Antonio Musarra (@antonio_musarra)
Software e Architectural Consultant
Author & Editor Antonio Musarra's Blog

Liferay & Salesforce

Come integrare Salesforce nel contesto OSGi di Liferay

LIFERAY

Italy User Group Meeting - #LRUGITALIA
September 15, 2017 | 10.00h | Bologna, Italia

Sommario

- Overview della soluzione d'integrazione
- Overview delle API di Salesforce
- Come realizzare il bundle OSGi del client Salesforce
- Come installare il bundle OSGi del client Salesforce
- Come utilizzare il bundle OSGi del client Salesforce
- Realizzare un'applicazione di esempio (Comandi Gogo Shell)
- Un caso d'integrazione

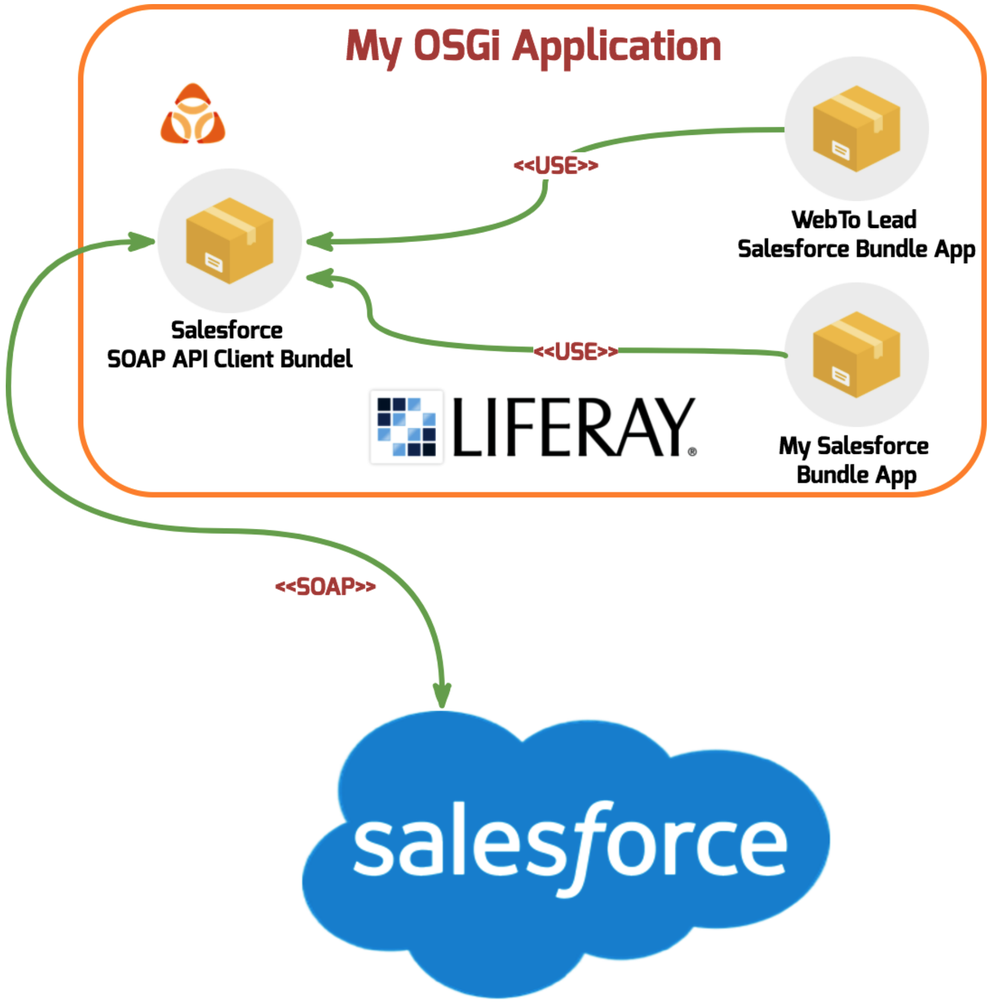
1. Overview soluzione d'integrazione

Cosa vogliamo ottenere dalla nostra soluzione d'integrazione?

1. Overview soluzione d'integrazione

- Vogliamo riuscire a far colloquiare **Liferay 7 & Salesforce.com**
- Non vogliamo utilizzare librerie da mettere a destra a sinistra, sopra o sotto
- Vogliamo cose da installare e gestire in modo semplice
- Vogliamo cose che siano semplici da configurare
- Vogliamo cose che siano standard (**che funzioni anche fuori Liferay**)
- Vogliamo scrivere meno codice possibile
- Vogliamo che sia semplice
- **Insomma, vogliamo integrare solo a colpi di bundle OSGi :-)**

1. Overview soluzione d'integrazione

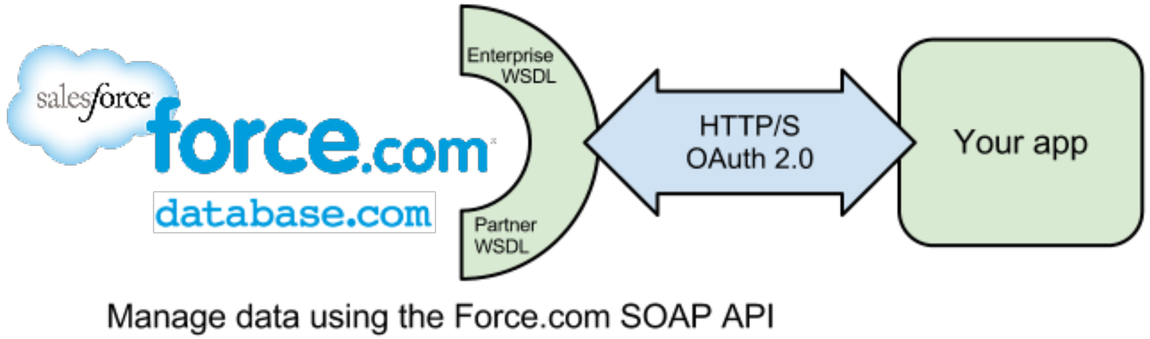


- Liferay & Salesforce.com utilizzeranno **SOAP** come protocollo di comunicazione
- Dobbiamo realizzare il bundle OSGi che implementa il client verso Salesforce.com e che parla SOAP
- Il bundle OSGi del client deve **esportare** il set di **API Java** di Salesforce.com
- Ogni altro bundle OSGi può **consumare le API Java** di Salesforce.com e farci ciò che vuole
- **Ogni altro bundle OSGi non sa nemmeno cosa sia SOAP :-)**

2. Overview delle API di Salesforce

- Salesforce.com espone un set di API via **SOAP** e **REST** che consentono a sistemi esterni di poter interagire con esso.
- Salesforce.com mette a disposizione il tool **Force.com Web Service Connector (WSC)** che crea per noi gli stub Java necessari per comunicare via SOAP
- Il nostro bundle OSGi esporrà il seguente set di API Java che potranno poi essere utilizzate da qualunque altro bundle che necessiterà d'interagire con Salesforce.com:
  - com.sforce.soap.partner.*
  - com.sforce.soap.enterprise.*
  - com.sforce.async.*
  - com.sforce.bulk.*
  - com.sforce.ws.*

2. Overview delle API di Salesforce



Manage data using the Force.com SOAP API

- Con le API SOAP di Salesforce.com possiamo fare **ogni operazione che siamo in grado di fare dalla GUI**
- Livello di sicurezza gestito da Salesforce.com con un alto livello di configurazione

- Salesforce.com mette a disposizione **due differenti SOAP API WSDLs**
- **Enterprise WSDL**: Utilizzato per sviluppare applicazioni client per singole organizzazioni perché molto tipizzato
- **Partner WSDL**: Utilizzato per sviluppare applicazioni client guidate dai meta-dati e dinamiche per natura, applicazioni in genere non legate alla singola organizzazione

3. Realizzare il bundle OSGi del client Saleforce

- Siamo in un contesto OSGi, dobbiamo quindi realizzare il client in forma di **bundle OSGi**
- Il tool **Force.com Web Service Connector (WSC)** creerà per noi e in modo corretto gli stub, sia per le API SOAP Enterprise sia per le API SOAP Partner
- Per la creazione del bundle andiamo su Apache Maven, in particolare:
    - **Exec Maven Plugin**: Utilizziamo questo plugin per eseguire il tool **WSC** per generare sia il Partner sia l'Enterprise SOAP API stubs
    - **BND Tools**: Utilizziamo questo plugin per creare il nostro bundle OSGi attraverso l'utilizzo delle direttive incluse nel file di configurazione **bnd.bnd**

3. Realizzare il bundle OSGi del client Saleforce

```
Bundle-Name: Salesforce SOAP API Client OSGi Bundle
Bundle-SymbolicName: it.dontesta.labs.liferay.salesforce.client.soap
Bundle-Description: Salesforce SOAP API Client OSGi bundle build with Force.com Web Service Connector (WS
Bundle-Version: 1.0.0-SNAPSHOT
Bundle-Category: Salesforce
Bundle-ContactAddress: https://www.dontesta.it/contatti/
Bundle-Developers: antonio.musarra@gmail.com
Bundle-DocURL: https://www.dontesta.it/salesforce-client-soap
Bundle-Copyright: Antonio Musarra's Blog (c) 2017
Bundle-License: MIT License
Bundle-Icon: Console
Bundle-Vendor: Antonio Musarra's Blog
Bundle-RequiredExecutionEnvironment: JavaSE-1.8

Import-Package: \
!com.google.appengine.api.urlfetch,\
!org.codehaus.jackson.*,\
!org.stringtemplate.v4.*,\
!org.antlr.runtime.*,\
!org.antlr.stringtemplate.*,\
!org.apache.commons.*,\
*

Export-Package: \
com.sforce.soap.partner.*;version="1.0.0",\
com.sforce.soap.enterprise.*;version="1.0.0",\
com.sforce.async.*;version="1.0.0",\
com.sforce.bulk.*;version="1.0.0",\
com.sforce.ws.*;version="1.0.0"

-includeresource: \
@libs/partner-${project.version}.jar,\
@libs/enterprise-${project.version}.jar
```

- Definizione dei package Java delle **API di Salesforce.com da esportare**, in questo modo rendiamo **pubblico l'accesso** ad ogni altro bundle che vive all'interno del container OSGi

- Dichiariamo che gli stub **Enterprise** e **Partner** che sono generati in fase di build devono essere inclusi nel bundle OSGi del client

3. Realizzare il bundle OSGi del client Saleforce

- Il build del progetto con il comando **`mvn clean package`** genera il bundle OSGi del client di Salesforce.com. All'interno della directory target ci sarà quindi il file **salesforce-client-soap-1.0.3-SNAPSHOT.jar**
- Il bundle così generato è pronto per essere installato su Liferay o **su qualunque altro container OSGi** che sia conforme alle specifiche OSGi R6
- L'intero progetto del bundle è disponibile sul mio repository github con il nome salesforce-client-soap
- Sul Maven Central Repository è disponibile la release 1.0.2 del **Salesforce SOAP API Client** OSGi bundle. Bundle bello e pronto per essere installato

3. Realizzare il bundle OSGi del client Saleforce

- Il bundle pubblicato sul Maven Central Repository può essere utilizzato per applicazioni che seguono sia il modello **Partner** sia il modello **Enterprise** (che utilizza le API standard)

- Il bundle pubblicato sul Maven Central Repository è indicato per lo sviluppo di applicazioni che seguono il modello **Partner**

- Ogni organizzazione può ottenere in modo semplice il bundle OSGi del client che **rispecchia il proprio modello *enterprise*** implementato sulla propria istanza di Salesforce.com

3. Realizzare il bundle OSGi del client Saleforce

- Se vogliamo costruire il bundle OSGi del client con il proprio **Enterprise WSDL** occorre seguire i seguenti passi:
    - Generare il WSDL Enterprise per l'organizzazione (ad esempio: ***enterprise_antonio_musarra_blog.wsdl***). Le indicazioni su come fare sono spiegate in Generate or Obtain the Web Service WSDL
    - Clonare il progetto salesforce-client-soap
    - Eseguire il build del progetto specificando il vostro WSDL Enterprise, così come indicato dal comando a seguire:

```
$ mvn -Dsalesforce.wsdl.enterprise.path=$ABSOLUTE_PATH_ENTEPRISE_WSDL clean package
```

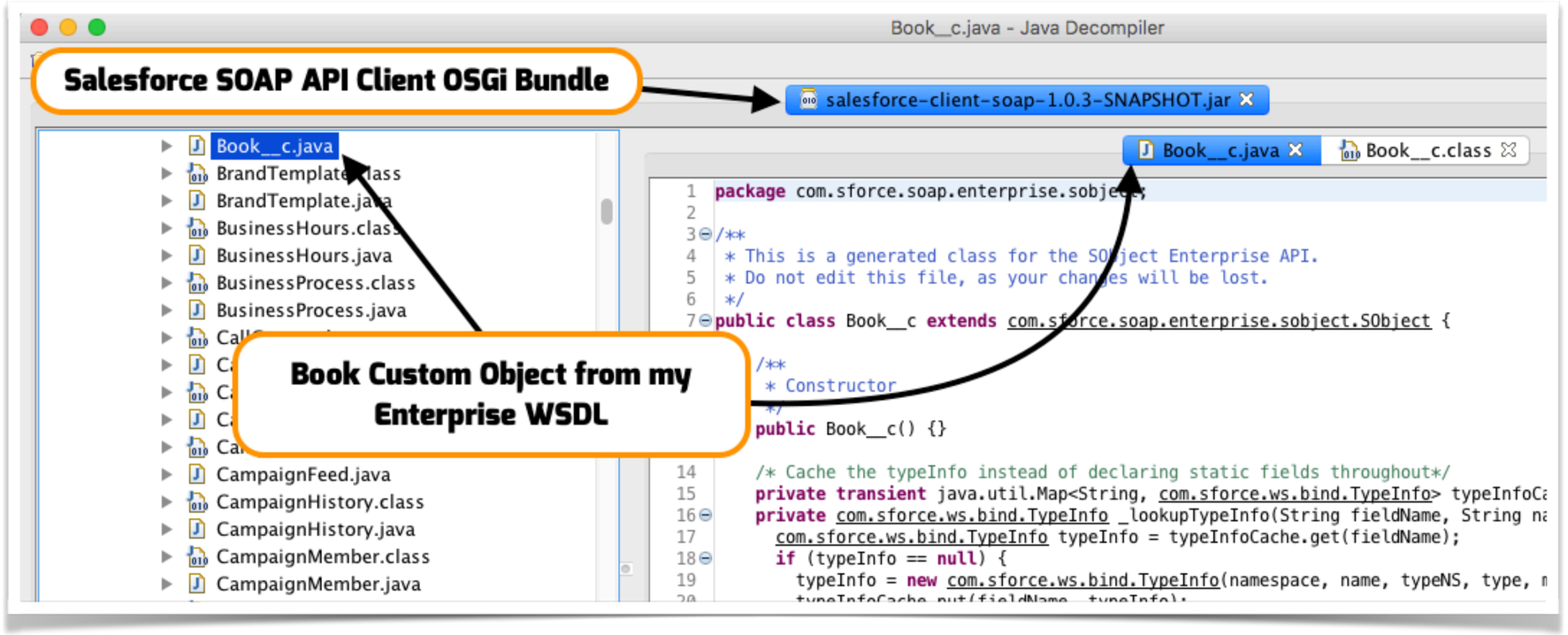
3. Realizzare il bundle OSGi del client Saleforce

A seguire sono mostrati i passi necessari per eseguire la build (dai sorgenti) e ottenere il bundle OSGi del client Salesforce.com. L'ultimo commando *maven* genera il bundle OSGi specificando il WSDL Enterprise della propria organizzazione.

```
$ git clone https://github.com/amusarra/salesforce-client-soap.git
$ cd salesforce-client-soap/
$ mvn clean package
$ mvn -Dsalesforce.wsdl.enterprise.path=$ABSOLUTE_PATH_ENTEPRISE_WSDL clean package
```

3. Realizzare il bundle OSGi del client Saleforce

- Il bundle OSGi per la nostra organizzazione conterrà anche le entità proprie (esempio: **Book**) con cui possiamo interagire attraverso le API Java esportate



4. Come installare il bundle OSGi del client Salesforce

- L'installazione del bundle OSGi è davvero semplice, abbiamo due strade:
    - Nel caso in cui dobbiamo realizzare un'applicazione che segue il modello **Partner**, allora basta scaricare il bundle OSGi dal Maven Central Repository e seguire la procedura standard di deploy su Liferay
    - Nel caso in cui dobbiamo realizzare un'applicazione che segue il modello **Enterprise**, dobbiamo seguire la strada indicata in precedenza (ovvero, ottenere il bundle dal nostro Enterprise WSDL); una volta ottenuto il bundle OSGi possiamo seguire la procedura standard di deploy su Liferay

4. Come installare il bundle OSGi del client Salesforce

A seguire un esempio d'installazione via Gogo Shell su Liferay del bundle OSGi. Ricordo comunque, che lo stesso bundle può essere installato anche su Apache Karaf (container OSGi che implementa le specifiche R6)

```
$ telnel localhost 11311

g! install http://repo1.maven.org/maven2/it/dontesta/labs/liferay/salesforce/
client/soap/salesforce-client-soap/1.0.2/salesforce-client-soap-1.0.2.jar

g! lb|grep Salesforce
583|Active     |    10|Salesforce SOAP Client (1.0.2)
```

A questo punto siamo pronti per sviluppare la nostra applicazione d'esempio che colloquia con Salesforce.com grazie ai servizi offerti dal bundle OSGi appena creato.

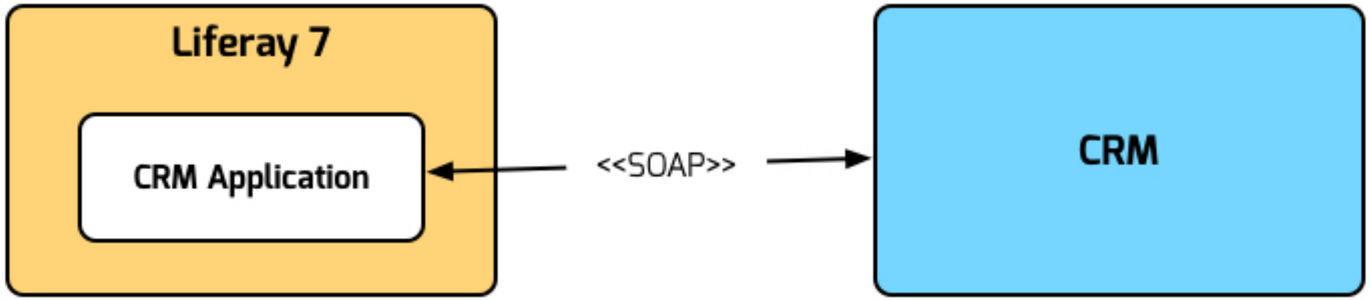
5. Come utilizzare il bundle OSGi del client Salesforce

- Adesso siamo nelle condizioni di poter sviluppare la nostra applicazione su Liferay che interagisce con Salesforce.com
- Una volta installato il bundle OSGi del client, utilizzare le **API Java** di Salesforce.com è davvero semplice; **dobbiamo solo includere la dipendenza sulla nostra applicazione Liferay** (a seguire la dipendenza maven e gradle)

```
<dependency>
 <groupId>it.dontesta.labs.liferay.salesforce.client.soap</groupId>
 <artifactId>salesforce-client-soap</artifactId>
 <version>1.0.2</version>
</dependency>
```

```
compile group:'it.dontesta.labs.liferay.salesforce.client.soap',
name:'salesforce-client-soap', version:'1.0.2'
```

6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)



- Lo schema mostrato in figura è un tipico scenario d'integrazione tra Liferay e un generico CRM, che in questo caso è Salesforce.com

- Come demo, ho sviluppato una semplice applicazione d'esempio (salesforce-client-gogoshell-command) che esegue le seguenti operazioni su Salesforce.com
  - **salesforce:login**: Login to your Salesforce instance.
  - **salesforce:createAccount**: Create account into your Salesforce instance.
  - **salesforce:getNewestAccount**: Query for the newest accounts.
  - **salesforce:loginEnterprise**: Login to your Salesforce instance using the Enterprise Connection.
  - **salesforce:getNewestAccountEnterprise**: Query for the newest accounts using the Enterprise Connection.

6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)

- L'applicazione d'esempio implementa una serie di comandi Gogo Shell
- I comandi implementati eseguono azioni elementari utilizzando le API Java esposte dal bundle **Salesforce SOAP API Client**
- I comandi implementati utilizzano sia le **Partner** sia le **Enterprise** API
- L'applicazione d'esempio è configurabile
- Quest'applicazione è un buon esempio di partenza per sviluppare le proprie applicazioni d'integrazione con Salesforce.com
- Questa applicazione d'esempio funziona anche su **Apache Karaf 4.x**

6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)

```
33  import aQute.bnd.annotation.metatype.Meta;
34
35  /**
36   * @author Antonio Musarra antonio.musarra@gmail.com
37   */
38  @Meta.OCD(
39          id = "it.dontesta.labs.liferay.salesforce.client.command.configuration.SalesforceClientCommandConfiguration",
40          localization = "content/Language",
41          name = "salesforce.client.command.configuration.name"
42  )
43  public interface SalesforceClientCommandConfiguration {
44
45          @Meta.AD(
46                  deflt = "https://login.salesforce.com/services/Soap/u/40.0",
47                  description = "Setting the Salesforce endpoint",
48                  required = false)
49          public String authEndpoint();
50
51          @Meta.AD(
52                  deflt = "https://login.salesforce.com/services/Soap/c/40.0",
53                  description = "Setting the Salesforce endpoint for Enterprise Connection",
54                  required = false)
55          public String authEndpointEnterprise();
56
57          @Meta.AD(
58                  deflt = "/tmp/traceSalesforcePartner.log",
59                  description = "Setting full path of the trace file",
60                  required = false)
61          public String traceFile();
62
```

- Grazie all'uso dei MetaType OSGi e relative annotation, è possibile predisporre una configurazione per la nostra applicazione di esempio.
- É possibile agire sulla configurazione dal pannello di controllo Liferay
- É possibile agire sulla configurazione da file di cfg

6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)

```
54  /**
55   * Gogo Shell Command Series for Salesforce
56   * (Example: create leads, create customers, search, etc.).
57   *
58   * @author Antonio Musarra antonio.musarra@gmail.com
59   */
60  @Component(
61                  configurationPid = "it.dontesta.labs.liferay.salesforce.client.command.configuration.SalesforceClientCommandCon
62                  property = {
63                                  "osgi.command.function=login",
64                                  "osgi.command.function=loginEnterprise",
65                                  "osgi.command.function=createAccount",
66                                  "osgi.command.function=getNewestAccount",
67                                  "osgi.command.function=getNewestAccountEnterprise",
68                                  "osgi.command.scope=salesforce"
69                  },
70                  service = Object.class
71  )
72  @Descriptor("Gogo Shell Command Series for Salesforce "
73                  + "(Example: create leads, create customers, search, etc.).")
74  public class SalesforceClientCommand {
75
76      private static PartnerConnection partnerConnection = null;
77      private static EnterpriseConnection enterpriseConnection = null;
78      private volatile SalesforceClientCommandConfiguration _configuration;
79
```

- ***SalesforceClientCommand*** la classe che implementa i comandi OSGi via Gogo Shell che ci consentono d'interagire con Salesforce.com

6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)

```
80      /**
81           * Login to your Saleforce instance using the Partner Connection
82           *
83           * @param username
84           *            Your Salesforce username
85           * @param password
86           *            Your Saleforce password (Note: append your API Key to Password)
87           * @throws Exception When login failed
88           */
89          @Descriptor("Login to your Salesforce instance using the Partner Connection")
90          public void login(
91                  @Descriptor("The your username") String username,
92                  @Descriptor("The your password + append your API Key") String password
93                  )
94                  throws Exception {
95
96                  boolean success = false;
97
98                  try {
99                          ConnectorConfig config = new ConnectorConfig();
100                         config.setUsername(username);
101                         config.setPassword(password);
102                         config.setAuthEndpoint(_configuration.authEndpoint());
103                         config.setTraceFile(_configuration.traceFile());
104                         config.setTraceMessage(_configuration.traceMessage());
105                         config.setPrettyPrintXml(_configuration.prettyPrintXml());
106
107                         partnerConnection = new PartnerConnection(config);
108                         success = true;
109
```

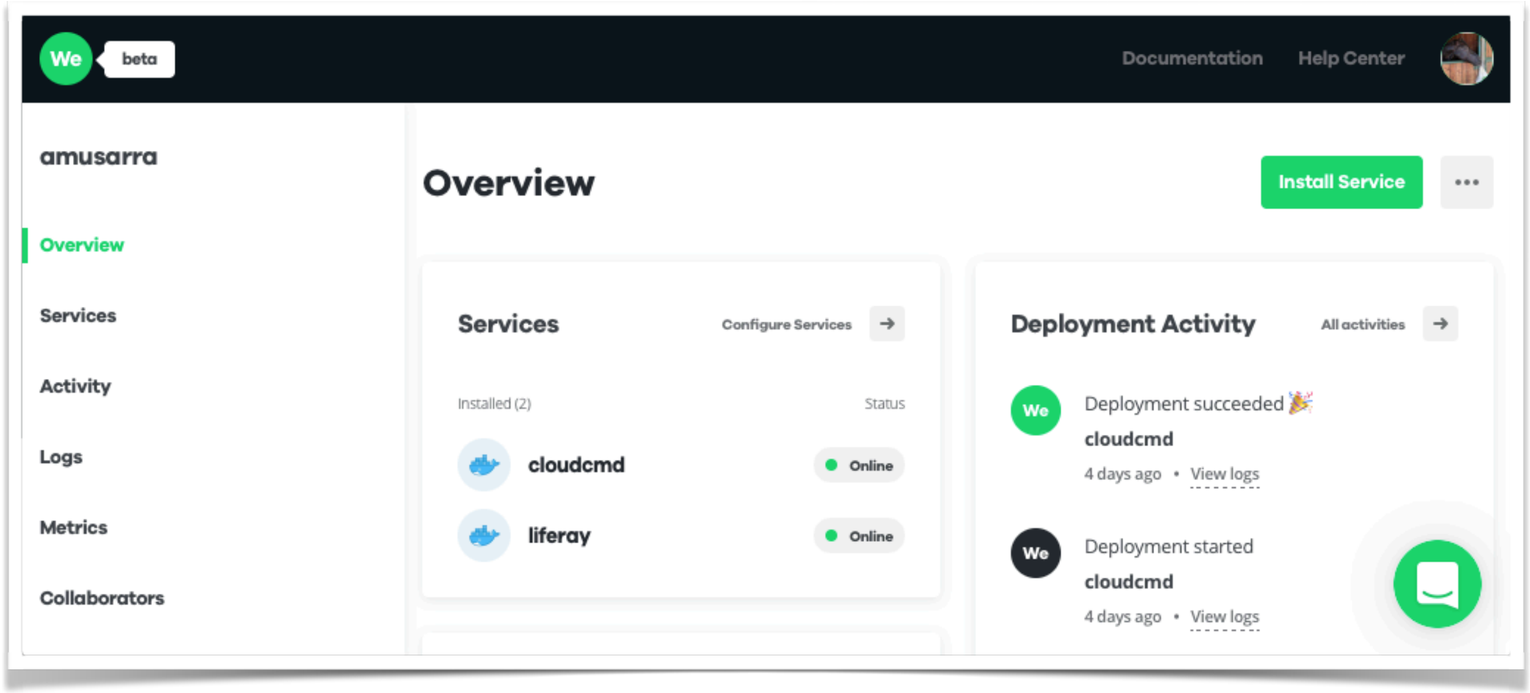
- Implementazione del comando *login* che utilizza le API Java di Salesforce.com esportate e quindi rese pubbliche dal bundle OSGi Salesforce SOAP API Client

6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)

Due interessanti demo che mostrano l'integrazione con Salesforce.com. Basta cliccare sopra per avviare il video.

6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)

Ho approfittato di **@wedeploy** per fare il deploy e test di Salesforce SOAP API Client Bundle e Salesforce Gogo Shell Bundle Application.



Liferay è disponibile all'indirizzo http://liferay-amusarra.wedeploy.io/

CloudCmd è disponibile all'indirizzo https://cloudcmd-amusarra.wedeploy.io

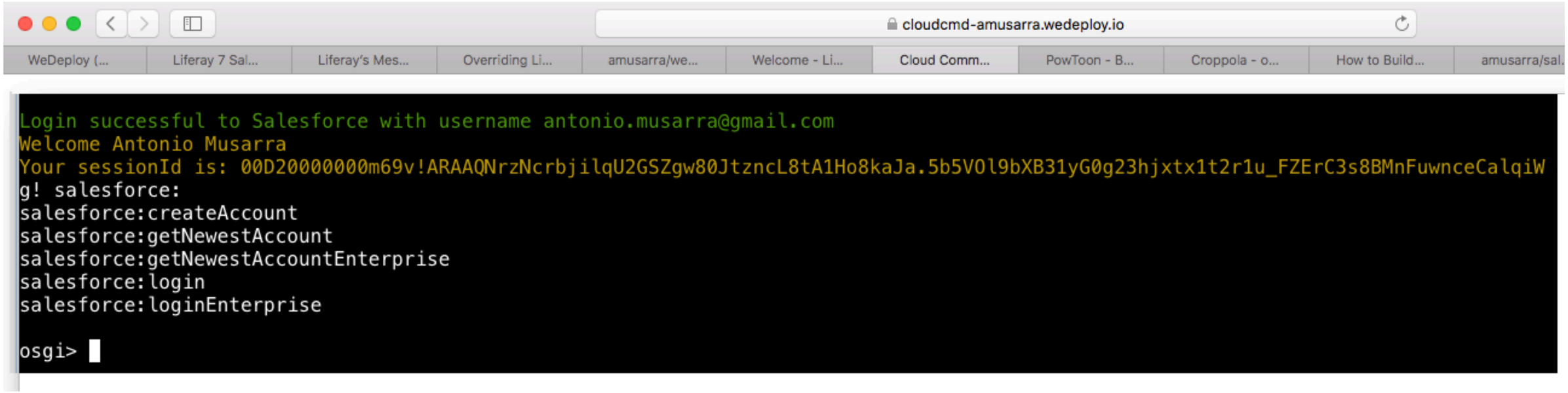
6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)

Su *wedeploy* ho creato due servizi:

- **liferay**: istanza di *Liferay 7 Community Edition GA4* con già installato il bundle OSGi del client di Salesforce.com e l'applicazione di esempio che sfrutta la Gogo Shell
- **cloudcmd**: un'applicazione NodeJS che si chiama *Cloud Commander*, utile per accedere al volume condiviso (tra i due servizi) ma in particolare modo per avere accesso alla Gogo Shell di Liferay via *telnet*.

I progetti dei due servizi sono pubblicati sul repository GitHub wedeploy-project che potete liberamente visionare e testare sul vostro account WeDeploy.

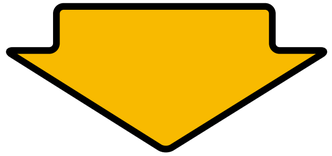
6. Realizzare un'applicazione d'esempio (Comandi Gogo Shell)



Accesso alla Gogo Shell di Liferay tramite *Cloud Commander* e esecuzione della *login* su Salesforce.com. Sono inoltre mostrati i comandi OSGi disponibili (quelli descritti nelle precedenti slide).

7. Un caso d'integrazione

- Nell'ambito dei sistemi CRM, esiste il processo ***Web to Lead Forms***
- L'obiettivo di questo processo è la generazione di *leads*, che attraverso un processo di ***MFA (Marketing Force Automation)***, potrebbero diventare *prospect* e forse *customer.*



Vogliamo far si che nel momento in cui un utente richieda la creazione del suo account per accedere al nostro portale, siano catturate le informazioni inserite nella form di creazione account e inviate a Salesforce.com per creare il rispettivo *lead*.

7. Un caso d'integrazione

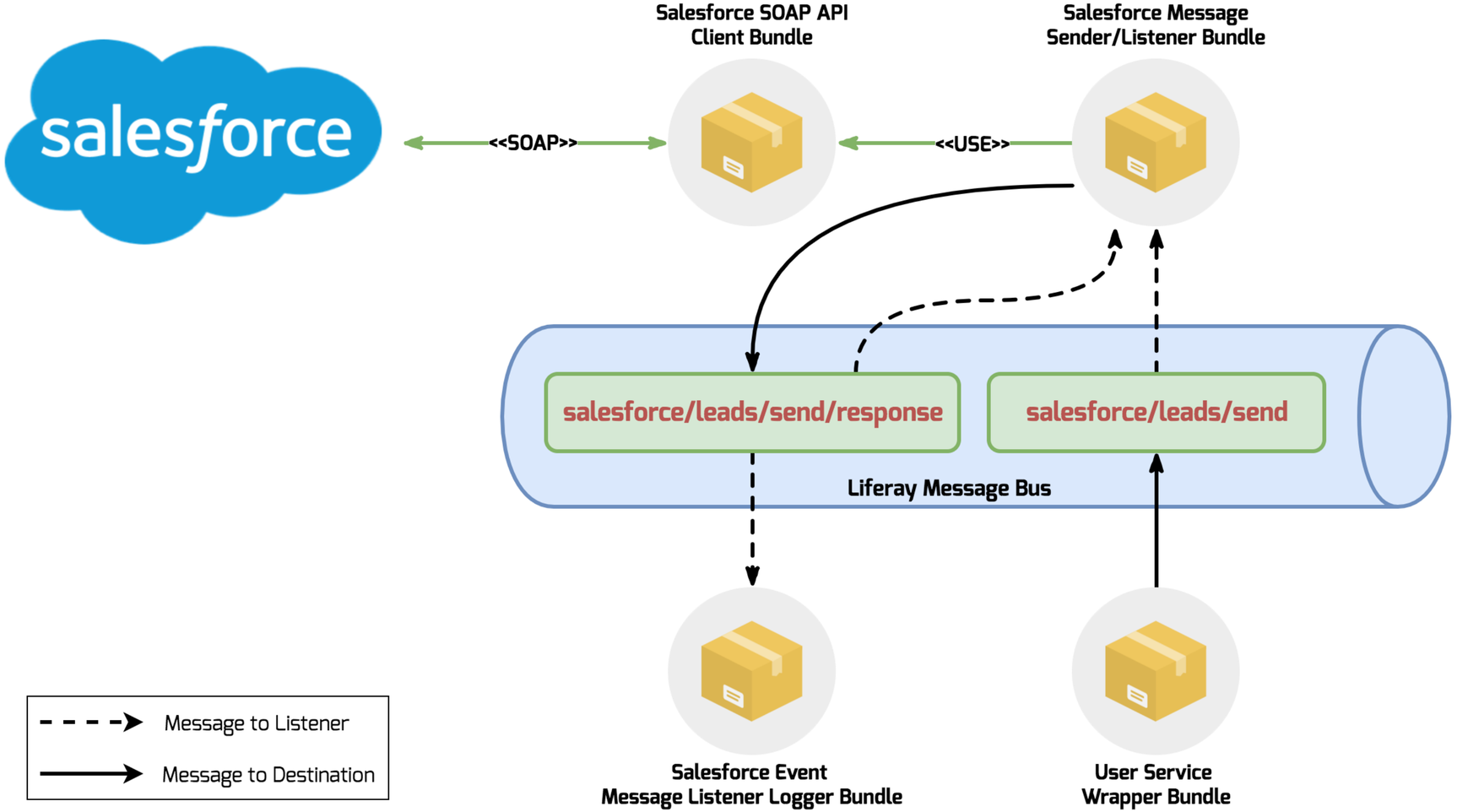
Adesso che abbiamo a disposizione il bundle OSGi che ci consente di colloquiare con Salesforce.com, dobbiamo farci una sola domanda.

Come possiamo implementare questo requisito d'integrazione su Liferay?

7. Un caso d'integrazione

- La funzionalità di *creazione account* di Liferay utilizza il servizio **UserService** per registrazione il nuovo utente
- Dobbiamo in qualche modo **"iniettare"** del nostro codice per catturare le informazioni del nuovo account da passare poi a Salesforce.com
- Possiamo sfruttare quindi la possibilità d'implementare il **Wrapper** del servizio standard di Liferay *UserService*. Troviamo un esempio su liferay-blade-samples
- L'operazione d'invio dei dati catturati è *asincrona*, inoltre, per favorire il disaccoppiamento tra i due sistemi, potremmo utilizzare il *Message Bus* di Liferay

7. Un caso d'integrazione



- La destinazione **salesforce/leads/send** riceve i dati catturati e li invia al message listener registrato
- Il message listener invia i dati a Salesforce.com tramite le API Java esportate dal bundle Salesforce SOAP API Client
- Alla ricezione della risposta questa è inviata alla destinazione **salesforce/leads/send/response**
- Il messaggio inviato alla response destination è elaborata da due message listener

Grazie per la vostra attenzione

Risorse utili

- Salesforce SOAP API Client OSGi Bundle
- Salesforce Gogo Shell Command Client
- Force.com Web Service Connector (WSC)
- Introducing SOAP API
- Force.com SOAP API Cheatsheet

Antonio Musarra
Twitter: @antonio_musarra
LinkedIn: https://www.linkedin.com/in/amusarra/
GitHub: https://github.com/amusarra
YouTube: Antonio Musarra's Blog Channel
Blog: https://www.dontesta.it