


# Karaf - When OSGI modularity meets Java EE world

Charles Moulliard  
&  
Romain Manni-Bucau

# Agenda

- Modularity
  - OSGI EE
  - Java EE
  - All together
  - Conclusion
- 

# Speaker : Charles Moulliard

- Engineer in Agronomy & Master in Zoology
- Project Manager & Architect for banking sector
- Solution Architect & Consultant @RedHat
- Committer → Karaf (PMC), Camel (PMC), ServiceMix, DeltaSpike, Fabric

 @cmoulliard

 <http://www.linkedin.com/in/charlesmoulliard>

 <http://cmoulliard.blogspot.com>

 <http://www.slideshare.net/cmoulliard>



# Speaker : Romain Manni-Bucau


- Java and JavaEE developer
- Participates to Apache OpenEJB & TomEE since 2011
- His professional area is particularly linked to banks and insurances
- His development environment mainly consists of Open Source & Apache projects (TomEE, Karaf, ...)

 Suivre @rmannibucau

# Introduction to Modularity



# What does modularity means ?

- Open Standard Gateway Initiative exists since 1999
  - Initially conceived to create dynamic environments for the “home gateway”
  - Lead by Telco, IT vendors
  - Class-loading isolation
  - Dependency control, versioning
- 

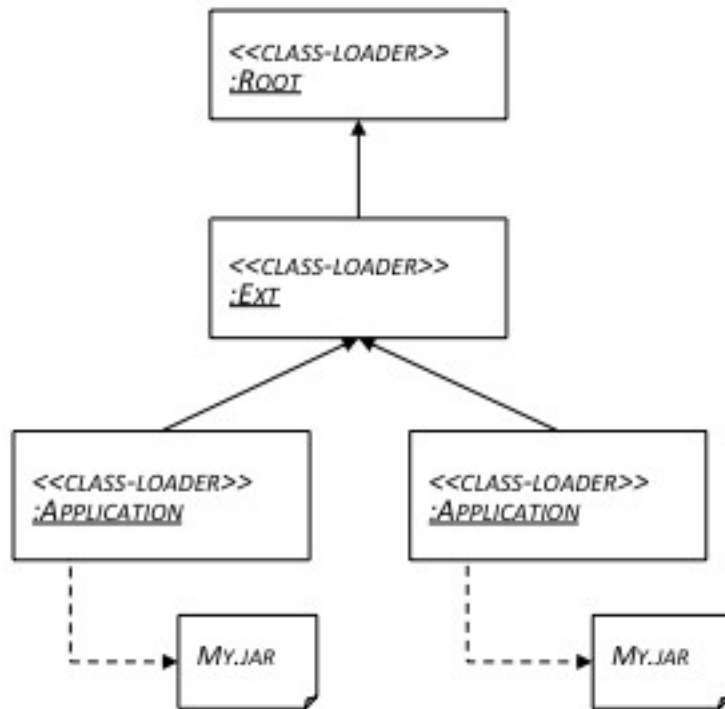
# What does modularity means ?

- Function → Class
- Class → Interface
- Classes → Jar
- Jar → Bundle (a Module)
- Bundle has a lifecycle !

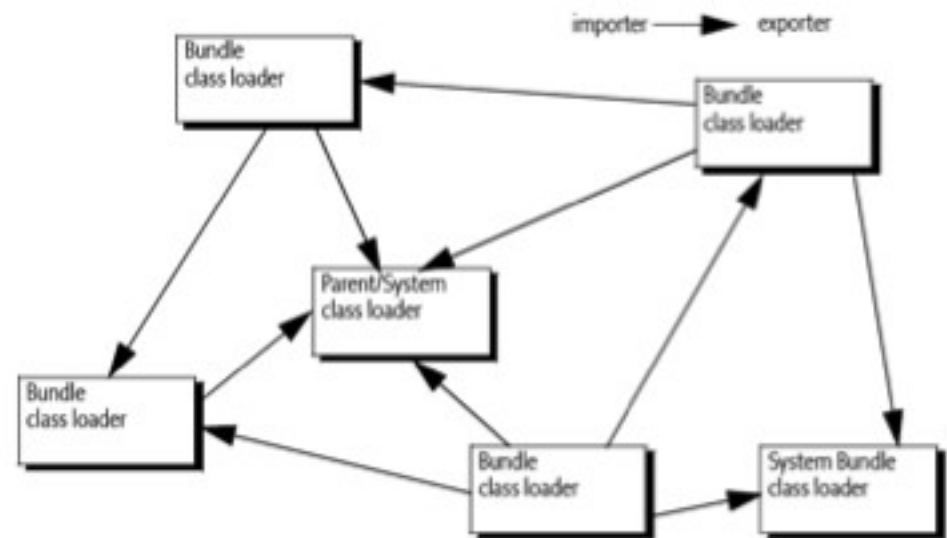


# What does modularity means ?

## JAVA ENTERPRISE EDITION TREE-BASED CLASSLOADING MODEL

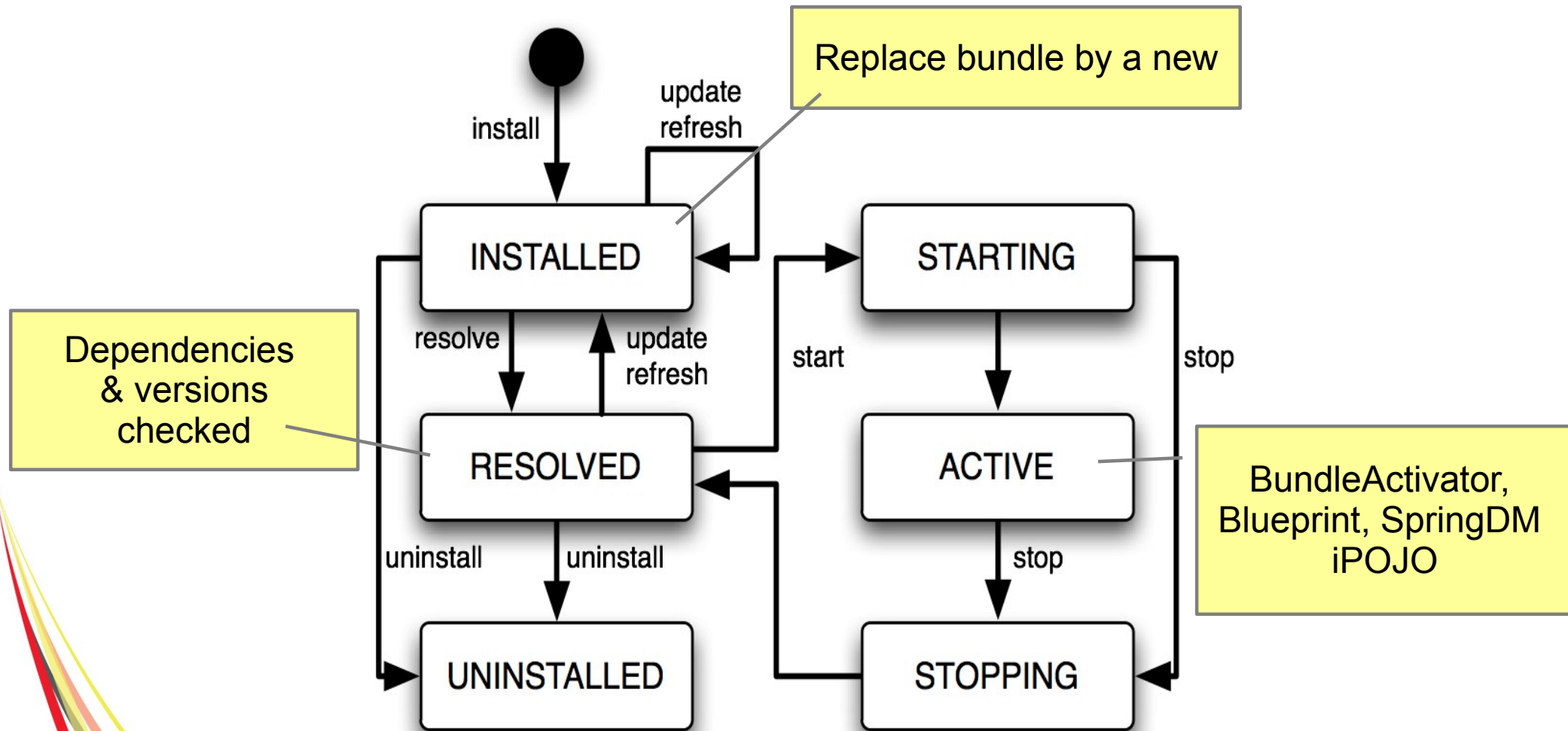


## OSGI GRAPH-BASED CLASSLOADING MODEL



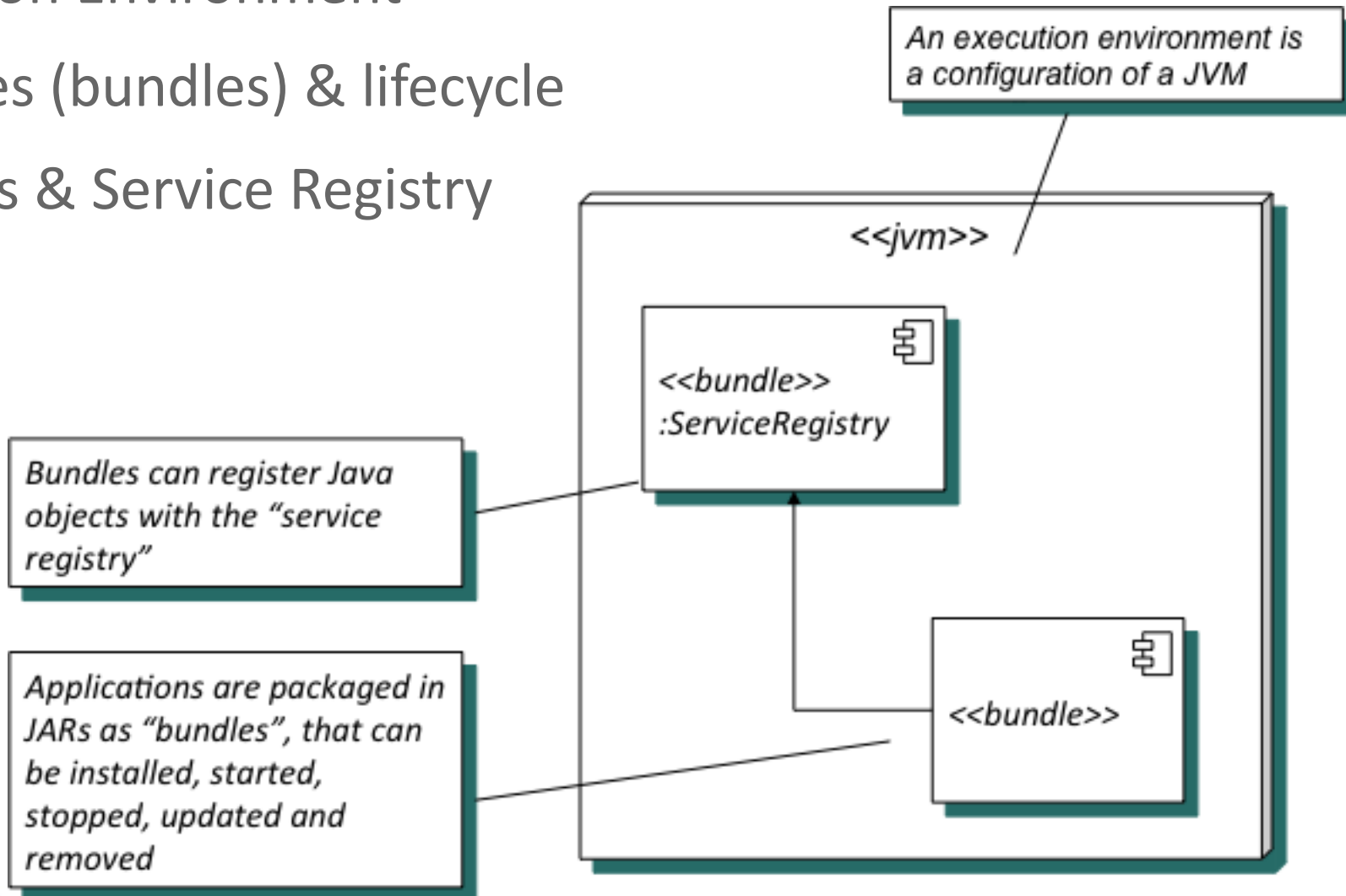


# What does modularity means ?



# What does modularity means ?

- OSGI Architecture introduces core concepts:
  - Execution Environment
  - Modules (bundles) & lifecycle
  - Services & Service Registry



# What does modularity means ?

- How modularity is expressed
  - METADATA added in MANIFEST.mf file

```
Bundle-Vendor = FuseSource Community
Bundle-Name = camel-jdbc
Bundle-DocURL = http://fusesource.com
Bundle-Description = Camel JDBC support
Bundle-SymbolicName = org.apache.camel.camel-jdbc
Bundle-Version = 2.9.0.fuse-70-097
Bundle-License = http://www.apache.org/licenses/LICENSE-2.0.txt
Bundle-ManifestVersion = 2
```

# What does modularity means ?

- And of course : Import & Export Package

```
Import-Package =  
    javax.sql,  
    org.apache.camel;version="[2.9,2.10)",  
    org.apache.camel.impl;version="[2.9,2.10)",  
    org.apache.camel.util;version="[2.9,2.10)",  
    org.slf4j;version="[1.6,2)"  
Export-Package =  
    org.apache.camel.component.jdbc;  
        uses:="org.apache.camel.impl,  
            org.apache.camel.util,  
            javax.sql,  
            org.apache.camel,  
            org.slf4j";  
    version=2.9.0.fuse-70-097
```

# What does modularity means ?

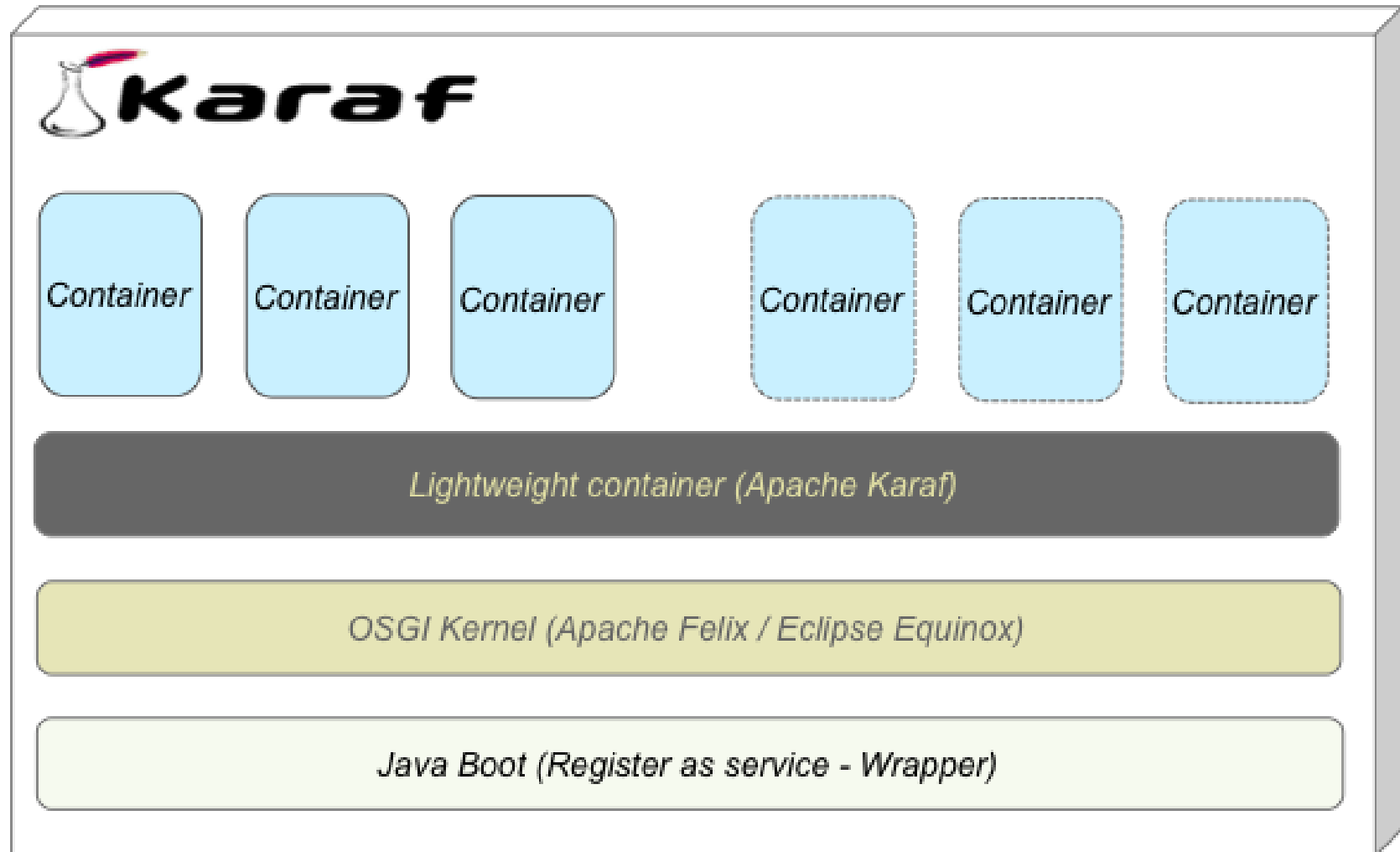
- TOTAL CONTROL OVER LIBRARY VERSIONS AND DEPENDENCIES**, different versions of the same library will happily coexist in the same container
- DECOUPLE SYSTEM AND APPLICATION MODULES BETWEEN EACH OTHER**, through an Service-Oriented Architecture inside your server to link your components together
- SHARE CLASSES AND DEPENDENCIES**, for increased efficiency in memory management and a lower runtime resource footprint
- INCREASE DEVELOPMENT AGILITY, TESTING AND VERSION UPDATING**, via quick hot deployment, restart, refresh operations on modules and libraries
- STOP SAYING "IT WORKS – DON'T TOUCH IT!"**; overcome brittleness by being able to fully inspect what packages and versions are being used where... during run-time!
- ISOLATE CHANGES WITHIN YOUR COMPONENTS**, so that they don't have an impact on the rest of the system
- MAKE YOUR SERVER GO ON A DIET**. OSGi = modularity, also on the server. Deploy exactly the modules and frameworks you'll need, and no more.

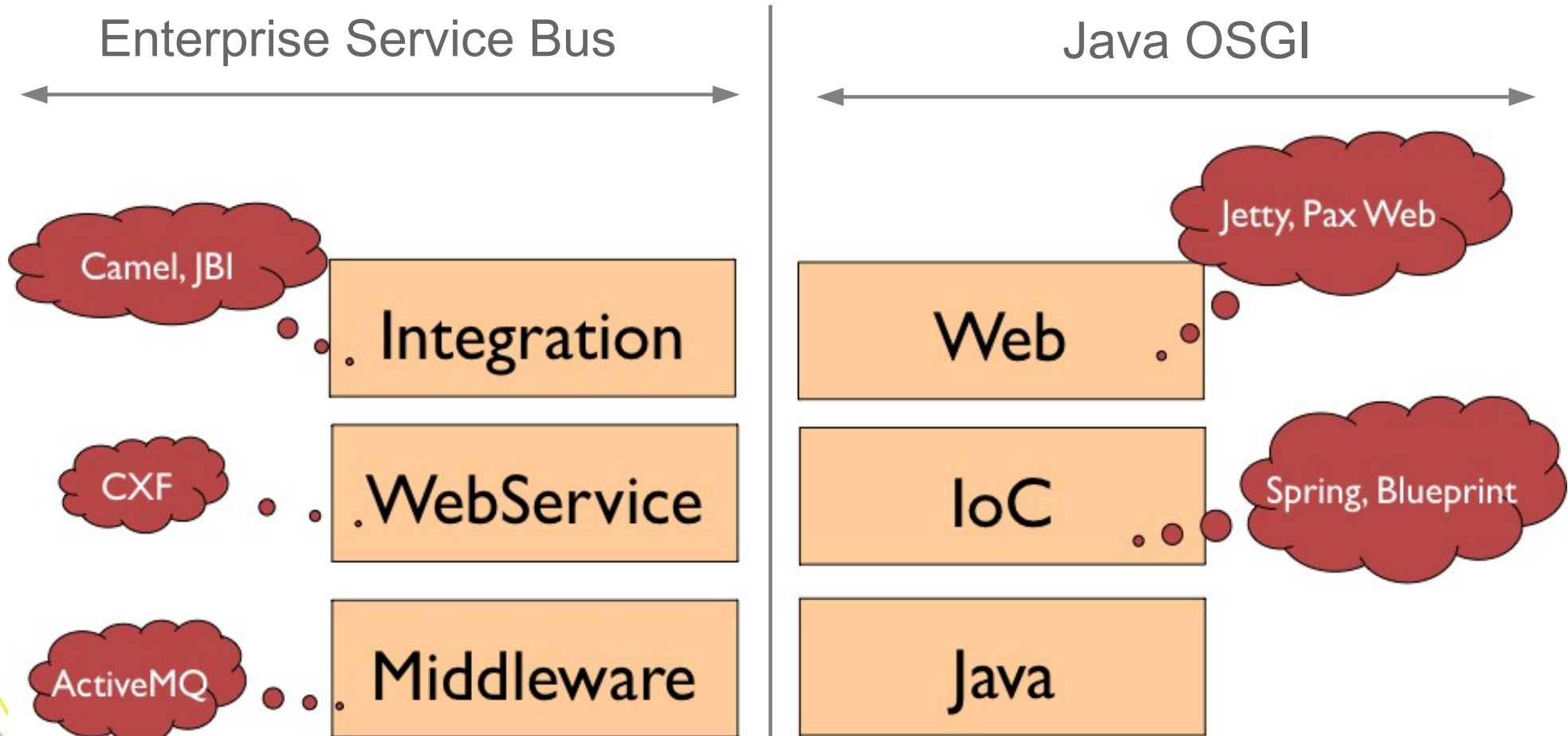


# Apache Karaf

- Birthdate - 16th of June 2010  **Karaf**
- Platform running on OSGI (Apache Felix, Eclipse Equinox)
- Provides a lightweight container where
  - Applications, Integration projects
  - Web Service, JMS Brokers
  - Java applications  
can be deployed
- Used by ServiceMix, Geronimo projects

# Apache Karaf







# Apache Karaf

- Administration console (locale, remote, ssh, web, jmx)
- Provisioning system (features)
- Hot deployment and configuration management (properties)
- Instances management
- Security integration (JAAS → ldap, jdbc, file)
- Logging unification (log4j, logger, commons logging, ....)

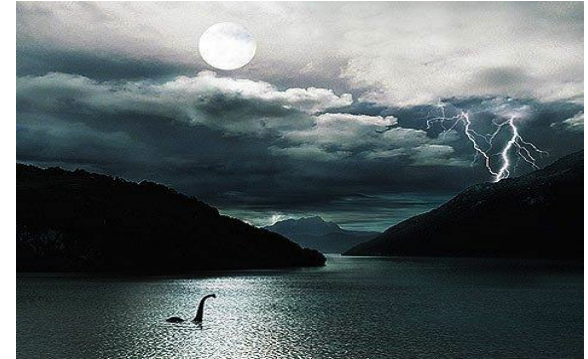
# Quick Apache Karaf demo

Is there an alternative ?

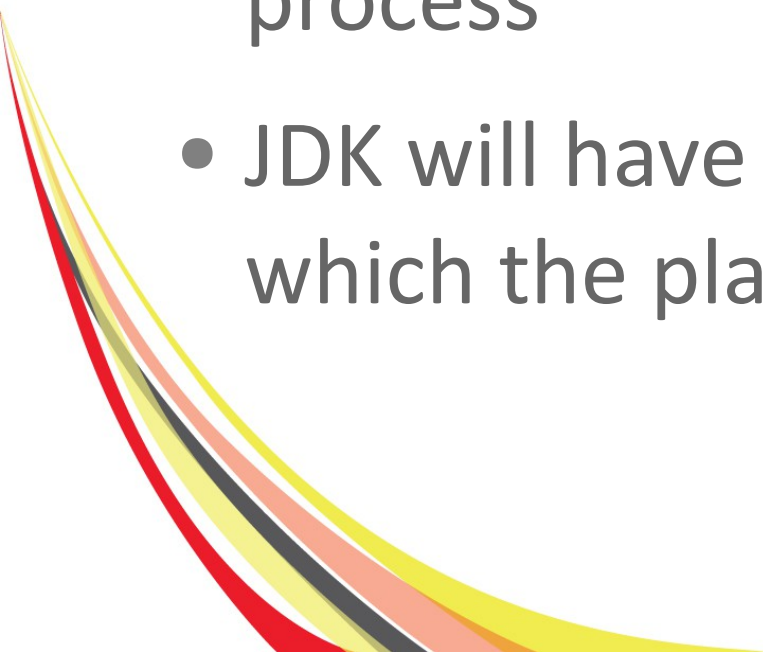


# Java Jigsaw vision

- Jigsaw project seems like Loch Ness Monster
- Announced since Java SE 7 & now -> Java SE 9 in 2015
- Faced to issues (backward compatibility)



# Java Jigsaw vision

- CLASSPATH death
  - New Class format
  - Static linkage ( $\leq$  Java card – 1996)
  - Byte code verification step moved to speed process
  - JDK will have a system module library in which the platform modules are installed
- 

# Java Jigsaw vision

- Dependency defined --> module-info.java file

```
module com.greetings @ 0.1 {  
    requires jdk.base;  
    requires org.astro @ 1.2;  
    class com.greetings.Hello;  
}
```

# Java Jigsaw vision

- Managed using specific commands

```
jmod -L mlib create # this creates module lib "mlib"
```

```
jmod -L mlib install modules org.astro com.greetings
```

```
rm -rf mlib/com.greetings
```

```
java -L mlib -m com.greetings
```

# OSGI EE

Extend Modularity with  
enterprise features





# OSGI EE

- OSGI is not only a specification defining a “modular” architecture
- Embrace EE world
- Provide support for Java standards :
  - JPA, JNDI, Web (HTTP), JTA, JDBC
  - Soon CDI, EJB

# OSGI EE

- Implemented by Apache Aries



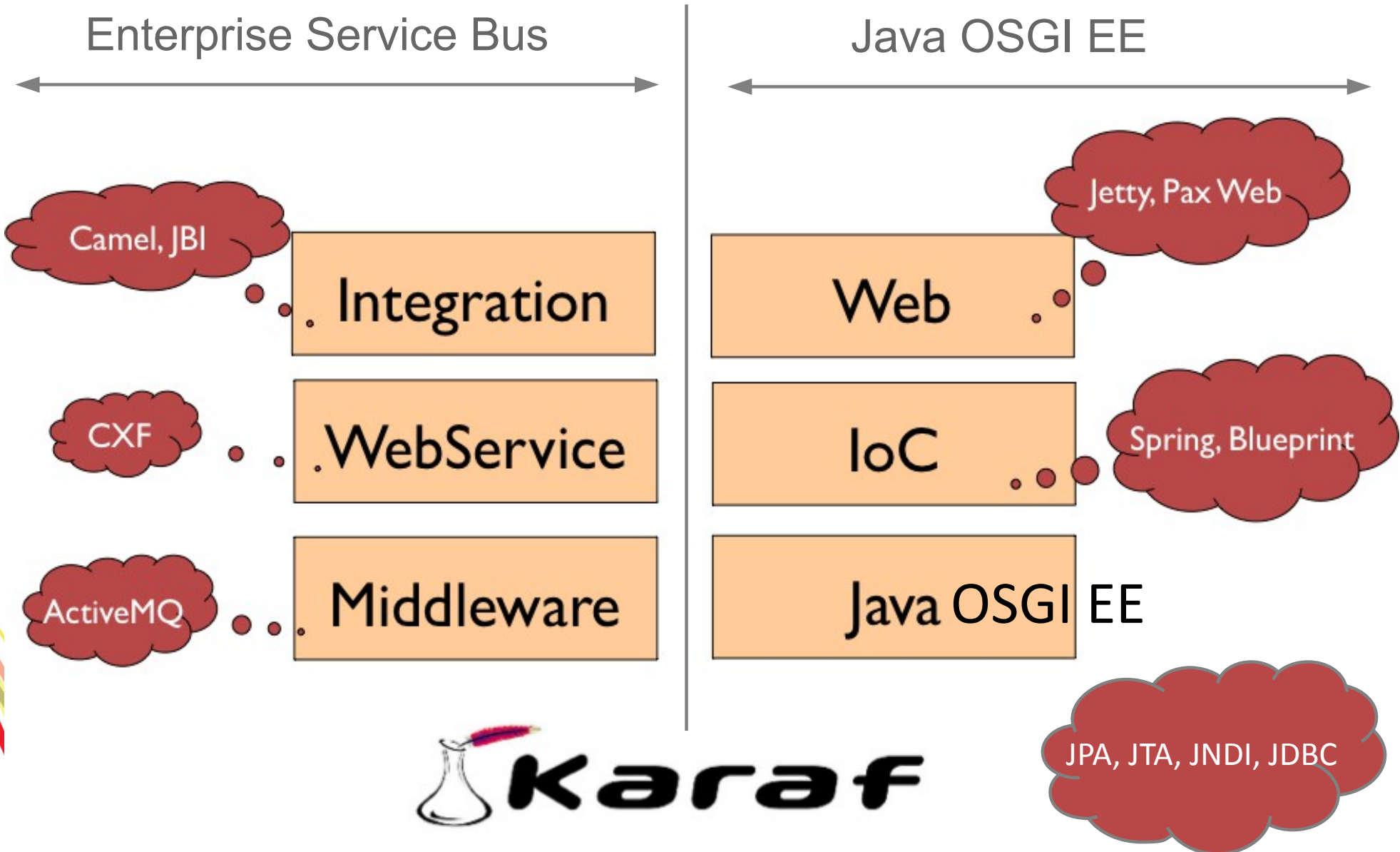
- But also OPS4J - Pax



# What is OSGI EE ?

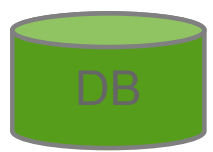
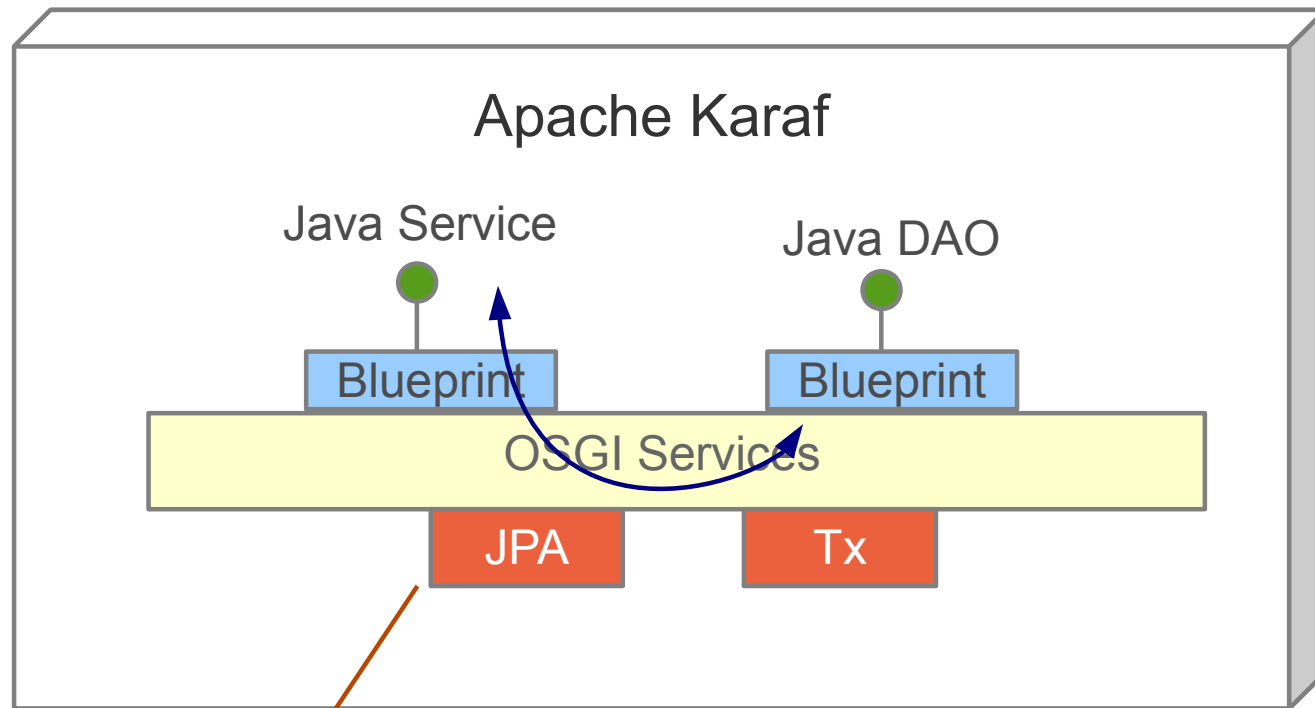
- OSGI services (Java Interfaces)
- JPA, JTA OSGI containers
- Web OSGI container
- Blueprint (IoC) to configure services, beans and access services
- But also
  - JNDI (wrapper)
  - SpiFly (Java ServiceLoader – static/dynamic)
  - Applications isolation (EBA)





# What is OSGI EE ?

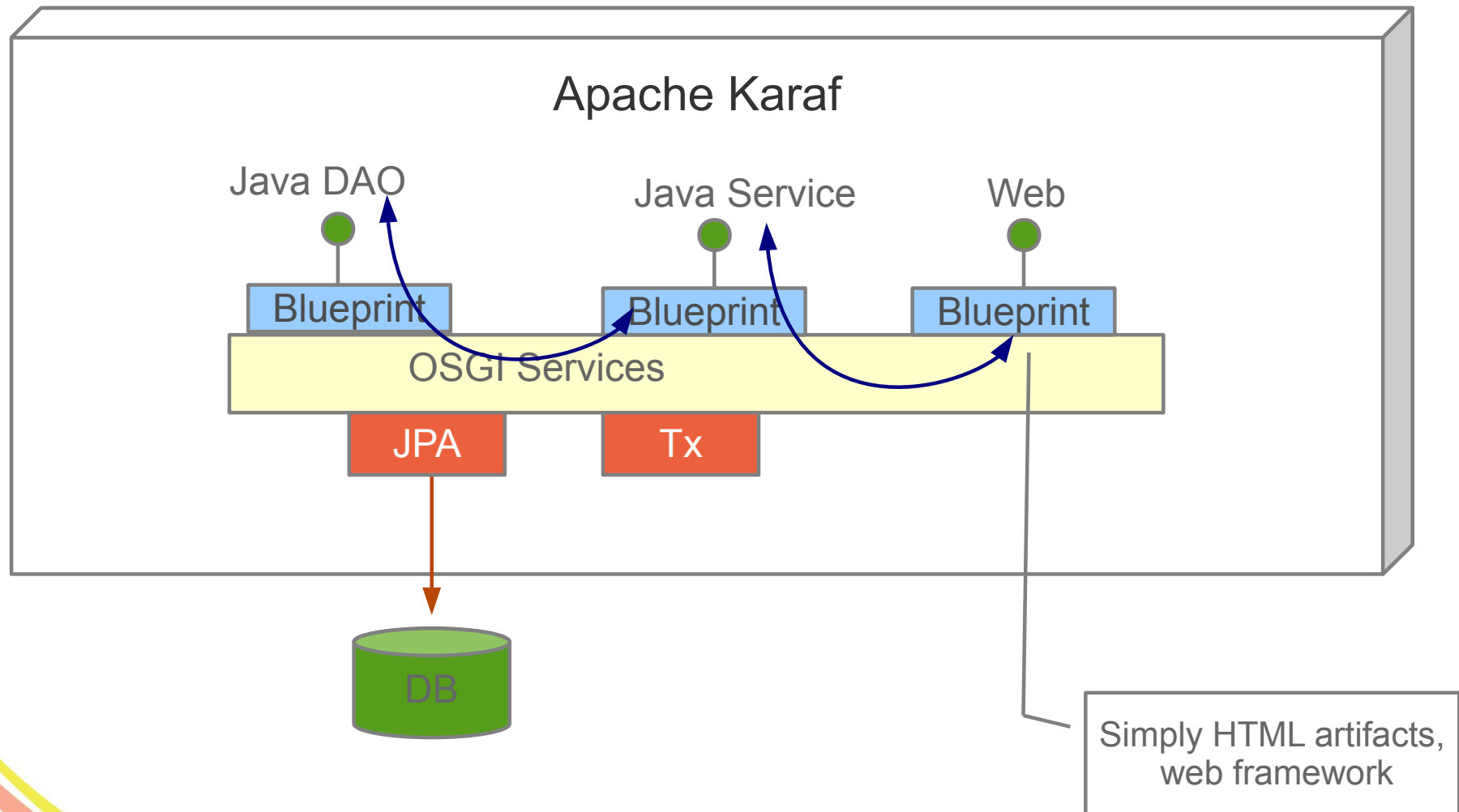
- A typical architecture



```
<bean id="conferenceService"  
class="org.apache.conf2012.service.impl.ConferenceService">  
  <property name="conferenceDAO">  
    <reference interface="org.apache.conf2012.service.ConferenceDAO"/>  
  </property>  
</bean>
```

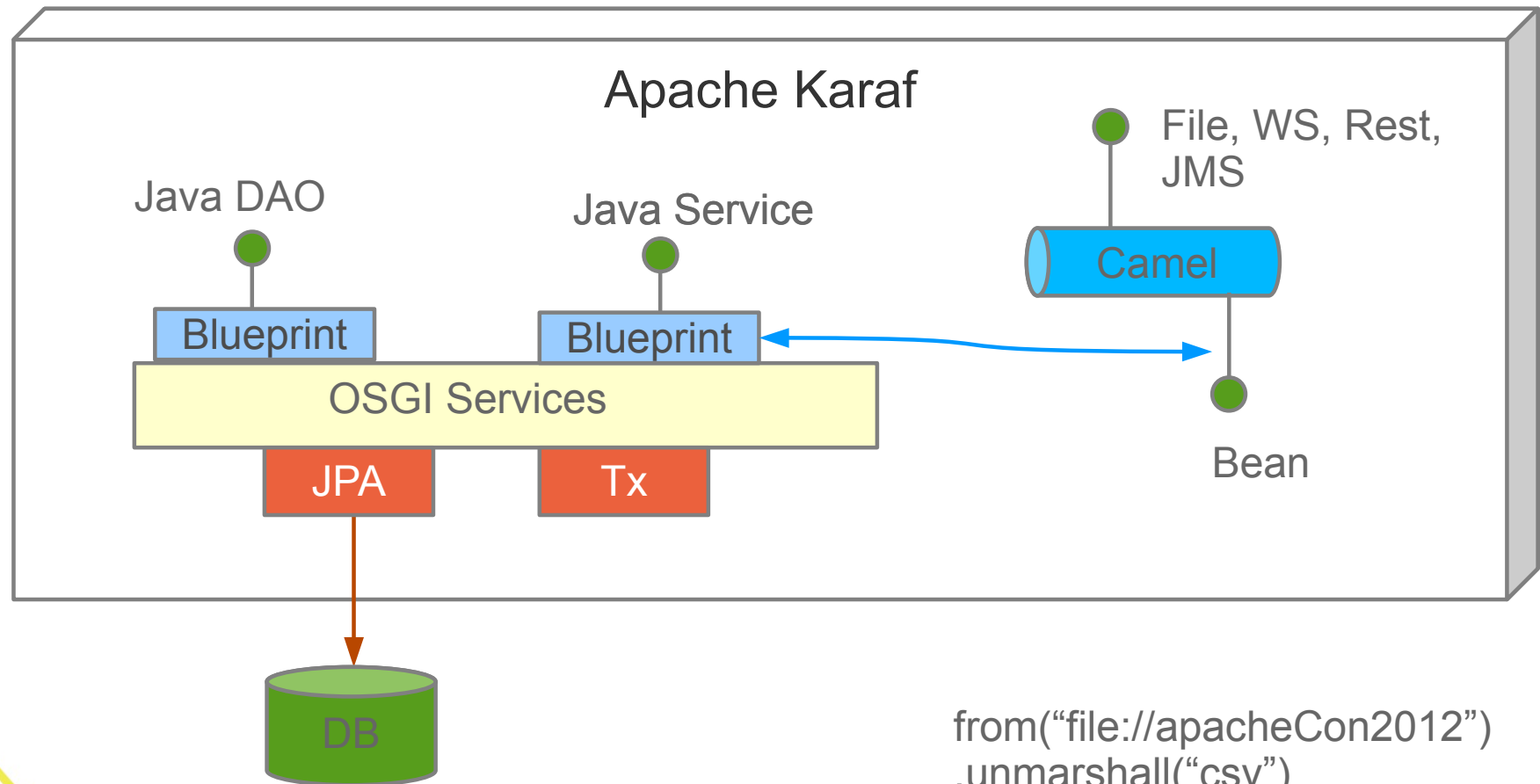
# What is OSGI EE ?

- Architecture enriched with Web layer



# What is OSGI EE ?

- Using Camel for Integration



```
from("file://apacheCon2012")  
.unmarshall("csv")  
.beanRef("insertConference");
```



# What is OSGI EE ?

- Persistence

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">

  <persistence-unit name="reportIncident" transaction-type="RESOURCE_LOCAL">

    <provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>
    <non-jta-data-source>reportincident</non-jta-data-source>
    <class>org.apache.con2012.karaf.ee.model.Incident</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>

    <properties>
      <property name="openjpa.jdbc.SynchronizeMappings"
        value="buildSchema(SchemaAction='add,deleteTableContents')"/>
      <property name="openjpa.Log"
        value="DefaultLevel=TRACE, Runtime=TRACE, Tool=TRACE, SQL=TRACE"/>
      <property name="openjpa.jdbc.DBDictionary"
        value="h2(useSchemaName=true)/>
      <property name="openjpa.jdbc.Schema"
        value="REPORT"/>
    </properties>

  </persistence-unit>
</persistence>
```



# What is OSGI EE ?

- DAO

```
public class IncidentDAO {  
  
    private static final Logger LOG = LoggerFactory.getLogger(IncidentDAO.class);  
  
    @PersistenceContext(unitName = "reportIncident", type = PersistenceContextType.EXTENDED)  
    private EntityManager em;  
    private static final String findIncidentByReference = "select i from Incident as i where i.incidentRef = :ref";  
    private static final String findIncident = "select i from Incident as i";  
  
    public List<Incident> findIncident() {  
        Query q = this.em.createQuery("select i from Incident as i");  
  
        List list = q.getResultList();  
  
        return list;  
    }  
  
    public List<Incident> findIncident(String key) {  
        Query q = this.em.createQuery("select i from Incident as i where i.incidentRef = :ref");  
        q.setParameter("ref", key);  
        List list = q.getResultList();  
  
        return list;  
    }  
  
    public Incident getIncident(long id) {  
        return (Incident)this.em.find(Incident.class, Long.valueOf(id));  
    }  
}
```

# What is OSGI EE ?

- DAO (Spring/Blueprint) using JPA/Hibernate

```
<context:annotation-config/>
```

```
<!-- enables interpretation of the @PersistenceUnit/@PersistenceContext annotations providing convenient access to EntityManagerFactory/EntityManager -->
```

```
<bean class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor"/>
```

```
<!-- DAO Declarations -->
```

```
<bean id="incidentDAO" class="org.fusesource.devoxx.reportincident.dao.impl.IncidentDAOImpl"/>
```

```
<!-- Expose DAO interface as OSGI Service -->
```

```
<osgi:service ref="incidentDAO" interface="org.fusesource.devoxx.reportincident.dao.IncidentDAO"/>
```

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
```

```
  <property name="persistenceUnitName" value="reportIncident"/>
```

```
  <property name="jpaVendorAdapter" ref="jpaAdapterH2"/>
```

```
  <property name="dataSource" ref="dataSourceH2"/>
```

```
</bean>
```

# What is OSGI EE ?

- Service

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

  <bean id="incidentDAO" class="org.apache.con2012.karafee.dao.IncidentDAO"/>

  <bean id="incidentServiceBean" class="org.apache.con2012.karafee.service.impl.IncidentServiceImpl">
    <property name="incidentDAO" ref="incidentDAO"/>
  </bean>

  <!-- Expose the service (= interface) using OSGI Service -->
  <service id="service" ref="incidentServiceBean" interface="org.apache.con2012.karafee.service.IncidentService"/>

</blueprint>
```

# Java EE



# Java EE - Specs

- JSR 907 (JTA)
- JSR 196 (JAAS)
- JSR 115 (JACC)
- JavaMail
- JSR 338 (JPA)
- JSR 303 (Bean Validation)
- JSR 299 + 330 (CDI)
- JSR 318 (EJB)



OpenWebBeans



# Java EE – Specs (next)

- JSR\_315 (servlet)
- JSR 245 (JSP)
- JSR 344 (JSF)
- JSR 914 (JMS)
- JSR 339 (JAX-RS)
- JSR 224 (JAX-WS)
- Connector



**Apache CXF**

**ActiveMQ**



# Java EE & Apache

- Supported by Apache foundation projects

OpenJPA



OpenEJB



OpenWebbeans



DeltaSpike



ActiveMQ



CXF, Camel

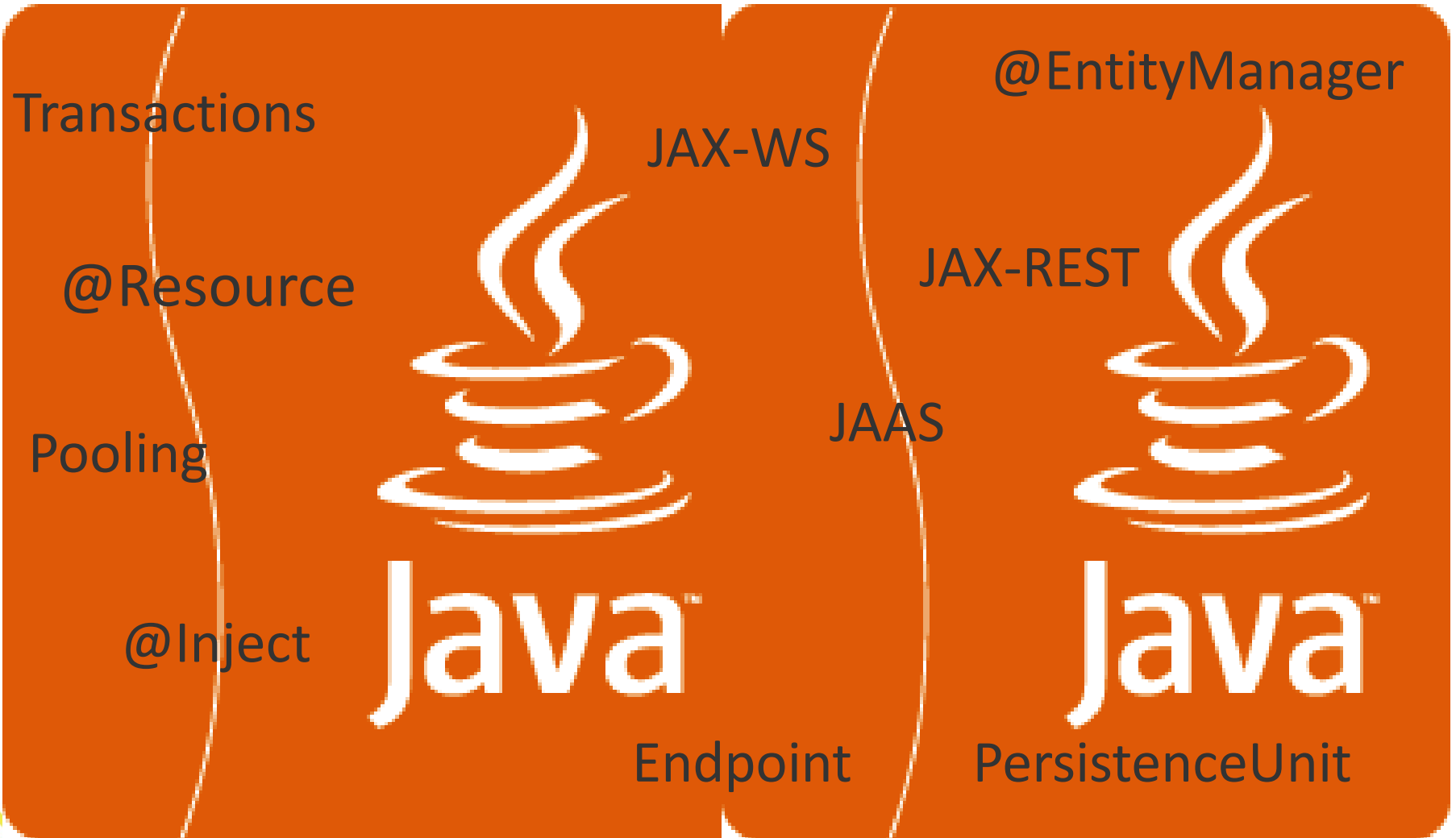


Tomcat, MyFaces





# Java EE - Concepts





# When Java EE meets OSGI

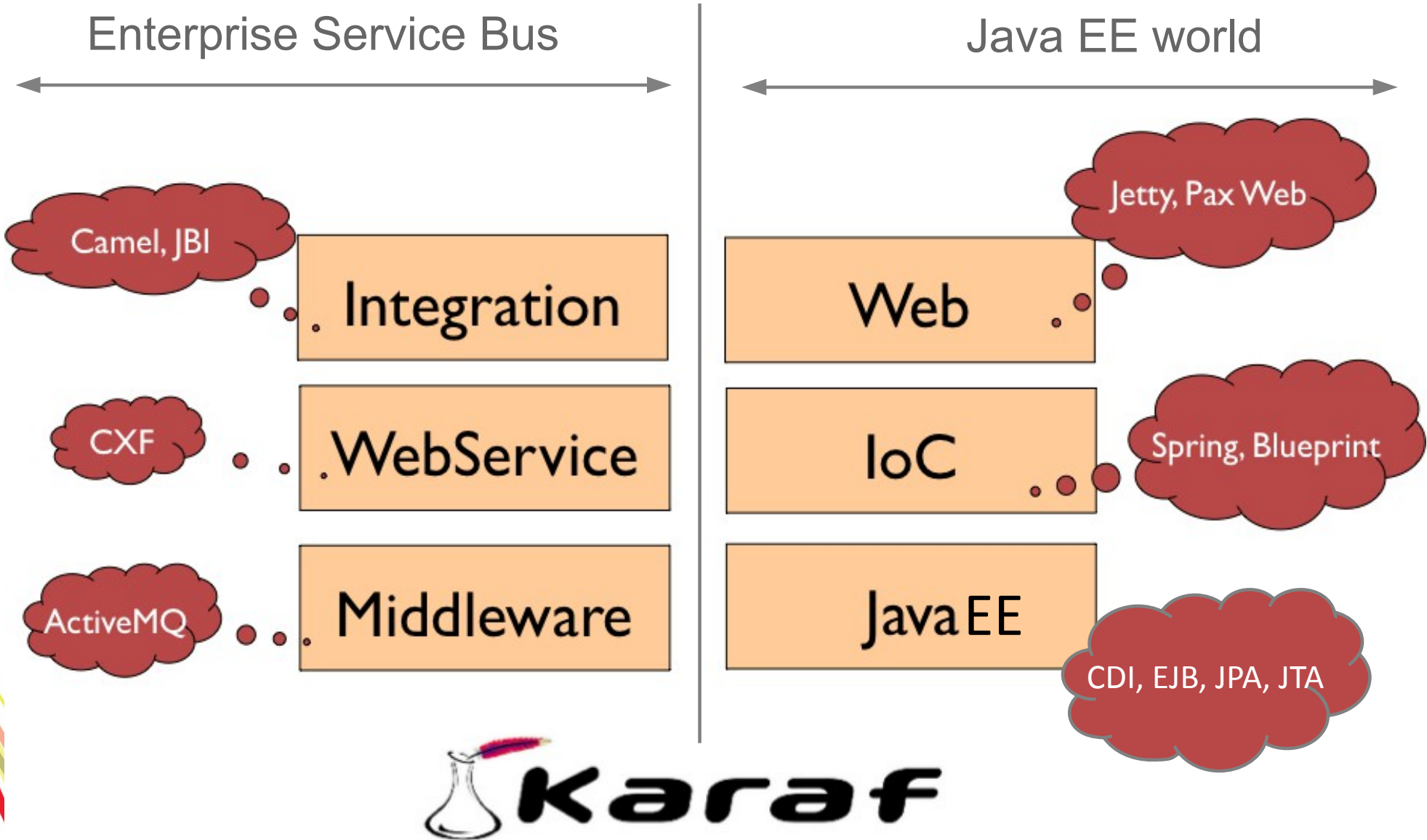


# Java EE on Karaf

- What is required
  - A Karaf runtime
  - Features (= provisioning)
  - New Karaf command (optional)
- Follow Java EE specs & development guidelines

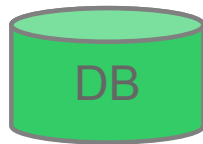
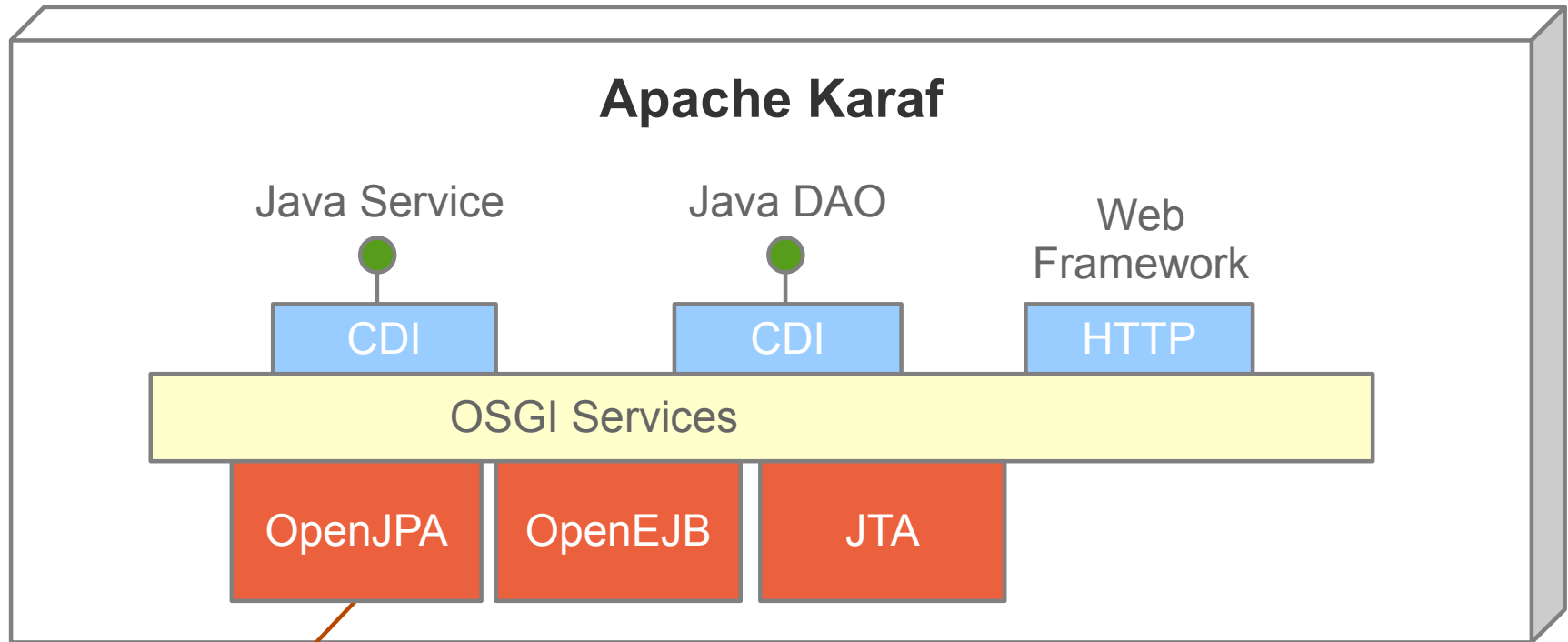
# Karaf plus EE

- Features →  
Java EE api + OpenEJB + OpenWebbeans +  
OpenJPA
- Turn on your Karaf into Java EE world :  
CDI + EJB3 + JPA + JTA + ...



# Karaf EE Architecture

- Architecture (JPA, EJB, CDI)



...  
@Inject  
@Startup  
@Singleton  
Conference DAO MyConferenceDAO  
....

# Karaf with EE - Example

- Persistence

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
  version="2.0">

  <persistence-unit name="apachecon2012">
    <jta-data-source>jdbc/apachecon2012</jta-data-source>
    <class>org.apache.con2012.karafee.model.Conference</class>
    <class>org.apache.con2012.karafee.model.EntityWithToString</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>

    <properties>
      <!--
      <property name="openjpa.jdbc.SynchronizeMappings"
        value="buildSchema(SchemaAction='add,deleteTableContents')"/>
      -->
      <property name="openjpa.jdbc.SynchronizeMappings"
        value="buildSchema(ForeignKeys=true)"/>
      <property name="openjpa.Log"
        value="DefaultLevel=INFO, Runtime=INFO, Tool=INFO, SQL=TRACE"/>
      <property name="openjpa.jdbc.DBDictionary"
        value="h2(useSchemaName=true)"/>
      <!--
      <property name="openjpa.jdbc.Schema"
        value="APACHECON"/>
      -->

      <!-- force initialization at startup -->
      <property name="openjpa.jpa.init-entitymanager"
        value="true" />
    </properties>
  </persistence-unit>
</persistence>
```



# Karaf with EE - Example

- Entity

```
    @NamedQueries({
        @NamedQuery(name = "Conference.findAll", query = "select i from Conference i"),
        @NamedQuery(name = "Conference.findByKey", query = "select i from Conference i where i.ref = :key"),
        @NamedQuery(name = "Conference.countAll", query = "select count(i) from Conference i")
    })
    @Entity
    @Table(name = "T_CONFERENCE")
    public class Conference extends EntityWithToString implements Serializable {

        private static final long serialVersionUID = 1L;

        @Id
        @GeneratedValue(strategy= GenerationType.AUTO)
        @Column(name = "CONFERENCE_ID")
        private long id;

        @Column(name = "REF", length = 55)
        private String ref;

        @Column(name = "GIVEN_NAME", length = 35)
        private String givenName;
    }
```

# Karaf with EE - Example

- DAO

```
@Lock(LockType.READ)
@Singleton
public class ConferenceRepository {

    @PersistenceContext(unitName = "apachecon2012")
    private EntityManager em;

    @Inject
    private RepositoryMonitoring monitor;

    public Conference store(final Conference conference) {
        em.persist(conference);
        em.flush();
        monitor.inc();
        return conference;
    }

    public void delete(final long id) {
        em.remove(id);
    }

    public Conference findById(long id) {
        return em.find(Conference.class, id);
    }

    public List<Conference> findAll() {
        return em.createNamedQuery("Conference.findAll", Conference.class).getResultList();
    }

    public List<Conference> findAll(final int first, final int count) {
        return em.createNamedQuery("Conference.findAll", Conference.class)
            .setFirstResult(first)
            .setMaxResults(count)
            .getResultList();
    }
}
```



# Karaf with EE - Example

- Service

```
@Lock(LockType.READ)
@Singleton
public class ConferenceServiceImpl implements ConferenceService {

    private static final Logger LOG = LoggerFactory.getLogger(ConferenceServiceImpl.class);

    @Inject
    private ConferenceRepository conferenceRepository;

    public void store(Conference conference) {
        conferenceRepository.store(conference);
    }

    public void delete(long id) {
        conferenceRepository.delete(id);
    }

    @Override
    public long totalNumber() {
        return conferenceRepository.countAll();
    }

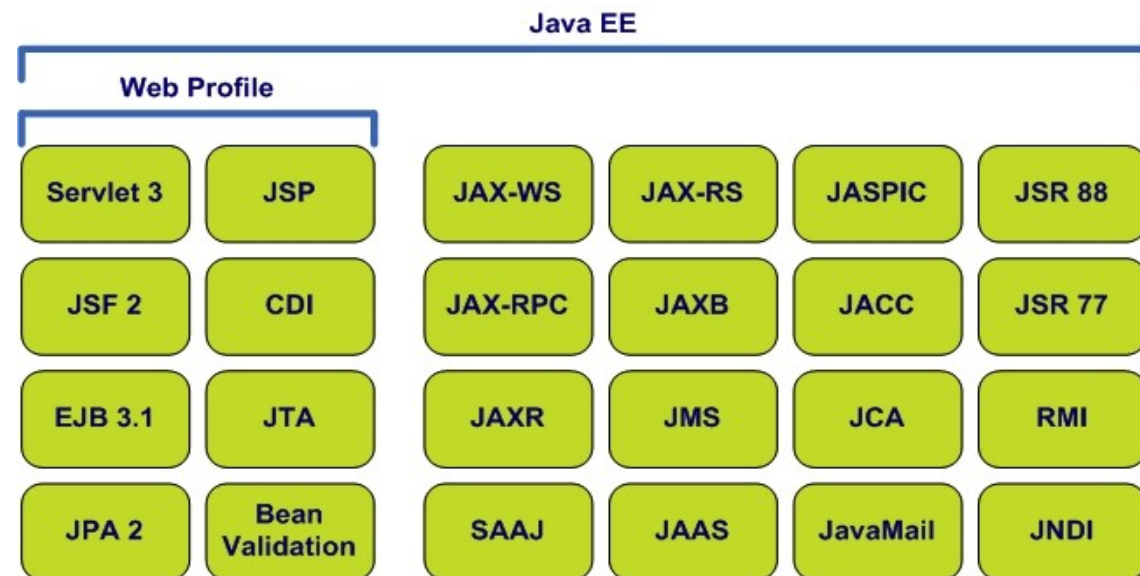
    public Conference findById(long id) {
        return conferenceRepository.findById(id);
    }

    public List<Conference> findAll() {
        return conferenceRepository.findAll();
    }
}
```

# DEMO & Conclusion



- Karaf EE = Combine multi-lightweight containers of Karaf with Java EE world
- Allow to develop modular projects
- Capitalize your Java EE knowledge
- Convergence between, best of both technologies OSGI & Java EE



# Questions



@cmoulliard  
@rmannibucau



Karaf with EE –

<http://repo1.maven.org/maven2/org/apache/openejb/apache-karaf-ee/>