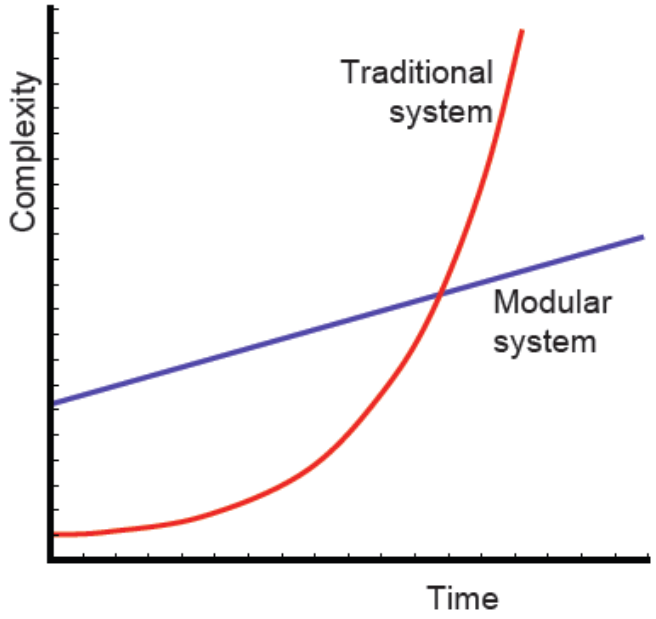
Modular & Dynamic OSGi Applications in WebSphere Application Server

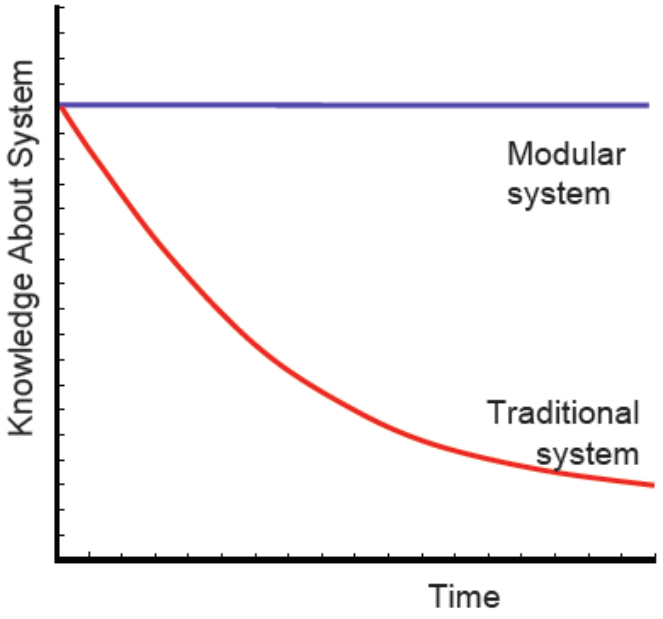
Zhu Xiu Lei (xiuleizh@cn.ibm.com)
Chen Zhi Xian (zxchen@cn.ibm.com)

Agenda

- **Part 1**
  - Why Does Complexity Tend to Increase?
  - Introduction to OSGi
- **Part 2**
  - OSGi Application Support in WebSphere
  - Using OSGi to Develop and Manage Enterprise Applications
    - Modular
    - Dynamic
    - Extensible

Complexity and System Rot





In a software system, *entanglement* is the primary cause of decay.

Why does it happen?
How do we prevent it?

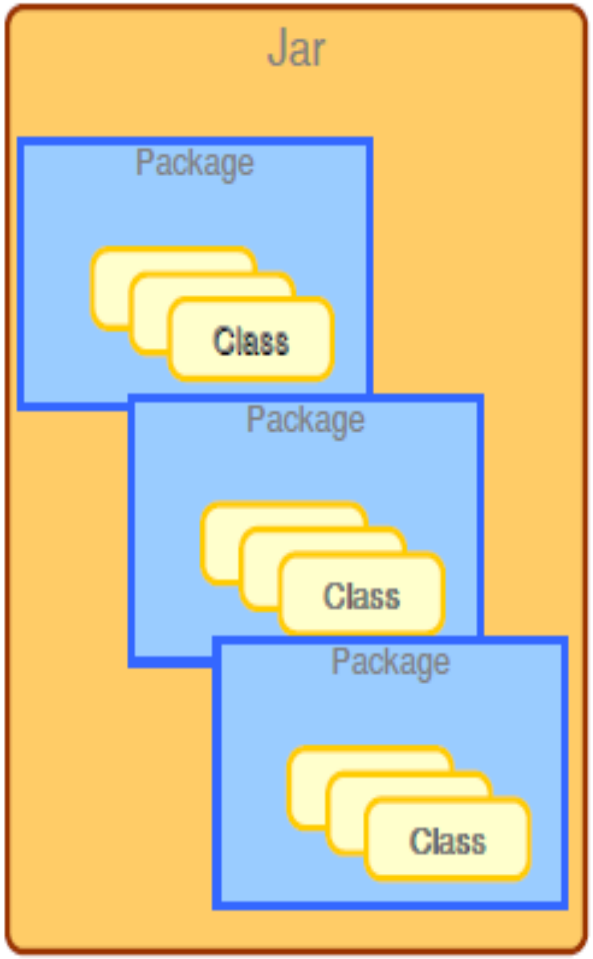
What and why of OSGi ?

Java's modularity limitations (1 of 2)

- Java unit of modularity = JAR*
- Enterprise apps are collections of JARs

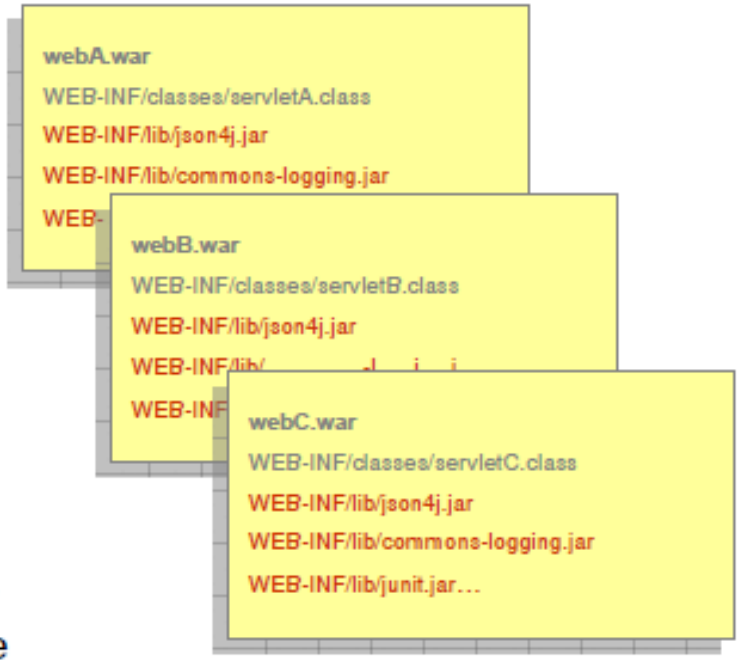
But a JAR lacks the primary characteristics of modularity:

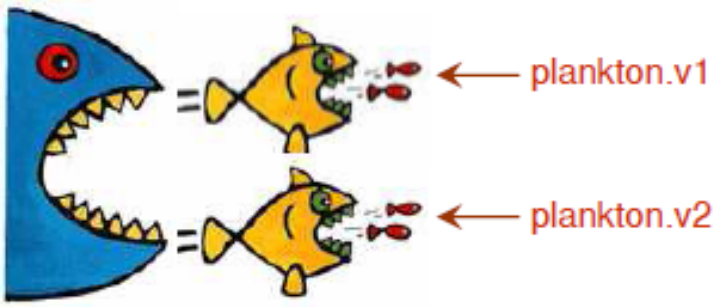
- It does NOT hide its internals
- It does NOT declare its externals
- The global Java class path does NOT support version handling



Java's modularity limitations (2 of 2)

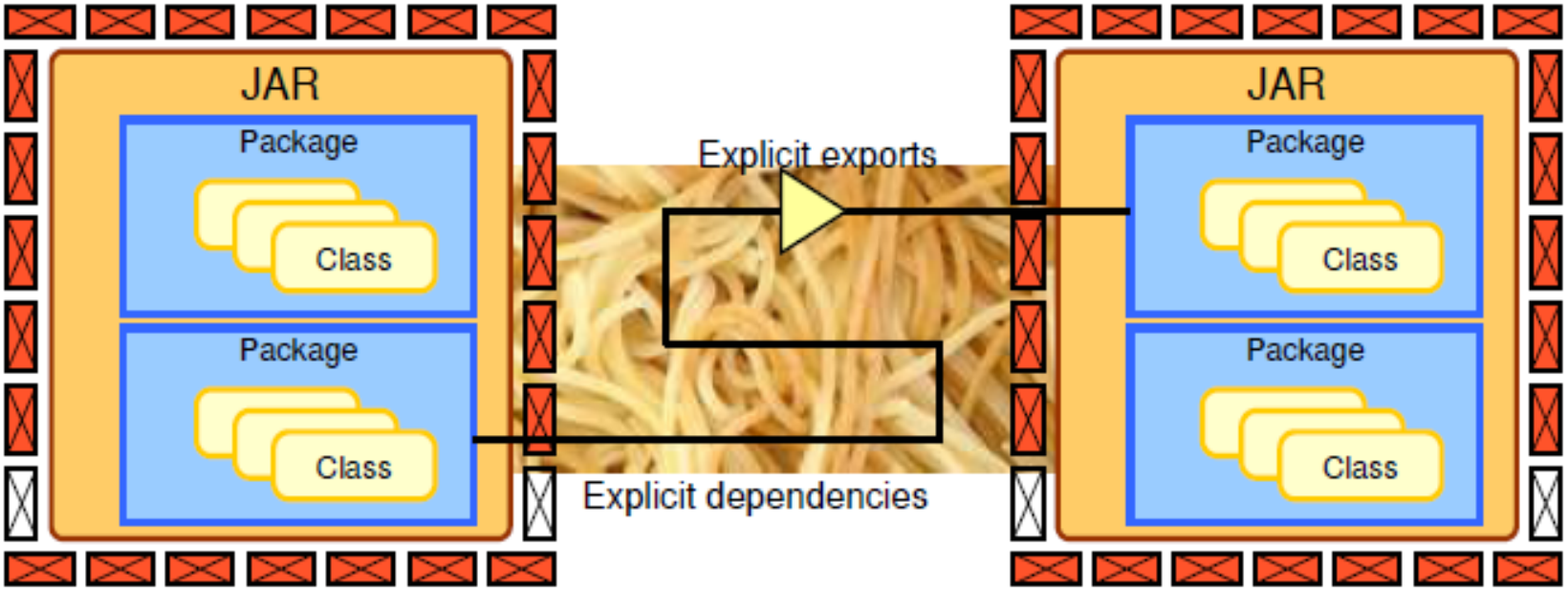
- For enterprise applications, no matter how modular the Application is, the EAR deployment process is lacking
- *Across applications* - *each* archive typically contains *all* the libraries required by the application
  - Common libraries/frameworks get installed with each application
  - Multiple copies of libraries in memory
- *Within applications* - third party libraries consume other third party libraries leading to conflicting versions on the application class path.





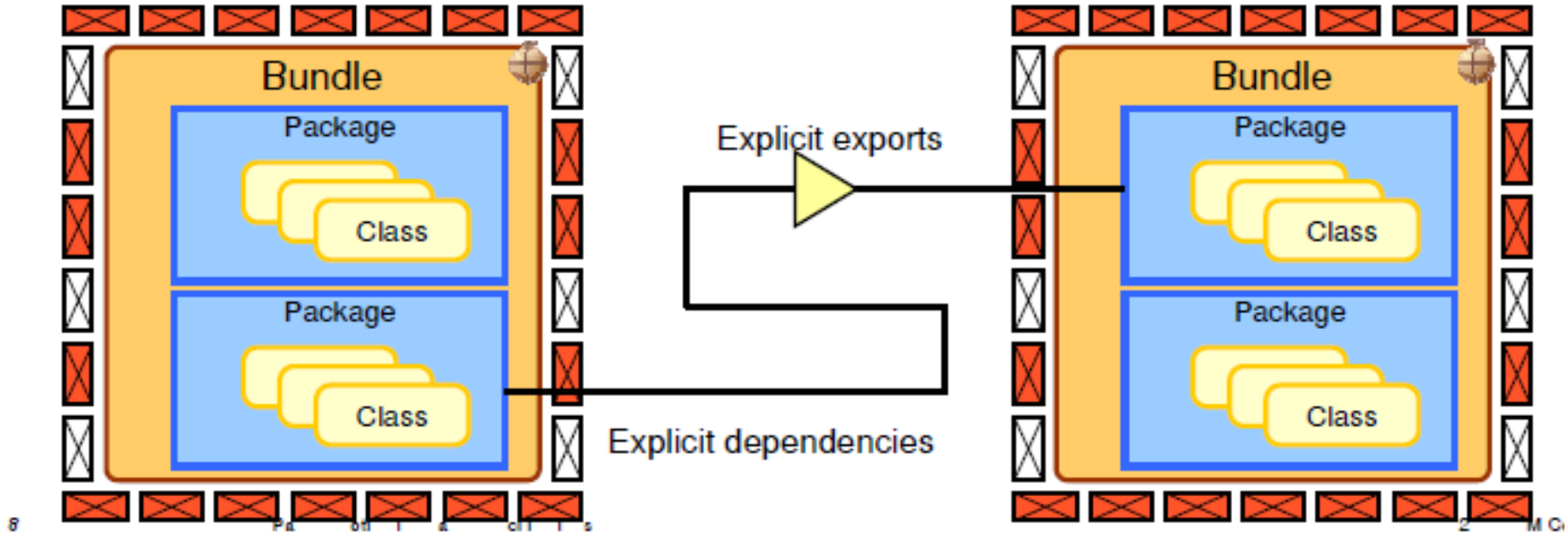
What is OSGi?

- "The dynamic module system for Java"
  - Mature 10-year old technology
  - Governed by OSGi Alliance: http://www.osgi.org
  - Used *inside* just about *all* Java-based middleware
    - IBM WebSphere, Oracle WebLogic, Red Hat JBoss, Sun GlassFish, Paremus Service Fabric, Eclipse Platform, Apache Geronimo, (non-exhaustive list) http://www.osgi.org/wiki/uploads/News/2008_09_16_worldwide_market.pdf



How does OSGi help reduce cost?

- Enforces architecture and simplifies maintenance
- Enables modular deployment
- Enables co-existence of multiple versions of libraries
  - Simplifies independent evolution of applications
  - Better separation of concern between application and middleware
- Enables truly dynamic update of modules within applications



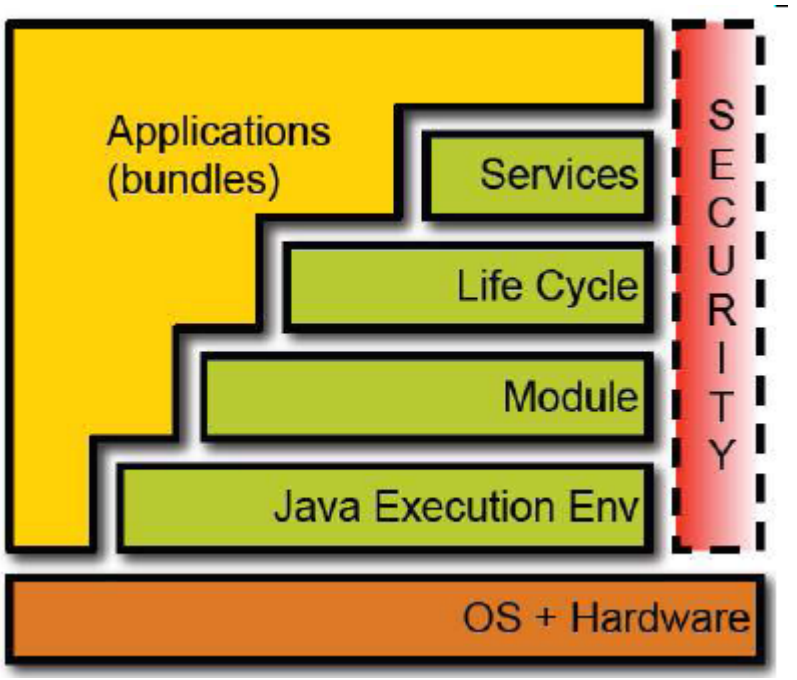
OSGi Benefits for JEE Developers

- Better modularity and reusability
- Defined and enforced dependency management
- Simplified classpath and visibility rules
- Cleaner integration of third-party libraries
  - Shared-library support is built-in
- Simplified unit test

OSGi Benefits for JEE Administrators

- Simplified deployment
- Fine-grained control over bundles
- Fine-grained restart/replacement
- Built-in shared library support
- Lower disk or memory footprint (through componentization)

Some details to understand OSGi

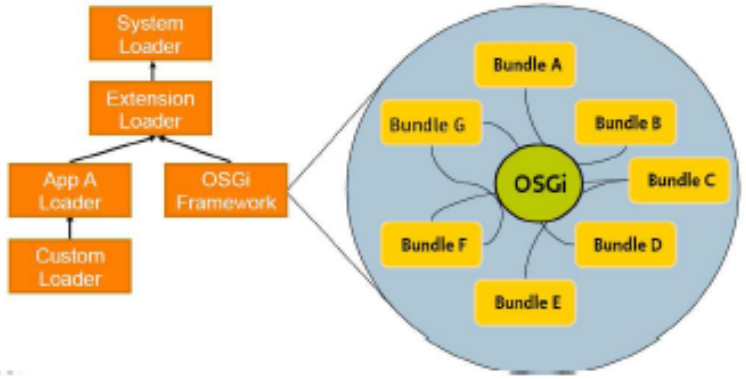


OSGi Bundles and Class Loading (1 of 2)

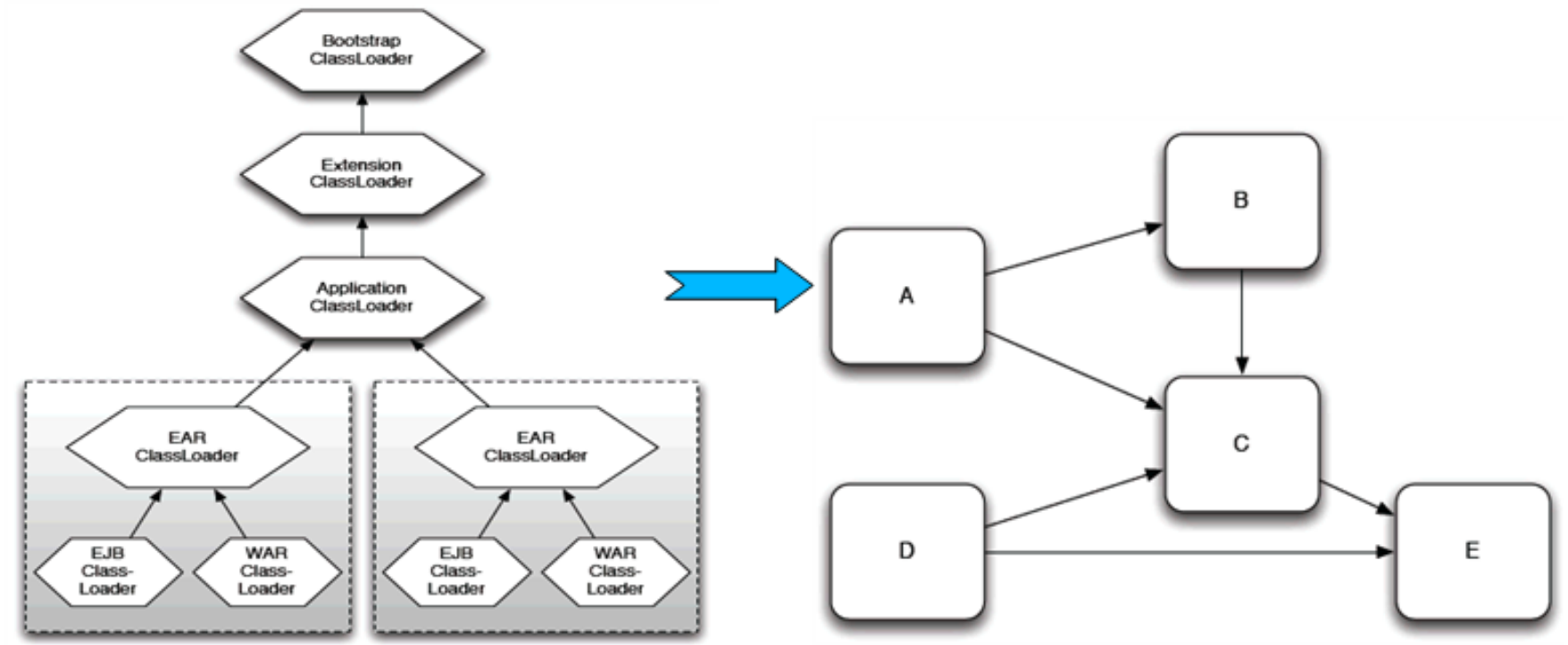
- OSGi Bundle – A jar containing:
  - Classes and resources.
  - OSGi Bundle manifest.
- What's in the manifest:
  - Bundle-Version: Multiple versions of bundles can live concurrently.
  - Import-Package: What packages from other bundles does this bundle depend upon?
  - Export-Package: What packages 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 class path.
  - Class sharing and visibility decided by declarative dependencies, not by class loader hierarchies.
  - OSGi framework works out the dependencies including versions



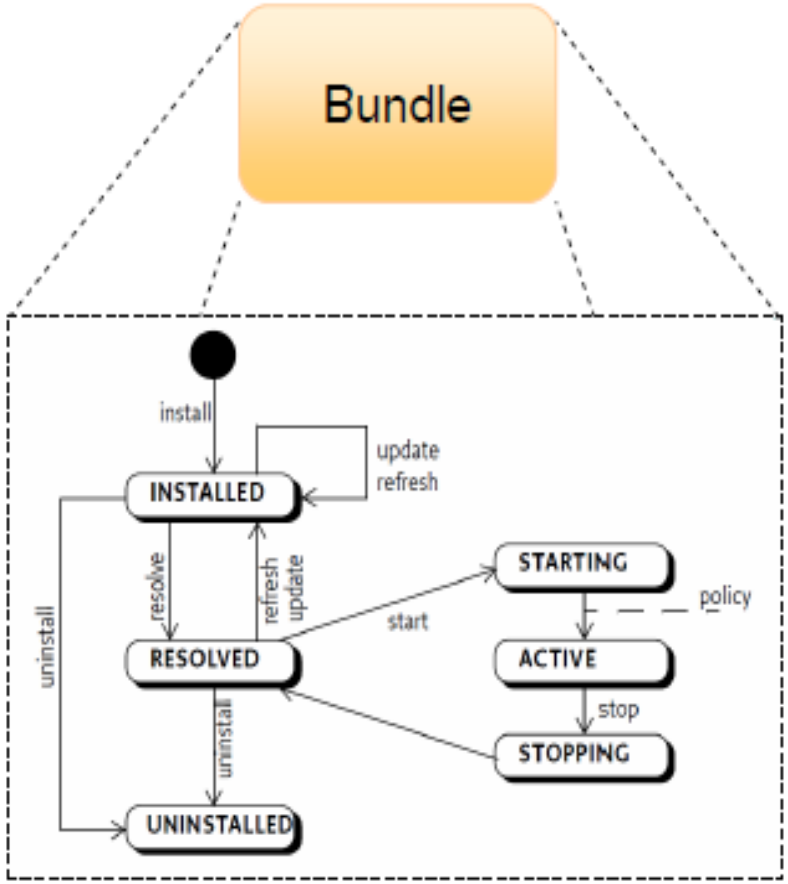
OSGi Bundles and Class Loading (2 of 2)



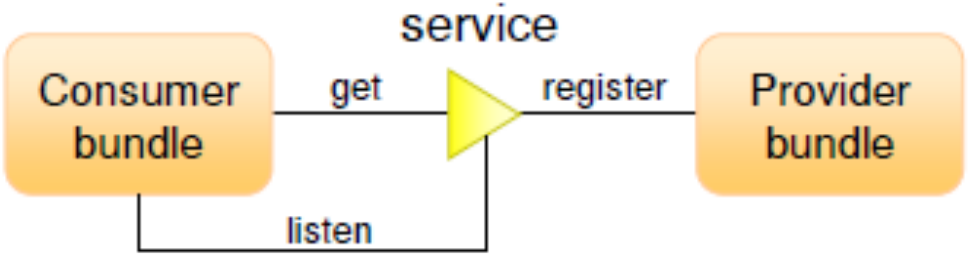
- The benefit of each bundle has its own class loader.
- How to resolve Version Conflict problem?
- How to implement Version Control?

Dynamic Lifecycle

- Bundles have a dynamic life cycle
- Can come and go independently
- APIs enable graceful reaction to changes



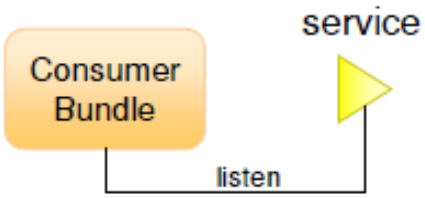
OSGi services



- Publish/find/bind service model
  - Fully dynamic
  - Local
  - Non-durable
- POJO* advertised with properties and/or interface and/or class
- Primary mechanism for bundle collaboration

Bundle and service dynamics (1 of 6)

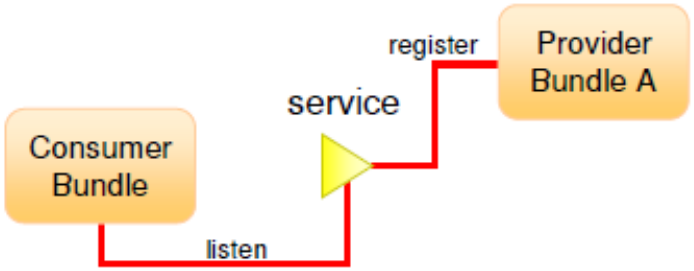
Consumer 'listens' for required service



For example, a bundle that wants to consume a service listens for that service to become available in the Service Registry. Initially there are no services available.

Bundle and service dynamics (2 of 6)

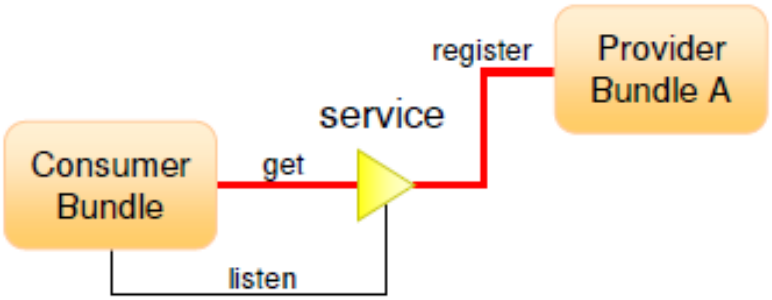
Provider bundle started
- registers service
- consumer notified



Then when a provider bundle is started and a service it provides is registered in the OSGi Service Registry, the consumer bundle is notified.

Bundle and service dynamics (3 of 6)

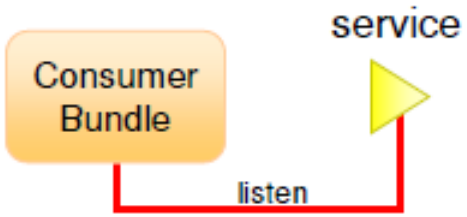
Consumer bundle gets and calls provider service



The consumer bundle gets an instance of the service and uses it.

Bundle and service dynamics (4 of 6)

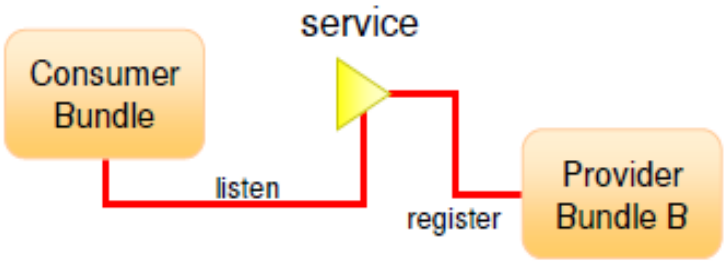
Provider bundle stopped
- unregisters service
- consumer notified



While the consumer is using the service, the bundle providing the service can be stopped, causing the service to be unregistered from the service registry. The consuming bundle is notified and from this point on, the consuming bundle can no longer use that service.

Bundle and service dynamics (5 of 6)

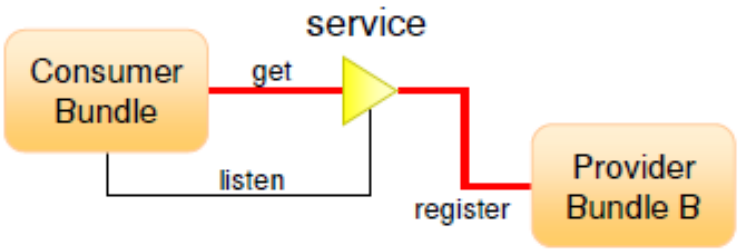
New provider bundle started
- *registers service*
- *consumer notified*



Subsequently a new bundle may be started and register an alternative implementation of the service interface being listened for by the consumer. The consumer bundle will be notified of this new registration.

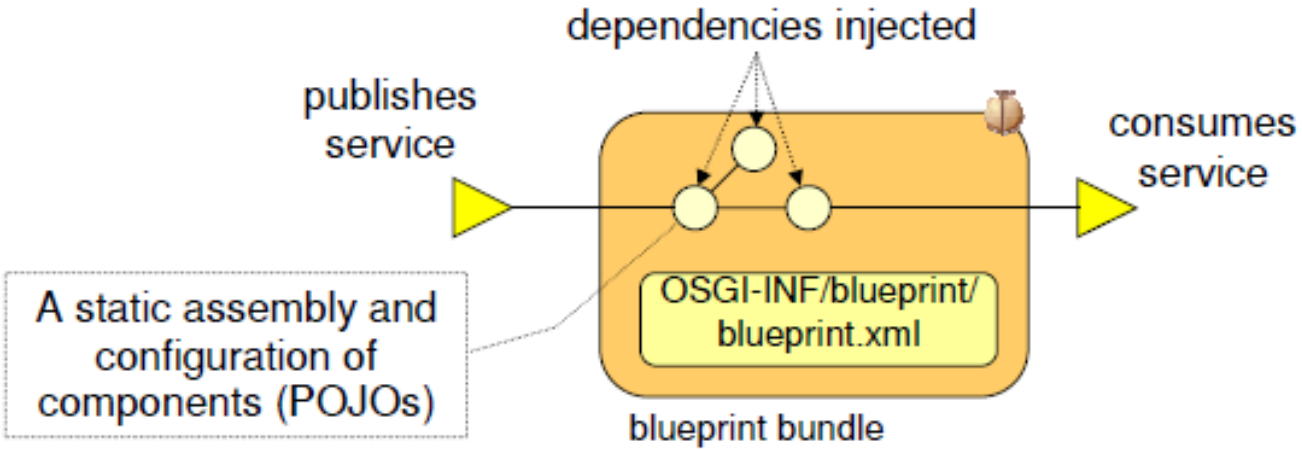
Bundle and service dynamics (6 of 6)

Consumer bundle gets and calls provider service



The consumer bundle may then get an instance of the service and one will be provided by the new provider.

Declarative OSGi Services Using Blueprint

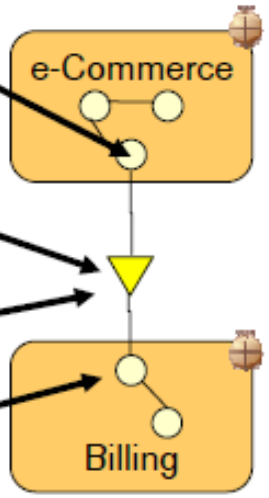


- XML Blueprint definition describes component configuration and scope
  - Optionally publish and consume components to/from OSGi service registry.
  - Standardizes established Spring conventions
- Simplifies unit test outside either Java EE or OSGi r/t.
- **In WebSphere Application Server, the Blueprint DI container is a part of the server runtime (compared to the Spring container which is part of the application.)**

Blueprint Simplifies Service Dynamism

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

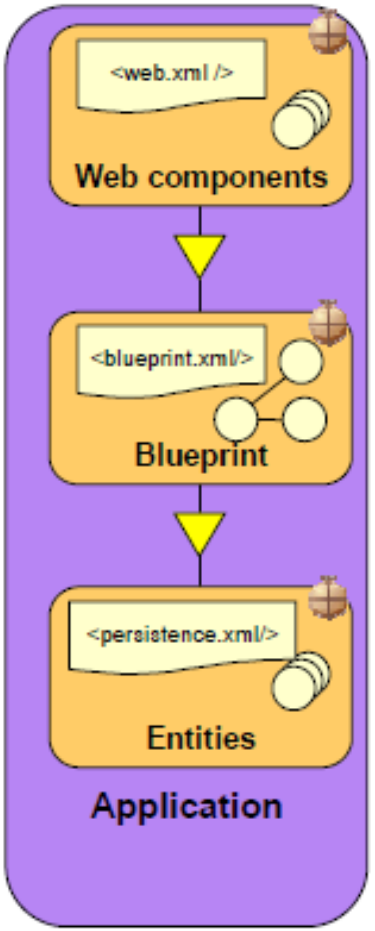
```
<blueprint>
  <service ref="service" interface =
           "org.example.bill.BillingService" />
  <bean id="biller" scope="prototype"
     class="org.example.bill.impl.BillingServiceImpl" />
</blueprint>
```



- Dynamic service life cycle is managed by the Blueprint container
- Service reference injected by container
  - service can change over time
  - can be temporarily absent without the bundle caring

How Do Enterprise Applications Use OSGi?

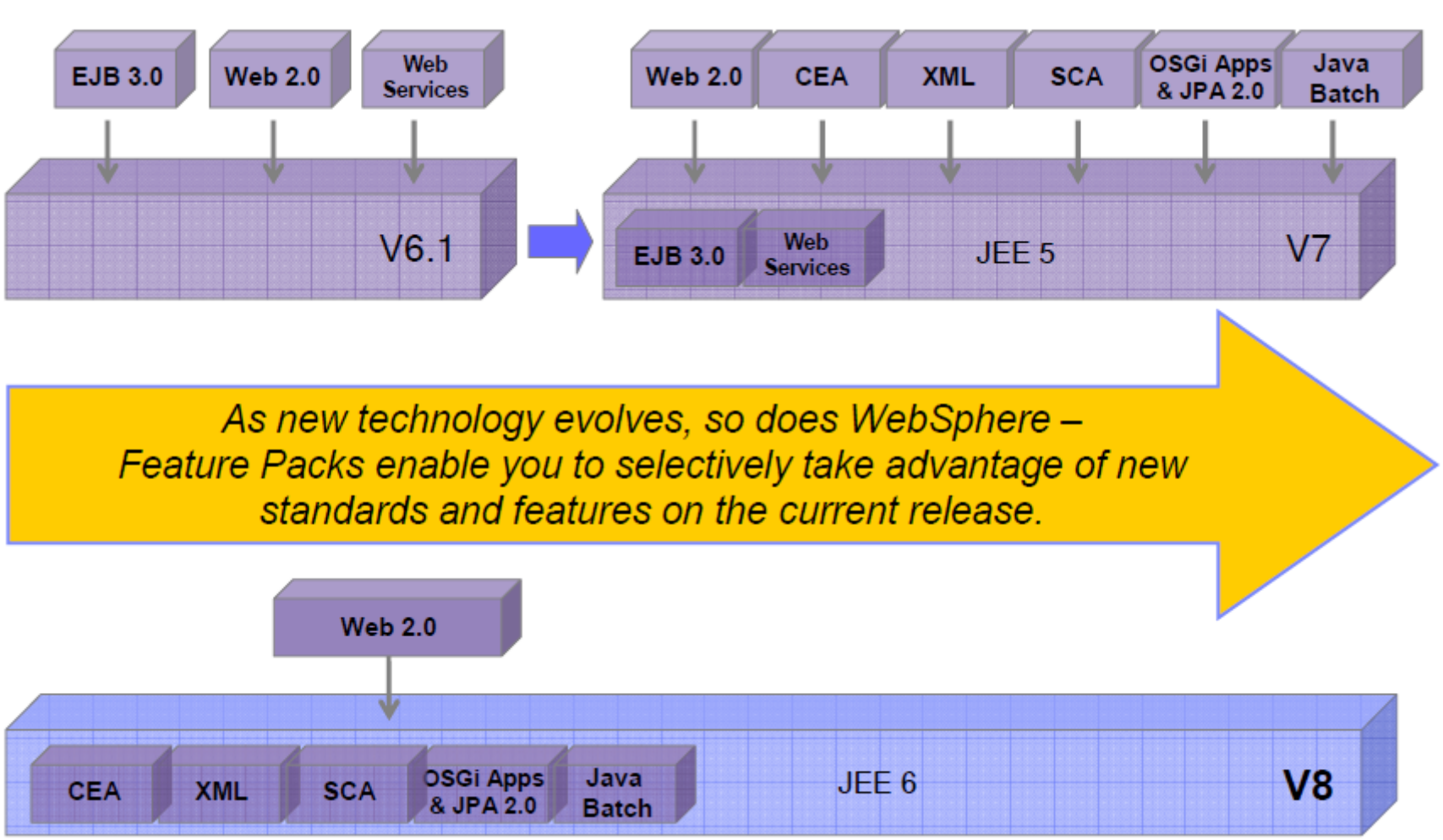
- *Enterprise OSGi* focuses on application concerns including web technologies, fine-grained component assembly and access to persistence frameworks
  - through exploitation of familiar Java EE technologies
  - in a manner suitable for a dynamic OSGi environment
  - using standards defined by the JCP and OSGi Alliance
  - introduces a multi-bundle application archive
- Enables enterprise application containers and deployment systems to provide better support for:
  - Sharing modules between applications
  - Multiple concurrent versions of modules
  - Dynamic update and extensions of applications
- Find out more in part 2…



- Part 1
  - Why Does Complexity Tend to Increase?
  - Introduction to OSGi

- **Part 2**
  - OSGi Application Support in WebSphere
  - Using OSGi to Develop and Manage Enterprise Applications
    - Modular
    - Dynamic
    - Extensible

WebSphere Application Server feature strategy



OSGi Applications in WebSphere Applications Server

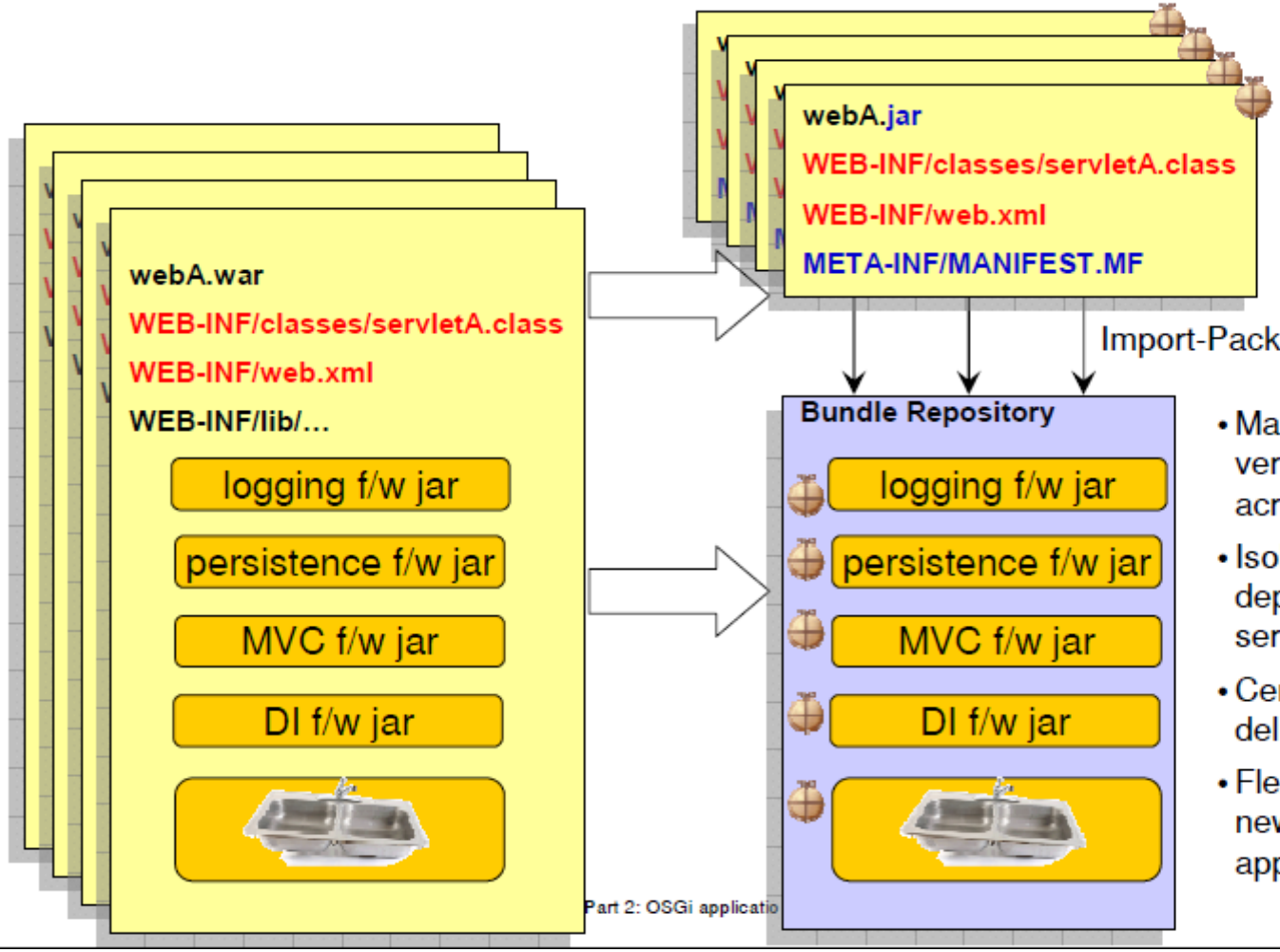
- OSGi has been used internally in WebSphere Application Server since V6.1 and in Eclipse since R3.
- Application-level exploitation is introduced in the WebSphere Application Server V7 Feature Pack for OSGi Applications and JPA 2.0
  http://www.ibm.com/websphere/was/osgi
  - Focused on modular development, deployment and management
  - Informed by experience gained in Apache Aries
  - Introduced the Blueprint-standardization of Spring XML bean definition format
  - Introduced web application bundles (Java EE 5 web technologies)
  - Integration with SCA for heterogeneous assembly and remote services.
  - Basic supports for application update and extension
- OSGi Application support enhanced in WebSphere Application Server V8
  - In-place application update and extension
  - Post-deployment configuration
  - Performance metrics
  - Integrates with Java EE 6 web technologies
- Development tools support in Rational® Application Developer V8 for both WebSphere Application Server V7 and WebSphere Application Server V8:
  http://www.ibm.com/software/awdtools/developer/application/index.html

Getting started with OSGi support in WebSphere

Enhanced modular deployment process

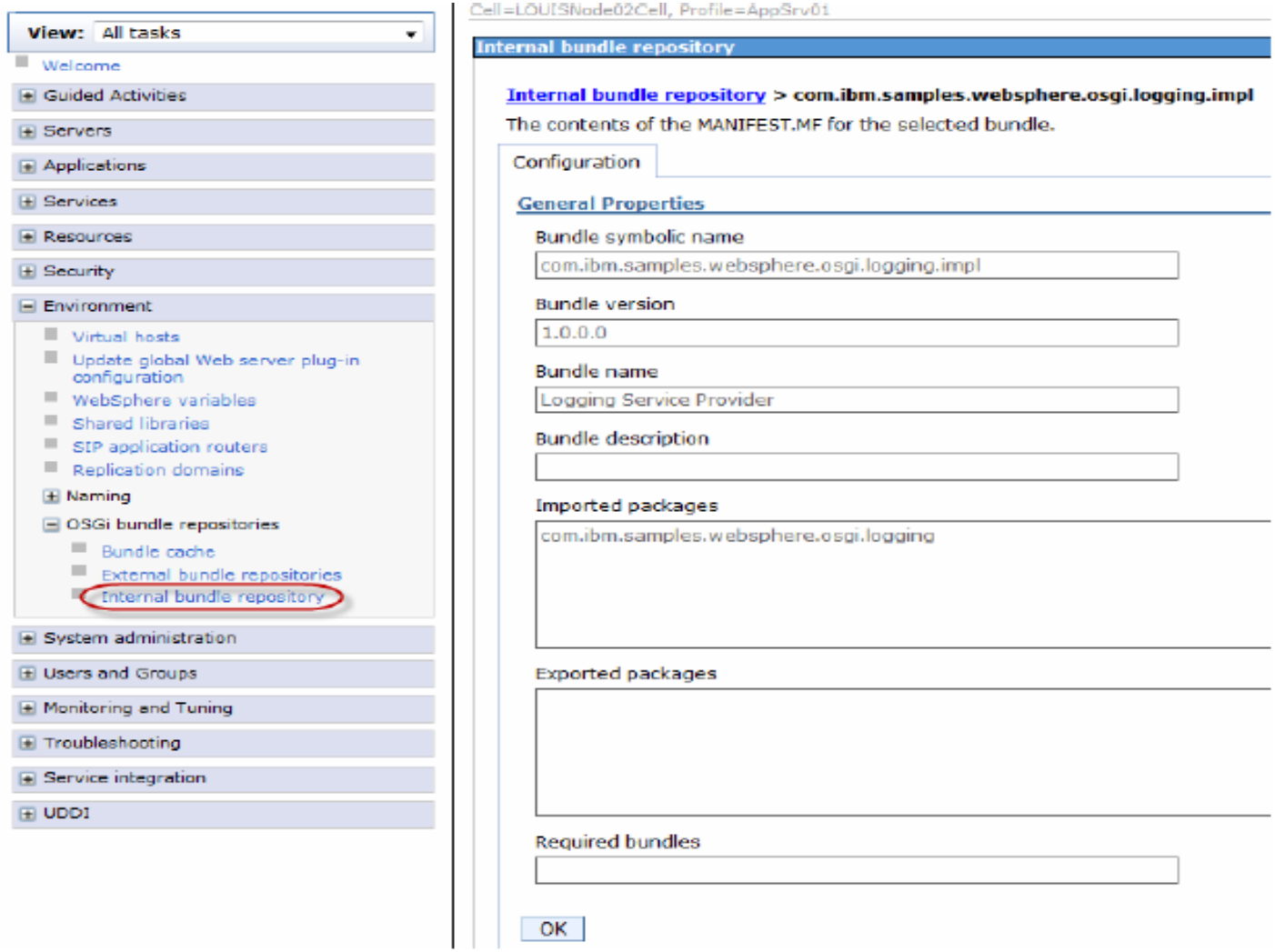
No Java code changes; war modules -> bundles

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



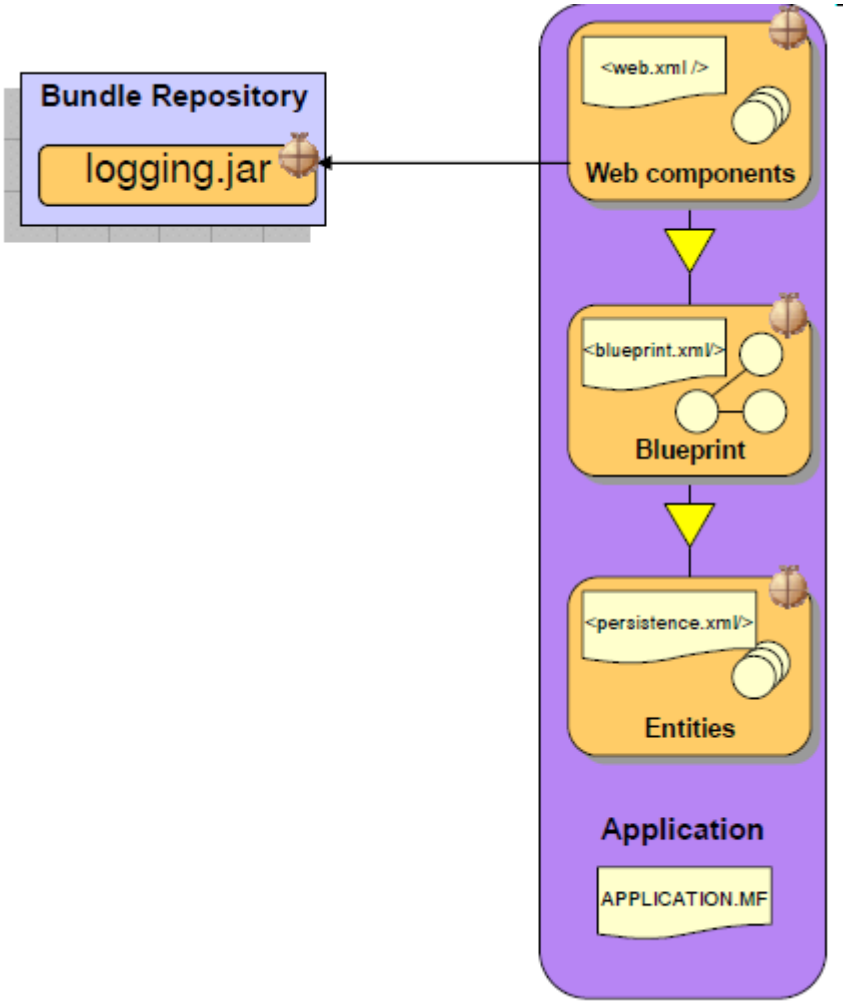
- Manage multiple versions of libraries across an enterprise
- Isolate application dependencies from the server runtime
- Centralized location to deliver critical fixes
- Flexibility to update to new versions one application at a time

Bundle repository for modular deployment



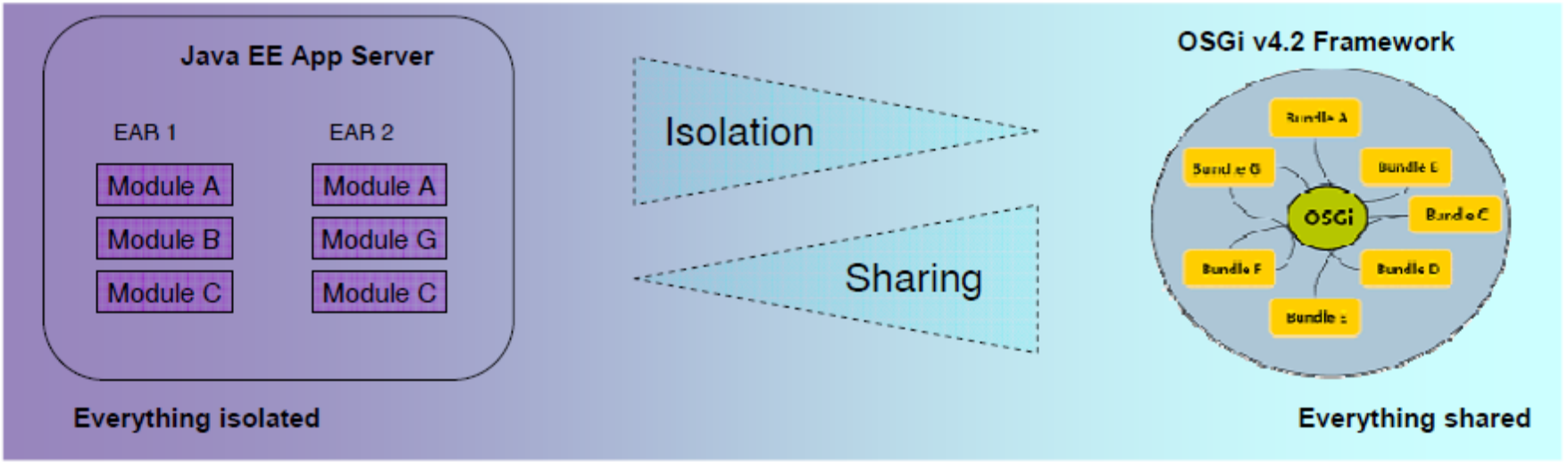
Application-level metadata and archive

- 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 are listed in the **APPLICATION.MF** and can be contained ("by-value") in the archive
  - if simply referred to, then they will be provisioned from a bundle repository at deployment time
- Services provided by the bundles in the application are isolated to the application unless explicitly exposed through EBA-level application manifest
- Configuration by exception – optional application-level metadata in APPLICATION.MF



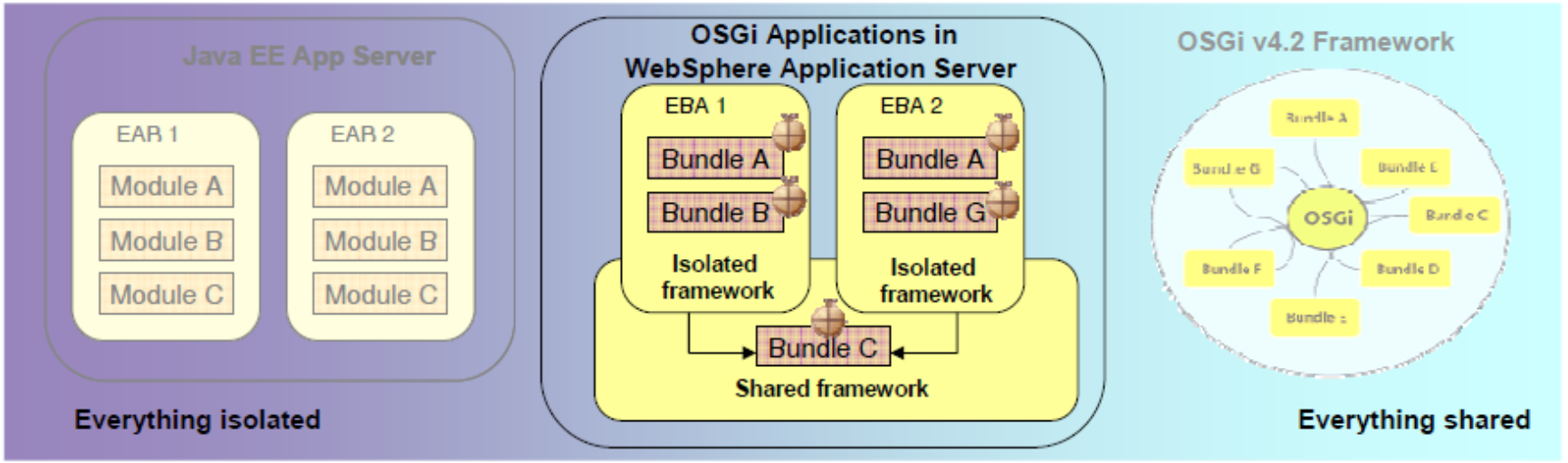
Isolated and shared bundles (1 of 2)

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



Isolated and shared bundles (2 of 2)

- The Equinox OSGi f/w used by WebSphere Application Server 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

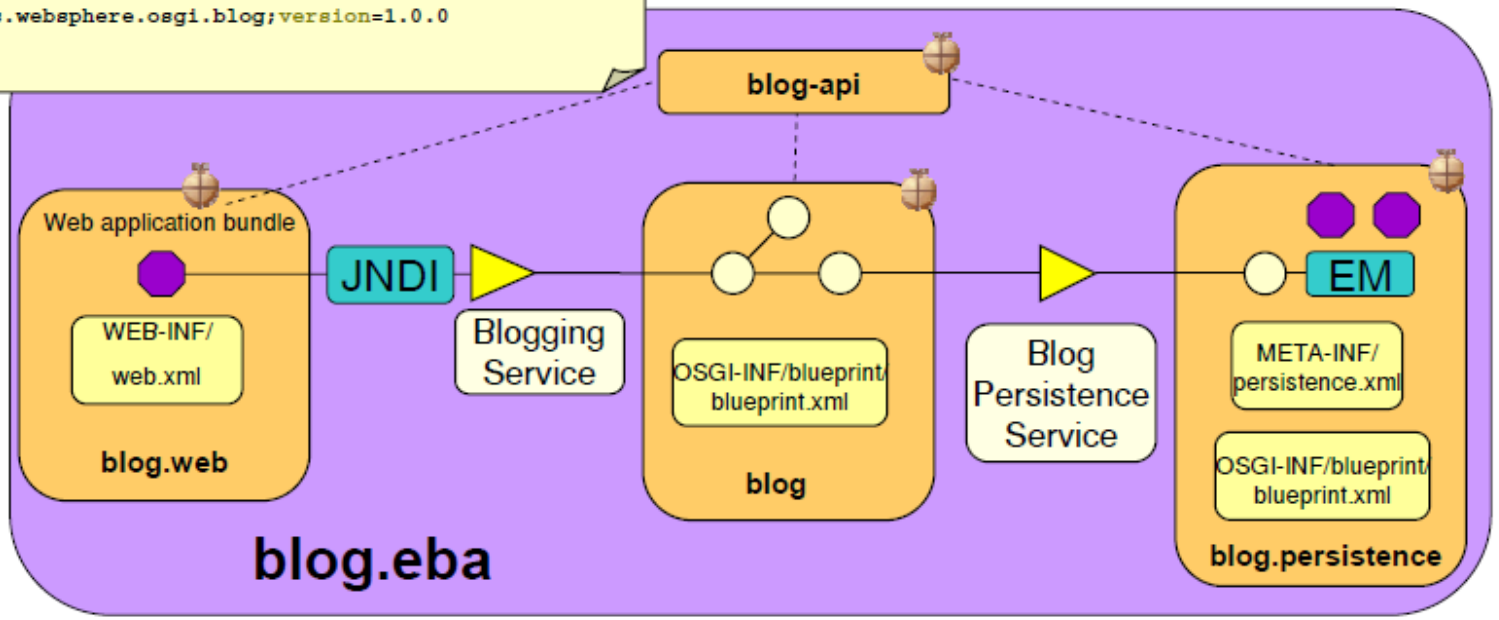


Demo:
“Blog” application install and dynamic update

Example “Blog” application architecture

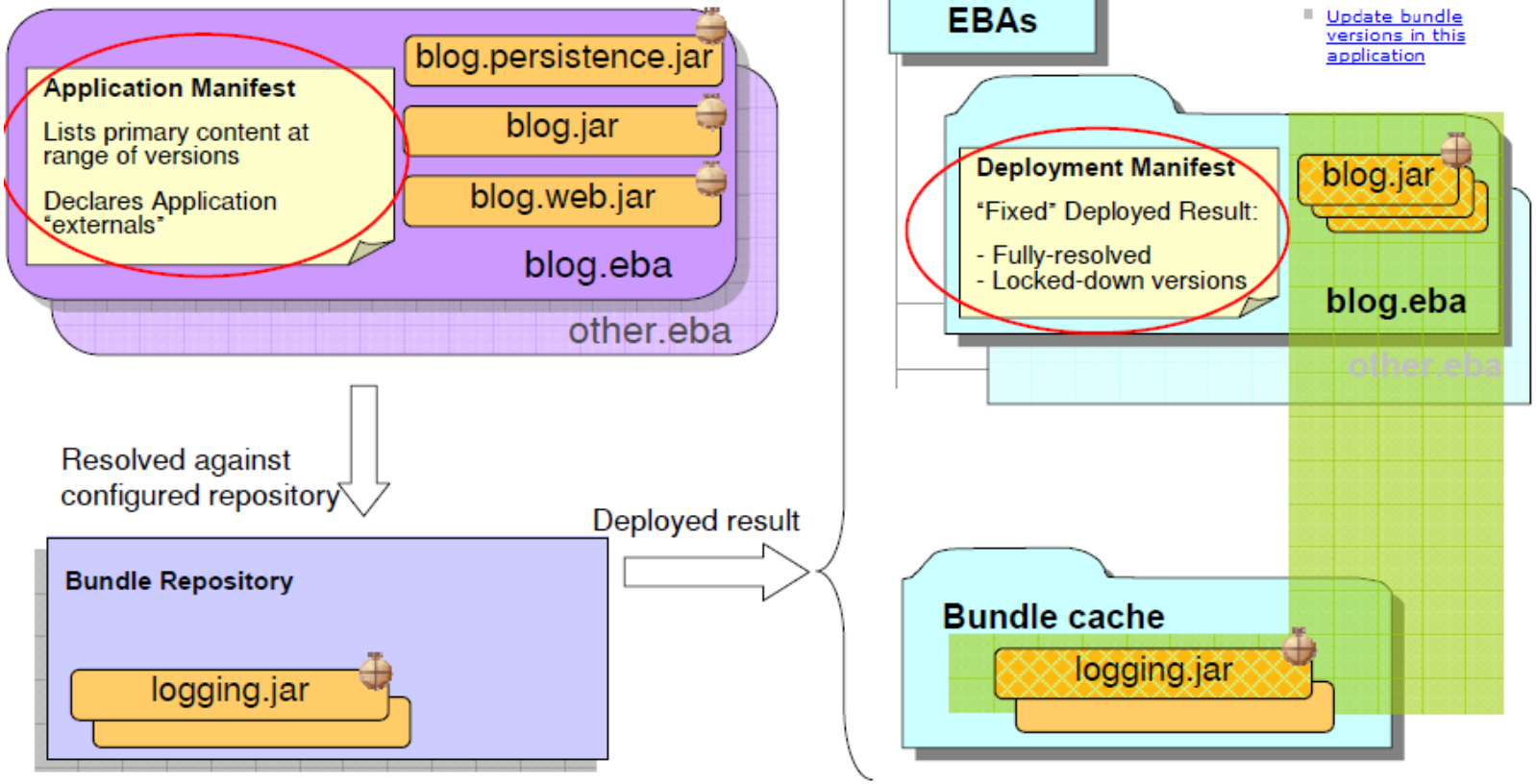
```
Manifest-Version: 1.0
Application-ManifestVersion: 1.0
Application-Name: Aries Blog
Application-SymbolicName: com.ibm.samples.o      og.app
Application-Version: 1.0
Application-Content:
 com.ibm.samples.websphere.osgi.blog.api;version=1.0.0,
 com.ibm.samples.websphere.osgi.blog.persistence;version=1.0.0,
 com.ibm.samples.websphere.osgi.blog.web;version=1.0.0,
 com.ibm.samples.websphere.osgi.blog;version=1.0.0
```

Application-Content header defines isolated application content



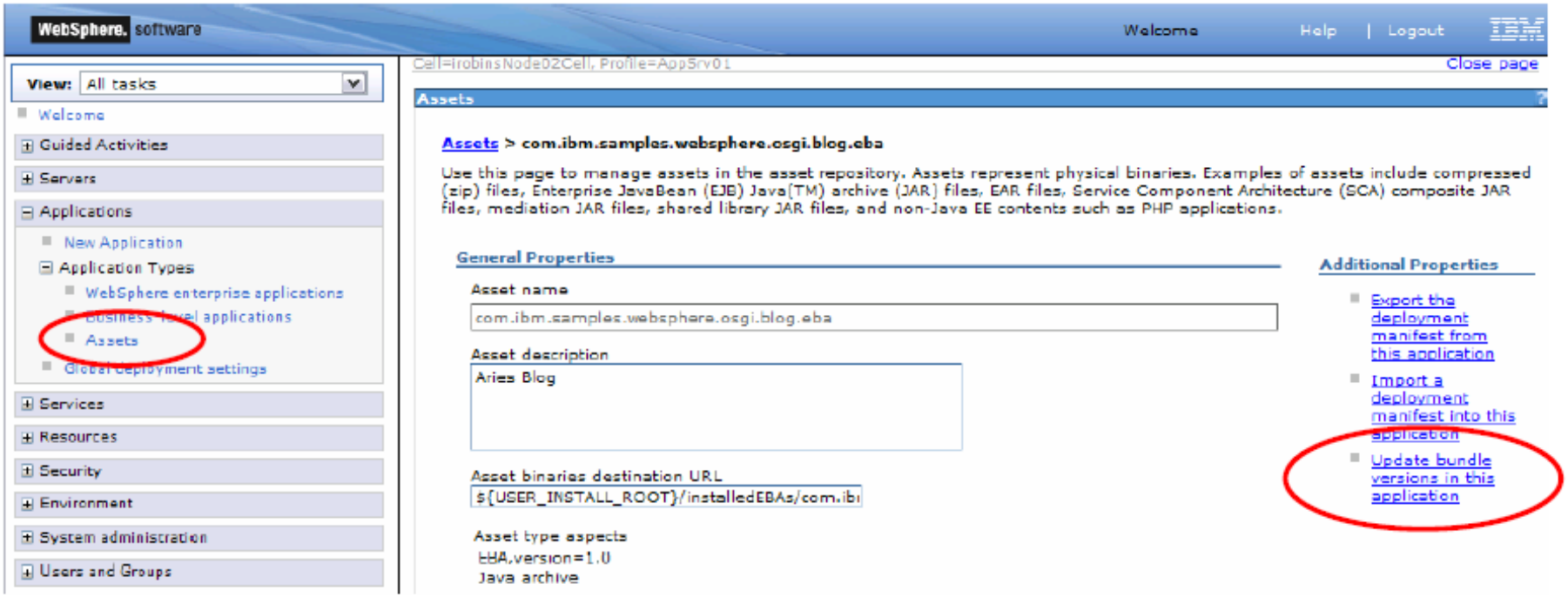
Repeatable, consistent behavior

Install (administrative console or wsadmin):



- The "Deployment manifest" is the concrete resolution of an application. It ensures the application behaves in a consistent, repeatable fashion.

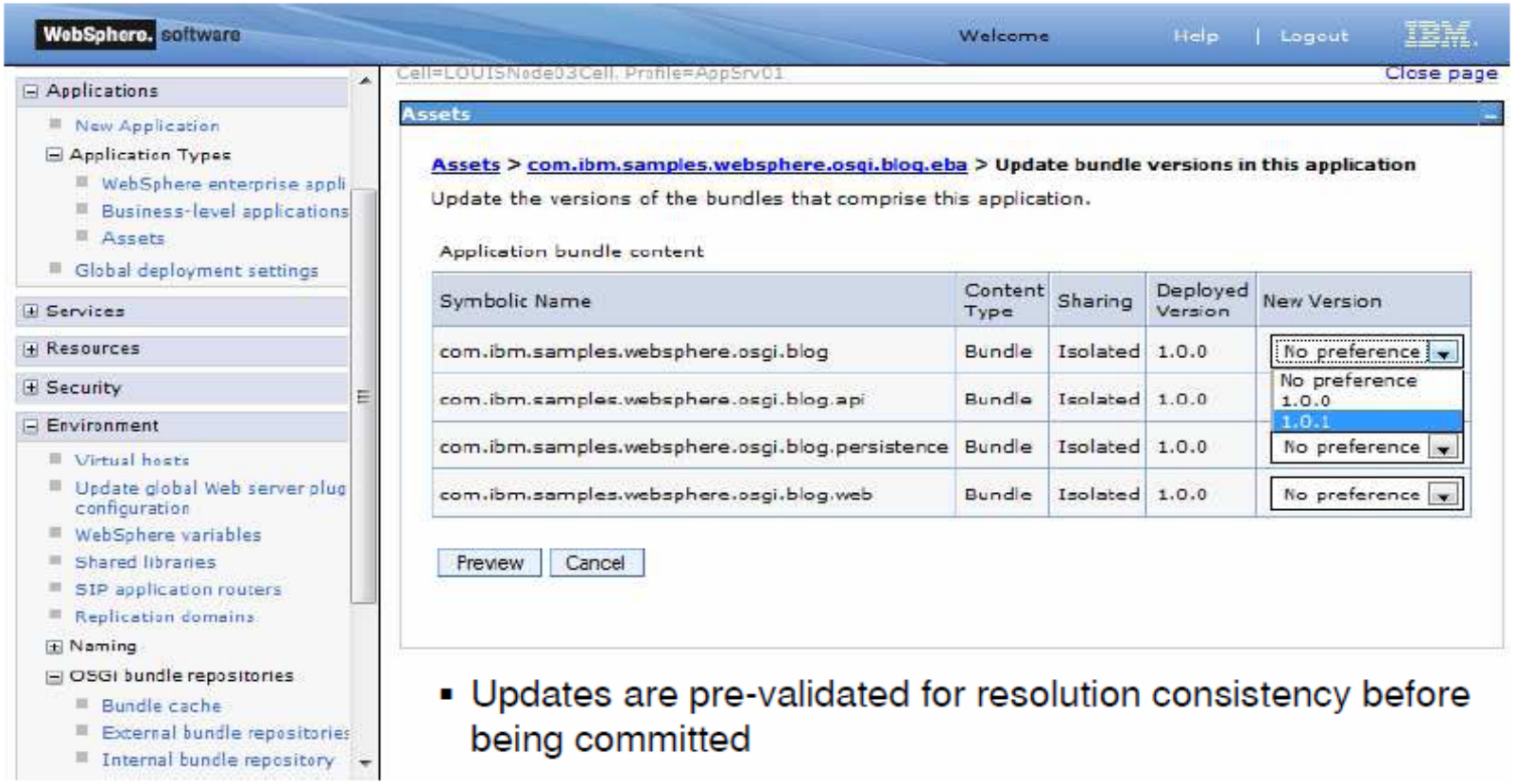
Application-centric bundle management (1 of 2)



- EBA Archives are installed as Application Assets in WebSphere Application Server systems management

Application-centric bundle management (2 of 2)

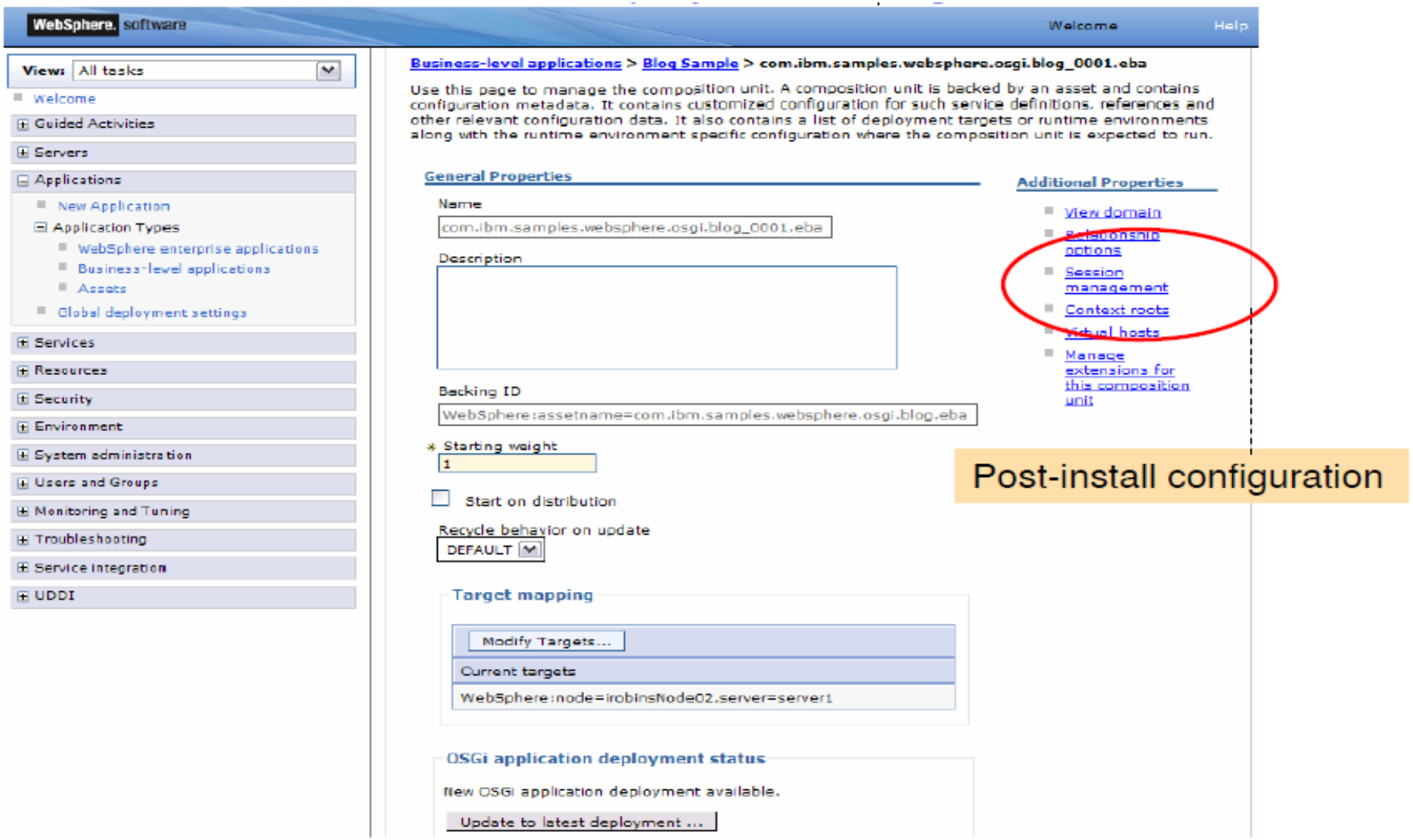
Application-centric bundle management (2 of 2)



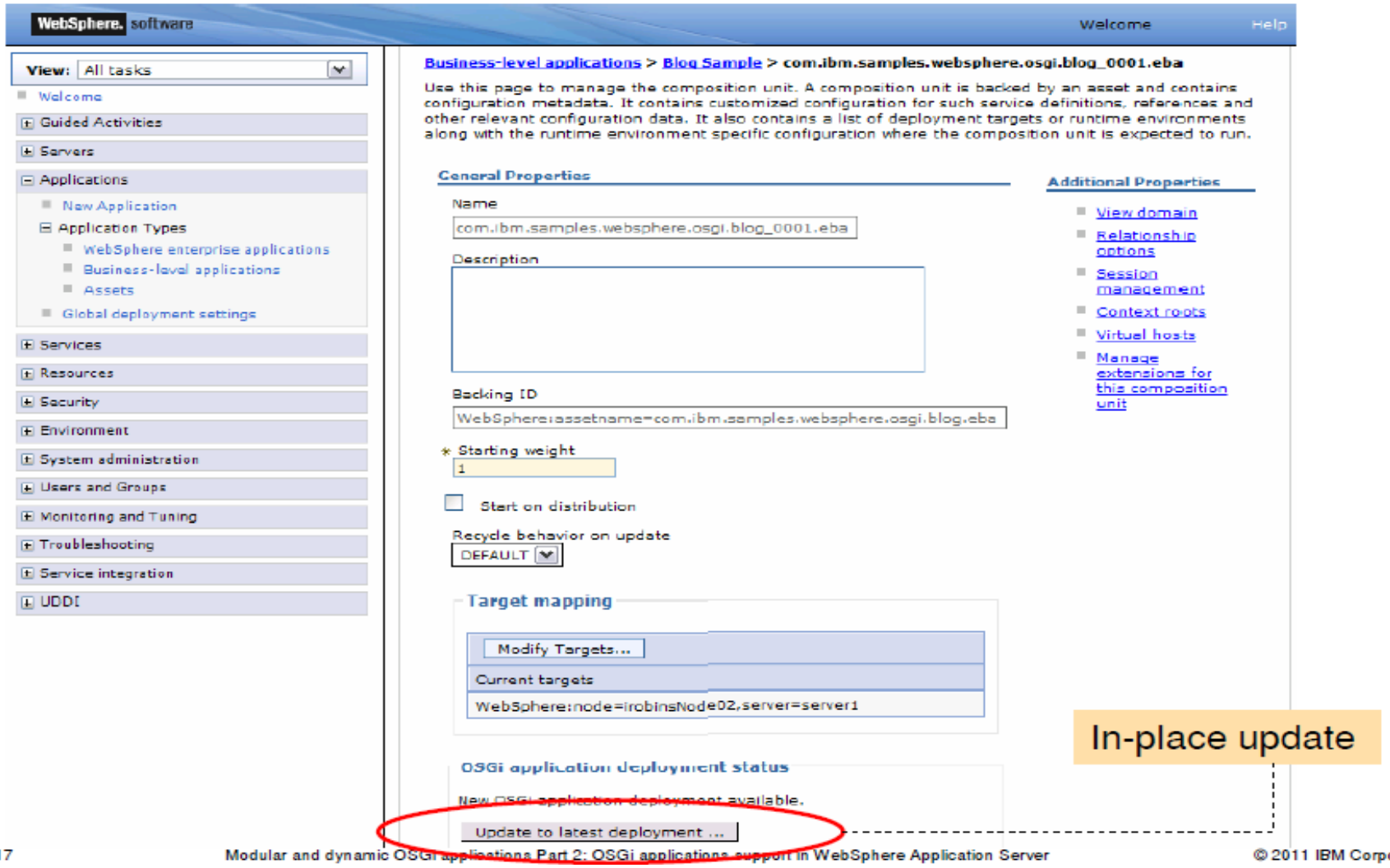
- Updates are pre-validated for resolution consistency before being committed
- Only the modified bundles are installed
- In V7, the update takes effect after the application is restarted

Enhancements in WAS V8

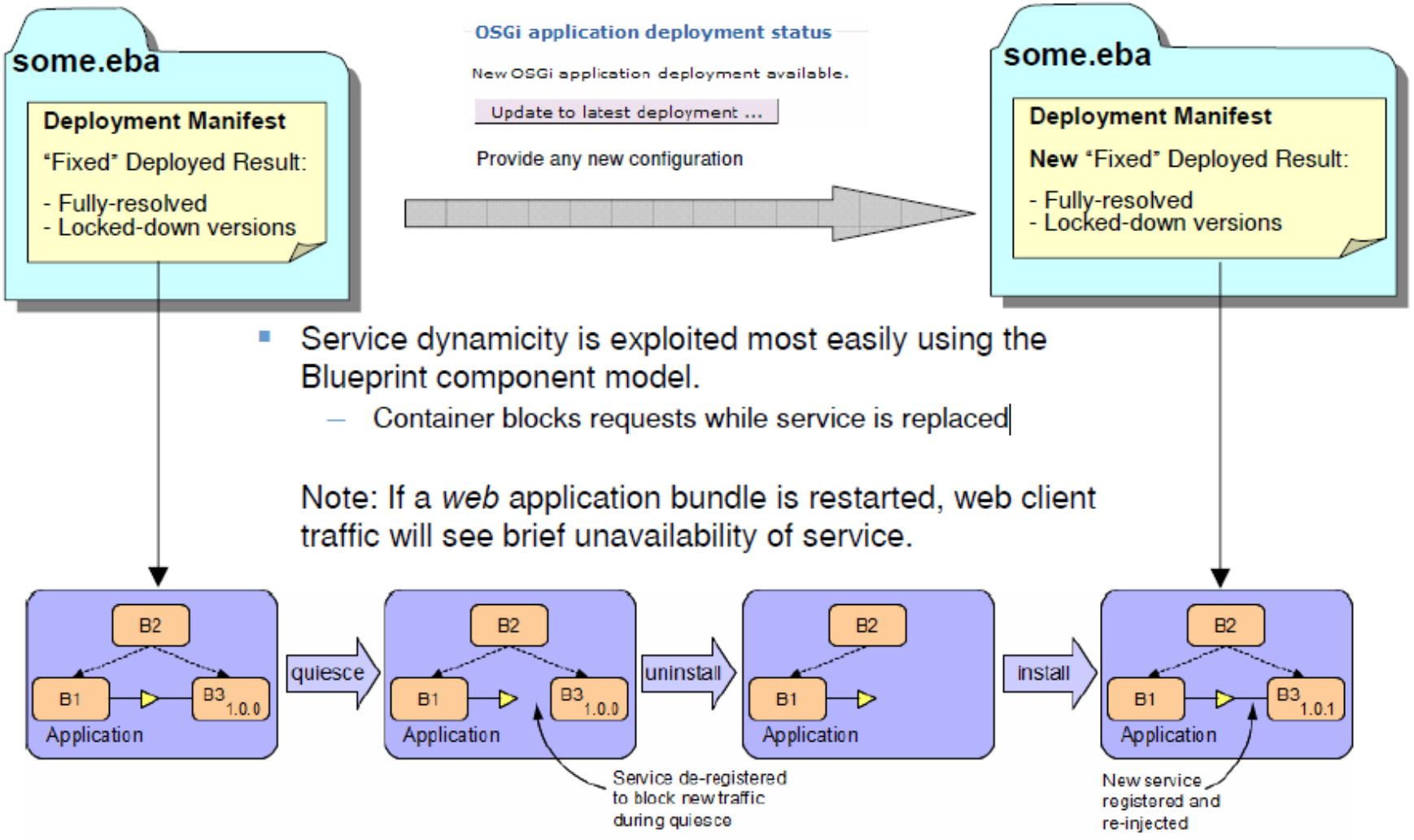
Post deployment configuration



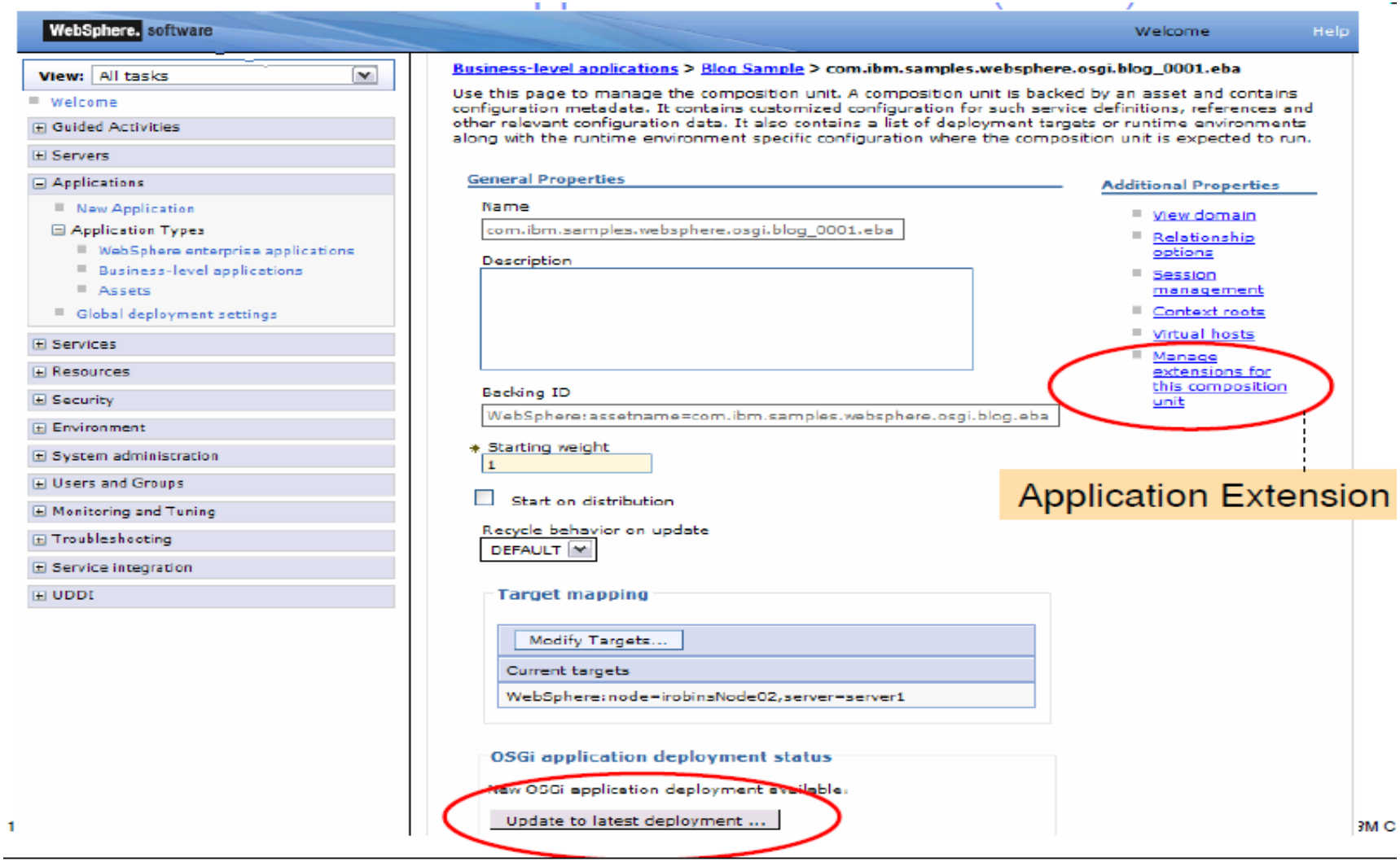
In-place update (1 of 2)



In-place update (2 of 2)

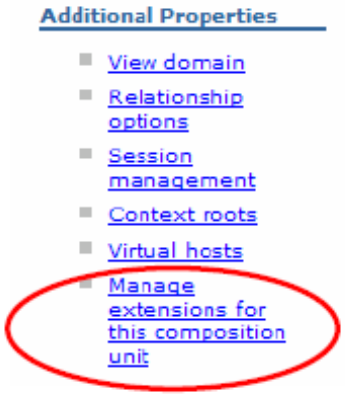


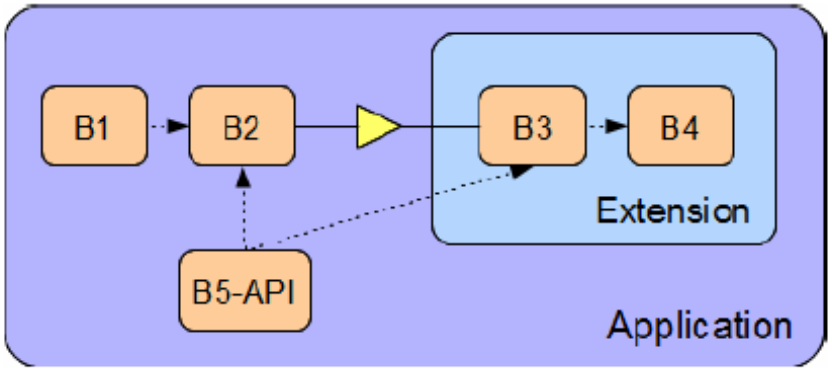
Application extensions



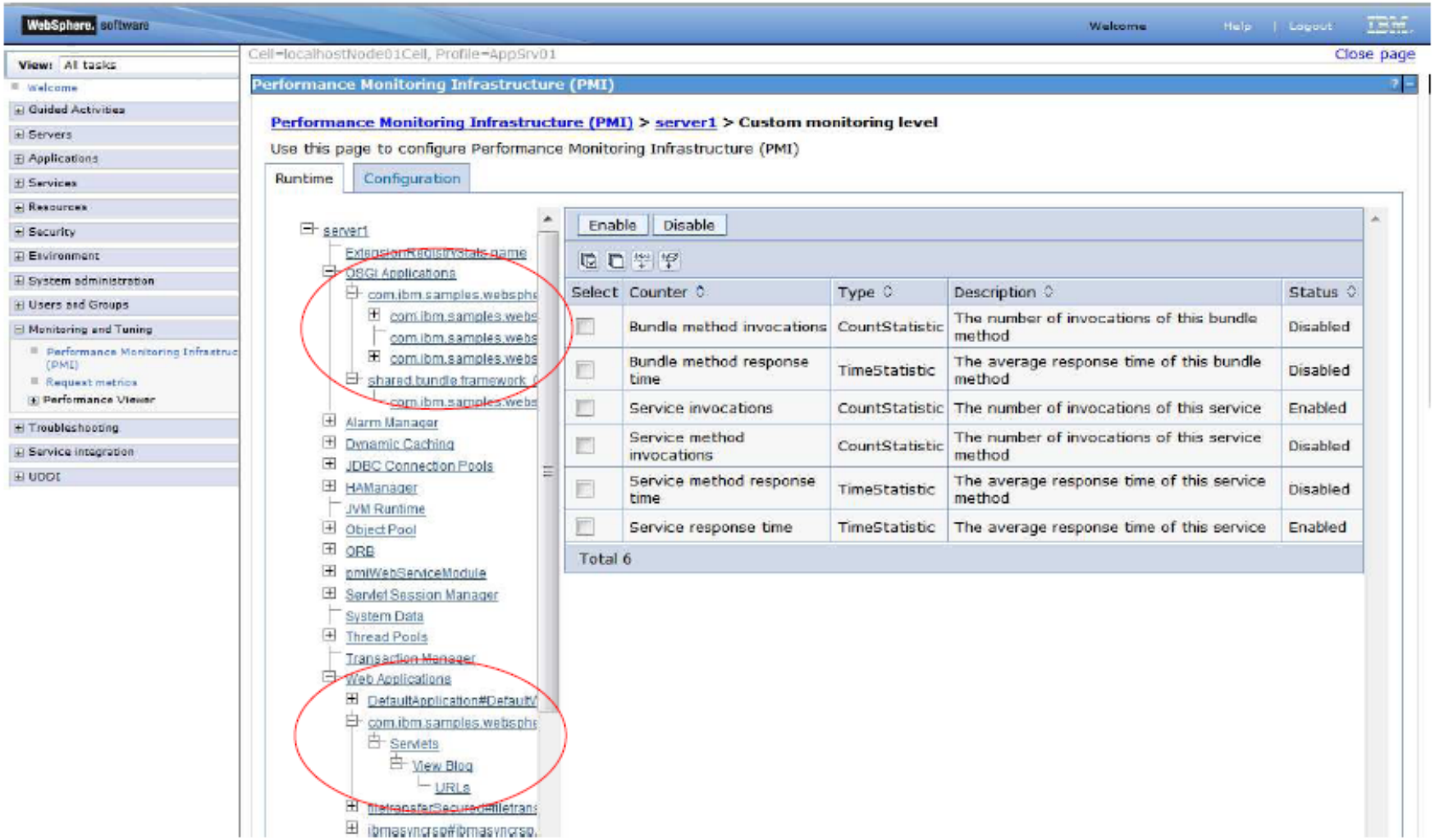
Administering application extensions

- An Extension is a composite bundle archive (CBA) containing one or more bundle. WebSphere Application Server administrator steps:
  - Install Extensions (.cba) in Internal Bundle Repository
  - Add Extensions to Application through “Manage Extensions”
  - When ready, update to latest deployment
    - Providing any extension configuration
  - Well designed extensions cause zero down-time
  - Golden design for extension:
    - Import packages from application
    - Export services to application
  - An OSGi equivalent to an Eclipse Extension

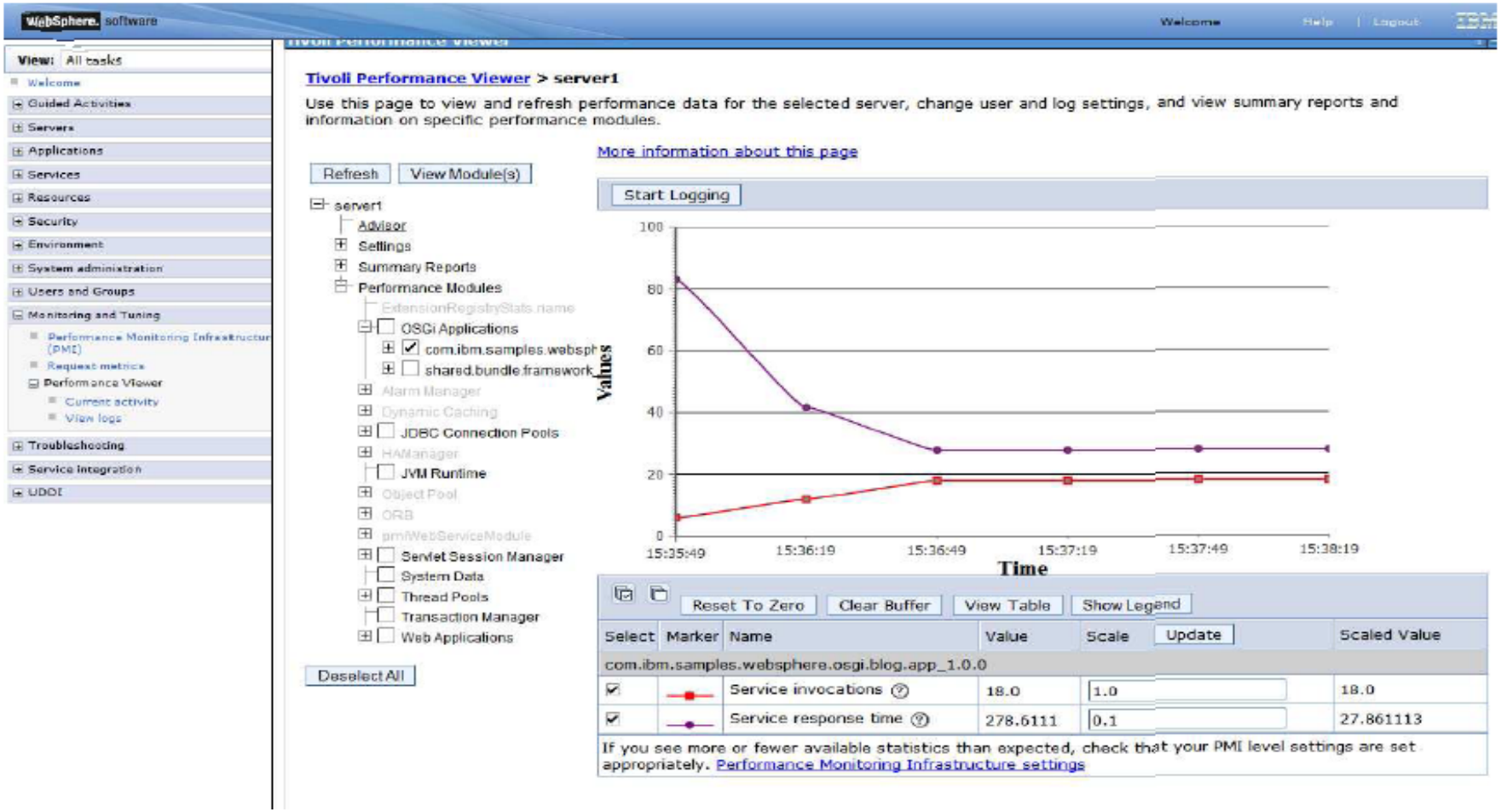




Performance monitoring infrastructure (PMI) (1 of 2)



Performance monitoring infrastructure (PMI) (2 of 2)



The performance viewer can be used to visualize the performance data for OSGi applications.

Other enhancements in V8

- Java EE 6 content in OSGi applications
  - For example, annotated servlet 3.0 components in web application bundles
- Administrative application Asset export/import is "deep" – includes all extended content
  - Retains history of extensions
  - Simplifies application life cycle management (for example, QA→Production)
- CBAs can be included as part of Application-Content
- CBAs can contain WABs
- Batch upload to Internal Bundle Repository
- Administrative Bundle Cache management
- Web component access to Blueprint *components* through JNDI or @Resource injection
  - Blueprint Service references insulate web components from OSGi dynamic service life cycle
  - OSGi *services* accessible through JNDI since V7

Build Modular OSGi Enterprise Applications

Converting Java EE Applications into OSGi Applications

- **Converting an enterprise application to an OSGi application**
  1. Rename the EAR file (.ear) to an EBA file (.eba)
  2. In WebSphere Application Server, import the EBA file as an asset
  3. Optional: Modify any migrated JavaTM 2 security settings

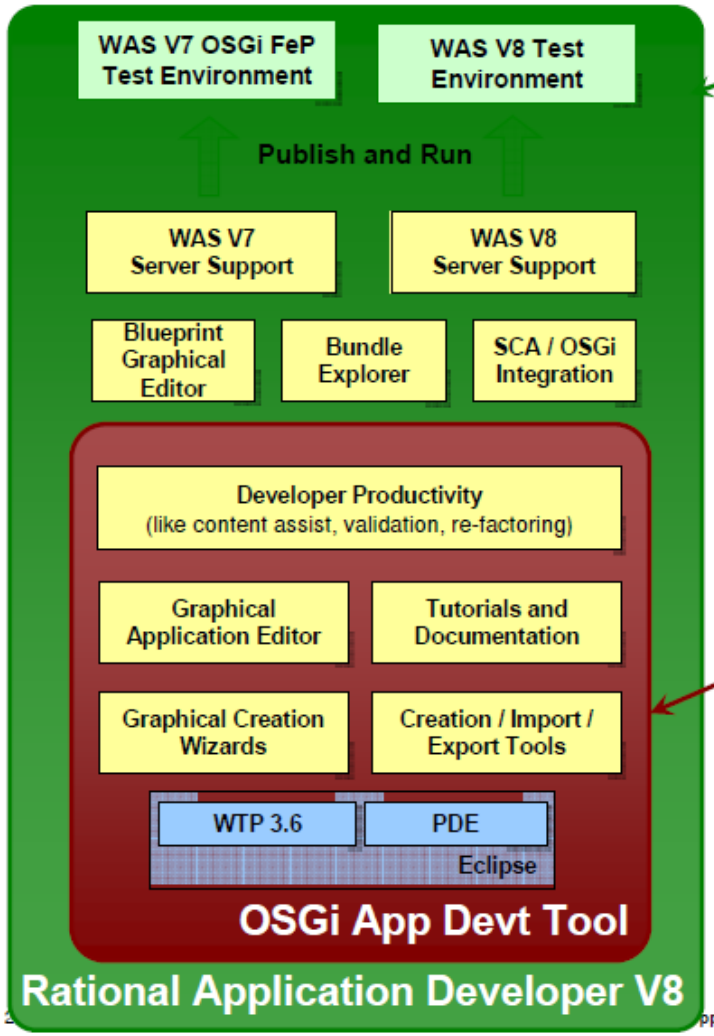
     Any existing was.policy file is converted into a permissions.perm file to be used with the OSGi permissions framework and all permissions are promoted to the application level. If finer granularity is needed, then you can modify the file after conversion.

Note:

- The WAR files are automatically converted to web application bundles (WABs).
- Any utility JAR or EJB JAR files can not be automatic converted
- Any was.policy file is converted to a permissions.perm file
- EJB function is not directly accessible in the EJB container

- **Converting a Spring application to an OSGi application**
  - After automatic conversion, the application is an EBA but will still use the Spring Framework instead of the OSGi Framework
  - Identify the Spring Framework components and replace them with the equivalent code to make them plain JavaTM objects (POJOs), then modify the XML files so that the OSGi framework manages the objects. This involves:
    - ✓ Removing any Spring Framework API calls
    - ✓ Converting Spring configuration XML into standard OSGi Blueprint XML

OSGi application development tools



OSGi application support in Rational Application Developer V8

- Provide integrated development and test of OSGi Applications on the WebSphere platform

- Integrated with Web Tools, JEE productivity tools, and other capabilities in Rational Application Developer
- Supports deployment to WebSphere Application Server V7 OSGi FeP and WebSphere Application Server V8
- Includes the FeP and the WebSphere Application Server V8 Test Environments
- SCA support for OSGi Applications
- Additional OSGi tools:
  - Graphical and wiring editor for Blueprint
  - Bundle Explorer
  - Tools for WebSphere Application Server OSGi extensions

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

Eclipse Plug-in 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) required

http://marketplace.eclipse.org/content/ibm-rational-development-tools-osgi-applications

pplications support in WebSphere Application Server

Summary

Summary

- OSGi is a mature modularity system for Java that has been used *inside* tools and runtime infrastructure for many years
- WebSphere Application Server OSGi Application support enables OSGi technology to be used by enterprise *applications*
- For Developers: new tools and techniques that enable and encourage development of modular applications
  - Explicit dependency management from development projects through to runtime bundles
    - Module relationships BY DESIGN rather than BY OPPORTUNITY
  - Simpler class path and visibility rules
  - POJO development and container-managed dynamic services through Blueprint DI
  - SCA Assembly into SOA components
- For Operations: enhanced modular deployment process, exploiting OSGi metadata
  - Integrated bundle repository to simplify deployment and management of common application infrastructure, simplifying module sharing and version handling
  - Mature, standard mechanism for managing multiple versions of jars (bundles) concurrently
  - In-place, dynamic application extension and update
  - Repeatable, consistent behavior that can progressed from QA to Production

Additional References

- Intro Paper on WebSphere OSGi Application Feature
  www.ibm.com/developerworks/websphere/techjournal/1007_robinson/1007_robinson.html
- OSGi Best Practices
  www.ibm.com/developerworks/websphere/techjournal/1007_charters/1007_charters.html
- Enterprise OSGi YouTube Channel:
  www.youtube.com/user/EnterpriseOSGi
- Redbook: OSGi Applications and JPA 2.0:
  www.redbooks.ibm.com/redpieces/abstracts/sg247911.html?Open
- WebSphere Discussion forum for OSGi Applications:
  www.ibm.com/developerworks/forums/forum.jspa?forumID=1928
- WAS V8
  www.ibm.com/software/webservers/appserv/was/
- WAS V7 Feature Pack for OSGi Applications
  www.ibm.com/websphere/was/osgi
- Rational Application Developer V8.0
  www.ibm.com/software/awdtools/developer/application/index.html

Q&A

Thank You!

Panel Discussion

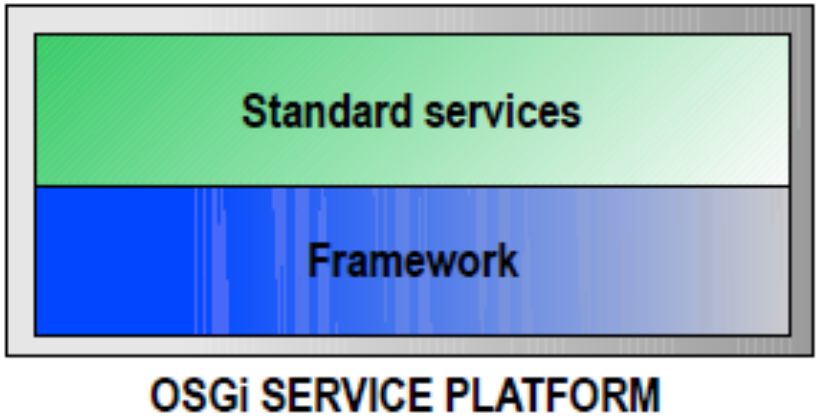
Panel Discussion

- OSGi was supported in the feature pack for OSGi Applications and Java Persistence API 2.0 in WAS v7, and then officially integrated into WAS v8. What's the current usage information for the customer.
- Is there any specific requirement by different industries when using OSGi
- How long it will take the developer to accept OSGi? Is the learning cost very expensive?
- What kind of difficulty the customer will have when using OSGi in WAS?
- What kind of help about OSGi the customer will expect from IBM? (info center, demo…)

Backup

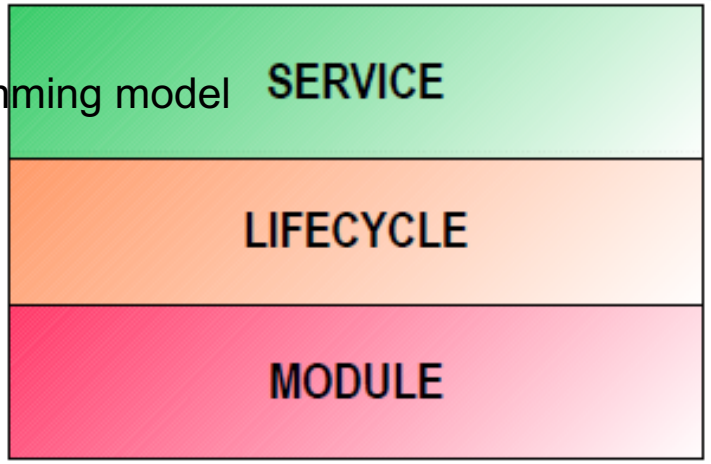
A quick OSGi overview (1)

- OSGi Service Platform is composed of two parts: the OSGi framework and OSGi standard services
  - The framework is the runtime that implements and provides OSGi functionality
  - The standard services define reusable APIs for common tasks,such as Logging and Preferences

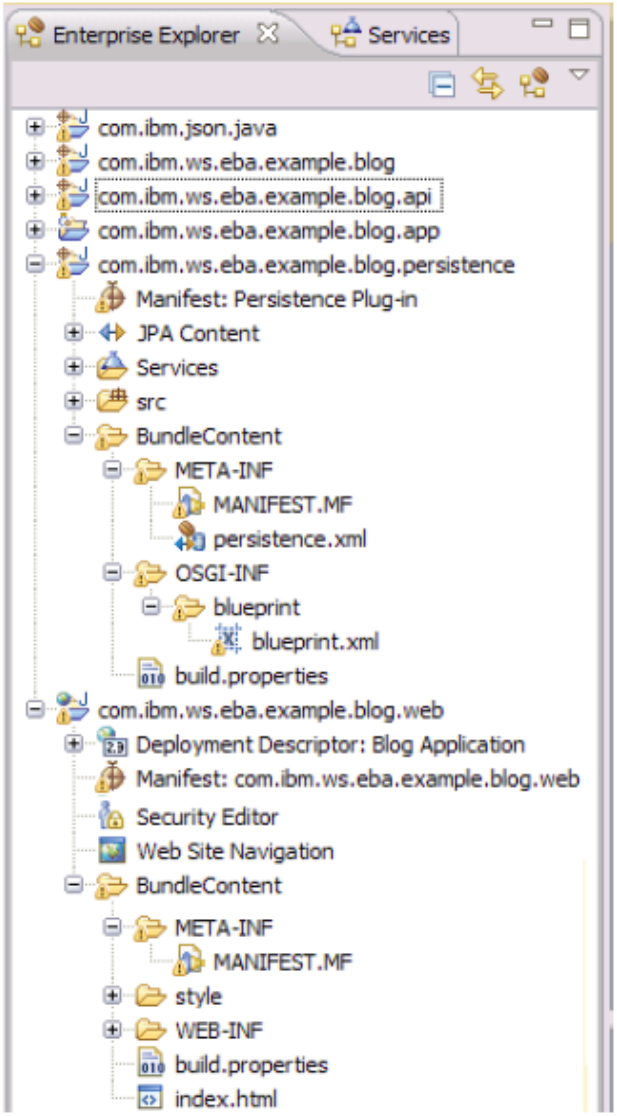


A quick OSGi overview (2)- OSGi Framework

- Module layer – this layer is concerned with packaging and saring code.
  - Defines the OSGi module concept- bundle, you are able to explicitly declare which contained packages are externally visible
  - You are also able to explicitly declare on which external packages your bundle depends
- Lifecycle layer – this layer is concerned with providing run-time module management and access to the underlying OSGi framework.
  - External to your application, it precisely defines the bundle lifecycle operations
  - Internal to your application,it defines how bundles gain access to their execution context
- Service layer – this layer is concerned with interaction and communication among modules, specifically the components contained in them.
  - Supports and promotes a flexible application programming model
  - SOA is largely associated with web services, but OSGi services are local to a single VM
  - Service layer expands the bundle-based dynamism of the lifecycle layer with service-based dynamism

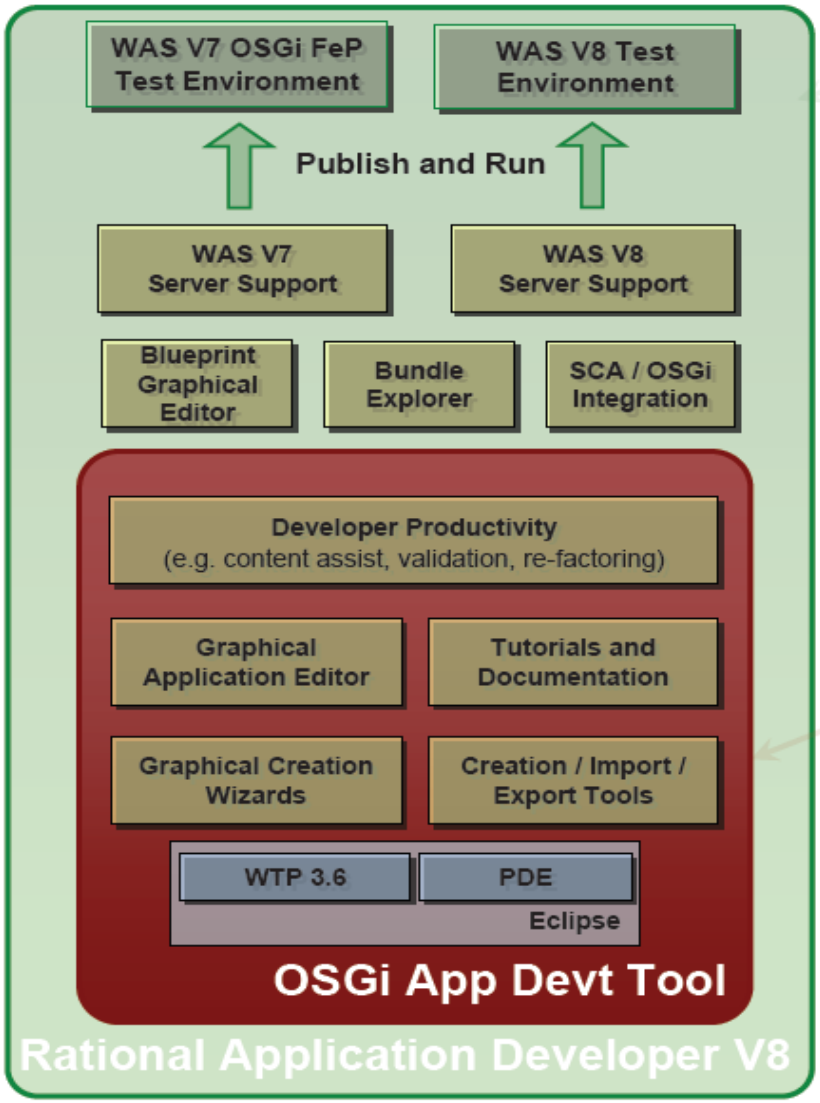


What About Development Tools?



- Need the power of existing tools for Java EE projects with web, persistence facets etc
  - web.xml for web modules
  - persistence.xml for persistence modules
- Combined with OSGi tools for the “bundle” nature
  - MANIFEST.MF
  - blueprint.xml
- Combined with build rules that respect OSGi package visibility and versioning rules
- Combined with the ability to test and debug applications in a specified runtime environment

OSGi Application Development Tools



OSGi Application Support in RAD V8

➢ 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 WAS V8 beta
- Includes the FeP and the WAS V8 beta Test Environments
- SCA support for OSGi Applications
- Additional OSGi tools:
  - Graphical and wiring editor for Blueprint
  - Bundle Explorer
  - Tools for WAS OSGi extensions

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

Eclipse Plugin for OSGi Applications

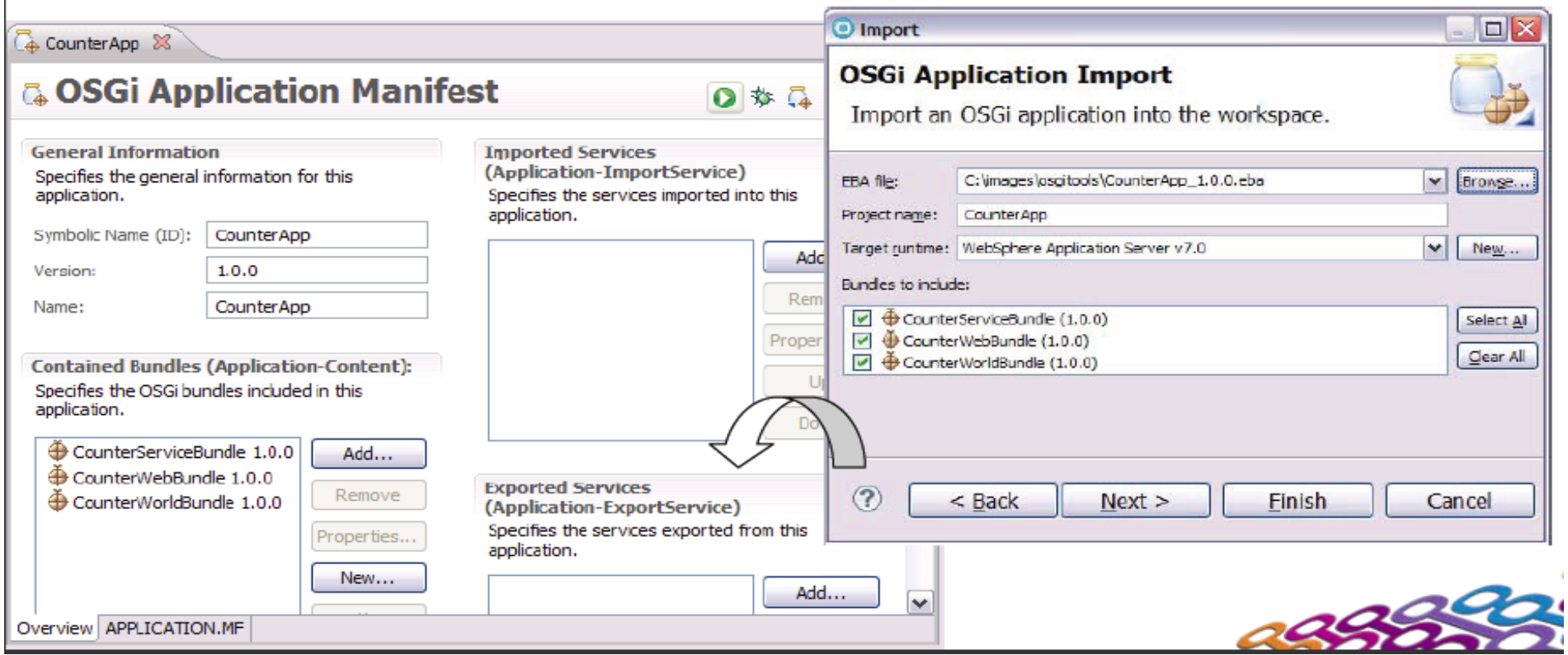
➢Graphical tools to develop OSGi applications and bundles

- Includes features that increase developer productivity
- Creates OSGi Applications for any Aries-based server runtime.
- Eclipse WTP 3.6 (Helios) required

http://marketplace.eclipse.org/content/ibm-rational-development-tools-osgi-applications

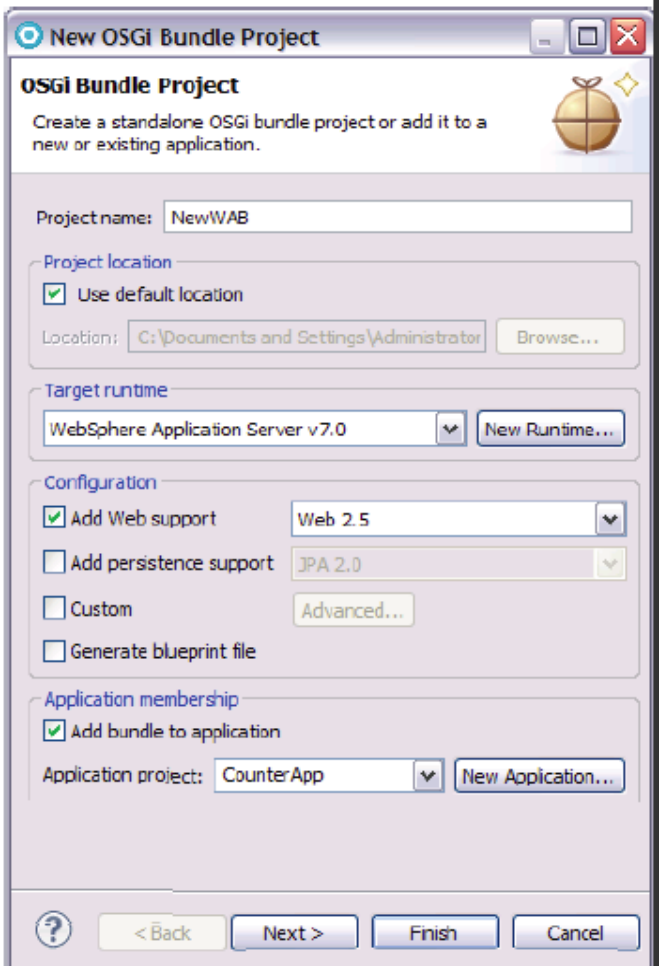
Development Support for OSGi Applications

- OSGi App Dev Tool supports import, creation, and export of applications
- Form-based editor for application manifest
- Validation of manifest syntax and properties



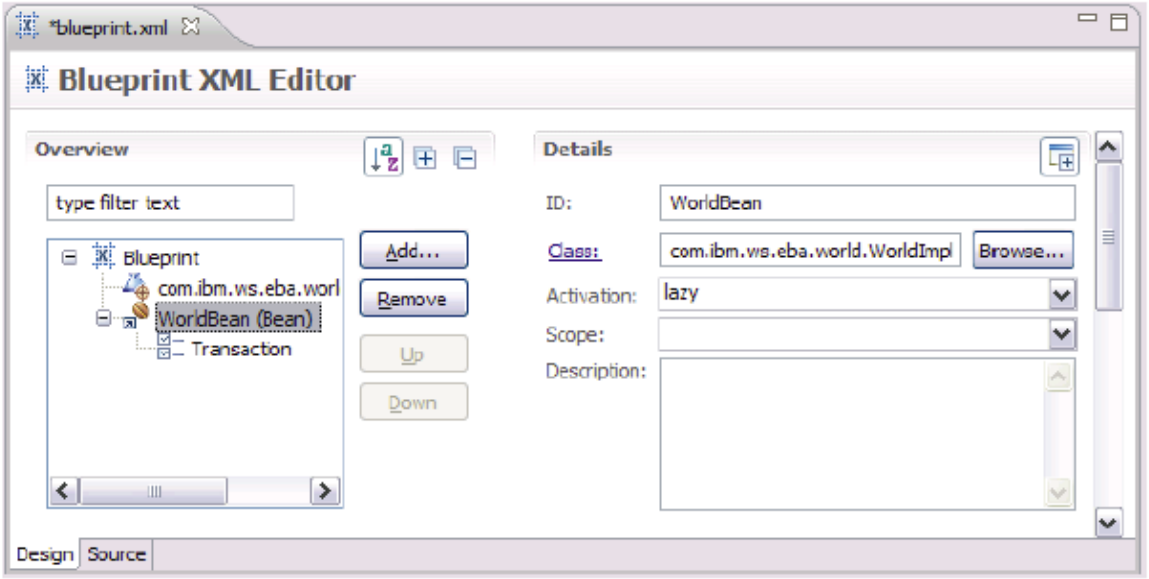
Development Support for Web and JPA Bundles

- Create OSGi bundle projects with Web or JPA configurations
- Can convert existing JEE Web or JPA projects into bundles:
  - Creates or updates a valid bundle manifest
  - Adds a Web-ContextPath or Meta-Persistence hdr
  - Adds package imports based on current module contents
- Graphical manifest editor for OSGi metadata. Project build paths respect OSGi visibility rules
  - Supports and enforces modular characteristics from design through to execution
- RAD's existing comprehensive support for Web and JPA can continue to be used as-is in OSGi projects, including:
  - New wizards (Servlet, JSP, etc.)
  - Page designer and other web editors
  - Persistence.xml editor



Development Support for Blueprint Bundles

- Blueprint creation wizard
  - Supports extension schema including JPA and transactions
- A blueprint editor, including:
  - Source page with syntax highlighting, content assist
  - Form-based editing similar to Java EE deployment descriptors
  - Syntax and reference validation
  - Hyperlinking to impl classes
  - Refactoring support



Bundle Explorer