OSGi in WebSphere : The Story so far…

Chris Wilkinson, IBM Hursley

chris.wilkinson@uk.ibm.com

Disclaimer

Agenda

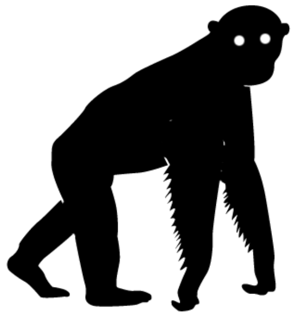
- A Brief history of OSGi…
- A Brief history of Enterprise OSGi…
- Enterprise OSGi in WebSphere Application Server
- What’s new in the WAS85 Beta?
  - EJB Support
- Migrating to OSGi

A Brief history of OSGi...

A (very) brief history of programming

- In the beginning, there were bits ...

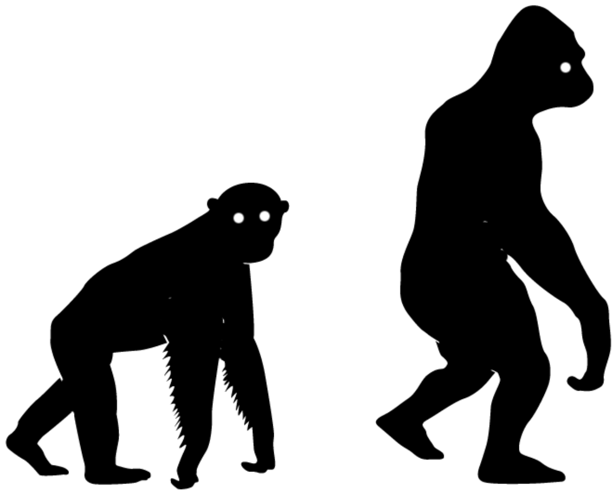
```
0101011101001001010010010011011101101101
```



A (very) brief history of programming

- Then came words ...

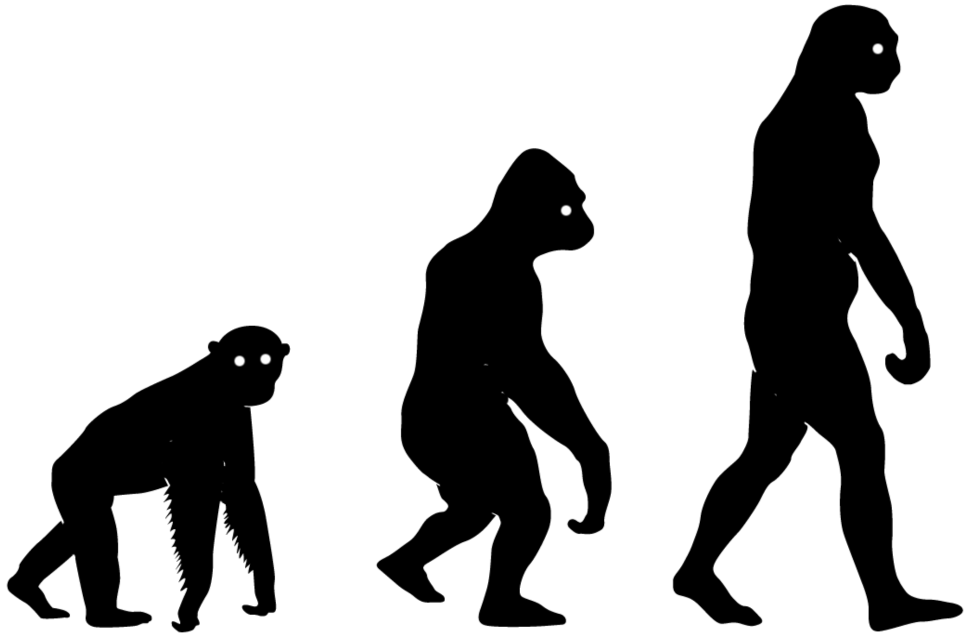
```
    mov     ah,9
    mov     dx,offset hello_message
    int     21h
```



A (very) brief history of programming

- ... functions and libraries ...

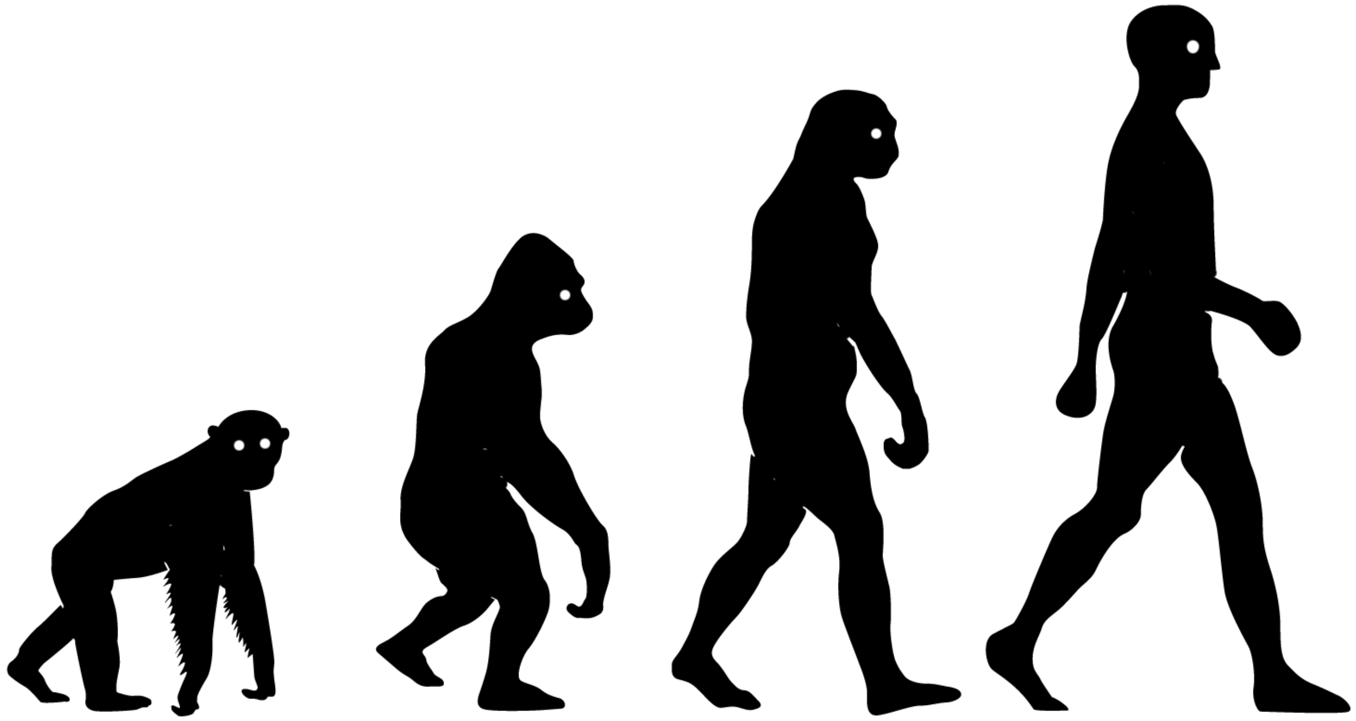
```
#include <stdio.h>
main() {
      printf ("Hello World!\n");
}
```



A (very) brief history of programming

- ... objects ...

```
public class HelloWorldSayer {

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

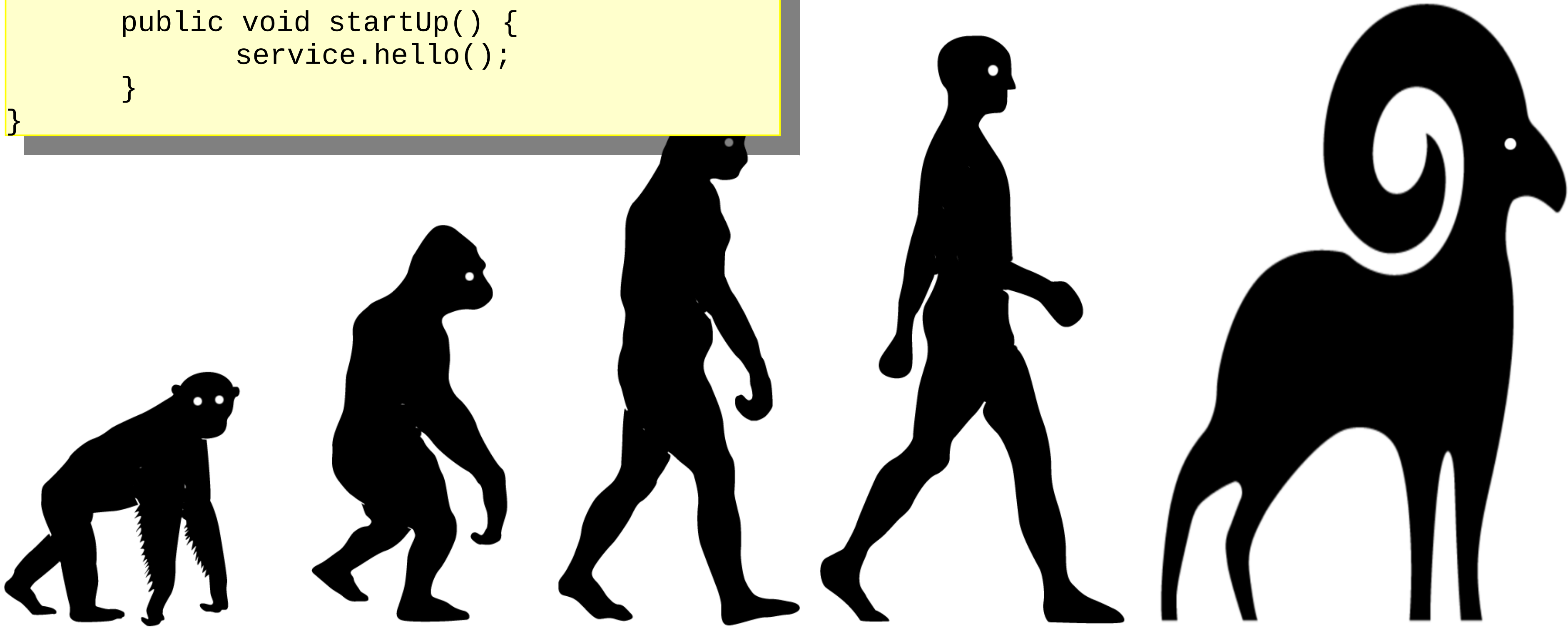


A (very) brief history of programming

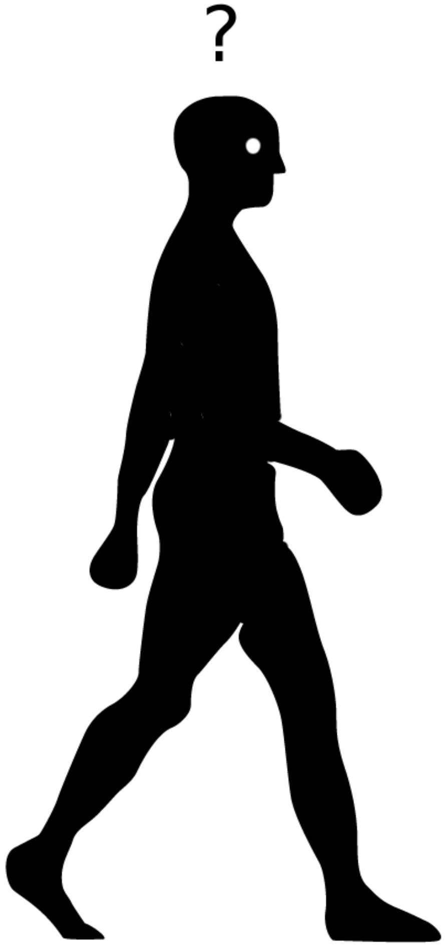
- ... and then – OSGi

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

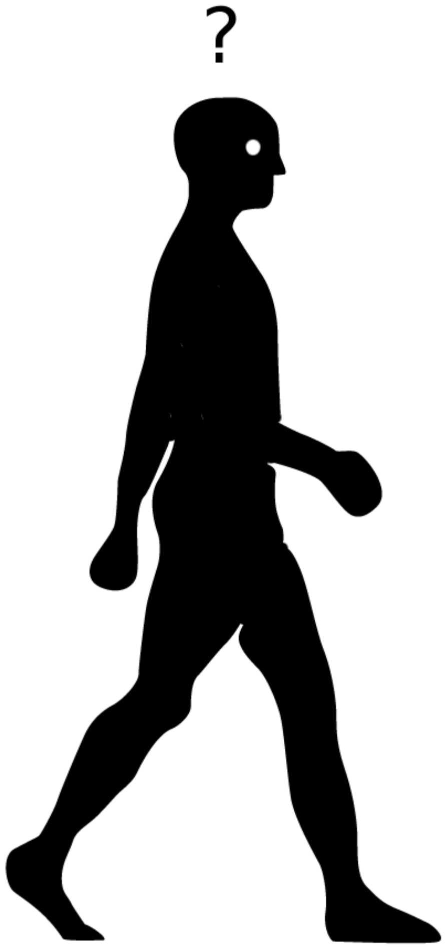
```
public class HelloWorldClient {
      HelloWorldService service = null;

      public void startUp() {
            service.hello();
      }
}
```



What was wrong with where we were?

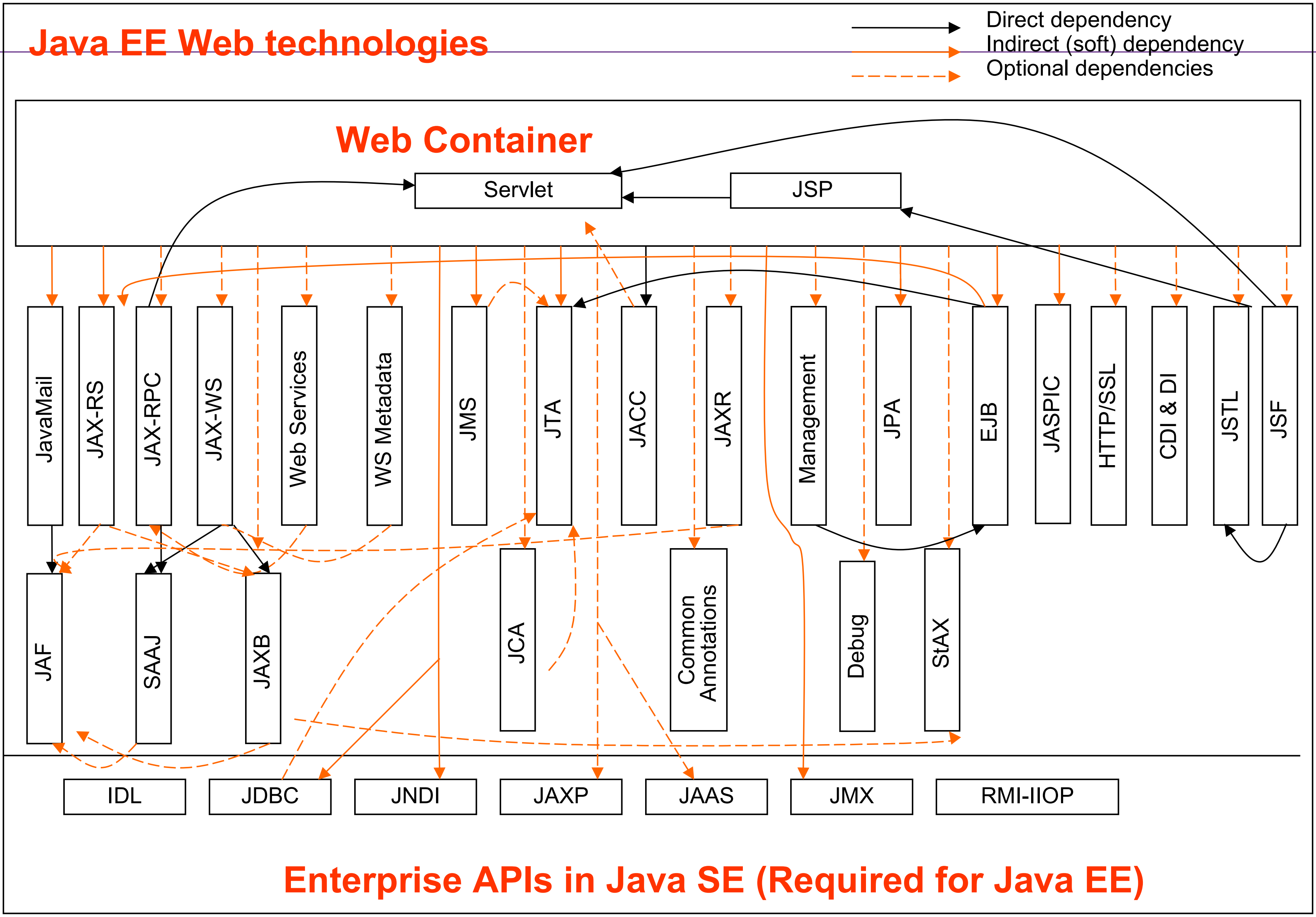


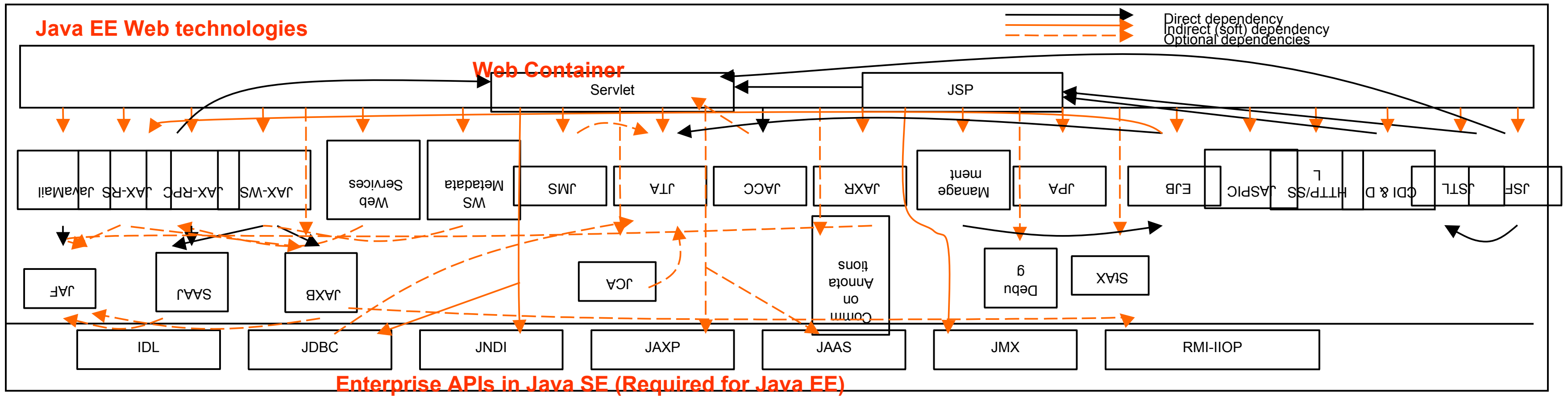
What was wrong with where we were?



- A question of scale

Java EE Web technologies





HTTP Session Replication

Connection Management

Persistence Providers

User Registries

Event Mgmt

Logging and Tracing

MBean Server

Admin Console and scripting

DataSources

Messaging Engines

Security

Transport Channels

Workload Management

HTTP Serving and routing

High Availability Management

Transaction Logging and Recovery

Thread management

Classloading

Provisioning

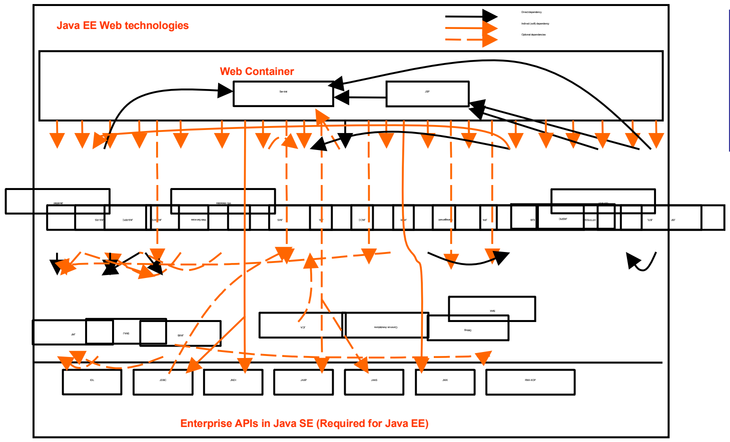
Monitoring

Policy and SLA mgmt

Configuration

...

Beyond Java EE…



Batch

Telephony

BPM

Dynamic Scripting

SCA

Business Rules

Complex Event Processing

…

HTTP Session Replication

Connection Management

Persistence Providers

User Registries

Event Mgmt

Logging and Tracing

MBean Server

Admin Console and scripting

DataSources

Messaging Engines

Security

Transport Channels

Workload Management

HTTP Serving and routing

High Availability Management

Transaction Logging and Recovery

Thread management

Classloading

Provisioning

Monitoring

Policy and SLA mgmt

Configuration

…

Requirements of Modular Programming

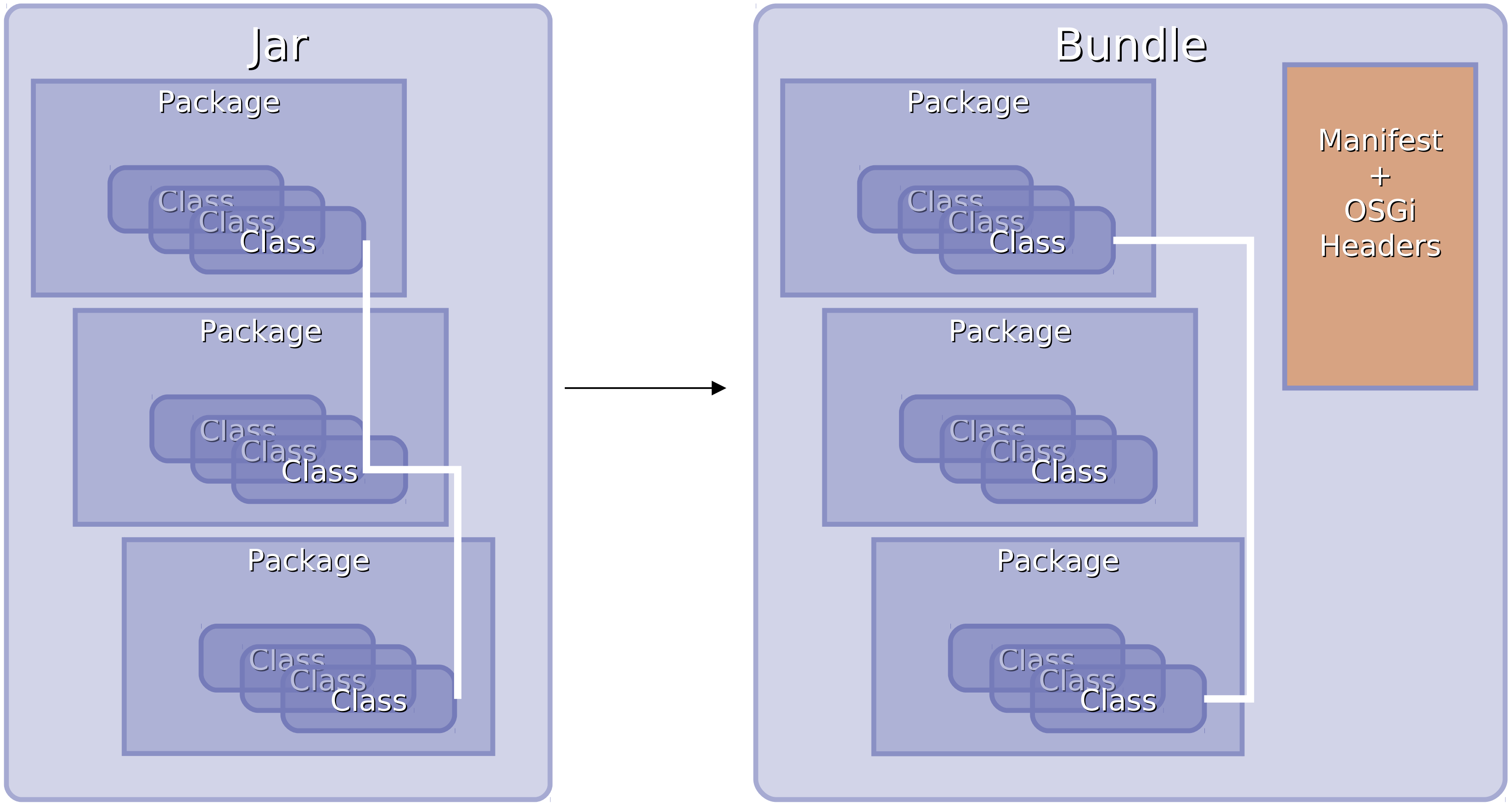
- Module Re-use
  - Reduce development time
- Module Sharing
  - Reduce Disk Footprint
  - Reduce Memory Footprint
- Module Access
  - Controlled access to specific contents
  - Ease of Development
- Module Maintenance
  - Loose-Coupling enables easy module updates
  - Versioning policy

Java Modularity Limitations

- Unit of Modularity is a JAR
- No explicit dependency control between JARs
  - Which packages does my JAR need?
  - Which JARs use my packages?
- At the mercy of the Classpath!

Modular Programming with OSGi

- Introducing the **OSGi Bundle**



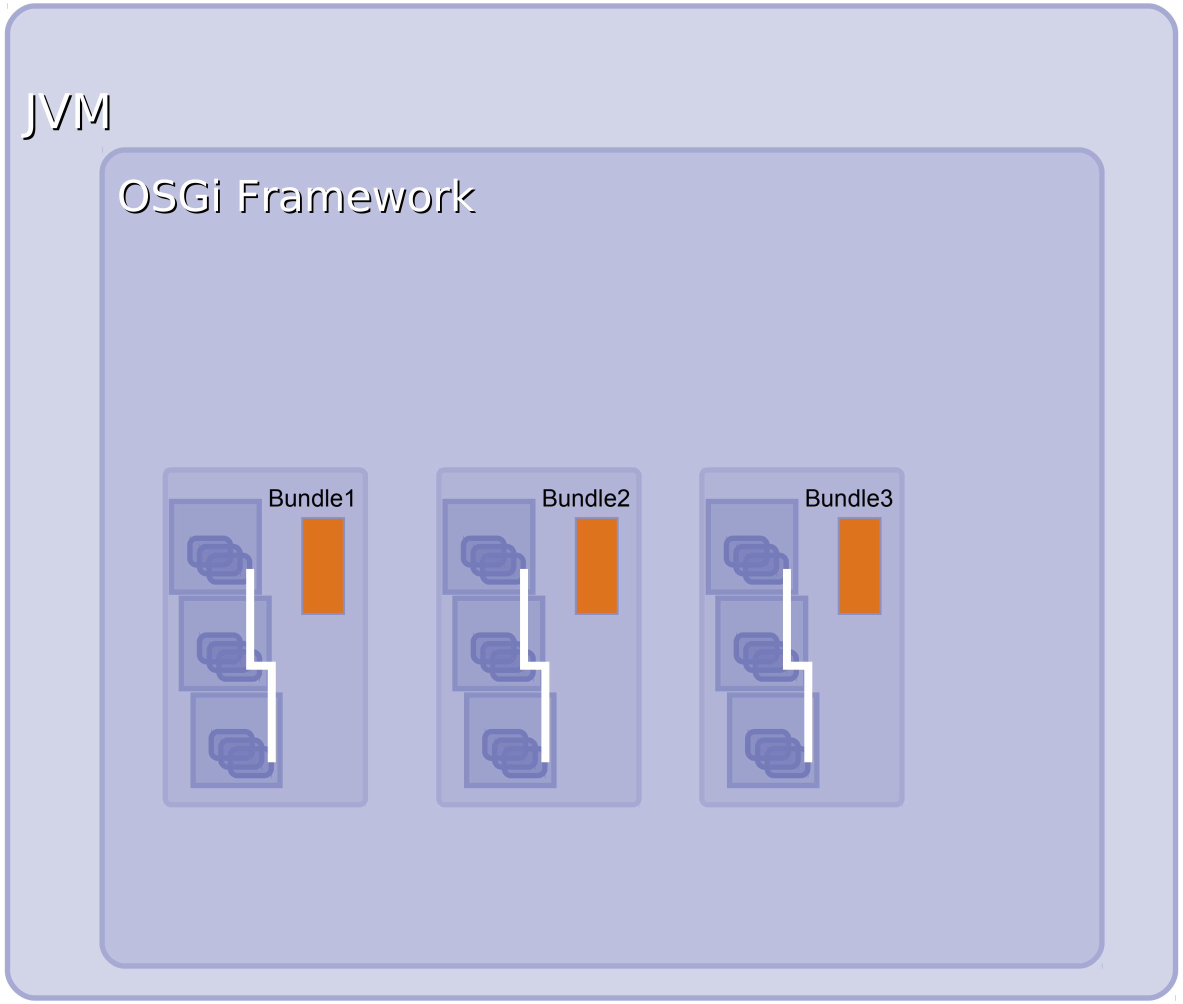
OSGi : The Bundle Manifest

- **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?

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

Modular Programming with OSGi

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

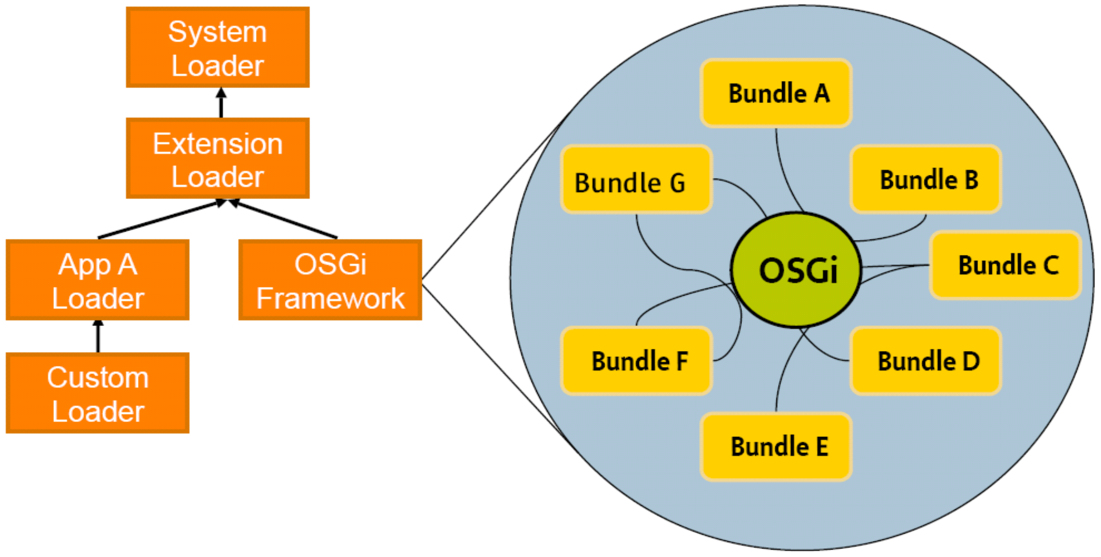


Provides 3 Layers

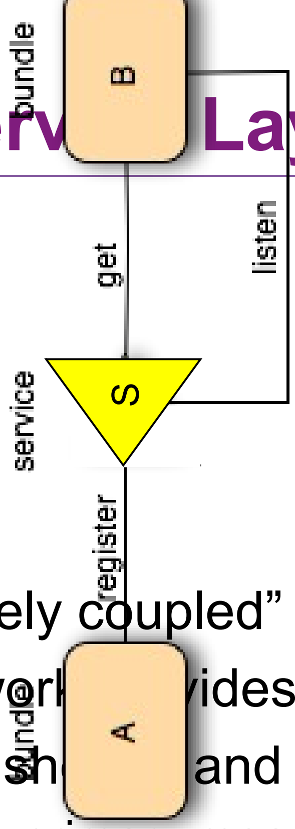
- Module
- Lifecycle
- Services

OSGi : The Module Layer

- **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



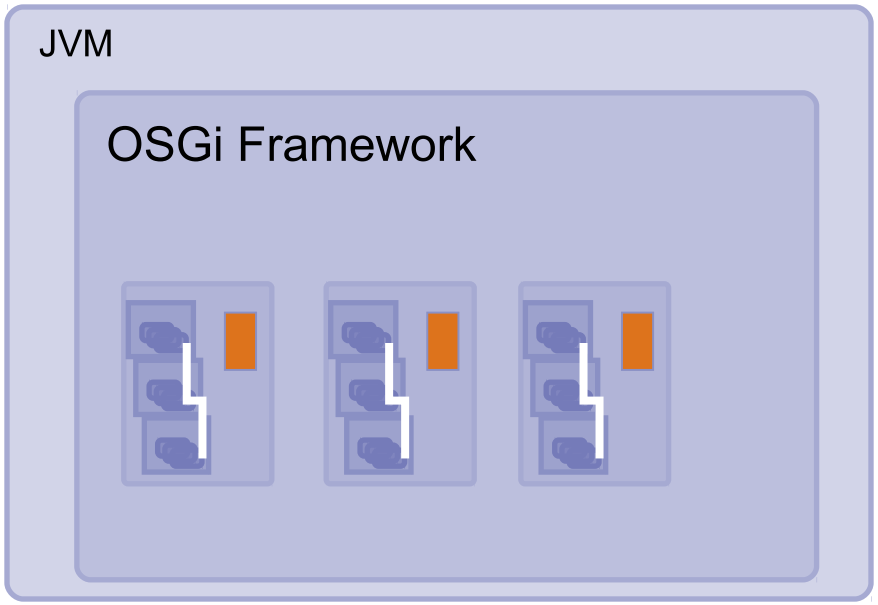
OSGi : The Serv Layer



- Bundles are “loosely coupled” by communicating via Services.
- The OSGi framework vides a non-durable “Service Registry”
- Services are publish 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.

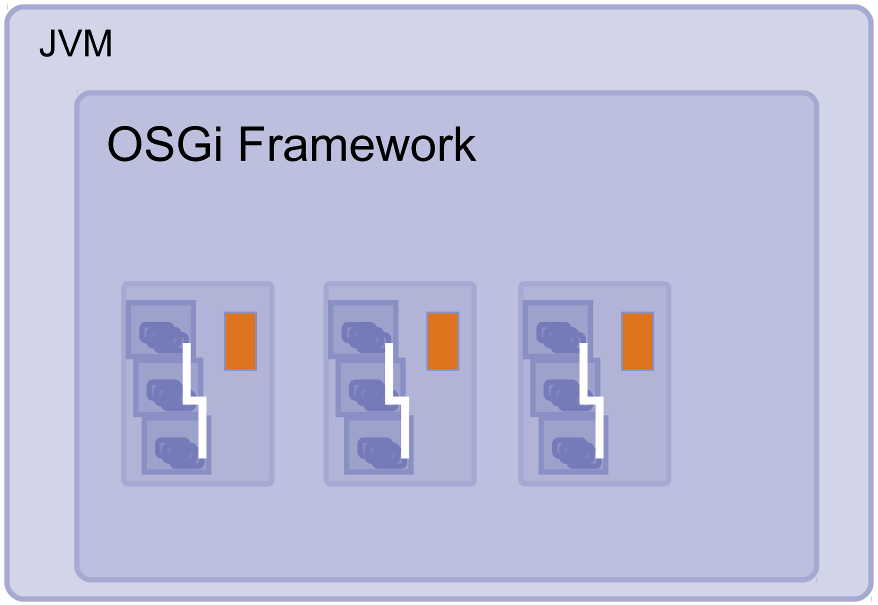
Modular Programming with OSGi

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



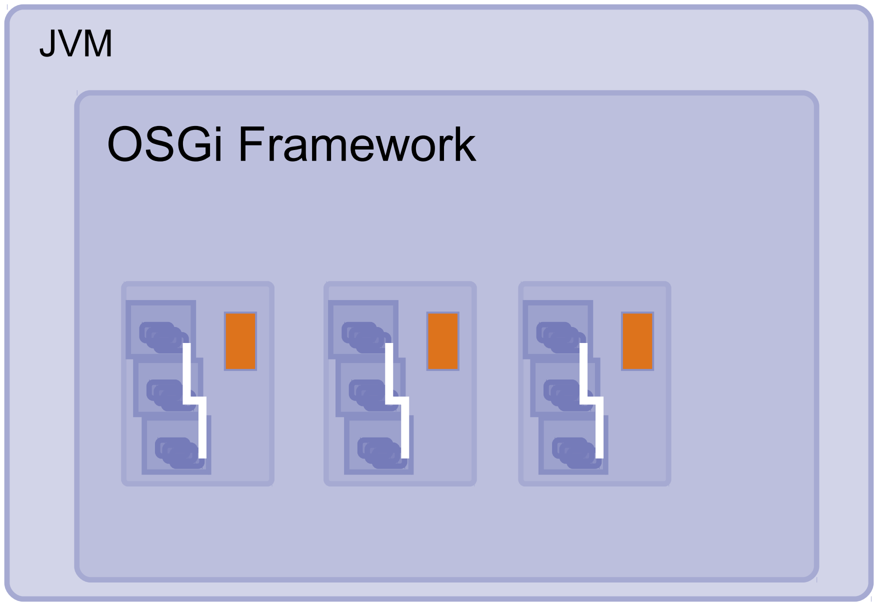
Modular Programming with OSGi

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



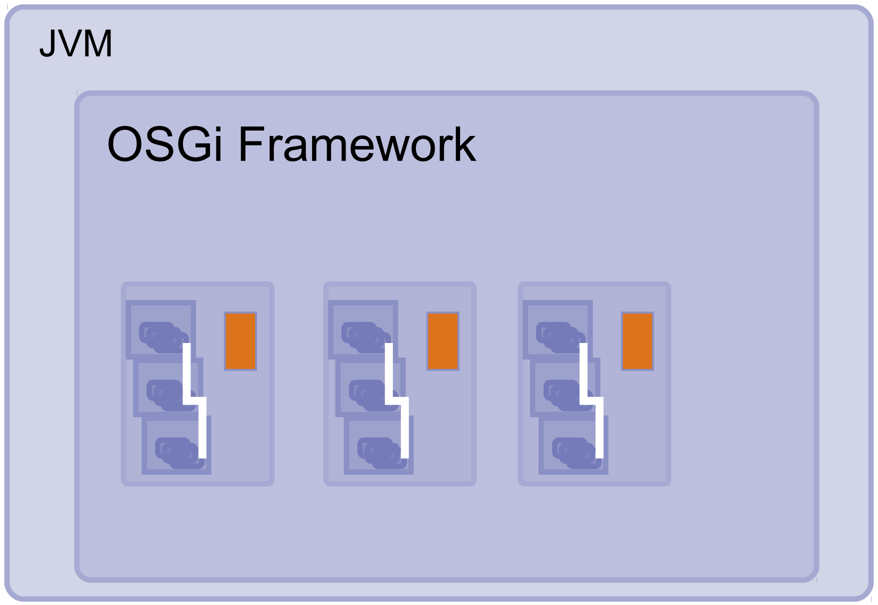
Modular Programming with OSGi

- **Module Sharing**
  - The OSGi framework controls classloading to allow a single bundle instance to be shared amongst multiple applications.



Modular Programming with OSGi

- **Module Maintenance**
  - The bundle manifest specifies a version.
  - Dependencies requirements can specify a version *range*.
  - Bundles with different versions can exist in the framework (We can have multiple versions of classes in use at same time in same JVM!)
  - Using services provides plug points for extending and updating



OSGi Origins : Modular Programming

OSGi Java (SE) Applications

- Module Sharing ✓
- Module Re-use ✓
- Module Maintenance ✓
- Module Access ✓

OSGi Design Patterns

- Best Practices IBM DeveloperWorks article :

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

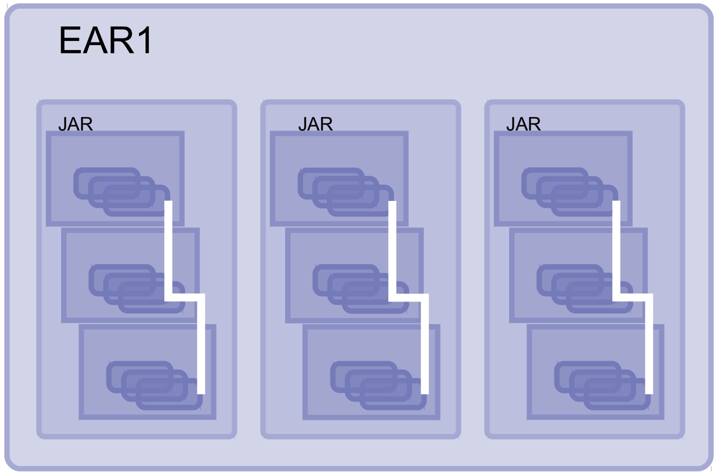
OSGi : An Open Standard

- 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!

A Brief history of Enterprise OSGi...

OSGi Origins : Modular Programming in JEE?

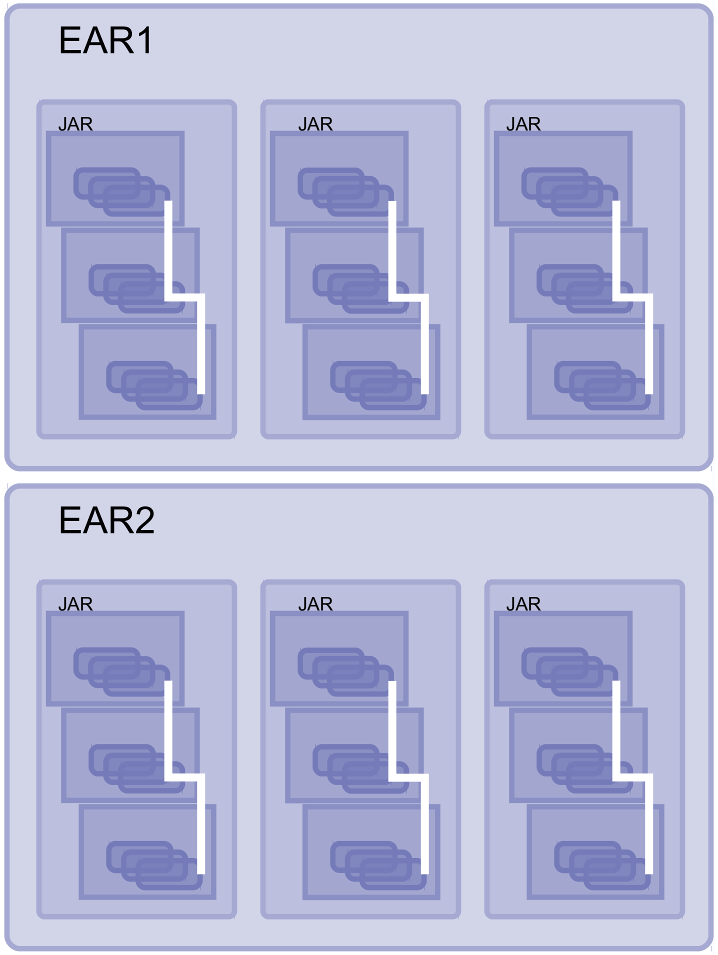
- Same Java modularity issues exists *inside* each app.



OSGi Origins : Modular Programming in JEE?

- Same Java modularity issues exists *inside* each app.

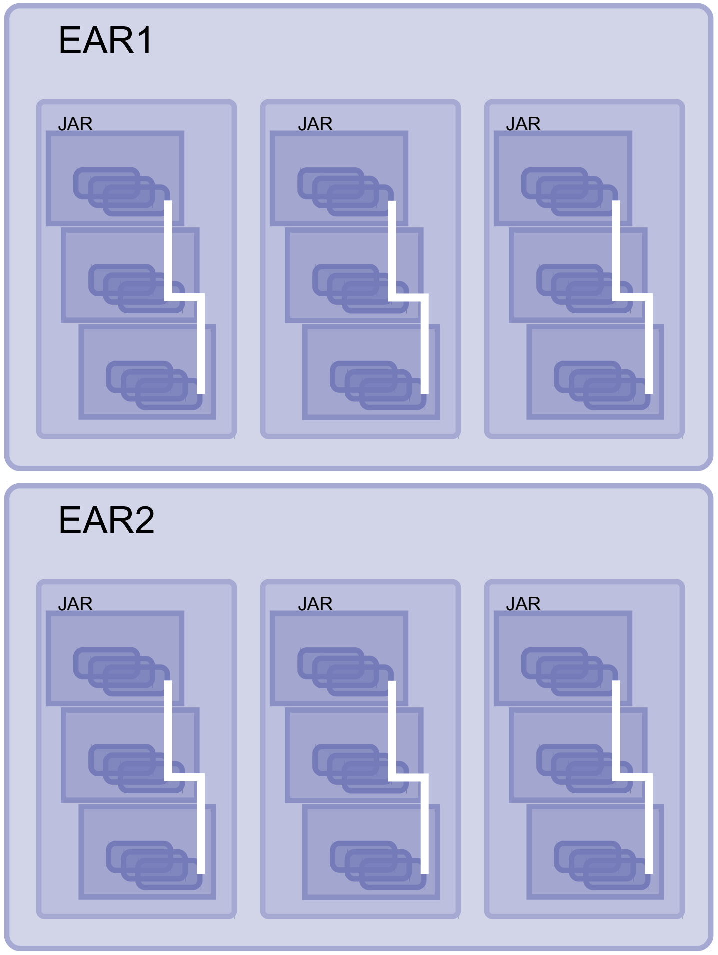
- **…and extra modularity issues between application modules!**



OSGi Origins : Modular Programming in JEE?

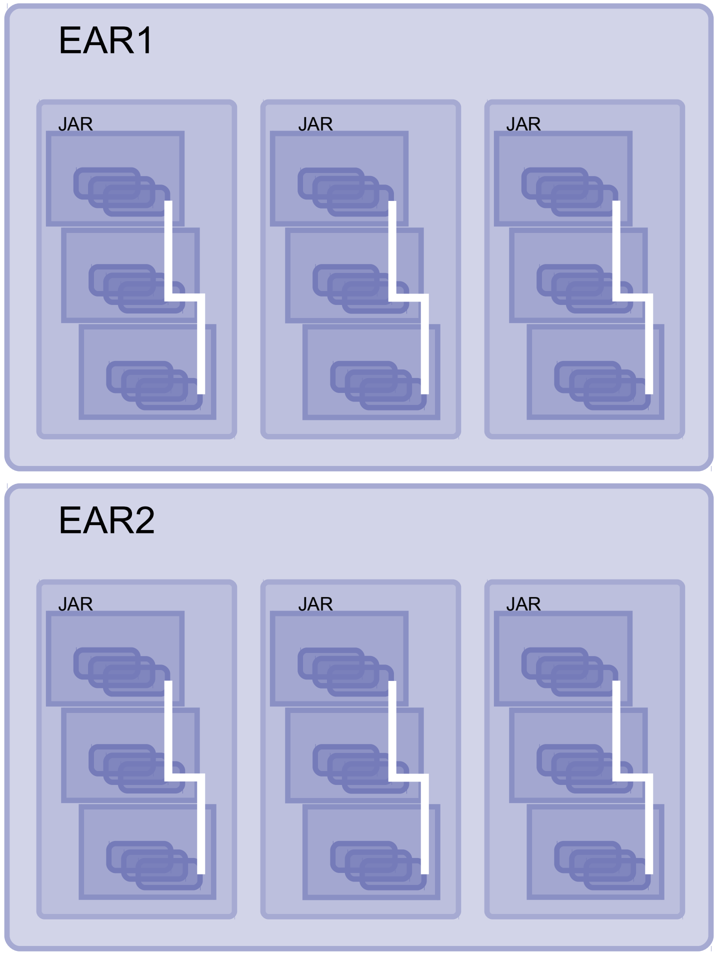
- **Module Sharing**
  - Applications are **isolated**.
  - Multiple copies of common JARs in each EBA
  - Wasted Disk space!
  - Wasted Memory!

- Some help provided via WAS shared libraries but config is static and difficult to set up.

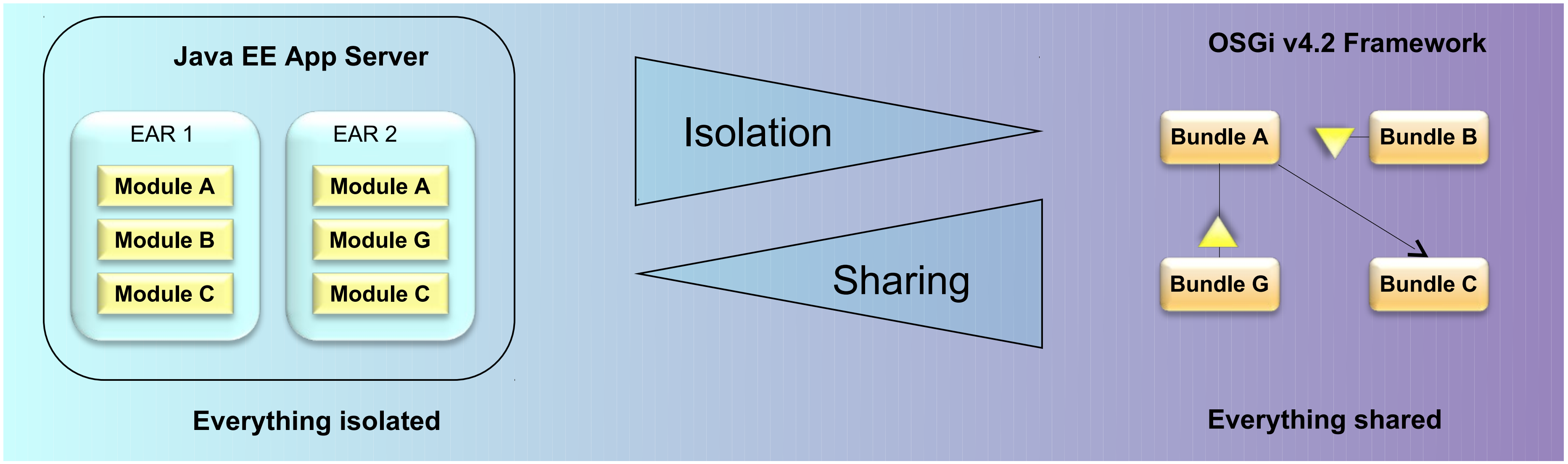


OSGi Origins : Modular Programming in JEE?

- **Module Maintenance**
  - Updating a JAR inside an EAR involves stopping entire application due to tight coupling of JARs.
  - Large maintenance overheads.



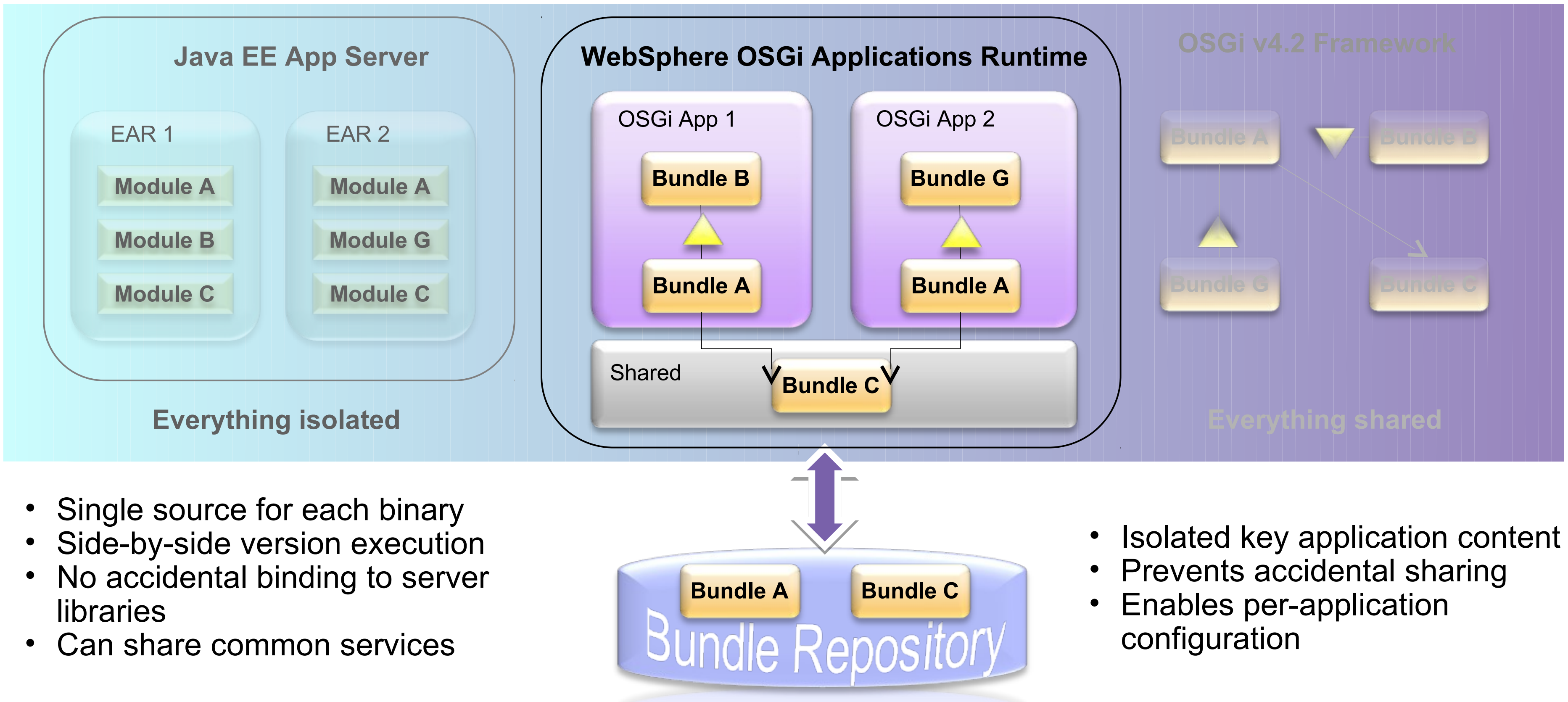
OSGi Applications: Isolation versus Sharing



- Which applications use module A?
- How many version of module C do I have deployed?
- Is module G being picked up from EAR 2 or the server runtime?

- Where are my applications?
- Did I intend application 1 to use application 2?
- How do I configure bundle A uniquely for each application?

Best of Both Worlds



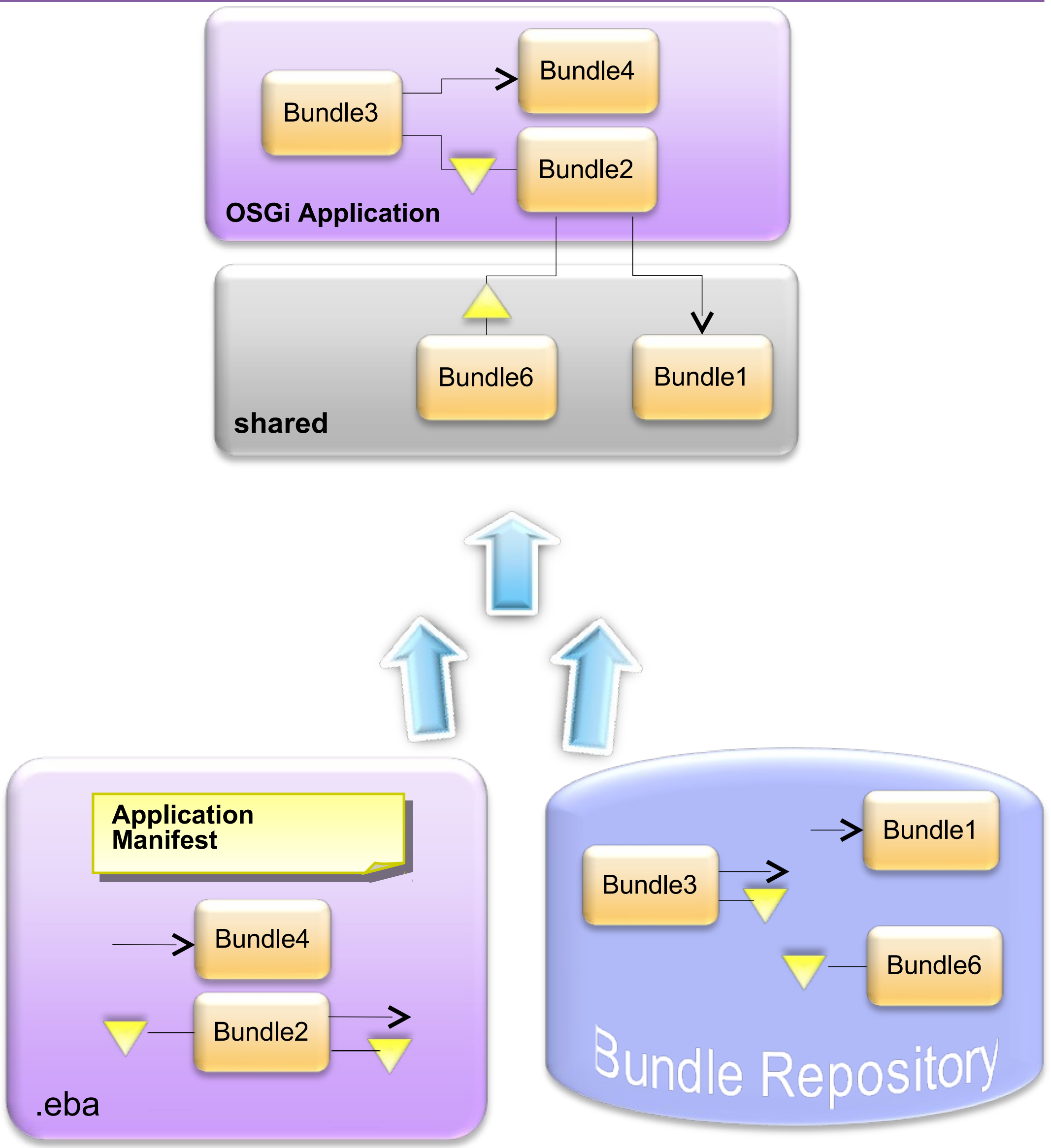
- Single source for each binary
- Side-by-side version execution
- No accidental binding to server libraries
- Can share common services

- In-place update of running applications
- Dynamic extension of running applications

- Isolated key application content
- Prevents accidental sharing
- Enables per-application configuration

OSGi Application

- Isolated, cohesive collection of bundles
- Defined by application manifest
  - Configuration by exception
- Deployed as .eba archive (zip file)
- Provisioning resolves application against archive contents and configured bundle repositories
- Transitive dependencies shared between applications



Enterprise OSGi : Also 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

- Modular Programming benefits in a JEE App!

Enterprise OSGi in WebSphere Application Server

OSGi in WebSphere Application Server

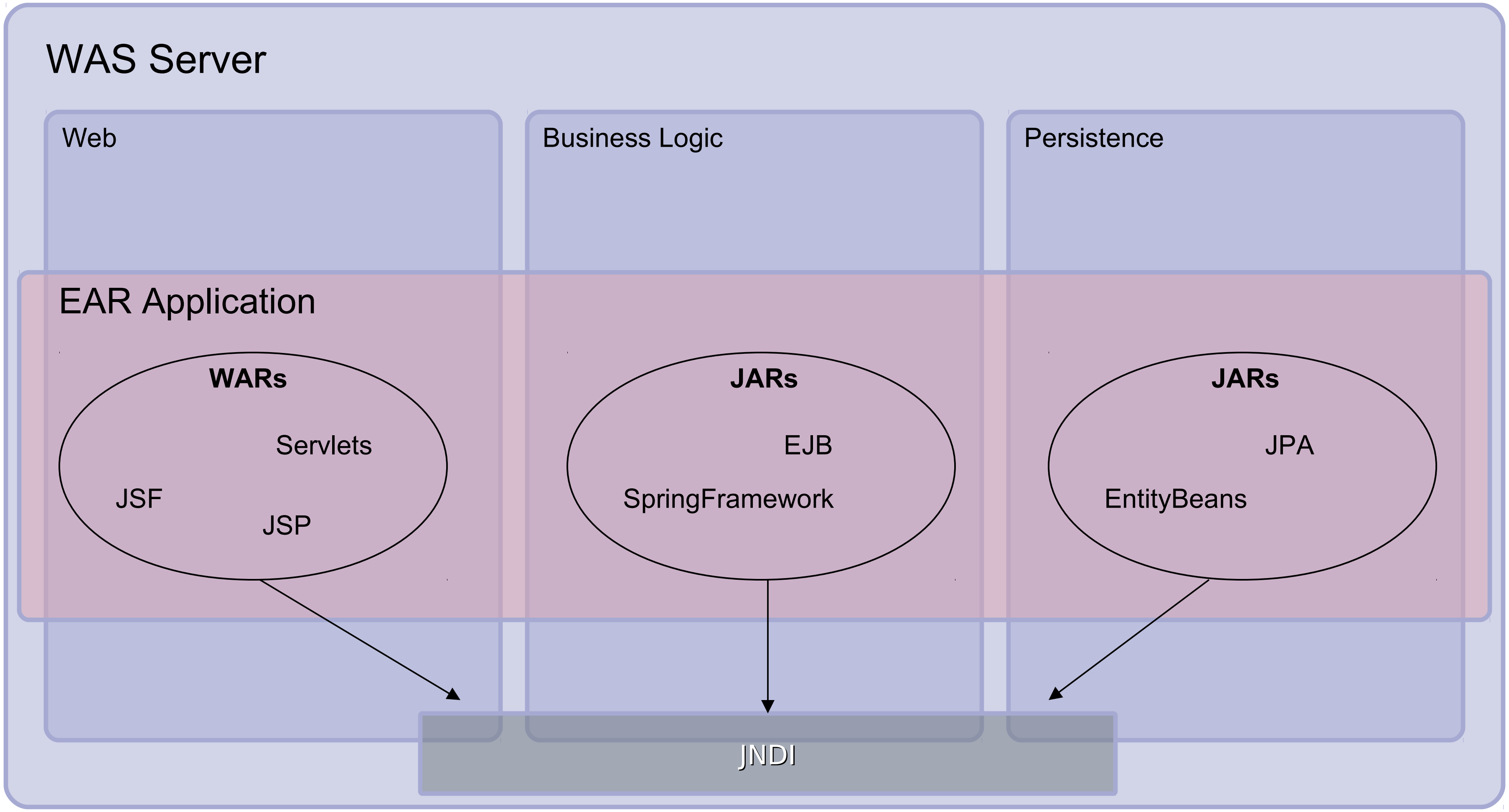
- **WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0**
  - Available since May 2010
    - More downloads in a shorter period of time than any previous WAS v7 feature pack

- **OSGi Feature enabled in core product for WebSphere Application Server v8.**

- **WebSphere Application Server v8.5 Beta 1 & 2**
    - EJB Support

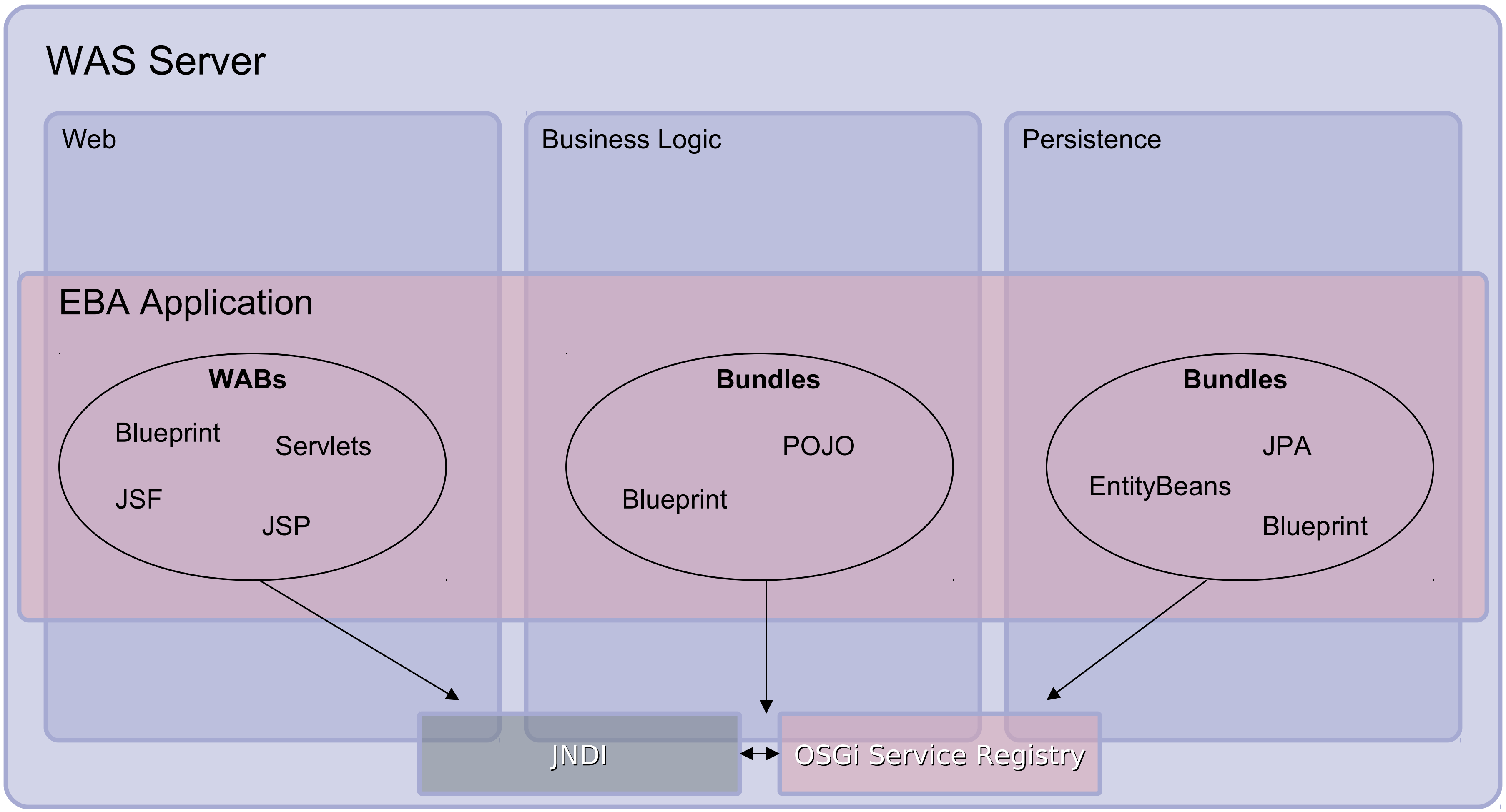
OSGi in the Application Server

- JEE Overview

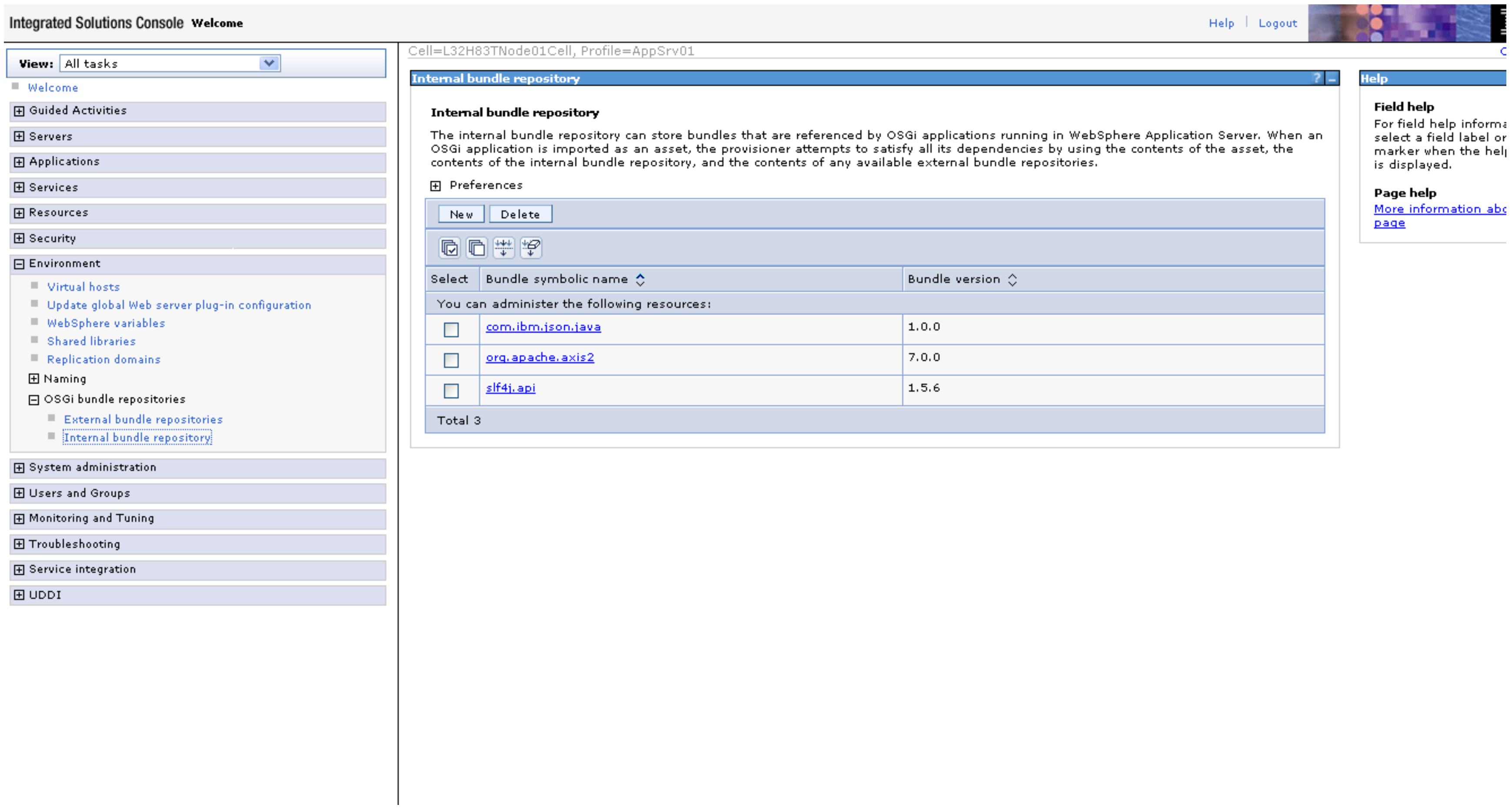


OSGi in the Application Server

- OSGi with JEE



Bundle Repository Config in WAS



Cell=L32H83TNode01Cell, Profile=AppSrv01

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

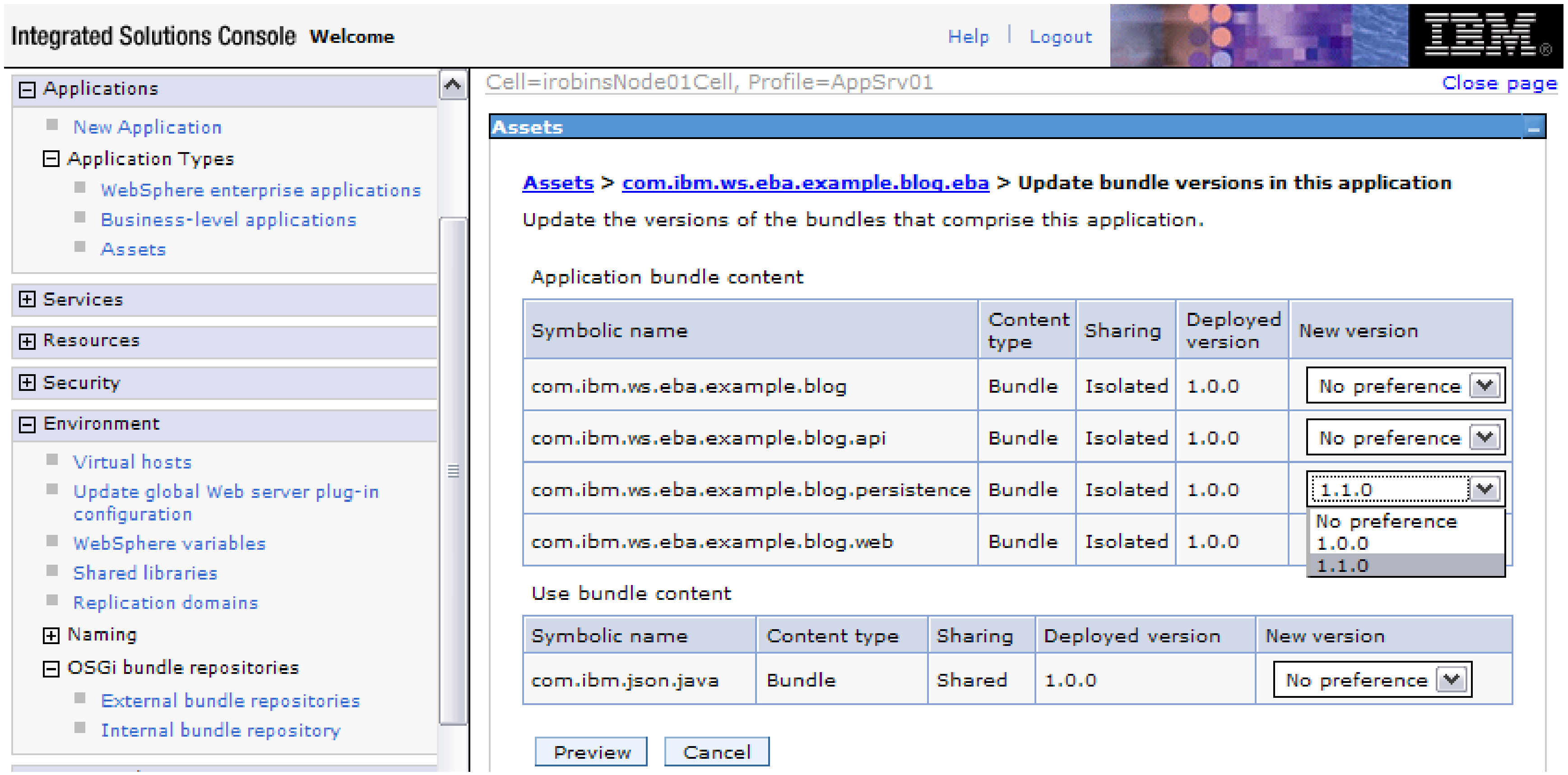
New | Delete

Select	Bundle symbolic name	Bundle version
You can administer the following resources:		
☐	com.ibm.json.java	1.0.0
☐	org.apache.axis2	7.0.0
☐	slf4j.api	1.5.6
Total 3		

Field help
For field help informa
select a field label or
marker when the hel
is displayed.

Page help
More information abo
page

Application-centric Bundle Management



Cell=irobinsNode01Cell, Profile=AppSrv01

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 (No preference / 1.0.0 / [✓ 1.1.0])
com.ibm.ws.eba.example.blog.web	Bundle	Isolated	1.0.0	

Use bundle content

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

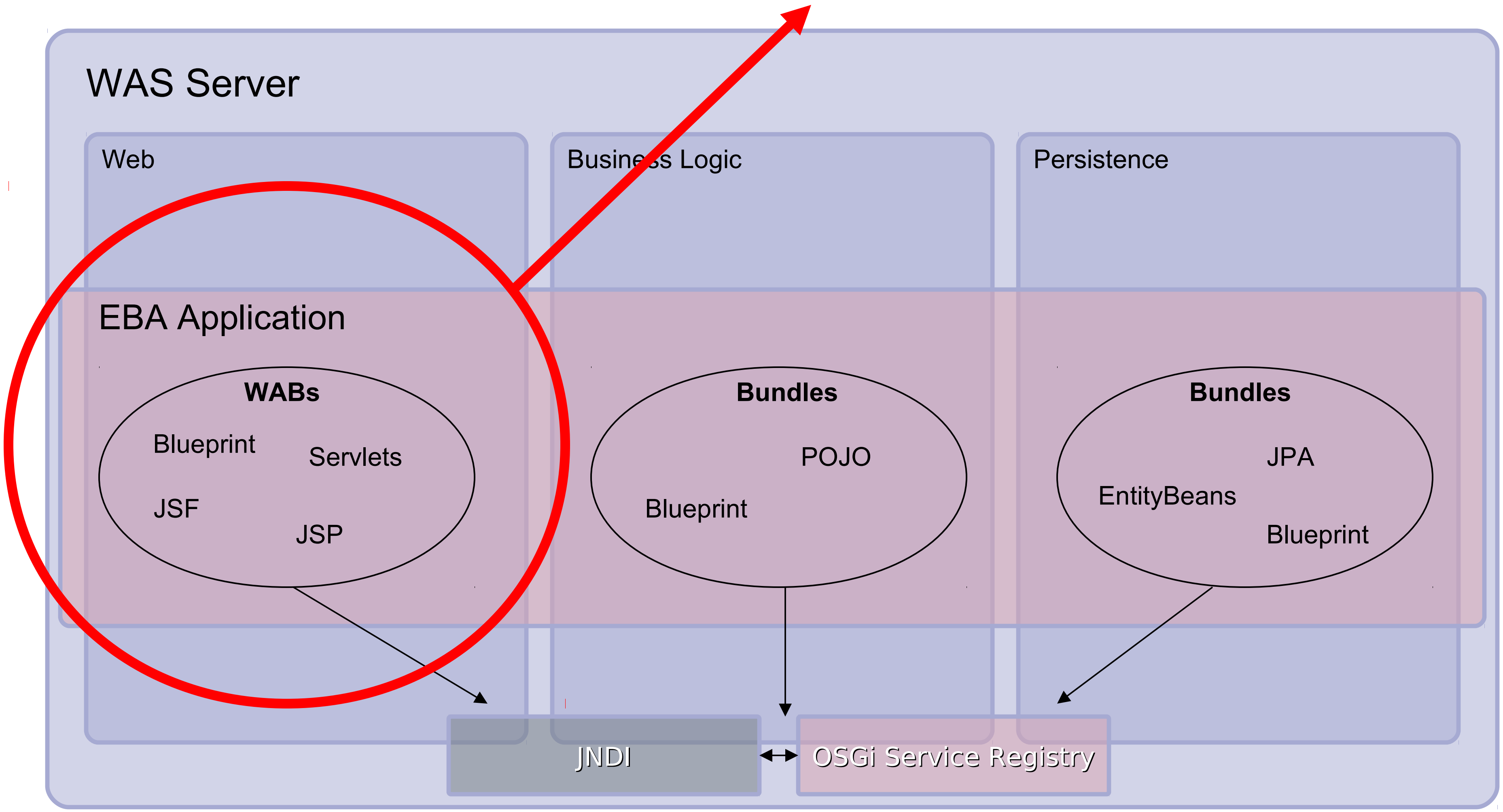
OSGi Service Registry and JNDI

- 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



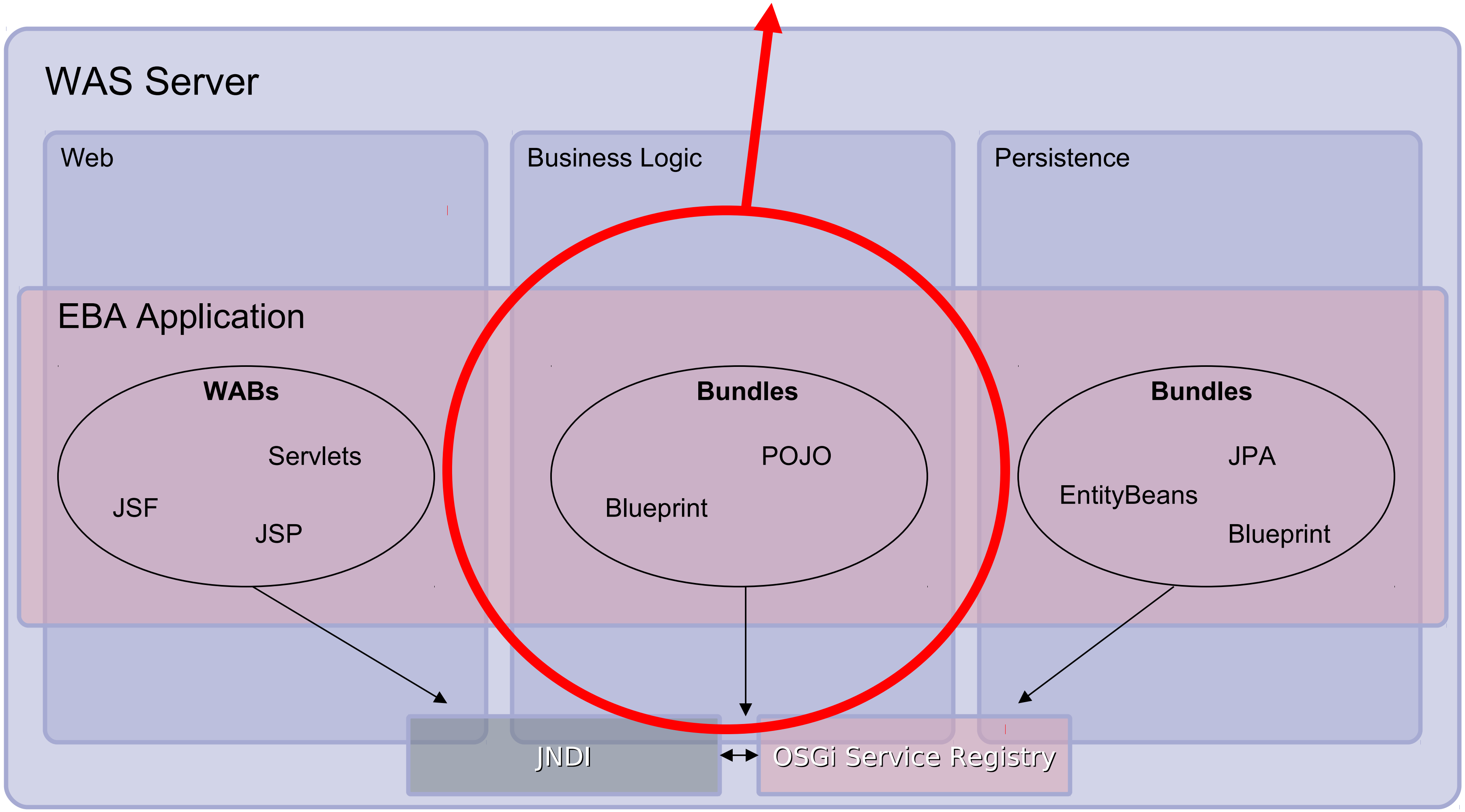
Web Application Bundles (WABs)

In-Built WAR to WAB convertor.

No Java code changes; WAR modules -> WAB bundles

OSGi in the Application Server

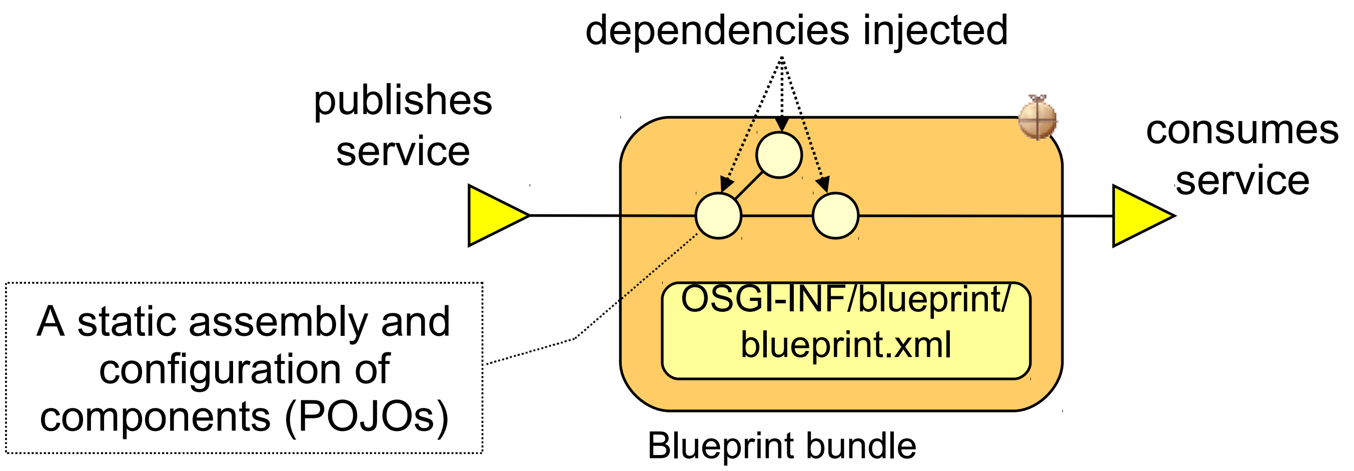
- Lets look closer at… Business Logic



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

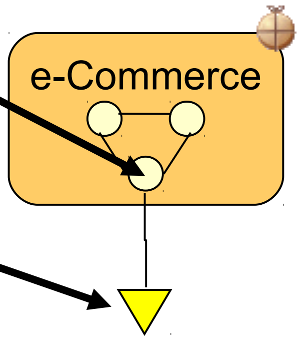

dependencies injected
publishes service
consumes service
A static assembly and configuration of components (POJOs)
OSGI-INF/blueprint/ blueprint.xml
Blueprint bundle

- *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>
```



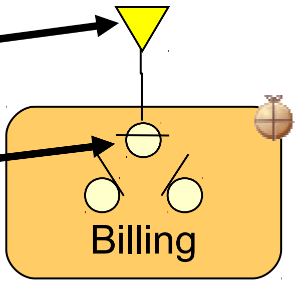
```
public class ShopImpl {

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

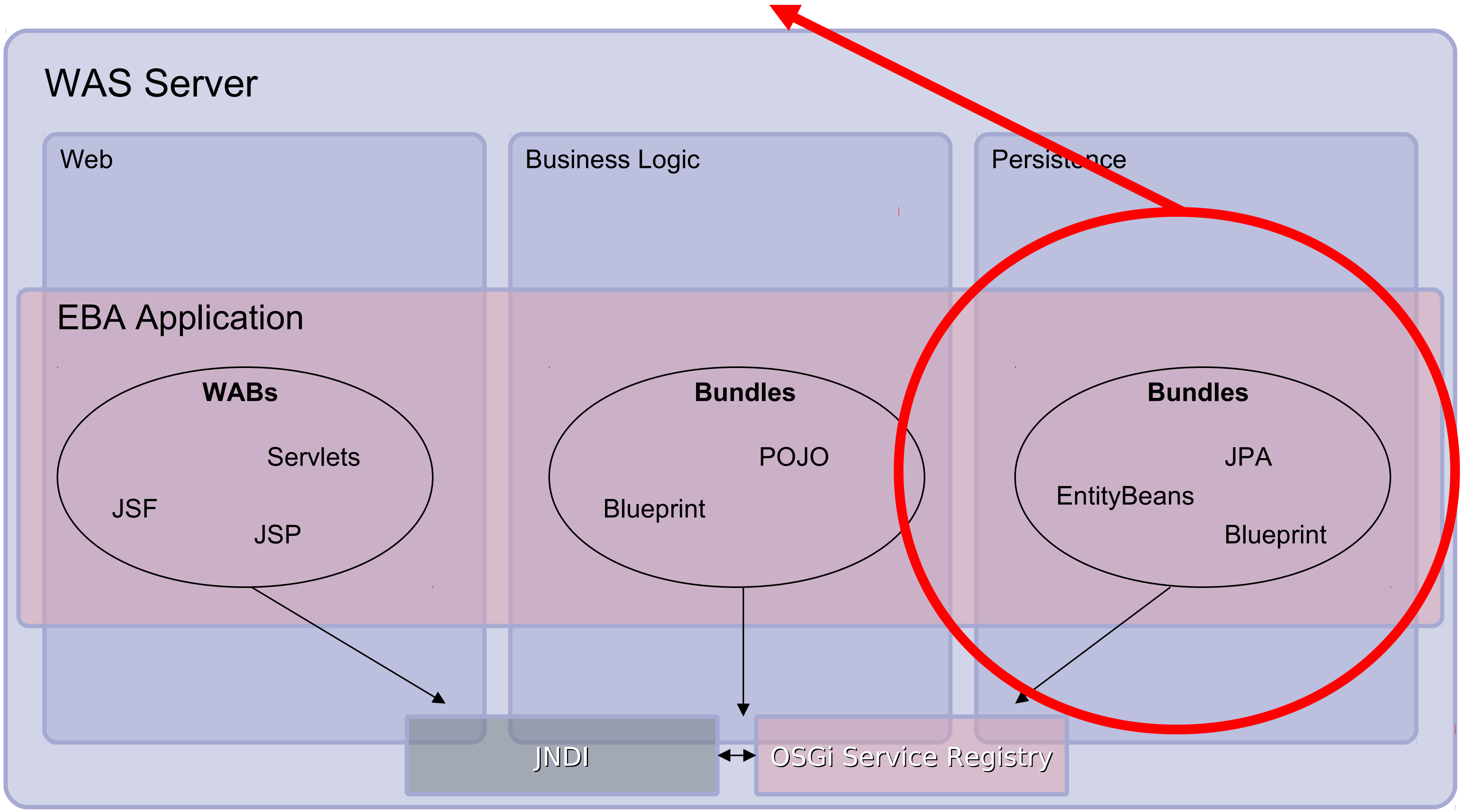
```
<blueprint>
  <service ref="service" interface =
                  "org.example.bill.BillingService" />
  <bean id="service" scope="prototype"
      class="org.example.bill.impl.BillingServiceImpl" />
</blueprint>
```



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

OSGi in the Application Server

- Lets look closer at… Persistence Layer

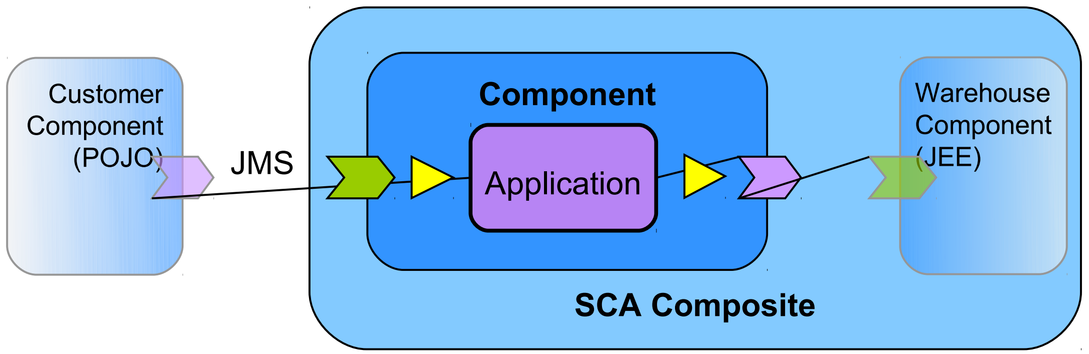


Blueprint Persistence and Transactions

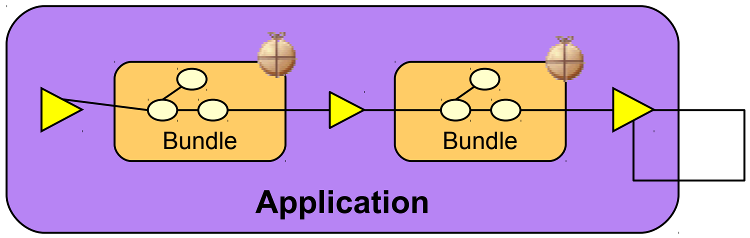
- OpenJPA is default persistence provider in WebSphere
- Container managed JPA support integrated into Blueprint container:
  - @PersistenceUnit or @PersistenceContext (managed)
  - or <jpa:unit>, <jpa:context> bean property injection
  - Familiar development experience for JPA developers
  - Load-time 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>
```

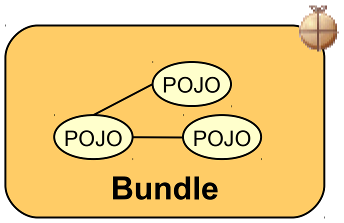
Further OSGi Support : SCA Integration



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



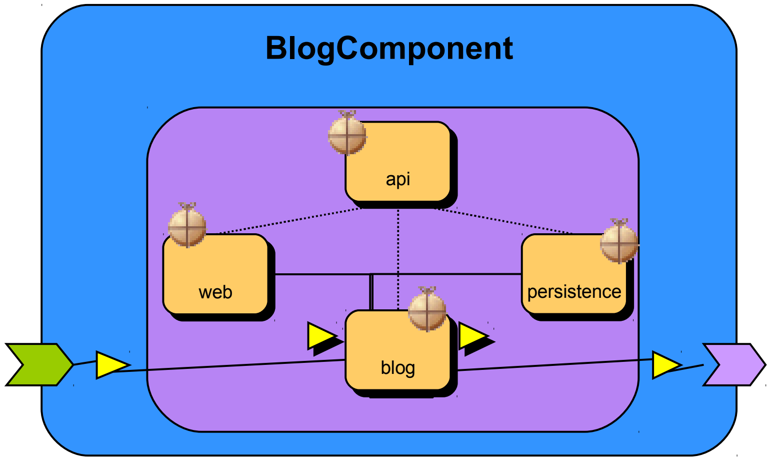
OSGi Bundles assembled in an **OSGi Application** and integrated through services in the OSGi service registry



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

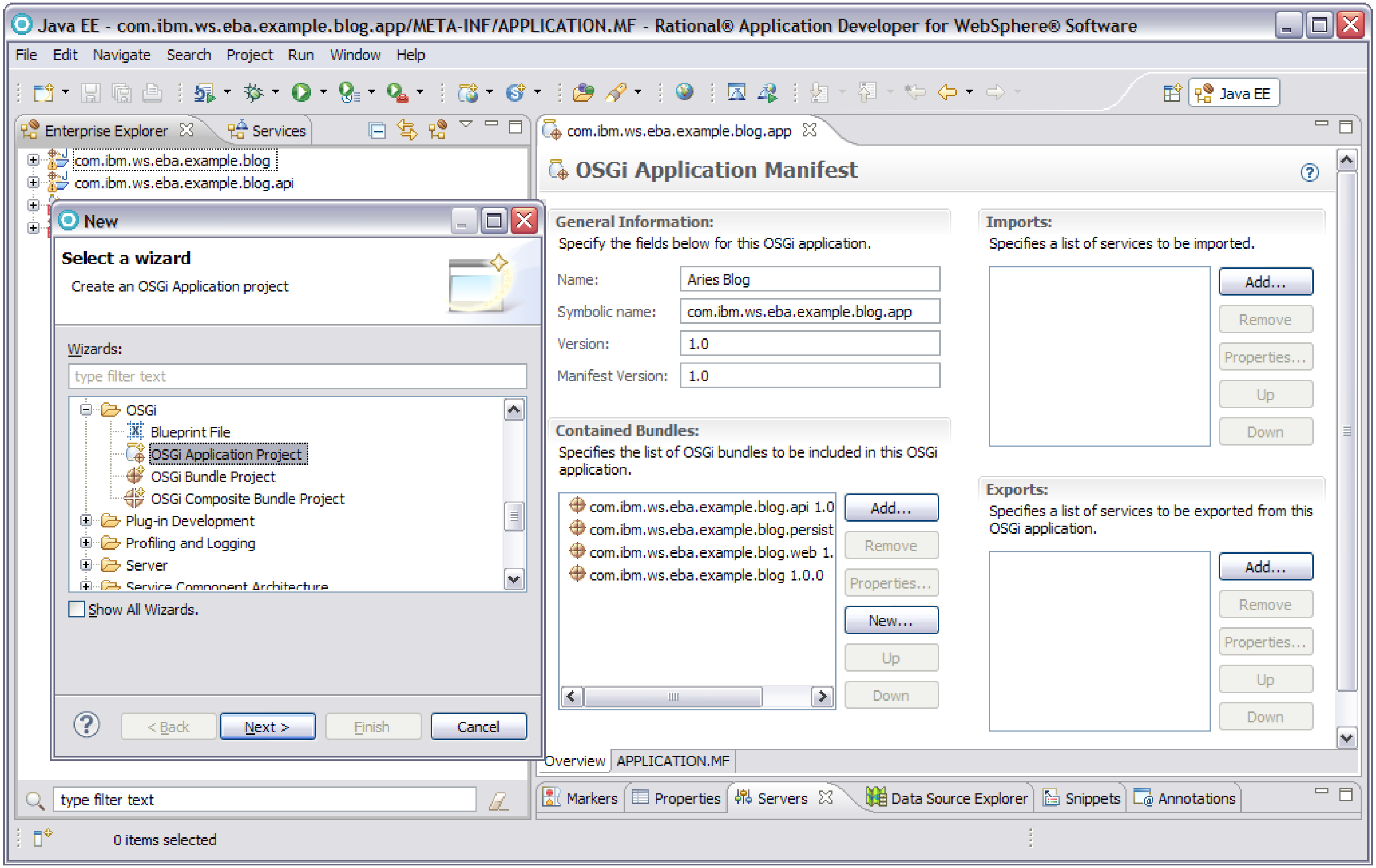
SCA integration: implementation.osgiapp

```
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



Java EE - com.ibm.ws.eba.example.blog.app/META-INF/APPLICATION.MF - Rational® Application Developer for WebSphere® Software

File Edit Navigate Search Project Run Window Help
Java EE
Enterprise Explorer
Services
com.ibm.ws.eba.example.blog
com.ibm.ws.eba.example.blog.api

New

Select a wizard

Create an OSGi Application project

Wizards:

type filter text

- OSGi
  - Blueprint File
  - OSGi Application Project
  - OSGi Bundle Project
  - OSGi Composite Bundle Project
- Plug-in Development
- Profiling and Logging
- Server
- Service Component Architecture

Show All Wizards.

com.ibm.ws.eba.example.blog.app

OSGi Application Manifest

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

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

Imports:

Specifies a list of services to be imported.

Exports:

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

0 items selected

What’s new in the WAS8.5 Betas?

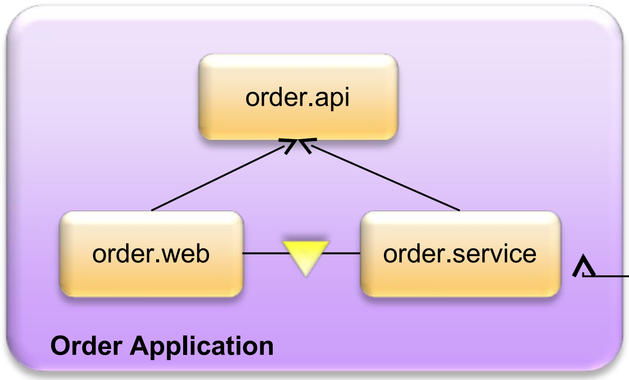
EJB Support!

- EJB Jar + OSGi Metadata = EJB Bundle
- **Export-EJB:** Opt-in header for EJB Bundles
  - Existence: process bundle for EJBs
  - Absence: do not process bundle for EJBs, even if it contains them
- Header value governs registration of EJBs as OSGi Services
  - Excludes Stateful beans
  - Best practice: only export EJBs to be shared outside bundle

Example	Meaning
Export-EJB:	Process all EJBs
Export-EJB: BlogBiz, BlogPersistence	Process all EJBs, register BlogBiz and BlogPersistence as Services if they exist
Export-EJB: NONE	RESERVED
Export-EJB: ALL	Process all EJBs and register all supported types as Services

- EJBs run in the same WAS EJB Container
- Uses OSGi for Classloading and Life-cycle

Example: Order Application using EJBs



- EJB Bundle included like any other bundle.
- Use of EJB is an implementation detail.

APPLICATION.MF

```
Application-Name: Order processing application
Application-SymbolicName: com.acme.order.app
Application-ManifestVersion: 1.0
Application-Version: 1.0.0
Manifest-Version: 1.0
Application-Content:
 com.acme.order.service;version="[1.0, 1.1)",
 com.acme.order.api;version="[1.0, 1.1)",
 com.acme.order.web;version="[1.0, 1.1)"
```

Example: Order EJB Bundle

- OrderEJB implemented as normal local EJB
  - EJB 3.x style only (2.x not supported)
- Best practices:
  - Put interfaces and EJB implementations in separate bundles and separate packages
  - Annotate @Local and/or @Remote on the EJB implementation classes*

```
/**
 * Local stateless session Order EJB
 */
@Local
@Stateless
public class OrderEJB implements Order {

    ...

    @Override
    public String process() {
        ...
        return "Order Processed";
    }
}
```

- OrderEJB named in Export-EJB header so registered as an OSGi service under the Order interface
- Imports the packages it needs:
  - Order API
  - EJB

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Acme order processing service
Bundle-SymbolicName: com.acme.order.service
Bundle-Version: 1.0.0
Export-EJB: OrderEJB
Import-Package:
 com.acme.order.api;version="[1.0.0,1.1.0)",
 javax.ejb;version="3.1"
```

*annotations not processed outside the EJB bundle

Example: Order Web Front-end

- EJB 3.1 allows EJBs in a WAR (supported) but best practices suggest separating business logic from presentation logic
- Order EJB access:
  - As EJB via injection or EJB JNDI lookup

```
/**
 * Order EJB to be injected
 */
@EJB
private Order orderService = null;
```

```
orderService =
   (Order) new InitialContext().lookup("java:comp/env/com.acme.order.View/orderService");
```

  - As an OSGi Service (helps decouple life-cycle for dynamic update)

```
orderService =
   (Order) new InitialContext().lookup("osgi:service/com.acme.order.api.Order");
```

```
orderService =
   (Order) new InitialContext().lookup("blueprint:comp/orderService");
```

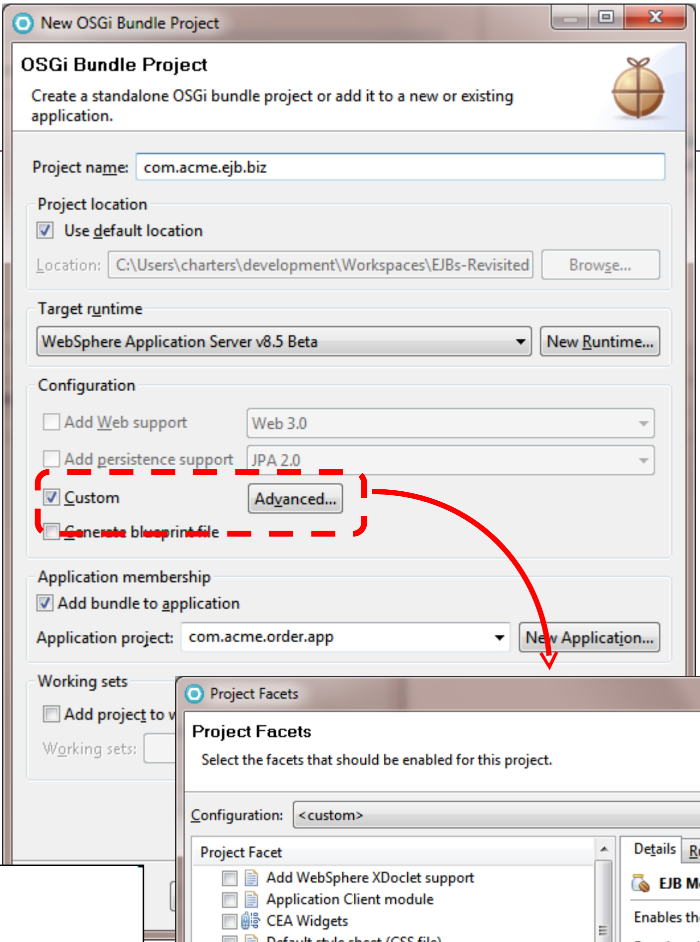
```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" >
  <reference id="orderService" interface="com.acme.order.api.Order"/>
</blueprint>
```

EJB/Service Look-up Comparison

Style	Re-acquirable	Re-injectable	Damped	Details
@EJB	No	No	No	Use when simple, static injection is all that's required
java:comp/env	Yes	No	No	Use for existing java:comp lookups
osgi:service	Yes	Yes	No	Use when dynamic lifecycle required, but no damping
blueprint:comp	Yes	Yes	Yes	Use when dynamic lifecycle and damping are required. Requires additional use of Blueprint

Tools: RAD 8.5 beta

- Initial EJB Bundle Project support
- OSGi Bundle Project + EJB Module Facet
- Familiar layout for EJB developers
- Uses PDE for build and validation


New OSGi Bundle Project

OSGi Bundle Project

Create a standalone OSGi bundle project or add it to a new or existing application.

Project name: com.acme.ejb.biz

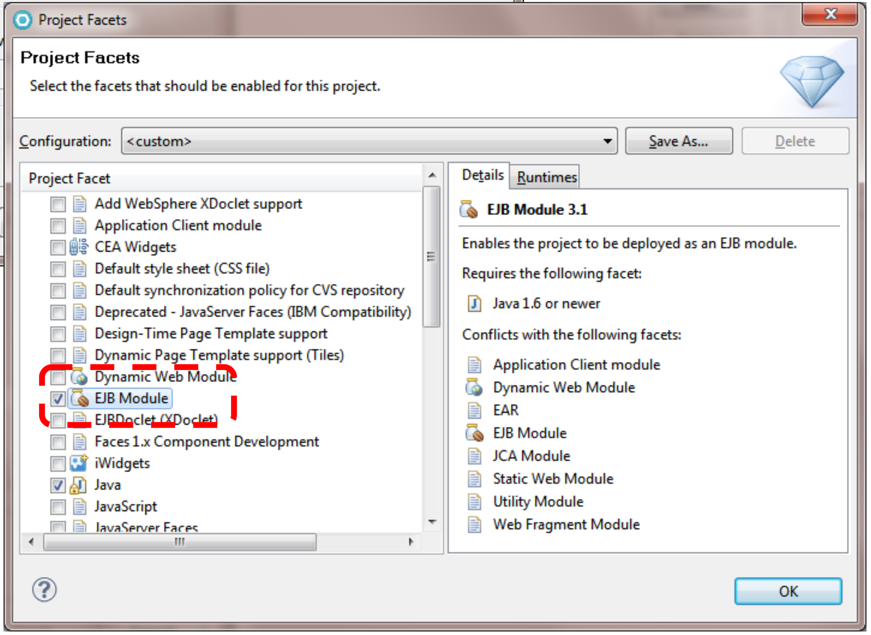
Project location
- [✓] Use default location
- Location: C:\Users\charters\development\Workspaces\EJBs-Revisited — Browse...

Target runtime
- WebSphere Application Server v8.5 Beta — New Runtime...

Configuration
- [ ] Add Web support — Web 3.0
- [ ] Add persistence support — JPA 2.0
- [✓] Custom — Advanced...
- [ ] Generate blueprint file

Application membership
- [✓] Add bundle to application
- Application project: com.acme.order.app — New Application...

Working sets
- [ ] Add project to w
- Working sets:


Project Facets

Project Facets

Select the facets that should be enabled for this project.

Configuration: <custom> Save As... Delete

Project Facet
- [ ] Add WebSphere XDoclet support
- [ ] Application Client module
- [ ] CEA Widgets
- [ ] Default style sheet (CSS file)
- [ ] Default synchronization policy for CVS repository
- [ ] Deprecated - JavaServer Faces (IBM Compatibility)
- [ ] Design-Time Page Template support
- [ ] Dynamic Page Template support (Tiles)
- [ ] Dynamic Web Module
- [✓] EJB Module
- [ ] EJBDoclet (XDoclet)
- [ ] Faces 1.x Component Development
- [ ] iWidgets
- [✓] Java
- [ ] JavaScript
- [ ] JavaServer Faces

Details | Runtimes

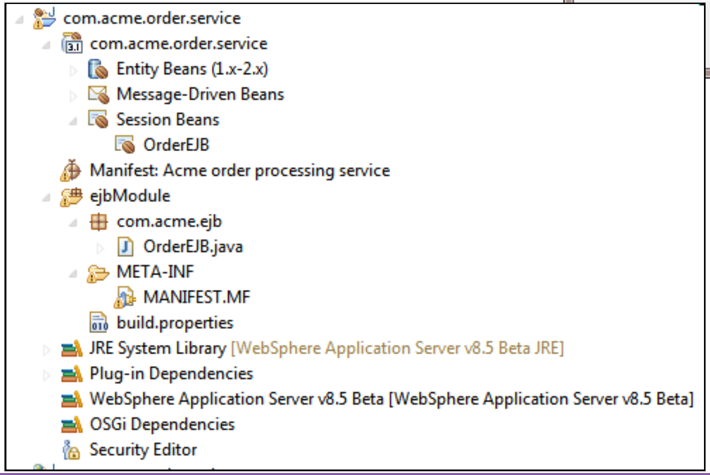
EJB Module 3.1

Enables the project to be deployed as an EJB module.

Requires the following facet:
- Java 1.6 or newer

Conflicts with the following facets:
- Application Client module
- Dynamic Web Module
- EAR
- EJB Module
- JCA Module
- Static Web Module
- Utility Module
- Web Fragment Module

OK

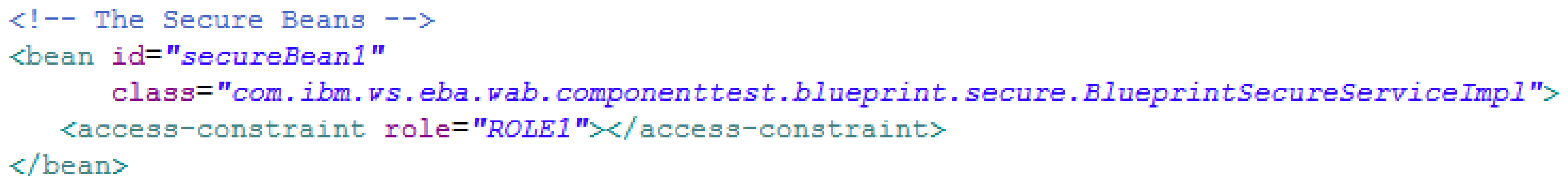

```
com.acme.order.service
  com.acme.order.service
    Entity Beans (1.x-2.x)
    Message-Driven Beans
    Session Beans
      OrderEJB
  Manifest: Acme order processing service
  ejbModule
    com.acme.ejb
      OrderEJB.java
    META-INF
      MANIFEST.MF
    build.properties
  JRE System Library [WebSphere Application Server v8.5 Beta JRE]
  Plug-in Dependencies
  WebSphere Application Server v8.5 Beta [WebSphere Application Server v8.5 Beta]
  OSGi Dependencies
  Security Editor
```

EJB Related Topics

- In-place Update
  - EJB Bundles can participate in in-place updates. Use of Blueprint references and OSGi Service registry for EJB access recommended to take advantage of service damping and minimise update impact.
- Application Extension
  - EJB Bundles can be included as extension content. Use of Blueprint references and OSGi Service registry for EJB access recommended to pick up extensions without application restart and to minimise impact of extensions on running application.
- Remoting
  - @Remote EJBs exported to OSGi service registry treated like any other remote OSGi service and can be exported/bound using SCA.
  - EJBs can look up SCA imported remote OSGi services through JNDI (using osgi:service or blueprint:comp schemes).
- Transactions & Security
  - Work as for non-OSGi Java EE EJBs

WAS8.5 Beta New Features : Blueprint Security

- Role based authorization
- Bean Security in XML



```
<!-- The Secure Beans -->
<bean id="secureBean1"
      class="com.ibm.ws.eba.wab.componenttest.blueprint.secure.BlueprintSecureServiceImpl">
   <access-constraint role="ROLE1"></access-constraint>
</bean>
```

- Roles mapped to users at deploy time

WAS 8.5 Beta New Features : Java 7 Support

- Full Java 7 Support in WAS v8.5 Beta
- Includes OSGi support

Migrating to OSGi : Hints and Tips

Migration

- Approach 1 : The Big Bundle
  - Put everything into one bundle
  - Get it working
  - Pick out libraries into individual bundles
  - Keep it working!
  - Pick out related content into separate bundles.
  - Keep it working!
  - Turn dependencies into service consumer/provider pairs
  - Keep it working!

Migration

- Approach 2 : Service First Approach
  - Identify logical relationships in the system
  - Expose as services (POJO SR)
  - Use the relationships to group content into similar JARs
  - Turn into bundles (BND tools)

Migration

- Migration Tools :
  - BND Tools
  - Maven Plugin
  - POJO SR

- Migration experience still in infancy

- Articles appearing :

Conversion of Apache Tuscany to OSGi
http://www.slideshare.net/luckbr1975/tuscany-applying-o-s-gi-after-the-fact

Converting Large apps to OSGi
http://download.oracle.com/javaone/javaone2008-ee.zip

Summary

- Modular Programming is not supported fully in :
  - JAVA (SE)
  - JAVA (EE)

- The Solution for JAVA (SE) applications… OSGi

- The Solution for JAVA (EE) applications… Enterprise OSGi

- Fully implemented in WASv7 and v8

- EJB Support in the new v8.5 Beta

- Hints and Tips for Migration to OSGi

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