Enterprise Service Bus

Daniel Hagimont

IRIT/ENSEEIHT
2 rue Charles Camichel - BP 7122
31071 TOULOUSE CEDEX 7

Daniel.Hagimont@enseeiht.fr
http://hagimont.perso.enseeiht.fr

Intégration - besoins

- Briques logicielles (applications)
  - A gros grain
  - Distribuées
  - Technologies différentes (protocoles, systèmes, API ...)
  - Après le développement
- Collaboration/Intégration
  - Communication en réparti
  - Adaptation des interfaces et des données
  - Schémas de collaboration complexes (pas que client-serveur)

Problématique

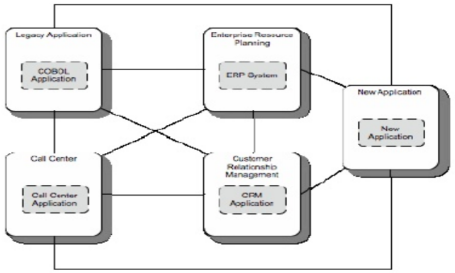
- Depuis 20 ans, les DSI se heurtent aux problèmes de
  - Intégrer des applications hétérogènes
  - Construire des architectures logicielles complexes
  - Les maintenir
- Avec des applications qui n'ont pas été prévues pour le faire

Intégration vs interopérabilité

- Définition : L'interopérabilité est la capacité pour un système d'échanger de l'information et des services dans un environnement technologique et organisationnel hétérogène (IEEE, 1990)
- L'interopérabilité peut être assurée par
  - Le développeur (CORBA, RPC, RMI)
  - L'intégrateur (les applications existent déjà)

Intégration point à point

- Technologies adhoc (différentes à travers le temps)
- *The accidental architecture*
- Effet spaghetti

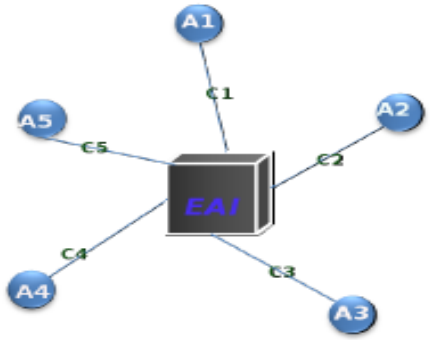


Les ETL (Extract, Transform, Load)

- Solution la plus populaire
- Exportation des données, adaptation et injection dans d'autres applications
- En mode batch (souvent la nuit)
- Problème de latence de mise à jour

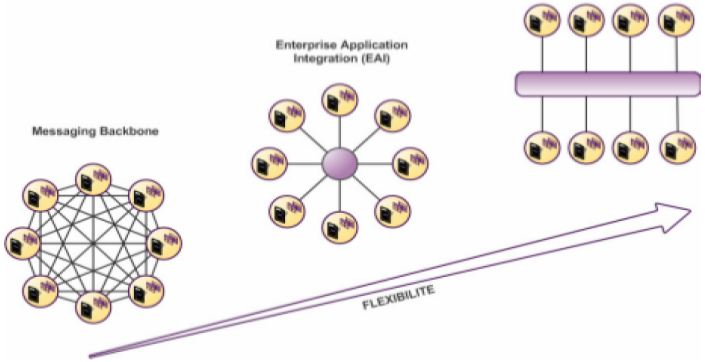
Les EAI (Enterprise Application Integration)

- Comme une multiprise
  - Un connecteur par application
  - L'EAI route les messages entre les applications



ESB (Enterprise Service Bus)

- Un EAI décentralisé
- Utilisation de standards (XML, WS, JMS ...)



Ce qui fait un ESB

- Un bus (MOM)
- Des données (souvent XML)
- Des adaptateurs/connecteurs (WS, ...)
- Un flot de contrôle (routage)

- Objectif : favoriser l'interconnexion

ESB : les produits

- Propriétaires
  - BEA Aqualogic (acheté par Oracle)
  - IBM WebSphere Enterprise Service Bus
  - Sonic ESB de Progress Software
  - Cape Clear (spinoff de IONA)
- OpenSource
  - Mule
  - Apache ServiceMix
  - Jboss ESB
  - OW2 Petals (Toulouse!)

Mule

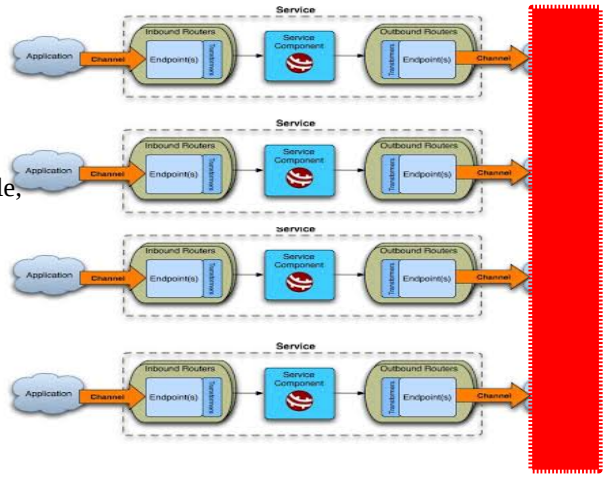
- Mule is a Java-based enterprise service bus (ESB) and integration platform that allows developers to quickly and easily connect applications to exchange data following the service-oriented architecture (SOA) methodology. Mule enables easy integration of existing systems, regardless of the different technologies that the applications use, including JMS, Web Services, JDBC, HTTP, and more.

What Mule ESB does

- Decouples business logic
- Location transparency
- Transport protocol conversion
- Message transformation
- Message routing
- Message enhancement
- Reliability (transactions)
- Security
- Scalability

Overall view

Generally File, WS, mail, JDBC ...

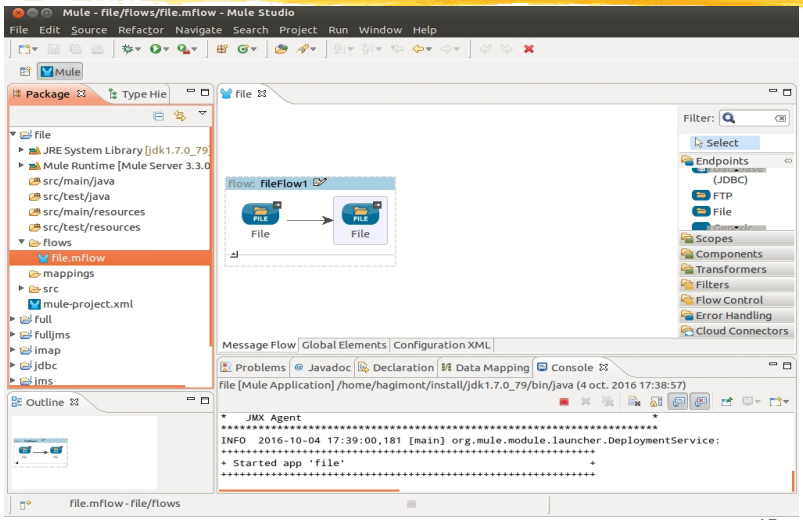


Generally MOM + XML

Mule concepts

- Endpoints
  - Channel for sending or receiving data
- Scopes
  - Processing blocks : polling, synchronizing, grouping ...
- Components
  - Custom logic
- Transformers
  - Data conversion
- Filters
  - Filtering messages in flows
- Flow controls
  - Routing messages in different branches of the flow
- Error handlers

Mule Studio

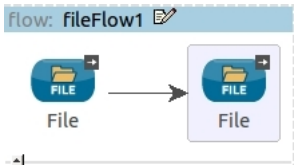


First example

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:file="http://www.mulesoft.org/schema/mule/file"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
xmlns:spring="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="CE-3.3.0" xsi:schemaLocation="
     http://www.mulesoft.org/schema/mule/file
     http://www.mulesoft.org/schema/mule/file/current/mule-file.xsd
     http://www.springframework.org/schema/beans
     http://www.springframework.org/schema/beans/spring-beans-current.xsd
     http://www.mulesoft.org/schema/mule/core
     http://www.mulesoft.org/schema/mule/core/current/mule.xsd ">

     <flow name="fileFlow1" doc:name="fileFlow1">
          <file:inbound-endpoint path="/tmp/in" responseTimeout="10000" doc:name="File"/>
          <file:outbound-endpoint path="/tmp/out" responseTimeout="10000" doc:name="File"/>
     </flow>
</mule>
```



Endpoints

- Examples
  - <file:inbound-endpoint path="/tmp/in" responseTimeout="10000" doc:name="File"/>
  - <jms:outbound-endpoint queue="MyQueue" connector-ref="Active_MQ" doc:name="JMS"/>
- Endpoints
  - Inbound and outbound
    - Ajax, JDBC, FTP, File, HTTP, JMS, RMI, SSL, TCP, UDP, VM ...
  - Inbound only
    - IMAP, POP3, Servlet, Twitter ...
  - Outbound only
    - SMTP ...
- New endpoints can be developed

Transformers

- Default transformers (associated with endpoint)
  - jmsmessage-to-object-transformer
  - byte-array-to-string-transformer
- Custom transformers
  - Override default transformers
  - object-to-xml, xml-to-object, json-to-object ...
- New transformers can be developed

```
public class MyTransformer extends AbstractTransformer {
        public Object doTransform(Object src, String encoding) throws TransformerException {
        }
}
```


Components

- Customize the message flows

```
public class Filter implements Callable {
        public Object onCall(MuleEventContext eventContext) throws Exception {
                person p = (person)eventContext.getMessage().getPayload();
                return null;
        }
}
```


Transformer-component example

```
<flow name="xmlFlow1" doc:name="xmlFlow1">
        <file:inbound-endpoint path="/tmp/in" responseTimeout="10000" doc:name="File"/>
        <mulexml:xml-to-object-transformer doc:name="XML to Object"/>
        <component class="Filter" doc:name="Java"/>
        <mulexml:object-to-xml-transformer doc:name="Object to XML"/>
        <file:outbound-endpoint path="/tmp/out" responseTimeout="10000" doc:name="File"/>
</flow>
```

Custom component

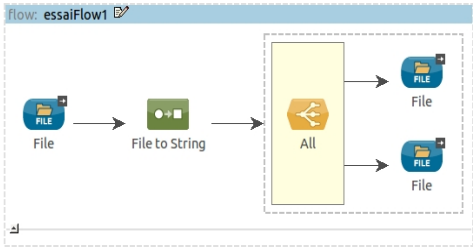
Use xstream (xml to Java bean conversion)

Flow controls

- All
  - Sends messages to all routes
- Choice
  - Routes messages based on expressions
- First successful
  - Sends a message to a list of routes until one is processed successfully
- Round robin
  - Send a message to the next route in the circular list of route
- ...

Flow control example

```
<flow name="essaiFlow1" doc:name="essaiFlow1">
    <file:inbound-endpoint path="/tmp/in" responseTimeout="10000" doc:name="File"/>
    <file:file-to-string-transformer doc:name="File to String"/>
    <all doc:name="All">
        <processor-chain>
            <file:outbound-endpoint path="/tmp/out1" responseTimeout="10000" doc:name="File"/>
        </processor-chain>
        <processor-chain>
            <file:outbound-endpoint path="/tmp/out2" responseTimeout="10000" doc:name="File"/>
        </processor-chain>
    </all>
</flow>
```



Global elements

- Global elements have to be declared to configure some mule elements
  - JMS connector

```
<jms:activemq-connector name="Active_MQ" specification="1.1" username="admin"
password="admin" brokerURL="tcp://localhost:61616" validateConnections="true"
doc:name="Active MQ"/>
```

  - IMAP connector

```
<imaps:connector name="IMAP" validateConnections="true" checkFrequency="1000"
doc:name="IMAP">
    <imaps:tls-client path="*" storePassword="*"/>
    <imaps:tls-trust-store path="*" storePassword="*"/>
</imaps:connector>
```


Global elements

- Global elements have to be declared to configure some mule elements
  - Data source (with a bean)

```
<spring:beans>
    <spring:bean id="dataSource" name="dataSource"
        class="org.enhydra.jdbc.standard.StandardDataSource"
        destroy-method="shutdown">
            <spring:property name="driverName" value="org.hsqldb.jdbcDriver"/>
            <spring:property name="url" value="jdbc:hsqldb:hsql://localhost/xdb"/>
            <spring:property name="user" value="sa"/>
    </spring:bean>
</spring:beans>
```

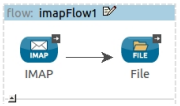
  - Database connection

```
<jdbc:connector name="Database__JDBC_" dataSource-ref="dataSource"
validateConnections="true" queryTimeout="-1" pollingFrequency="0"
doc:name="Database (JDBC)"/>
```

IMAP / SMTP examples

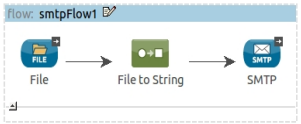
- IMAP

```
<flow name="imapFlow1" doc:name="imapFlow1">
    <imaps:inbound-endpoint host="imap.gmail.com" port="993" user="tpdhlogin"
    password="tpdhpasswd" responseTimeout="10000" connector-ref="IMAP" doc:name="IMAP"/>
    <file:outbound-endpoint path="/tmp/out" responseTimeout="10000" doc:name="File"/>
</flow>
```



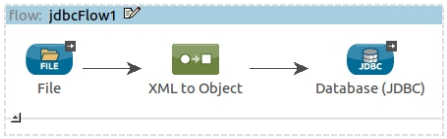
- SMTP

```
<flow name="smtpFlow1" doc:name="smtpFlow1">
    <file:inbound-endpoint path="/tmp/in" responseTimeout="10000" doc:name="File"/>
    <file:file-to-string-transformer doc:name="File to String"/>
    <smtp:outbound-endpoint host="mail.enseeiht.fr" port="587" user="hagimont"
        to="hagimont@enseeiht.fr" from="hagimont@enseeiht.fr" subject="email from Mule"
        replyTo="tpdhlogin@gmail.com" responseTimeout="10000" connector-ref="SMTP"
        doc:name="SMTP"/>
</flow>
```



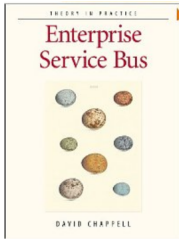
JDBC example

```
<flow name="jdbcFlow1" doc:name="jdbcFlow1">
    <file:inbound-endpoint path="/tmp/in" responseTimeout="10000" doc:name="File"/>
    <mulexml:xml-to-object-transformer doc:name="XML to Object"/>
    <jdbc:outbound-endpoint exchange-pattern="one-way" queryKey="insertion"
        queryTimeout="-1" connector-ref="Database__JDBC_" doc:name="Database (JDBC)">
        <jdbc:query key="insertion" value="insert into accounts values
                (#[payload.nom], #[payload.prenom], #[payload.email]);"/>
    </jdbc:outbound-endpoint>
</flow>
```

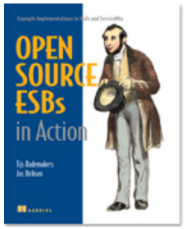


Bibliography

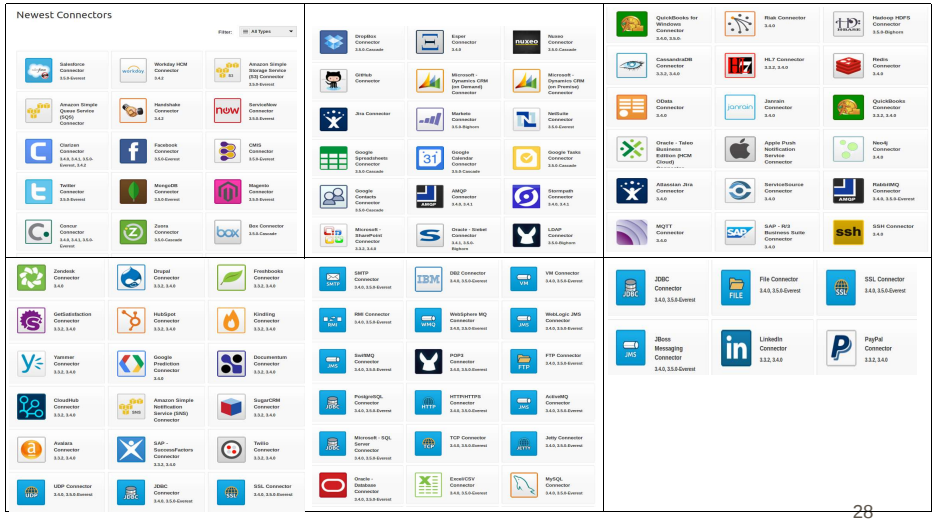
- General
  - Enterprise Service Bus: Theory in Practice (O'Reilly)



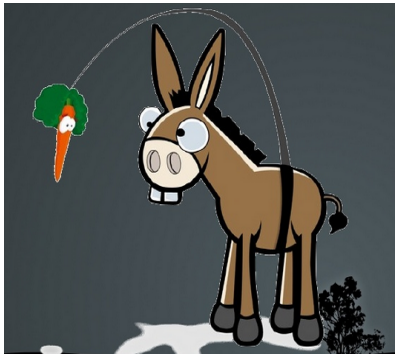
- Technical
  - Open-Source ESBs in Action (Manning)



Connector market place

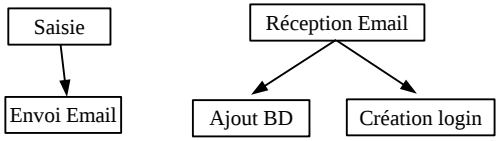


...

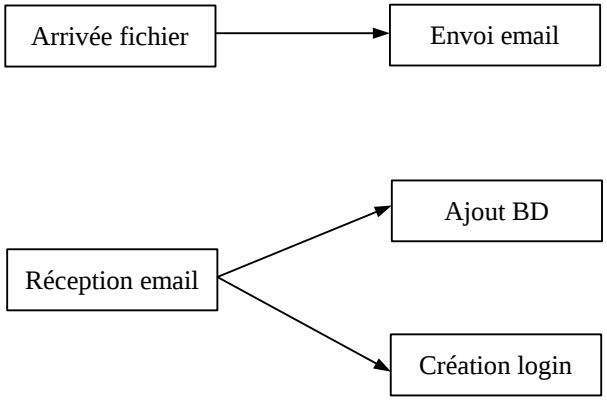


TP -ESB - Mule

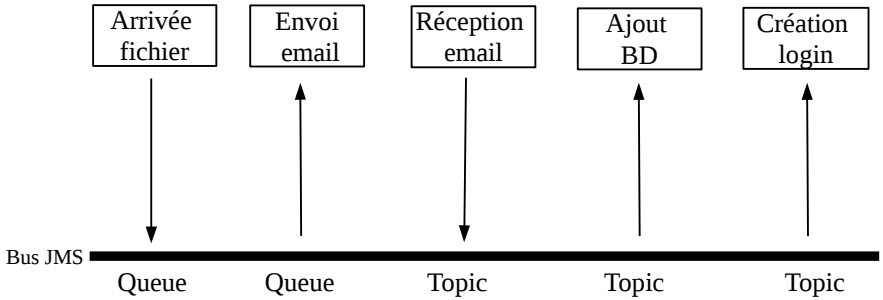
- Scénario d'inscription d'un étudiant
  - Saisie des données (application Saisie)
    - Permet d'exporter les données en XML
  - Réception par email et validation par une secrétaire
    - Validation en répondant à l'email
  - Intégration dans la liste gérée dans le serveur Web
    - Ajout dans une BD
  - Création d'un login pour l'étudiant
    - Disponible sous la forme d'un web service



Première intégration (spaghetti)



Deuxième intégration (ESB)



Différents exemples