

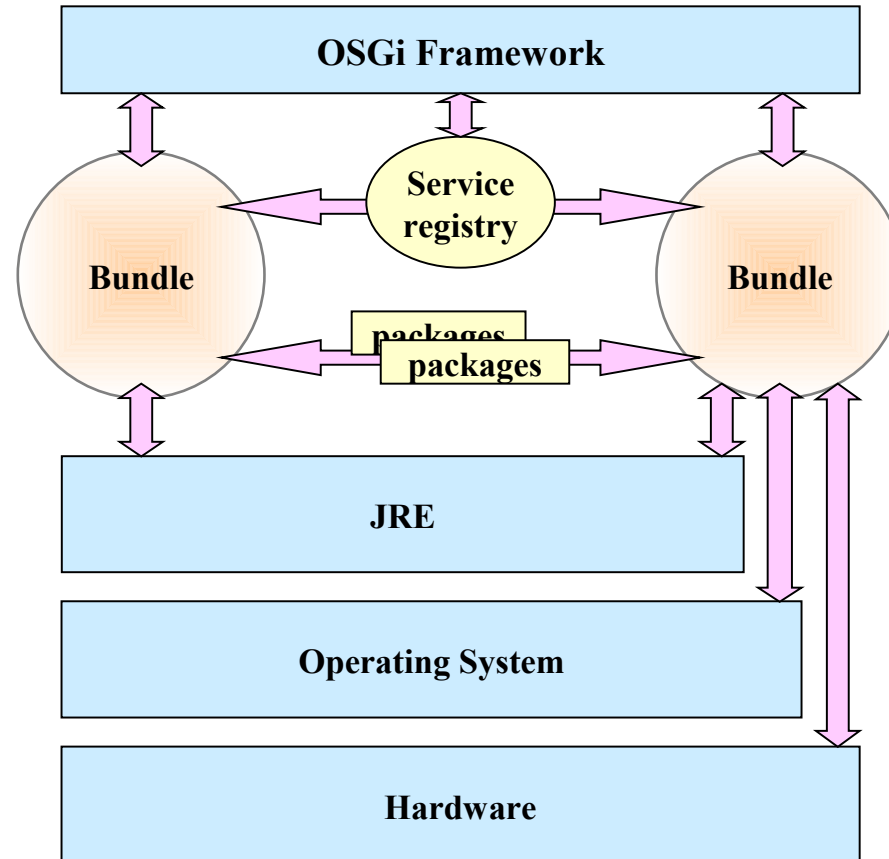
Il framework Open Services Gateway initiative (OSGi)

Domotics Lab – ISTI (CNR)
cesare.concordia@isti.cnr.it

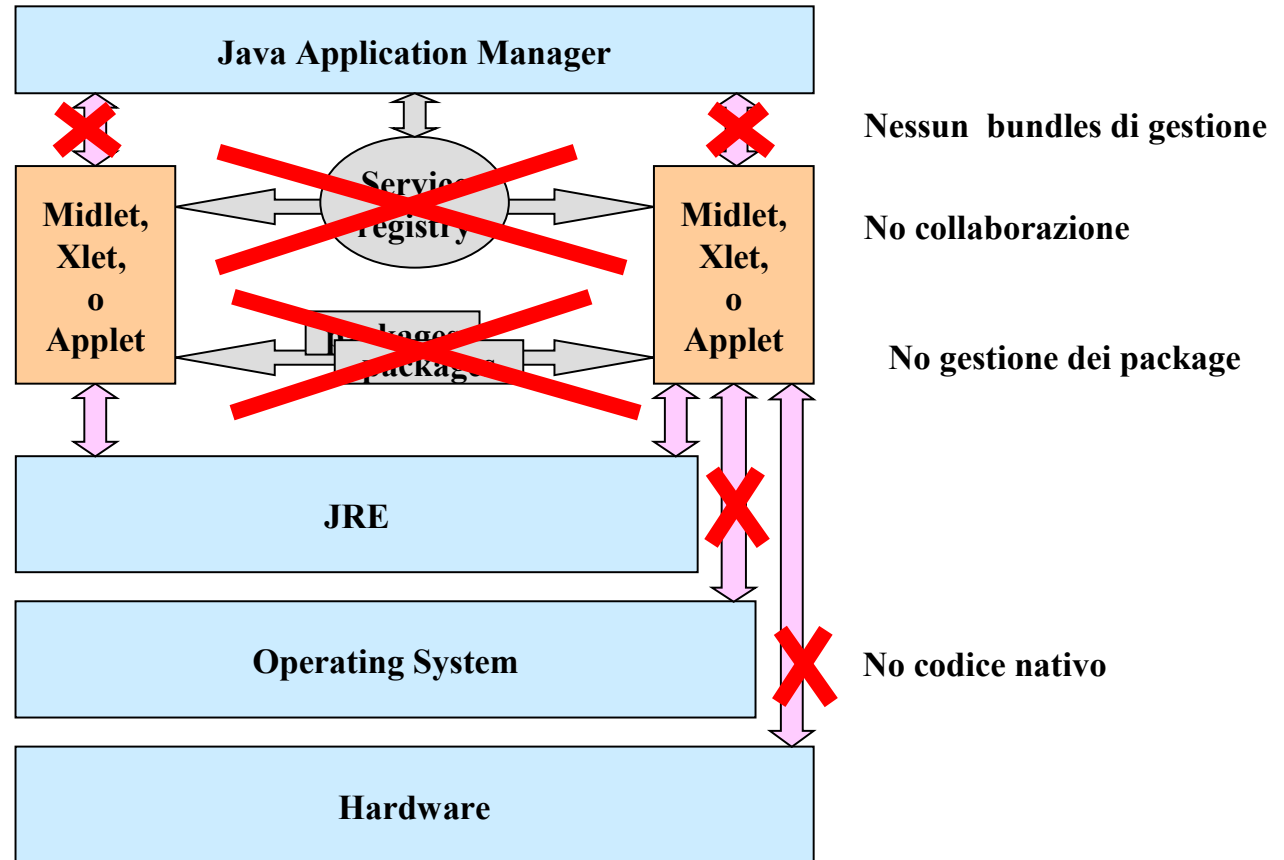
OSGi: modello collaborativo

- I bundle possono interagire e collaborare attraverso:
 - I servizi
 - La condivisione di package
- Un registro permette ai bundle di trovare e tracciare i servizi
- La collaborazione è gestita dal framework

OSGi: modello collaborativo



OSGi: modello collaborativo



OSGi: gestione del classpath

- Le applicazioni java sono composte da classi organizzate in packages
- Un programma java cerca le classi di cui ha bisogno in un insieme di directory o file jar, questo insieme è definito dal *classpath*
- Nel framework OSGi ogni bundle viene gestito da un diverso class loader e può definire il suo classpath nel proprio file manifest

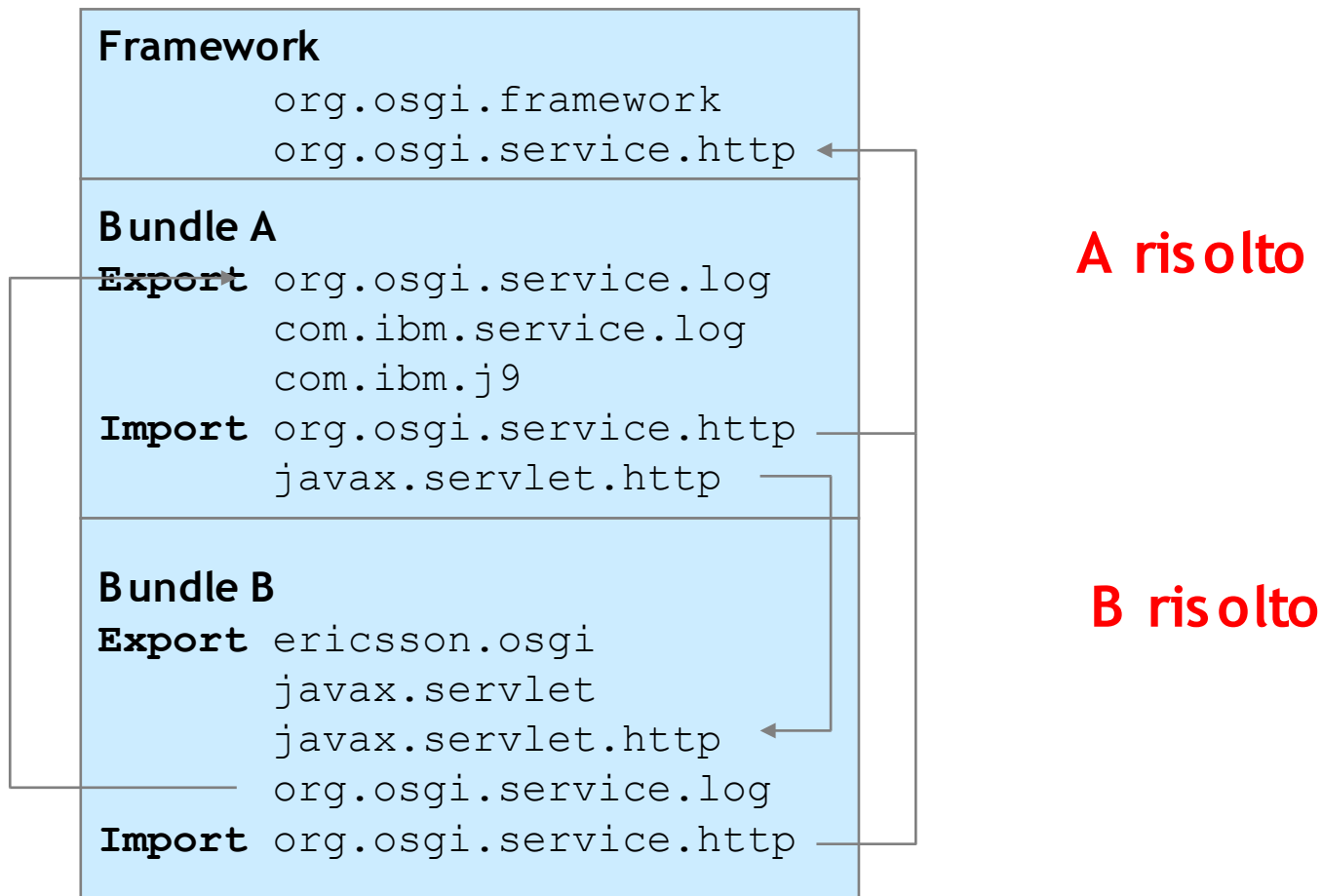
I bundle OSGi: le dipendenze

- La gestione delle dipendenze avviene a due livelli:
 - Dipendenze dei bundle/packages (Deployment dependency)
 - Require-Bundle
 - Import-Package
 - Dipendenze dei servizi (Service dependency)

I bundle OSGi: deployment dependency

- *Bundle-to-package*: un bundle può richiedere codice non contenuto nell'archivio jar, in questo caso importa il codice sotto forma di package
 - il codice richiesto deve essere dichiarato esplicitamente nel file *manifest* del richiedente
 - il codice richiesto deve essere esplicitamente esportato da un altro bundle
 - i bundle richiesti devono essere presenti nel framework per permettere al richiedente di essere attivato (active)
 - Questo tipo di dipendenza è gestito dal framework

I bundle OSGi: dipendenze dai packages



I bundle OSGi: dipendenze dai packages o dai bundles?

- Require-Bundle crea dipendenza con un intero bundle,
 - è semplice da usare ma puo' importare packages che non vengono usati
- Import-Package crea dipendenze con i soli packages specificati
 - preferibile perchè rende piu' semplice il versioning dei bundle

I bundle OSGi: service dependency

- Gestione delle attività di pubblicazione, scoperta, binding dei servizi in un ambiente dinamico (servizi che vengono attivati o fermati dinamicamente)
- *Bundle-to-service*: un oggetto contenuto in un bundle può usare servizi esterni
 - Dichiarate nel file manifest per informazione, **non** gestite dal framework
- *Service-to-service*: l'implementazione di un servizio può richiedere l'uso di un servizio registrato
 - Non definite esplicitamente

I bundle OSGi: le applicazioni

- Le applicazioni sono il risultato della *composizione* di più servizi
- Ciascun servizio deve scoprire se i servizi da lui richiesti sono disponibili
- I bundle devono pubblicare i servizi che espongono
- Ciascun servizio deve gestire il binding ed il rilascio automatico dei servizi richiesti
 - Dynamic assembly
 - Dynamic adaption

I bundle OSGi: le applicazioni

- Dynamic assembly
 - Una applicazione è assemblata dinamicamente man mano che i servizi vengono pubblicati
 - Se i servizi non sono immediatamente disponibili il bundle va messo in attesa (idle)
- Dynamic adaption
 - Capacità di adattarsi alle modifiche del service registry
 - *Monitoring*: ricezione delle notifiche del service registry
 - *Reconfiguration*: gestione delle modifiche

I bundle OSGi: le dipendenze

- Le dipendenze sono caratterizzate da due fattori:
 - Cardinalità: 1, 1..n, etc.
 - Binding policy
 - Statica: la dipendenza non può cambiare a runtime
 - Dinamica
- Le dipendenze *service-to-service* e *bundle-to-service* si stabiliscono tra *istanze* e servizi.
- Una *istanza* è un oggetto creato da una classe contenuta in un bundle

I servizi OSGi

- Il framework OSGi offre un servizio *registry* che permette la collaborazione tra i bundle
- La semantica di un servizio è definita dalla sua *interfaccia* (service interface),
 - Bundle forniti da provider differenti possono implementare la stessa interfaccia
- Un service object può implementare più interfacce
- Una implementazione di un servizio è riferita come un *oggetto servizio* (service object).

I servizi OSGi

- Un servizio ha un identificatore unico
- Se la security è abilitata può essere necessario assegnare ai servizi i permission opportuni
- Un service object è legato ad un bundle e deve essere registrato da questi nel registry tramite il *BundleContext*
- A ciascun servizio possono essere associate delle proprietà
 - Le proprietà possono essere modificate a runtime
- Per la ricerca dei servizi viene usata la sintassi *LDAP*

OSGi: registrazione/ricerca servizi

- I servizi sono registrati passando al framework attraverso il BundleContext i seguenti dati:
 - il nome dell'interfaccia
 - l'implementazione del servizio
 - le proprietà
- La ricerca dei servizi viene fatta fornendo al framework
 - il nome dell'interfaccia
 - un filtro LDAP con delle condizioni di ricerca

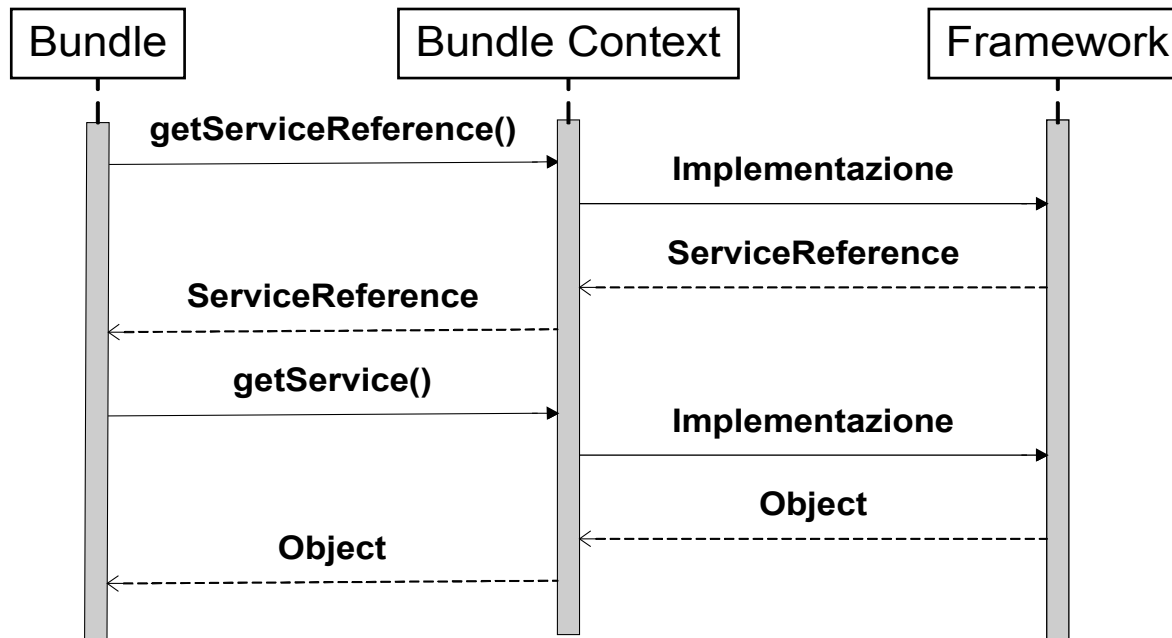
OSGi : il filtro LDAP

```
<filter> ::= '(' <filtercomp> ')'  
<filtercomp> ::= <and> | <or> | <not> | <item>  
<and> ::= '&' <filterlist>  
<or> ::= '|' <filterlist>  
<not> ::= '!' <filter>  
<filterlist> ::= <filter> | <filter> <filterlist>  
<item> ::= <simple> | <present> | <substring>  
<simple> ::= <attr> <filtertype> <value>  
<filtertype> ::= <equal> | <approx> | <greater> | <less>  
<equal> ::= '='  
<approx> ::= '~='  
<greater> ::= '>='  
<less> ::= '<='  
<present> ::= <attr> '=*'  
<substring> ::= <attr> '=' <initial> <any> <final>  
<initial> ::= NULL | <value>  
<any> ::= '*' <starval>  
<starval> ::= NULL | <value> '*' <starval>  
<final> ::= NULL | <value>
```

OSGi: servizi

- Il framework consente di riferire servizi attraverso la classe `ServiceReference`
- Per ogni servizio registrato nel framework viene creato un oggetto `ServiceRegistration`
 - unico viene utilizzato per de registrare il servizio
- È possibile registrare un service factory invece che un oggetto
 - Permette di creare un servizio unico per ogni cliente

OSGi: servizi



OSGi: servizi

```
ServiceRegistration registerService(  
    String cls,  
    Object srvc,  
    Dictionary prprts)
```

```
ServiceReference[]  
    getServiceReferences(  
        String cls,  
        String fltr)
```

```
Object getService(  
    ServiceReference reference)
```

```
boolean ungetService(  
    ServiceReference rfrnc);
```

OSGi : registrazione di un servizio

```
public class NotifyActivator implements
BundleActivator {
    private BundleContext context = null;

    private ServiceRegistration reg = null;
    public void start(BundleContext context) {
        this.context = context;

        //Notifica del nuovo servizio

        Properties props=new Properties();
        props.put("version", "1.0");
        reg = context.registerService(
            "org.ungoverned.MyService",
            new MyServiceImpl(), props);
        ....
    }
    public void stop(BundleContext context) {
        //gestione della disattivazione del bundle
    }
}
```

OSGi : ricerca di un servizio

```
public class MessengerActivator implements BundleActivator
{
    private BundleContext context = null;
    public void start(BundleContext context) {
        this.context = context;

        // Cerca il servizio

        ServiceReference refs[]=context.getServiceReferences (
            "org.ungoverned.MyService", "(version=1.0)");

        if(refs!=null)
        {
            NotificationService ns =
                (MyService) context.getService (refs[0]);
        }

        //Uso del servizio
    }
    public void stop(BundleContext context) {
        //gestione della disattivazione del bundle
    }
}
```

OSGI: tutorial

OSGi tutorial: set up

- Software necessario:
 - Eclipse SDK (3.2 or 3.3) (www.eclipse.org)
 - aQute.tutorial.runtime.zip (<http://www.aqute.biz/OSGi/Tutorial>)
 - il plugin “bnd” installato (nella jar directory del tutorial runtime)
 - Facilita la creazione dei bundle generando automaticamente il manifestdi file manifest

OSGi tutorial: installazione del plug in “bnd”

- Copiare il file bnd.jar dalla directory jar del aQute.tutorial.runtime alla directory dei plug in di eclipse
- Restart di eclipse
- Verificare: Help > About > Plugin Details
- Controllare che sia presente la descrizione “aQute Bundle Tool”

OSGi : Hello World bundle

```
package aQute.tutorial.world;

import org.osgi.framework.*;

public class Activator implements BundleActivator {

    public void start(BundleContext context) throws Exception {
        System.out.println("Hello World");
    }

    public void stop(BundleContext context) throws Exception {
        System.out.println("Goodbye World");
    }
}
```

OSGi : DictionaryClient bundle

```
ServiceReference[] refs = ctxt.getServiceReferences(  
    DictionaryService.class.getName(), "(Language=*)");
```

