



REGIONE AUTONOMA  
DELLA SARDEGNA



## **Servizio di Transizione**

**Titolo: XMPI/JBF/JAUF**

**Codice: AREAS\_AP\_01\_JBF**

**Profilo: Analista Programmatore**

[www.eng.it](http://www.eng.it)



REGIONE AUTONOMA  
DELLA SARDEGNA



# ARGOMENTI DI QUESTE SLIDE

- Oggetti comuni e OSGI
- Integrazioni LDAP e SSO (oggetti comuni)
- Webservices ABF (eventi)
- Rilascio delle modifiche al DB

[www.eng.it](http://www.eng.it)

# Introduzione a JBF-XMPI

**Oggetti comuni e OSGi**

**LDAP e SSO**

**WebService ABF**

**Modifiche al DB**

# Plugin / Oggetti comuni

Alcune componenti software sono rilasciate in modalità **plugin** allo scopo di estendere le funzionalità base di AREAS. Il principale vantaggio del plugin è la facilità di installazione (non richiede l'aggiornamento dell'intero sistema).

I plugin di AREAS sono gestiti tramite il menu **Configurazione>>Avanzate>>Oggetti comuni**. Ciascun plugin è tipizzato in modo da istruire il sistema su come il plugin debba essere utilizzato

Oggetti Comuni	
Codice	WSAMCBD
Descrizione	Webservice areas AMC Budget
Tipo	C - Classe <span>Gestione</span>
Tipologia Classe	WebServicesDelegate - Delegate per Web Services
Path Classe	it.eng.iride.budget.ws.BudgetWS
Parametri Costruttore	
Stato	<input checked="" type="radio"/> Attivo <input type="radio"/> Disattivo

Conferma Elimina Annulla Nuovo Esci

# Plugin / Oggetti comuni

Tecnicamente un oggetto comune non è altro che un jar al cui interno è presente una classe “**entry-point**” che implementa una particolare interfaccia. Questa classe viene poi dichiarata nel campo **Path classe** in fase di definizione dell’oggetto comune.

The screenshot shows a software configuration window titled "Oggetti Comuni". It contains a form with the following fields and values:

Field	Value
Codice	WSAMCBD
Descrizione	Webservice areas AMC Budget
Tipo	C - Classe
Tipologia Classe	WebServicesDelegate - Delegate per Web Services
Path Classe	it.eng.iride.budget.ws.BudgetWS
Parametri Costruttore	
Stato	<input checked="" type="radio"/> Attivo <input type="radio"/> Disattivo

At the bottom of the window, there are five buttons: Conferma, Elimina, Annulla, Nuovo, and Esci. The "Path Classe" field is highlighted with a red rectangle.

# Plugin / Oggetti comuni

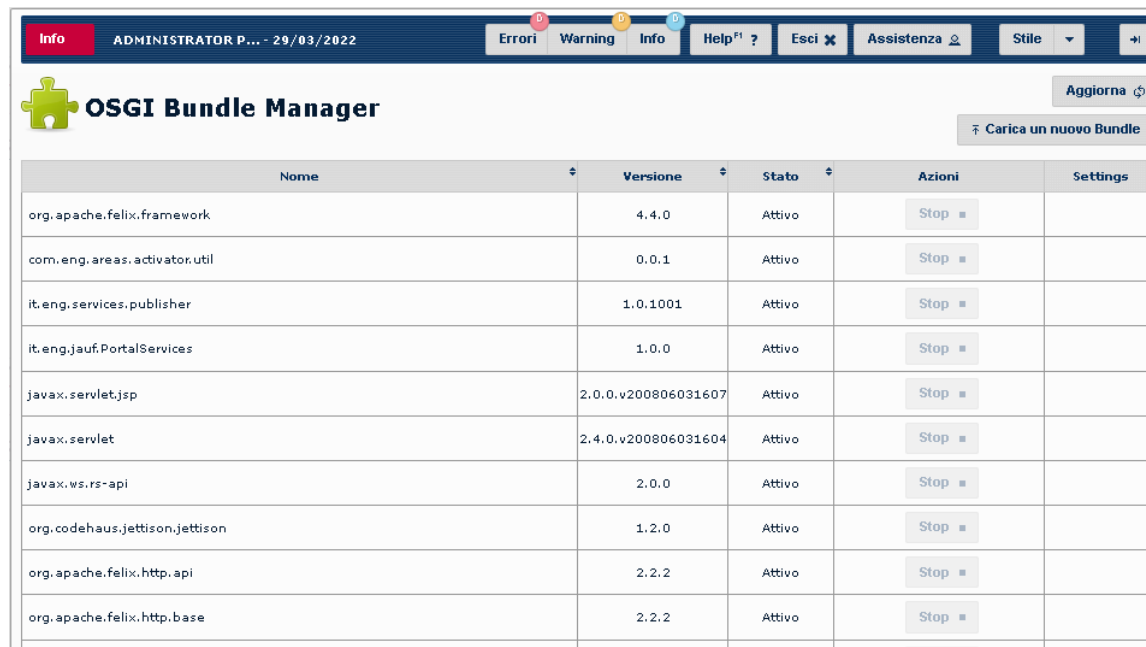
È possibile fare il deploy di diverse tipologie di plugin. Ciascuna tipologia richiede l'implementazione di una diversa interfaccia. Tra le tipologie a disposizione abbiamo:

- **OpenSSO**, per l'integrazione con sistemi SSO terzi
- **LDAP**, per l'integrazione con server LDAP
- **Ticket provider SSO**, per gestire il rilascio di token di autenticazione (chiamate di contesto)
- **Ricerca remota**, per integrare l'anagrafica pazienti con MPI terzi
- **Delegate WebService**, per l'implementazione di un servizio web service
- **Agent plugin**, per l'implementazione di un agent

# Plugin / OSGI Manager

**OSGi** è un framework definito dalla OSGi Alliance. I plugin OSGi sono anche denominati **bundle** ed è possibile installarli tramite il menu **Configurazione>>Avanzate>>OSGi Manager**

Rispetto agli oggetti comuni si tratta di uno standard estremamente diffuso e non richiede il riavvio dell'applicazione (salvo necessità specifiche dei singoli bundle).



ADMINISTRATOR P... - 29/03/2022				
Info Errori Warning Info Help? Esci Assistenza Stile				
OSGi Bundle Manager				
Aggiorna Carica un nuovo Bundle				
Nome	Versione	Stato	Azioni	Settings
org.apache.felix.framework	4.4.0	Attivo	Stop	
com.eng.areas.activator.util	0.0.1	Attivo	Stop	
it.eng.services.publisher	1.0.1001	Attivo	Stop	
it.eng.jauf.Portalservices	1.0.0	Attivo	Stop	
javax.servlet.jsp	2.0.0.v200806031607	Attivo	Stop	
javax.servlet	2.4.0.v200806031604	Attivo	Stop	
javax.ws.rs-api	2.0.0	Attivo	Stop	
org.codehaus.jettison.jettison	1.2.0	Attivo	Stop	
org.apache.felix.http.api	2.2.2	Attivo	Stop	
org.apache.felix.http.base	2.2.2	Attivo	Stop	

# Plugin / OSGI Manager

A differenza degli oggetti comuni, inoltre, non è necessario l'implementazione di particolari interfacce o la dichiarazione preventiva della tipologia di plugin.

Ogni bundle espone una classe **Activator** che si occupa di gestire il proprio processo di start/stop. Il plugin potrà implementare in questa classe quanto necessario per interagire con il resto del sistema tramite opportune chiamate alla classe **PSGExt.osgi.OSGIFramework**.

```
public class Activator implements BundleActivator {  
    public void start(BundleContext bundleContext) throws Exception {  
        OSGIFramework.registerDynamicMapping("abfSign.SignTest", SignTestLayout.class, null);  
    }  
    public void stop(BundleContext bundleContext) throws Exception {  
        OSGIFramework.unregisterDynamicMapping("abfSign.SignTest");  
    }  
}
```



# Plugin / OSGI Manager

La classe **PSGExt.osgi.OSGIFramework** espone vari metodi per la registrazione di:

- **Controller**, OSGIFramework.registerDynamicMapping
- **Bean**, OSGIFramework.registerBean
- **Query**, OSGIFramework.registerQuery
- **Agent**, OSGIFramework.registerAgent

Scopo di questa classe è anche la dichiarazione della **boot delegation**: una lista di package che risiedono nei progetti standard (quindi non in plugin) ma che sono comunque “visibili” dai bundle (es: log4j, axis, dom4j e package dei progetti)

# OSGi / Esercitazione

Proviamo a configurare insieme un nuovo plugin OSGi (solo struttura e scheletro)

# OSGI / Creazione di un plugin step by step

- 1 Create a new project and select the type "Plug-in Project"
- 2 Give a name and select "standard" in the "an OSGi Framework" field

**New Plug-in Project**

Plug-in Project

Create a new plug-in project

Project name: YourProjectName

☒ Use default location

Location: Drives:\\_bf\_add\YourProjectName

Choose file system: default

Project Settings:

☒ Create a Java project

Source folder: src

Output folder: bin

Target Platform

This plug-in is targeted to run with:

☐ Eclipse version: 3.5 or greater

☒ an OSGi framework: standard

- 3 In the **Activator** field, set the package **it.eng.name\_of\_your\_project\_in\_lowercase.Activator**

**New Plug-in Project**

Content

Enter the data required to generate the plug-in.

Properties

ID: YourProjectName

Version: 1.0.0

Name: YourProjectName

Vendor:

Execution Environment: JavaSE-1.8

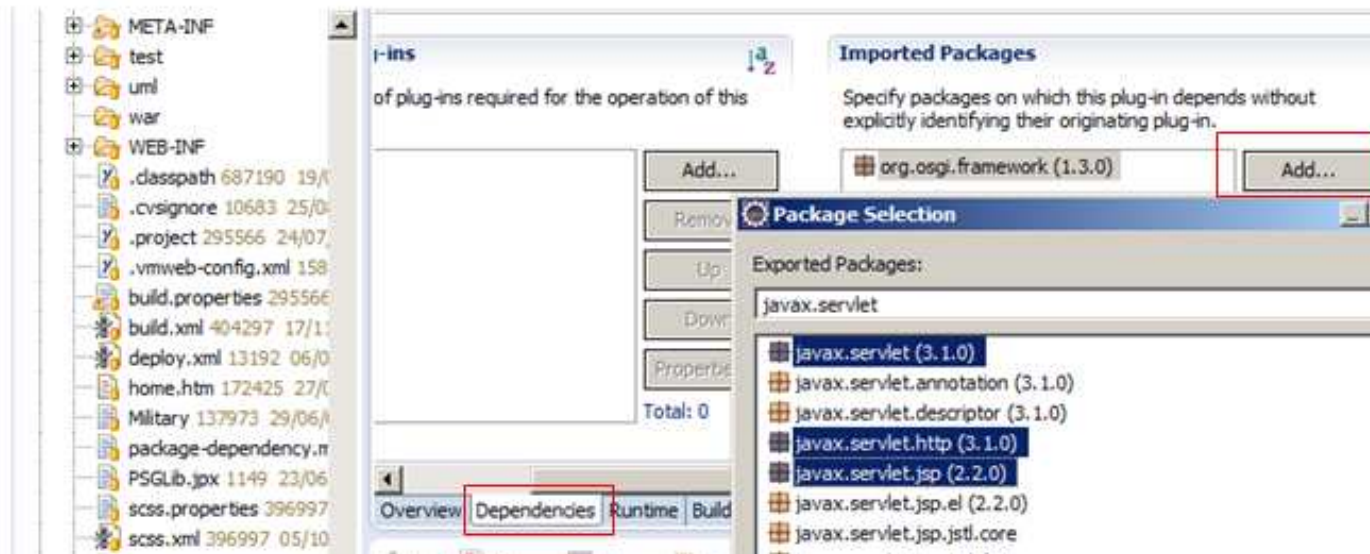
Options

☒ Generate an activator, a Java class that controls the plug-in's life cycle

Activator: it.eng.yourprojectname.Activator

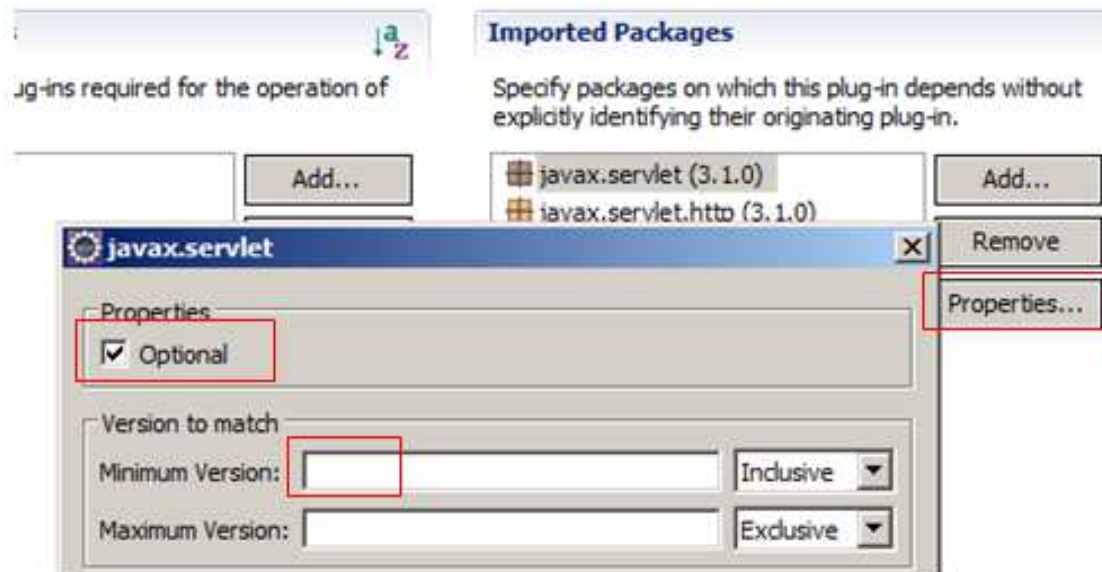
# OSGI / Creazione di un plugin step by step

- 4 Open the **MANIFEST.MF** file
  - 4.1 Select the “**Dependencies**” tab
  - 4.2 Click “**Add**” button
  - 4.3 Search the package **javax.servlet**
  - 4.4 Select the packages **javax.servlet**, **javax.servlet.http**, **javax.servlet.jsp**
  - 4.5 Click “**Ok**” button



# OSGI / Creazione di un plugin step by step

- 5 Select the packages you just added (one by one) and click the button **"Properties"**
  - 5.1 Select the **"Optional"** check
  - 5.2 Remove the value in the **Minimum Version** field



- 6 Click the **"export button"** in the right top corner and generate the jar file  
You can now upload your jar via OSGIManager panel

# Introduzione a JBF-XMPI

Oggetti comuni e OSGi

**LDAP e SSO**

WebService ABF

Modifiche al DB

# Plugin / LDAP e SSO

AREAS è in grado di delegare a terzi il processo di autenticazione utente integrandosi, ad esempio, con server **LDAP** o sistemi di **Single Sign On**. L'implementazione delle logiche necessarie all'integrazione avviene tramite il rilascio di un **oggetto comune**.

In questo caso il jar dovrà contenere una classe che implementa l'interfaccia **PSGExt.Idap.ILdapManager** per integrazioni via LDAP o **PSGExt.profile.w3.ExternalLogin** per SSO

Oggetti Comuni	
Codice	LDAP
Descrizione	Plugin per integrazione LDAP
Tipo	C - Classe
Tipologia Classe	LDAP - Integrazione LDAP
Path Classe	it.eng.Idap.MyLdapClass
Parametri Costruttore	PARAM1=VAL1,PARAM2=VAL2
Stato	<input checked="" type="radio"/> Attivo <input type="radio"/> Disattivo

Conferma Pulisci Esci

# Plugin / LDAP e SSO

Una volta installato l'oggetto comune è necessario “attivarlo” tramite un apposito parametro d'installazione sotto Sistema>>Login

The screenshot shows a configuration window titled 'Login' with various settings for user authentication. At the top, there's a 'Data Iniziale' field set to '27/01/2020'. Below this are tabs for 'DB', 'Monitor', 'Messaggi', 'Agent', 'Ricerche e Liste', 'Login', and 'Demoni'. The 'Login' tab is active. The settings include:

- 'Numero minimo di caratteri per la password di login': 8
- ☒ 'Modifica password al primo ingresso'
- 'Periodo di validità password (mesi)': 3
- ☒ 'Password diversa dallo username'
- ☒ 'Password diversa dalla precedente'
- 'Periodo di validità login in assenza connessioni (mesi)': 12
- ☐ 'Utilizza Crittografia non reversibile'
- ☐ 'Blocco login da due macchine differenti'
- 'Blocco login dopo n tentativi errati': 10
- 'Validatore Login': (empty field)
- 'Tipo di integrazione utilizzata': (empty field)

The bottom section, containing the 'Validatore Login' and 'Tipo di integrazione utilizzata' fields, is highlighted with a red rectangle.



# Introduzione a JBF-XMPI

Oggetti comuni e OSGi

LDAP e SSO

**Webservice ABF**

Modifiche al DB

# WebService ABF / Introduzione

AREAS è un sistema aperto e permette l'accesso a tutti i suoi bean (che incapsulano le logiche di business) tramite opportuni **Web Service**.

Il servizio è di tipo **SOAP-RPC** (implementato tramite AXIS) in cui si chiama l'unico metodo call() che prende come parametro una stringa XML e restituisce una stringa XML. Il servizio effettua il parsing della stringa in input ed esegue il comando specificato.

Tutti i servizi, quindi, sono esposti attraverso un unico endpoint centrale:

<http://miosito/contesto/jbfServices/JBFService>

# WebService ABF / Introduzione

Di seguito un esempio di chiamata al Web Service per il servizio standard di Open Session (login):

```
<?xml version="1.0" ?>
<!--
  Test con richiesta completa
-->
<Input>
  <OpenSession>
    <user>UTENTE</user>
    <password>PASSWORD</password>
    <azienda>AZ</azienda>
    <ufficio>UFFICIO</ufficio>
    <dataLavoro>20040101</dataLavoro>
  </OpenSession>
</Input>
```

# WebService ABF / Introduzione

La chiamata precedente, inclusa all'interno di un envelope SOAP, diventa quindi così:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <call soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <xml xsi:type="xsd:string">&lt;?xml version="1.0" ?>&#xd;
&lt;!--&#xd;
  Test con richiesta completa&#xd;
--&gt;&#xd;
&lt;Input&gt;&#xd;
  &lt;OpenSession&gt;&#xd;
    &lt;user&gt;JUNIT&lt;/user&gt;&#xd;
    &lt;password&gt;JUNIT&lt;/password&gt;&#xd;
    &lt;azienda&gt;JU&lt;/azienda&gt;&#xd;
    &lt;ufficio&gt;JUNITUFF&lt;/ufficio&gt;&#xd;
    &lt;dataLavoro&gt;20040101&lt;/dataLavoro&gt;&#xd;
  &lt;/OpenSession&gt;&#xd;
&lt;/Input&gt;&#xd;
&lt;/xml>
    </call>
  </soapenv:Body>
</soapenv:Envelope>
```

# WebService ABF / Flusso e servizi standard

All'interno di AREAS sono previsti una serie di servizi standard per l'accesso in modalità CRUD di tutti i suoi Bean e Query. In particolare:

- servizio di **OpenSession**, si occupa di aprire una sessione applicativa. In input riceve i dati di autenticazione e in output restituisce un SID (token) da utilizzare nelle successive chiamate
- servizio di **Find**, si occupa di eseguire una classe Query. In input è possibile indicare dei filtri e l'output è il risultato di ricerca della query
- servizio di **Get**, si occupa di caricare un singolo Bean. In input è possibile indicare l'identificativo del record e l'output è la lista completa delle proprietà del bean
- servizio di **Set**, si occupa di inserire/aggiornare un singolo Bean. In input è possibile indicare la lista delle proprietà da valorizzare sul bean e l'output è la lista completa delle proprietà e l'identificativo assegnato al bean
- servizio di **Delete**, si occupa di eliminare un singolo Bean. In input è possibile indicare l'identificativo del record e l'output è un messaggio di conferma o di errore
- servizio di **CloseSession**, si occupa di chiudere una sessione applicativa

**NB: È possibile ottenere uno stub di tutti i servizi tramite il menu Configurazione>>Avanzate>>Help Webservice**

# WebService ABF / Configurazione

Per poter utilizzare un servizio, l'utente che apre la sessione (OpenSession) dovrà essere opportunamente abilitato. Le abilitazioni si basano sul concetto di **entità applicativa**, che rappresenta il sistema che vuole integrarsi con AREAS. Gli step completi da seguire sono:

1. Creazione di un'entità applicativa (Configurazione>>Sistema>>Demoni>>Entità applicativa)
2. Definizione del messaggio sul demone di tipo SI\_WEBSERVICES (Configurazione>>Sistema>>Demoni>>Demone, tab "Messaggi")
3. Associazione dell'entità e abilitazione del messaggio appena creato (tab "Entità")

# WebService ABF / Esercitazione

Proviamo ad utilizzare i servizi ABF tramite **SoapUI**. Vedremo sia la parte di configurazione e sia la parte di esecuzione

# WebService ABF / Generazione di un evento

È possibile configurare AREAS in modo da generare dei messaggi in uscita al verificarsi di determinati **eventi**. Con eventi intendiamo l'inserimento, la modifica o la cancellazione di un oggetto su AREAS (Bean).

I vari eventi verranno accodati all'interno della tabella SI\_DEMONEMSGQUEUE e conterranno il riferimento ad un XML con i dati completi del bean oggetto dell'evento.

La configurazione degli eventi che deve generare l'applicativo avviene nel punto di menu **Configurazione>>Sistema>>Demoni>>Demone**, tab "Messaggi". Gli eventi vanno poi abilitati alle entità applicative (tab Entità, Msg di output)



# WebService ABF / **Esercitazione**

Vediamo come configurare un evento e il relativo messaggio di uscita generato sulla  
SI\_DEMONMSGQUEUE

# Introduzione a JBF-XMPI

Oggetti comuni e OSGi

LDAP e SSO

WebService ABF

**Modifiche al DB**

# Modifiche al DB / Introduzione

Tutto ciò che sviluppiamo deve essere rilasciato anche in termini di struttura e di oggetti sulla base dati. Esistono in AREAS due modalità per esportare le modifiche al DB:

- Tramite XML di versione
- Tramite la libreria opensource **Liquibase** (<https://liquibase.org/>)

Il primo metodo è utile **solo** in caso si preveda il rilascio di nuove versioni Major o minor. L'XML può essere creato tramite procedure guidate presenti all'url

<http://localhost:8080/areas/PSGExt.MonitorSviluppatori.do>

Per tutte le altre casistiche è possibile creare una semplice modifica liquibase (**changelog**) seguendo i formalismi previsti dalla libreria.

# Modifiche al DB / Liquibase in 4 passi

1. Individuare il proprio changelog master (HotDBModify.registerChangeLogMaster)
2. Aggiungere il proprio changeset (occhio al context)
3. Scrivere la propria modifica al DB secondo i formalismi disponibili nella documentazione ufficiale di liquibase (<https://docs.liquibase.com/concepts/changelogs/changelog-formats.html>)
4. Testare la propria modifica tramite il monitor liquibase (<http://localhost:8080/areas/hdm.ControlPanel.do>)

# Modifiche al DB / Esercitazione

Proviamo a creare una nostra modifica liquibase per la creazione di una tabella e di un lookup.