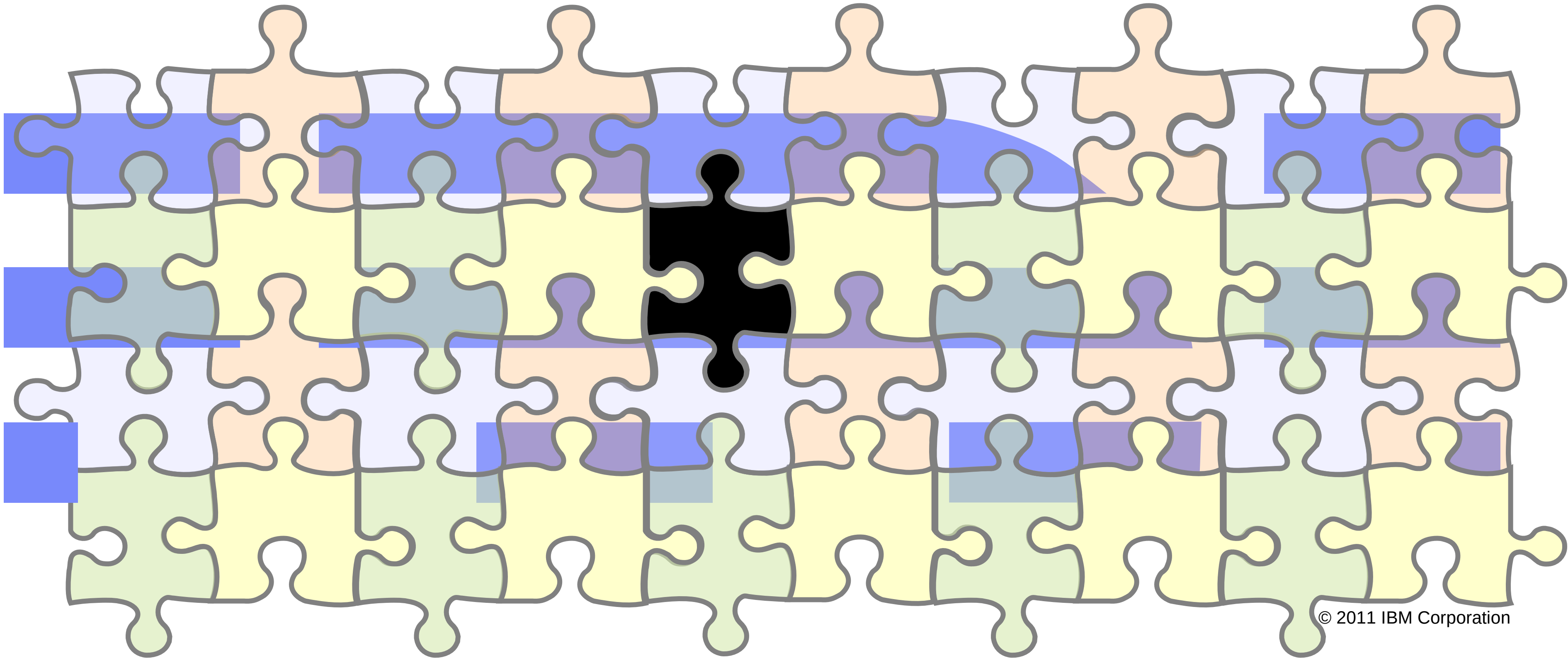
OSGi – The Missing Piece Of The Jigsaw

Ian Robinson, IBM Distinguished Engineer



It Seems Like Only Yesterday….

- JSR 8 – Open Services Gateway specification (1999)
  - A gateway for providing services to devices the home.
  - *cradle-to-grave life cycle management of services, inter-service dependencies, installing, versioning, and configuring the services*

- OSGi Alliance, formed in 1999, took over the specification work and delivered V1.0 in May 2000.

  http://www.osgi.org

The First 12 Years…

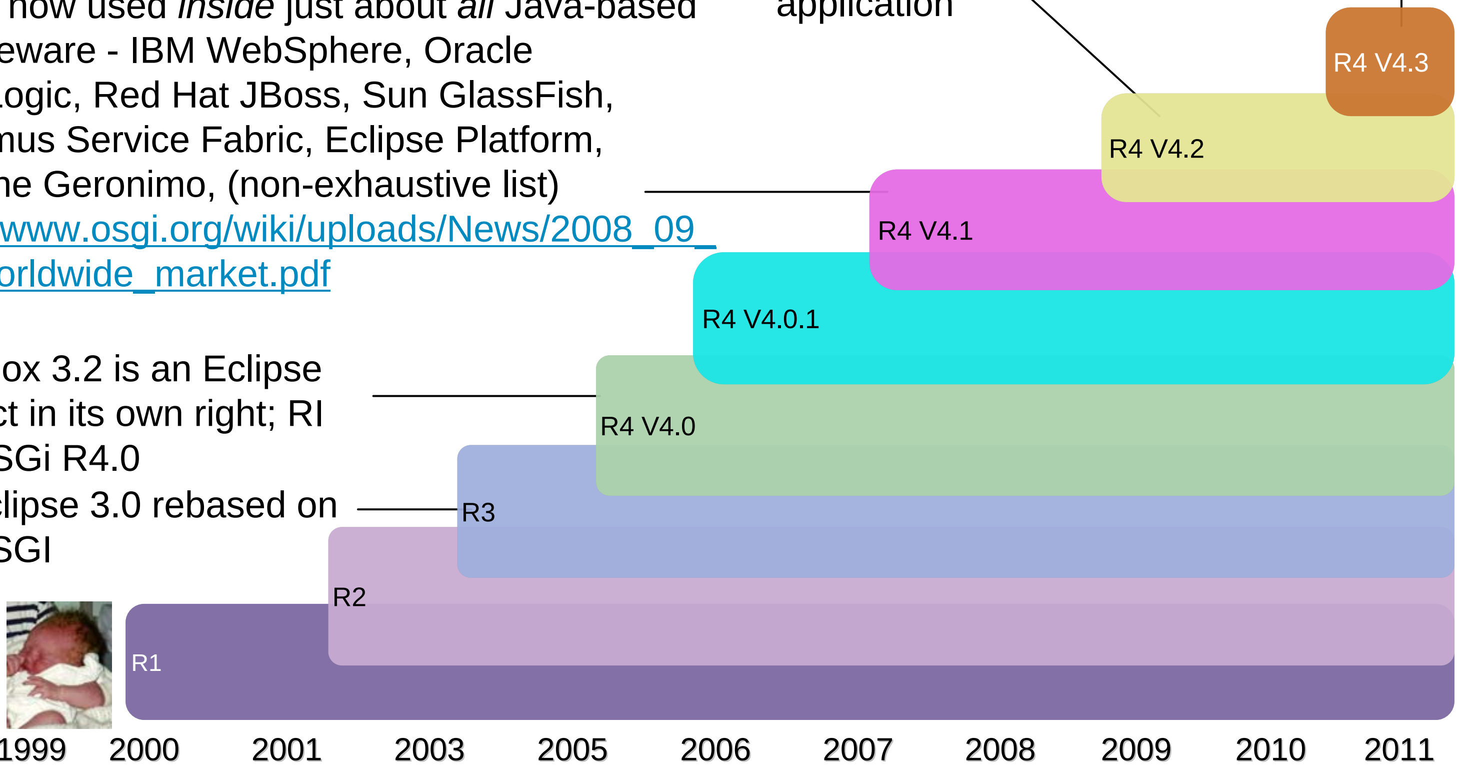
Imminent: framework hooks for multi-bundle scopes, byte-code weaving, …

Enterprise focus, enabling OSGi to move out of middleware into the application

OSGi now 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

Equinox 3.2 is an Eclipse project in its own right; RI for OSGi R4.0

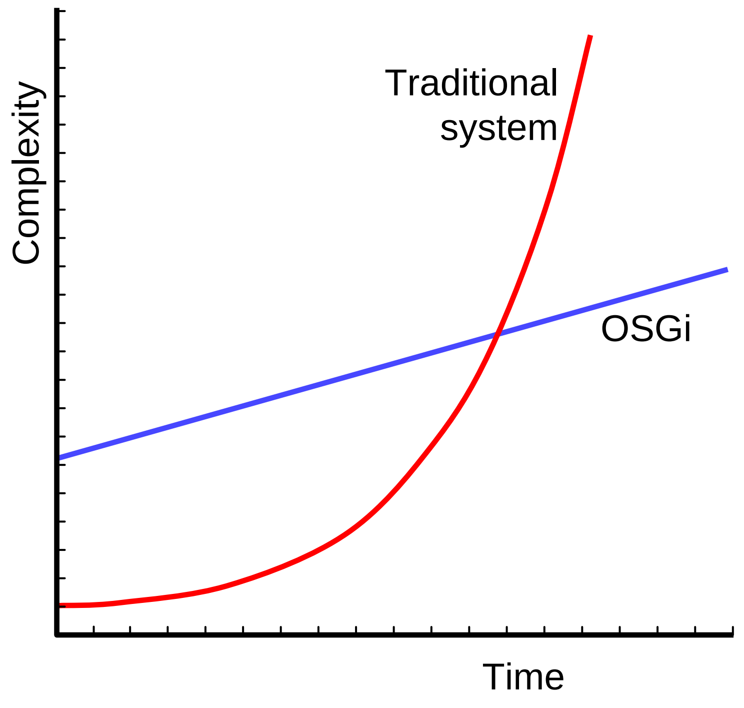
Eclipse 3.0 rebased on OSGI

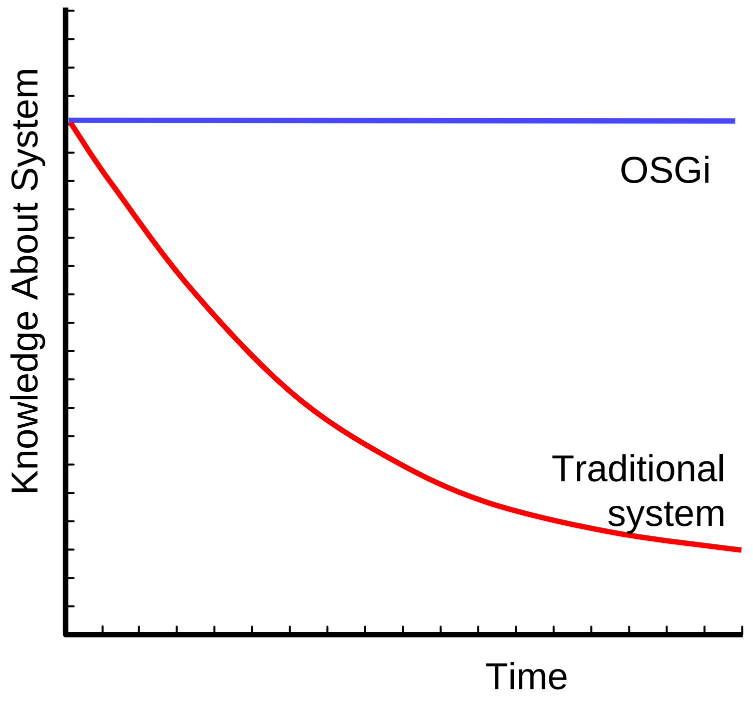


Teenage Kicks

- Crossing the Chasm http://ianskerrett.wordpress.com/2010/04/13/is-osgi-crossing-the-chasm/
- Do we Really Need OSGi? http://www.onstrategies.com/blog/2010/03/23/do-we-really-need-osgi/
- OSGi – Feast or Famine? http://techdistrict.kirkk.com/2010/04/12/osgi-feast-or-famine/
- OSGi is Too Complex http://www.sdtimes.com/FROM_THE_EDITORS_OSGI_IS_TOO_COMPLEX/By_SD_TIMES_EDITORIAL_BOARD/About_OPENSOURCE_and_OSGI/34281

Complexity and Entropy





http://www.tensegrity.hellblazer.com

http://adaptevolve.blogspot.com

Does Everything Decay?

- According to the 2nd Law of Thermodynamics, it does.
- And quantum physics says everything is entangled.

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

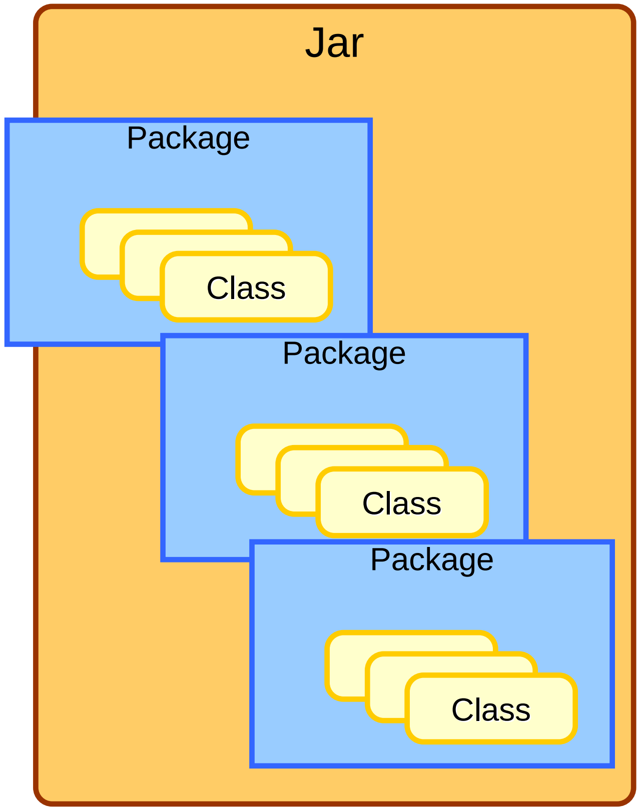
Why does it happen?
How do we prevent it?

#1 – *Java* Needs Help

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

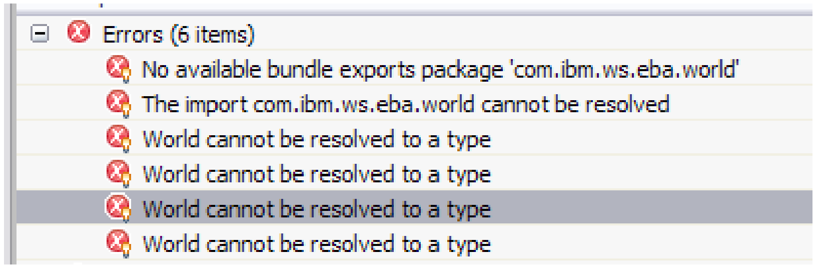
But a JAR lacks the primary characteristics of modularity:

- It does NOT hide its internals
- It does NOT declare its externals
- The global Java classpath does NOT support versioning



#2 - Java *Developers* Need Help

- Patterns like SOA and DI are helpful but Java EE tools have provided limited assistance for enabling the development of truly re-usable software modules.
- We need more that just a set of patterns and good design practice to produce re-usable software.
- Developers need tools to encourage and enforce modular characteristics during development.
  - Its not enough that the code compiles cleanly.
  - Need dev-time warnings and quick fixes to indicate when code is getting entangled.

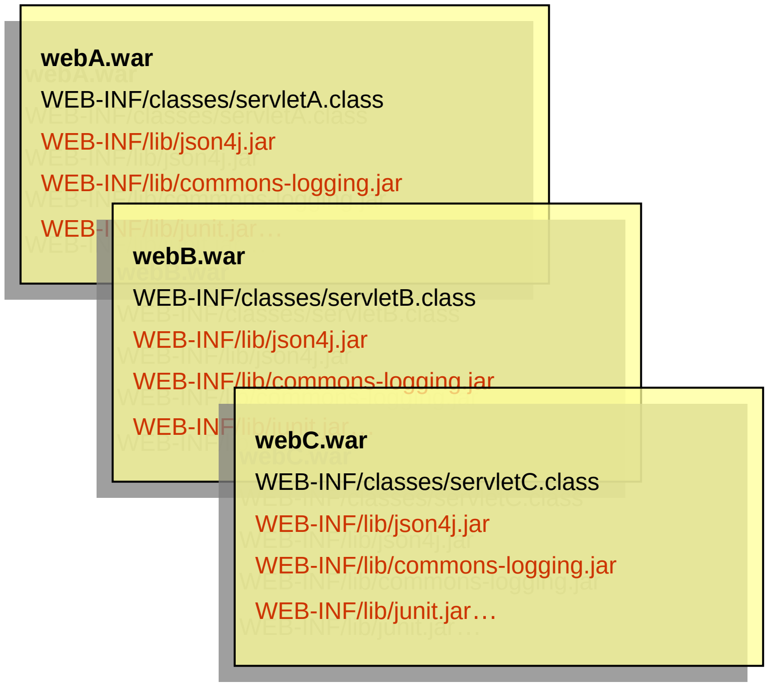


```
⊟ Errors (6 items)
    No available bundle exports package 'com.ibm.ws.eba.world'
    The import com.ibm.ws.eba.world cannot be resolved
    World cannot be resolved to a type
    World cannot be resolved to a type
    World cannot be resolved to a type
    World cannot be resolved to a type
```

#3 – Noddy Classpaths and Big EARs

For enterprise applications, no matter how modular the App is, the EAR deployment process is lacking

- *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

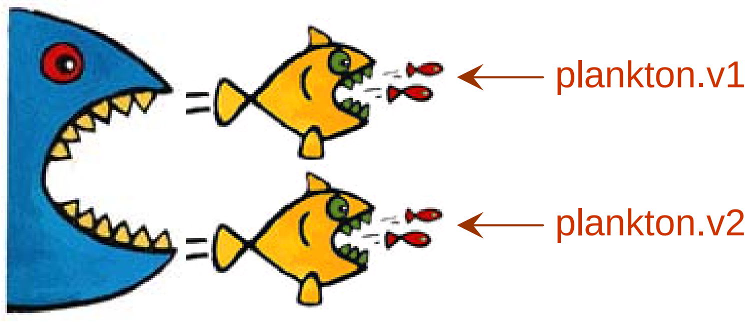


webA.war
WEB-INF/classes/servletA.class
WEB-INF/lib/json4j.jar
WEB-INF/lib/commons-logging.jar
WEB-INF/lib/junit.jar…

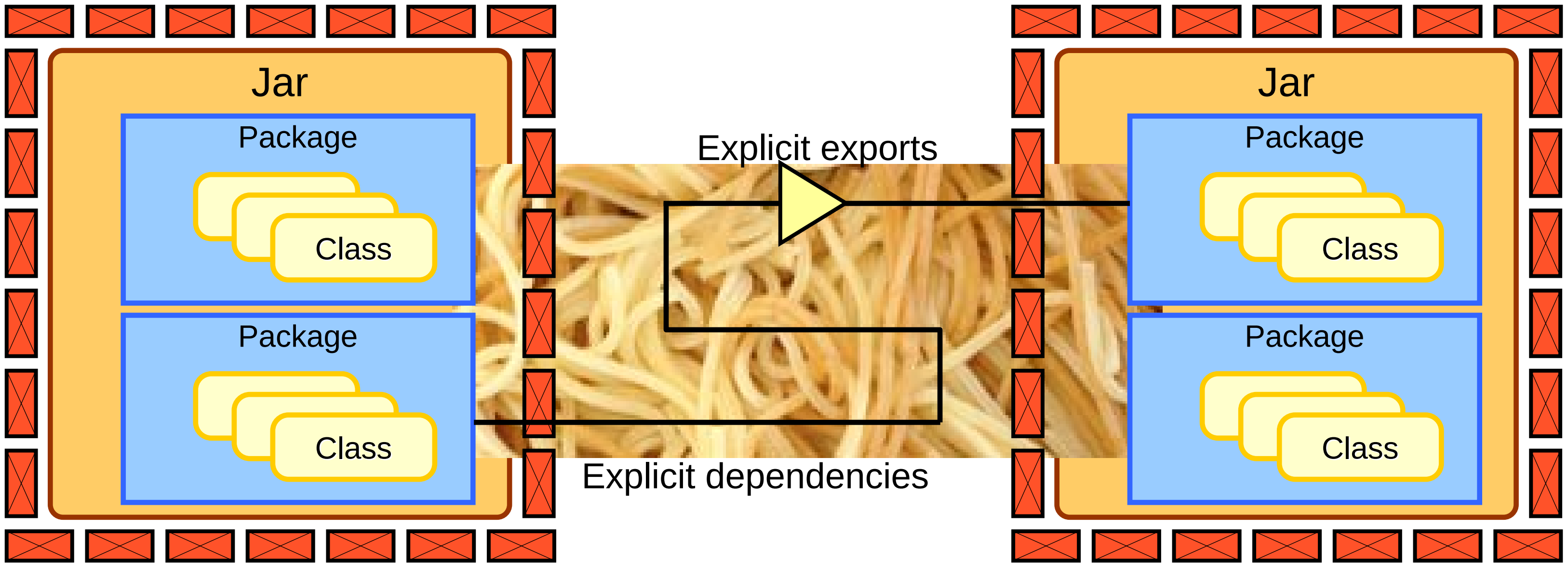
webB.war
WEB-INF/classes/servletB.class
WEB-INF/lib/json4j.jar
WEB-INF/lib/commons-logging.jar
WEB-INF/lib/junit.jar…

webC.war
WEB-INF/classes/servletC.class
WEB-INF/lib/json4j.jar
WEB-INF/lib/commons-logging.jar
WEB-INF/lib/junit.jar…

- *Within apps* - 3rd party libraries consume other 3rd party libraries leading to conflicting versions on the application classpath.

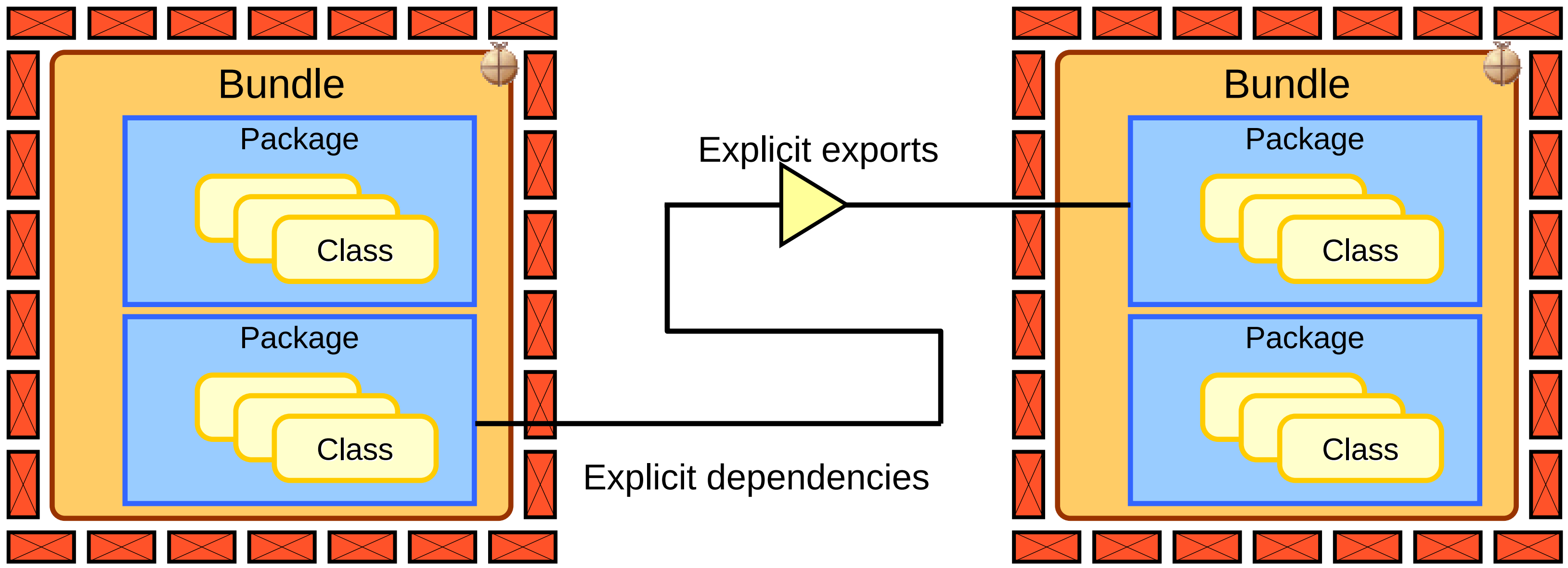


How Does OSGi Stop the Rot?



- JARs become proper modules whose internals are no longer visible and whose dependencies are explicit
- New dependencies are easier to scrutinize because they are made explicit in module metadata.
- Wiring between modules is BY DESIGN rather than BY OPPORTUNITY
- OSGi services enable loose-coupling between modules so that a consumer is insulated from needing to burn-in the concrete provider of a service.

How Does OSGi Reduce Complexity?

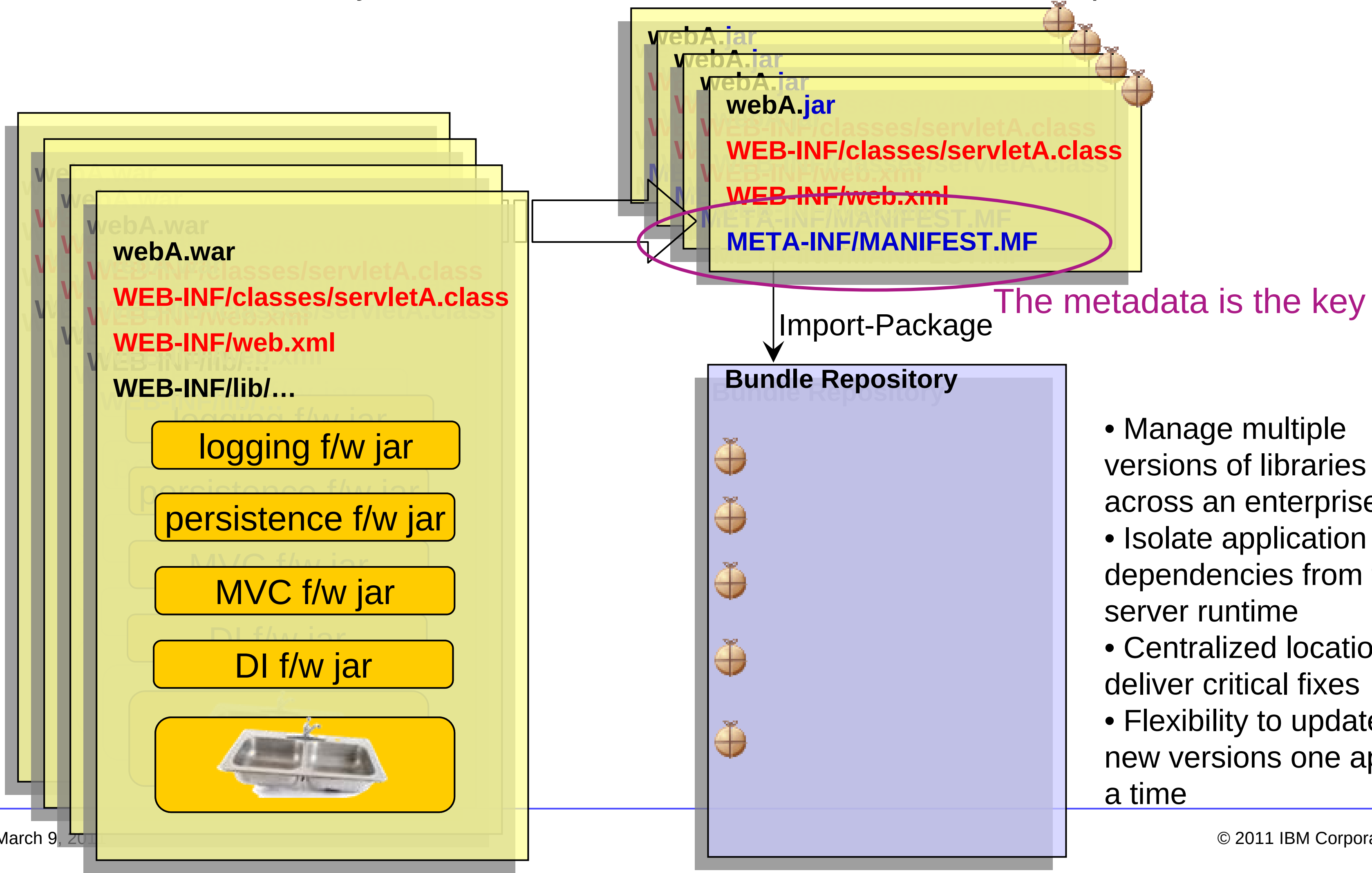


- Enables the dynamic update and extension of modules within a wider system
- The impact of change is more easily quantifiable, making testing easier to scope/target.
- Enables co-existence of multiple versions of libraries
  - Simplifies independent evolution of applications
- Bundle repositories enable modular application deployment

Modular Deployment of Enterprise Applications

If the App can declare its dependencies then these can be provisioned from a central, shared repository.

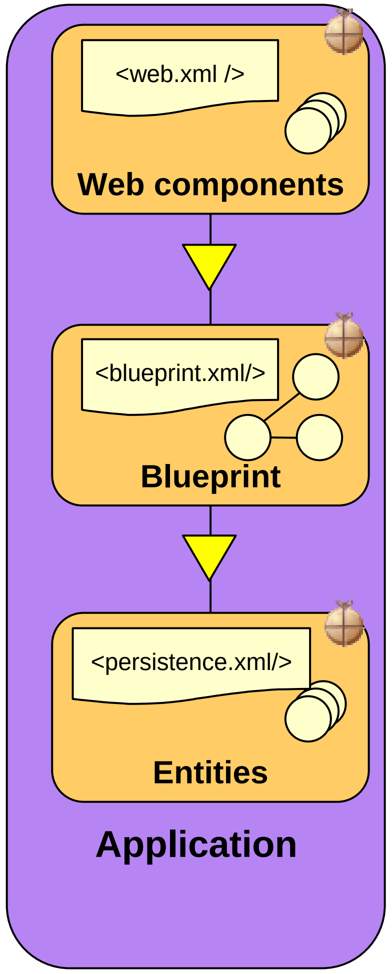
Common bundles easily factored out of the EARs/WARs and used at specific versions



- 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 app at a time

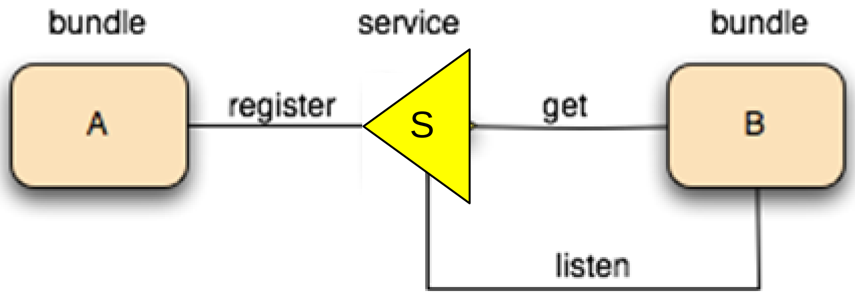
How Do Enterprise Applications Use OSGi?

- OSGi V4.2 has made OSGi much more attractive to applications.
- For the first time, application concerns like web technologies, fine-grained component assembly, access to persistence frameworks etc are addressed
  - through exploitation of familiar Java EE technologies
  - in a manner suitable for a dynamic OSGi environment
- With increasing numbers of development tools and server runtimes to support the development, deployment and management of OSGi **applications.**



What About Dynamic OSGi Services?

- There is no analog for dynamic services in Java EE.



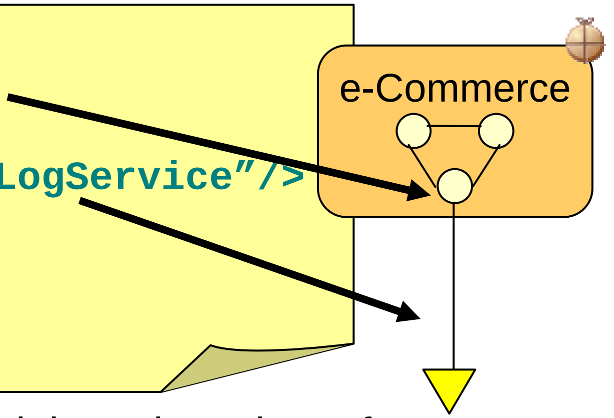
- Services registered in Service Registry.
- Services are dynamic
- Clients (bundle B) need to cope with them going away

Using OSGi Services can be hard

```
private BundleContext ctx;

        private AtomicReference<LogService> ls = new AtomicReference<LogService>();
        private AtomicReference<ServiceReference> lr = new AtomicReference<ServiceReference>();

        public void start(BundleContext ctx) throws InvalidSyntaxException
        {
                this.ctx = ctx;
                ctx.addServiceListener(this, "(objectClass=org.osgi.service.log.LogService)");
                ServiceReference ref = ctx.getServiceReference(LogService.class.getName());
                if (ref != null) {
                        ls.set((LogService) ctx.getService(ref));
                        lr.set(ref);
                }
        }

        @Override
        public void serviceChanged(ServiceEvent event)
        {
                ServiceReference ref = event.getServiceReference();

                if (ls.get() == null && event.getType() == ServiceEvent.REGISTERED) {
                        ls.set((LogService) ctx.getService(ref));
                } else if (ls.get() != null && event.getType() == ServiceEvent.UNREGISTERING &&
                                ref == lr.get()) {
                        ref = ctx.getServiceReference(LogService.class.getName());
                        if (ref != null) {
                                ls.set((LogService) ctx.getService(ref));
                                lr.set(ref);
                        }
                }
        }
```

Blueprint Simplification

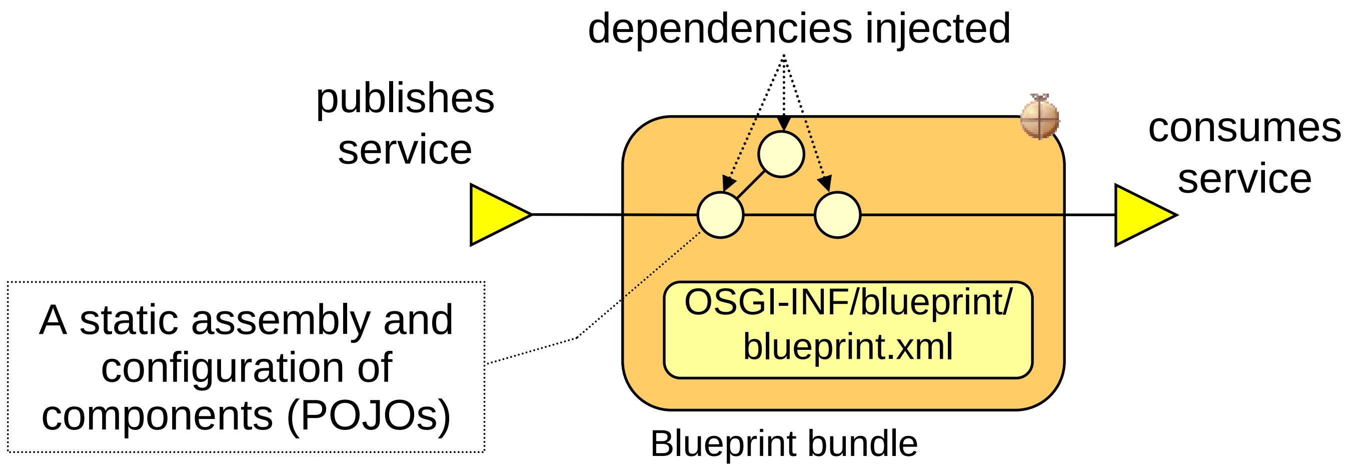
```
<blueprint>
  <bean id="shop" class="org.example.ecomm.Bean">
    <property name="log">
      <reference interface="org.osgi.service.log.LogService"/>
    </property>
  </bean>
</blueprint>
```



```
public class Bean {

private LogService log;
void setLog(LogService logService) {
      log = logService;
      }

void process(Order o) {
      log.log(LOG_INFO, "processing: " + o);
      }
}
```

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

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

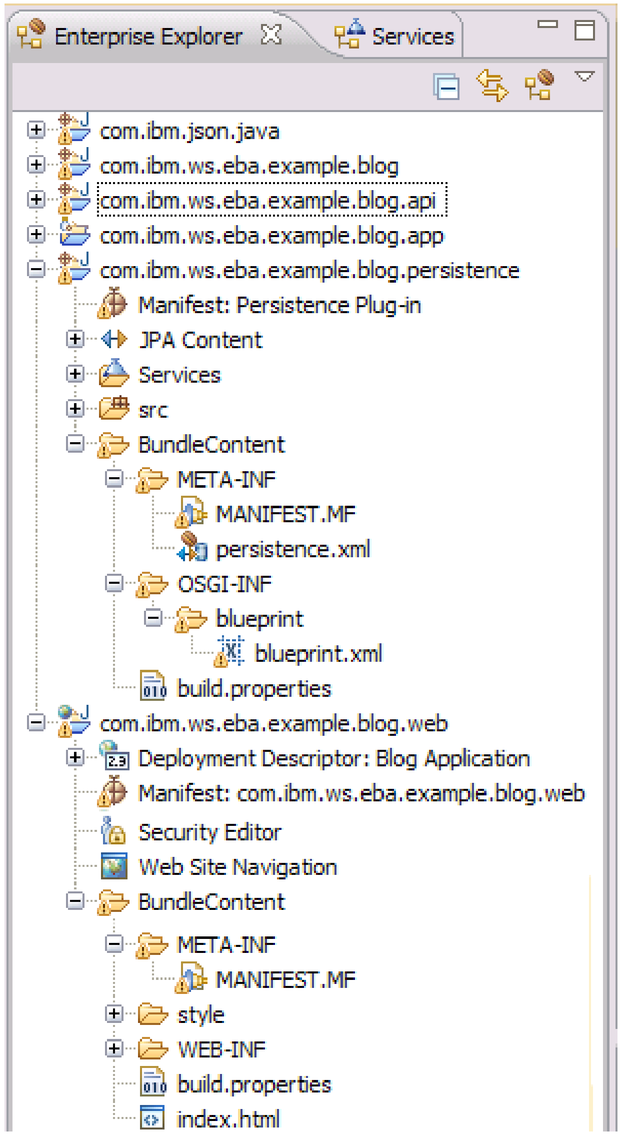
- XML Blueprint bean definition describes component configuration and scope
  - Optionally publish and consume components to/from OSGi service registry.
  - Standardizes established Spring conventions
- Simplifies unit test outside either Java EE or OSGi r/t.

Open Source Activities

- Apache "Aries" Project
  http://aries.apache.org/
  - Provides enterprise OSGi componentry such as the Blueprint container and projects to integrate enterprise technologies such as JTA, JMX, JNDI and JPA into an OSGi environment to make these available to OSGi bundles.
  - Provides an assembly format for multi-bundle applications, for deployment to a variety of OSGi based runtimes.
  - Incubator created Sep 2009. Graduated to Top Level Project in 2010

- Eclipse "Gemini" Enterprise Modules Project
  http://www.eclipse.org/gemini/
  - The Enterprise Modules Project (Gemini) is all about modular implementations of Java EE technology. It provides the ability for users to consume individual modules as needed, without requiring unnecessary additional runtime pieces.
  - Gemini Web sub-project released in 2010; other sub-projects incubating.

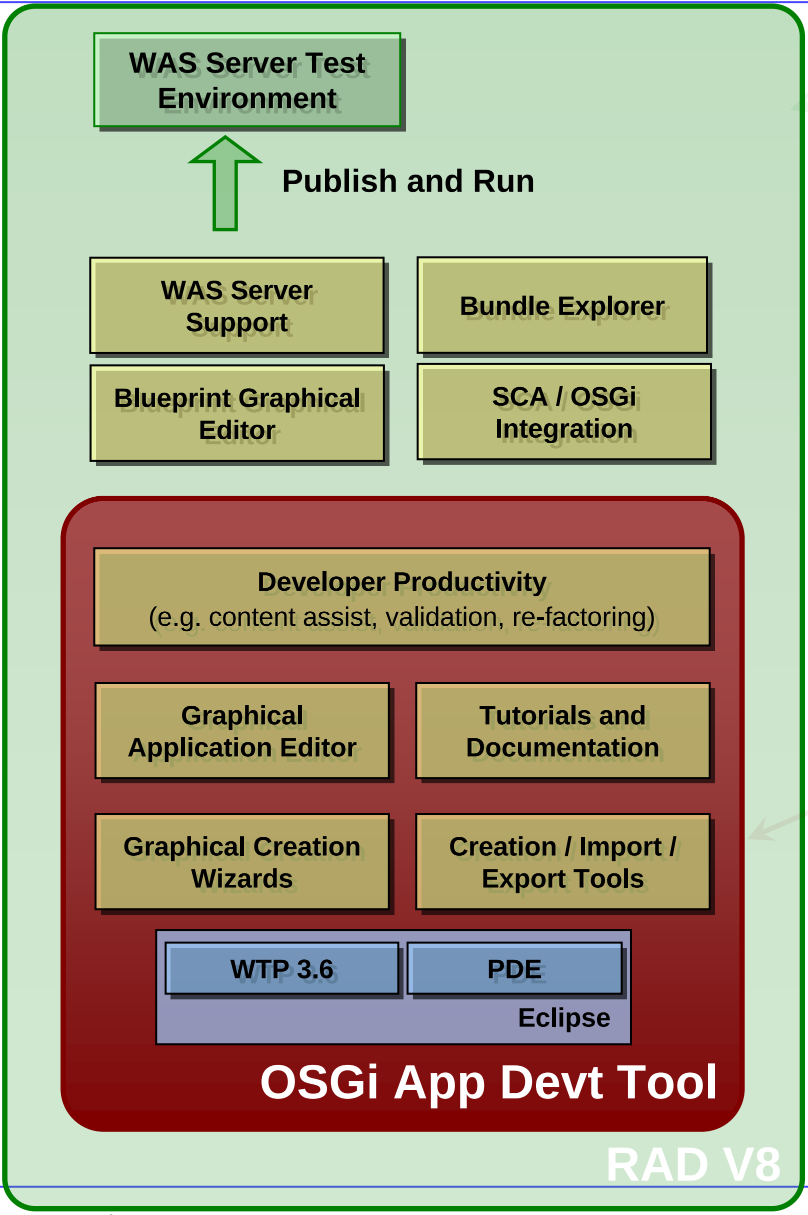
What About Development Tools?



```
+ com.ibm.json.java
+ com.ibm.ws.eba.example.blog
+ com.ibm.ws.eba.example.blog.api
+ com.ibm.ws.eba.example.blog.app
- com.ibm.ws.eba.example.blog.persistence
    Manifest: Persistence Plug-in
  + JPA Content
  + Services
  + src
  - BundleContent
    - META-INF
        MANIFEST.MF
        persistence.xml
    - OSGI-INF
      - blueprint
          blueprint.xml
      build.properties
- com.ibm.ws.eba.example.blog.web
  + Deployment Descriptor: Blog Application
    Manifest: com.ibm.ws.eba.example.blog.web
    Security Editor
    Web Site Navigation
  - BundleContent
    - META-INF
        MANIFEST.MF
    + style
    + WEB-INF
      build.properties
      index.html
```

- Need the power of existing tools for Java 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


WAS Server Test Environment
Publish and Run
WAS Server Support
Bundle Explorer
Blueprint Graphical Editor
SCA / OSGi Integration
Developer Productivity
(e.g. content assist, validation, re-factoring)
Graphical Application Editor
Tutorials and Documentation
Graphical Creation Wizards
Creation / Import / Export Tools
WTP 3.6
PDE
Eclipse
OSGi App Devt Tool
RAD V8

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/v8 and includes the WAS Test Environment
- 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

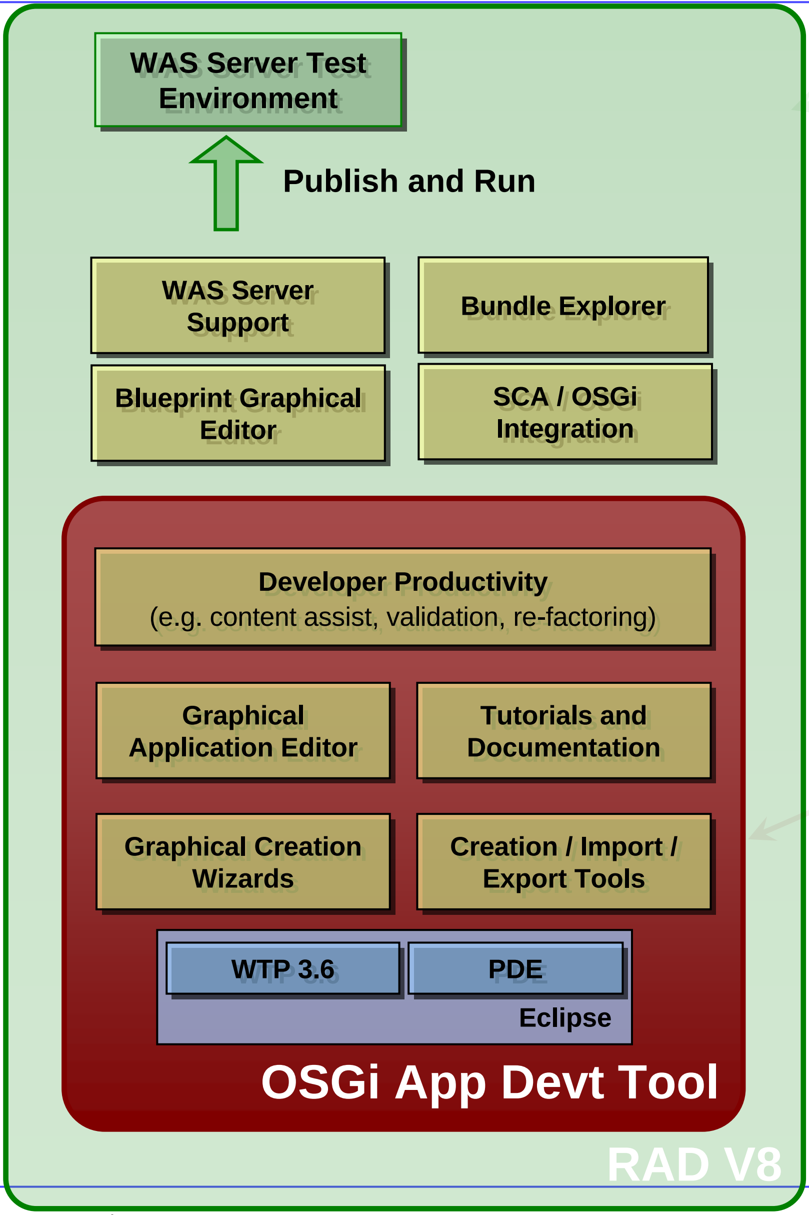
Free Eclipse Plugin for OSGi Applications

➢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) required

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

OSGi Application Development Tools



WAS Server Test Environment
Publish and Run
WAS Server Support
Bundle Explorer
Blueprint Graphical Editor
SCA / OSGi Integration
Developer Productivity
(e.g. content assist, validation, re-factoring)
Graphical Application Editor
Tutorials and Documentation
Graphical Creation Wizards
Creation / Import / Export Tools
WTP 3.6
PDE
Eclipse
OSGi App Devt Tool
RAD V8

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/v8 and includes the WAS Test Environment
- 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

Scope of new Eclipse Libra Project

➢Open source tooling for Enterprise OSGi applications

- Brings together PDE and WTP

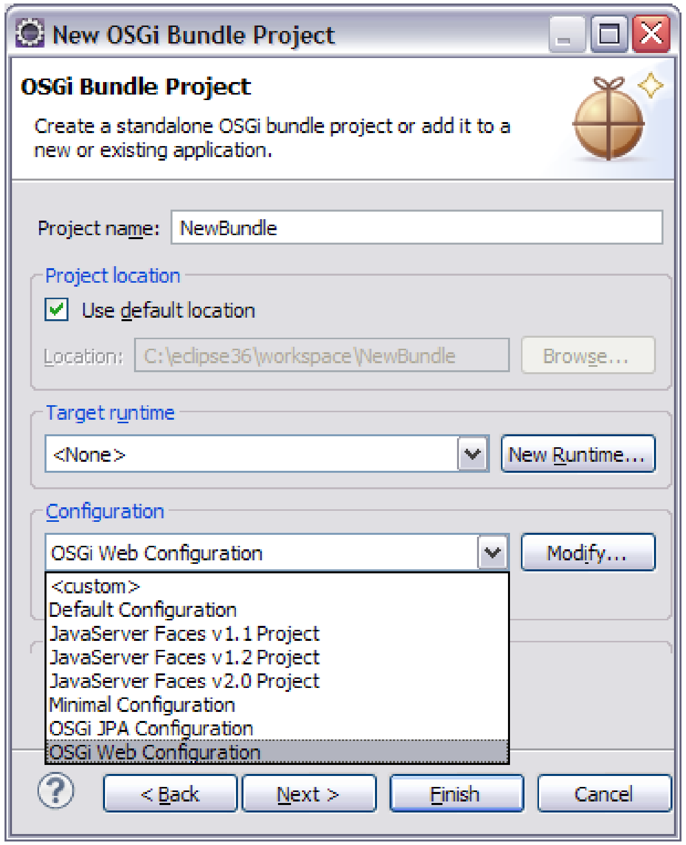
IBM intends to contribute to Libra project

http://www.eclipse.org/libra/

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 header
  – 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

- Plus in RAD: Comprehensive support for Web and JPA. These tools 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


New OSGi Bundle Project

OSGi Bundle Project

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

Project name: NewBundle

Project location

[✓] Use default location

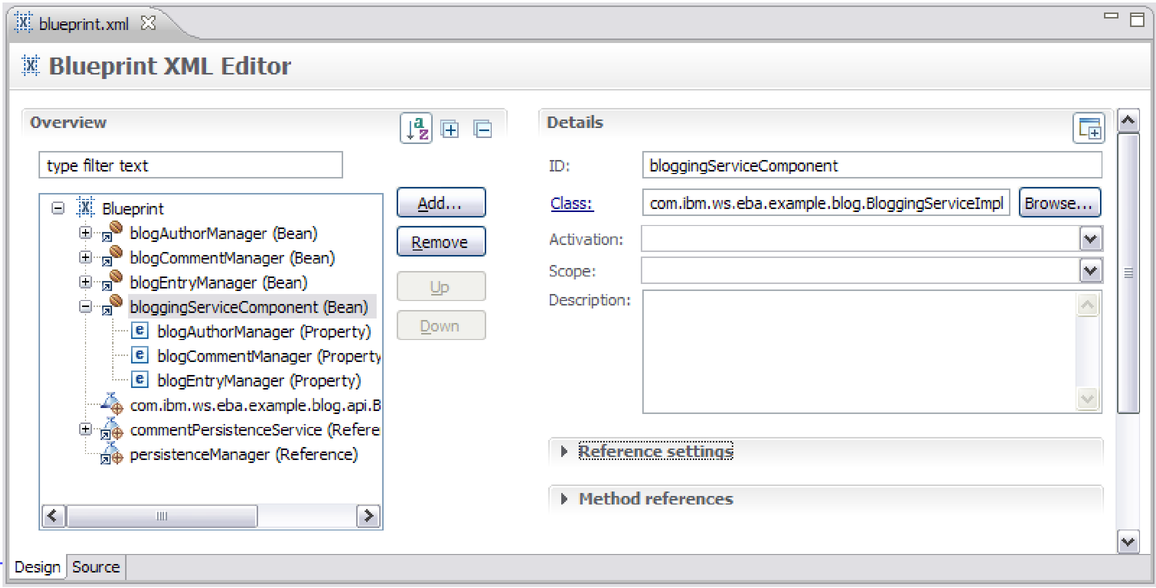
Location: C:\eclipse36\workspace\NewBundle

Target runtime: <None>

Configuration: OSGi Web Configuration

Development Support for Blueprint Bundles

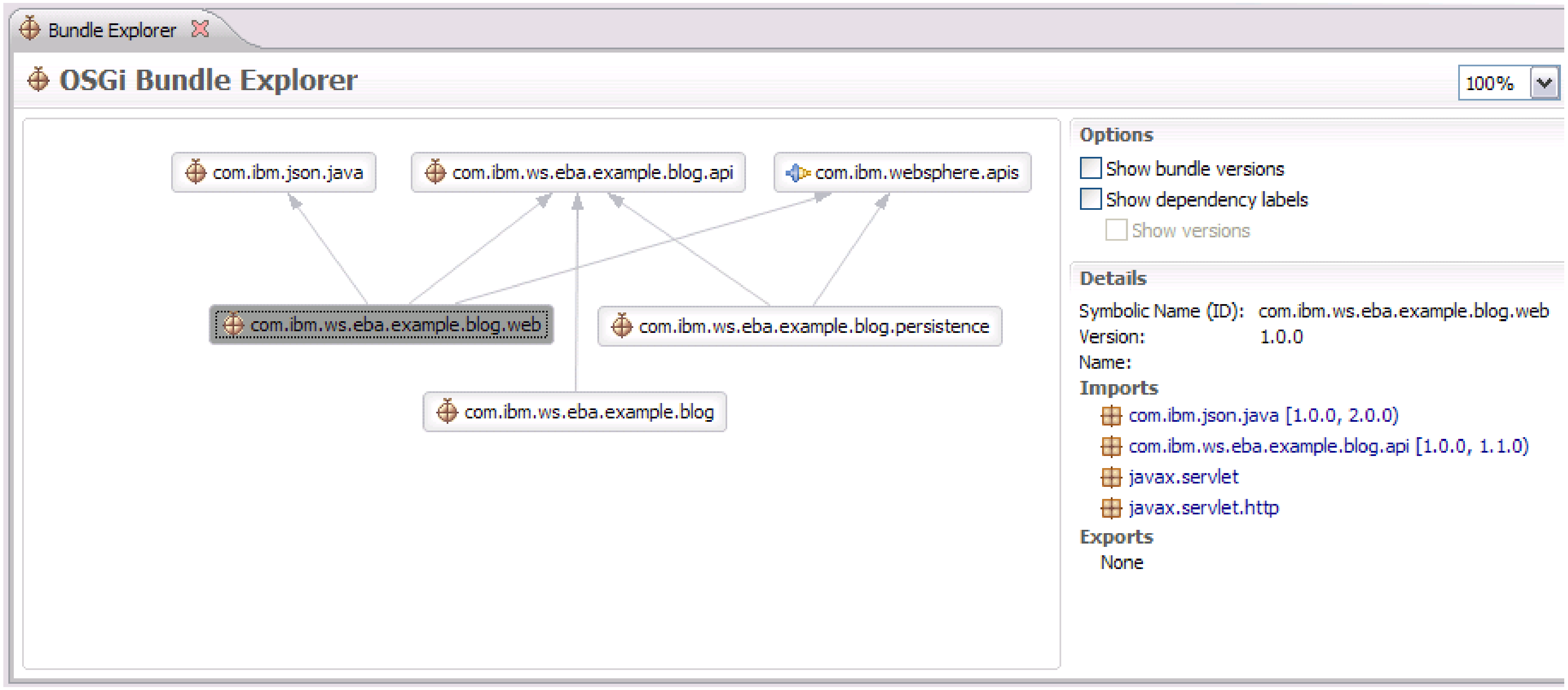
- Blueprint creation wizard
  - Supports extension schema including JPA and transactions
- A blueprint editor, including:
  - Source page with syntax highlighting, content assist
  - Form-based editing similar to Java EE deployment descriptors (RAD)
  - Syntax and reference validation
  - Hyperlinking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



Blueprint
- blogAuthorManager (Bean)
- blogCommentManager (Bean)
- blogEntryManager (Bean)
- bloggingServiceComponent (Bean)
  - blogAuthorManager (Property)
  - blogCommentManager (Property
  - blogEntryManager (Property)
- com.ibm.ws.eba.example.blog.api.B
- commentPersistenceService (Refere
- persistenceManager (Reference)

ID: bloggingServiceComponent
Class: com.ibm.ws.eba.example.blog.BloggingServiceImpl Browse...
Activation:
Scope:
Description:

Bundle Explorer


Bundle Explorer
OSGi Bundle Explorer
100%
com.ibm.json.java
com.ibm.ws.eba.example.blog.api
com.ibm.websphere.apis
com.ibm.ws.eba.example.blog.web
com.ibm.ws.eba.example.blog.persistence
com.ibm.ws.eba.example.blog

Options

- [ ] Show bundle versions
- [ ] Show dependency labels
  - [ ] Show versions

Details

Symbolic Name (ID):	com.ibm.ws.eba.example.blog.web
Version:	1.0.0
Name:	

Imports

- com.ibm.json.java [1.0.0, 2.0.0)
- com.ibm.ws.eba.example.blog.api [1.0.0, 1.1.0)
- javax.servlet
- javax.servlet.http

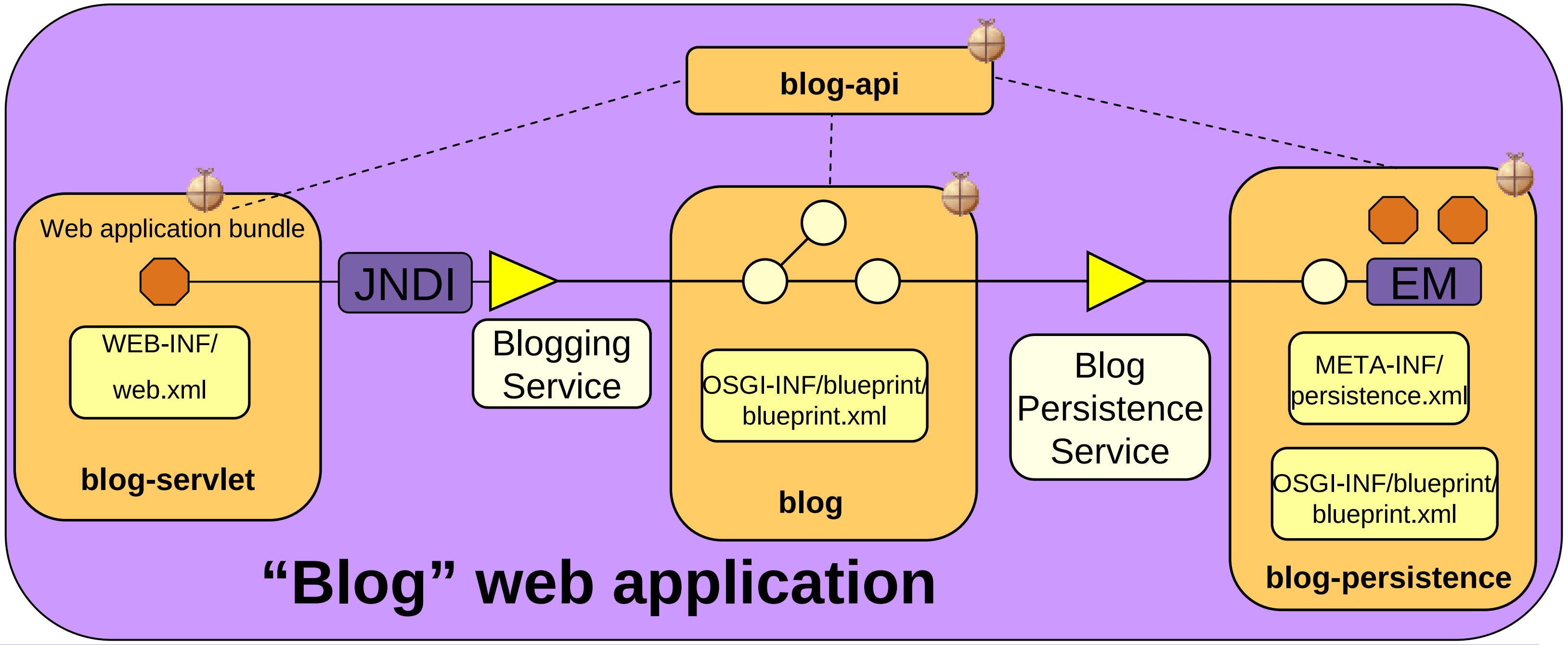
Exports

None

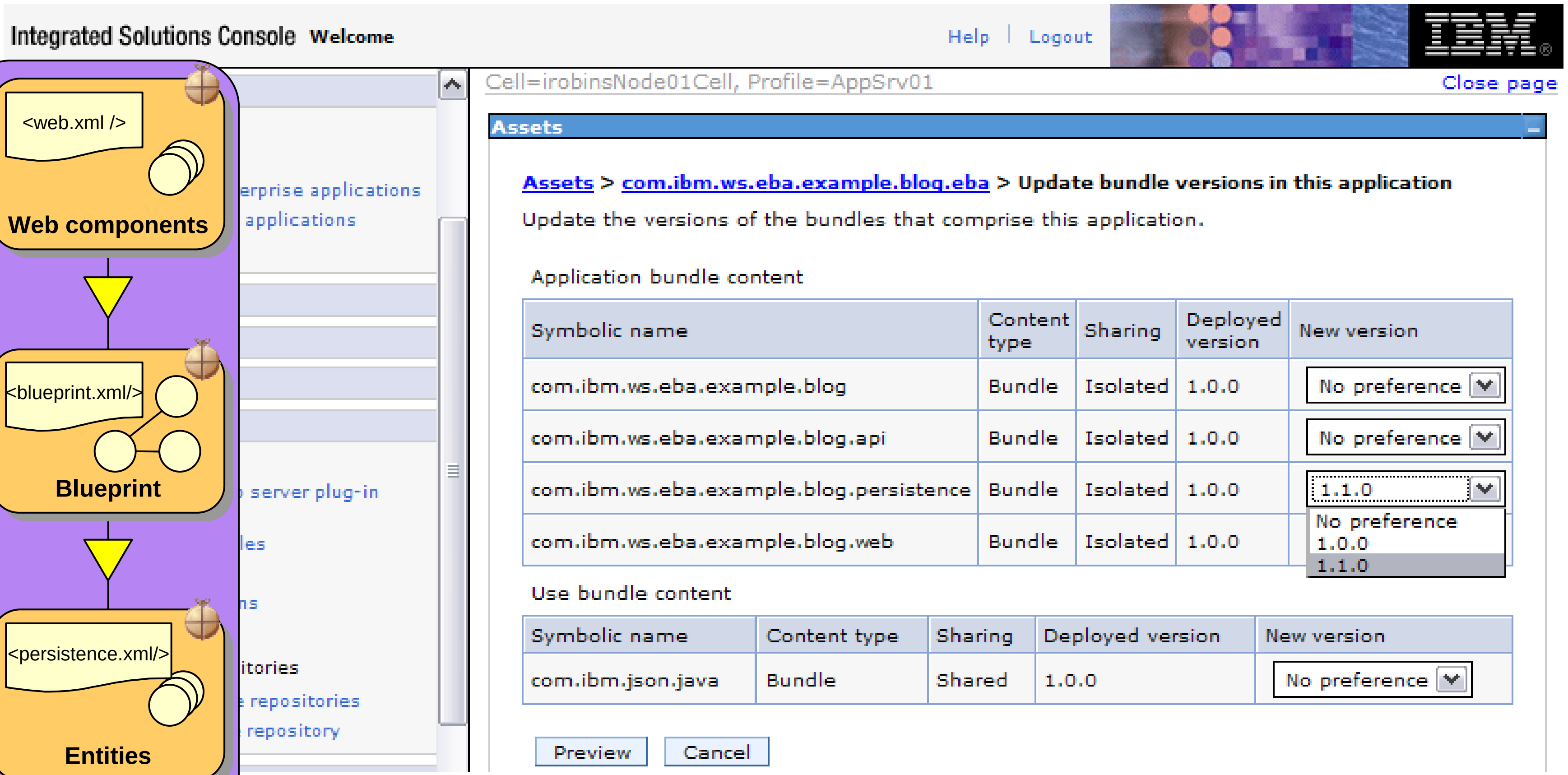
An Example OSGi Application

Includes:

- Web and Persistence bundles
- POJO bean assembly and declarative services through OSGi Blueprint container.
- Bridging between JNDI and OSGi Services
- Dynamic application update



Reducing Complexity in Operational Management



Integrated Solutions Console Welcome

Help | Logout

Cell=irobinsNode01Cell, Profile=AppSrv01 Close page

Assets

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

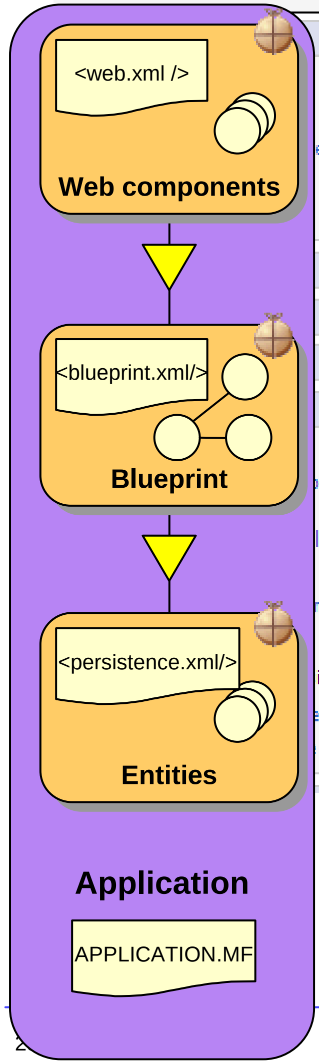
Update the versions of the bundles that comprise this application.

Application bundle content

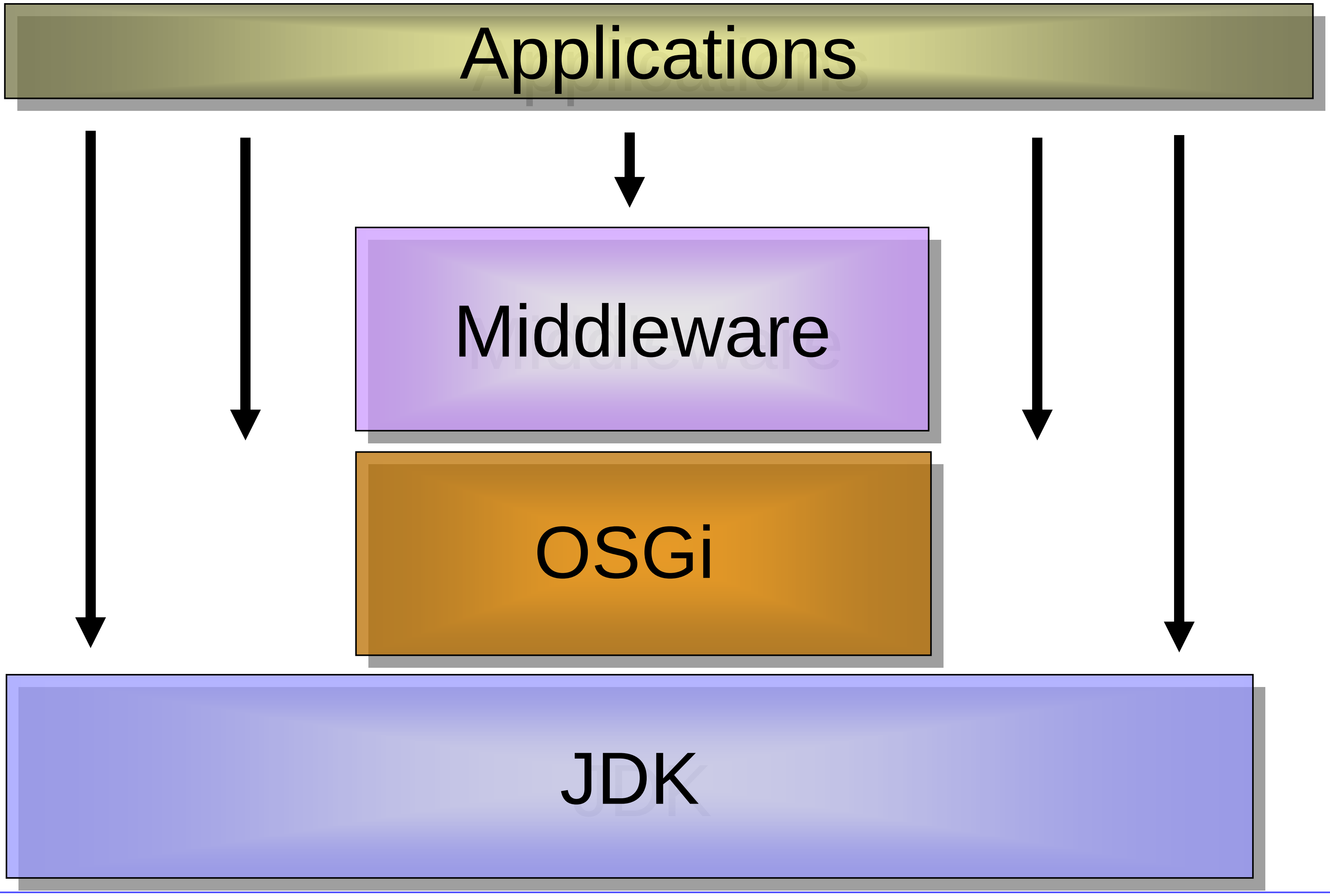
Symbolic name	Content type	Sharing	Deployed version	New version
com.ibm.ws.eba.example.blog	Bundle	Isolated	1.0.0	No preference
com.ibm.ws.eba.example.blog.api	Bundle	Isolated	1.0.0	No preference
com.ibm.ws.eba.example.blog.persistence	Bundle	Isolated	1.0.0	1.1.0
com.ibm.ws.eba.example.blog.web	Bundle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

Use bundle content

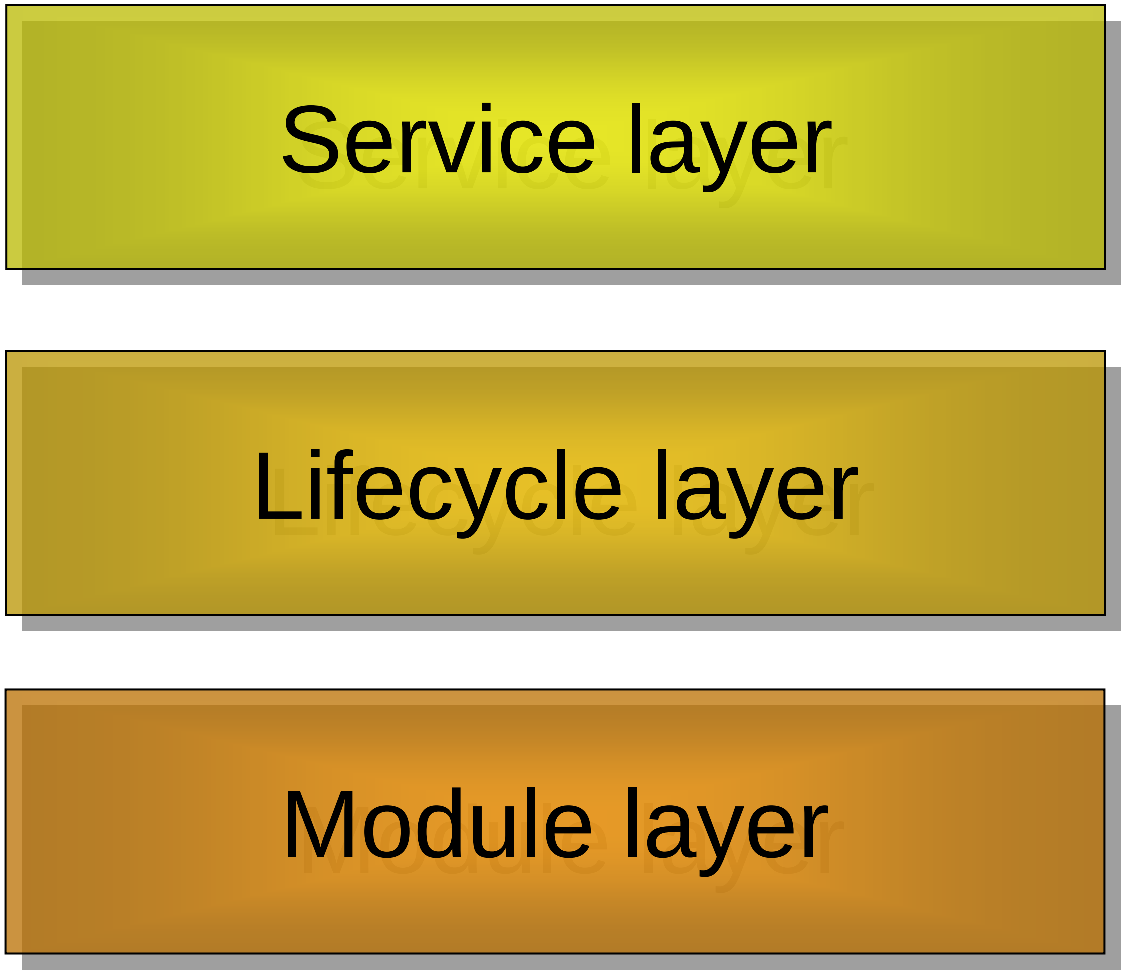
Symbolic name	Content type	Sharing	Deployed version	New version
com.ibm.json.java	Bundle	Shared	1.0.0	No preference



Where Next For OSGi?



OSGi Layer Cake - Today



api, events

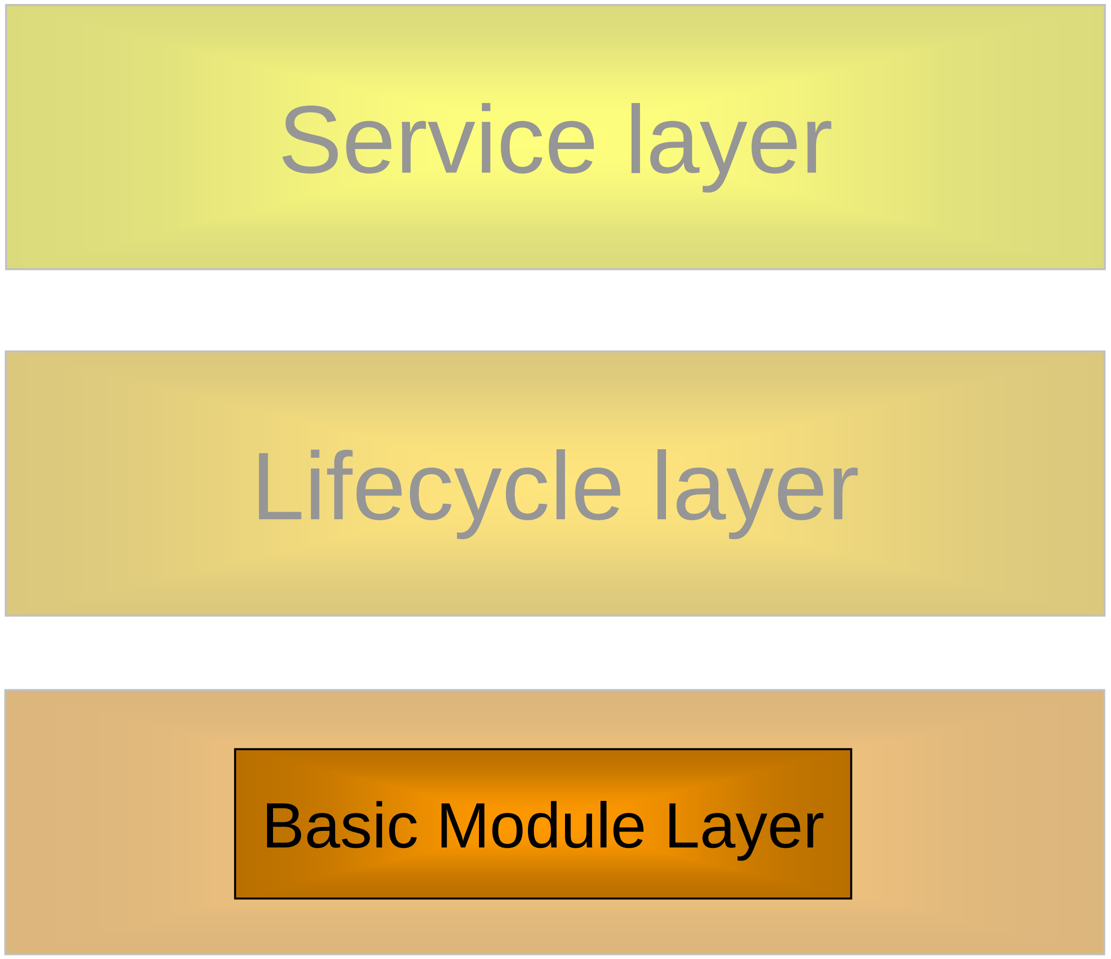
api, events

metadata, reflective
api on wiring state

Modularity Within the JDK

- More constrained requirements than addressed by entire OSGi framework
- Needs metadata for:
  - versioned packages exported outside module
  - versioned packages imported by module
- Doesn’t need dynamic lifecycle.
  - Therefore doesn’t need service layer either
- OSGi-- then?
  - More likely (OSGi--)++
  - A basic OSGi subset meeting requirements specific to a modular JDK

OSGi within the JDK?



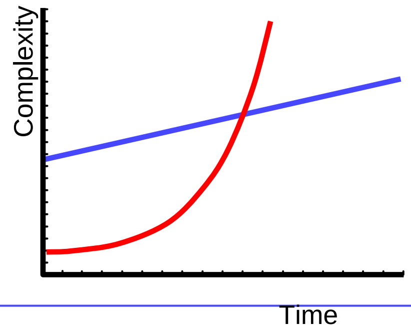
api, events

api, events

metadata, reflective api on wiring state

Or Create a Tailored Modularity System?

- *“History repeats itself because no one was listening the first time.”* Anon
- Modularity requirements of the JDK are likely to be of general use.
  - Apps with simple modularity requirements should be able to share the same solution
- Apps requiring more dynamic support will continue to need the full capabilities of OSGi.
- Only if we can make the modularity solution for the JDK **a precise subset of OSGi** will the transition from “basic” to “full” modularity be seamless.
  - Enable Java developers to invest in a SINGLE approach to modularity that adds capability as it is required.


Complexity
Time