Using OSGi in Modular WebSphere Applications

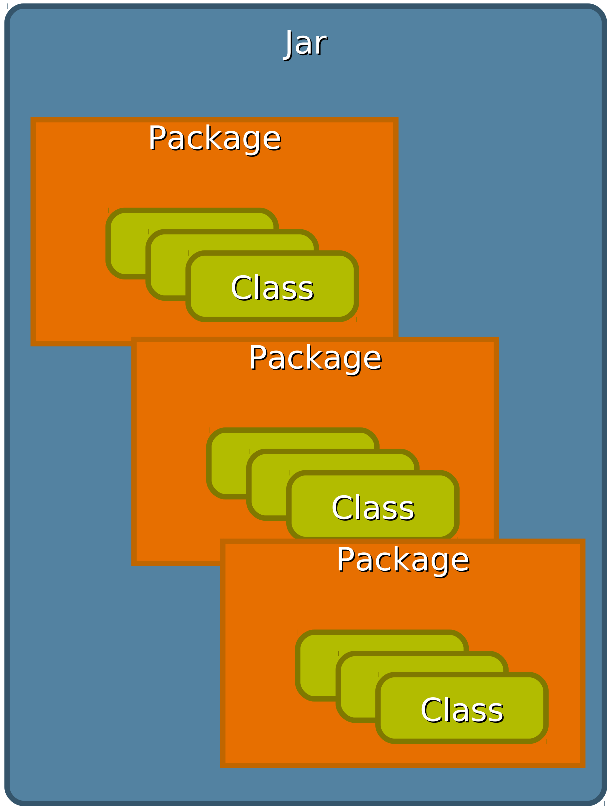
Hannah Ramlee, IBM Software Engineer

Cutting to the chase…

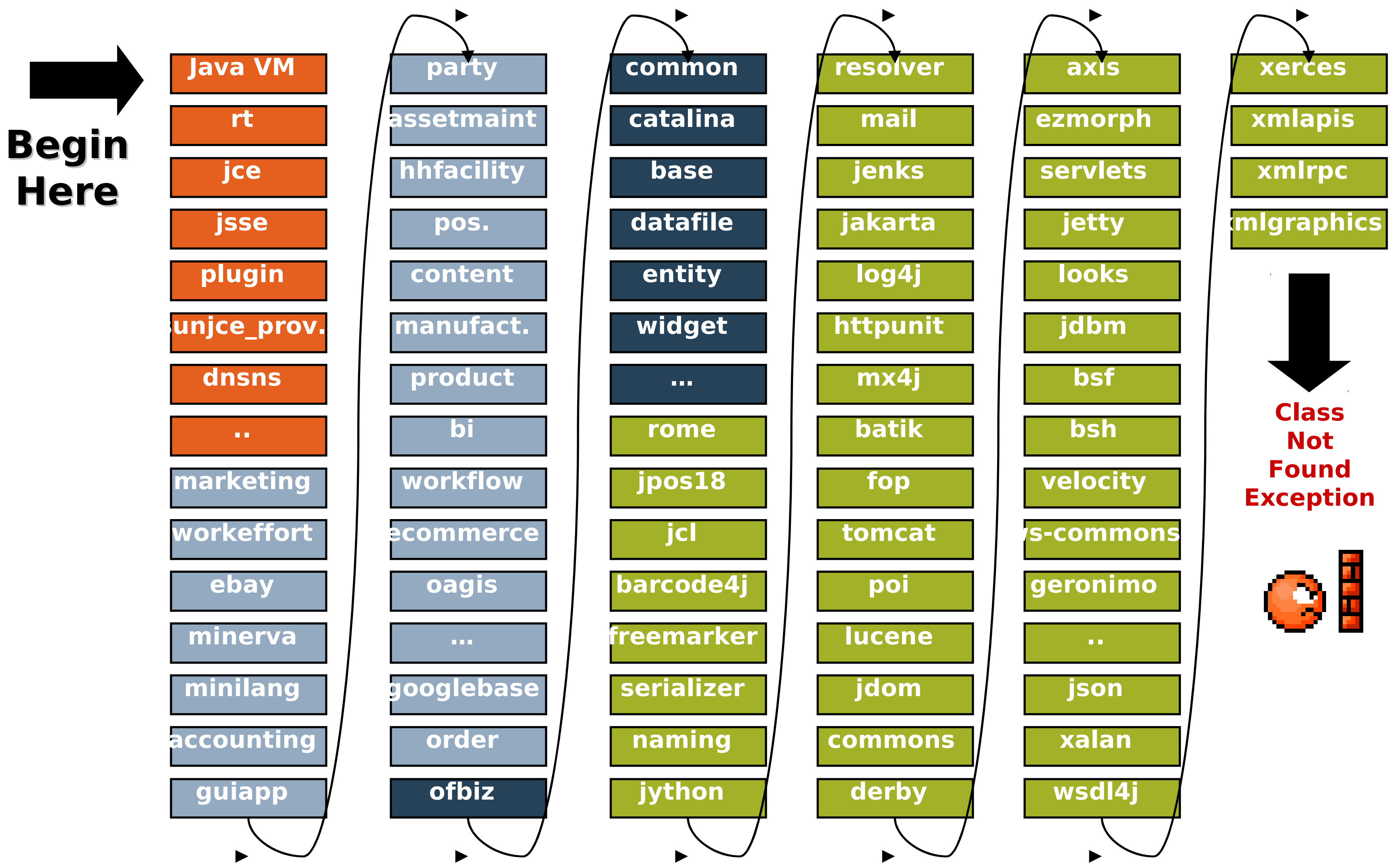
- What: The WAS V7 Feature Pack for OSGi Applications and JPA 2.0
- Why: Common Problems You've All Seen
- How: OSGi and Java EE together
  - In Standards
  - In Open Source
  - In the WebSphere OSGi Application Feature Pack

Modularization in Java – Problems with Jars

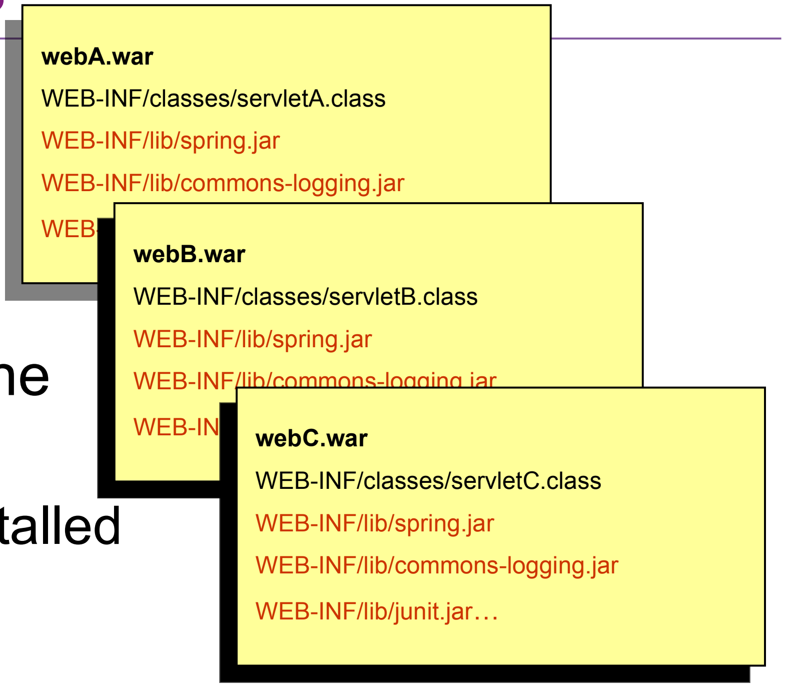
- Java Platform Modularity
  - Classes encapsulate data
  - Packages contain classes
  - Jars contain packages
- Class visibility:
  - private, package private, protected, public
- No “jar scoped” access modifiers.
- No means for a jar to declare its dependencies.
- No versioning.
- Jars have no modularization characteristics
  - At runtime there is just a collection of classes on a global classpath



Problems with Global Java ClassPath



Problems with EARs/WARs



webA.war
WEB-INF/classes/servletA.class
WEB-INF/lib/spring.jar
WEB-INF/lib/commons-logging.jar
WEB

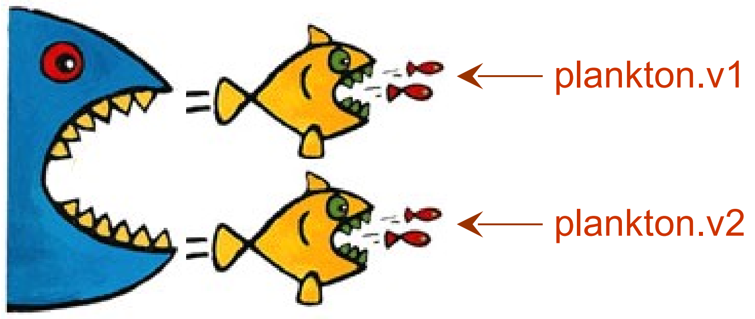
webB.war
WEB-INF/classes/servletB.class
WEB-INF/lib/spring.jar
WEB-INF/lib/commons-logging.jar
WEB-IN

webC.war
WEB-INF/classes/servletC.class
WEB-INF/lib/spring.jar
WEB-INF/lib/commons-logging.jar
WEB-INF/lib/junit.jar…

Enterprise Apps have isolated classpaths but…

- *Across apps* - each archive typically contains all the libraries required by the application
  - Common libraries/frameworks get installed with each application
  - Multiple copies of libraries in memory

- *Within apps* - 3rd party libraries consume other 3rd party libraries leading to version conflicts

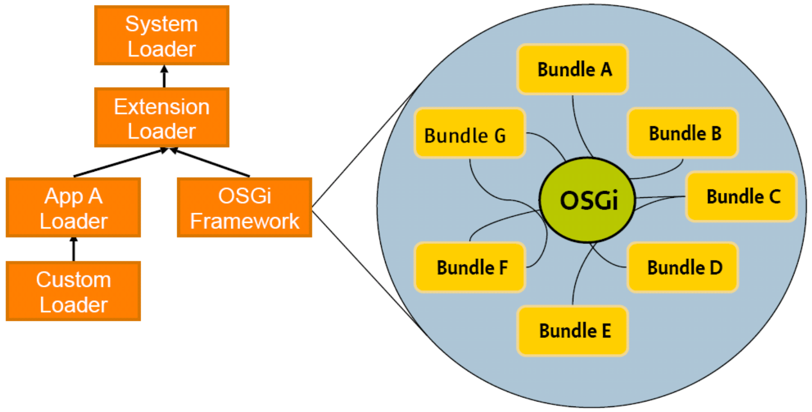


OSGi Bundles and Class Loading

- OSGi Bundle – A jar containing:
  - Classes and resources.
  - OSGi Bundle manifest.
- What's in the manifest:
  - Bundle-Version: Multiple versions of bundles can live concurrently.
  - Import-Package: What packages from other bundles does this bundle depend upon?
  - Export-Package: What packages from this bundle are visible and reusable outside of the bundle?

```
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"
```

- 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 Enterprise Specification

- Released 22 March 2010
  - The product of the OSGi Enterprise Expert Group (EEG)
- Brings Enterprise technologies and OSGi together
- Using existing Java SE/EE specifications:
  - JTA, JPA, JNDI, JMX, WebApps…
- Adds Spring-derived *Blueprint* component model and DI container

- **Java EE provides the core enterprise application programming model**
- **Deploying modules as OSGi bundles simplifies reuse between applications, provides versioning, encourages (and enforces) modular design and enables dynamic module updates.**

Enterprise OSGi in Open Source

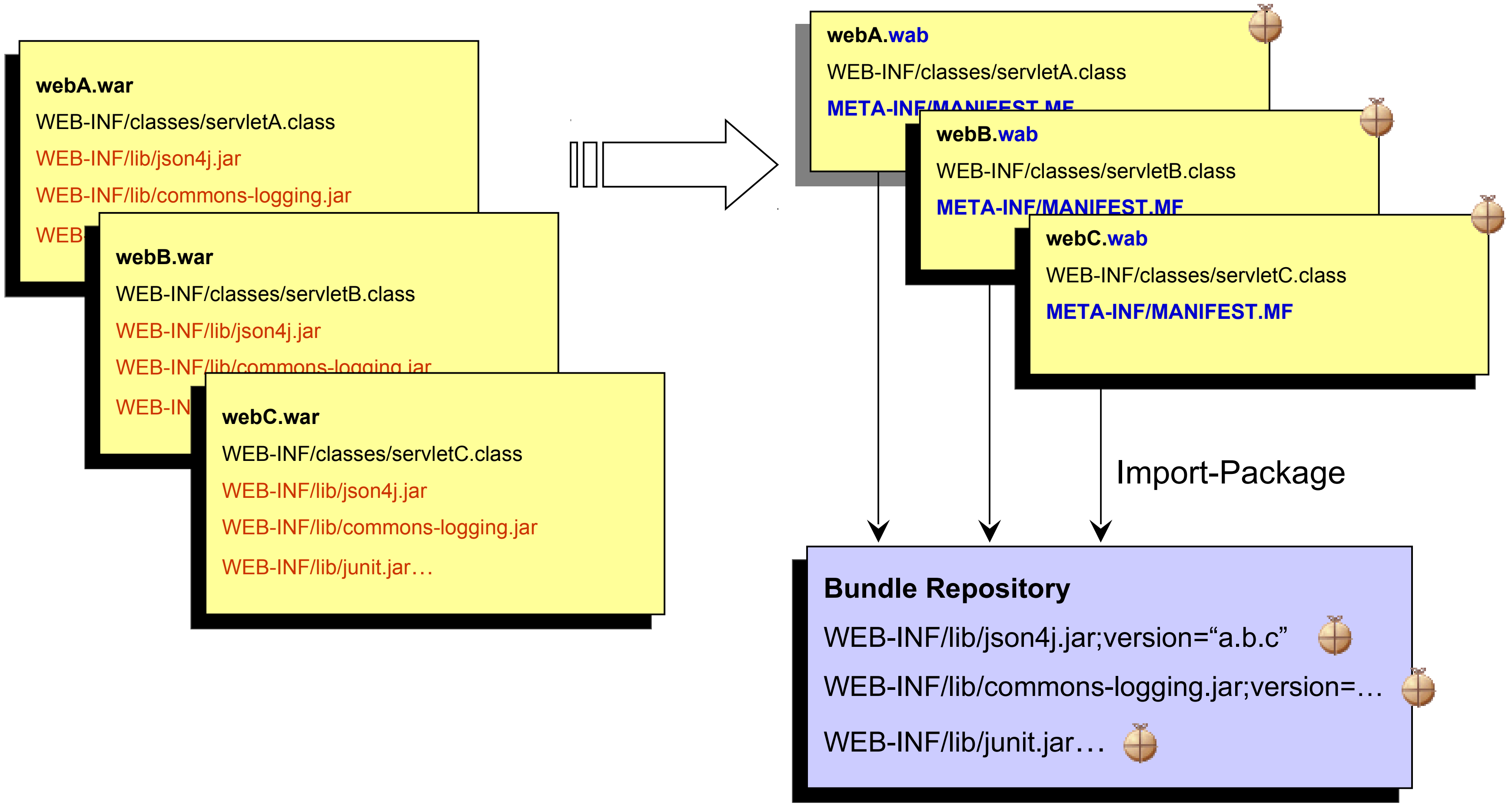
- Apache "Aries" created as a new Apache incubator project in Sep 2009:
  - to provide enterprise OSGi spec implementations http://incubator.apache.org/aries/
  - to provide an environment to collaborate and experiment with new technologies to inform further EEG standardization.
    - In particular the programming model aspects of OSGi applications in an enterprise environment such as the Blueprint container and multi-bundle composites.
  - to build a broad development community to encourage implementation and adoption of EEG specs
- Aries componentry supporting an enterprise OSGi programming model are being integrated into both Geronimo and WAS.
  - As well as Apache Felix Karaf, JBossOSGi and others

Application exploitation of OSGi in WebSphere

- OSGi has been used internally in WAS since V6.1 and in Eclipse since R3.
- Application-level exploitation is introduced in the **WebSphere Application Server V7 Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0**
  - http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/
  - Generally available May 2010
- Early Program available since Nov 2009
  - https://www14.software.ibm.com/iwm/web/cc/earlyprograms/websphere/wasfposgiajp
  - More downloads in a shorter period of time than any previous WAS v7 FEP open beta
- Two installable features:
  - OSGi Application feature simplifies the development, assembly, and deploy of enterprise
  - JPA 2.0 feature introduces Java EE 6 JPA 2.0 enhancements to object-relational persiste data access and optimize performance

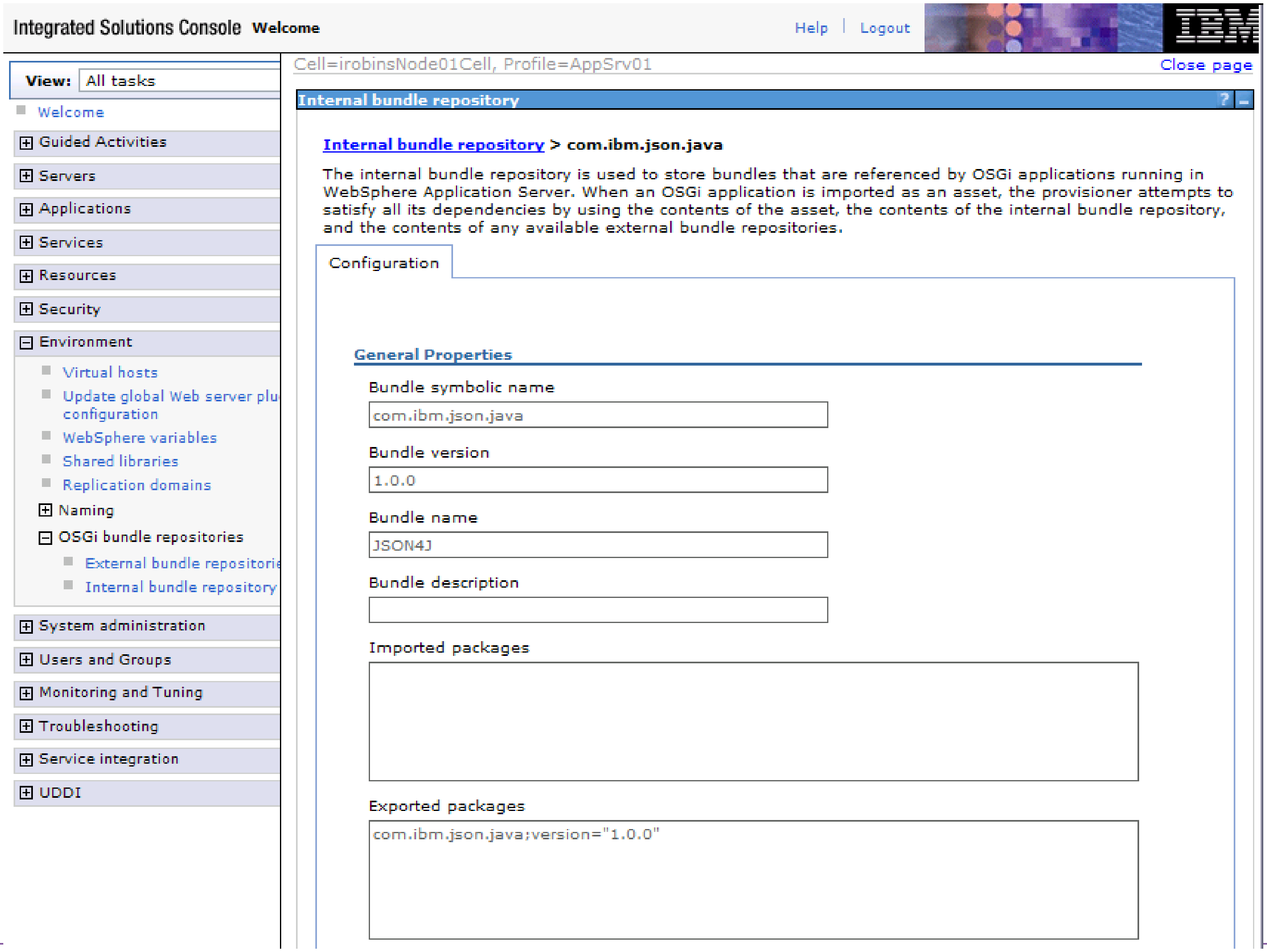
Getting Started: Bundlizing vanilla JEE

No Java code changes; war modules -> bundles
Common, bundles may be easily factored out of the WARs and used at specific versions



webA.war
WEB-INF/classes/servletA.class
WEB-INF/lib/json4j.jar
WEB-INF/lib/commons-logging.jar
WEB-
webB.war
WEB-INF/classes/servletB.class
WEB-INF/lib/json4j.jar
WEB-INF/lib/commons-logging.jar
WEB-IN
webC.war
WEB-INF/classes/servletC.class
WEB-INF/lib/json4j.jar
WEB-INF/lib/commons-logging.jar
WEB-INF/lib/junit.jar…
webA.wab
WEB-INF/classes/servletA.class
META-INF/MANIFEST.MF
webB.wab
WEB-INF/classes/servletB.class
META-INF/MANIFEST.MF
webC.wab
WEB-INF/classes/servletC.class
META-INF/MANIFEST.MF
Import-Package
Bundle Repository
WEB-INF/lib/json4j.jar;version="a.b.c"
WEB-INF/lib/commons-logging.jar;version=…
WEB-INF/lib/junit.jar…

New: Bundle Repository Config in WAS



Cell=irobinsNode01Cell, Profile=AppSrv01

Internal bundle repository

Internal bundle repository > com.ibm.json.java

The internal bundle repository is used to 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.

General Properties

Bundle symbolic name: com.ibm.json.java

Bundle version: 1.0.0

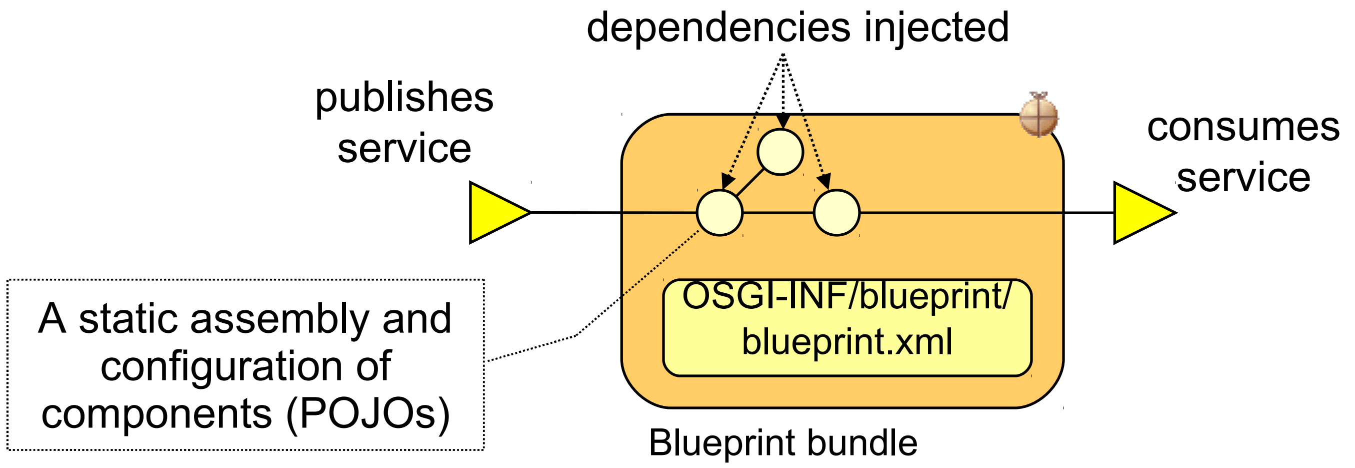
Bundle name: JSON4J

Bundle description:

Imported packages:

Exported packages: com.ibm.json.java;version="1.0.0"

Blueprint Components and Services

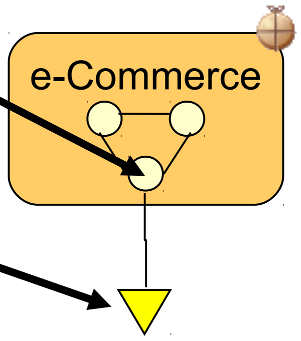

dependencies injected
publishes service
consumes service
A static assembly and configuration of components (POJOs)
OSGI-INF/blueprint/ blueprint.xml
Blueprint bundle

- Specifies a Dependency Injection container, standardizing established Spring conventions
- Configuration and dependencies declared in XML “module blueprint”, which is a standardization of Spring “application context” XML.
  - Extended for OSGi: publishes and consumes components as OSGi services
- Simplifies unit test outside either Java EE or OSGi r/t.
- **The Blueprint DI container is a part of the server runtime (compared to the Spring container which is part of the application.)**

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>
```



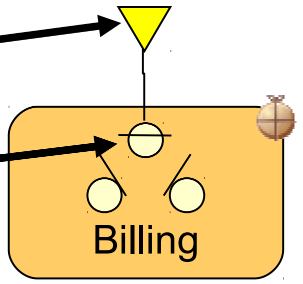
```
public class ShopImpl {

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);
}
```

-“prototype” scope indicates a new instance is created by the container for each use.
-“singleton” scope is the default.

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

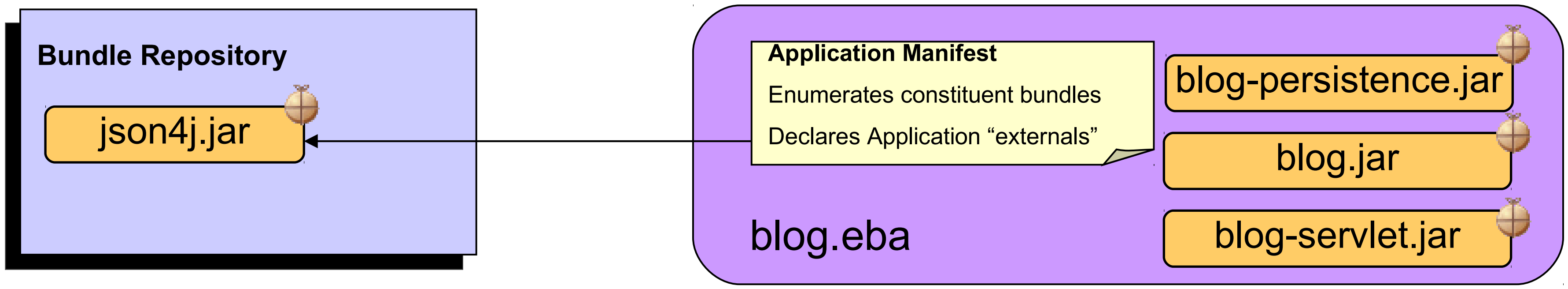
```
<blueprint>
  <bean id=”shop” class=”org.example.ecomm.ShopImpl”>
      <jpa:context property="em" unitname="myUnit"/>
      <tx:transaction method="*" value="Required"/>
  </bean>
</blueprint>
```

OSGi Service Registry and JNDI

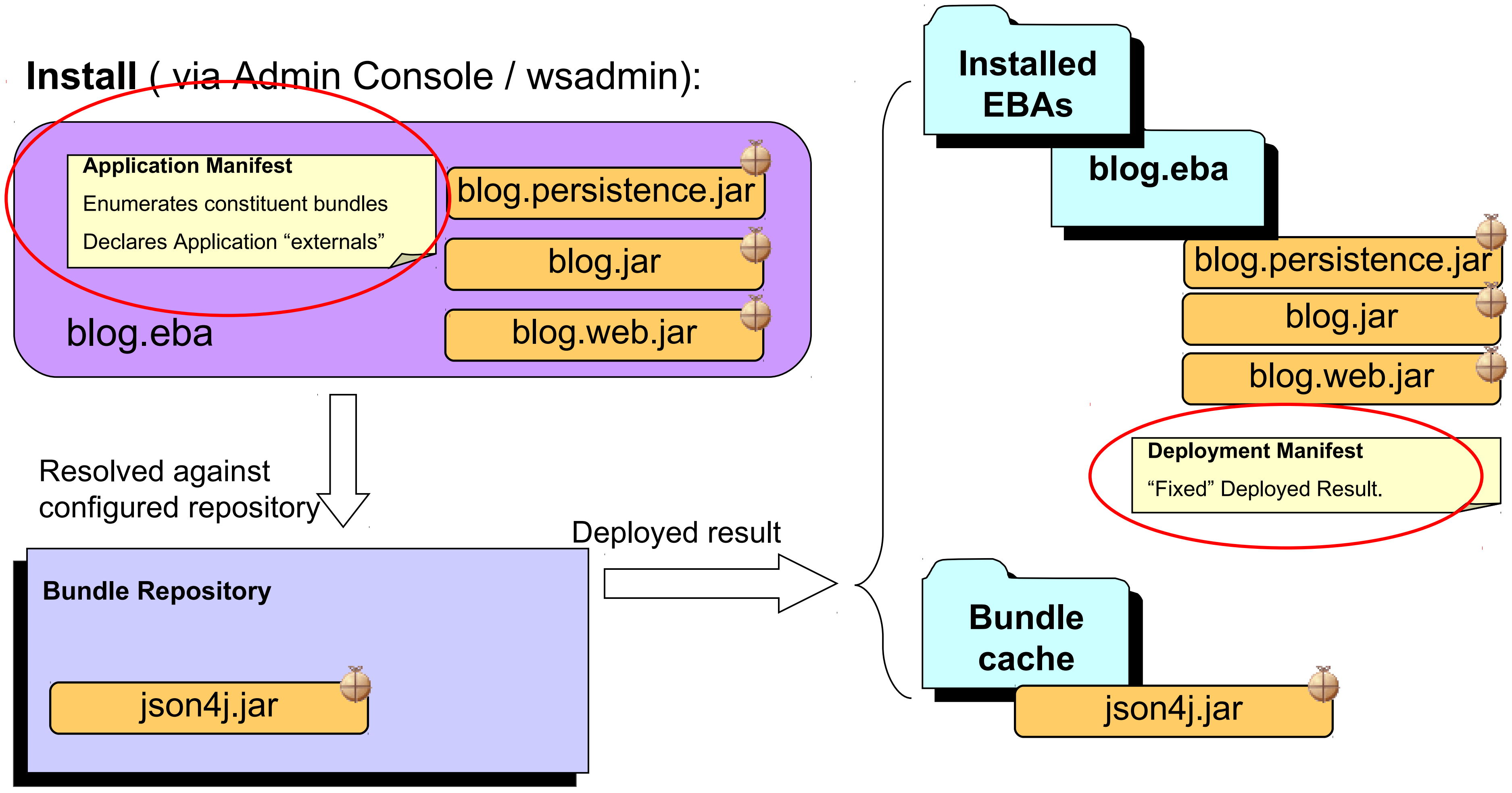
- OSGi services are published to and looked up from OSGi service registry.
  - From declarations in Blueprint XML
- Simplify integrating with existing JEE components:
  - OSGi Services registered in the OSGi Service Registry are also available in JNDI via the osgi:service URL scheme
  - Administered 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”

New: “Enterprise Bundle Archive” (EBA)

- An isolated, cohesive application consisting of a collection of bundles, is deployed as a logical unit in a “.eba” archive
  - An “OSGi Application”.
- Constituent bundles may be contained (“by-value”) or referenced from a bundle repository.
- Services provided by the application are isolated to the application unless explicitly exposed through EBA-level application manifest
- Config by exception - absence of APPLICATION.MF means:
  - application content is the set of bundles contained by-value plus any repository-hosted dependencies identified during deployment.

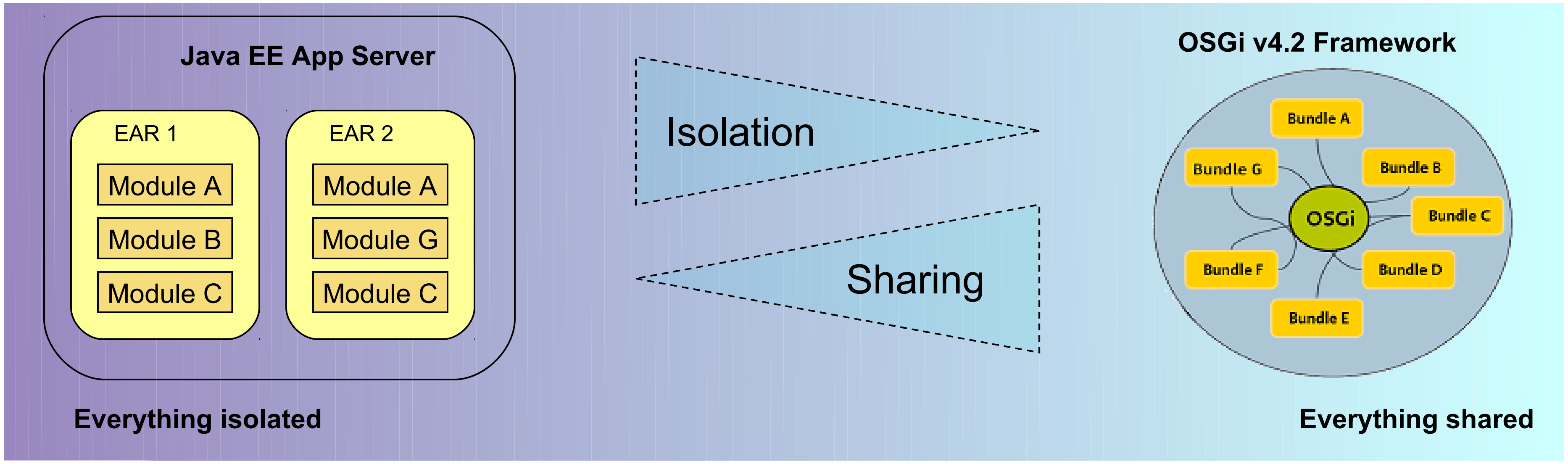


Resolving and “freezing” the deployment



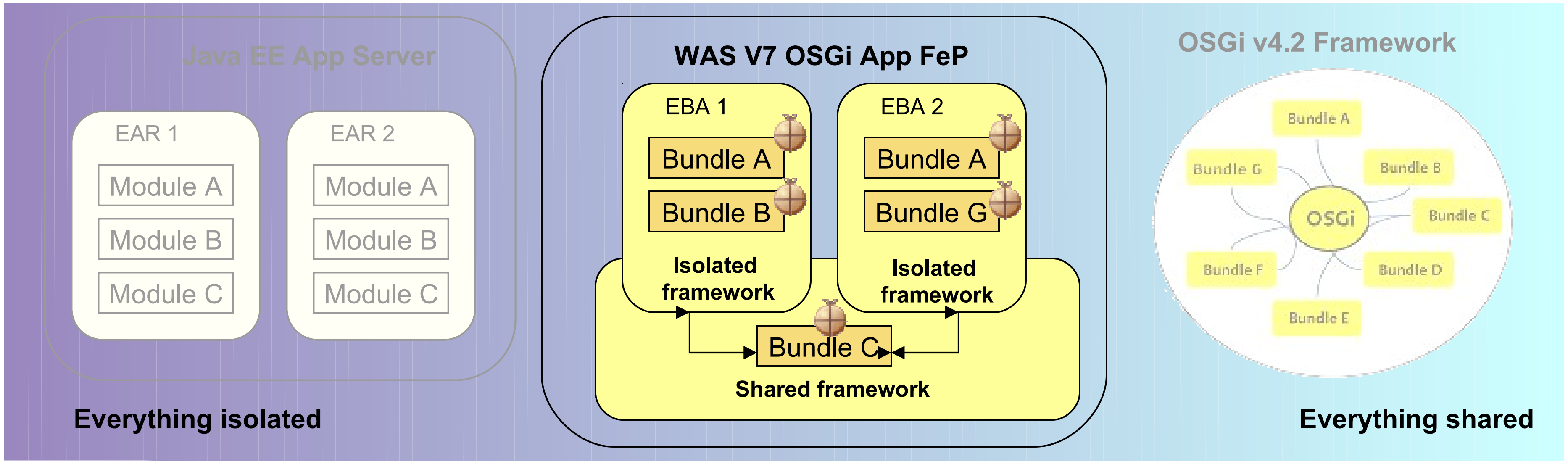
Details: Isolated and Shared Bundles

- In Java EE, modules are isolated within an application and applications are isolated from one another.
  - Makes sharing modules difficult
- OSGi 4.2 all bundles have shared visibility to the externals of all others bundles within an OSGi framework (JVM)
  - Makes isolating applications difficult



Details: Isolated and Shared Bundles

- Equinox 3.5 “nested framework” support enables “composite bundles” to run in isolated child frameworks
  - WebSphere installs each OSGi Application into an isolated child framework.
  - Shared bundles are installed into the (single) parent framework.

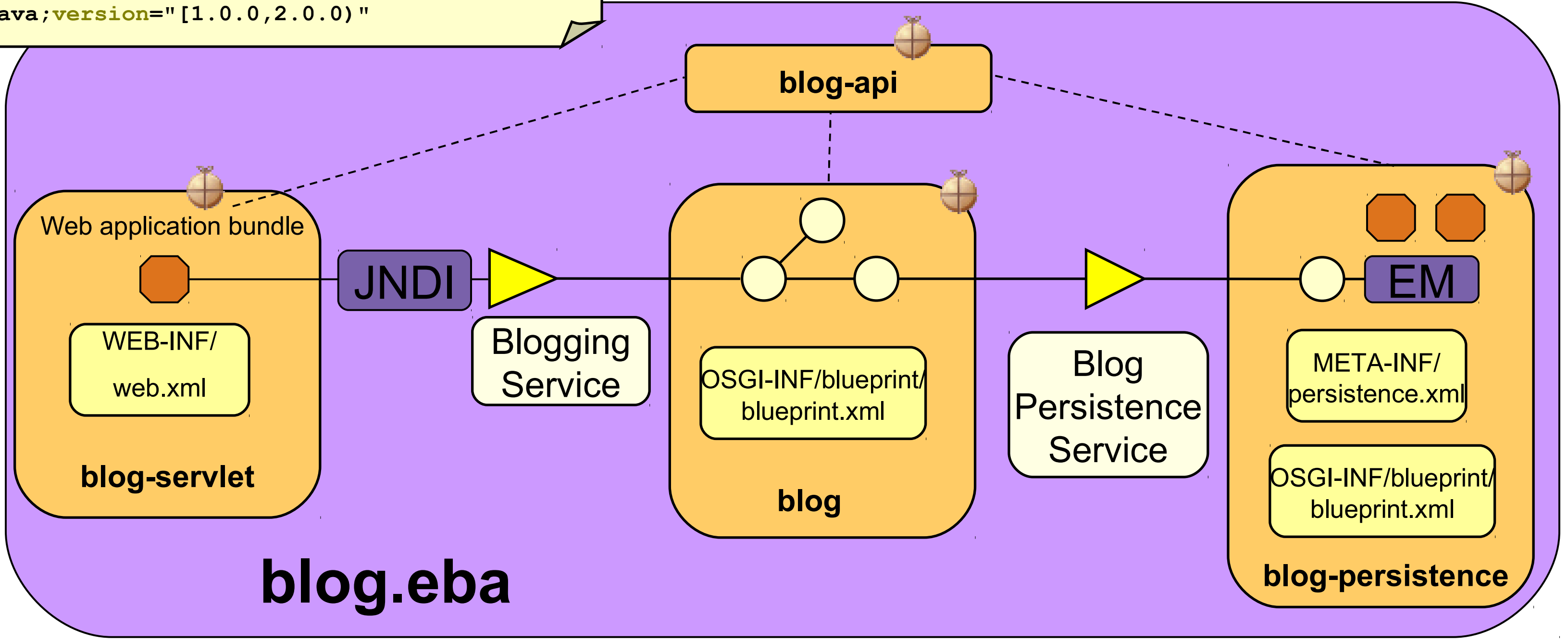


Example "Blog" Application Architecture

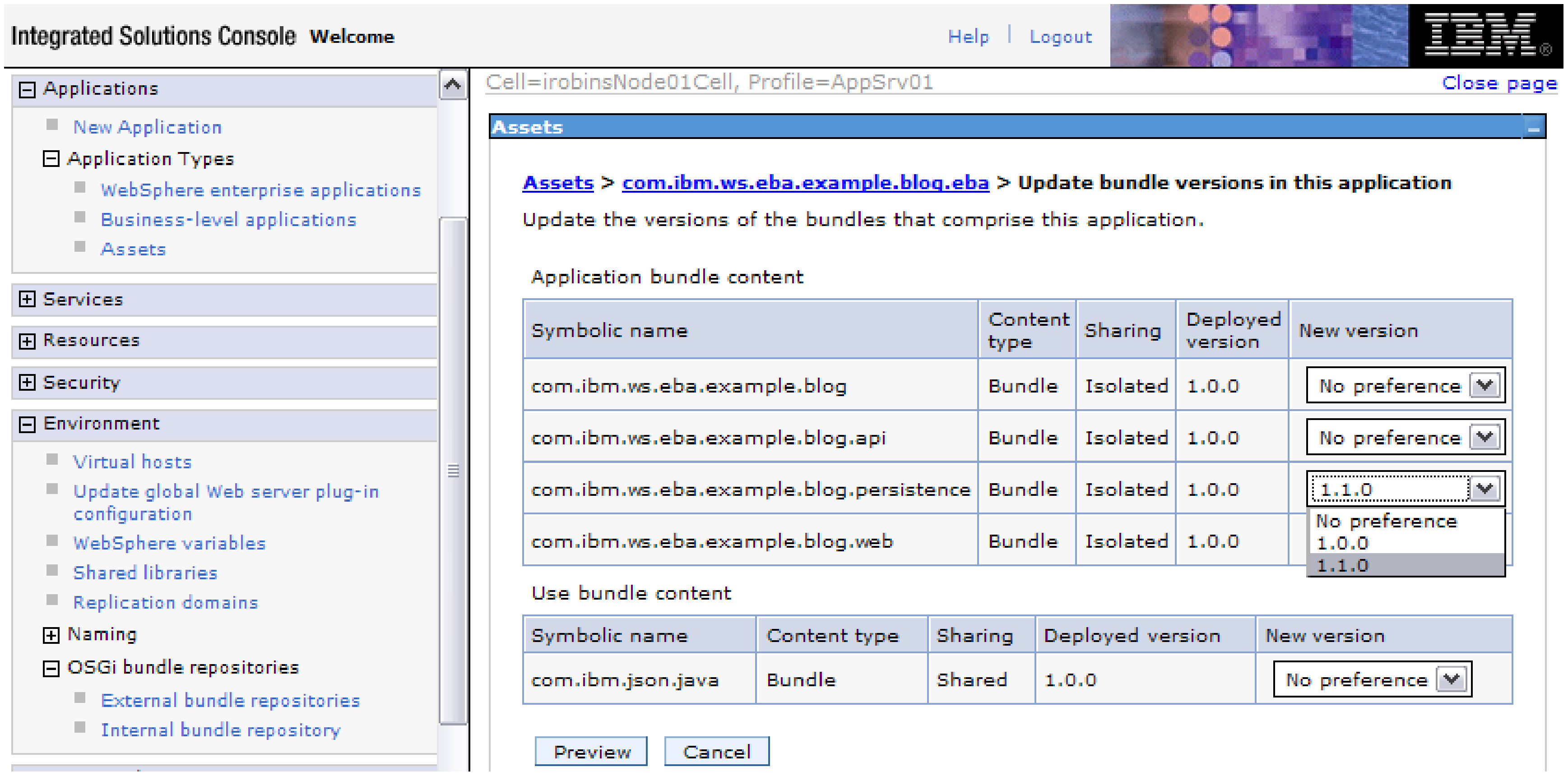
```
Manifest-Version: 1.0
Application-ManifestVersion: 1.0
Application-Name: Aries Blog
Application-SymbolicName: com.ibm.ws.eba.example.blog.app
Application-Version: 1.0
Application-Content:
 com.ibm.ws.eba.example.blog.api;version=1.0.0,
 com.ibm.ws.eba.example.blog.persistence;version=1.0.0,
 com.ibm.ws.eba.example.blog.web;version=1.0.0,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

isolated content

shared content



Application-centric Bundle Management


Integrated Solutions Console Welcome
Help | Logout
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
Cell=irobinsNode01Cell, Profile=AppSrv01
Close page
Assets
Preview
Cancel

Assets > com.ibm.ws.eba.example.blog.eba > **Update bundle versions in this application**

Update the versions of the bundles that comprise this application.

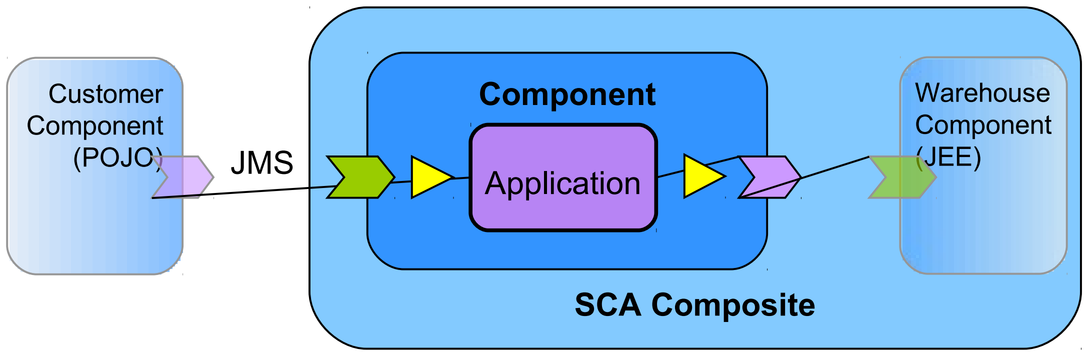
Application bundle content

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	

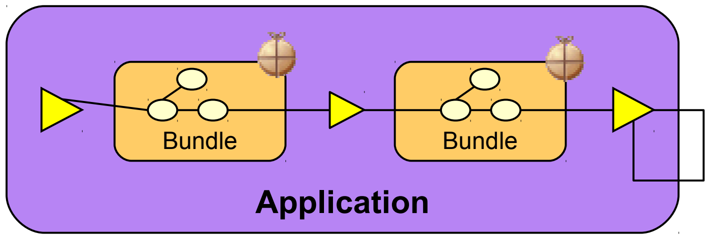
Use bundle content

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

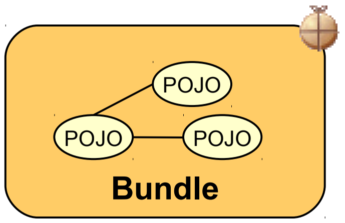
OSGi and SCA: the assembly food chain



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



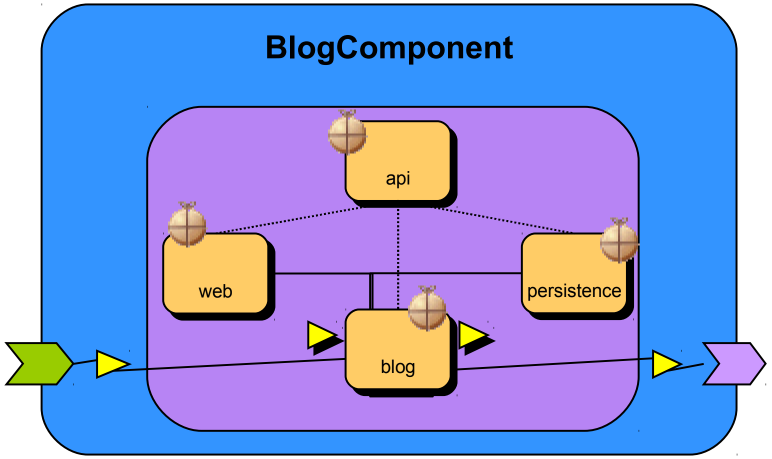
OSGi Bundles assembled in an **OSGi Application** and integrated through services in the OSGi service registry



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

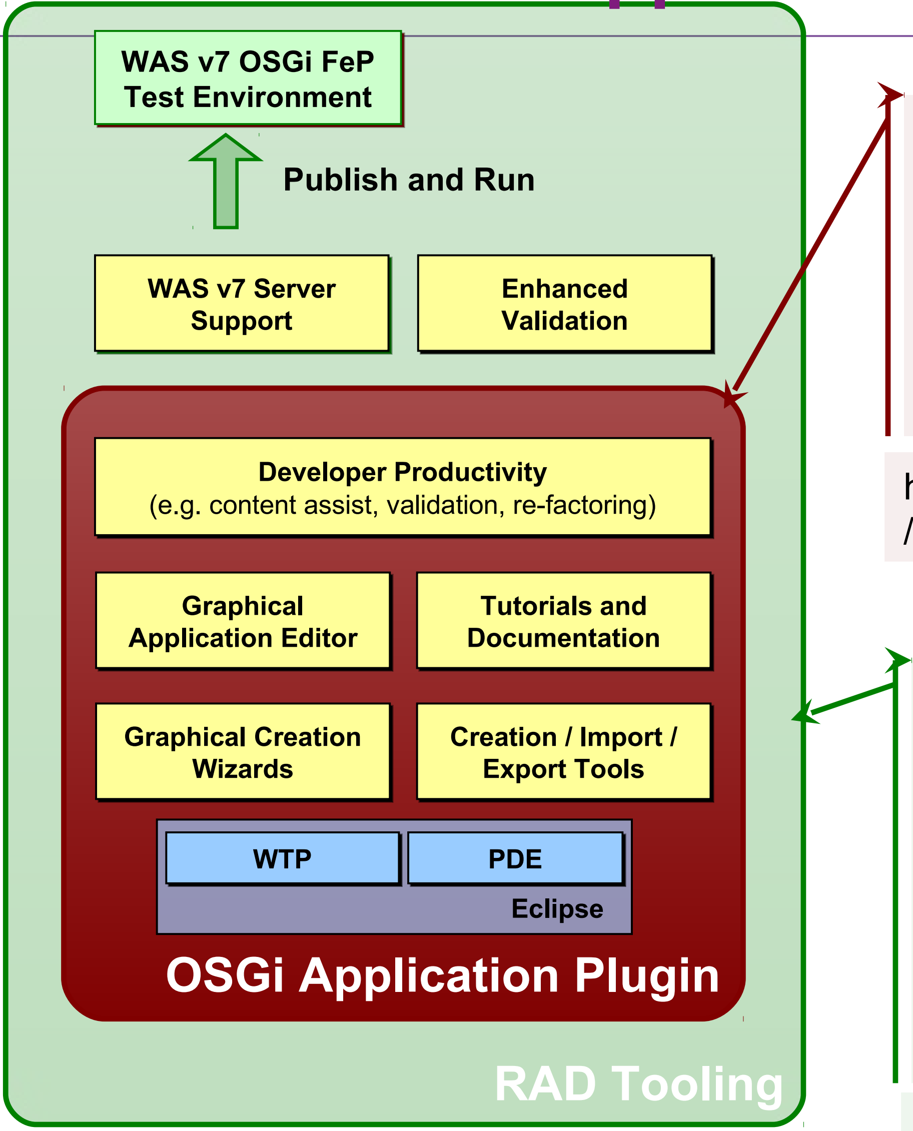
SCA integration: implementation.osgiapp

```
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 Project Support


WAS v7 OSGi FeP Test Environment
Publish and Run
WAS v7 Server Support
Enhanced Validation
Developer Productivity (e.g. content assist, validation, re-factoring)
Graphical Application Editor
Tutorials and Documentation
Graphical Creation Wizards
Creation / Import / Export Tools
WTP
PDE
Eclipse
OSGi Application Plugin
RAD Tooling

Free Eclipse Plugin for OSGi Applications

- ➢Graphical tools to develop OSGi applications and bundles
- ▪ Includes features that increase developer productivity
- ▪ Creates OSGi Applications for any Aries-based server runtime.
- ▪ Eclipse WTP 3.6 (Helios) M6 or later required

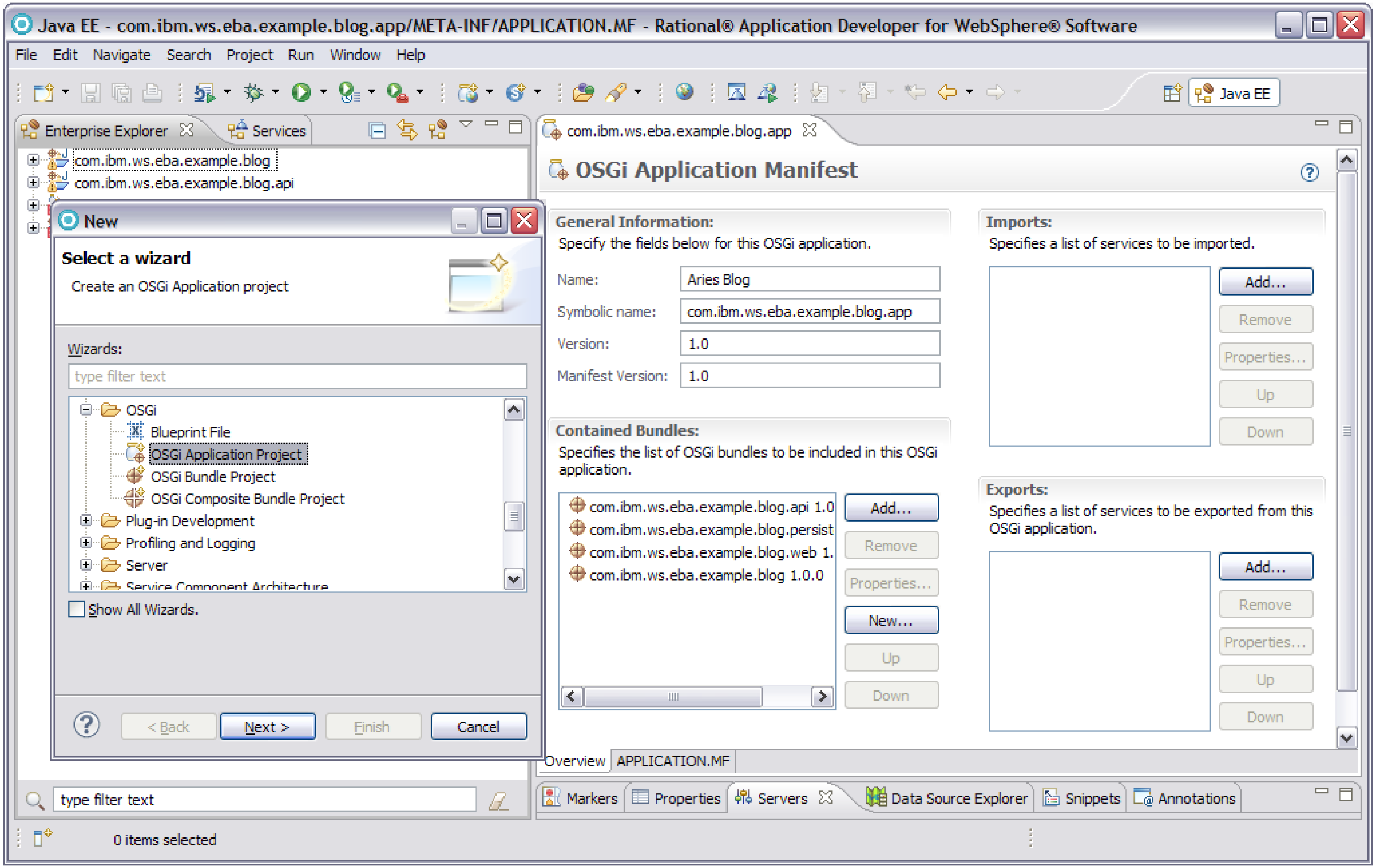
http://www.ibm.com/developerworks/rational/downloads/10/rationaldevtoolsforosgiapplications.html

OSGi Application Support in RAD 8.0

- ➢ Provide integrated development and test of OSGi Applications on the WebSphere platform
- ▪ Integrated with Web Tools, JEE productivity tools, and other capabilities in RAD
- ▪ Supports deployment to WAS v7 OSGi FeP and includes the FeP in the WAS Test Environment
- ▪Enhanced validation

http://www-01.ibm.com/software/awdtools/developer/application/index.html

RAD OSGi Application Project Support



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

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.

type filter text

0 items selected

Key OSGi Feature Pack benefits

Modular and Extensible Applications

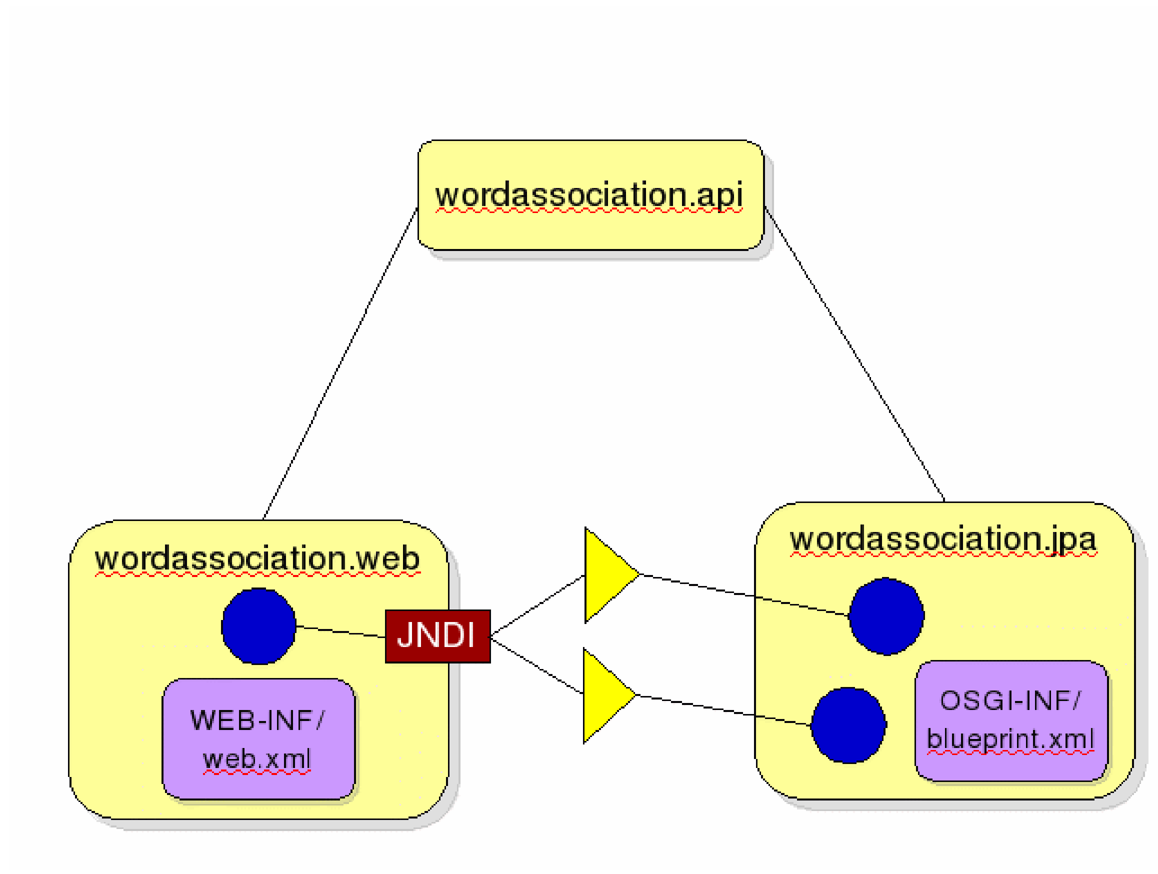
- **Problem:** Complex Java EE applications rely on good development practice to deliver the modular designs required to reduce maintenance cost and promote extensibility.
- **Solution:** The feature pack simplifies the development and deployment of reusable application modules, exploiting application-level OSGi infrastructure integrated into the server runtime.
- **Problem:** Multiple Java EE applications using the same libraries each install their own isolated copies of these libraries
- **Solution:** OSGi applications can be assembled to resolve against libraries in a shared repository, reducing disk and memory footprint
- **Problem:** Application using 3rd party libraries have little control over the dependencies these libraries have and can suffer from versioning conflicts between different libraries.
- **Solution:** OSGi bundle versioning enables multiple versions of the same library to be loaded concurrently within a server

Development Simplicity

- Leverage Spring-like simplicity, including unit test of business components outside the server, but governed by OSGi standards and integrated with proven Java EE technologies
- WebSphere integrated to deliver enterprise scalability, management and QoS
- Optionally composeable as SCA components into SOA applications

Questions?

Word association architecture



Copyright and Trademarks