
An Introduction to OSGi in WebSphere Application Server

Chris Wilkinson, IBM Hursley



Who Am I?

- Chris Wilkinson – Software Engineer
- Joined IBM Hursley development lab in 2001
 - Worked on test/development of WAS V6, V6.1, V7
 - SIBus (JMS) Developer
 - OSGi Feature Pack Developer



Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Conference Agenda

- An Introduction to OSGi
- Enterprise OSGi Best Practices
- WAS Support for Enterprise OSGi Applications
- Spring to Blueprint migration

Agenda

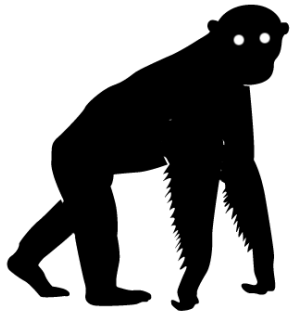
- OSGi – What is it?
- OSGi and the Enterprise
- Enterprise OSGi in WAS v7 and WAS v8 Beta
- Demo's (time permitting)

OSGi : What is it?

A (very) brief history of programming

- ◉ In the beginning, there were bits ...

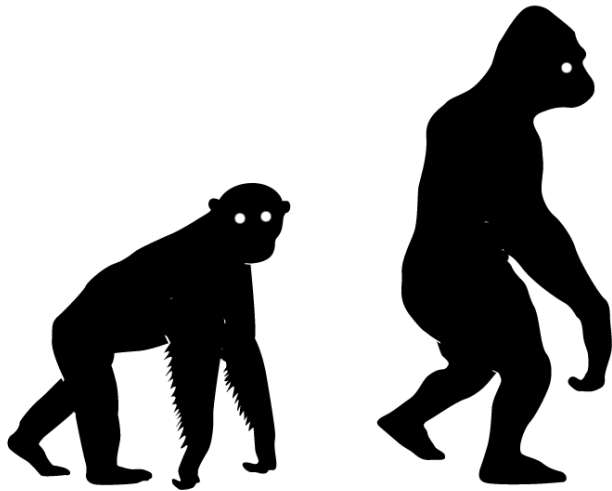
```
0101011101001001010010010011011101101101101
```



A (very) brief history of programming

- Then came words ...

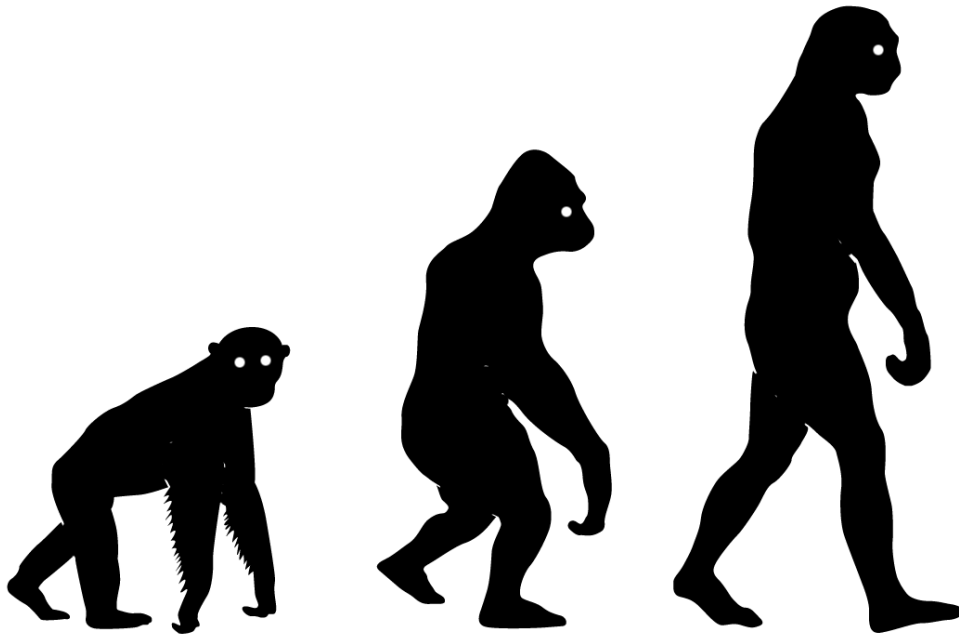
```
mov     ah, 9
mov     dx, offset hello_message
int     21h
```



A (very) brief history of programming

- ... functions and libraries ...

```
#include <stdio.h>
main() {
    printf ("Hello World!\n");
}
```



March 25, 2011

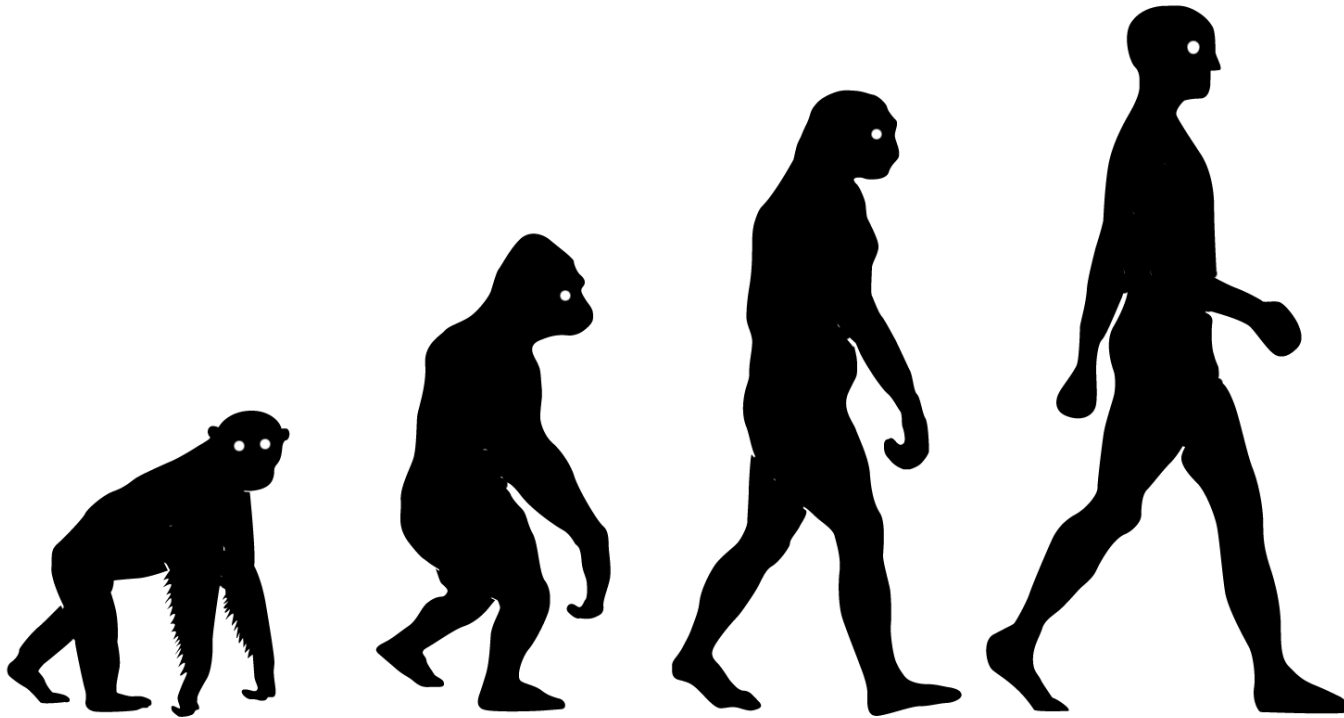
QCon London 2010

© 2011 IBM Corporation
9 Ian Robinson

A (very) brief history of programming

🌀 ... objects ...

```
public class HelloWorldSayer {  
    public void sayHello() {  
        System.out.println("Hello, world!\n");  
    }  
}
```



March 25, 2011

QCon London 2010

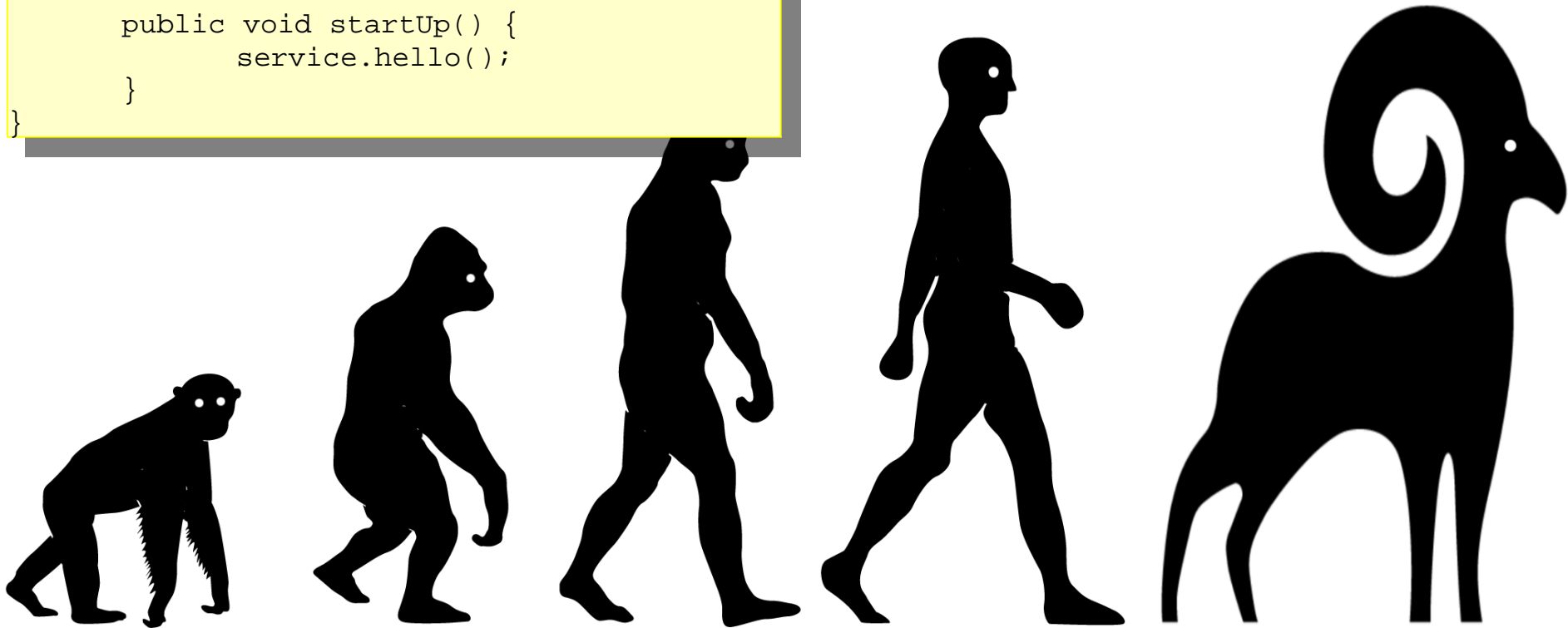
© 2011 IBM Corporation
Ian Robinson

A (very) brief history of programming

... and then – OSGi

```
Import-Package : com.ibm.services.HelloWorldService;version=8.0.0;  
Export-Package : com.ibm.client.HelloWorldClient
```

```
public class HelloWorldClient {  
    HelloWorldService service = null;  
  
    public void startUp() {  
        service.hello();  
    }  
}
```

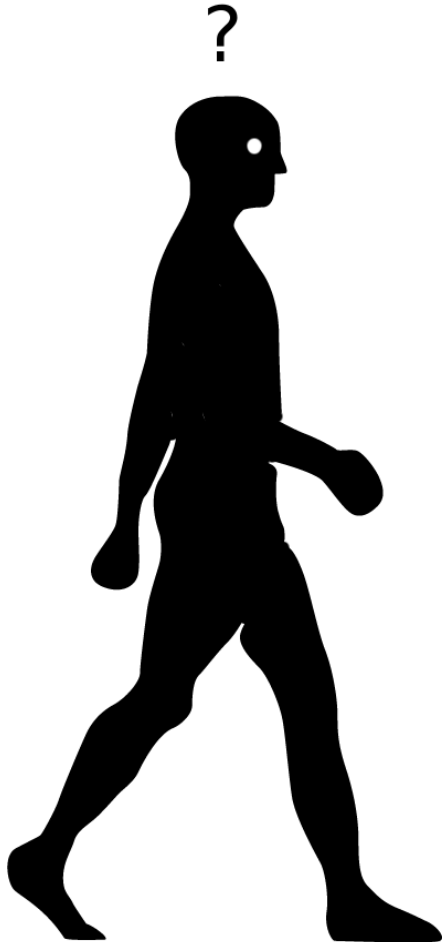


March 25, 2011

QCon London 2010

© 2011 IBM Corporation
11an Robinson

What was wrong with where we were?



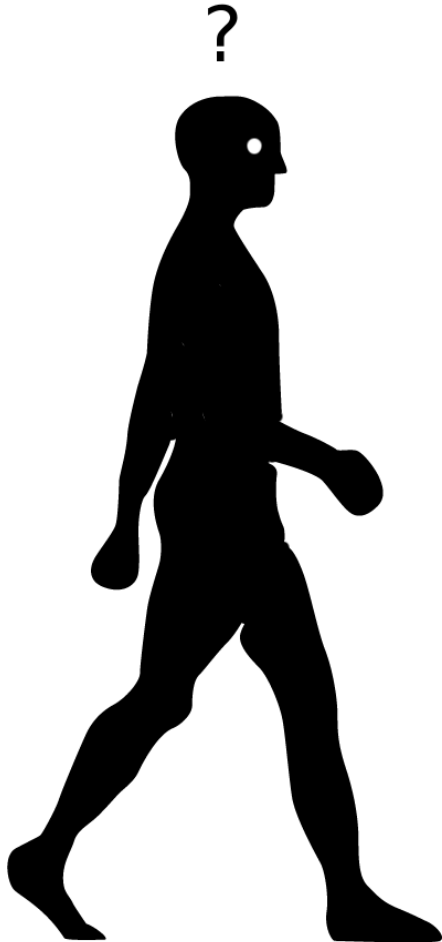
March 25, 2011

QCon London 2010

© 2011 IBM Corporation
Ian Robinson

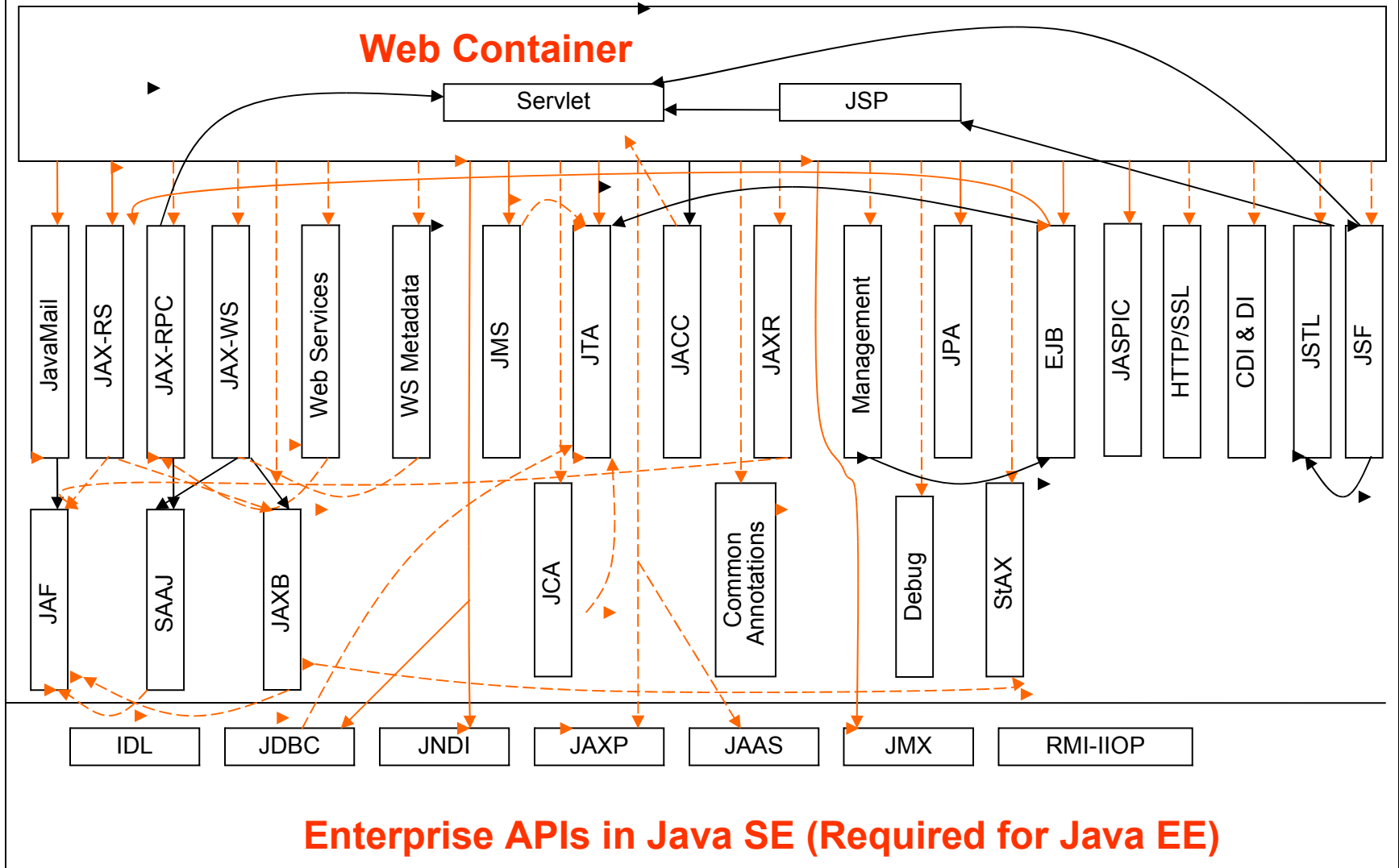
What was wrong with where we were?

- A question of scale



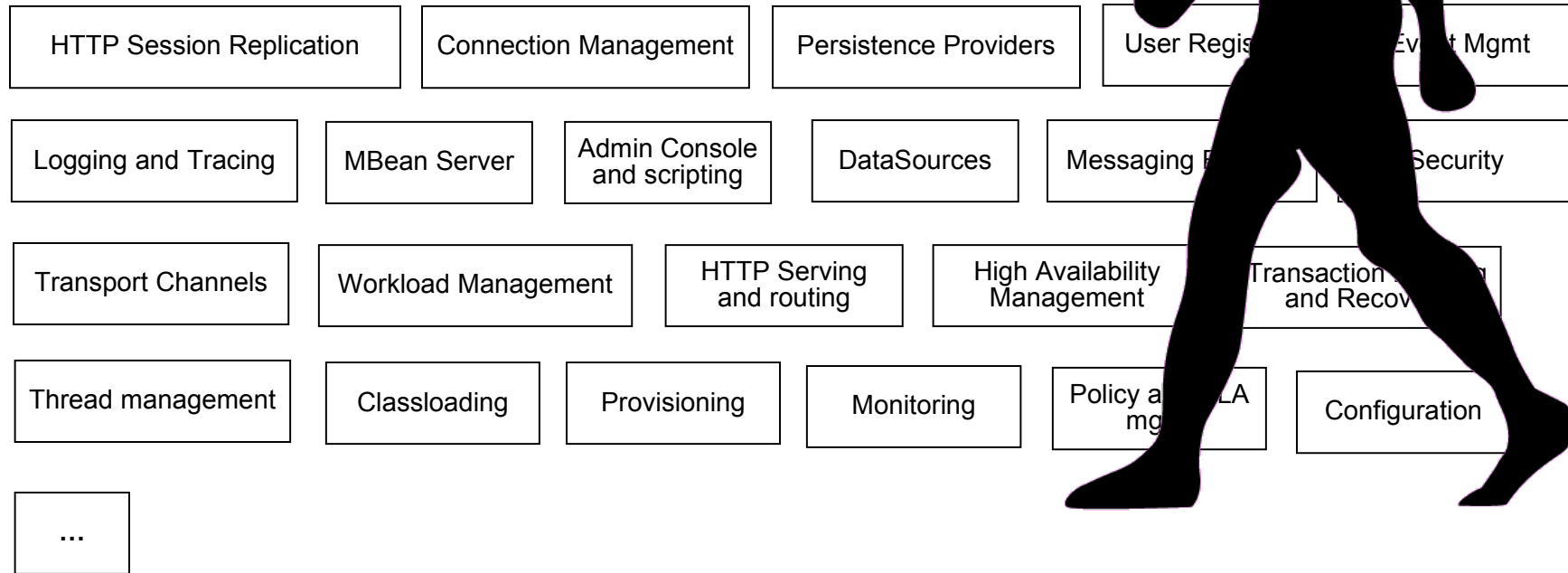
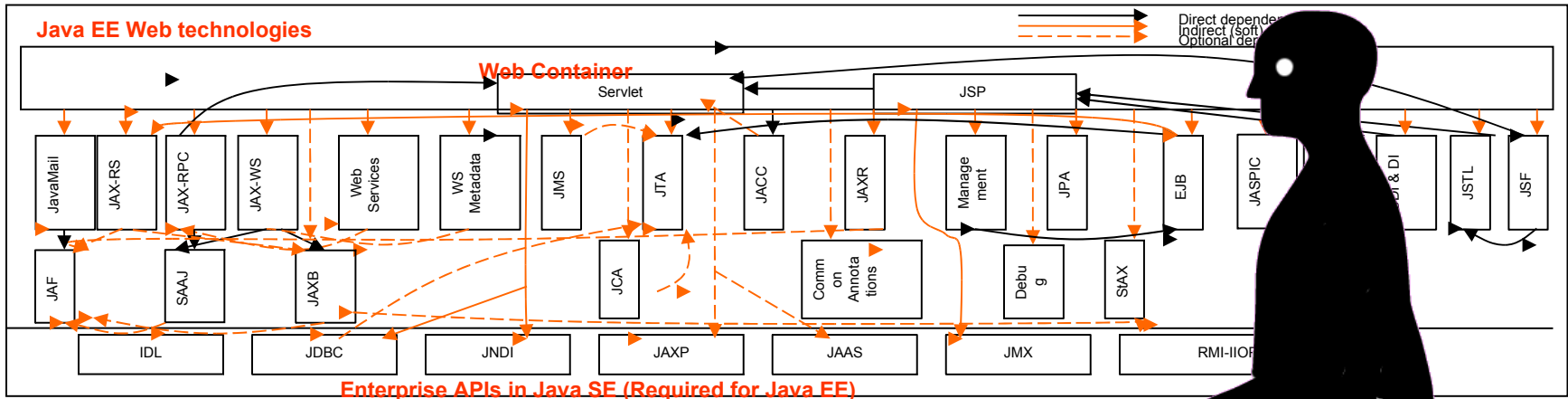
Java EE Web technologies

- Direct dependency
- Indirect (soft) dependency
- - - Optional dependencies



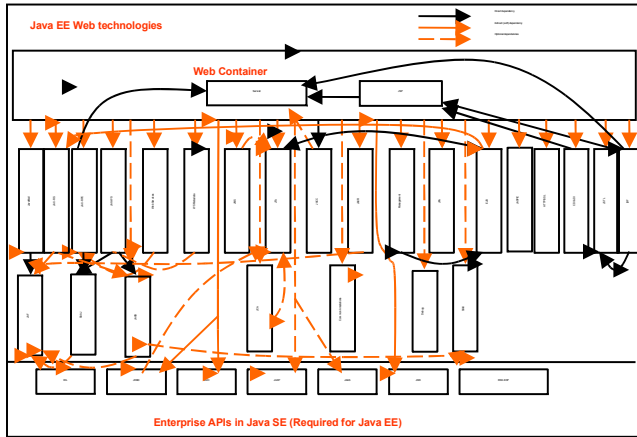
(capability == bloat)?

!



Beyond Java EE...

!@*!



Batch

Telephony

BPM

Dynamic Scripting

SCA

Business Rules

Complex Event Processing

...

HTTP Session Replication

Connection Management

Persistence Providers

User Directories and Authentication

Logging and Tracing

MBean Server

Admin Console and scripting

DataSources

Messaging Engine and Security

Transport Channels

Workload Management

HTTP Serving and routing

High Availability Management

Transaction Logging and Recovery

Thread management

Classloading

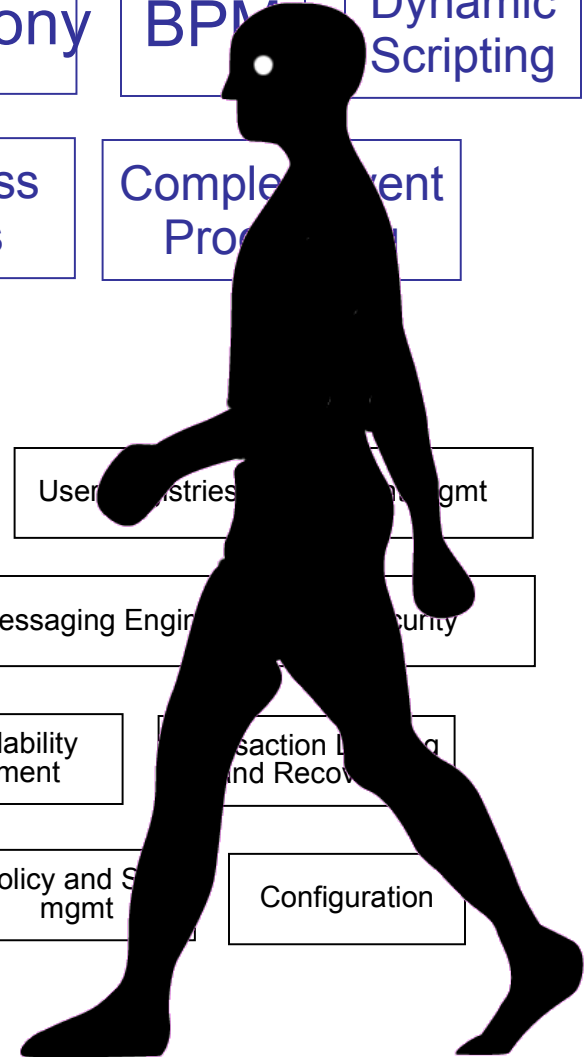
Provisioning

Monitoring

Policy and Security mgmt

Configuration

...



Modularity *inside* Enterprise platforms

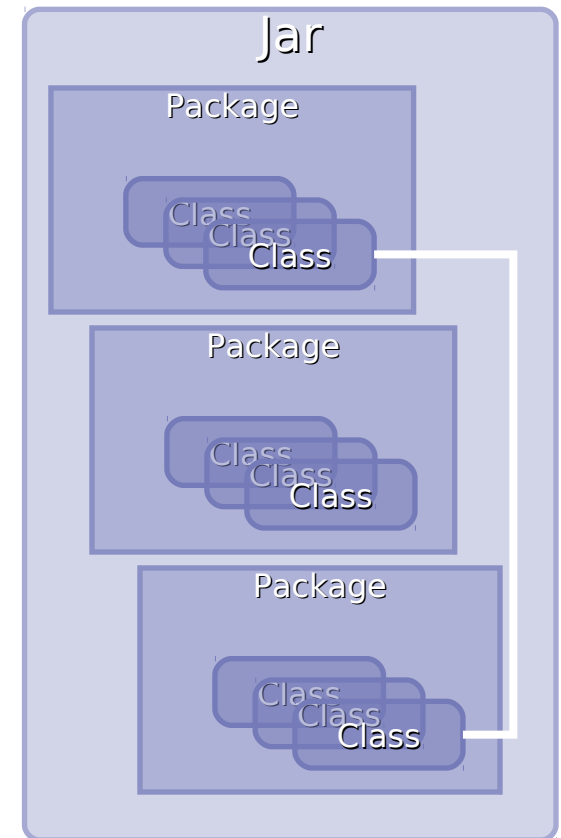
- The major AppServer and Integration vendors and Open Source communities have all embraced OSGi for their own internal architecture
 - IBM WebSphere, Oracle WebLogic, Red Hat JBoss, Sun GlassFish, SpringSource dm Server, Paremus Service Fabric, Eclipse Platform, Apache Geronimo, (non-exhaustive list)
 - http://www.osgi.org/wiki/uploads/News/2008_09_16_worldwide_market.pdf
- These complex engineering projects required a modular approach to development as they evolved and grew
 - Breaking the problem into well-defined coherent modules enables development teams to focus on their area of concern.
 - OSGi not only *enables* but *enforces* modularity

Axioms of Modular Programming

- Code Re-use
 - Reduce development time
- Code Sharing
 - Reduce Disk Footprint
 - Reduce Memory Footprint
- Code Isolation
 - Ease of Development
- Code Versioning
 - Ease of Maintenance

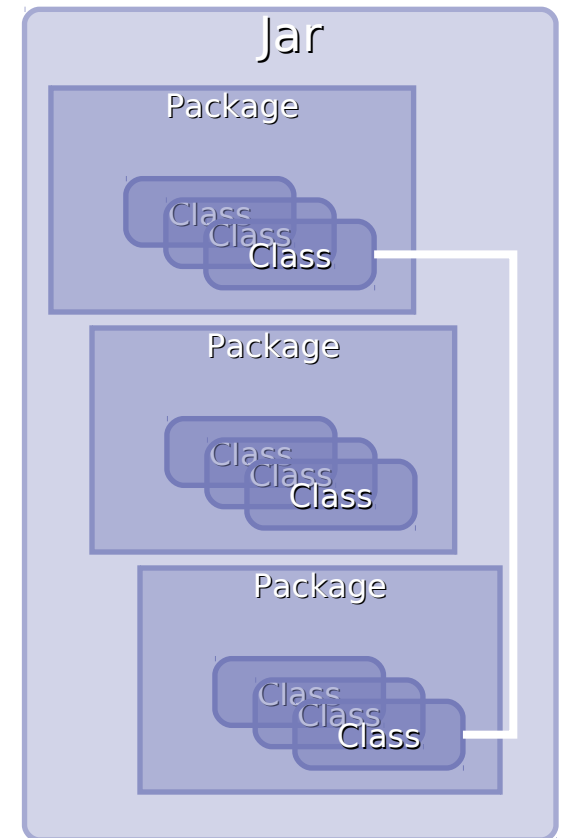
OSGi Origins : Modular Programming in Java?

- Java's hierarchy comprises classes, packages and jars
- A Java module is a **JAR**



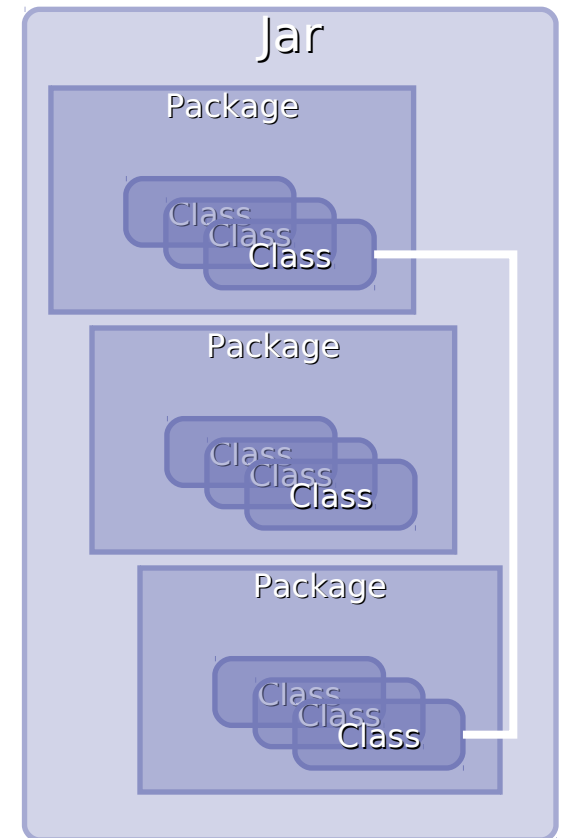
OSGi Origins : Modular Programming in JavaSE ?

- Java's hierarchy comprises classes, packages and jars
- A Java module is a **JAR**
- **Code Isolation?**
 - **NO**
 - **Isolation ONLY inside packages/classes**
 - **Other JARs can access contents of public classes of all other JARs**
 - **JARs depend on other JARs and dependencies are hidden!**



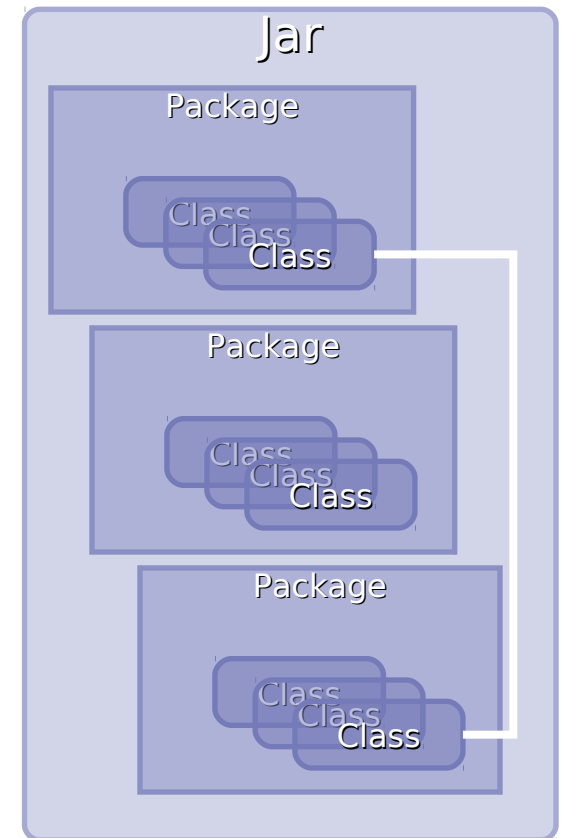
OSGi Origins : Modular Programming in Java?

- Java's hierarchy comprises classes, packages and jars
- A Java module is a **JAR**
- **Code Re-Use?**
 - Difficult
 - **JARs have hidden dependencies on other JARs. Hard to re-use a JAR.**



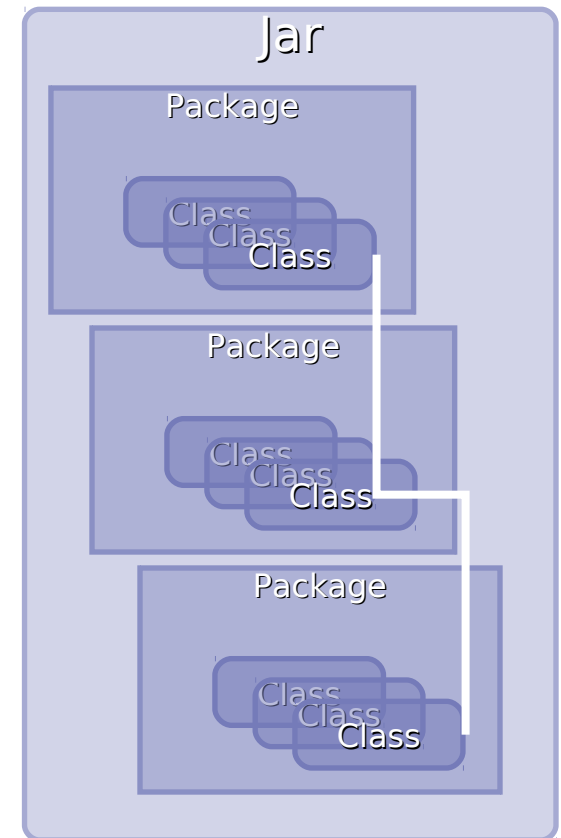
OSGi Origins : Modular Programming in Java?

- Java's hierarchy comprises classes, packages and jars
- A Java module is a **JAR**
- **Code Versioning?**
 - NO
 - **Only one version of a class is loaded**
 - **Whichever is first on the classpath!**



OSGi Origins : Modular Programming in Java?

- Java's hierarchy comprises classes, packages and jars
- A Java module is a **JAR**
- **Code Sharing?**
 - YES
 - Multiple JARs *can* access the same 3rd Party JAR.



OSGi Origins : Modular Programming

Java (SE) Applications

- Code Sharing



- Code Re-use



- Code Versioning



- Code Isolation



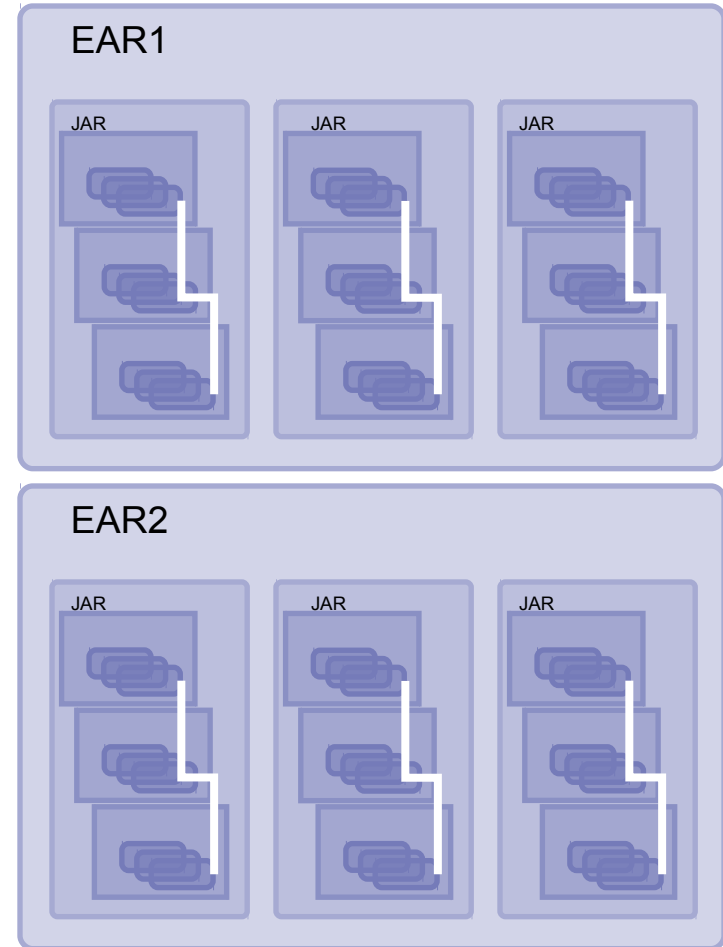
OSGi Origins : Modular Programming in JEE?

- JEE Helps...but...
- Same Modular issues exists *inside* each app.



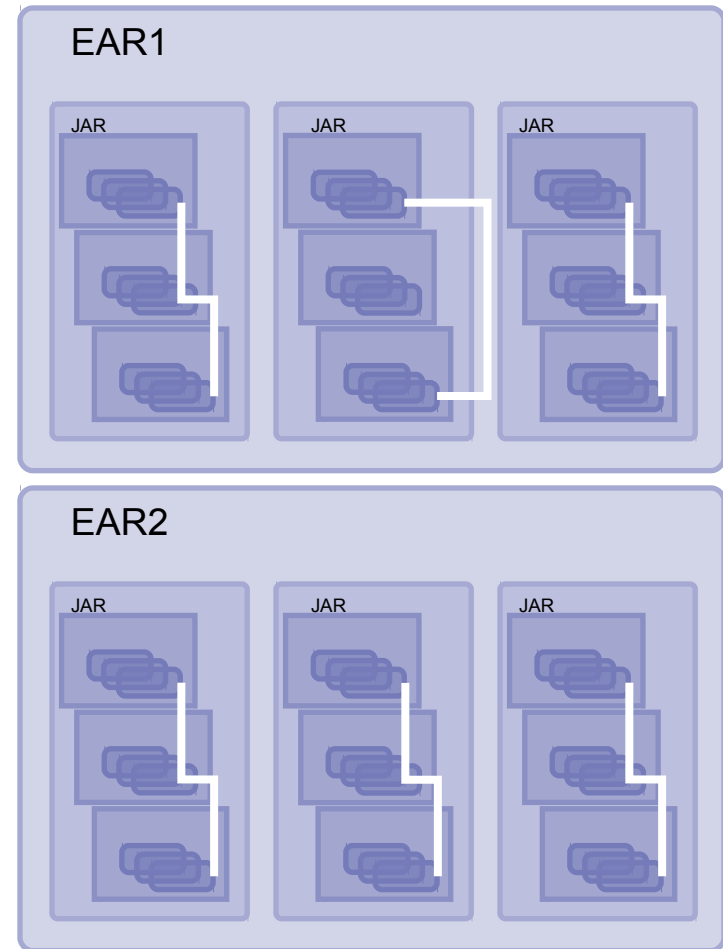
OSGi Origins : Modular Programming in JEE?

- JEE Helps...but...
- Same Modular issues exists *inside* each app.
- **Code Isolation?**
 - Only across applications
 - NOT inside application



OSGi Origins : Modular Programming in JEE?

- JEE Helps...but...
- Same Modular issues exists *inside* each app.
- **Code Re-use?**
 - Difficult
 - Can develop a new EBA using JARs from existing EBAs
 - Still difficult if JARs have numerous dependencies



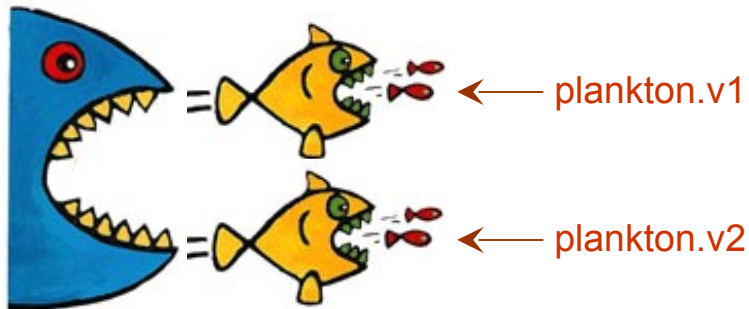
OSGi Origins : Modular Programming in JEE?

- JEE Helps...but...
- Same Modular issues exists *inside* each app.
- **Code Sharing?**
 - NO across application due to isolation
 - Multiple copies of common JARs in each EBA
 - Wasted Disk space!
 - Wasted Memory!







OSGi Origins : Modular Programming in JEE?

- JEE Helps...but...
- Same Modular issues exists *inside* each app..
- **Code Versioning?**
 - Across Applications only
 - 3rd party libraries may need different versions of a class to other 3rd party libraries



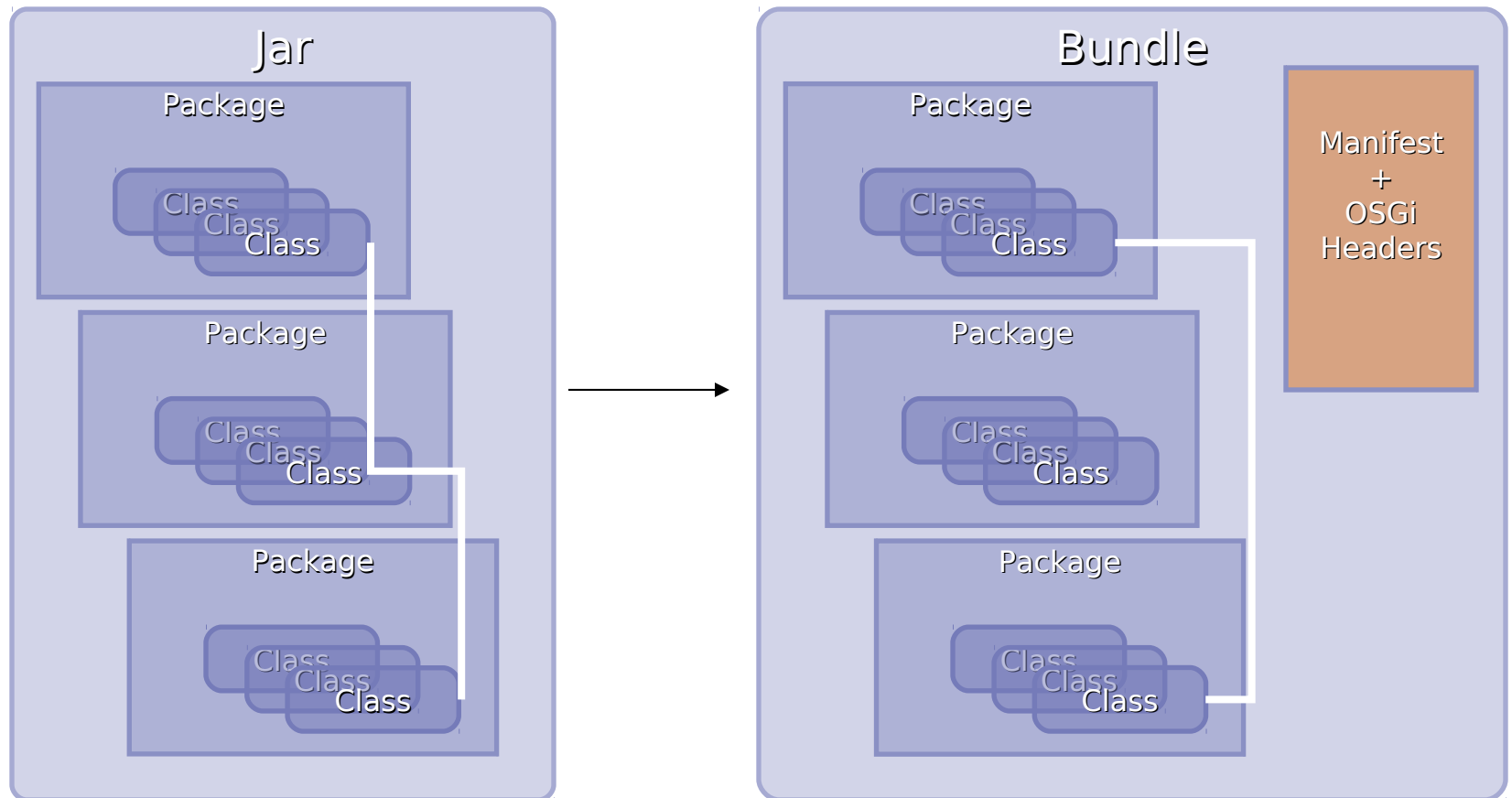
OSGi Origins : Modular Programming

Enterprise Java (EE) Applications

- Code Sharing 
- Code Re-use 
- Code Versioning 
- Code Isolation 

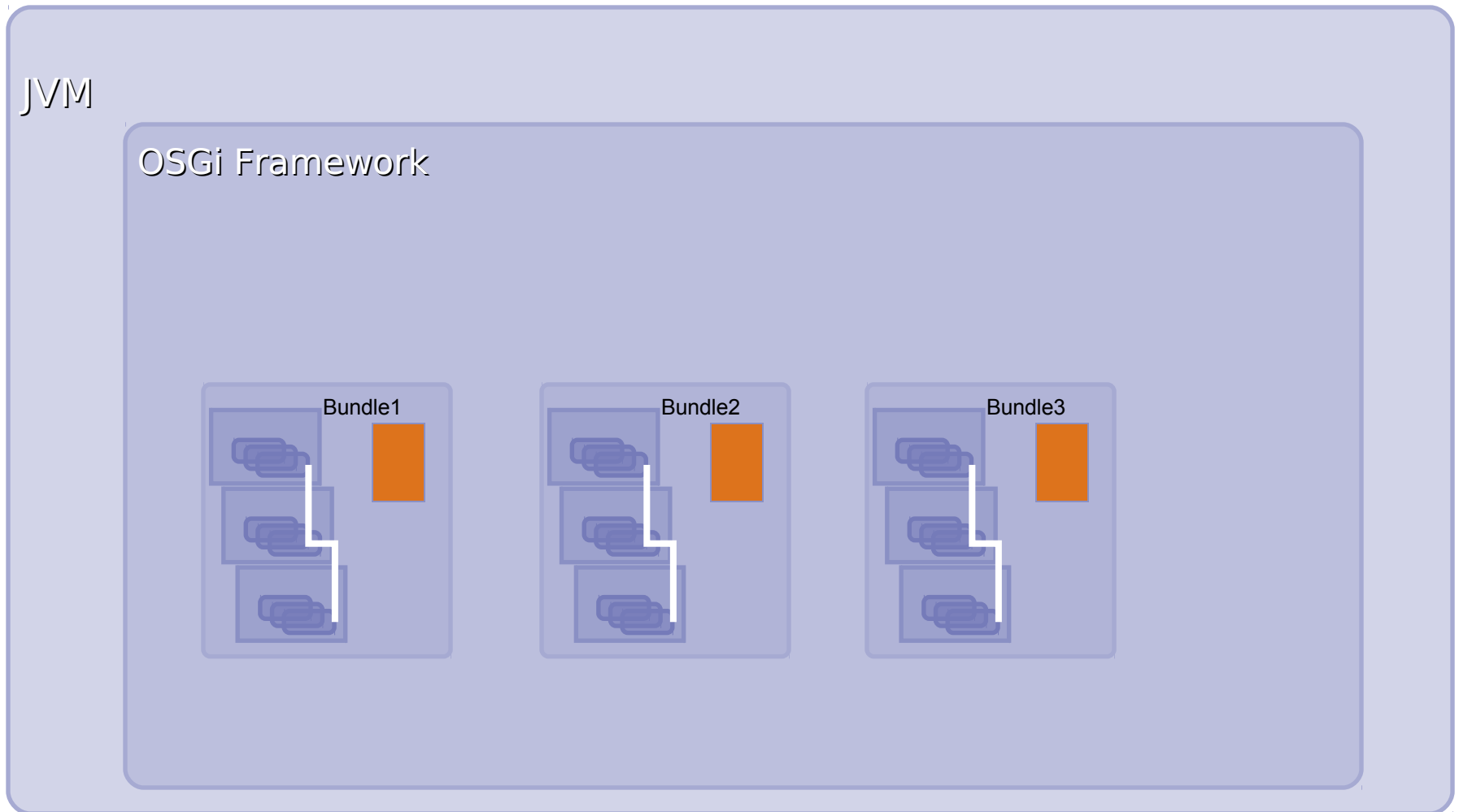
Modular Programming with OSGi

- Introducing the **OSGi Bundle**



Modular Programming with OSGi

- **Bundles** are installed to an embedded **OSGi Framework**



Modular Programming with OSGi

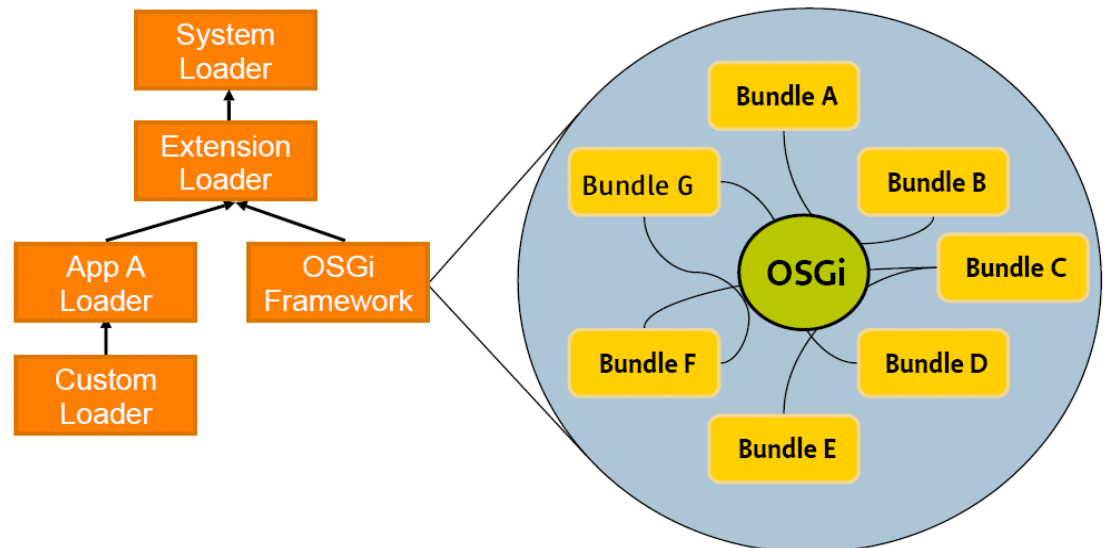
- **What's in the manifest?**
 - **Export-Package:** What packages from this bundle are visible and reusable outside of the bundle?
 - **Import-Package:** What packages from other bundles does this bundle depend upon?
 - **Bundle-Version:** Multiple versions of bundles can live concurrently

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: MyService bundle
Bundle-SymbolicName: com.sample.myervice
Bundle-Version: 1.0.0
Bundle-Activator: com.sample.myervice.Activator
Import-Package: com.something.i.need;version="1.1.2"
Export-Package: com.myervice.api;version="1.0.0"
```

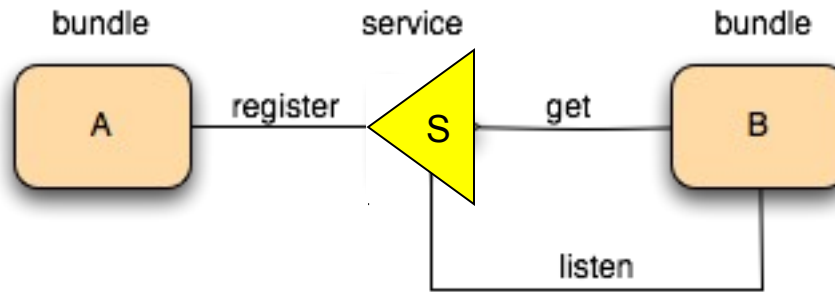
Modular Programming with OSGi

- **Class Loading**

- Each bundle has its own loader
- No flat or monolithic classpath
- Class sharing and visibility decided by declarative dependencies, not by class loader hierarchies
- OSGi framework works out the dependencies including versions



Further de-coupling with the OSGi Service Registry



- Bundles are “loosely coupled” by communicating via Services.
- The OSGi framework provides a non-durable “Service Registry”
- Services are published to and discovered from this Service Registry.
 - Services are the primary means of collaboration between bundles.
- Services are fully dynamic and typically have the same lifecycle as the bundle that provides them.

Modular Programming with OSGi

- **Code Isolation**
 - YES
 - Each bundle can be isolated by declaring exactly what is exported from within.
 - Total control over access to bundle contents



Modular Programming with OSGi

- **Code Re-Use**
 - YES
 - All dependencies are declared by the import header of the manifest
 - Services help to de-couple bundles
 - Easy to reuse bundles in other scenarios.



Modular Programming with OSGi

- **Code Sharing**
 - YES
 - A bundle can be accessed by several bundles simultaneously



Modular Programming with OSGi

- **Code Versioning**
 - YES
 - The bundle manifest specifies a version. Bundles with different versions can exist in the framework
 - I.e. We can have multiple versions of classes in use at same time in same JVM!



OSGi Origins : Modular Programming

OSGi Java (SE) Applications

- Code Sharing
- Code Re-use
- Code Versioning
- Code Isolation



OSGi : An Open Standard

- OSGi is an open standard governed by the **OSGi Alliance**
- <http://www.osgi.org>
- OSGi has been used internally in WAS since V6.1 and in Eclipse since R3.
- The next step?... OSGI for Enterprise Applications!

Enterprise OSGi : An Open Standard

- Enterprise OSGi an Open Standard
 - OSGi Enterprise Expert Group (EEG)
 - Apache Aries
- Using existing Java SE/EE specifications:
 - JTA, JPA, JNDI, JMX, WebApps...
- Includes SpringFramework-derived **Blueprint Component Model** for Dependency Injection
- Java EE is still the core enterprise application programming model
- All Modular Programming benefits in a JEE App!

OSGi in WebSphere Application Server

- **WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0**
 - Available since May 2010
 - More downloads in a shorter period of time than any previous WAS v7 feature pack
 - Two installable features:
 - **JPA 2.0** feature introduces Java EE 6 JPA 2.0 enhancements to object-relational persistence to simplify data access and optimize performance
 - **OSGi Application** feature provides the development, assembly, and deployment of OSGi enterprise applications

<http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/osgi/index.html/>

WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API 2.0

IBM United States [change] Search

Home Solutions Services Products Support & downloads My IBM Welcome [IBM Sign in] [Register]

WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API 2.0

Features and benefits

System requirements

How to buy

Related links

- Warranty info
- Product Accessibility Information

Software > WebSphere > WebSphere Application Server Feature Packs >

WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API 2.0

Efficient development and management and enhanced application performance



WebSphere Application Server Feature Packs overview
→ Launch demo



WAS V7.0: Planning and Design
→ Download Redbook

We're here to help



IBM Installation Manager

Install Packages

Select the packages to install.

Installation Packages

- IBM WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API 2.0
 - Version 1.0.0.0

Show all versions

Details

IBM WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API 2.0 1.0.
The IBM WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API 2.0 provides a lightweight...

- Repository: <https://ausgsa.ibm.com/gsa/ausgsa/projects/w/was.build/bo/tp/JPA10.JIWAS/test/b1018.02.linux/linux/zippe>

- <http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/osgi/index.html>

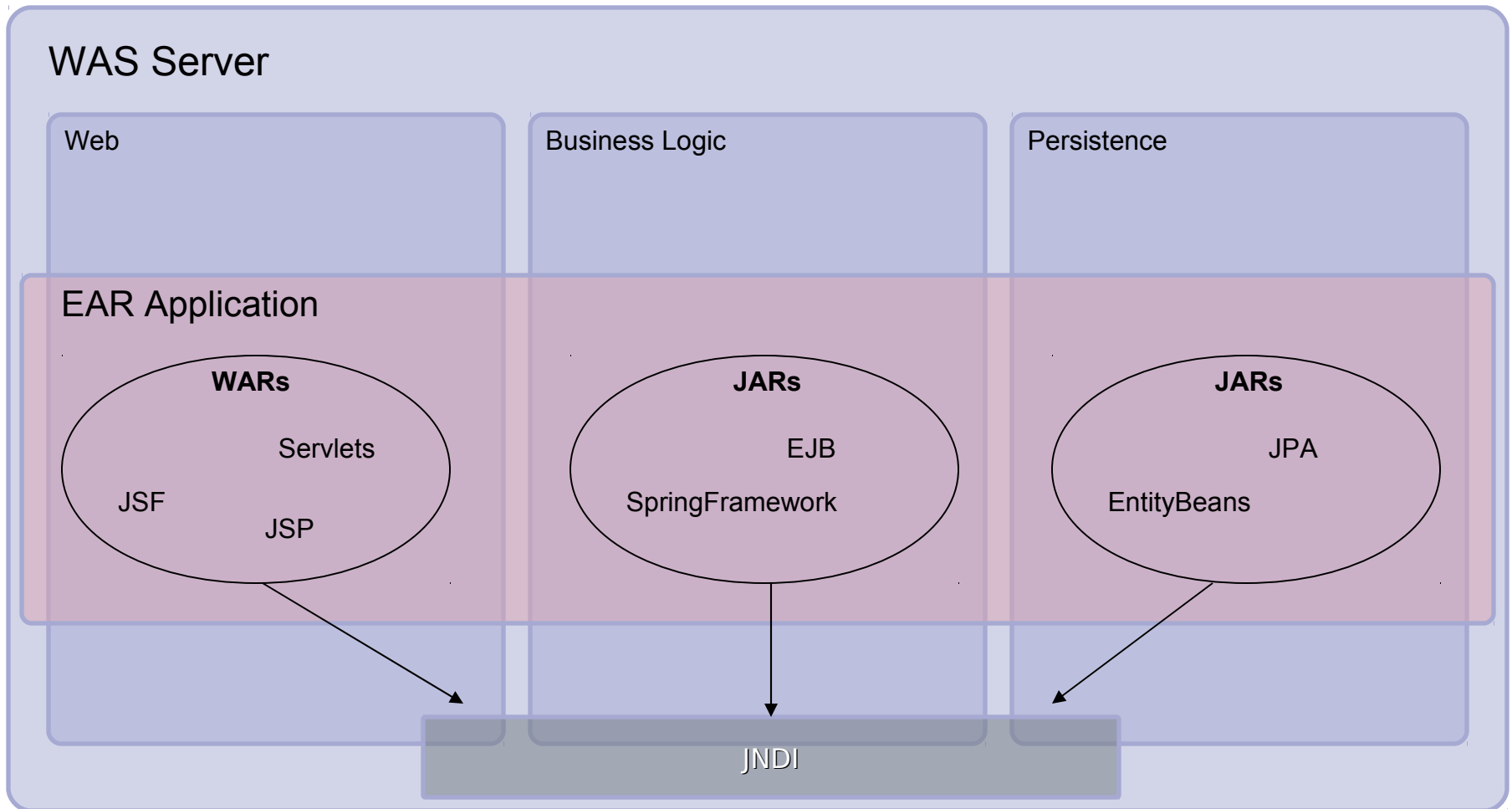
OSGi in WebSphere Application Server

- **WebSphere Application Server V8 Beta**
 - OSGi part of the core platform in v8 Beta
 - Host of new features
 - See Duane Applebys talk later today !

<http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/osgi/index.html/>

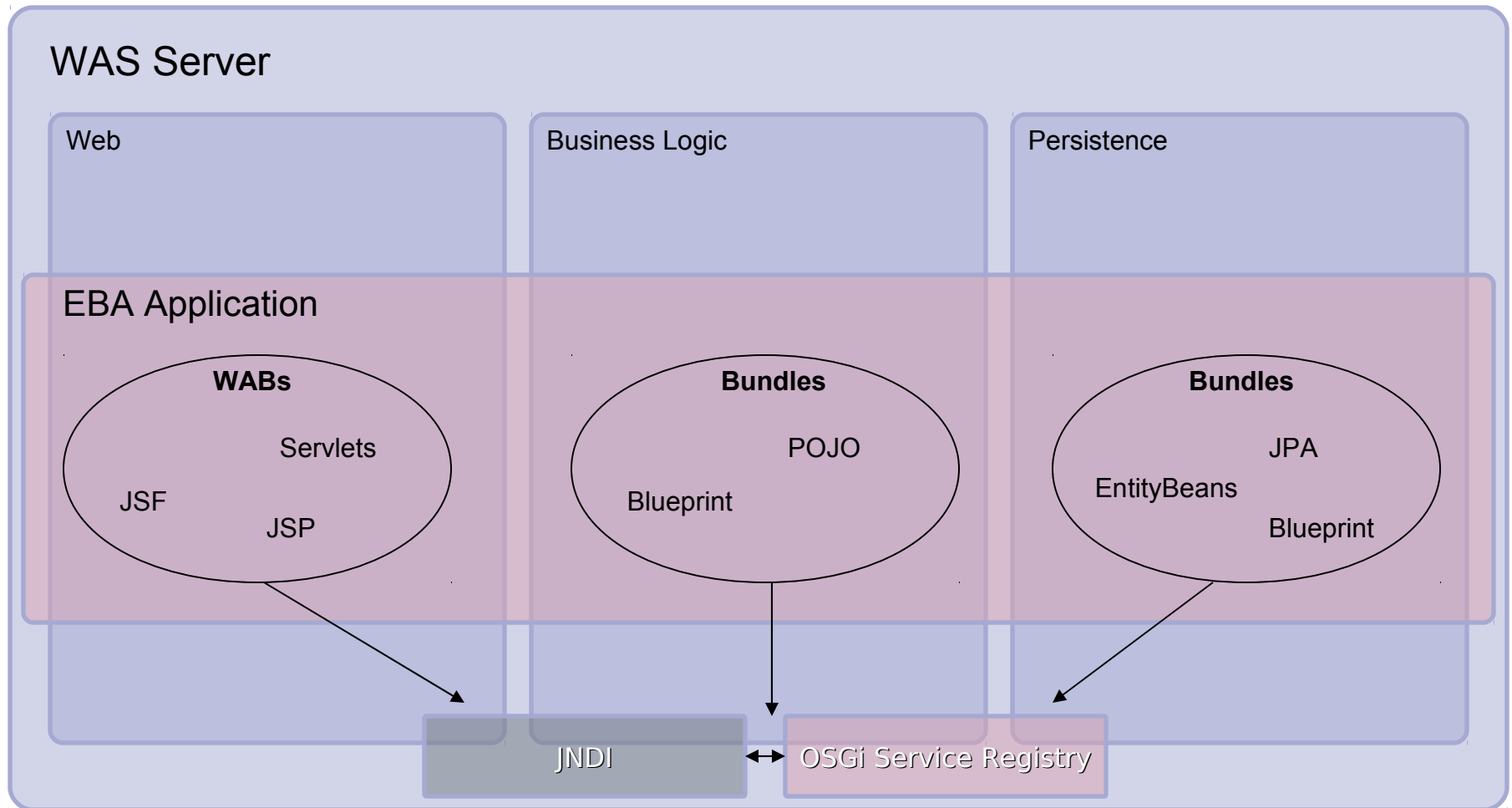
OSGi in the Application Server

- JEE Overview



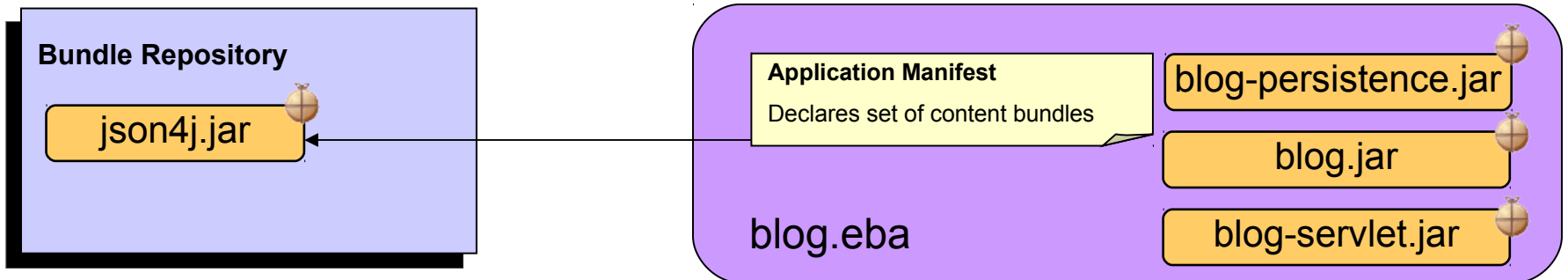
OSGi in the Application Server

- OSGi with JEE



Enterprise Bundle Archive (EBA)

- An isolated application consisting of a collection of bundles. Deployed as a logical unit in a “.eba” archive
 - An “OSGi Application”.
- Constituent bundles may be contained inside the archive or referenced from a bundle repository.
- Content specified in EBA-level application manifest – an APPLICATION.MF file.



Bundle Repository Config in WAS

Integrated Solutions Console Welcome Help | Logout

Cell=L32H83TNode01Cell, Profile=AppSrv01

View: All tasks

- Welcome
- Guided Activities
- Servers
- Applications
- Services
- Resources
- Security
- Environment
 - Virtual hosts
 - Update global Web server plug-in configuration
 - WebSphere variables
 - Shared libraries
 - Replication domains
- Naming
- OSGi bundle repositories
 - External bundle repositories
 - Internal bundle repository**
- System administration
- Users and Groups
- Monitoring and Tuning
- Troubleshooting
- Service integration
- UDDI

Internal bundle repository

The internal bundle repository can store bundles that are referenced by OSGi applications running in WebSphere Application Server. When an OSGi application is imported as an asset, the provisioner attempts to satisfy all its dependencies by using the contents of the asset, the contents of the internal bundle repository, and the contents of any available external bundle repositories.

Preferences

New Delete

Select	Bundle symbolic name	Bundle version
You can administer the following resources:		
<input type="checkbox"/>	com.ibm.json.java	1.0.0
<input type="checkbox"/>	org.apache.axis2	7.0.0
<input type="checkbox"/>	slf4j-api	1.5.6


Total 3

Help

Field help
For field help inform: select a field label or marker when the help is displayed.

Page help
[More information about page](#)

Application-centric Bundle Management

Integrated Solutions Console Welcome Help | Logout 

Cell=irobinsNode01Cell, Profile=AppSrv01 [Close page](#)

- [-] Applications
 - New Application
 - [-] Application Types
 - WebSphere enterprise applications
 - Business-level applications
 - Assets
- [+] Services
- [+] Resources
- [+] Security
- [-] Environment
 - Virtual hosts
 - [Update global Web server plug-in configuration](#)
 - [WebSphere variables](#)
 - [Shared libraries](#)
 - [Replication domains](#)
- [+] Naming
- [-] OSGi bundle repositories
 - [External bundle repositories](#)
 - [Internal bundle repository](#)

Assets

[Assets](#) > [com.ibm.ws.eba.example.blog.eba](#) > **Update bundle versions in this application**

Update the versions of the bundles that comprise this application.

Application bundle content

Symbolic name	Content type	Sharing	Deployed version	New version
com.ibm.ws.eba.example.blog	Bundle	Isolated	1.0.0	No preference ▾
com.ibm.ws.eba.example.blog.api	Bundle	Isolated	1.0.0	No preference ▾
com.ibm.ws.eba.example.blog.persistence	Bundle	Isolated	1.0.0	1.1.0 ▾
com.ibm.ws.eba.example.blog.web	Bundle	Isolated	1.0.0	No preference 1.0.0 1.1.0

Use bundle content

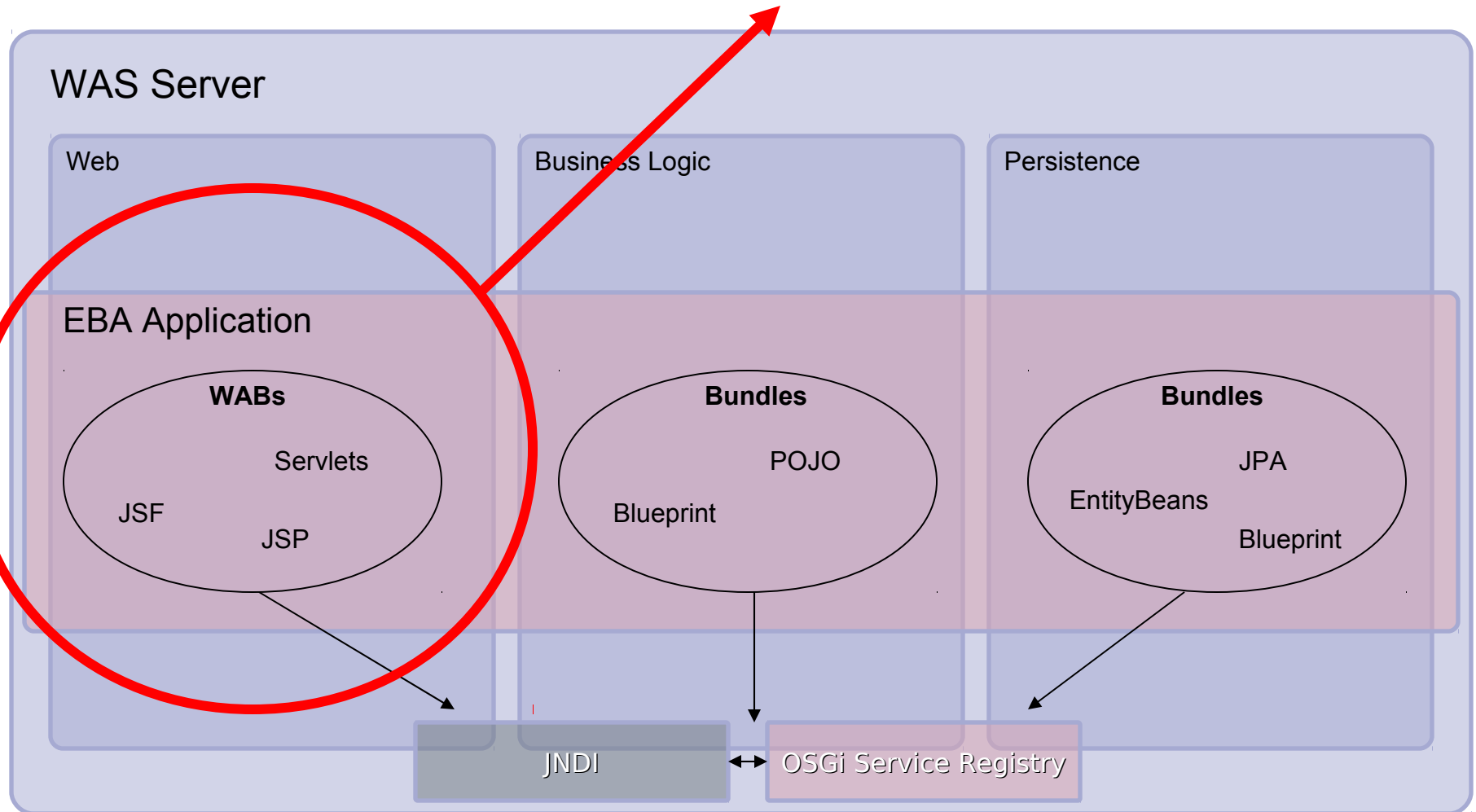
Symbolic name	Content type	Sharing	Deployed version	New version
com.ibm.json.java	Bundle	Shared	1.0.0	No preference ▾

OSGi Service Registry and JNDI

- OSGi services are published to and looked up from OSGi service registry.
 - Directly or from declarations in Blueprint XML
- Simplify integrating with existing JEE components:
 - OSGi Services registered in the OSGi Service Registry are also available in JNDI via the osgi:service URL scheme
 - Administered resources bound to JNDI are also published as services in the OSGi the Service Registry. The JNDI name is published as a service property called “osgi.jndi.service.name”

OSGi in the Application Server

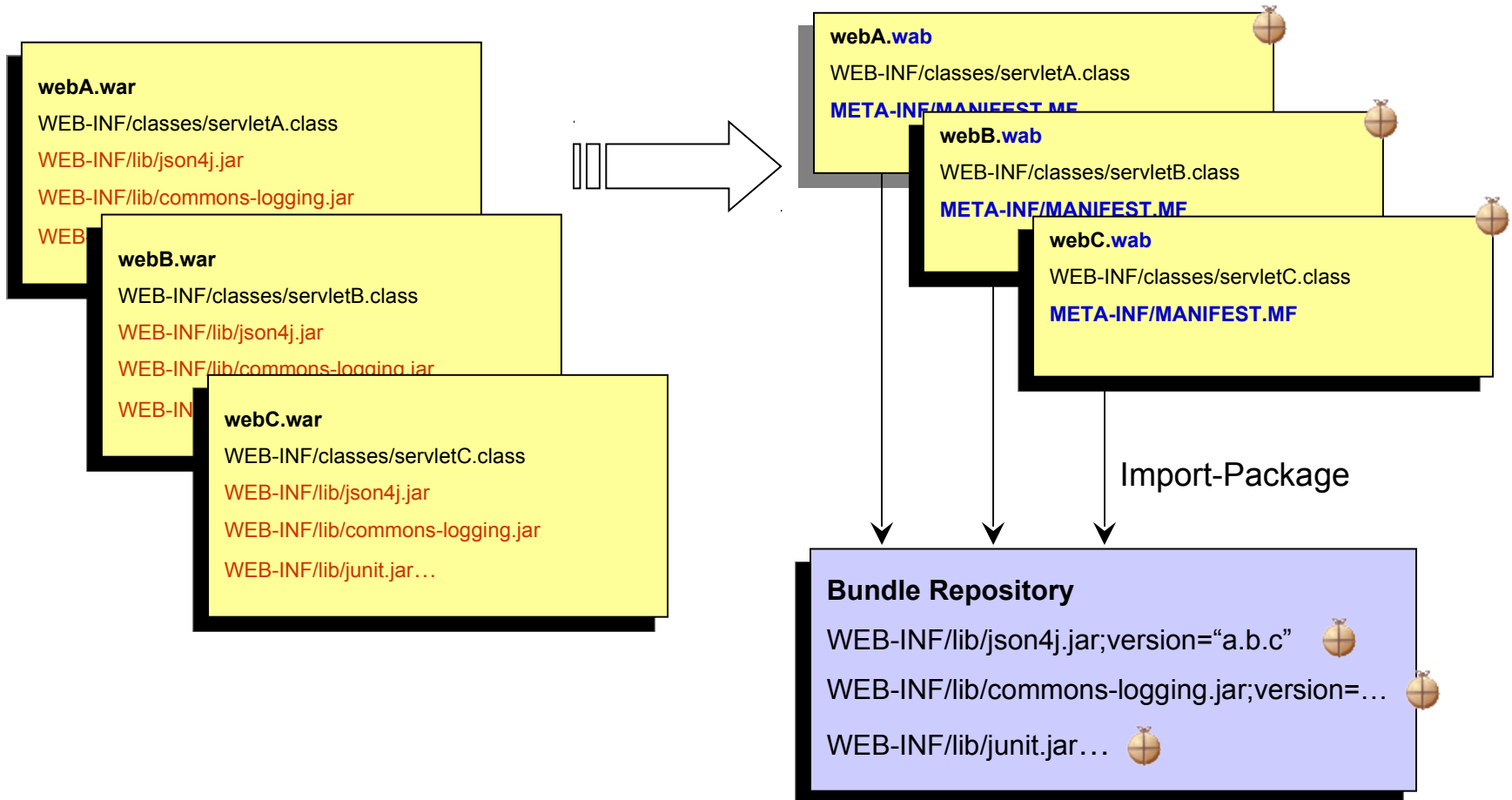
- Lets look closer at... **Web Applications**



Web Application Bundles (WABs)

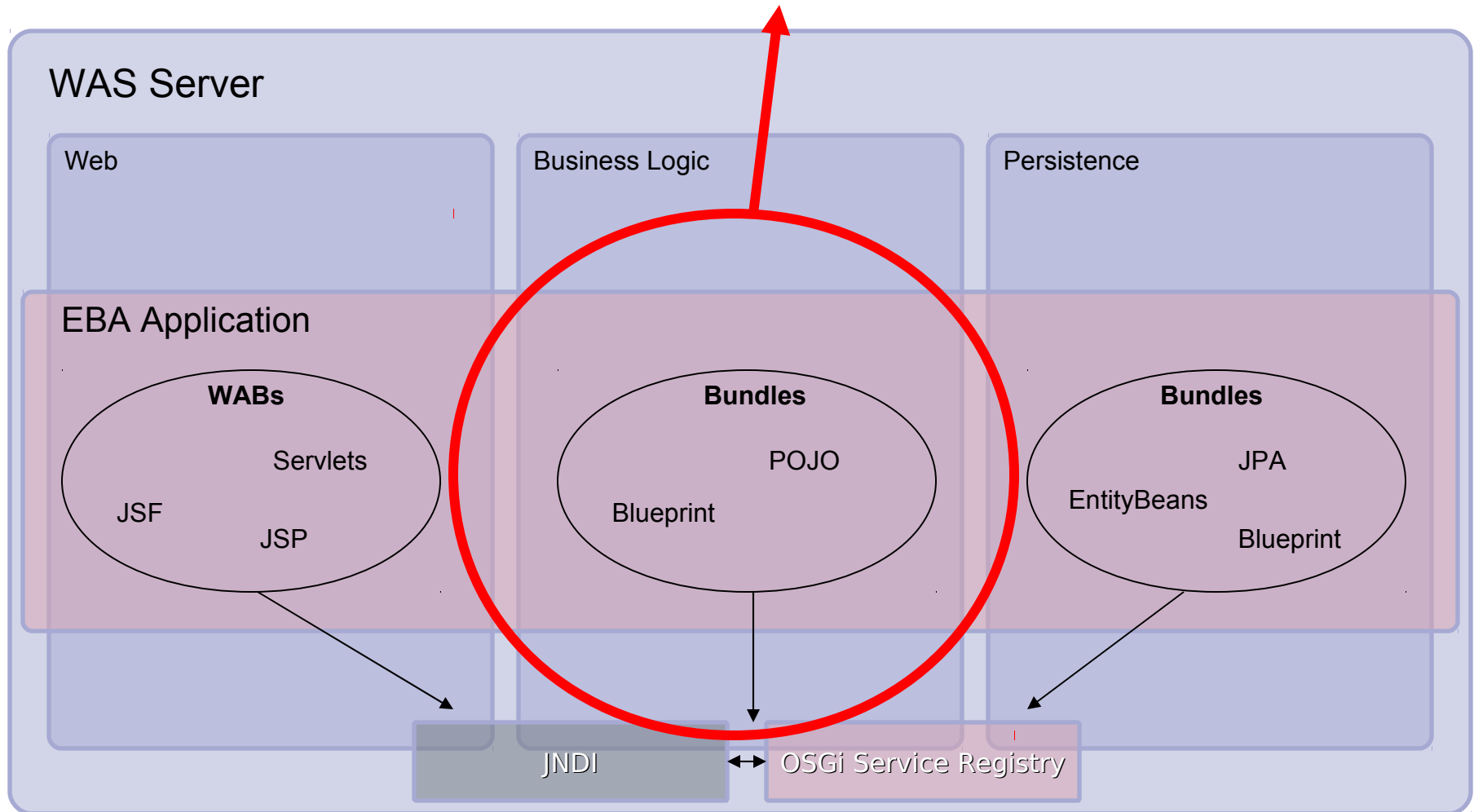
No Java code changes; WAR modules -> WAB bundles

Common, bundles may be easily factored out of the WARs and used at specific versions



OSGi in the Application Server

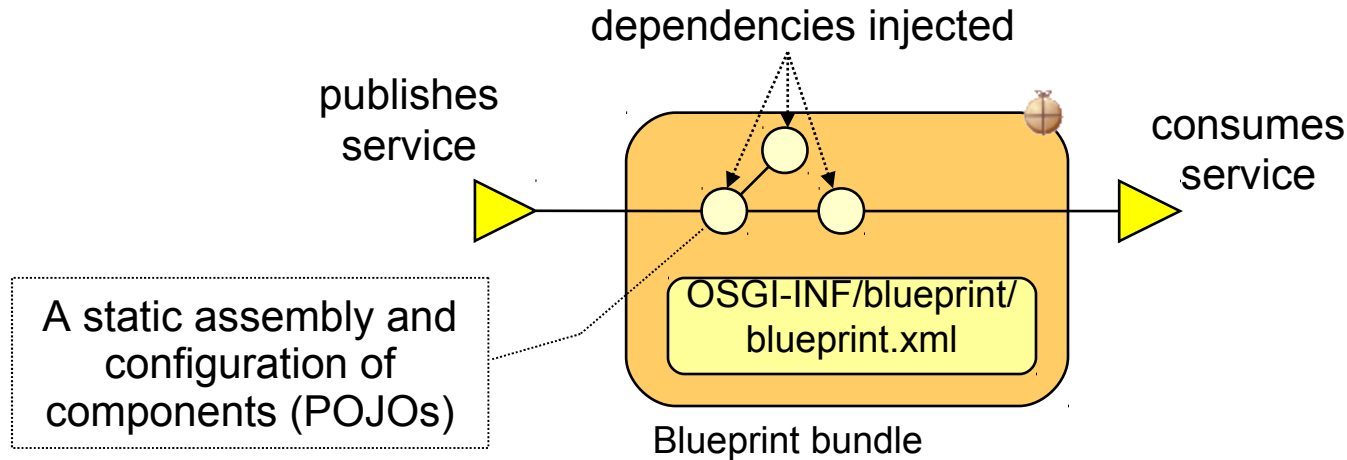
- Lets look closer at... **Business Logic**



Blueprint

- *Open Standard* version of the SpringFramework Dependency Injection component.
- Configuration and Dependencies declared in Blueprint XML.
- **In WAS, the Blueprint Container is a part of the Server runtime (SpringFramework is part of the application!)**

Blueprint Components and Services

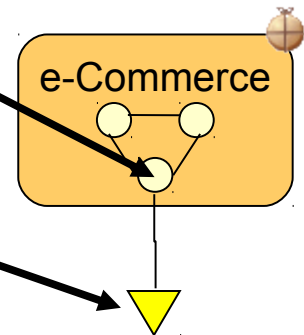


- *Standardized* Spring.
- Blueprint XML Extended for OSGi Services
- Simplifies unit test outside either Java EE or OSGi runtime

Exploiting Blueprint Components and Services

e-Commerce bundle

```
<blueprint>
  <bean id="shop" class="org.example.ecomm.ShopImpl">
    <property name="billingService" ref="billingService" />
  </bean>
  <reference id="billingService"
    interface="org.example.bill.BillingService" />
</blueprint>
```



```
public class ShopImpl {
  private BillingService billingService;
  void setBillingService(BillingService srv) {
    billingService = srv;
  }

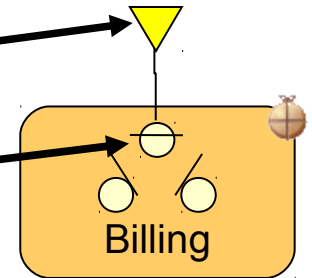
  void process(Order o) {
    billingService.bill(o);
  }
}
```

- injected service reference
- service can change over time
- can be temporarily absent without the bundle caring
- managed by Blueprint container

Exploiting Blueprint Components and Services

Billing service bundle

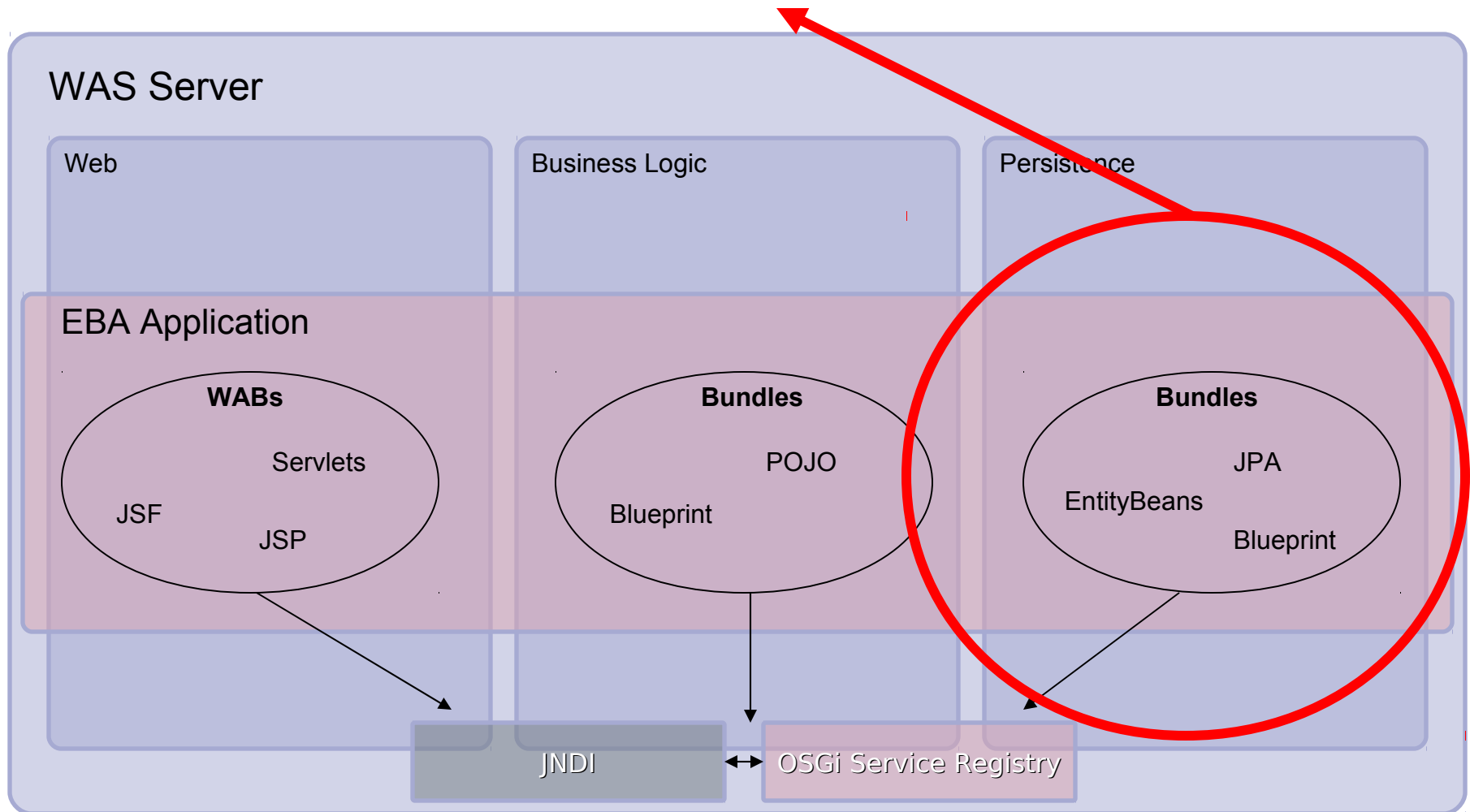
```
<blueprint>
  <service ref="service" interface =
    "org.example.bill.BillingService" />
  <bean id="service" scope="prototype"
    class="org.example.bill.impl.BillingServiceImpl" />
</blueprint>
```



```
public interface BillingService {
    void bill(Order o);
}
```

OSGi in the Application Server

- Lets look closer at... Persistence Layer



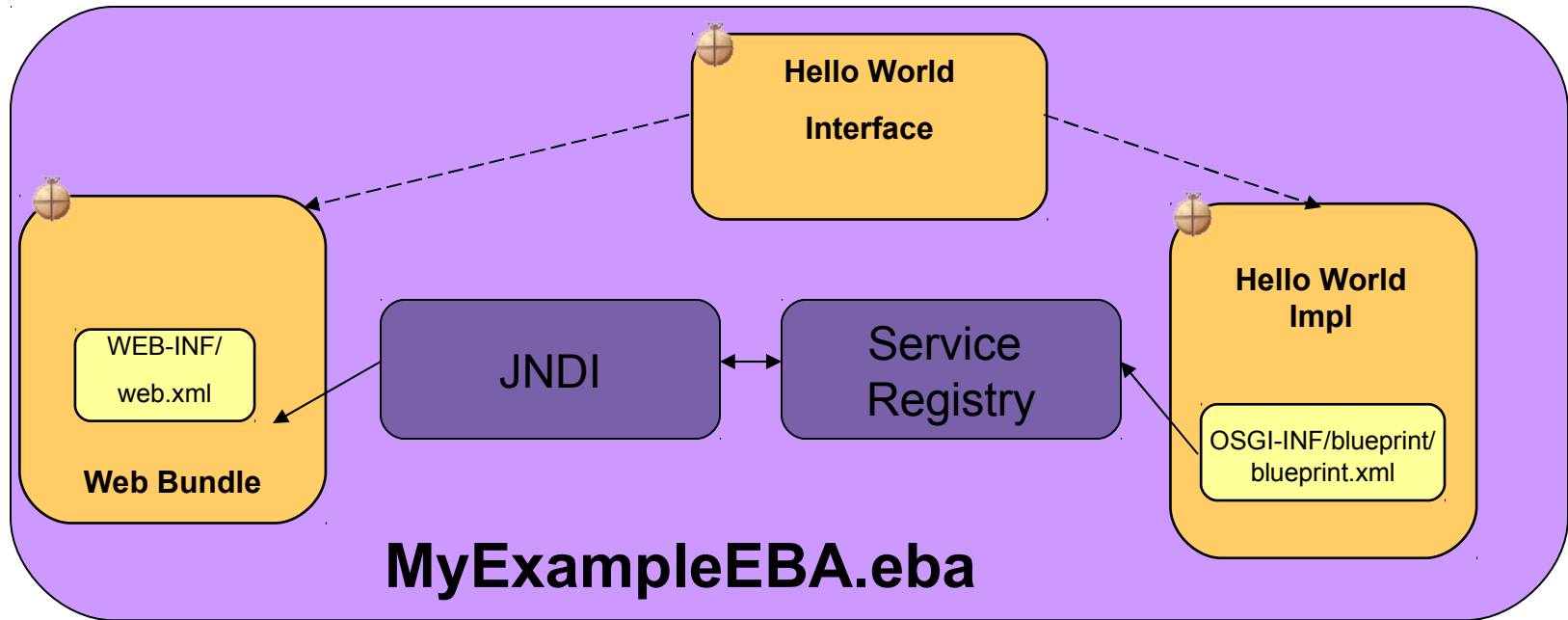
Blueprint Persistence and Transactions

- OpenJPA is default persistence provider in WebSphere
- Container managed JPA support integrated into Blueprint container:
 - @PersistenceUnit or @PersistenceContext (managed)
 - or <jpa:unit>, <jpa:context> bean property injection
 - Familiar development experience for JPA developers
 - Load-time enhancement of Entity classes
- Same container managed transaction attributes as EJBs:
 - Required, RequiresNew, Mandatory, NotSupported, Supports, Never

```
<blueprint>
  <bean id="shop" class="org.example.ecomm.ShopImpl">
    <jpa:context property="em" unitname="myUnit"/>
    <tx:transaction method="*" value="Required"/>
  </bean>
</blueprint>
```

OSGi Enterprise Applications : An Example

Example EBA : “Hello World” Application

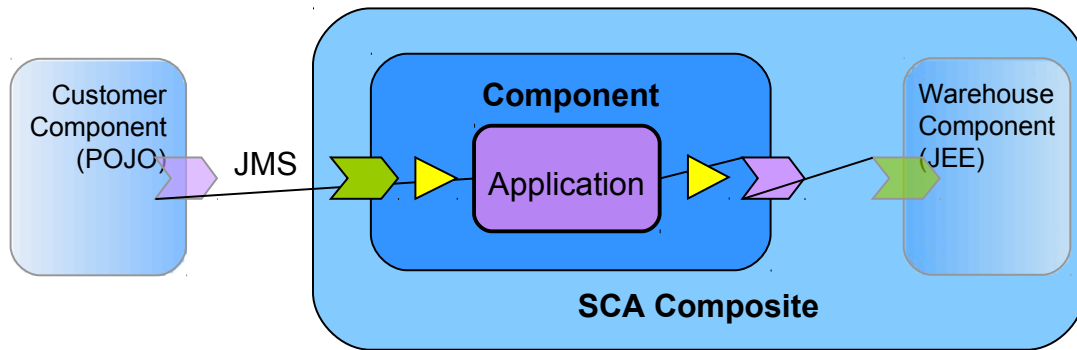


```
Manifest-Version: 1.0
Application-ManifestVersion: 1.0
Application-Name: MyExampleEBA
Application-SymbolicName: MyExampleEBA
Application-Version: 1.0.0
Application-Content:
  MyInterfaceBundle;version=1.0.0,
  MyImplementationBundle;version=1.0.0,
  MyWebBundle;version=1.0.0
```

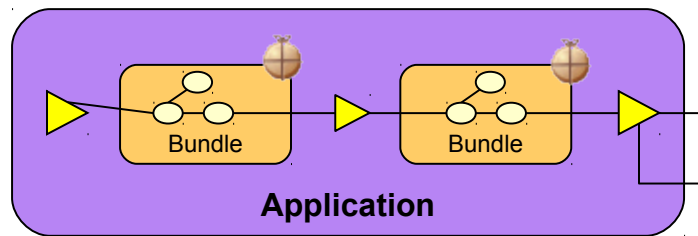
APPLICATION.MF
Contents

Enterprise OSGi : Further Support

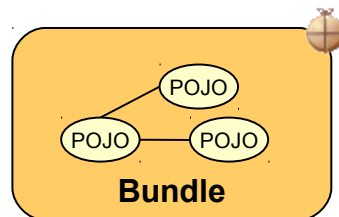
OSGi and SCA: the assembly food chain



SCA Composite assembled from heterogeneous components including an **OSGi Application** component, and integrated through SCA services with configurable bindings (JMS, web services...).



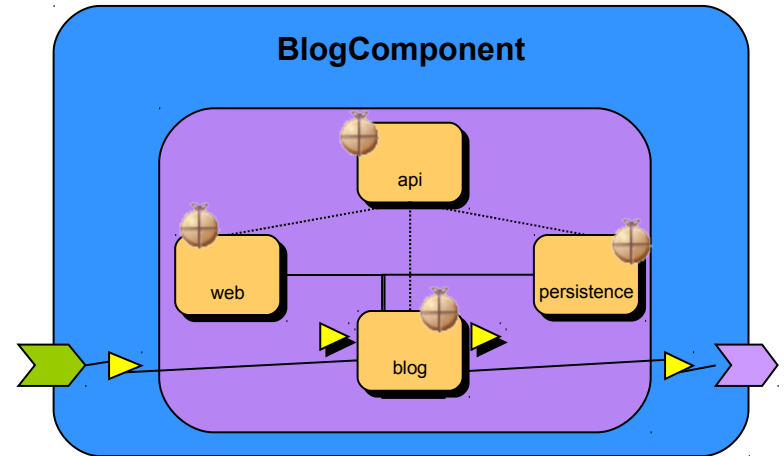
OSGi Bundles assembled in an **OSGi Application** and integrated through services in the OSGi service registry



POJOs assembled using a Blueprint context and scoped by an **OSGi Bundle**.

SCA integration: implementation.osgiapp

```
Manifest-Version: 1.0
Application-ManifestVersion: 1.0
Application-Name: Aries Blog
Application-SymbolicName: com.ibm.ws.eba.example.blog.app
Application-Version: 1.0
Application-Content:
  com.ibm.ws.eba.example.blog.api;version="1.0.0",
  com.ibm.ws.eba.example.blog.persistence;version="1.0.0",
  com.ibm.ws.eba.example.blog.web;version="1.0.0",
  com.ibm.ws.eba.example.blog;version="1.0.0"
Use-Bundle: com.ibm.json.java;version="[1.0.0,2.0.0)"
Application-ExportService:
  com.ibm.ws.eba.example.blog.Blog
Application-ImportService:
  com.ibm.ws.eba.example.blog.UserAuthorization
```



```
<component name="com.ibm.ws.aries.example.BlogComponent">
  <service name="bloggingService">
    <interface.java interface="com.ibm.ws.eba.example.blog.Blog" />
    <binding.ws
      port="http://www.blogging.org/BlogService#wSDL.endpoint(BlogService/BlogServiceSOAP)" />
  </service>
  <reference name="userAuthorization">
    <interface.java interface="com.ibm.ws.eba.example.blog.UserAuthorization" />
  </reference>
  <sfp:implementation.osgiapp applicationSymbolicName="com.ibm.ws.aries.example.blog.app"
    applicationVersion="1.0.0" />
</component>
```

RAD v8 OSGi Application Development Support

The screenshot shows the Rational Application Developer (RAD) v8 interface. The main window is titled "Java EE - com.ibm.ws.eba.example.blog.app/META-INF/APPLICATION.MF - Rational® Application Developer for WebSphere® Software". The "Enterprise Explorer" on the left shows a project structure with "com.ibm.ws.eba.example.blog" and "com.ibm.ws.eba.example.blog.api". The "OSGi Application Manifest" editor is open, displaying the following configuration:

General Information:
Specify the fields below for this OSGi application.

Name: Aries Blog
Symbolic name: com.ibm.ws.eba.example.blog.app
Version: 1.0
Manifest Version: 1.0

Imports:
Specifies a list of services to be imported.

Contained Bundles:
Specifies the list of OSGi bundles to be included in this OSGi application.

- com.ibm.ws.eba.example.blog.api 1.0
- com.ibm.ws.eba.example.blog.persist
- com.ibm.ws.eba.example.blog.web 1.
- com.ibm.ws.eba.example.blog 1.0.0

Exports:
Specifies a list of services to be exported from this OSGi application.

The "New" wizard dialog is open, showing the "Select a wizard" screen. The "OSGi" folder is expanded, and the "OSGi Application Project" wizard is selected. The "Next >" button is highlighted.

When?

- Now!
- Where is OSGi going?
 - Significant investment by IBM so far in **technology**
 - Significant investment by IBM so far in **standards**
 - Early feedback is extremely promising

Summary

- Modular Programming is not supported fully in :
 - JAVA (SE)
 - JAVA (EE)
- The Solution for JAVA (SE) applications... OSGi
- The NEW Solution for JAVA (EE) applications... Enterprise OSGi
- Fully implemented in WASv7 with the **OSGi Feature Pack**, and in WASv8 Beta
- Based on Open Standards.
- Lots of interest from the industry.

Useful Links

- OSGi Best Practices Developerworks Article

http://www.ibm.com/developerworks/websphere/techjournal/1007_charters/1007_charters.html?ca=drs-

- Spring Application Migration

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.osgifep.multiplatform.doc/topics/ta_mig.html

- Resource Hub (articles, tutorials, redbooks, forums)

<http://www.ibm.com/software/websphere/osgi>

- Team Blogs/Twitter/YouTube

www.ianrobinson.blogspot.com

www.devangelist.blogspot.com

www.youtube.com/user/EnterpriseOSGi

@sjmaple @notatibm @TimothyWard

Questions?

Copyright and Trademarks

© IBM Corporation 2010. All rights reserved. IBM, the IBM logo, ibm.com and the globe design are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml. Other company, product, or service names may be trademarks or service marks of others.