XML `<data>`
  - envoyer ce bloc de données sur une API HTTP en `POST`
  - mais pas plus de 100 appels / minute
  - si erreur I/O, réessayer 3 fois
- archiver le fichier source dans un sous-répertoire done, si tout s'est bien passé, error sinon

Récupérer le même fichier, mais :

- faire attention à ne pas le traiter s'il est en cours d'écriture sur la source
- le découper sur la balise XML `<data>`
  - envoyer ce bloc de données sur une API HTTP en `POST`
  - mais pas plus de 100 appels / minute
  - si erreur I/O, réessayer 3 fois
- archiver le fichier source dans un sous-répertoire done, si tout s'est bien passé, error sinon

```
# ???
```

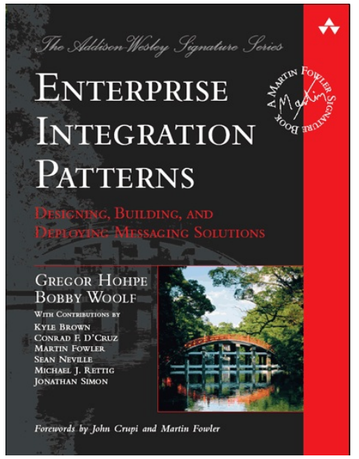
Différents niveaux d'intégration



Apache Camel

Apache Camel

Projet Open Source d'implémentation des Integration Patterns.

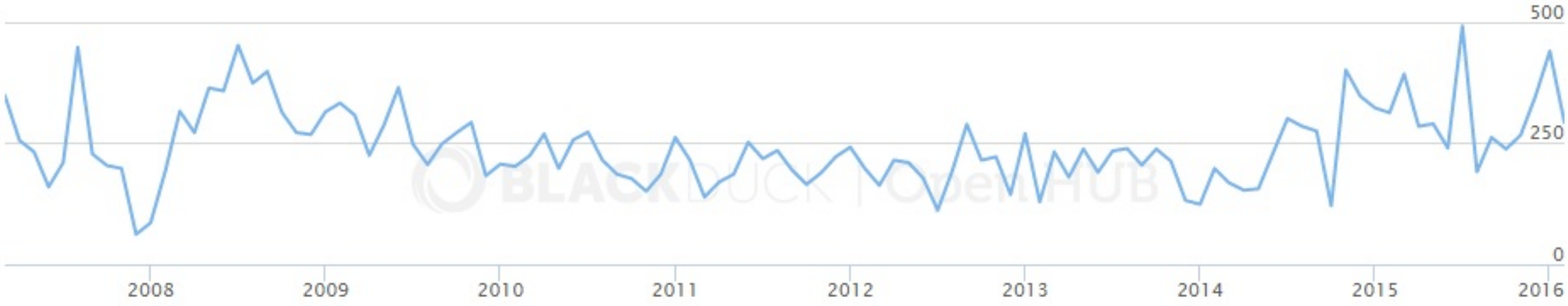


Fournit un Domain Specific Language pour décrire les transformations sur les données.

Source : www.enterpriseintegrationpatterns.com/

Pérennité

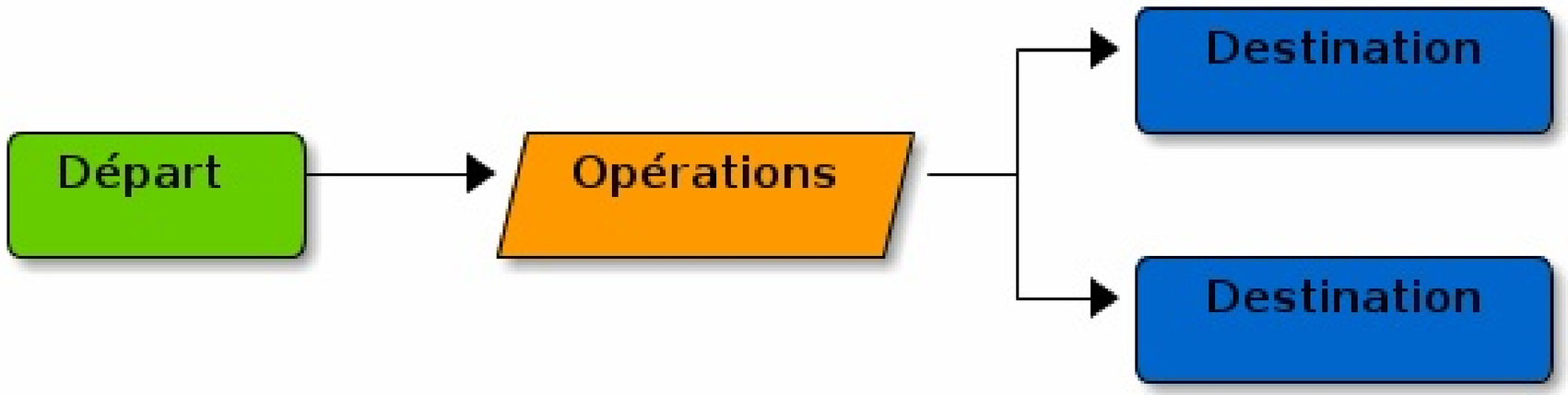
- Premier commit en 2007
- "Top Project" de la fondation Apache en 2009
- Principaux commiters : Red Hat
- Utilisé au LHC



Source : www.openhub.net/p/camel/commits/summary

Integration de données

- Récupérer des données d'une source
- Effectuer des opérations sur ces données
- Envoyer le résultats à une ou plusieurs destinations
- Gérer le reste (archivage, erreurs, …)



Exemple simple

- Transférer les fichiers du répertoire `/data/in`
- Les déplacer dans `/data/out`

Exemple simple

- Transférer les fichiers du répertoire `/data/in`
- Les déplacer dans `/data/out`

```
from("file:///data/in").to("file:///data/out");
```

Exemple simple

- Transférer les fichiers du répertoire `/data/in`
- Les déplacer dans `/data/out`

```
from("file:///data/in").to("file:///data/out");
```

Route Camel

Composant ~ URI

```
from("file:///data/in?delay=1000&move=../archive)
```

- `file://` : scheme (nom du composant)
- `/data/in` : ressource
- `delay` et `move` : paramètres (du composant)

Différents composants

Il existe plus de 200 composants
(camel.apache.org/components.html)

File
(S)FTP(S)
HTTP
LDAP
Mongo
...

Nagios
SMTP/POP3/IMAP
WebServices
RDBMS
TCP/UDP
...

Anatomie d’un échange

Un échange est constitué de :

- Messages
- Properties
- Exception

Un message est constitué de :

- Body
- Headers
- Attachments

Route plus complexe (exemple du début)

```
from("sftp://user@serv1/path/to?" ❶
  + "fileName=file" ❷
  + "&move=done&moveFailed=error" ❸
  + "&delay=600000" ❹
  + "&readLock=changed") ❺
```

❶ Composant SFTP

❷ Nom du fichier

❸ Répertoire d'archive (sur le SFTP) si **OK** ou **KO**

❹ Délai (10 min)

❺ Ne pas consommer si le fichier est en cours d'écriture

suite

```
String sftp = "..."; ❶
from(sftp)
   .split().tokenizeXML("data").streaming() ❷
   .throttle(100).timePeriodMillis(60_000) ❸
   .to("http4://server/rest/api") ❹
```

❶ Notre composant sur la slide précédente

❷ Découpage sur l'élément `<data>` ; `streaming` permet de gérer les gros fichiers

❸ Pas plus de 100 appels par minute

❹ Si des données sont à envoyer (body ≠ null), alors le verbe HTTP est `POST` par défaut

suite et fin

```
onException(IOException.class) ❶
   .maximumRedeliveries(3) ❷
   .log("Oups").handled(true); ❸

from(sftp).split().tokenizeXML("data").streaming()
   .throttle(100).timePeriodMillis(60_000)
   .to("http4://server/rest/api");
```

❶ Si une erreur I/O survient

❷ Réessayer 3 fois

❸ Si erreur au bout de 3 fois, on loggue, et on ne remonte pas l'erreur à la source (ici, le composant SFTP)

Autre fonctionnalités

Splitter

```
.split(body(String.class)).tokenize("\n")
.split(xpath("//data"))
.split().parallelProcessing()
.split().streaming()
```

- format (CSV, XML, JSON, ...)
- parallélisme
- groupes de N
- pluggable splitters

Aggregator

```
.aggregate(header("correlationId"))
.aggregate(xpath("/item/@id", String.class))
.aggregate(new
GroupedExchangeAggregationStrategy())
   .constant(true).completionSize(5)
```

- sur la taille, le temps, ...
- pluggable aggregation strategies

Sequenceur, load balancing

```
.resequence(body()).batch().size(10).timeout(2000


from("timer:plop?period=100")
   .loadBalance().roundRobin()
   .to("direct:A", "direct:B")
```

Filtre, choix

```
from("direct:start")
   .filter().xpath("/person[@name='Bob']")
   .choice()
      .when(header("id").isEqualTo("foo"))
         .to("direct:foo")
      .when(header("id").isEqualTo("bar"))
         .to("direct:bar")
      .otherwise()
         .to("direct:other");
```

Passage d'un format à un autre

- Convertir le body dans différents formats automatiquement
- Exemples : CSV, XML, GZIP, JSON, …
- Si besoin, possible de forcer une conversion
- Possible d'écrire ses propres converters

Autre

- Configuration (propriétés) externalisée
- Chiffrage
- Supervision (JMX)
- RoutePolicy (gestion du cycle de vie d'une route)
- Transactions
- Intercepteurs (~ Aspects)
- Notificateurs d'événements
- Très extensible

Comment le faire tourner

- Programme Java standalone
- Embarqué dans votre programme Java
- Tomcat / JBoss / …
- Conteneur OSGi
- Spring (et Spring Boot)

Démo

très actif
très extensible
nombreux composants
ajout de fonctionnalités

Merci !

Questions ?