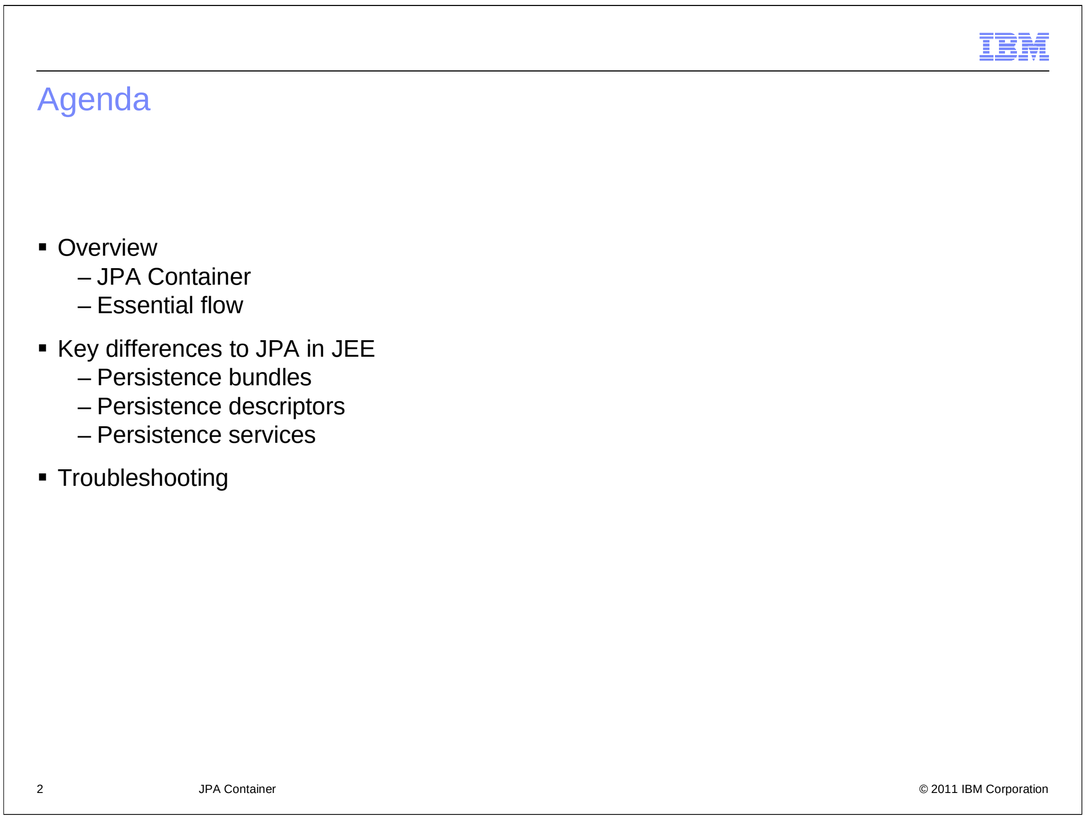
JPA Container

IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API 2.0

WebSphere software

This module provides education on the JPA container of the OSGi feature of the IBM WebSphere Application Feature Pack for OSGi Applications and Java Persistence API 2.0.

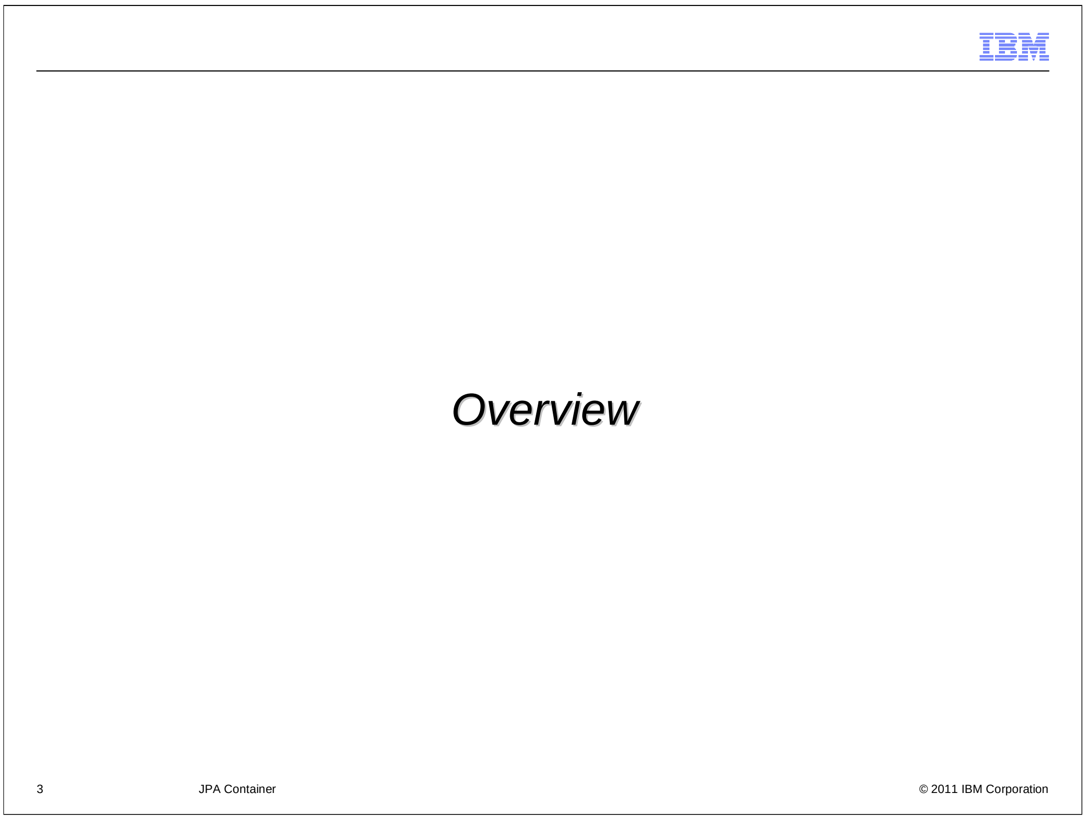

Agenda

- Overview
  - JPA Container
  - Essential flow
- Key differences to JPA in JEE
  - Persistence bundles
  - Persistence descriptors
  - Persistence services
- Troubleshooting

2 JPA Container © 2011 IBM Corporation

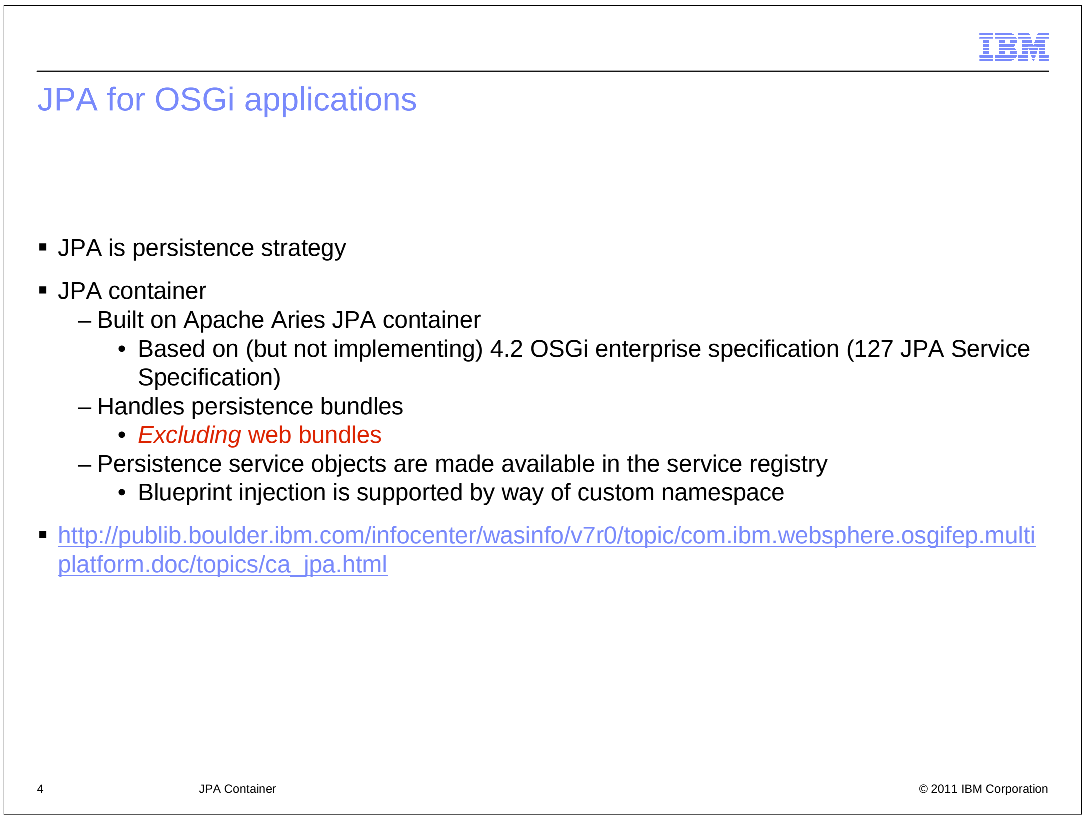
This presentation covers the JPA container in the OSGi applications feature pack. In essence JPA for an OSGi application works pretty much the same as JPA for a traditional JEE application.

As a result this talk is focused on the key differences between JPA for OSGi applications and traditional JPA, such as the notion of persistence bundles, additional capabilities in the persistence descriptor (persitence.xml) and finally the pure OSGi persistence services, which allow new usage patterns around JPA.



Overview

This section provides an overview.



JPA for OSGi applications

- JPA is persistence strategy
- JPA container
  - Built on Apache Aries JPA container
    - Based on (but not implementing) 4.2 OSGi enterprise specification (127 JPA Service Specification)
  - Handles persistence bundles
    - *Excluding* web bundles
  - Persistence service objects are made available in the service registry
    - Blueprint injection is supported by way of custom namespace
- http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.osgifep.multiplatform.doc/topics/ca_jpa.html

4 JPA Container © 2011 IBM Corporation

You can use JDBC with the OSGi feature but JPA is the strategic technology. If you come from a JPA in JEE or a Hibernate environment you will find this environment familiar.

Overall JPA for an OSGi bundle works almost exactly like JPA in a non-OSGi scenario. However, there are some subtle differences as regards to classloading. To address these differences the OSGi alliances has created the JPA Service Specification as part of the OSGi 4.2 Enterprise specification.

The specification at this time only defines application-managed JPA. However, the Apache Aries JPA container has been extended on the base of the OSGi spec to allow container-managed JPA while also not implementing the full specification. The JPA container for the OSGi applications feature pack has been built on top of the Apache Aries JPA container, using a plug-point specifically designed for this, to integrate with WebSphere Application Server and provide extra features like annotation processing.

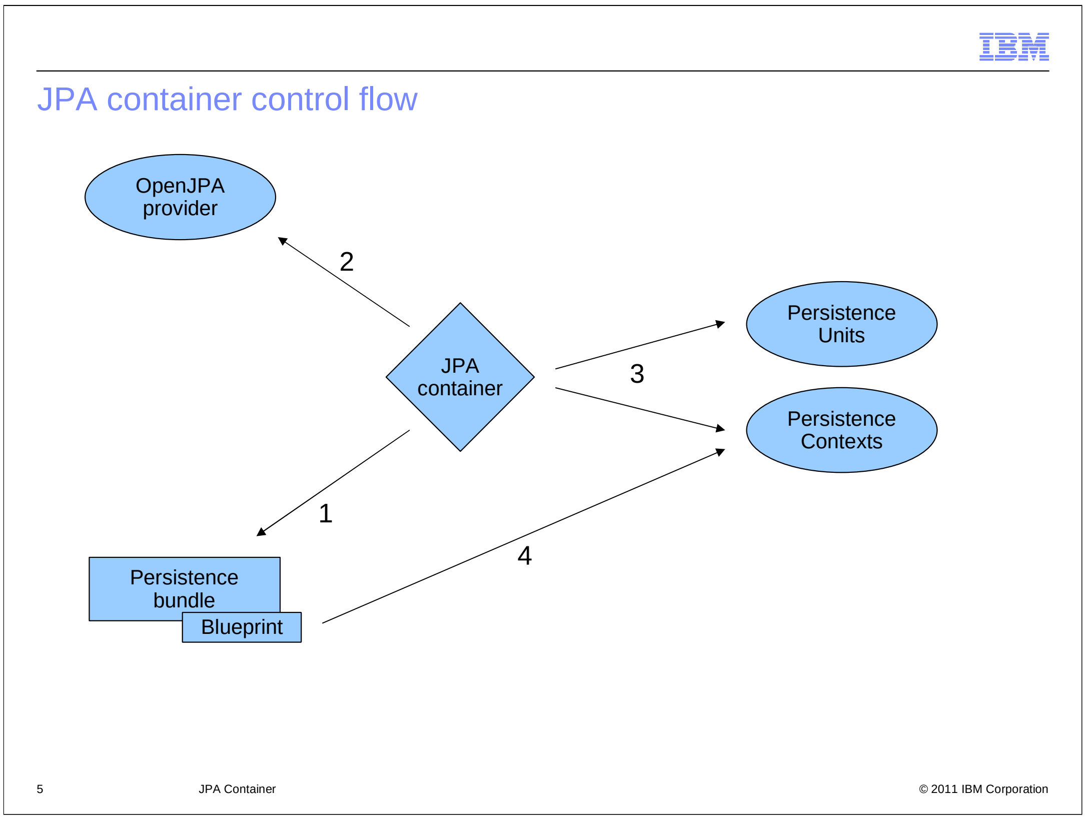
While what is shipped in this feature is based on the OSGi specification for JPA, the specification only deals with unmanaged JPA, that is JPA that doesn't have transaction, integration or injection facilities. With the feature pack you can make use of managed JPA.

The JPA container is all about persistence bundles, which are essentially plain bundles that contain entities and persistence descriptors) plus an additional manifest header. Note that web bundles are not allowed as persistence bundles in this release.

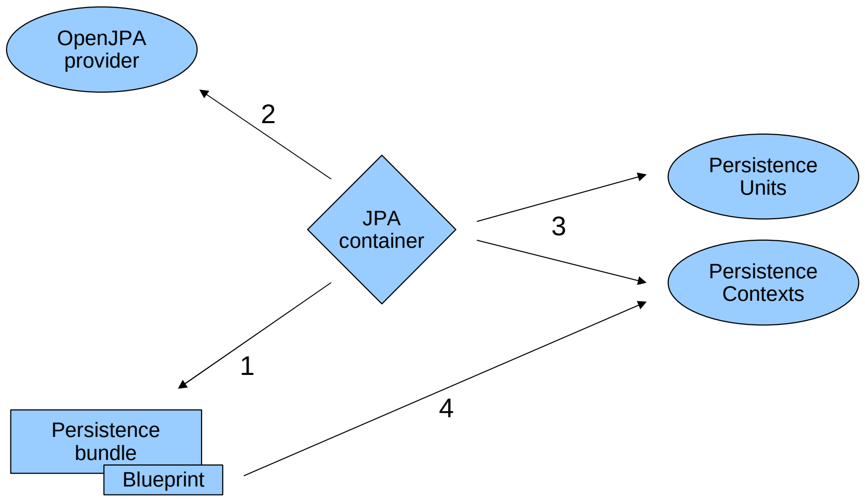
However, JPA in web applications is supported through the standard WebSphere Application Server webcontainer and JPA support.
It is important to note that there is a clear dividing line between a web bundle and a persistence bundle. A web bundle has its JPA handled through the WebSphere Application Server web container. It does not use any of the facilities described in this module.

Hence, the content of this presentation does not apply to web applications, even if they contain JPA.

The information center topic accessible through the link at the bottom of this slide covers similar content as this module.



JPA container control flow



Note that in this depiction the JPA container diamond actually represents multiple bundles that collaborate to achieve the required result. Similarly, the steps of collaboration are simplified.

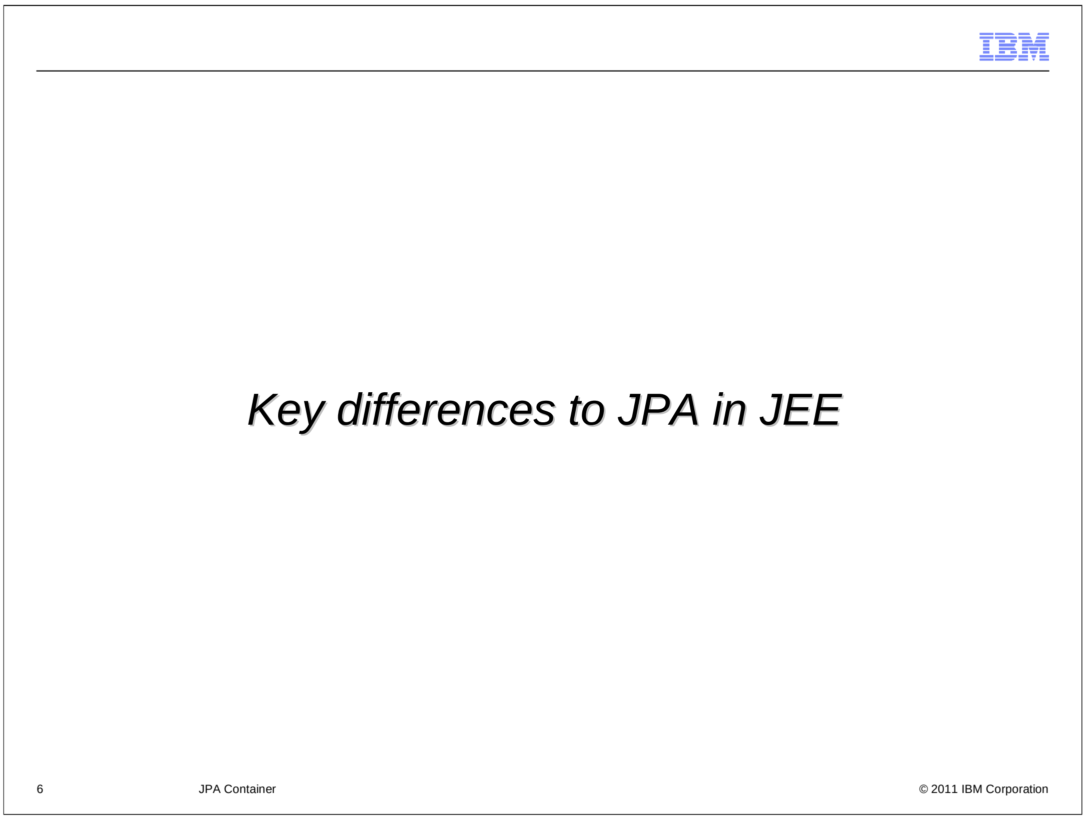
So here is in essence what happens.

First, when a persistence is installed, the JPA container is triggered to parse the persistence descriptor as specified in the Meta-Persistence header, to scan for JPA annotations and transform the blueprint if appropriate.

Next, on the basis of this information a matching persistence provider is selected. For correct operation the JPA container provides access to the data sources, information about JPA entities and byte code weaving hooks to the persistence provider.

Next, with the persistence provider the JPA container creates the JPA services for persistence units and registers them in the service registry. A bundle can either look them up directly or define these to be injected in the blueprint.

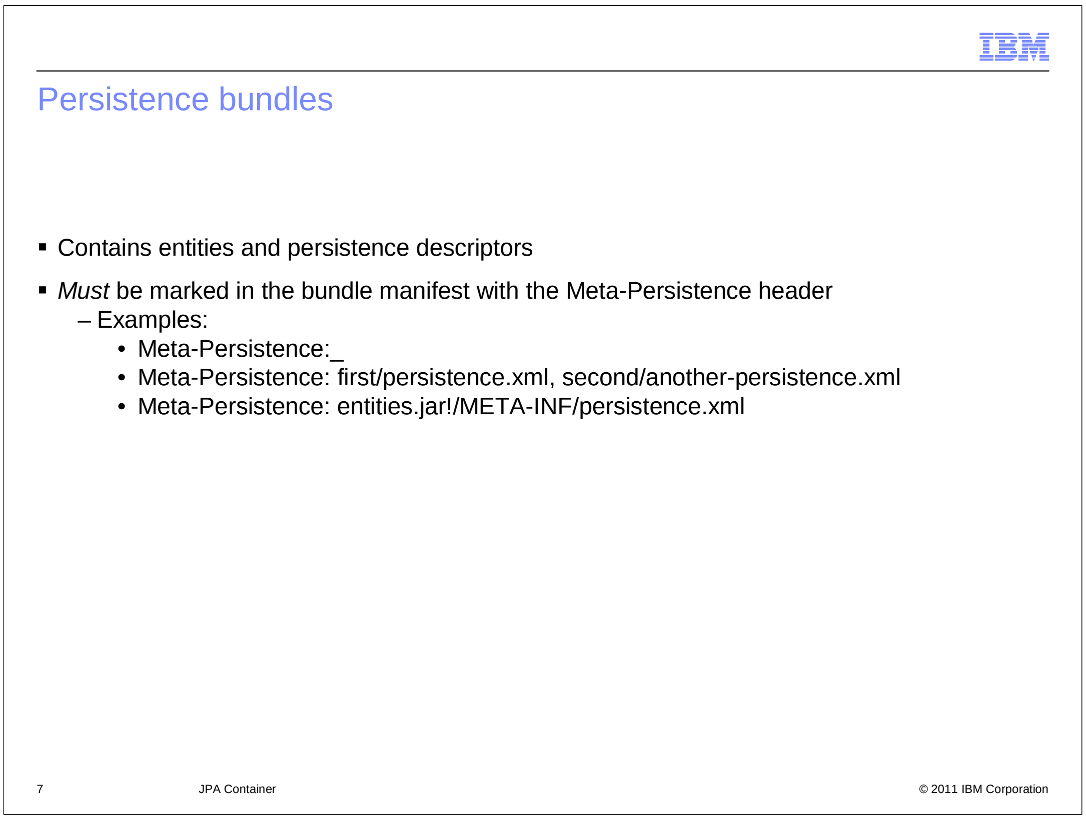
Finally, if @PersistenceUnit or @PersistenceContext injection annotations or there blueprint equivalents exist in the persistence bundle, the services are pulled in by way of the blueprint container, potentially a modified blueprint from the first step.



Key differences to JPA in JEE

6 JPA Container © 2011 IBM Corporation

This section will cover the key differences between JPA in JEE and OSGi.



Persistence bundles

- Contains entities and persistence descriptors
- *Must* be marked in the bundle manifest with the Meta-Persistence header
  - Examples:
    - Meta-Persistence:_
    - Meta-Persistence: first/persistence.xml, second/another-persistence.xml
    - Meta-Persistence: entities.jar!/META-INF/persistence.xml

7 JPA Container © 2011 IBM Corporation

A persistence bundle contains entities and at least one persistence descriptor. In addition, it must have the Meta-Persistence manifest header.

The Meta-Persistence headers marks a bundle as a persistence bundle. Any bundle that does not have this header set is ignored regardless of any contained JPA entities or persistence descriptors. Also, for this release web bundles as marked by the (Web-ContextPath manifest header) cannot be persistence bundles.

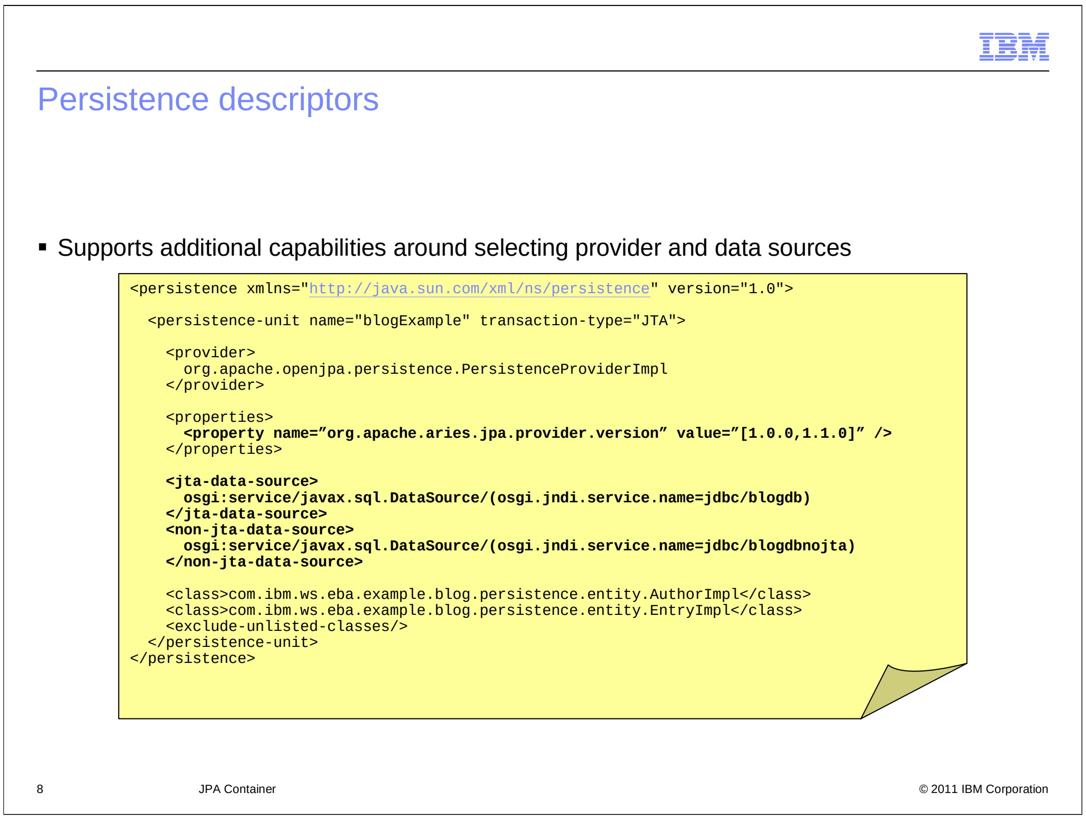
The Meta-Persistence header contains the location of all the persistence descriptors in the persistence bundle. The location META-INF/persistence.xml is always searched.

Hence, the first example where the content of the header is just a single space, the space is required to make this a valid header. This should be the most common case.

Multiple locations are separated by comma as shown in the second example.

Jar files use the jar-url syntax as shown in the last example.

Entities must be located in the persistence bundle. They cannot be in imported packages or fragments. This requirement is necessary for entity enhancement.



Persistence descriptors

- Supports additional capabilities around selecting provider and data sources

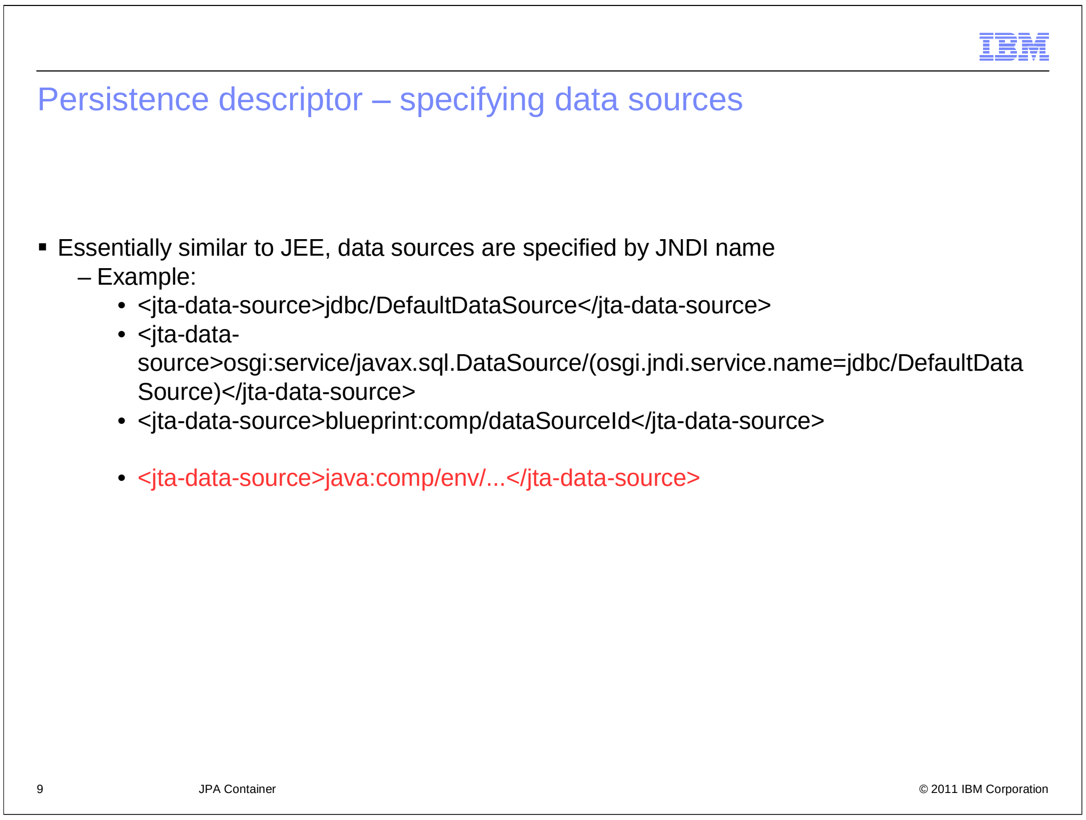
```
<persistence xmlns="http://java.sun.com/xml/ns/persistence" version="1.0">

  <persistence-unit name="blogExample" transaction-type="JTA">

    <provider>
      org.apache.openjpa.persistence.PersistenceProviderImpl
    </provider>

    <properties>
      <property name="org.apache.aries.jpa.provider.version" value="[1.0.0,1.1.0]" />
    </properties>

    <jta-data-source>
      osgi:service/javax.sql.DataSource/(osgi.jndi.service.name=jdbc/blogdb)
    </jta-data-source>
    <non-jta-data-source>
      osgi:service/javax.sql.DataSource/(osgi.jndi.service.name=jdbc/blogdbnojta)
    </non-jta-data-source>

    <class>com.ibm.ws.eba.example.blog.persistence.entity.AuthorImpl</class>
    <class>com.ibm.ws.eba.example.blog.persistence.entity.EntryImpl</class>
    <exclude-unlisted-classes/>
  </persistence-unit>
</persistence>
```

8 JPA Container © 2011 IBM Corporation

A persistence descriptor in an OSGi application differs from a JEE one in two relatively trivial aspects, specifying data sources and selecting persistence providers which is optional.

The slide shows an example of a persistence descriptor (adopted with slight modifications from the blog sample). The parts that are OSGi specific are high-lighted in bold.

Note in this example what is special about a persistent descriptor in OSGi. There are additional arguments which allow you to select your persistence provider in the same way you specify bundles with a range of versions and secondly, defining your data sources.



Persistence descriptor – specifying data sources

- Essentially similar to JEE, data sources are specified by JNDI name
  - Example:
    - <jta-data-source>jdbc/DefaultDataSource</jta-data-source>
    - <jta-data-source>osgi:service/javax.sql.DataSource/(osgi.jndi.service.name=jdbc/DefaultDataSource)</jta-data-source>
    - <jta-data-source>blueprint:comp/dataSourceId</jta-data-source>
    - <jta-data-source>java:comp/env/...</jta-data-source>

9 JPA Container © 2011 IBM Corporation

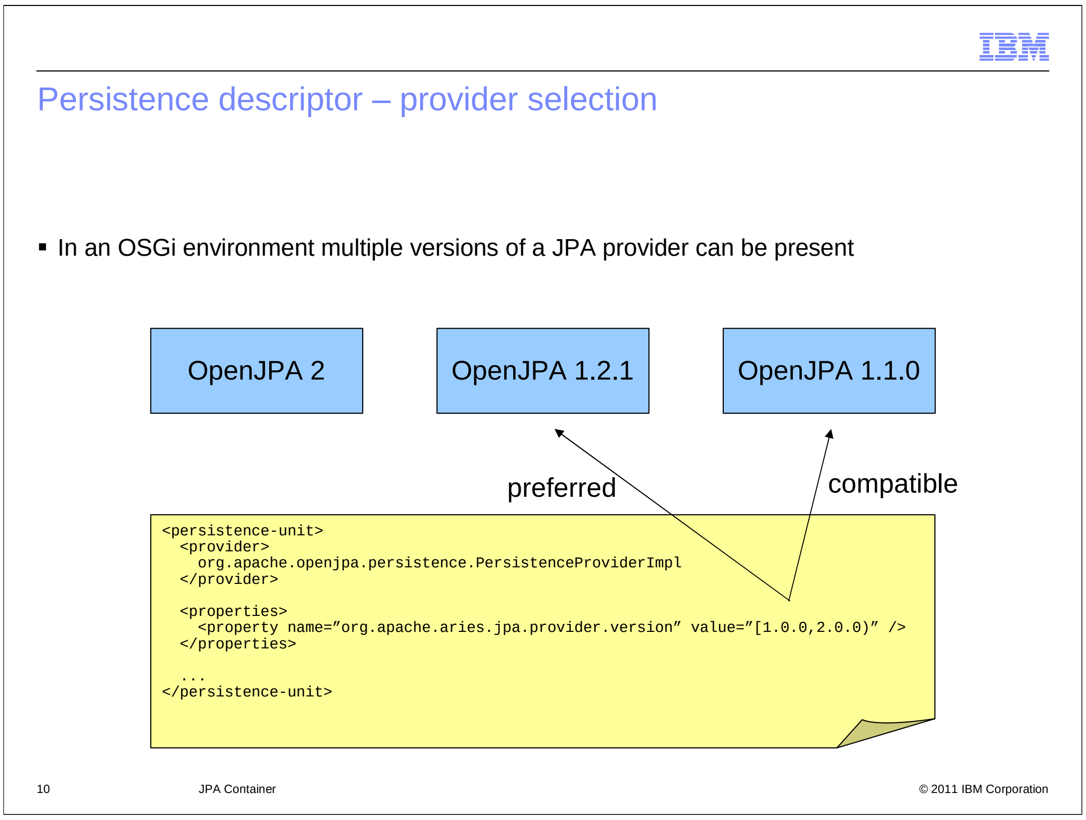
Similar to JEE data sources for JPA are specified using JNDI names. The difference lie in what types of JNDI names are supported. For OSGi applications there are three options

One option is Raw WebSphere Application Server JNDI names. These must match exactly to the JNDI name for the datasource as defined in WebSphere Application Server.

Another option is osgi:service JNDI name. This is very similar to a raw JNDI but looks the Data source up in the service registry where all the common WebSphere Application Server resources are available with there unmapped WebSphere Application Server JNDI name.

A third option is blueprint resource reference name. This specifies the ID of the blueprint resource reference for the datasource. This option requires the persistence bundle to have a blueprint xml file that defines the resource reference. However, note that this does not require the persistence bundle to use blueprint for anything else.
The preferred option is the blueprint resource reference name since it allows additional security information to be associated and configured with a datasource.

Note that a java:comp/env lookup is not valid since an OSGi application does not have an associated java: JNDI namespace. In particular, this implies that for converting entity jars to persistence bundles, the persistence xml has to be changed if it encodes datasources by way of java:comp/env lookups.



Persistence descriptor – provider selection

- In an OSGi environment multiple versions of a JPA provider can be present

OpenJPA 2
OpenJPA 1.2.1
OpenJPA 1.1.0
preferred
compatible

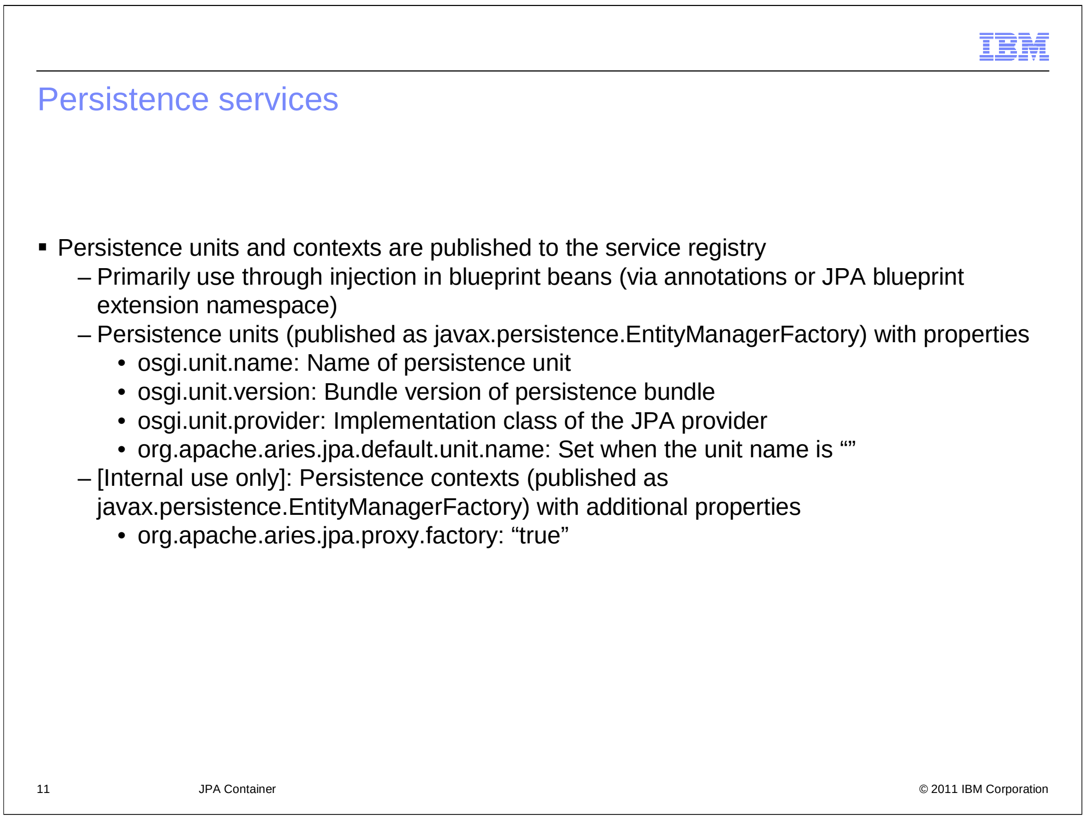
```
<persistence-unit>
  <provider>
    org.apache.openjpa.persistence.PersistenceProviderImpl
  </provider>

  <properties>
    <property name=”org.apache.aries.jpa.provider.version” value=”[1.0.0,2.0.0)” />
  </properties>

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

10 JPA Container © 2011 IBM Corporation

In an OSGi environment persistence providers should be able to coexist in multiple versions. In such a scenario one might want to be able to differentiate between different providers. This can be achieved by way of an Apache Aries custom property in the persistence descriptor as shown in this example. Note that specifying this property is completely optional. In the majority of cases there will only one JPA provider, the WebSphere Application Server provider, and hence this property is not needed.

The version range specified selects a provider bundle within the given range and with the given persistence provider implementation.

In the example here the persistence provider version range allows any OpenJPA provider in the 1.x range. Hence, the OpenJPA 2 is incompatible while OpenJPA 1.2.1 and 1.1.0 are both compatible. If the latter two are both available, then the higher versioned provider, in this case OpenJPA 1.2.1, is preferred.



Persistence services

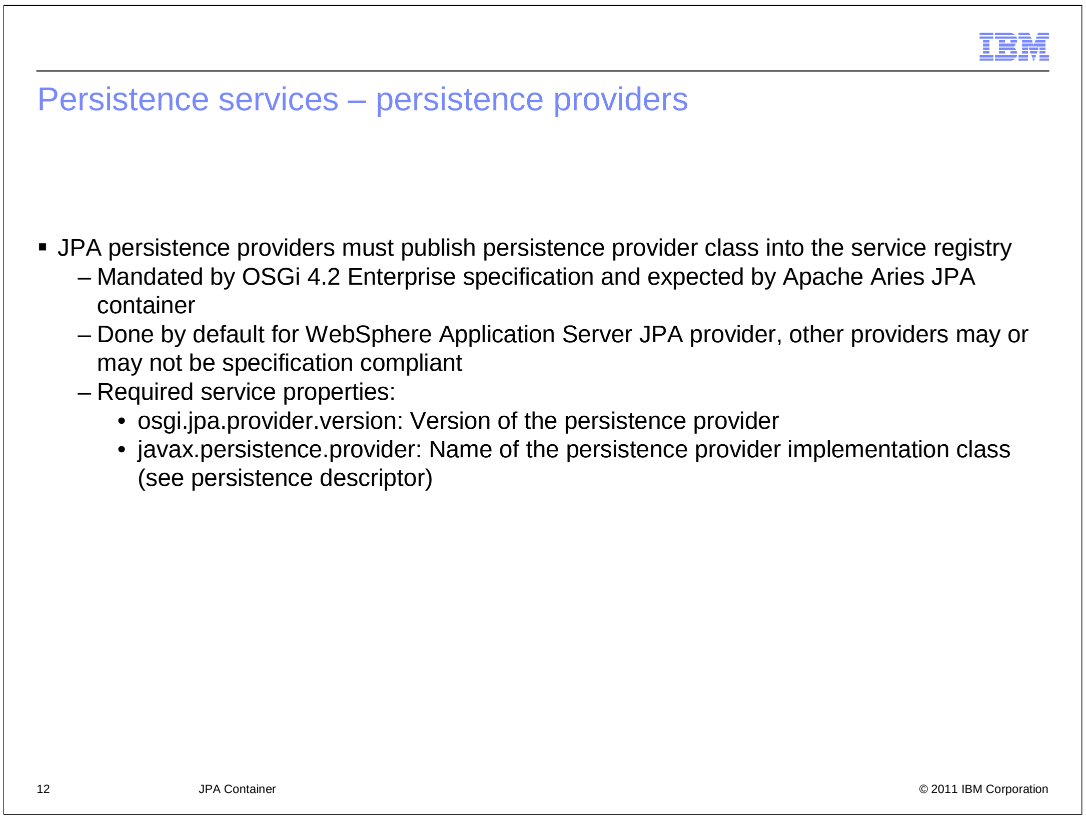
- Persistence units and contexts are published to the service registry
  - Primarily use through injection in blueprint beans (via annotations or JPA blueprint extension namespace)
  - Persistence units (published as javax.persistence.EntityManagerFactory) with properties
    - osgi.unit.name: Name of persistence unit
    - osgi.unit.version: Bundle version of persistence bundle
    - osgi.unit.provider: Implementation class of the JPA provider
    - org.apache.aries.jpa.default.unit.name: Set when the unit name is “”
  - [Internal use only]: Persistence contexts (published as javax.persistence.EntityManagerFactory) with additional properties
    - org.apache.aries.jpa.proxy.factory: “true”

11 JPA Container © 2011 IBM Corporation

The JPA container registers two types of services for persistence units and persistence contexts. Both services implement the EntityManagerFactory, but are distinguishable by the presence of the org.apache.aries.jpa.proxy.factory service property.

The service properties for the registered services provides further information about the persistence unit name (as specified in the persistence descriptor), the persistence bundle and the persistence provider. The org.apache.aries.jpa.default.unit.name allows you to construct service filters for the default persistence unit name, an empty string, which is not possible using just the osgi.unit.name property since an OSGi filter for the empty string cannot be constructed.

The persistence context service is only for use of the JPA container itself to provide container-managed, that is transaction bound, EntityManagers.



Persistence services – persistence providers

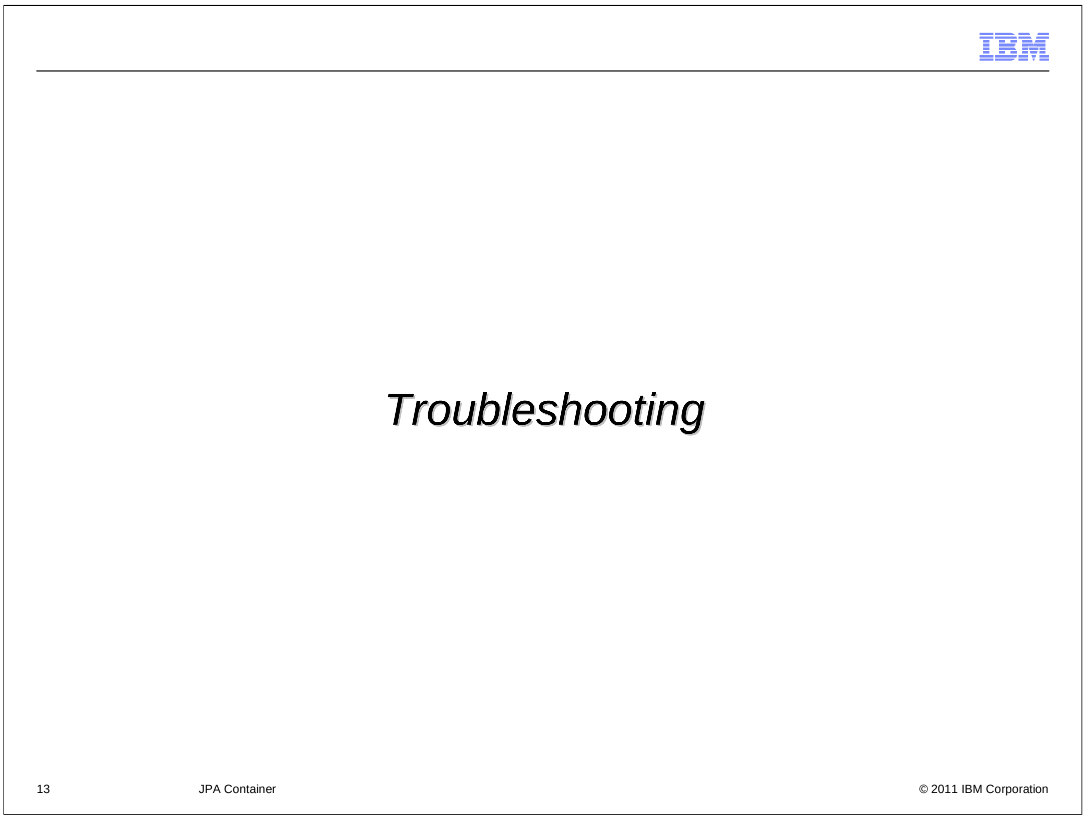
- JPA persistence providers must publish persistence provider class into the service registry
  - Mandated by OSGi 4.2 Enterprise specification and expected by Apache Aries JPA container
  - Done by default for WebSphere Application Server JPA provider, other providers may or may not be specification compliant
  - Required service properties:
    - osgi.jpa.provider.version: Version of the persistence provider
    - javax.persistence.provider: Name of the persistence provider implementation class (see persistence descriptor)

12 JPA Container © 2011 IBM Corporation

In addition to the services published by the Apache Aries JPA container, the JPA persistence provider also uses the service registry in OSGi.

In particular, any OSGi specification compliant persistence provider must make its persistence provider implementation available as a service with the osgi.jpa.provider.version and javax.persistence.provider properties. This service is required in order for the Apache Aries JPA container to recognize the persistence provider.

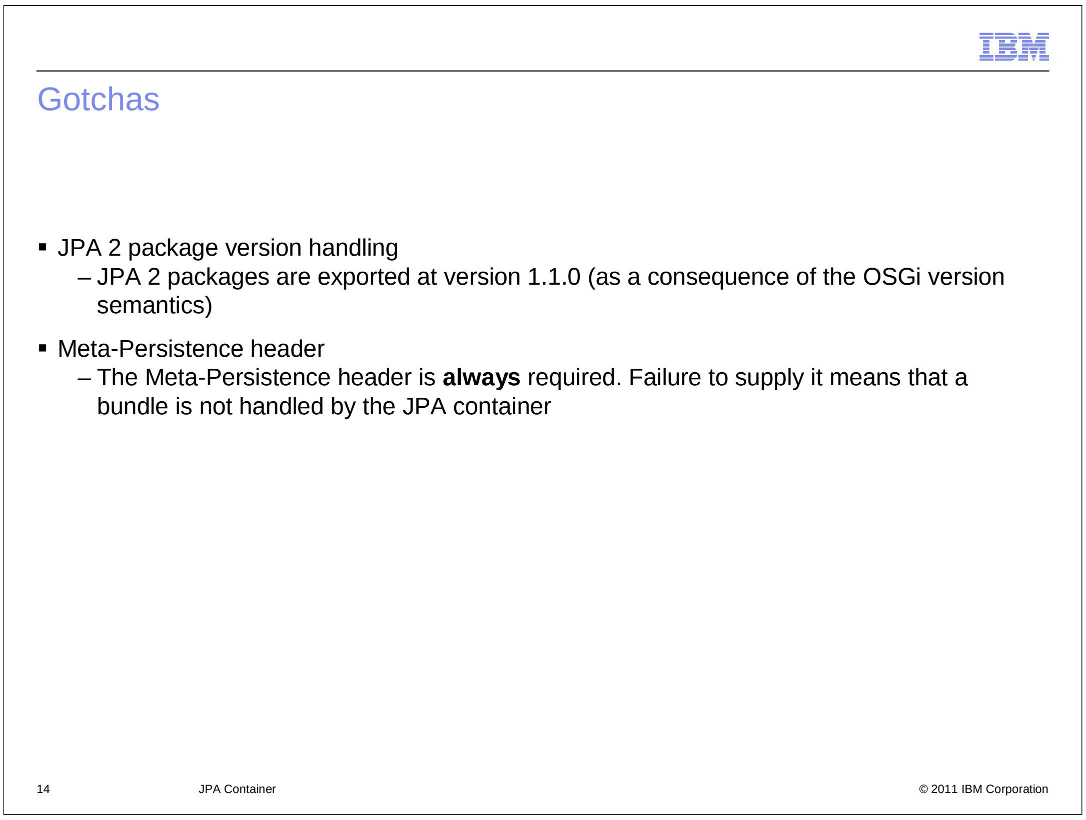
This becomes important when you want to use a JPA provider other than the WebSphere Application Server default one, which has to be installed as part of the application. In this case the you must ensure that the persistence provider is OSGi compliant. If it is not, it is relatively simple to add a blueprint declaration to export the persistence provider implementation.



Troubleshooting

13 JPA Container © 2011 IBM Corporation

This section covers troubleshooting tips.



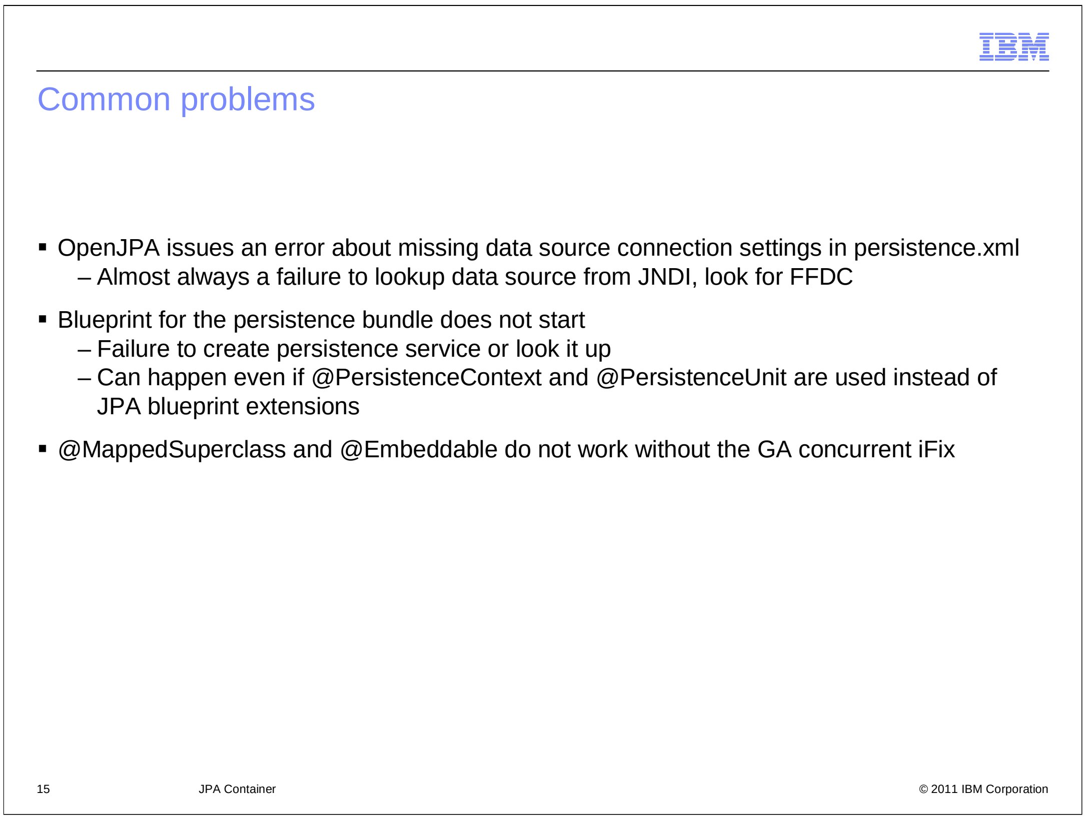
Gotchas

- JPA 2 package version handling
  - JPA 2 packages are exported at version 1.1.0 (as a consequence of the OSGi version semantics)
- Meta-Persistence header
  - The Meta-Persistence header is **always** required. Failure to supply it means that a bundle is not handled by the JPA container

14 JPA Container © 2011 IBM Corporation

One source of confusion can be the version of the JPA 2 packages. These follow the OSGi version handling semantics and hence are versioned at 1.1.0 because they are compatible with earlier versions with the JPA 1 packages (versioned at 1.0.0).

A second source of confusion can be the requirement to have a Meta-Persistence header even if the value can be empty. Without this header the bundle is not picked up by the JPA container and absolutely nothing JPA related will work. This problem should not occur when using the Rational Application Developer tools for OSGi applications.



Common problems

- OpenJPA issues an error about missing data source connection settings in persistence.xml
  - Almost always a failure to lookup data source from JNDI, look for FFDC
- Blueprint for the persistence bundle does not start
  - Failure to create persistence service or look it up
  - Can happen even if @PersistenceContext and @PersistenceUnit are used instead of JPA blueprint extensions
- @MappedSuperclass and @Embeddable do not work without the GA concurrent iFix

15 JPA Container © 2011 IBM Corporation

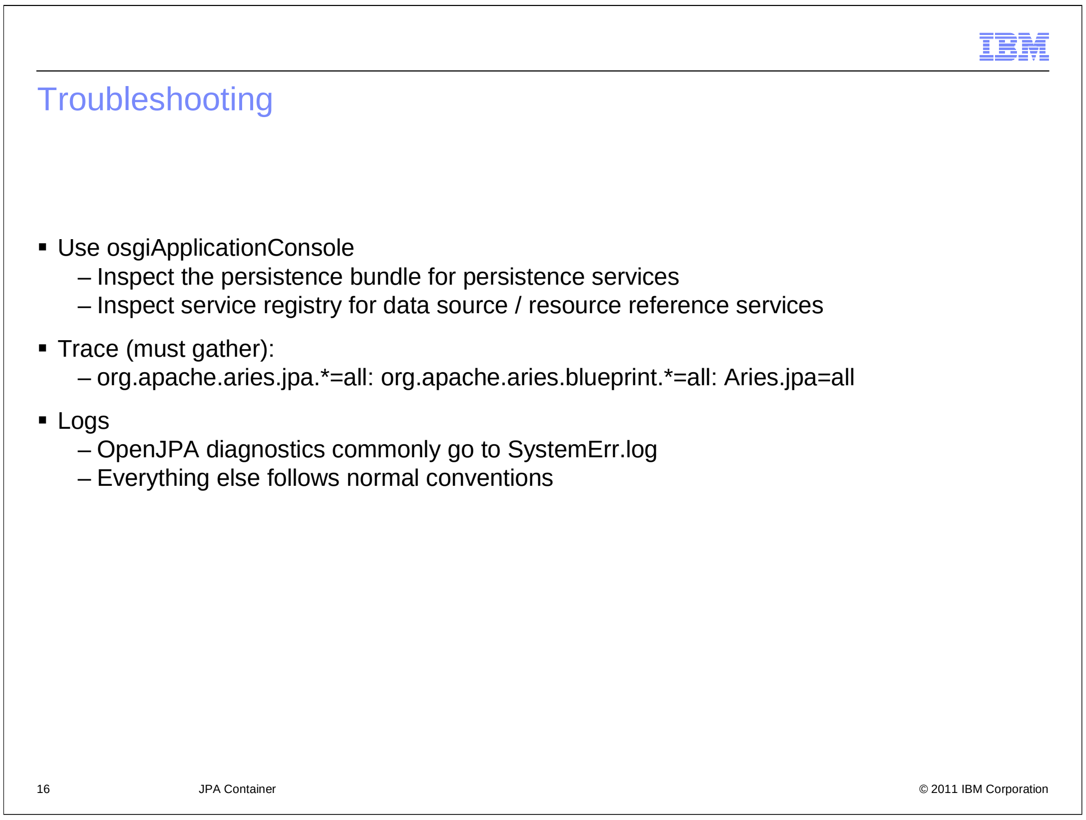
Two commonly found problems are configured incorrectly data sources and configured incorrectly persistence services (potentially incorrect unit names or genuine defects).

A missing data source will manifest itself as an FFDC with the javax.naming.NamingException that was thrown for the failed JNDI lookup. However, in the SystemOut.log the problem is reported by OpenJPA as missing datasource connection settings in the persistence.xml.

The second problem around configured incorrectly persistence services prevents the blueprint container for the persistence bundle from starting. Commonly, this causes any services that the persistence bundle itself registers to be unavailable to the rest of the application.

The reason for this is that persistence services are injected by way of blueprint, regardless of whether standard JPA annotations or the JPA blueprint extensions are used.

Finally, one issue around JPA, @Embeddable and @MappedSuperclass annotations not being supported, is only fixed in the GA concurrent iFix.



Troubleshooting

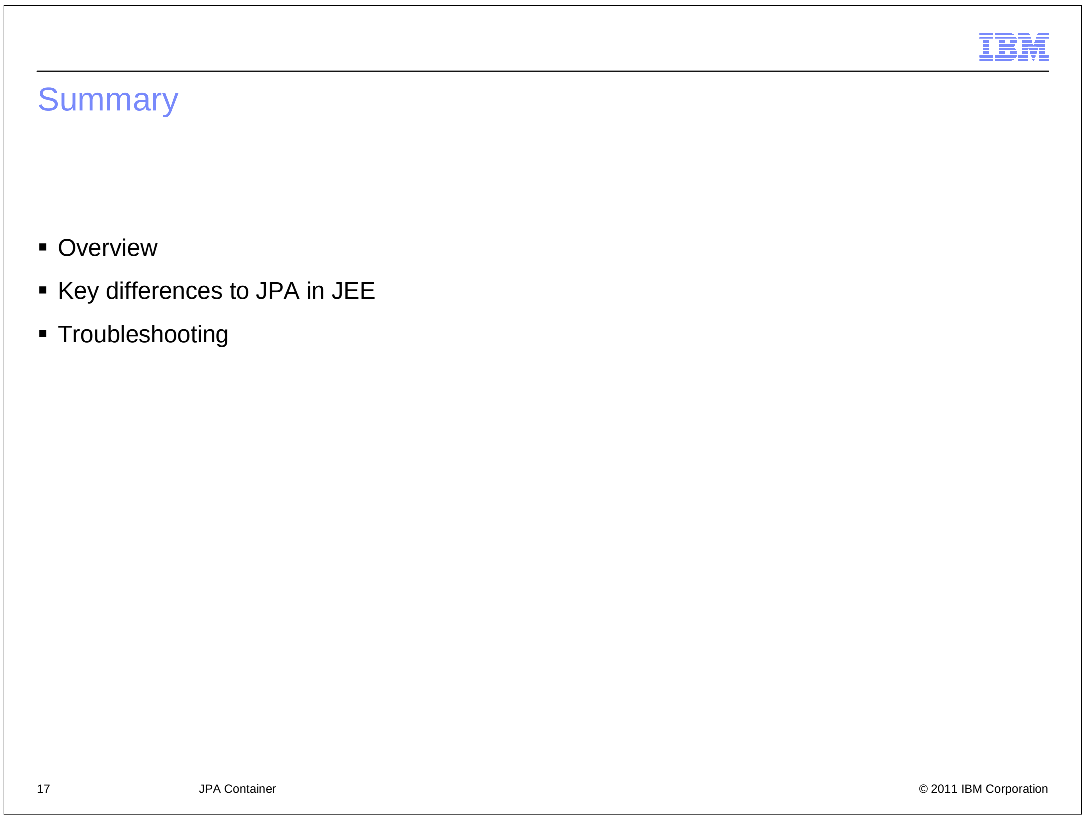
- Use osgiApplicationConsole
  – Inspect the persistence bundle for persistence services
  – Inspect service registry for data source / resource reference services
- Trace (must gather):
  – org.apache.aries.jpa.*=all: org.apache.aries.blueprint.*=all: Aries.jpa=all
- Logs
  – OpenJPA diagnostics commonly go to SystemErr.log
  – Everything else follows normal conventions

16 JPA Container © 2011 IBM Corporation

The JPA container only facilitates JPA it does not actually act as a JPA provider. Instead that part is handled by the WebSphere Application Server JPA provider (either 1 or 2). Hence, problems that deal with the actual behavior of JPA when it is running are not part of Aries.

However, problems with the provisioning of JPA services most likely are due to the OSGi applications feature. For, example: missing persistence services, non-enhanced entities, or unscanned persistence annotations or failures when creating persistence units such as failure to connect to database. This is commonly a configuration problem.

The only speciality around logging is the fact that OpenJPA diagnostic messages go to SystemErr.log instead of SystemOut.log. The Aries JPA container itself logs normally to SystemOut.log and trace.log and prints Exceptions to FFDCs or SystemErr.log depending on whether the problem happen in Apache Aries code or feature pack code.

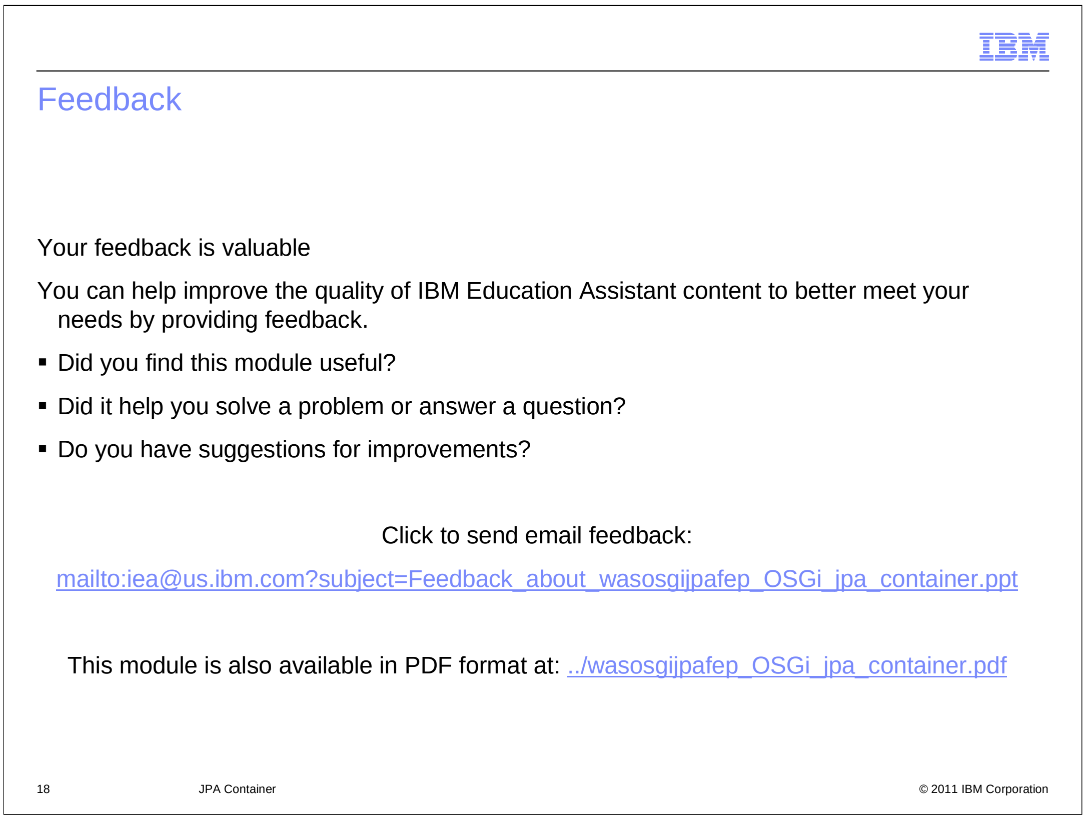


Summary

- Overview
- Key differences to JPA in JEE
- Troubleshooting

17 JPA Container © 2011 IBM Corporation

This module provided an overview of the JPA container used in the OSGi framework. It showed specific differences in JPA in JEE and JPA in OSGi. It described persistence bundles, persistent descriptors and persistence services. Finally it provided some tips for troubleshooting your applications.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

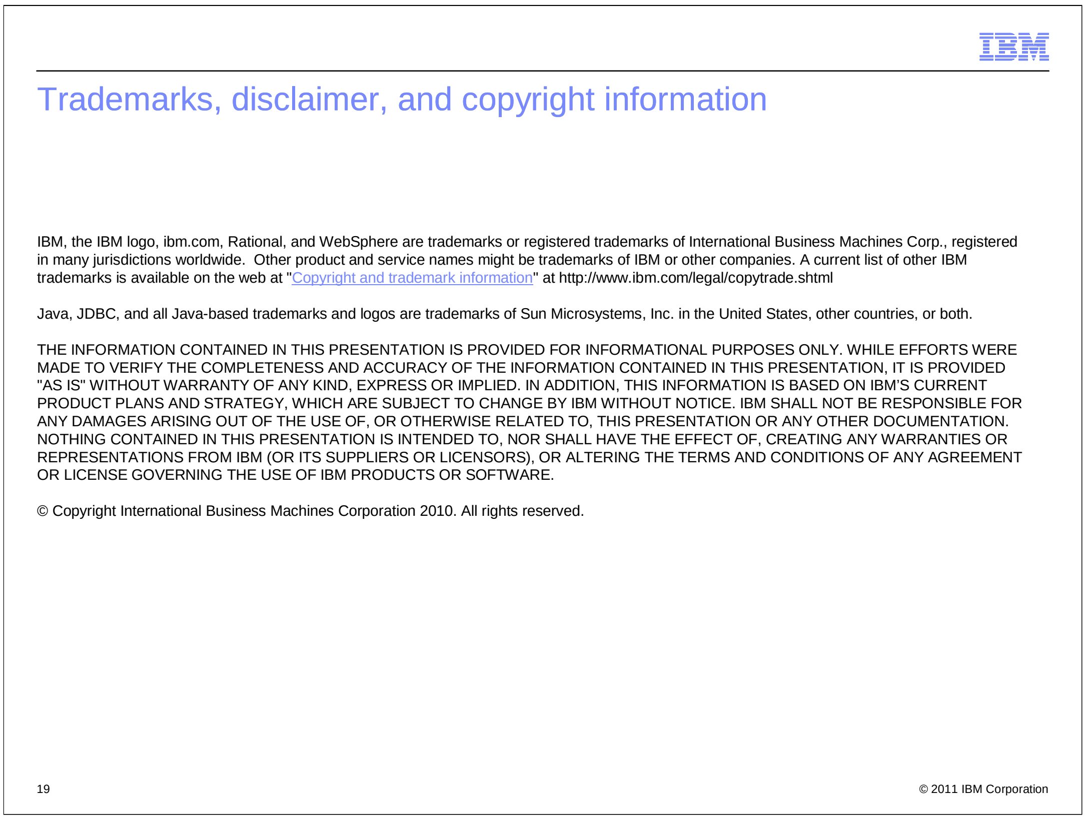
Click to send email feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_wasosgijpafep_OSGi_jpa_container.ppt

This module is also available in PDF format at: ../wasosgijpafep_OSGi_jpa_container.pdf

18 JPA Container

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, disclaimer, and copyright information