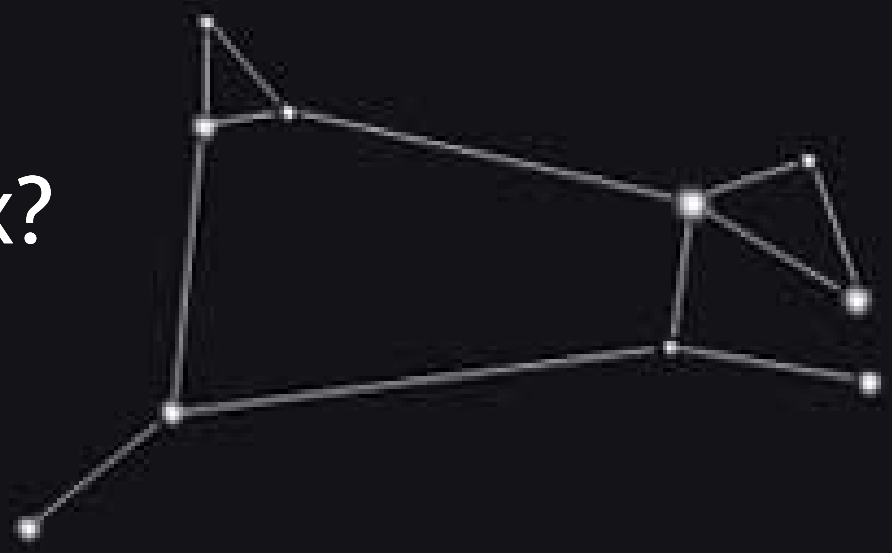
Apache ARIES

OSGi and the Enterprise
A match made in a … box?



Holly Cummins

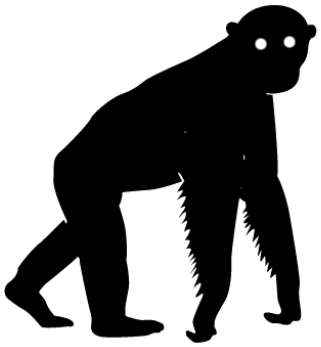
Paris Java User Group

June 8, 2010

A (very) brief history of programming

- In the beginning, there were bits …

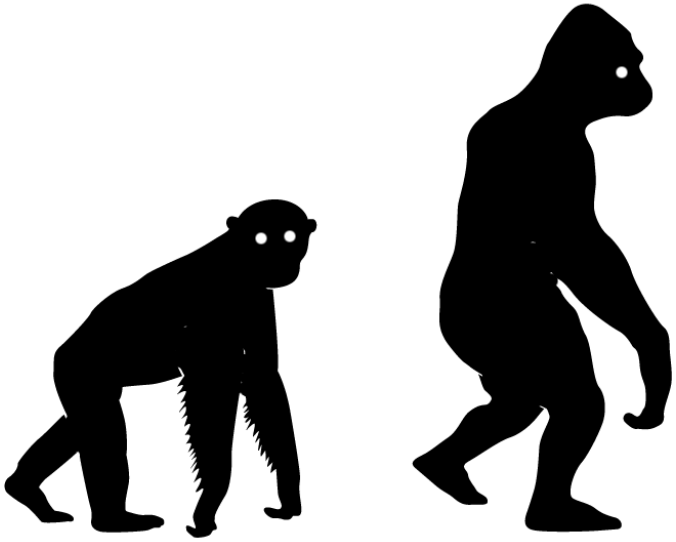
```
0101011101001001010010010011011101101101
```



A (very) brief history of programming

- Then came words …

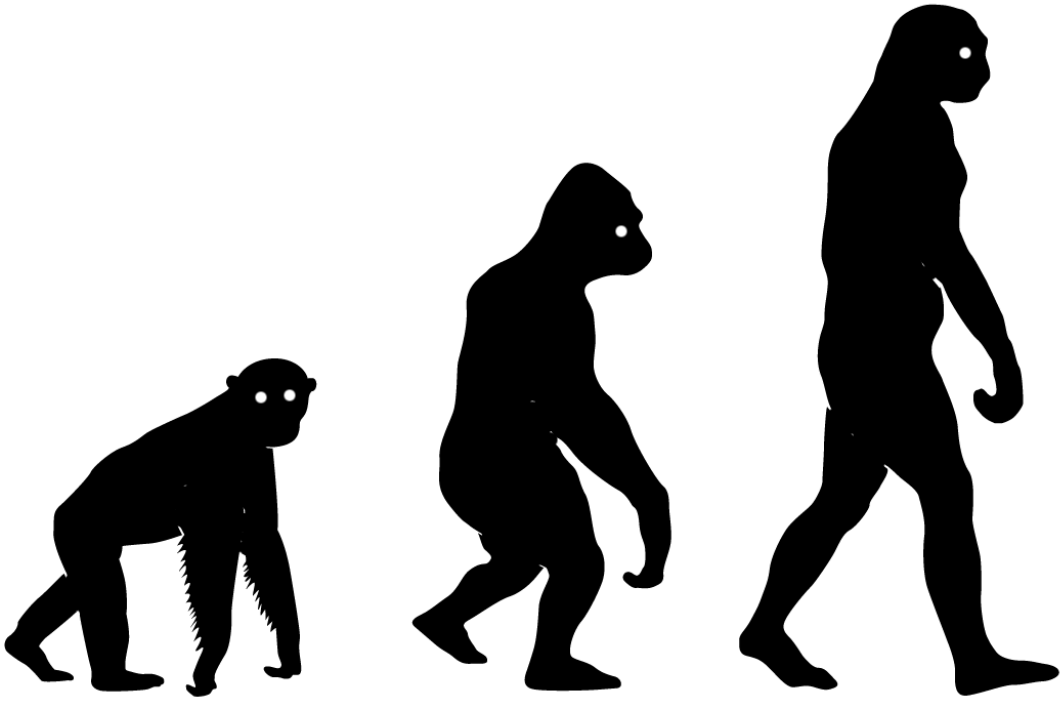
```
mov     ah,9
mov     dx,offset hello_message
int     21h
```



A (very) brief history of programming

- … functions and libraries …

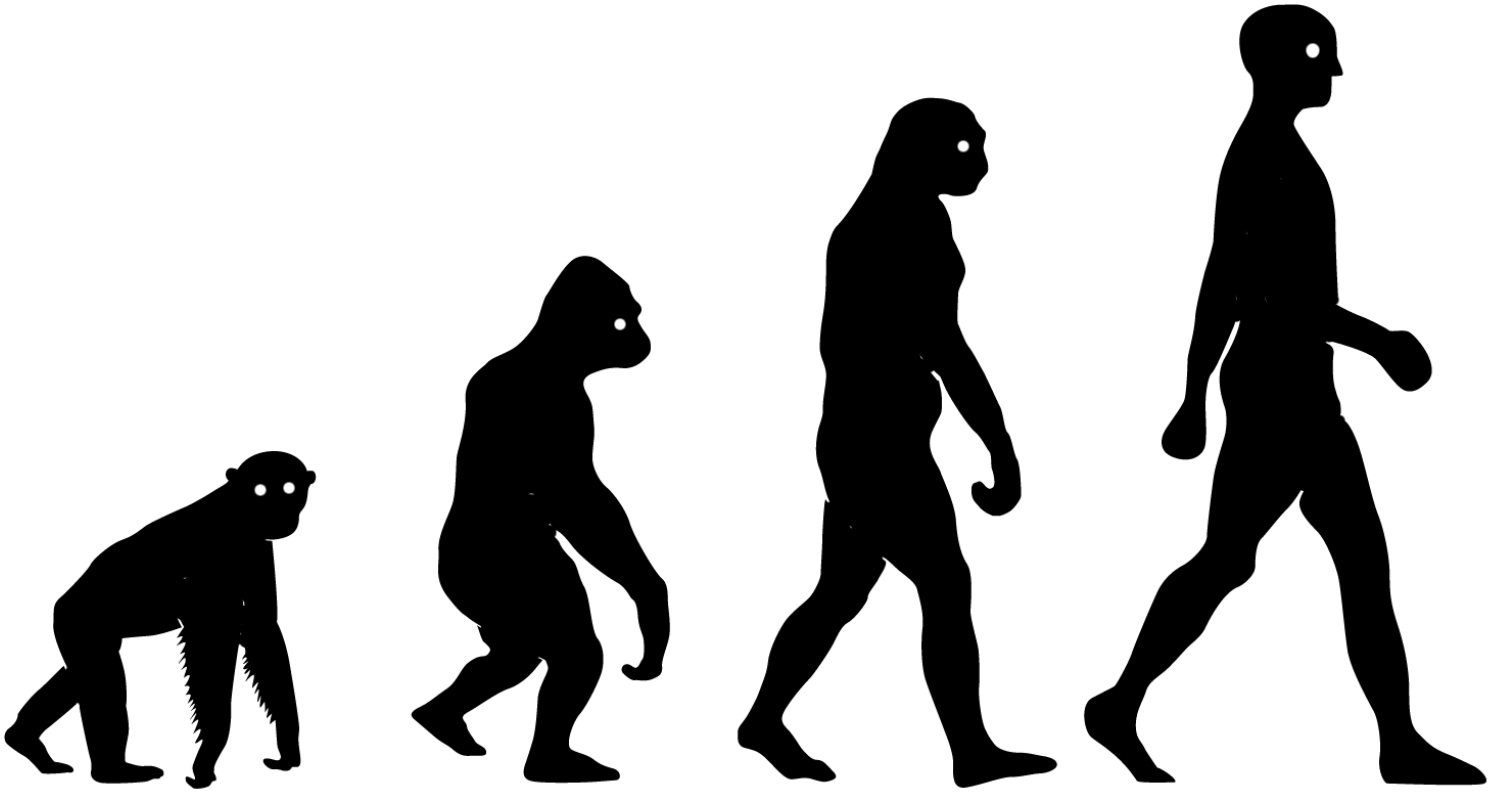
```
#include <stdio.h>
main() {
    printf ("Hello World!\n");
}
```



A (very) brief history of programming

- … objects …

```
public class HelloWorldSayer {

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

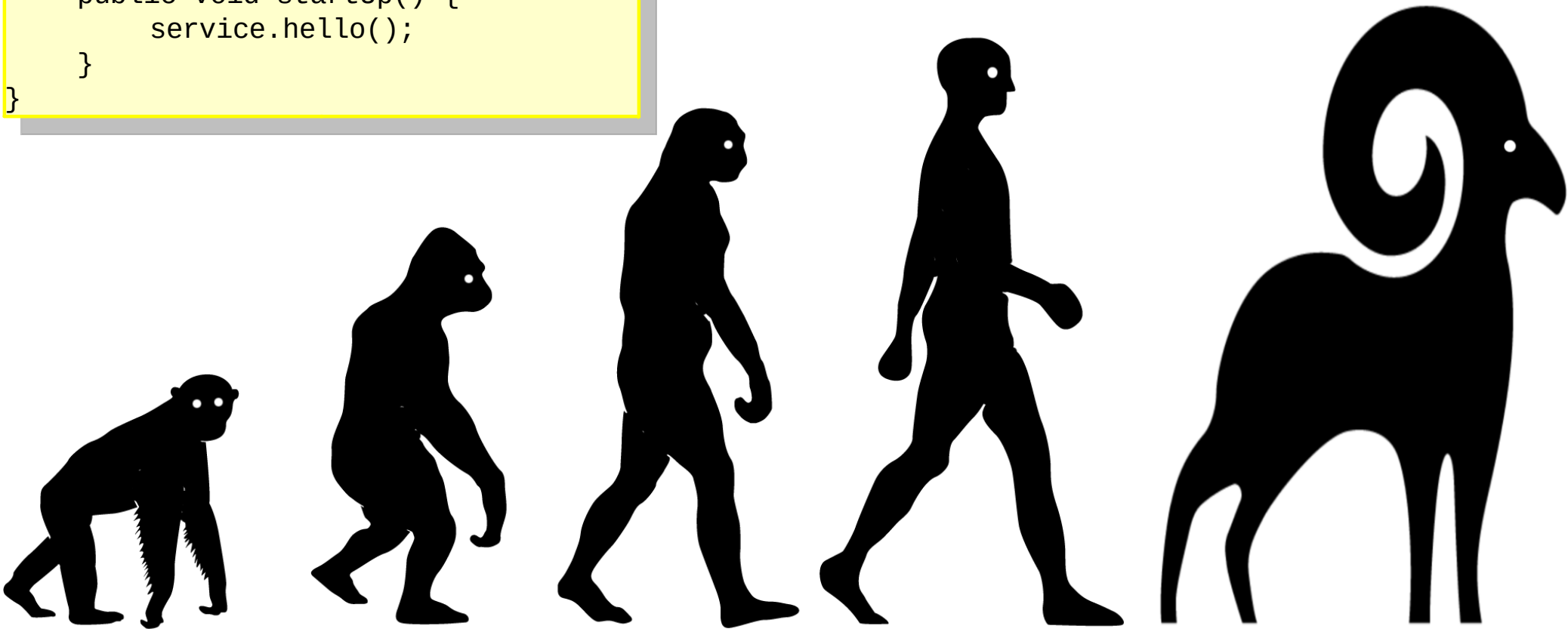


A (very) brief history of programming

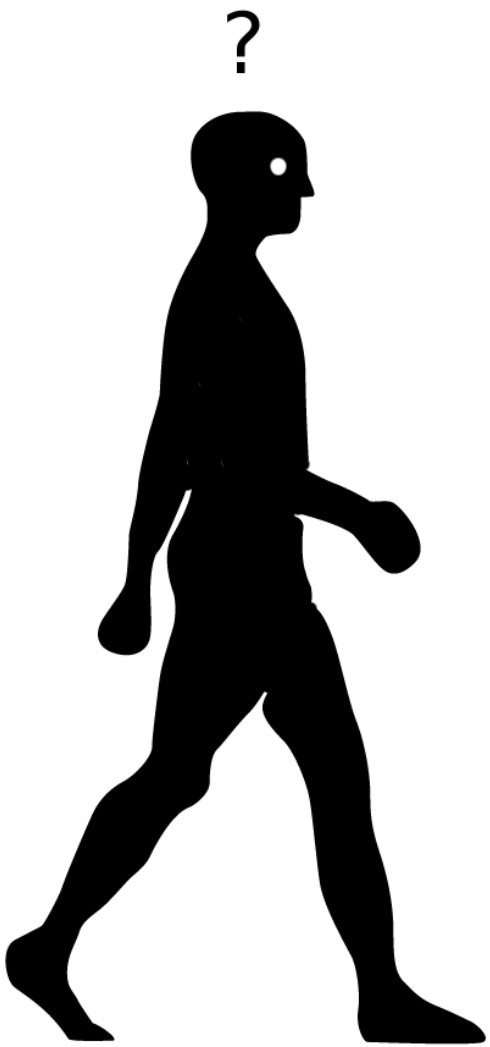
- ... and now - Enterprise OSGi

```
<service ref="helloservice"  interface="helloworld.HelloWorldService" />
```

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

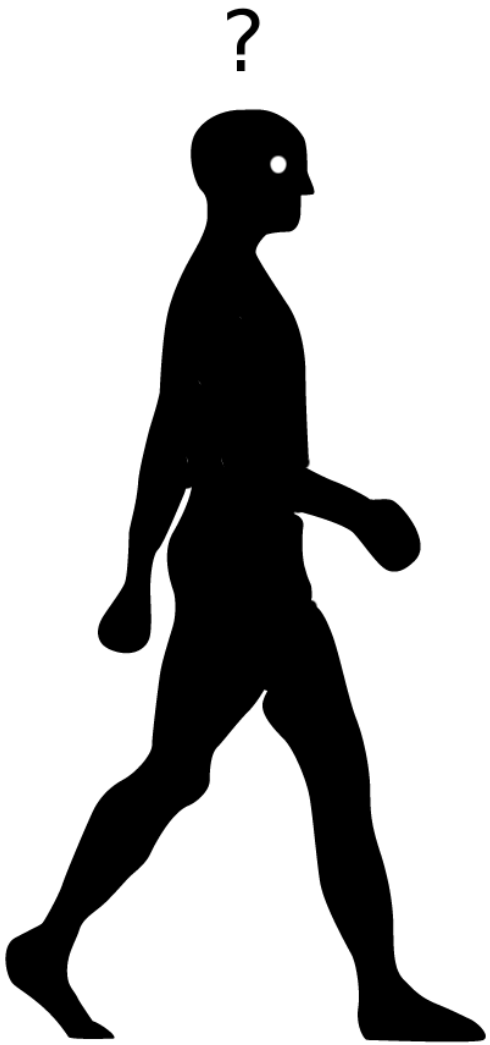


What was wrong with where we were?

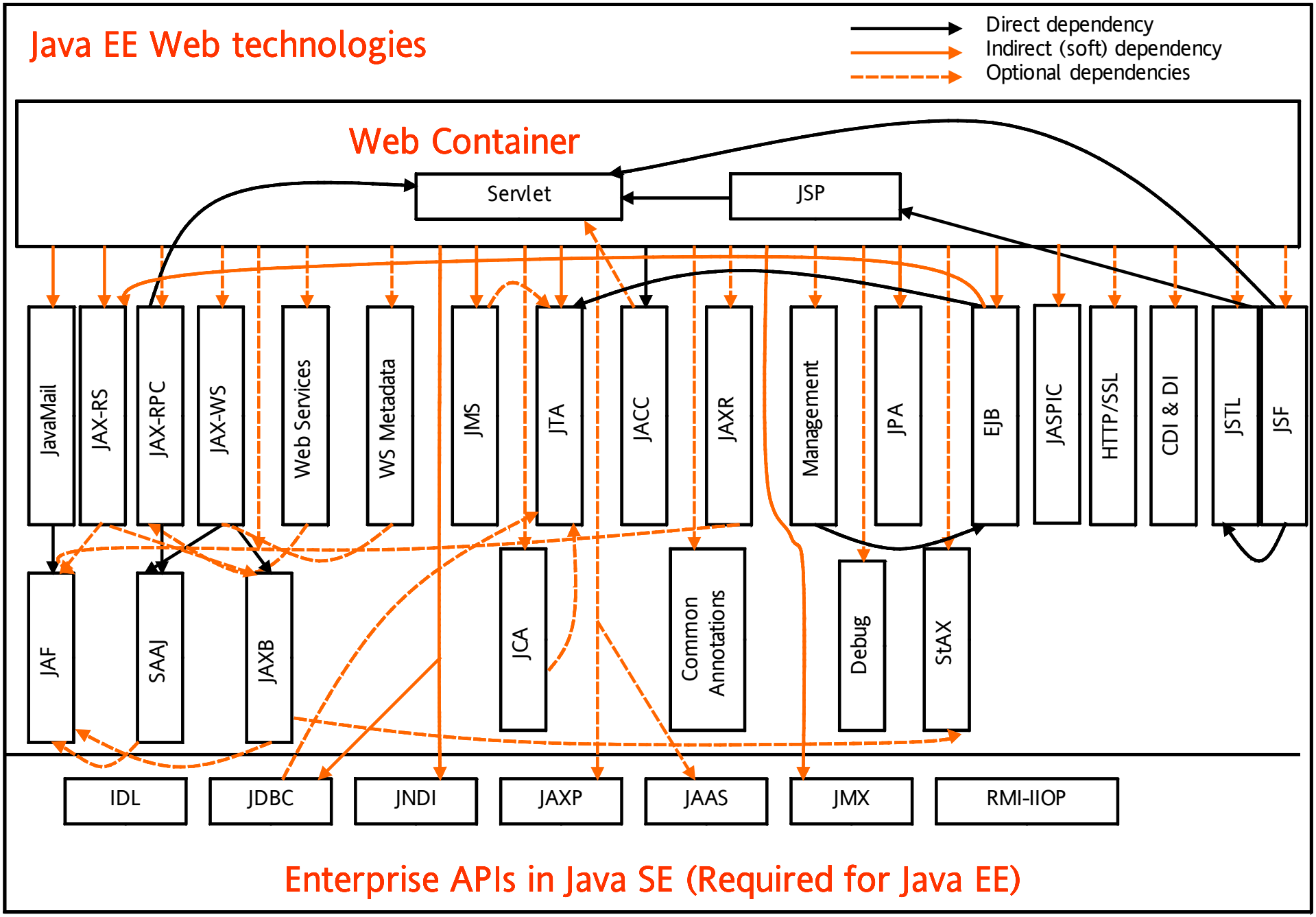


What was wrong with where we were?

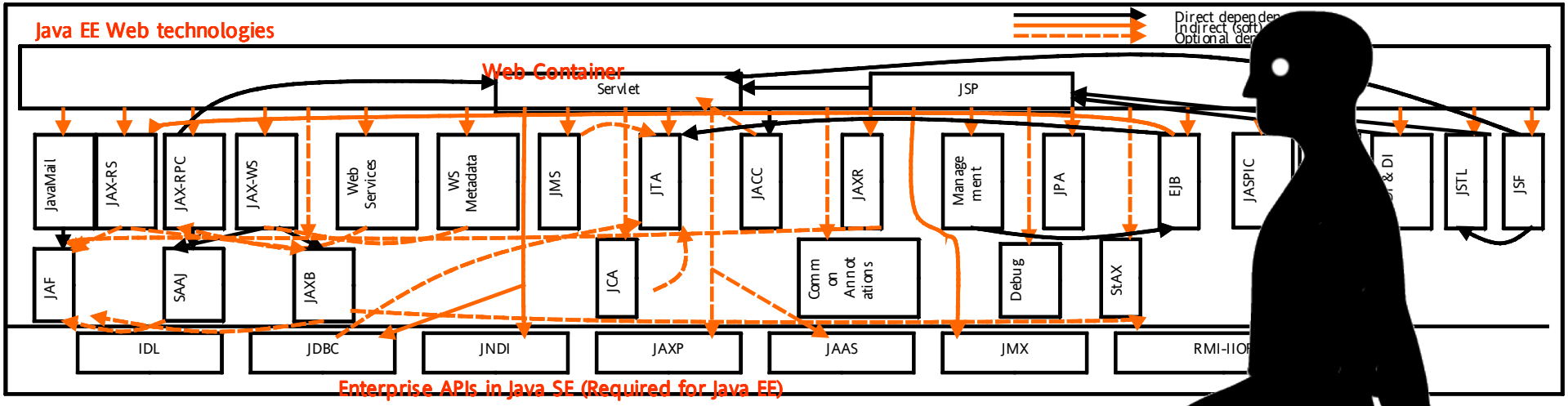
- A question of scale

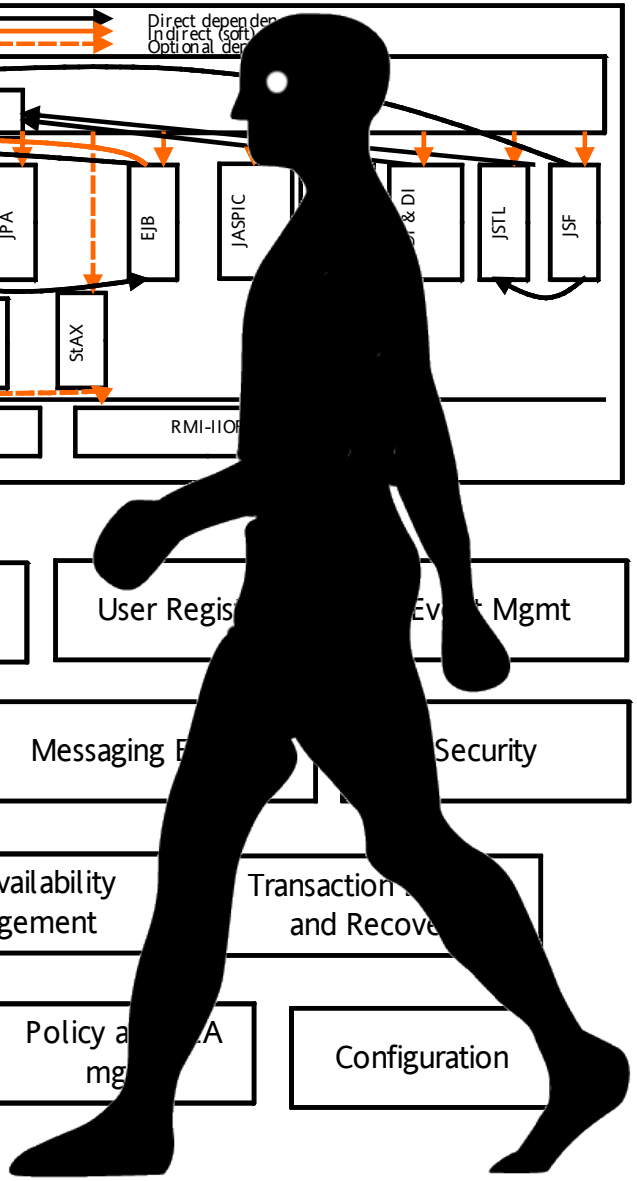


Java EE Web technologies



(capability == bloat)?





HTTP Session Replication

Connection Management

Persistence Providers

User Regis… Mgmt

Logging and Tracing

MBean Server

Admin Console and scripting

DataSources

Messaging … Security

Transport Channels

Workload Management

HTTP Serving and routing

High Availability Management

Transaction … and Recove…

Thread management

Classloading

Provisioning

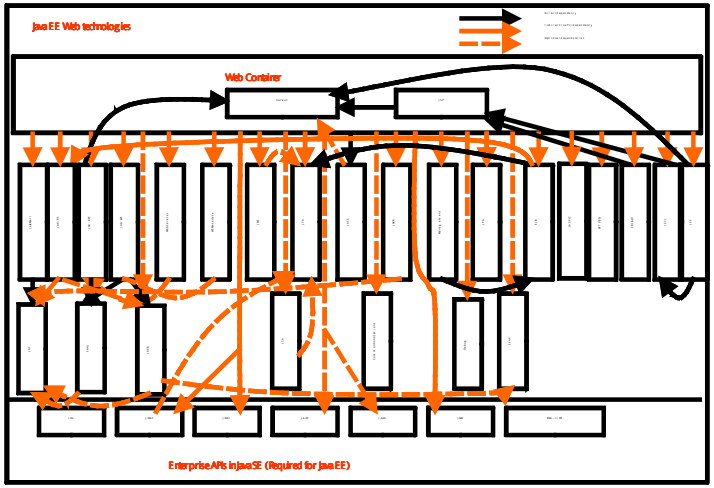
Monitoring

Policy a… mg…

Configuration

…

Beyond Java EE…



!@*!

Batch

Telephony

BPM

Dynamic Scripting

SCA

Business Rules

Complex Pro… ent

…

HTTP Session Replication

Connection Management

Persistence Providers

Use… stries … mgmt

Logging and Tracing

MBean Server

Admin Console and scripting

DataSources

Messaging Engine … urity

Transport Channels

Workload Management

HTTP Serving and routing

High Availability Management

…saction L… and Recov…

Thread management

Classloading

Provisioning

Monitoring

Policy and S… mgmt

Configuration

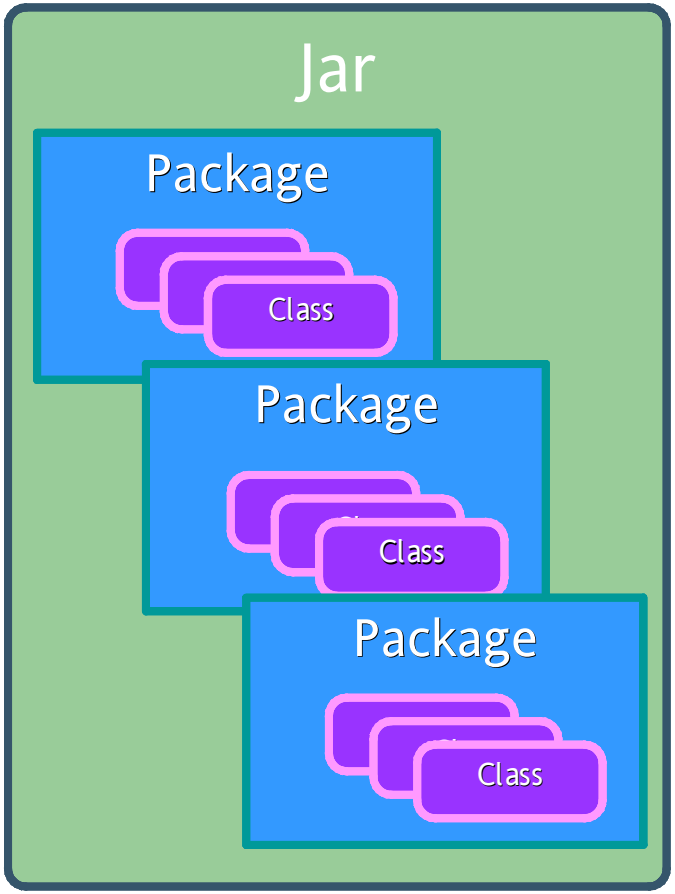
…

Modularity *inside* Enterprise platforms

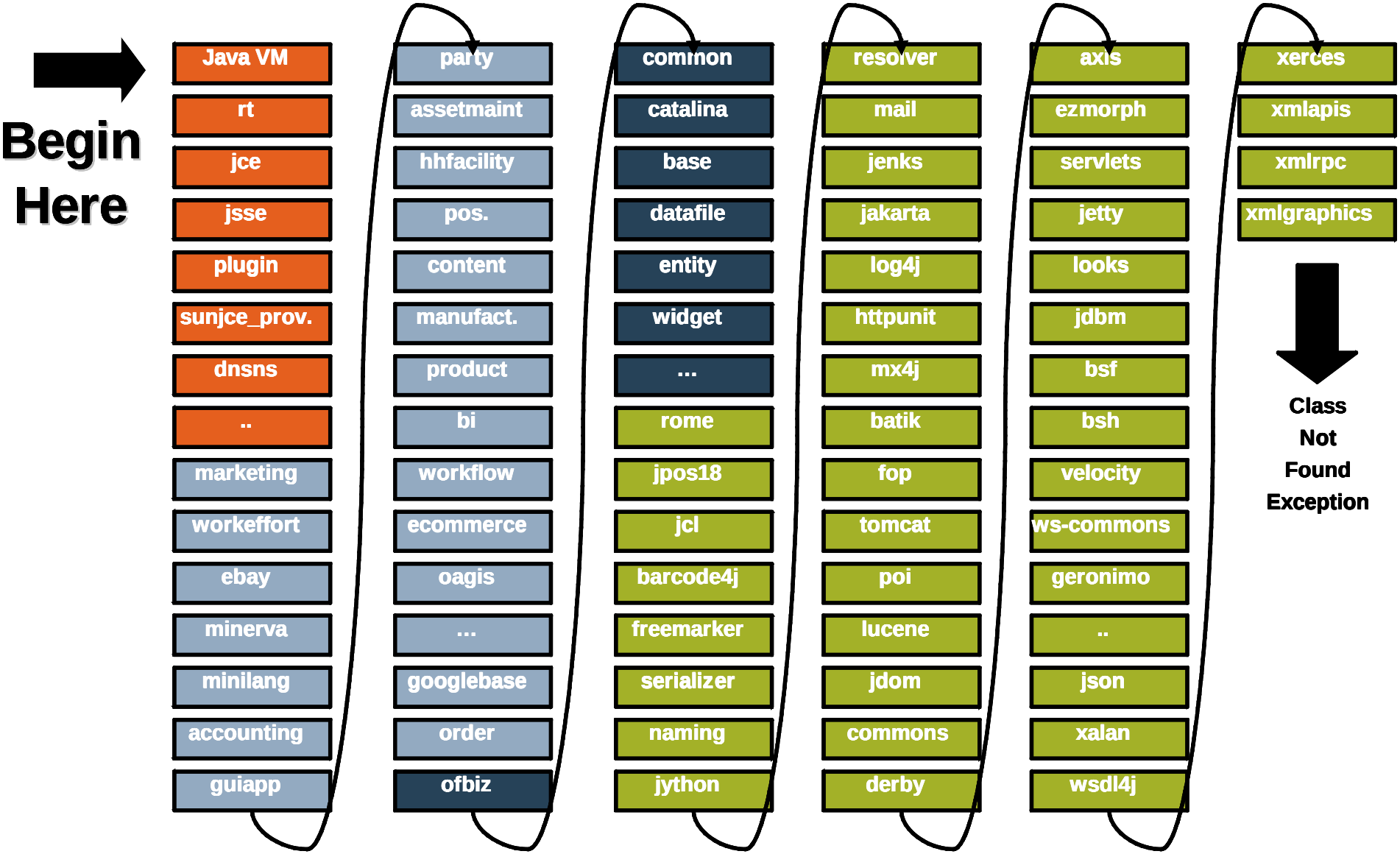
- The major AppServer and Integration vendors and Open Source communities have all embraced OSGi for their own internal architecture
  - IBM WebSphere, Oracle WebLogic, Red Hat JBoss, Sun GlassFish, SpringSource dm Server, Paremus Service Fabric, Eclipse Platform, Apache Geronimo, (non-exhaustive list)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

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



The Global Classpath



Problems with EARs/WARs

Enterprise Apps have isolated classpaths but…

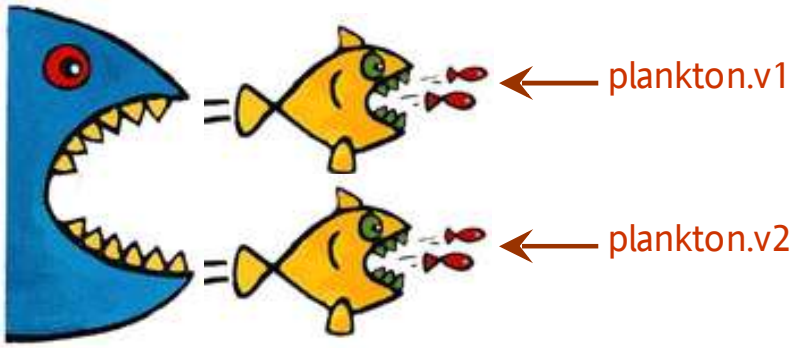
- *Across apps* - each archive typically contains all the libraries required by the application
  - Common libraries/frameworks get installed wit each application
  - Multiple copies of libraries in memory

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

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

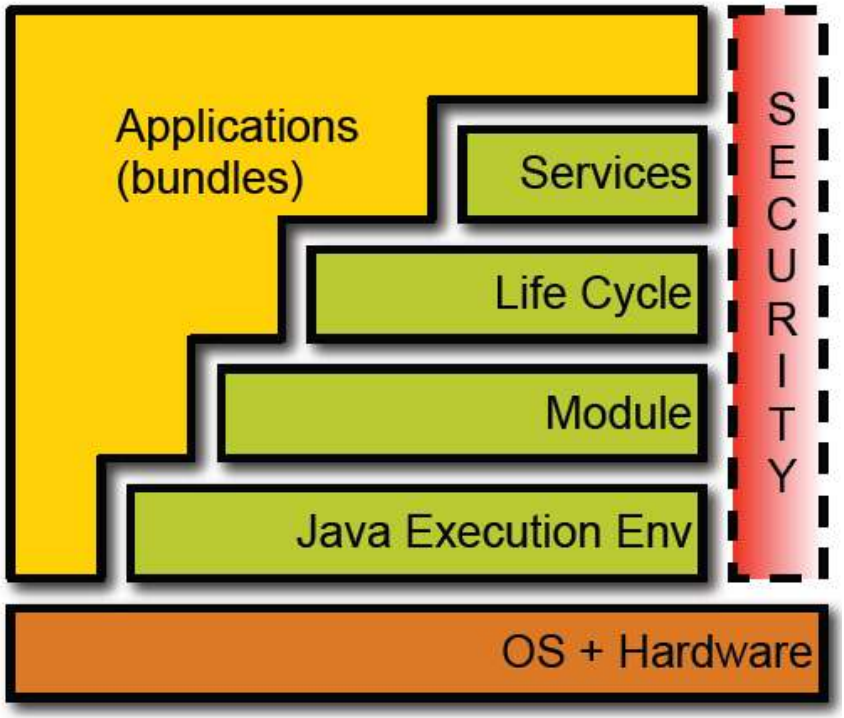
webC.war
WEB-INF/classes/servletC.class
WEB-INF/lib/spring.jar
WEB-INF/lib/commons-logging.jar
WEB-INF/lib/junit.jar…

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



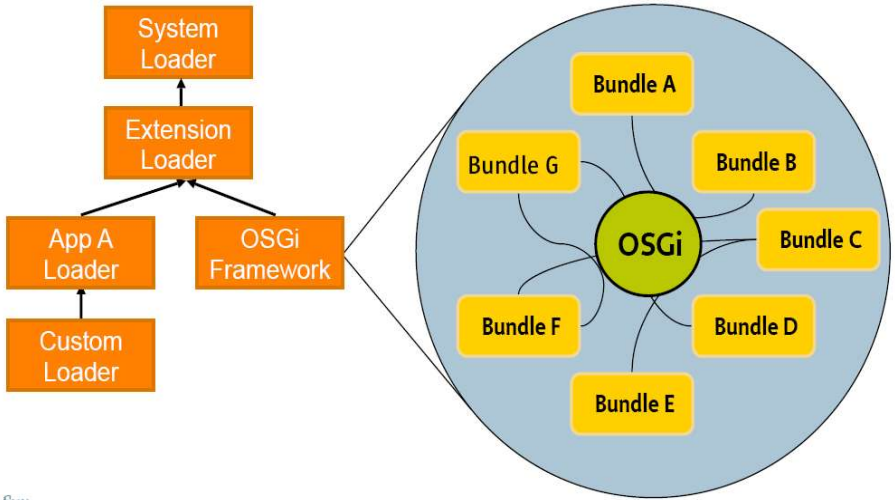
OSGi – The Dynamic Module System for Java

- The OSGi Service Platform specifies a modular architecture for dynamic component based systems
  - Execution Environment
  - Module Layer
  - Life Cycle Layer
  - Service Layer
- Runs on a variety of standard Java profiles.
- Introduces *Bundles* as modules



OSGi Class Loading

- No global, flat classpath to search. Each bundle has its own classloader / classpath. More efficient for large systems.
- Class sharing and visibility is based on declarative dependencies
  - Not constrained by hierarchical class loader trees built up at runtime.
  - OSGi f/w takes care of working out the dependencies.
- Multiple versions of bundles supported concurrently.



Bundle Metadata

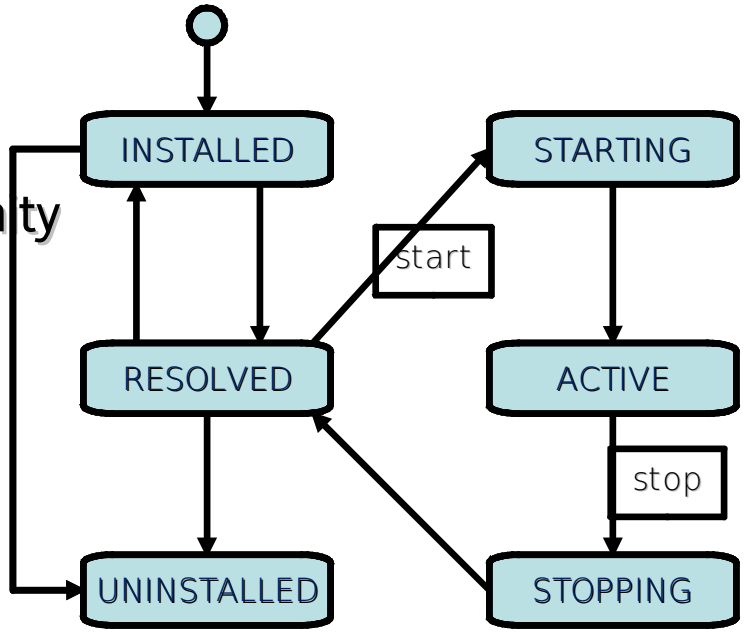
- The higher-level module is the “bundle” which is a normal jar containing:
  - Classes
  - Resources
  - Manifest

META-INF/MANIFEST.MF

```
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:
 org.apache.commons.logging;version="1.0.4"
Export-Package:
 com.sample.myservice.api;version="1.0.0"
```

- **Multiple versions** of bundles can be active concurrently.
- **Dependencies** on other bundles are explicitly declared
- **Package visibility** outside the bundle is declared.
  - Other packages are encapsulated by the bundle.
- Bundles can act on lifecycle changes
- Eclipse tooling provides convenient editors for the manifest

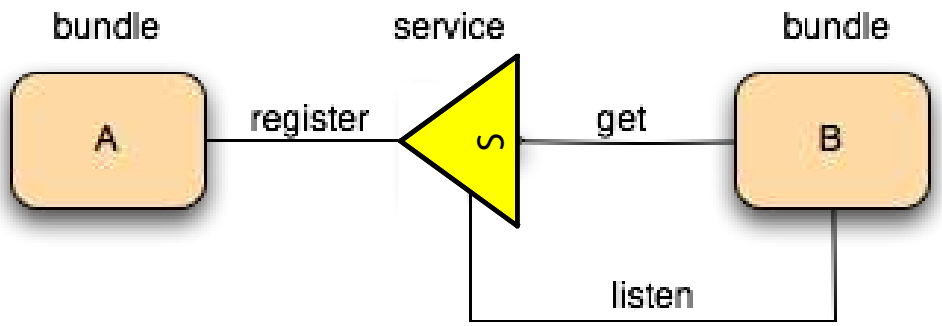
Bundle Lifecycle

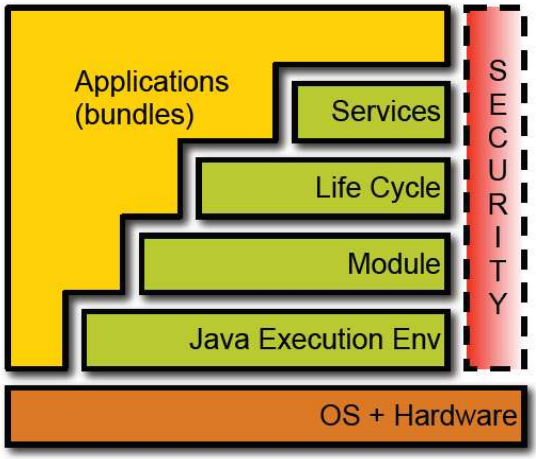


- Optional BundleActivator class has opportunity to initialize and cleanup a bundle
  - Header in Manifest refers to this class
- BundleActivator has 2 methods
  - Start: Initialize bundle
  - Stop: Cleanup

- Resolution of dependencies at deployment time. If resolution fails, bundle is not available for use.
  - No ClassNotFoundExceptions at runtime
- Bundle lifecycle is dynamic and independent of the rest of the f/w.
  - BundleListener can be registered for bundle state change events.

OSGi Services





- OSGi intrinsically supports SOA through its service layer.
- The core platform includes a non-durable “Service Registry” component.
- Services are published to and discovered from this Service Registry.
  - Services are the primary means of collaboration between bundles.
- An OSGi service is a POJO published to the SR under one or more Java interface names with optional key/value pair metadata
  - Service discovery can be filtered on the key/value metadata
- Services are fully dynamic and typically have the same lifecycle as the bundle that provides them.

What about the Applications?

- The enterprise landscape until recently has been OSGi on the inside only. Why?
- Primarily because:
  - Java EE defines an enterprise application programming model and container contracts to honour it
  - OSGi has typically defined platform specifications and services for those platforms.
    - It hasn't had much to say on application concerns such as
      - What's the component model?
      - What's the persistence architecture?
      - How are qualities of service configured?
      - Remoting
- This has changed in OSGi v4.2.

OSGi Enterprise Specification

- Release date – 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 component model and dependency injection container

- Java EE provides the core enterprise application programming model

What's the benefit?

- Deploying modules as OSGi bundles
  - simplifies reuse between applications
  - encourages (and enforces) modular design
  - provides versioning
  - enables dynamic module updates.

New Open Source Activities

- Apache "Aries" created as a new Apache incubator project in Sep 2009:
  - to provide enterprise OSGi spec implementations
  - to provide an environment for collaboration and innovation about the standards
  - to build a broad development community
- Eclipse Enterprise Modules ("Gemini") - currently being provisioned.
  - Sub-projects will provide EEG spec implementations http://www.eclipse.org/proposals/gemini/

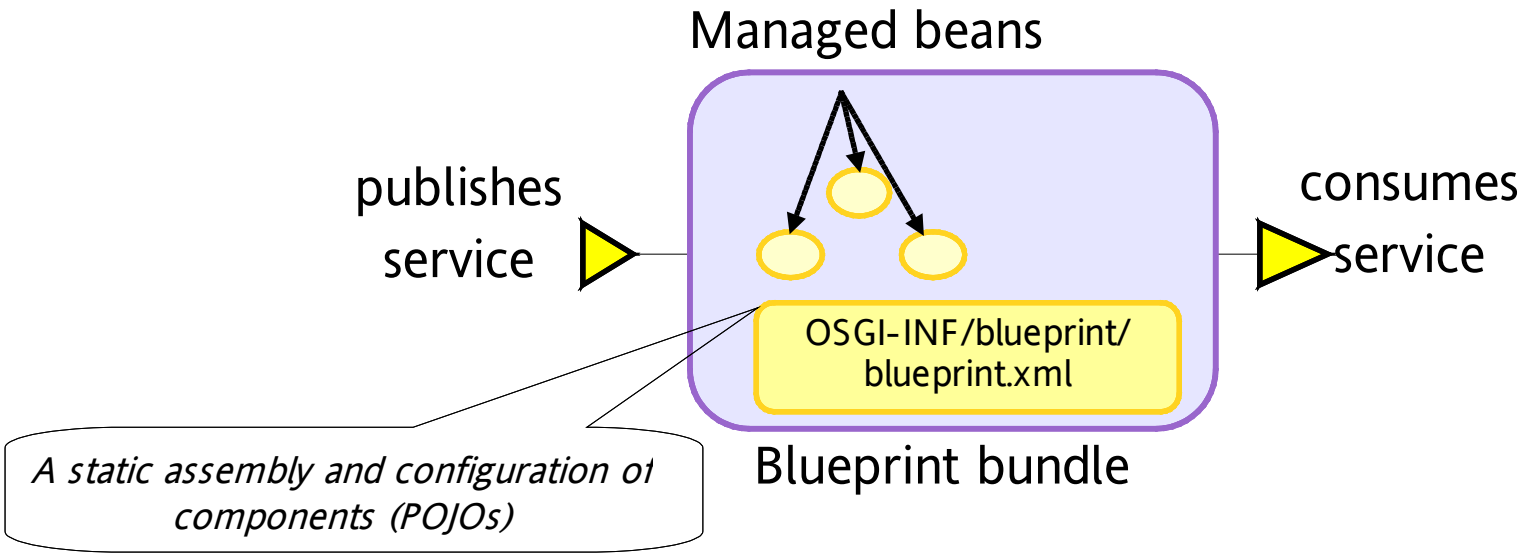
History

- "Aries" created as a new Apache incubator project in Sep 2009
- Available to use now
  - 0.1 version released last month
  - Integrated into WebSphere V7 Feature Pack for OSGi Applications – released last week
- Independent of
  - OSGi framework provider
  - Integration/server runtime
- 43 committers from a broad range of companies

Aries Content includes…

- Blueprint container
- JPA integration
- JTA integration
- JMX
- JNDI integration
- Application assembly and deployment
- META-INF/services handler
- Samples, documentation, integrator's guide

Aries Blueprint Container

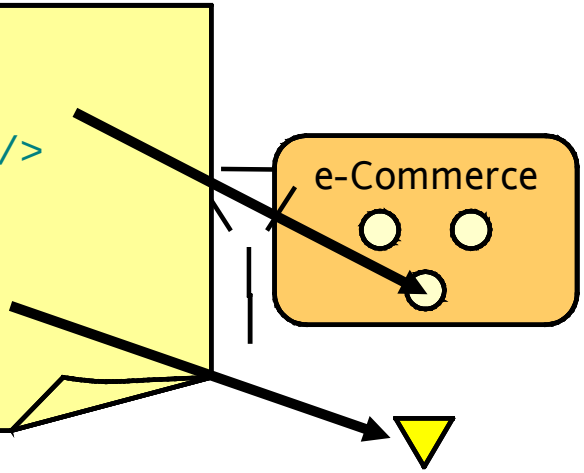


- A dependency Injection container
  - Standardizes established Spring conventions
  - Part of the server runtime (compared to Spring which is part of the app.)
- Configuration and dependencies declared in XML “module blueprint”, which is a standardization of Spring “application context” XML.
  - Extended for OSGi: publishes and consumes components as OSGi services
- Simplifies using OSGi services
  - No ServiceTrackers
  - No lookups

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>
```

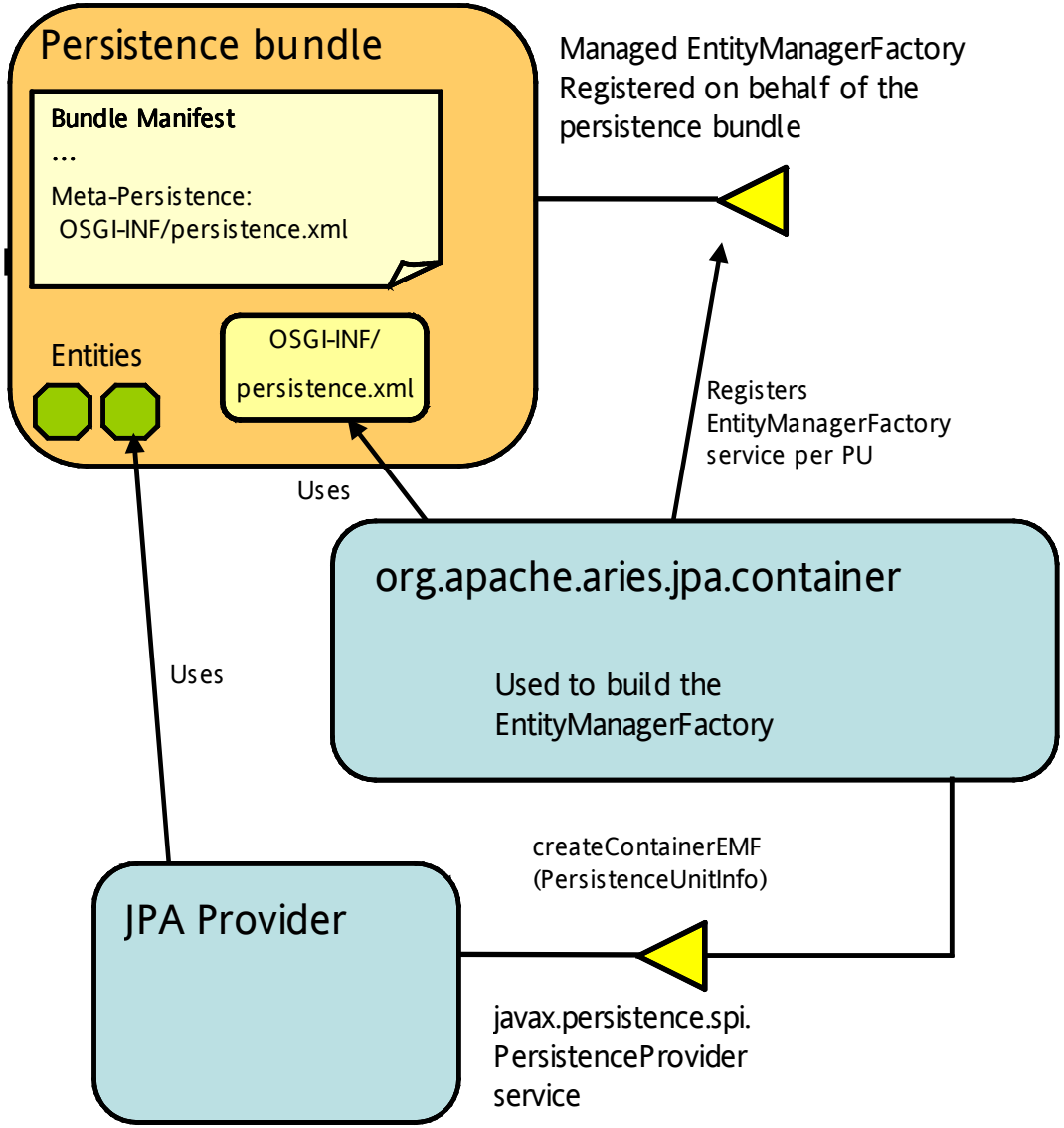



```
public class ShopImpl {

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

Java Persistence and Persistence Bundles

- Each persistence bundle has its standard JPA metadata located through the Meta-Persistence bundle header.
- JPA Container locates a JPA Provider
  - Registers a Provider-created EntityManagerFactory service for each PU
  - Optionally injects an EntityManager

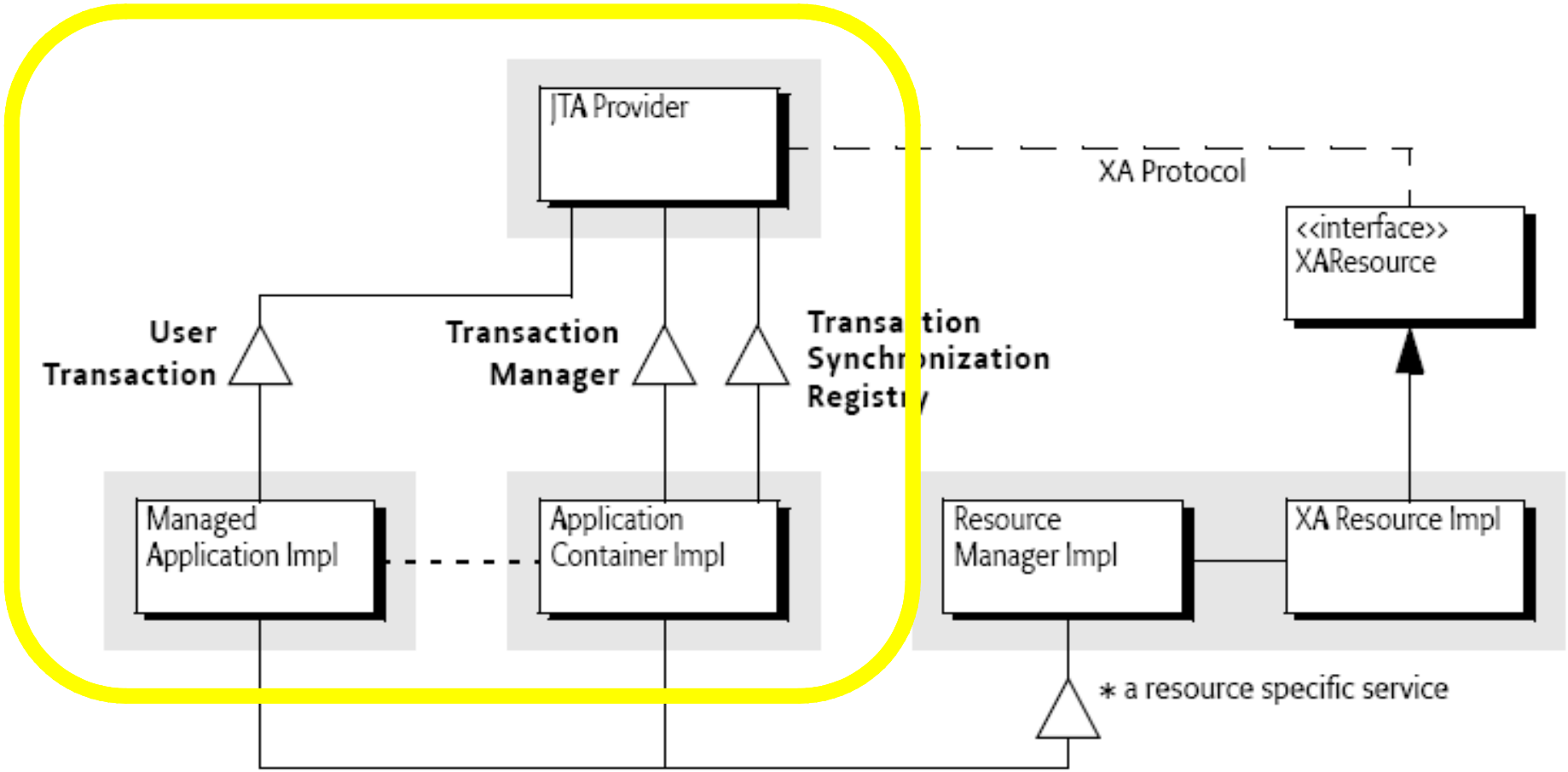


Aries JPA Container – Blueprint Integration

Example blueprint with JPA resource injection and container-managed transactions

```
<blueprint xmlns= "http://www.osgi.org/xmlns/blueprint/v1.0.0"
xmlns:jta="http://aries.apache.org/xmlns/transactions/v1.0.0"
xmlns:jpa= "http://aries.apache.org/xmlns/jpa/v1.0.0">

  <bean id= “appMgd“ class=“com.acme.AppManaged”>
    <jpa:unit property= "emf" unitname= "myUnit" />
  </bean>

  <bean id= "containerMgd“ class=“com.acme.Container”>
    <jpa:context property= "em" unitname= “myUnit“/>
    <jta:transaction method=“*" value=“Required" />
  </bean>

</blueprint>
```

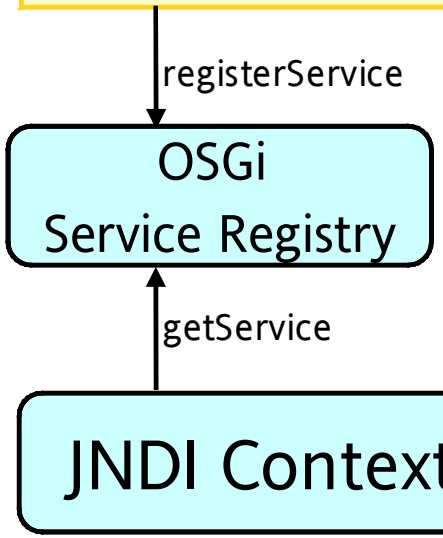
Aries JTA integration



Aries JNDI integration

- Provides JNDI-based access to OSGi Service Registry

```
<blueprint xmlns=...>
   <bean id="bloggingServiceComponent"
       class="org.apache.aries.BloggingServiceImpl">
   </bean>
   <service ref="bloggingServiceComponent"
      interface="org.apache.aries.samples.blog.api.BloggingService"/>
   ...
</blueprint>
```

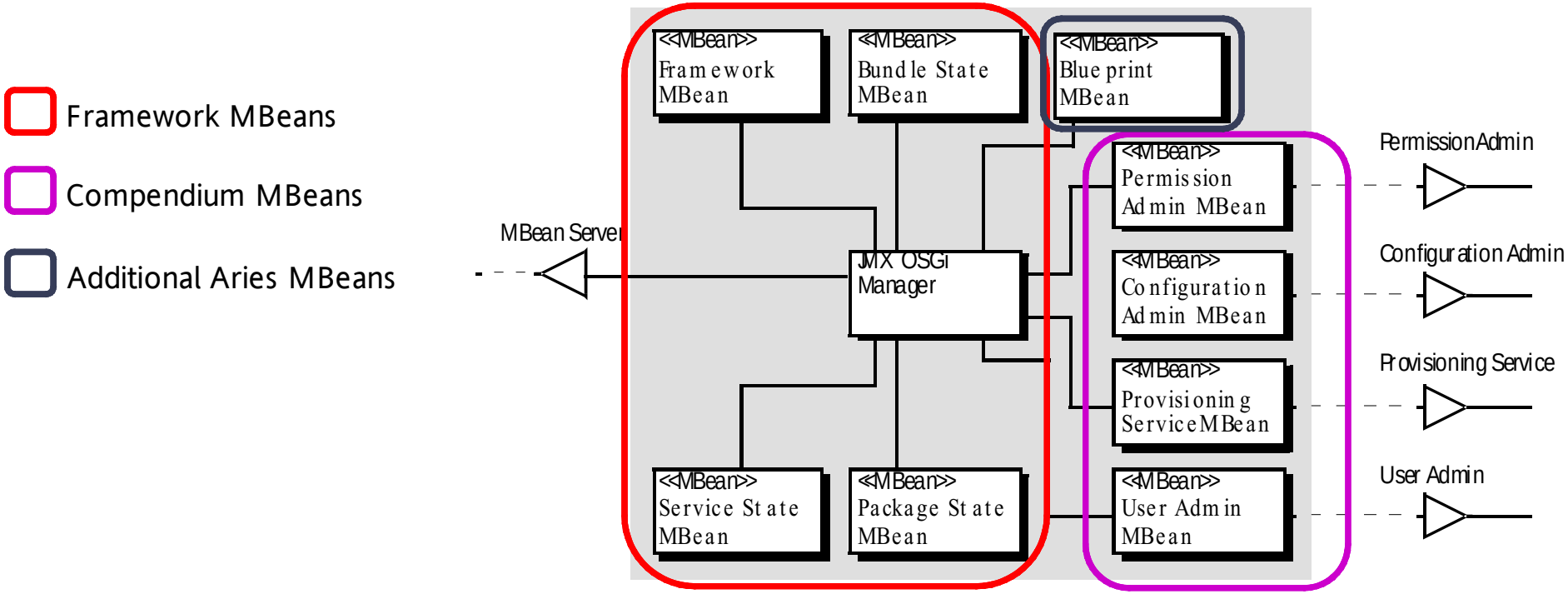


- A way for a Web component to access a Blueprint component

```
InitialContext ic = new InitialContext();
BloggingService blog= ic.lookup("osgi:services/"
                + BloggingService.class.getName());
```

Aries JMX Integration

- Implementation of OSGi JMX specification.
- Aries JMX bundle automatically registers the JMX MBeans into any javax.management.MBeanServer service in the OSGi Service Registry.

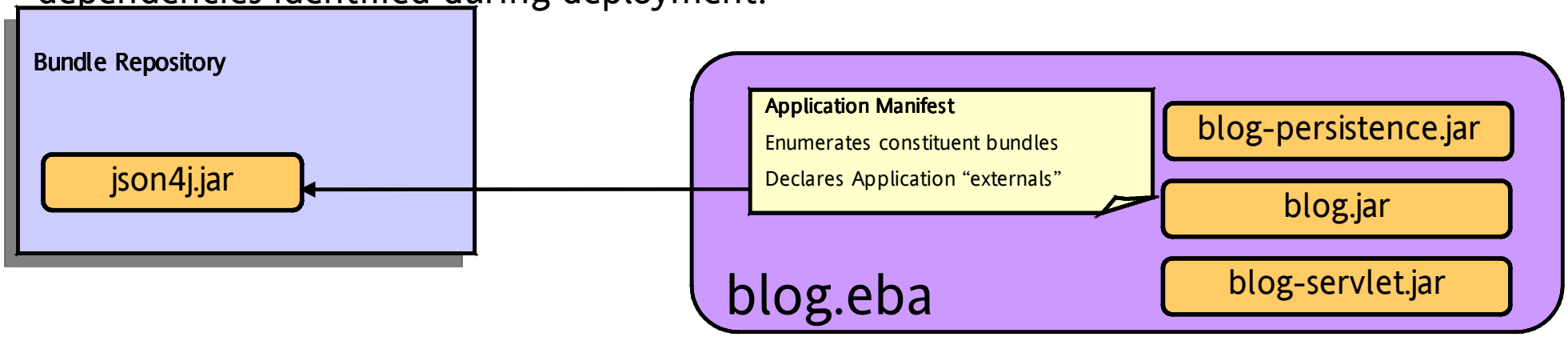


Web Application Bundles

- Bundles are recognized as Web applications if they contain a standard OSGi manifest header to identify the context path
  - Web-ContextPath:
- A hosting platform
  - Listens for bundles starting & stopping with WAB content
  - Hands the web components of to the Web Container to manage.

Aries Application Assembly and Deploy

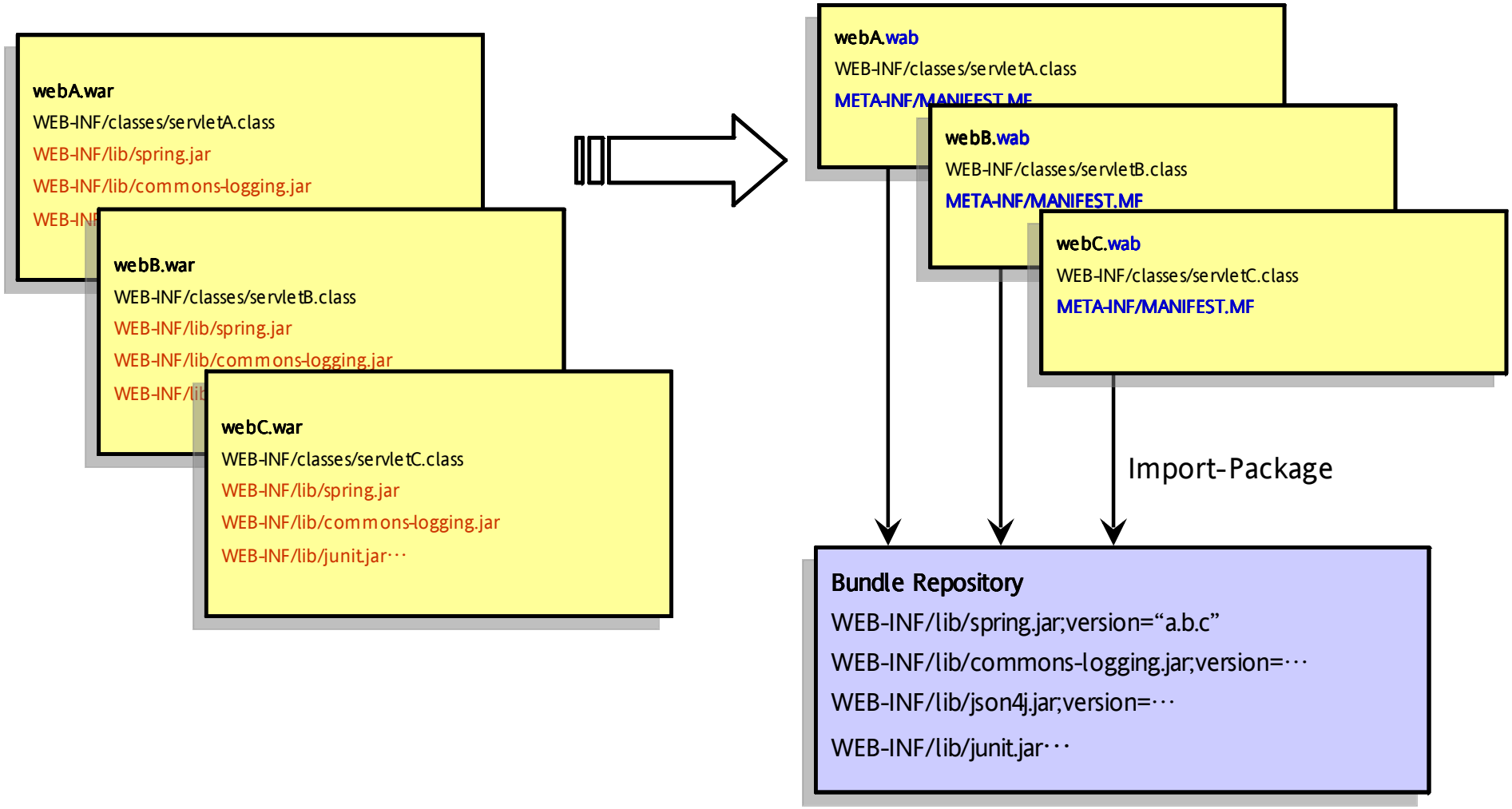
- Pluggable infrastructure for deployment of an application
  - A collection of bundles forming a logical unit
  - Packaged as an enterprise bundle archive (archive with .eba extn).
- An “Enterprise OSGi Application”.
- Constituent bundles may be contained (“by-value”) in the .eba archive or referenced in APPLICATION.MF
- Only explicitly declared Services are exposed from the application.
- 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.



Module Re-Use Made Easy

No Java code changes; war modules -> bundles

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



Bundle-level Management

WebSphere Administrative Console

Integrated Solutions Console Welcome

Help | Logout

View: All tasks

- Welcome
- Guided Activities
- Servers
- Applications
  - New Application
  - Application Types
    - WebSphere enterprise applicatio
    - Business-level applications
    - Assets
- Services
- Resources
- Security
- Environment

Cell=irobinsNode01Cell, Profile=AppSrv01

Close page

Assets

Assets > Update bundle versions in this application

Update the versions of the bundles that comprise this application.

Symbolic name	Content type	Provisioning	Sharing	Deployed version	New version
com.ibm.ws.eba.example.blog.web	Bundle	Declared	Isolated	1.0.0	No preference
com.ibm.json.java	Bundle	Inferred	Shared	1.0.0	No preference
com.ibm.ws.eba.example.blog.persistence	Bundle	Declared	Isolated	1.0.0	No preference 1.0.0
com.ibm.ws.eba.example.blog	Bundle	Declared	Isolated	1.0.0	No preference

Preview Cancel

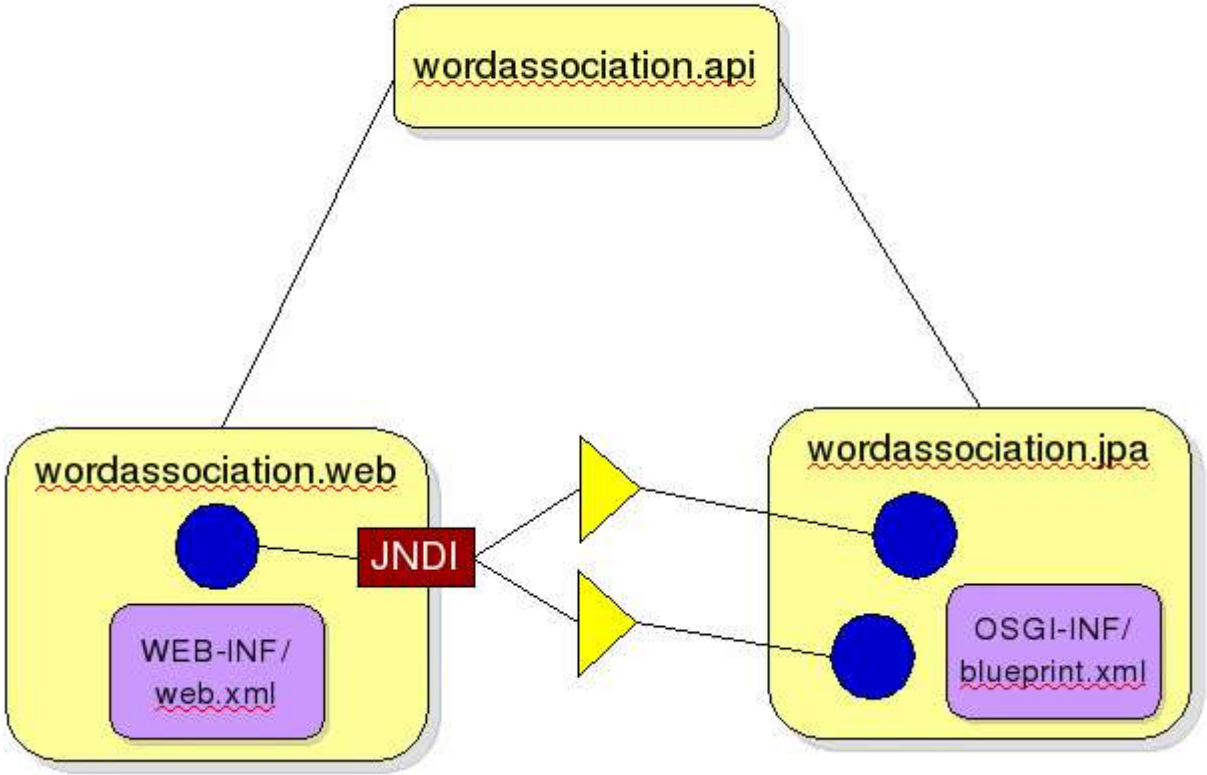
Demo

On luck …

- “I'm a great believer in luck and I find the harder I work, the more I have of it.”
  - Thomas Jefferson

Demo (maybe)

Demo (maybe)


wordassociation.api
wordassociation.web
JNDI
WEB-INF/
web.xml
wordassociation.jpa
OSGI-INF/
blueprint.xml

word association - Mozilla Firefox: IBM Edition

File Edit View History Bookmarks Tools Help

http://www.wordassociation.org/

word associati

Most Visited IBM

play | about this site | site stats | search

word association

it's your turn.

warning: this product may contain naughty words.

"jocelyn"

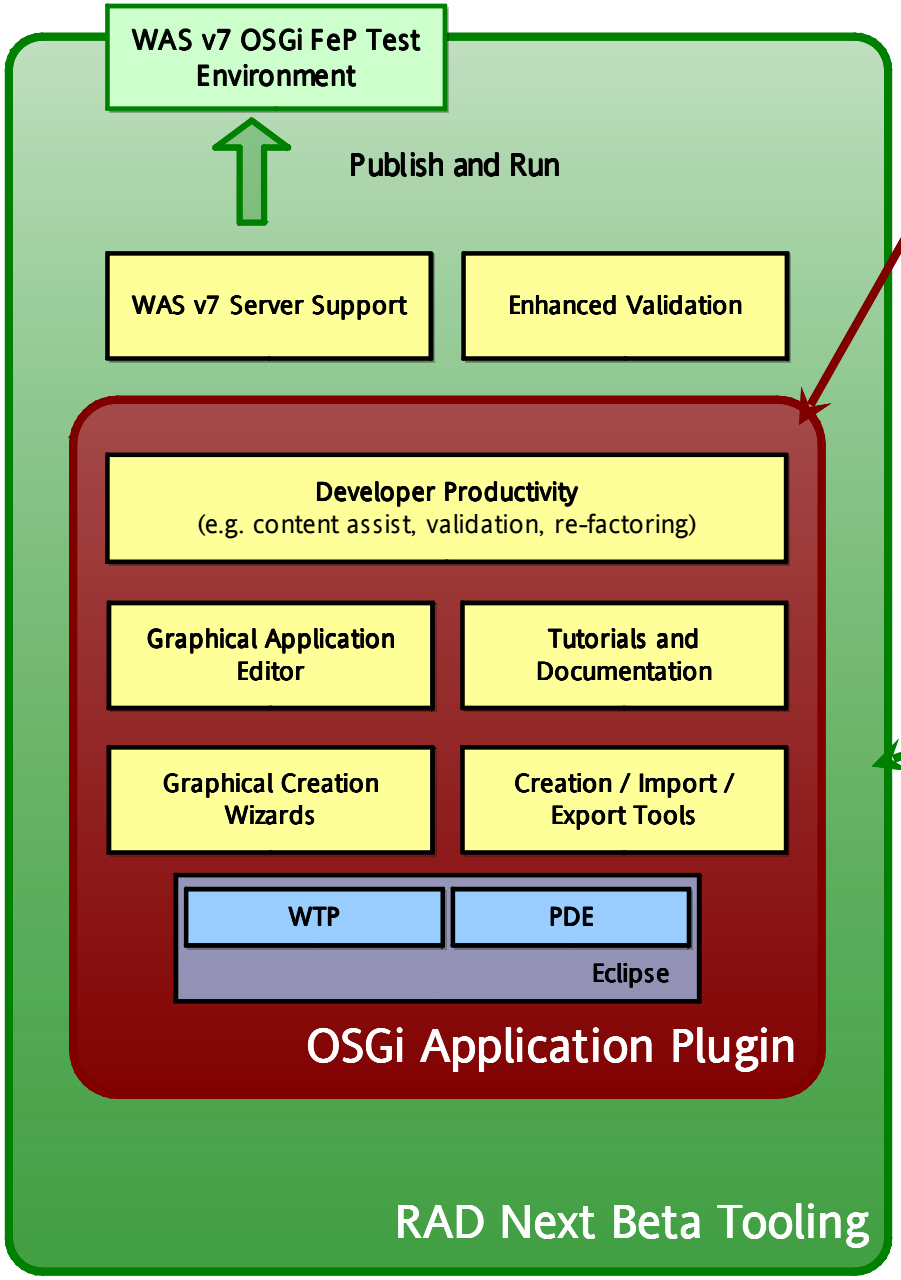
enter the **first single word** you think of …

play or pass

enable content filter

Done

Free tools



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 Beta

- 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

https://www14.software.ibm.com/iwm/web/cc/earlyprograms/rational/radob/index.shtml

Resources

- Apache Aries
  - http://incubator.apache.org/aries/
- Free tools
  - http://www.ibm.com/developerworks/rational/downloads/10/rationaldevtoolsforosgiapplications.html
- IBM WebSphere Feature Pack for OSGi Applications and Java Persistence API 2.0
  - http://www-01.ibm.com/software/webservers/appserv/was/featurepacks/osgi/

In the near future…
all major Java
enterprise runtimes
will support Apps
deployed as bundles

Any questions?

Futures

- There are many new application-centric features that Aries may develop including:
  - message-driven blueprint components and services
  - declarative role-based security for blueprint components
  - annotation-based alternative to XML configuration
  - resource-reference metadata and bindings

    (Original Proposal: http://wiki.apache.org/incubator/AriesProposal)

- Interested in getting involved?
  - http://incubator.apache.org/aries/gettinginvolved.html