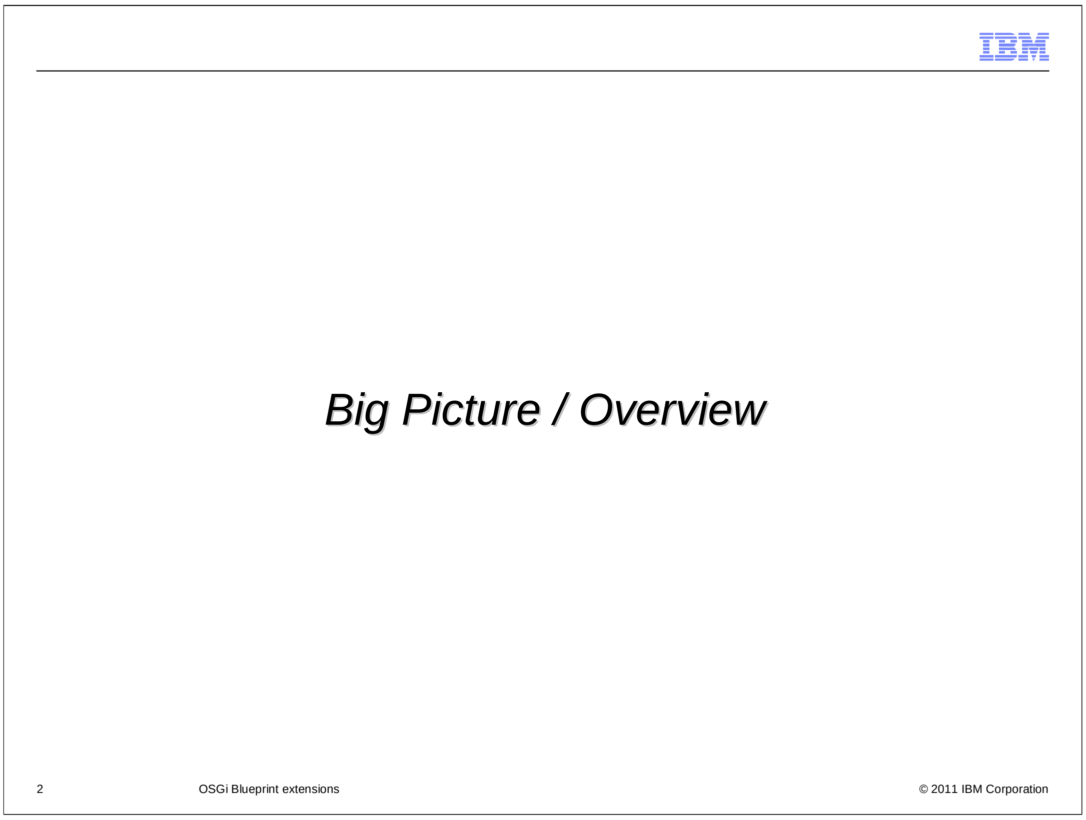
IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API 2.0

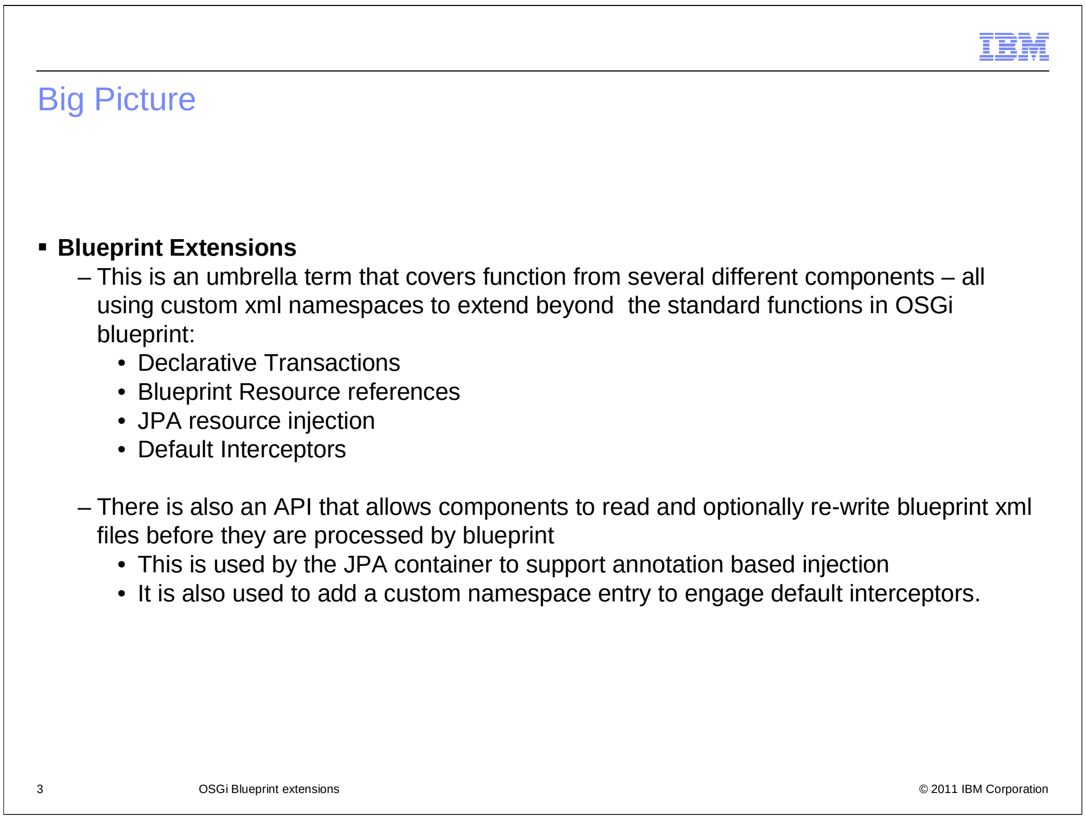
This module covers Blueprint extensions for the IBM WebSphere Application Server Feature Pack for OSGi Applications and Java Persistence API 2.0 OSGi.



Big Picture / Overview

2 OSGi Blueprint extensions © 2011 IBM Corporation

This section contains an overview of blueprint extensions.



Big Picture

- **Blueprint Extensions**
  - This is an umbrella term that covers function from several different components – all using custom xml namespaces to extend beyond the standard functions in OSGi blueprint:
    - Declarative Transactions
    - Blueprint Resource references
    - JPA resource injection
    - Default Interceptors
  - There is also an API that allows components to read and optionally re-write blueprint xml files before they are processed by blueprint
    - This is used by the JPA container to support annotation based injection
    - It is also used to add a custom namespace entry to engage default interceptors.

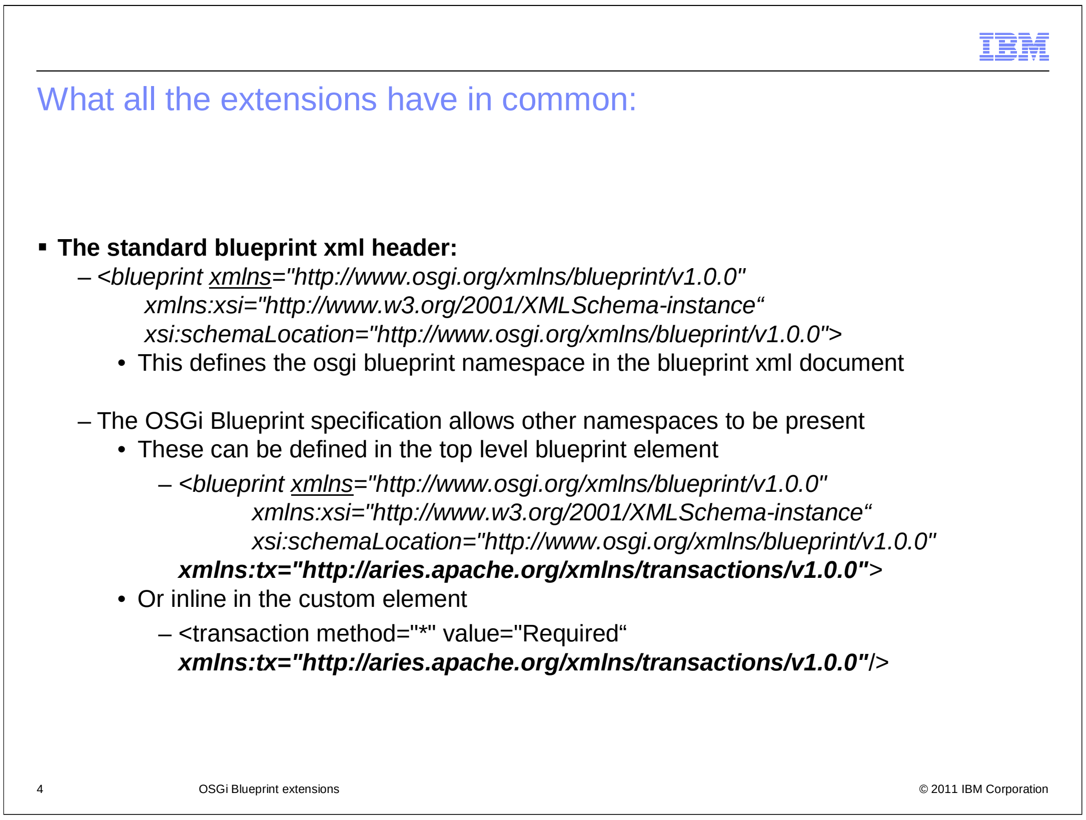
3 OSGi Blueprint extensions © 2011 IBM Corporation

There is no one “Blueprint Extensions Bundle” – but there are several different components that add functionality to the blueprint container.

All of these components use the same extension mechanism to add their extra function, custom XML namespaces, which is currently the only way to extend the blueprint model.

Some components go even further, and do not require you to add the custom XML namespaces, however these still use the custom namespace mechanism. To make this work the components re-write the original blueprint xml to include the necessary namespaces.

A good example of this is the JPA container, which when it is doing annotation based injection rewrites your blueprint to include the xml that you should have written to give the equivalent functionality. It is also used to add some default intercepter support in blueprint.



What all the extensions have in common:

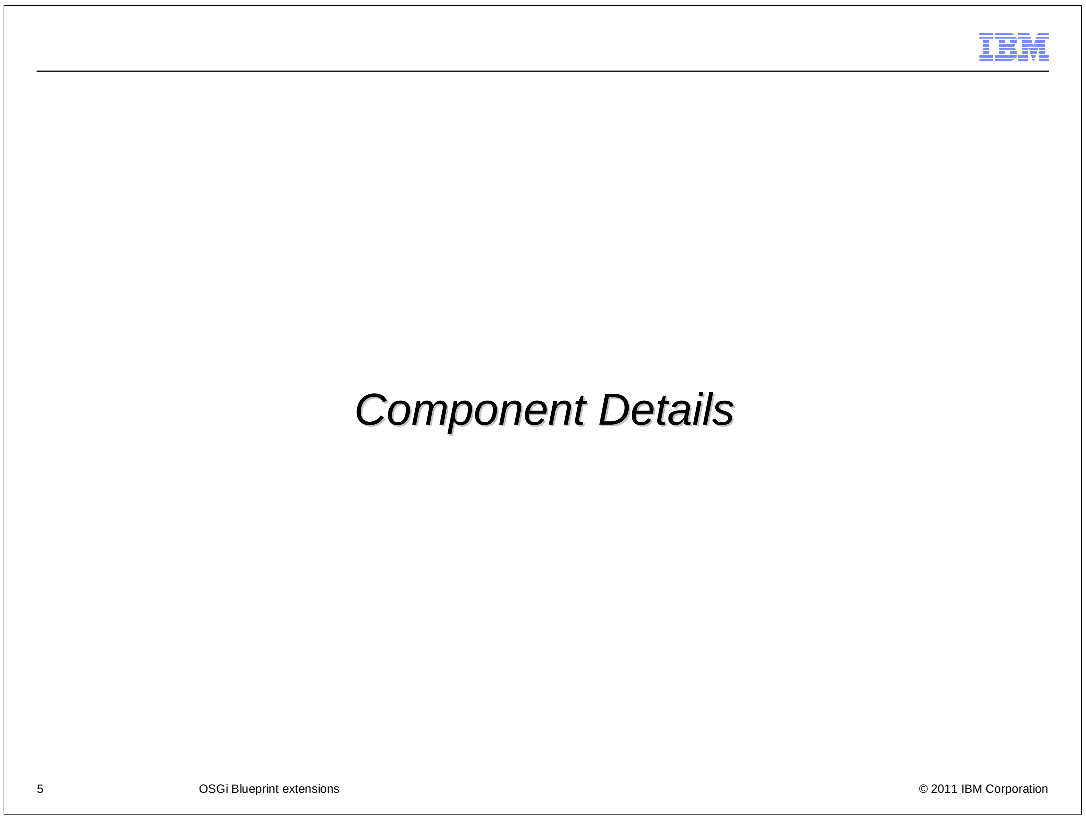
- **The standard blueprint xml header:**
  - *<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“ xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0">*
    - This defines the osgi blueprint namespace in the blueprint xml document
  - The OSGi Blueprint specification allows other namespaces to be present
    - These can be defined in the top level blueprint element
      - *<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance“ xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0"* ***xmlns:tx="http://aries.apache.org/xmlns/transactions/v1.0.0"****>*
    - Or inline in the custom element
      - <transaction method="*" value="Required“ ***xmlns:tx="http://aries.apache.org/xmlns/transactions/v1.0.0"****/>*

4 OSGi Blueprint extensions © 2011 IBM Corporation

The standard blueprint header is defined, and required in each and every blueprint xml document, by the OSGi blueprint specification. It sets the default document namespace to “http://www.osgi.org/xmlns/blueprint/v1.0.0”

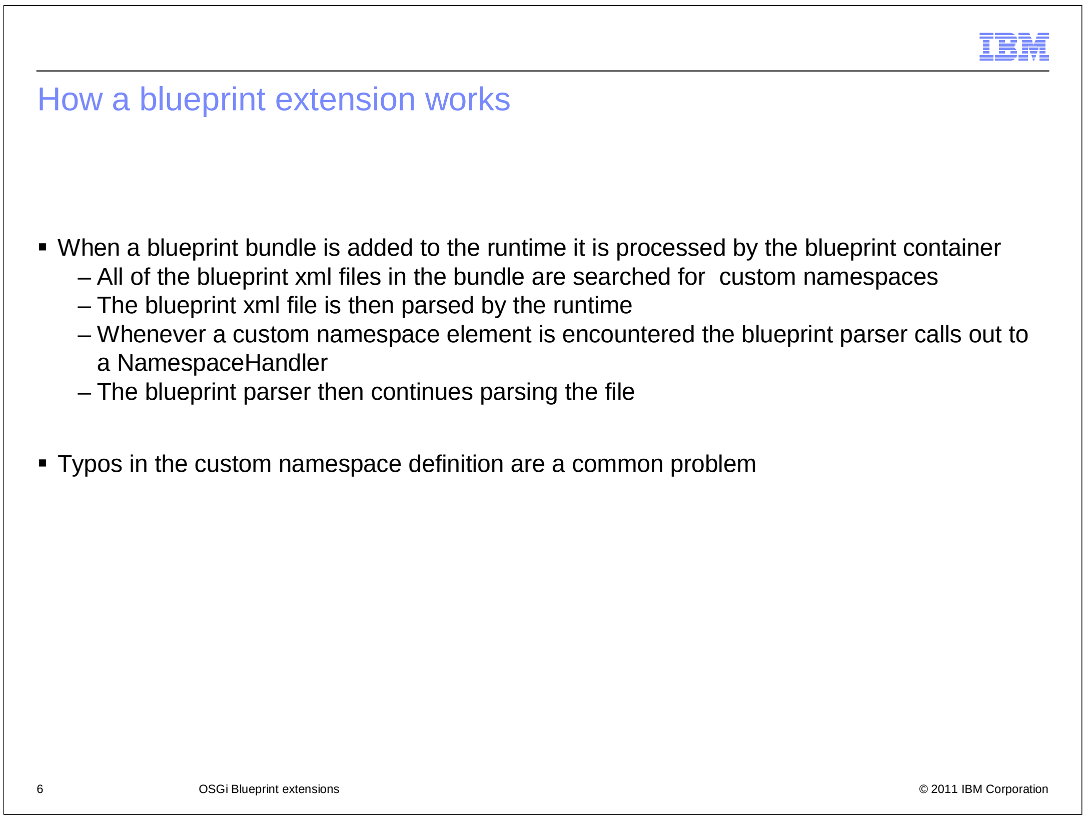
Other namespaces can be added to the blueprint using standard XML rules, either as prefixed entries, or directly within the custom XML elements if you plan to use the other functions. These can be added either at the top level or they can be inline with the custom XML elements. As long as it is valid XML it should be parsed correctly.

It should be noted that when parsing, schema validation is required to be turned on so if these schemas are not specified there are failures and the blueprint will not come up.



Component Details

This section covers details of the component.



How a blueprint extension works

- When a blueprint bundle is added to the runtime it is processed by the blueprint container
  - All of the blueprint xml files in the bundle are searched for custom namespaces
  - The blueprint xml file is then parsed by the runtime
  - Whenever a custom namespace element is encountered the blueprint parser calls out to a NamespaceHandler
  - The blueprint parser then continues parsing the file
- Typos in the custom namespace definition are a common problem

6 OSGi Blueprint extensions © 2011 IBM Corporation

The blueprint runtime actually parses the blueprint descriptors twice. The first pass is fast, and only finds every namespace used in by the blueprint bundle.

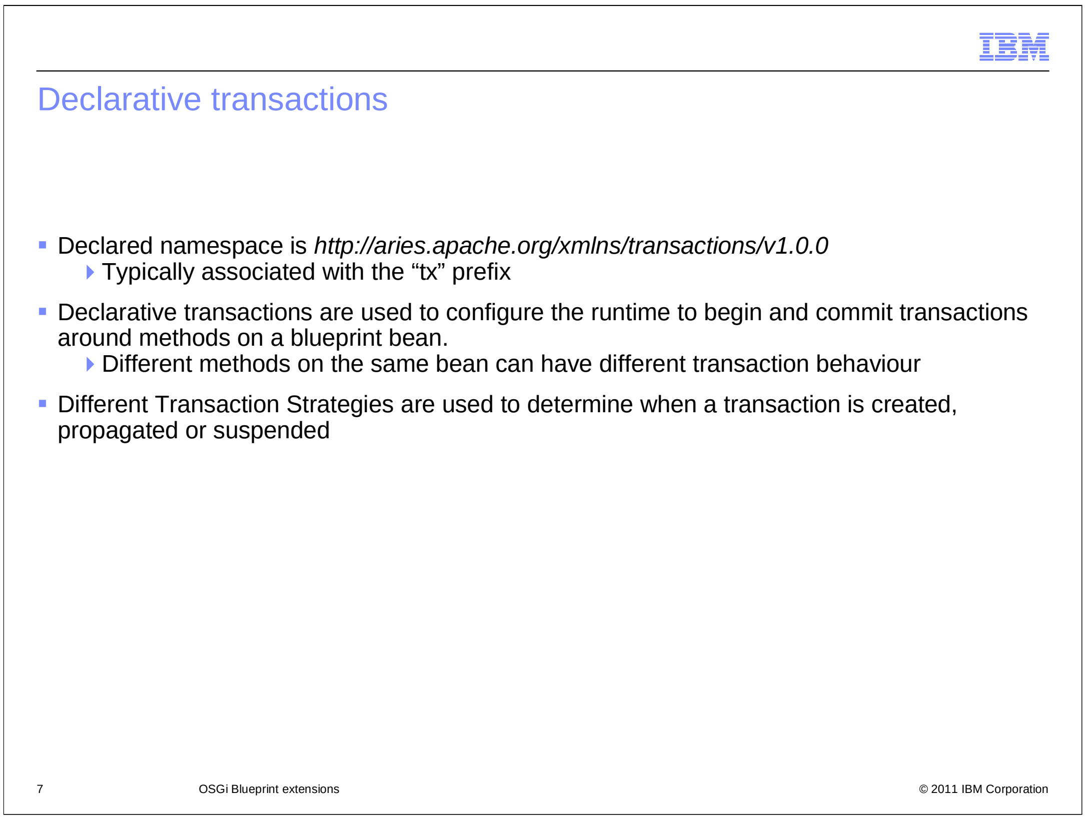
If the blueprint bundle uses a non-standard namespace then the blueprint container will attempt to locate NamespaceHandler services in the OSGi service registry for each custom namespace. A NamespaceHandler service advertises every xml namespace that it can process using OSGi service properties.

The blueprint runtime will not parse the blueprint xml until NamespaceHandler services can be found for every custom namespace used in the bundle. Unless NamespaceHandler services can be found for every custom namespace the blueprint container is unable to process the bundle. This can result in the blueprint container waiting indefinitely if no NamespaceHandler exists. If this situation is encountered then the blueprint container will issue a warning to the log.

Once the blueprint parser begins to parse the blueprint xml files, it will parse any standard blueprint elements. When the parser reaches a custom element the parser calls out to the NamespaceHandler that advertised support for the namespace of the custom element.

At this point the NamespaceHandler has the opportunity to process the information in the custom element, modify the runtime blueprint model, register an interceptor, or perform any other operation.

If there is a typing error in any of the namespace definitions then the blueprint will almost certainly fail to start.



Declarative transactions

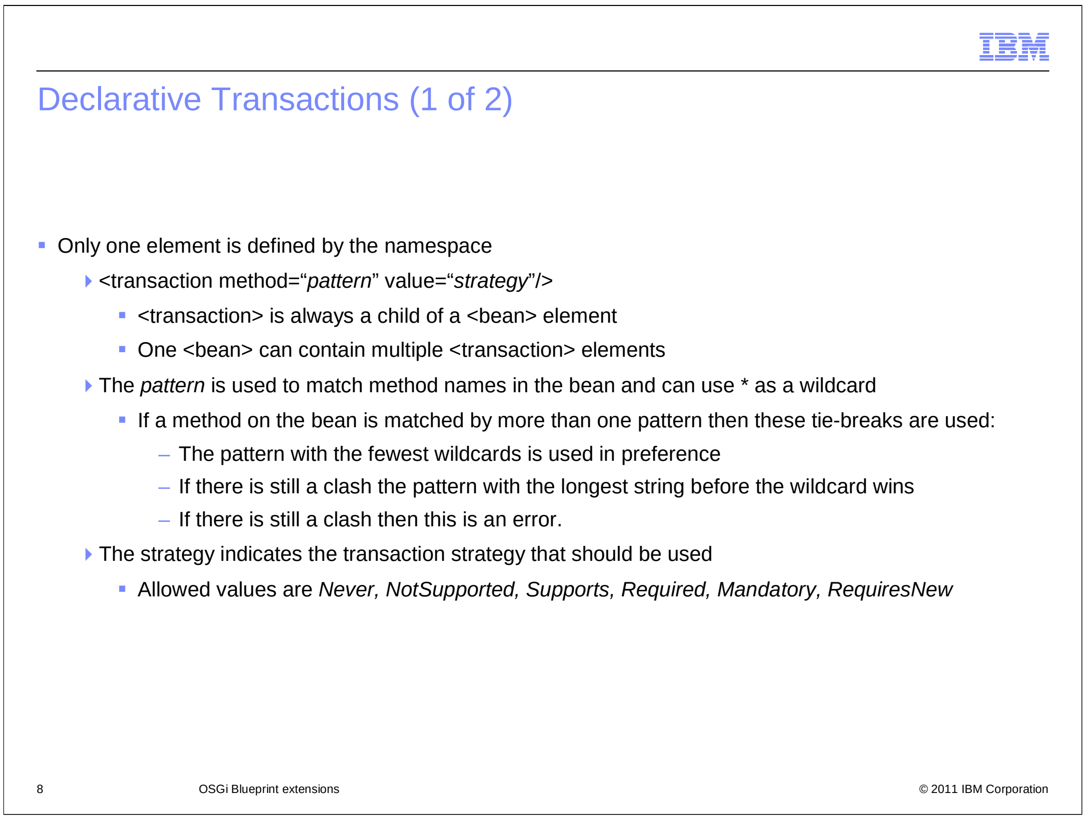
- Declared namespace is *http://aries.apache.org/xmlns/transactions/v1.0.0*
  - Typically associated with the “tx” prefix
- Declarative transactions are used to configure the runtime to begin and commit transactions around methods on a blueprint bean.
  - Different methods on the same bean can have different transaction behaviour
- Different Transaction Strategies are used to determine when a transaction is created, propagated or suspended

7 OSGi Blueprint extensions © 2011 IBM Corporation

Declarative transactions are used when the application wants the container to manage the transaction life cycle. No application code needs to be written to take advantage of the transactions.

Transactions are declared on a per-bean basis, and can be scoped to only apply to certain methods in the bean. Different methods can have different transaction semantics.

The transaction strategies available are identical to those defined by Enterprise Java. There are six of them, they are defined by the Enterprise Java specification and they have exactly the same semantics.



Declarative Transactions (1 of 2)

- Only one element is defined by the namespace
  - <transaction method="*pattern*" value="*strategy*"/>
    - <transaction> is always a child of a <bean> element
    - One <bean> can contain multiple <transaction> elements
  - The *pattern* is used to match method names in the bean and can use * as a wildcard
    - If a method on the bean is matched by more than one pattern then these tie-breaks are used:
      - The pattern with the fewest wildcards is used in preference
      - If there is still a clash the pattern with the longest string before the wildcard wins
      - If there is still a clash then this is an error.
  - The strategy indicates the transaction strategy that should be used
    - Allowed values are *Never, NotSupported, Supports, Required, Mandatory, RequiresNew*

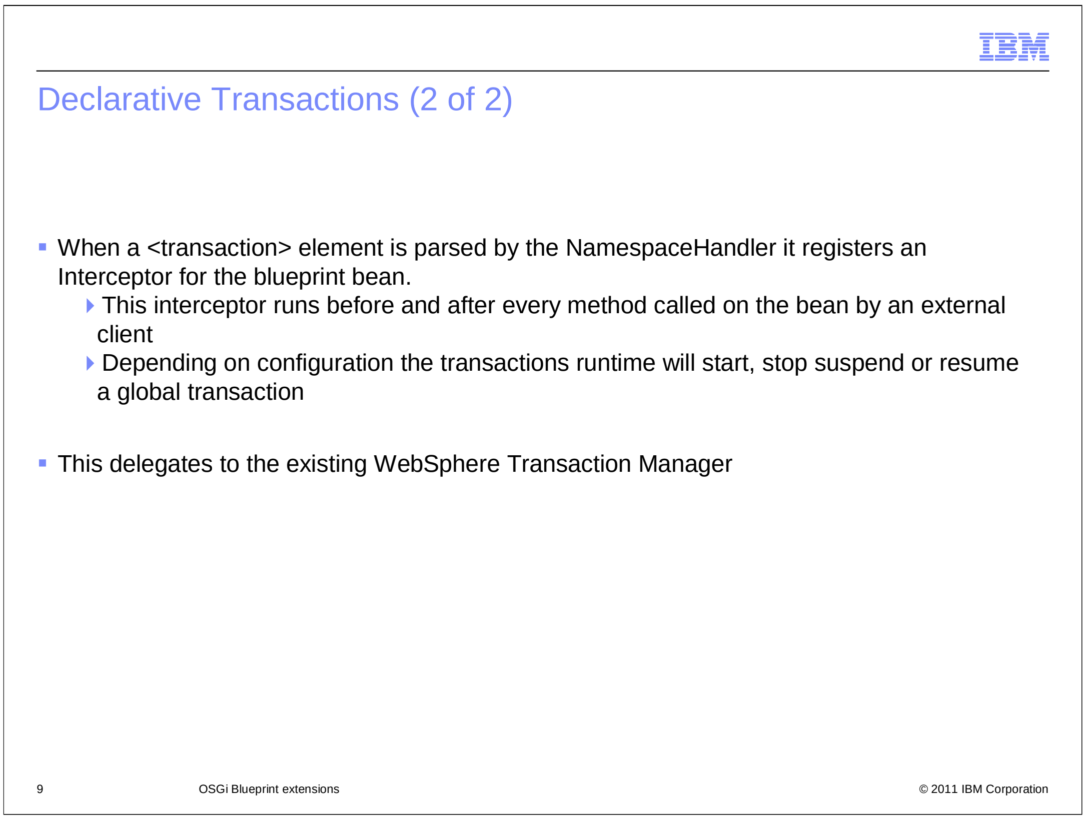
8 OSGi Blueprint extensions © 2011 IBM Corporation

The transaction element is used to specify declarative transactions for a blueprint bean. The pattern attribute provides a pattern used to match one or more methods on the bean. This pattern can use one or more * characters as wildcards, for instance, ins*R* will match "insertRows"

If multiple transaction elements are specified that can apply to the same method then there is a well defined tie-break mechanism. In the event that more than one pattern is still applicable after the tie-break then there is an IllegalStateException and an error logged. The blueprint will fail to start

There will also be an error logged if the transaction strategy is not one of the allowed values. The blueprint will fail to start.

The transaction strategy is the value attribute of the transaction element and one of these six values must be provided. They are text sensitive so the chosen value must be supplied as listed here. Failure to do so will result in a configuration error and a failure of the blueprint to start.

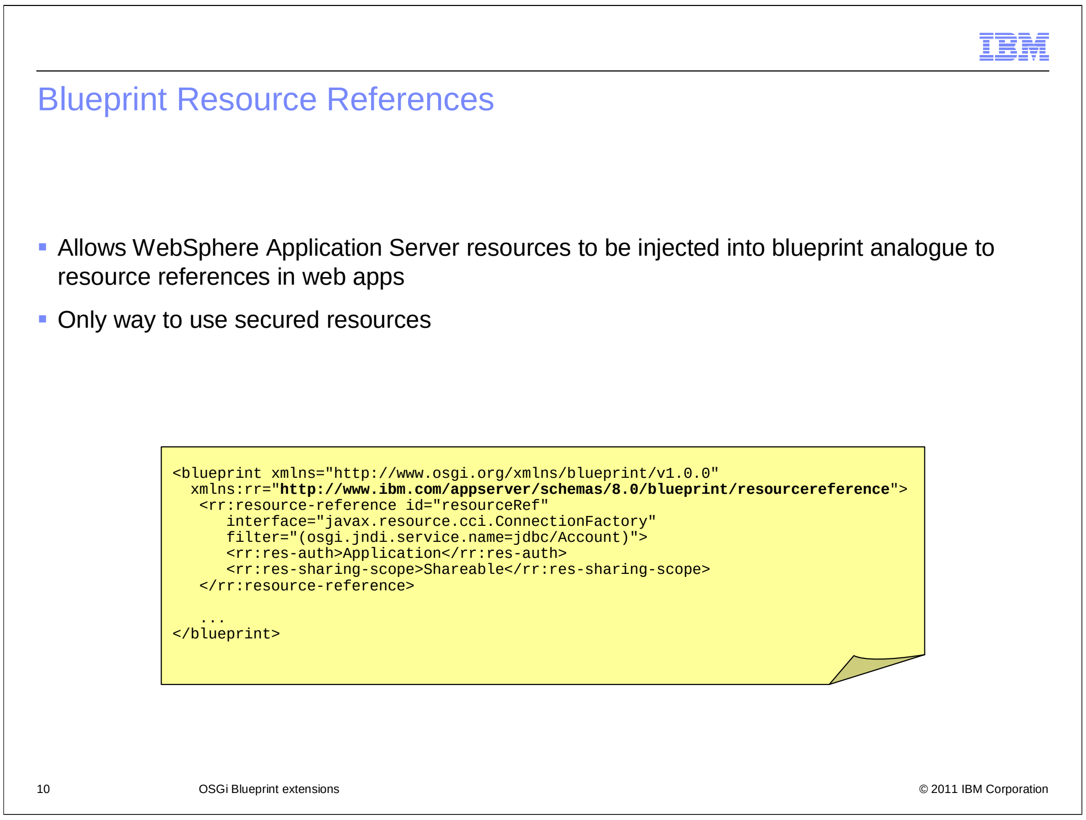

Declarative Transactions (2 of 2)

- When a <transaction> element is parsed by the NamespaceHandler it registers an Interceptor for the blueprint bean.
  - This interceptor runs before and after every method called on the bean by an external client
  - Depending on configuration the transactions runtime will start, stop suspend or resume a global transaction
- This delegates to the existing WebSphere Transaction Manager

9 OSGi Blueprint extensions © 2011 IBM Corporation

Once the NamespaceHandler is called the behavior is very simple. The NamespaceHandler registers an Interceptor for the bean with the blueprint container. This is called before and after any method is invoked by another bean or client. This interceptor uses the configuration information to enforce the required transaction boundaries. The actual transaction processing is delegated to the existing WebSphere Transaction Manager.

If the WebSphere Transaction Manager is unavailable then the declarative transactions runtime will not start.



Blueprint Resource References

- Allows WebSphere Application Server resources to be injected into blueprint analogue to resource references in web apps
- Only way to use secured resources

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
  xmlns:rr="http://www.ibm.com/appserver/schemas/8.0/blueprint/resourcereference">
   <rr:resource-reference id="resourceRef"
      interface="javax.resource.cci.ConnectionFactory"
      filter="(osgi.jndi.service.name=jdbc/Account)">
      <rr:res-auth>Application</rr:res-auth>
      <rr:res-sharing-scope>Shareable</rr:res-sharing-scope>
   </rr:resource-reference>

   ...
</blueprint>
```

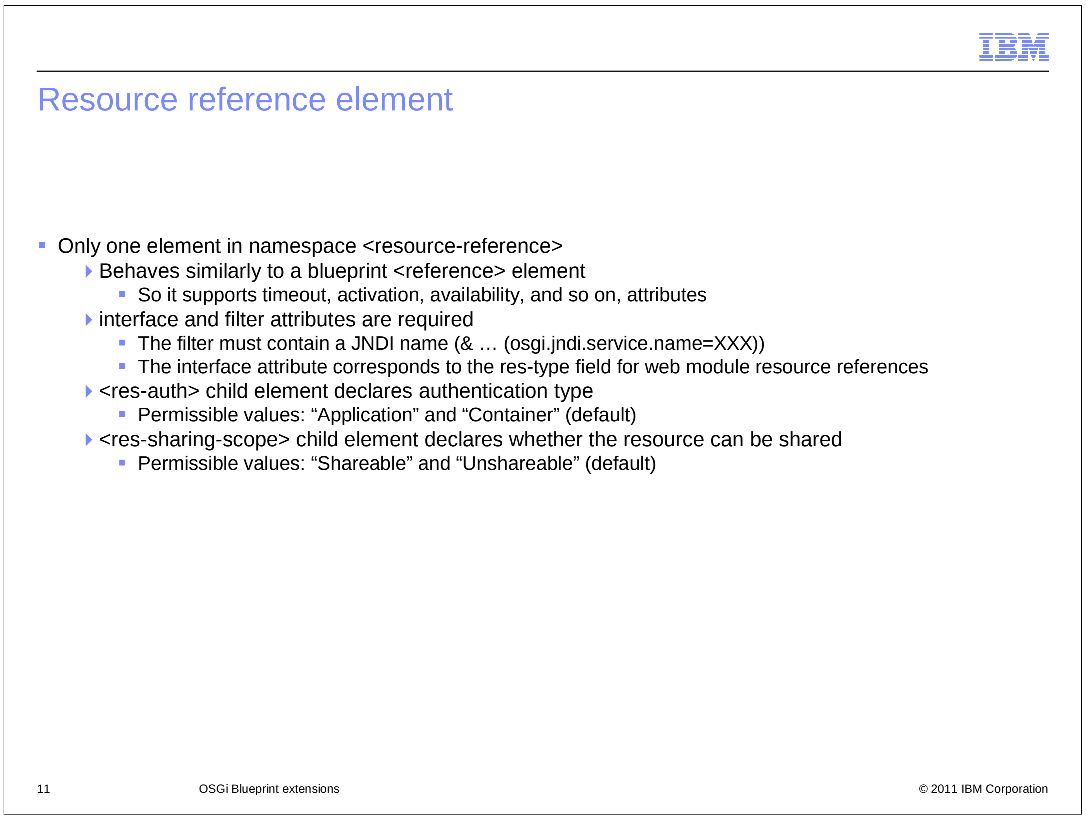
10 OSGi Blueprint extensions © 2011 IBM Corporation

Blueprint resource references are the another custom namespace that is likely to be widely used.

The main use case is secured resources, which need to be accessed with authentication information, such as an associated authentication alias. It is the only way to use such resources from blueprint. It allows you to specify configuration on an application basis rather than having anyone who can access the resource in JNDI to have access to your data. You can specify an authentication alias against your resource reference at deployment time so that only your application is allowed to access your database.

The example blueprint shown highlights the resource reference namespace http://www.ibm.com/appserver/schemas/8.0/blueprint/resourcereference and the commonly chosen namespace prefix “rr”.

The filter is important. It is what specifies the actual physical WebSphere Application Server resource that will be mapped. So the “jdbc/Account” here indicates that the resource used will be found at the JNDI location jdbc/Account. A common error here is not to have a matching JNDI name here and in WebSphere Application Server.



Resource reference element

- Only one element in namespace <resource-reference>
  - Behaves similarly to a blueprint <reference> element
    - So it supports timeout, activation, availability, and so on, attributes
  - interface and filter attributes are required
    - The filter must contain a JNDI name (& … (osgi.jndi.service.name=XXX))
    - The interface attribute corresponds to the res-type field for web module resource references
  - <res-auth> child element declares authentication type
    - Permissible values: “Application” and “Container” (default)
  - <res-sharing-scope> child element declares whether the resource can be shared
    - Permissible values: “Shareable” and “Unshareable” (default)

11 OSGi Blueprint extensions © 2011 IBM Corporation

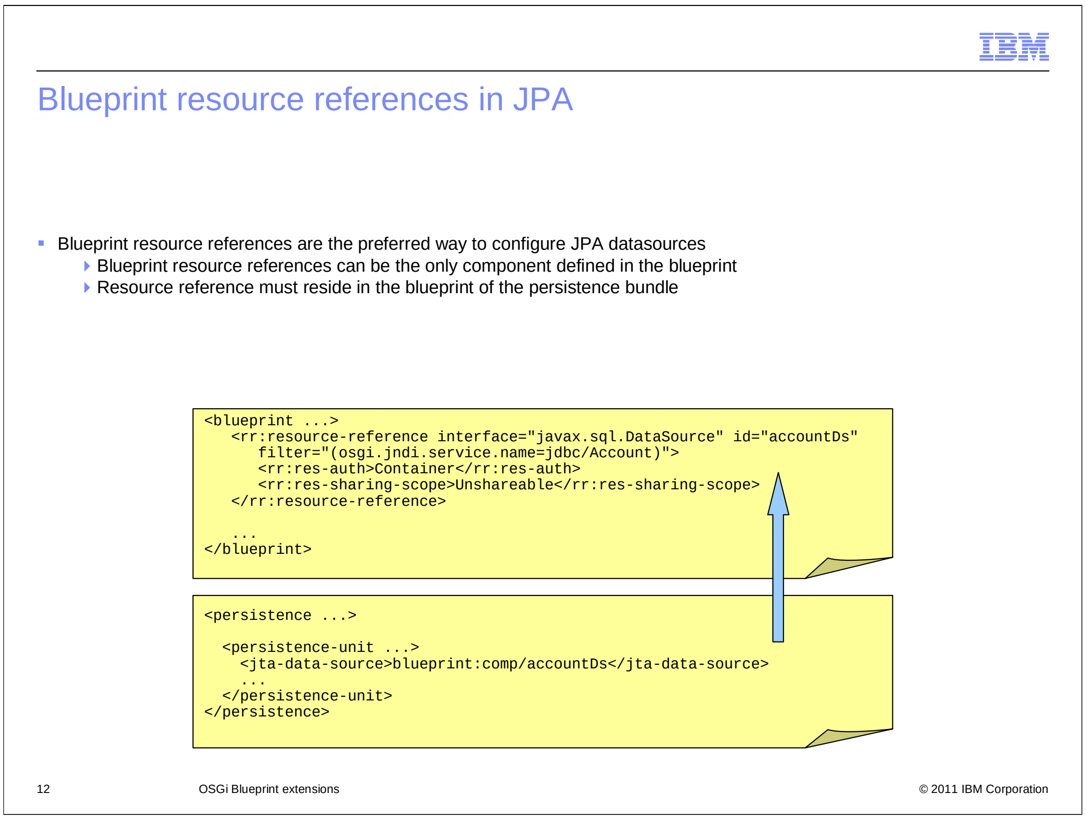
The only top-level element in the resource reference namespace is the <resource-reference> tag. This defines a stand-alone blueprint component that is based on the standard <reference> component. So it supports timeout, activation, availability and any other attributes available for reference components.

In particular, the interface and filter attributes that are inherited from reference components acquire additional semantics. The interface attribute defines the resource type (similar to the res-type tag for web module resource references). The filter attribute defines the JNDI name. Note that the filter can contain more than just the JNDI name, but nonetheless the JNDI name must be present.

The two permissible child elements <res-auth> and <res-sharing-scope> correspond exactly to the web module resource reference elements of the same name. Both elements are optional and are defaulted as shown.

The <res-auth> child element specifies whether the application will supply its own authorization credentials or whether the container is doing it for you. If you specify “Container” then you will be asked to map an authentication alias at deployment time.

The <res-sharing-scope> child element specifies whether the resource is shared between applications or whether the connections provided for it should never be sharable.



Blueprint resource references in JPA

- Blueprint resource references are the preferred way to configure JPA datasources
  - Blueprint resource references can be the only component defined in the blueprint
  - Resource reference must reside in the blueprint of the persistence bundle

```
<blueprint ...>
   <rr:resource-reference interface="javax.sql.DataSource" id="accountDs"
      filter="(osgi.jndi.service.name=jdbc/Account)">
      <rr:res-auth>Container</rr:res-auth>
      <rr:res-sharing-scope>Unshareable</rr:res-sharing-scope>
   </rr:resource-reference>

   ...
</blueprint>
```

```
<persistence ...>

  <persistence-unit ...>
    <jta-data-source>blueprint:comp/accountDs</jta-data-source>
    ...
  </persistence-unit>
</persistence>
```

12 OSGi Blueprint extensions © 2011 IBM Corporation

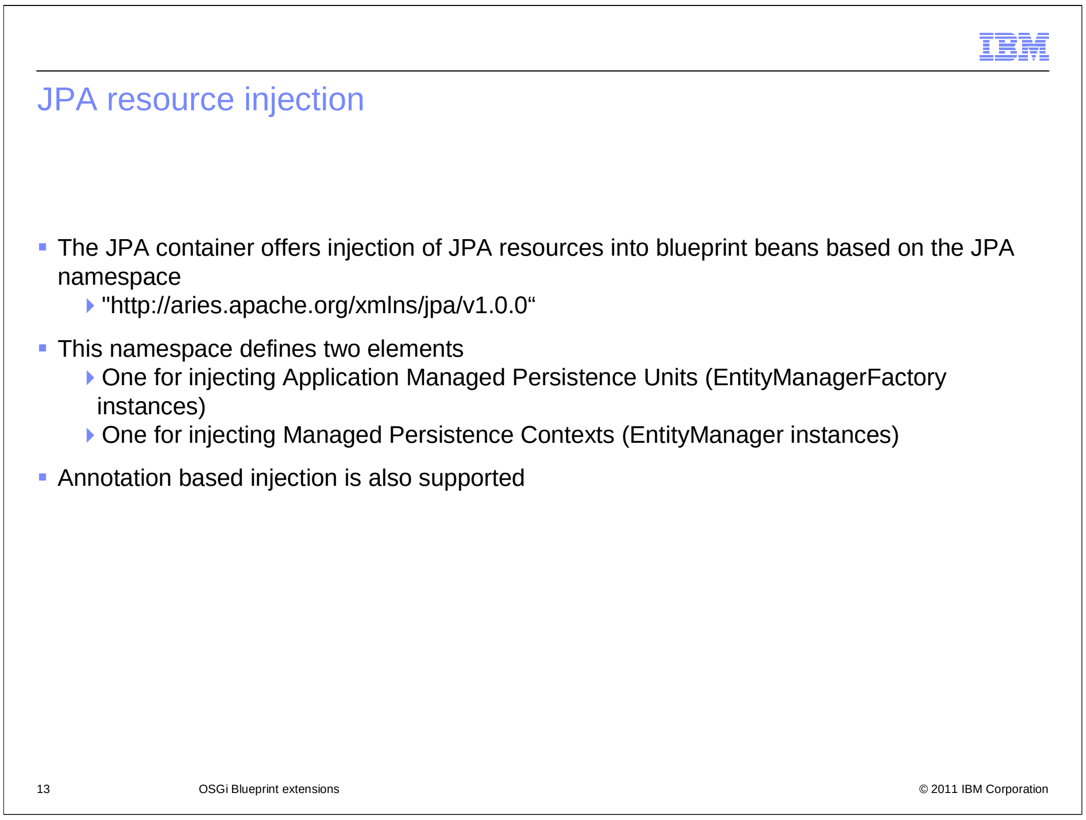
One particular use case of resource references is JPA data sources. This is actually the preferred way to configure a JPA resource because this is the only way to use secured data sources.

The resource reference is configured as described before but always with a component id. This component ID can then be used in a blueprint:comp/ lookup from the persistence descriptor as shown.

Note that the blueprint xml must be located in the persistence bundle (the bundle that contains the persistence descriptor with the blueprint:comp/ lookup). Blueprint resource references from other bundles cannot be used by JPA.

Also note that it is entirely legal for a persistence bundle's blueprint to contain nothing but the resource reference definitions for JPA. So using resource references for JPA does not require the persistence bundle to use blueprint for anything else.

It is entirely legal for the blueprint to contain only the resource reference but the resource reference must be declared in the persistence bundle, that is the same bundle which contains the persistence.xml.



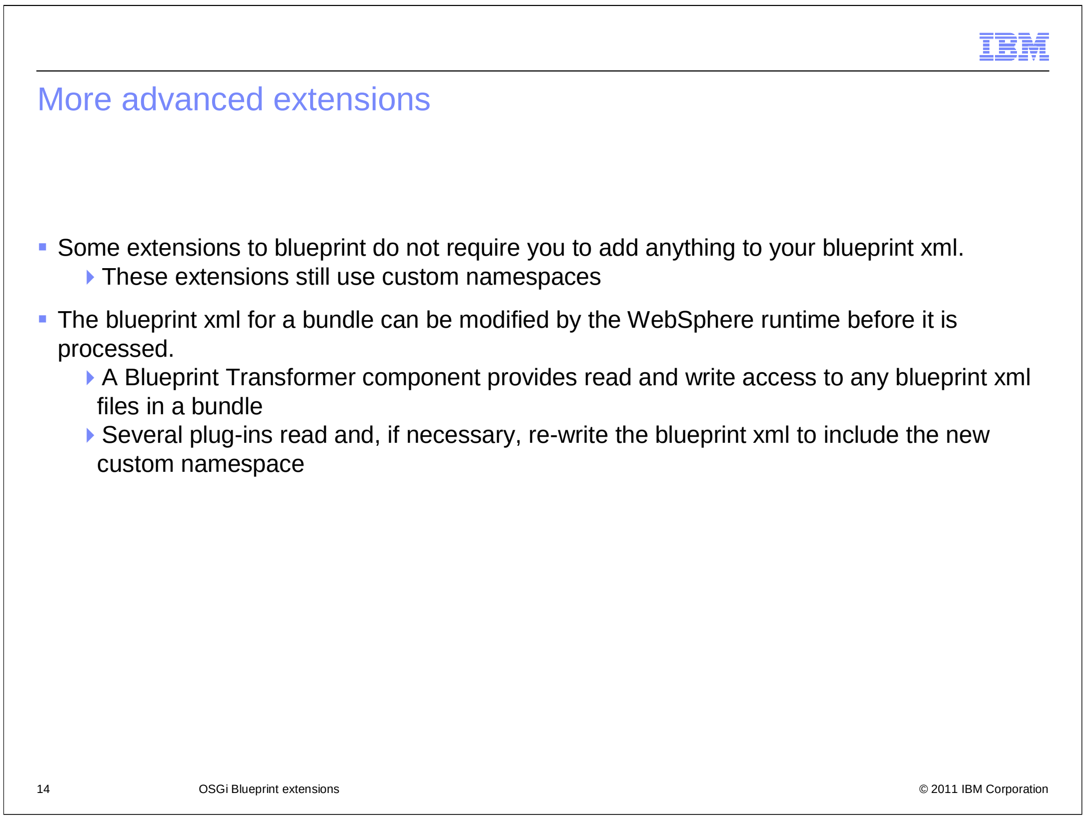
JPA resource injection

- The JPA container offers injection of JPA resources into blueprint beans based on the JPA namespace
  - "http://aries.apache.org/xmlns/jpa/v1.0.0“
- This namespace defines two elements
  - One for injecting Application Managed Persistence Units (EntityManagerFactory instances)
  - One for injecting Managed Persistence Contexts (EntityManager instances)
- Annotation based injection is also supported

13 OSGi Blueprint extensions © 2011 IBM Corporation

The JPA container uses blueprint extensions to support the injection of EntityManagerFactory and EntityManager instances.

This is covered in more detail in the JPA education module.



More advanced extensions

- Some extensions to blueprint do not require you to add anything to your blueprint xml.
  - These extensions still use custom namespaces
- The blueprint xml for a bundle can be modified by the WebSphere runtime before it is processed.
  - A Blueprint Transformer component provides read and write access to any blueprint xml files in a bundle
  - Several plug-ins read and, if necessary, re-write the blueprint xml to include the new custom namespace

14 OSGi Blueprint extensions © 2011 IBM Corporation

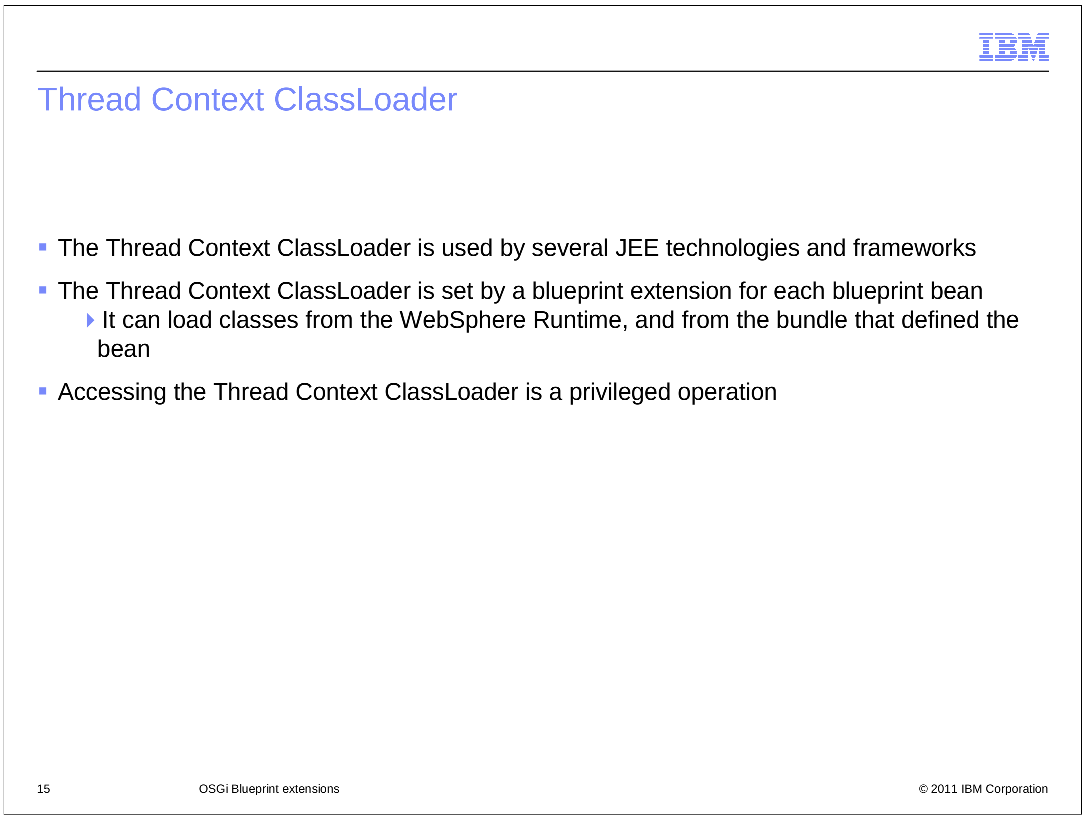
Some blueprint extensions (for example JPA) need to be able to operate without being explicitly listed in the blueprint xml. In order to make this work the JPA container uses a plug point to read and re-write the blueprint xml files to include the JPA xml namespace.

A similar technique is used to add “Default Interceptors”

If there is an error in re-writing the blueprint xml then there are errors in the logs, and the application is unlikely to work correctly.

It should also be noted that as the blueprint xml is re-written, any warnings or errors can include generated xml that was not present in the original xml document.

Your bundle is not modified, the re-written xml will only be present in the runtime.

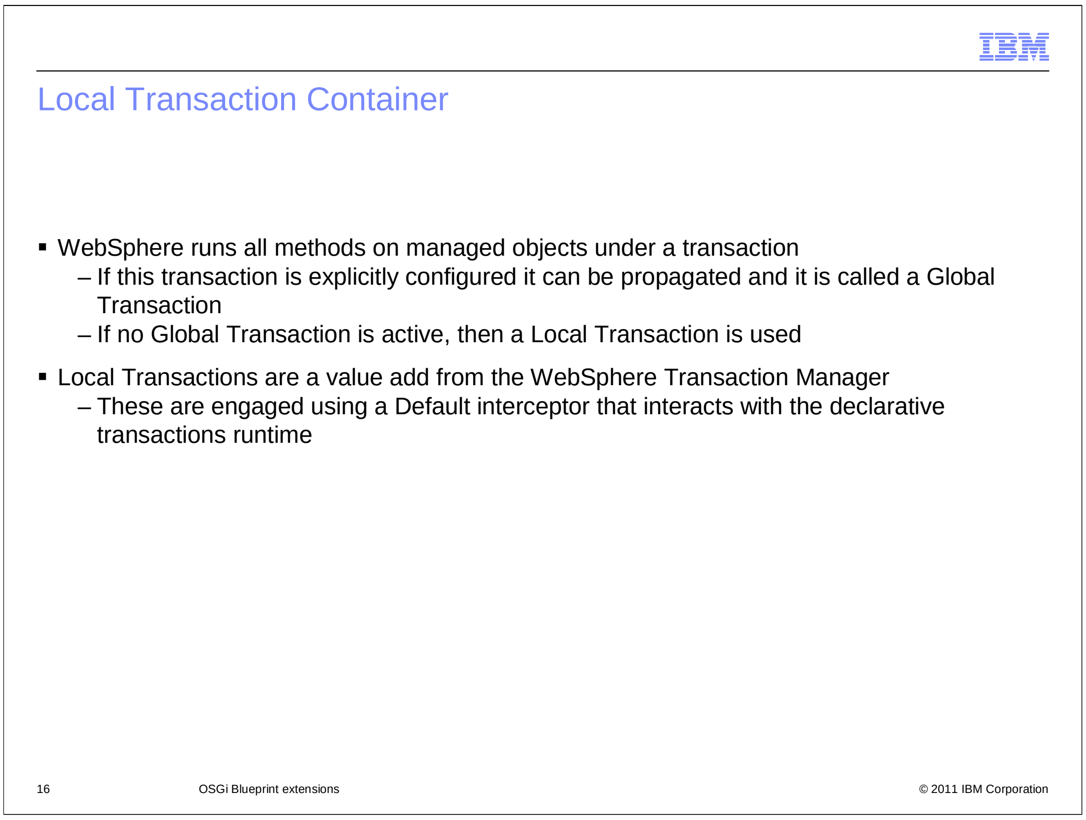

Thread Context ClassLoader

- The Thread Context ClassLoader is used by several JEE technologies and frameworks
- The Thread Context ClassLoader is set by a blueprint extension for each blueprint bean
  - It can load classes from the WebSphere Runtime, and from the bundle that defined the bean
- Accessing the Thread Context ClassLoader is a privileged operation

15 OSGi Blueprint extensions © 2011 IBM Corporation

Many JEE technologies use the Thread Context ClassLoader to access application classes, so the blueprint extensions runtime sets the Thread Context ClassLoader appropriately for each blueprint bean. The classloader that is set can load core WebSphere Application Server classes, and classes from the bundle that defines the blueprint bean.

If an Application is deployed to a server with Java 2 security enabled, and that application wants to access the Thread Context ClassLoader then it will need to have the relevant permissions.



Local Transaction Container

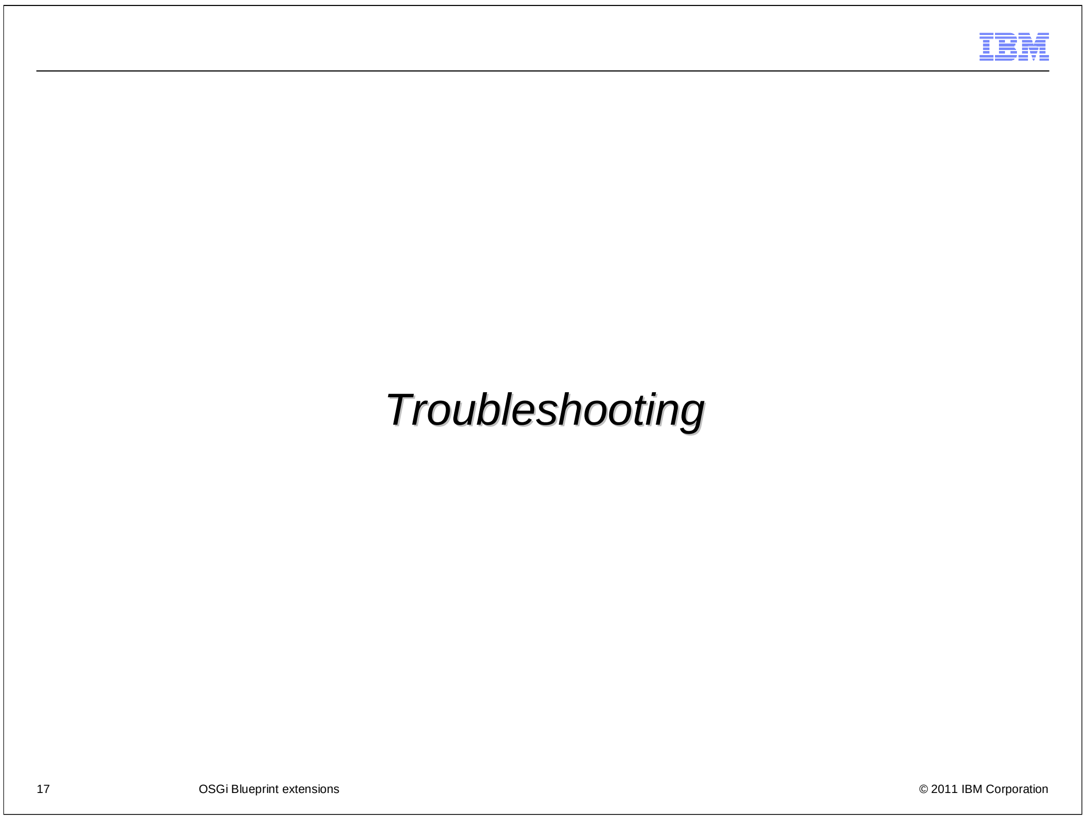
- WebSphere runs all methods on managed objects under a transaction
  - If this transaction is explicitly configured it can be propagated and it is called a Global Transaction
  - If no Global Transaction is active, then a Local Transaction is used
- Local Transactions are a value add from the WebSphere Transaction Manager
  - These are engaged using a Default interceptor that interacts with the declarative transactions runtime

16 OSGi Blueprint extensions © 2011 IBM Corporation

All blueprint bean invocations are run under a transaction. If explicitly configured, these are global transactions, if not Local Transactions are used. This is one of the key pieces of value add offered by the WebSphere runtime.

Local Transactions are provided in a very similar way to the declarative transactions listed earlier, but do not require any specific configuration. If a Global Transaction is configured it takes precedence over a Local Transaction, however there will ALWAYS be a transaction active when a blueprint bean is invoked.

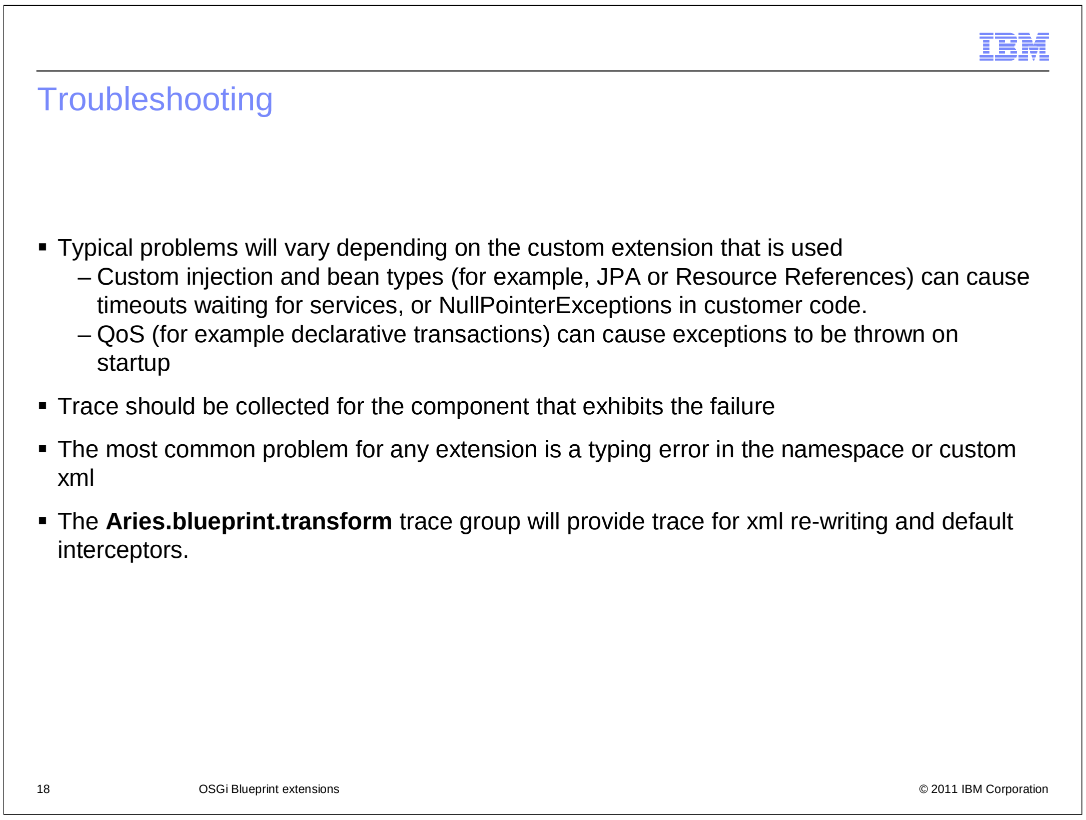
The Local Transactions Container interceptor cooperates with the declarative transactions runtime to ensure the correct transaction semantics are obeyed.



Troubleshooting

17 OSGi Blueprint extensions © 2011 IBM Corporation

This section covers some basic troubleshooting tips.



Troubleshooting

- Typical problems will vary depending on the custom extension that is used
  - Custom injection and bean types (for example, JPA or Resource References) can cause timeouts waiting for services, or NullPointerExceptions in customer code.
  - QoS (for example declarative transactions) can cause exceptions to be thrown on startup
- Trace should be collected for the component that exhibits the failure
- The most common problem for any extension is a typing error in the namespace or custom xml
- The **Aries.blueprint.transform** trace group will provide trace for xml re-writing and default interceptors.

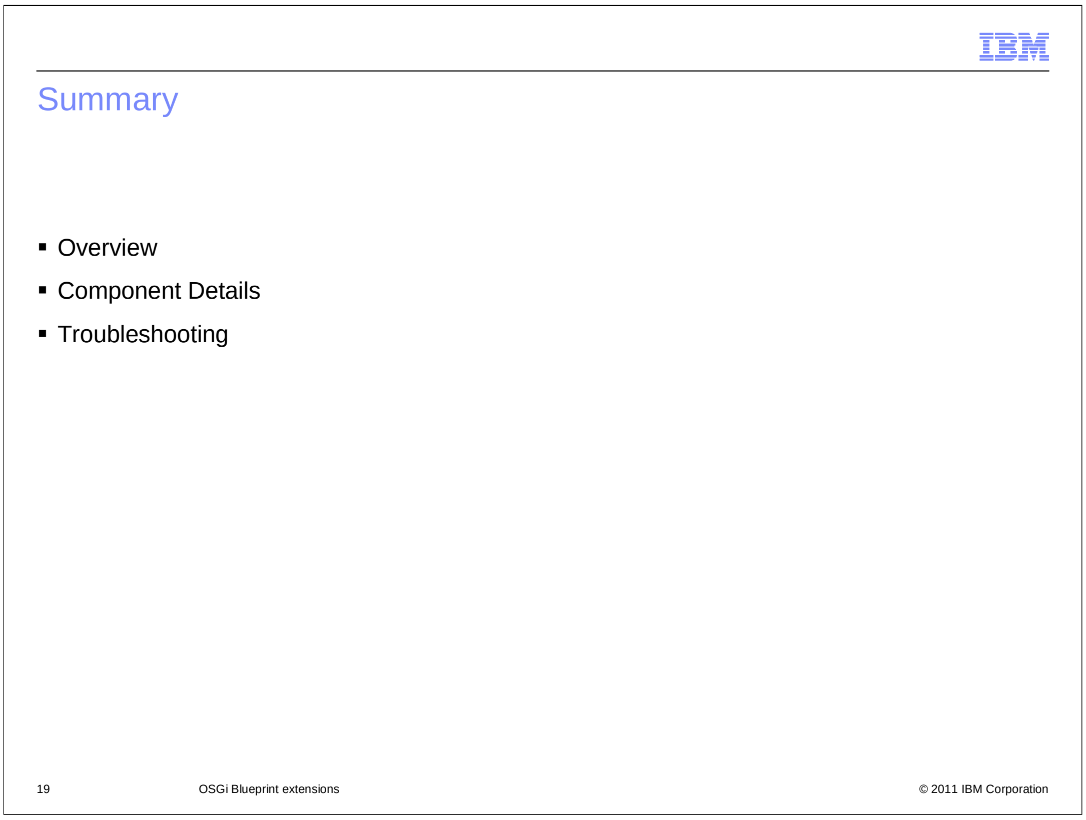
18 OSGi Blueprint extensions © 2011 IBM Corporation

There are a large number of different potential problems for blueprint extensions, depending on the type of the extension. The best debug information to collect is from the component that is affected. If it is a failure referencing transactions then you should collect from the Aries transactions runtime, if it is JPA then you should collect from the JPA container.

It is also useful to collect trace for Aries.blueprint and Aries.blueprint.transform because they will tell you about the XML rewriting and will give you more information about the blueprint processing.

It should be remembered that the most common errors are mistakes in the custom namespace or element names. Since these things are not checked by the tools it is entirely possible that a one character typing error will occur and be the cause of an error.

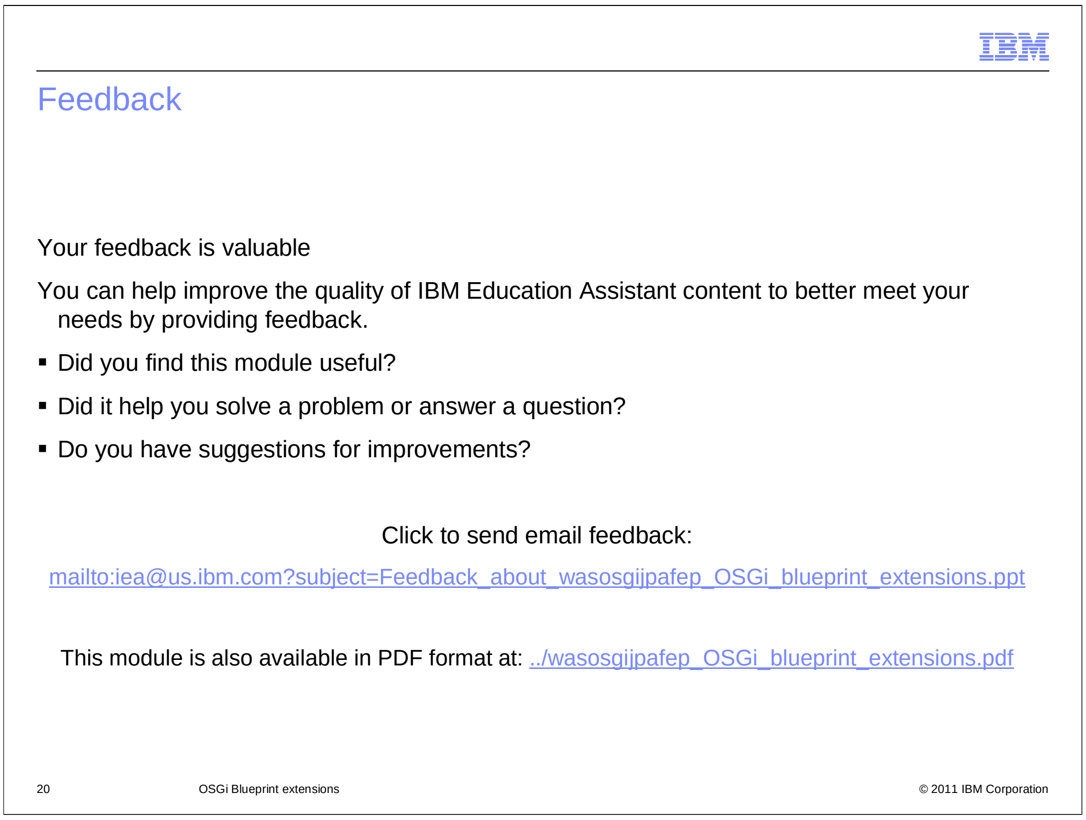
It is also possible to get a null pointer exception if you make a mistake in an annotation and the resulting XML rewrite is then incorrect. The original error is higher up in the logs where it tells you why it was unable to transform the XML.



Summary

- Overview
- Component Details
- Troubleshooting

In this module you have seen an overview of what a blueprint extension is. In the details section you learned how these extensions are accomplished and specifically learned about declarative transactions and resource references. Finally there were some tips on 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?

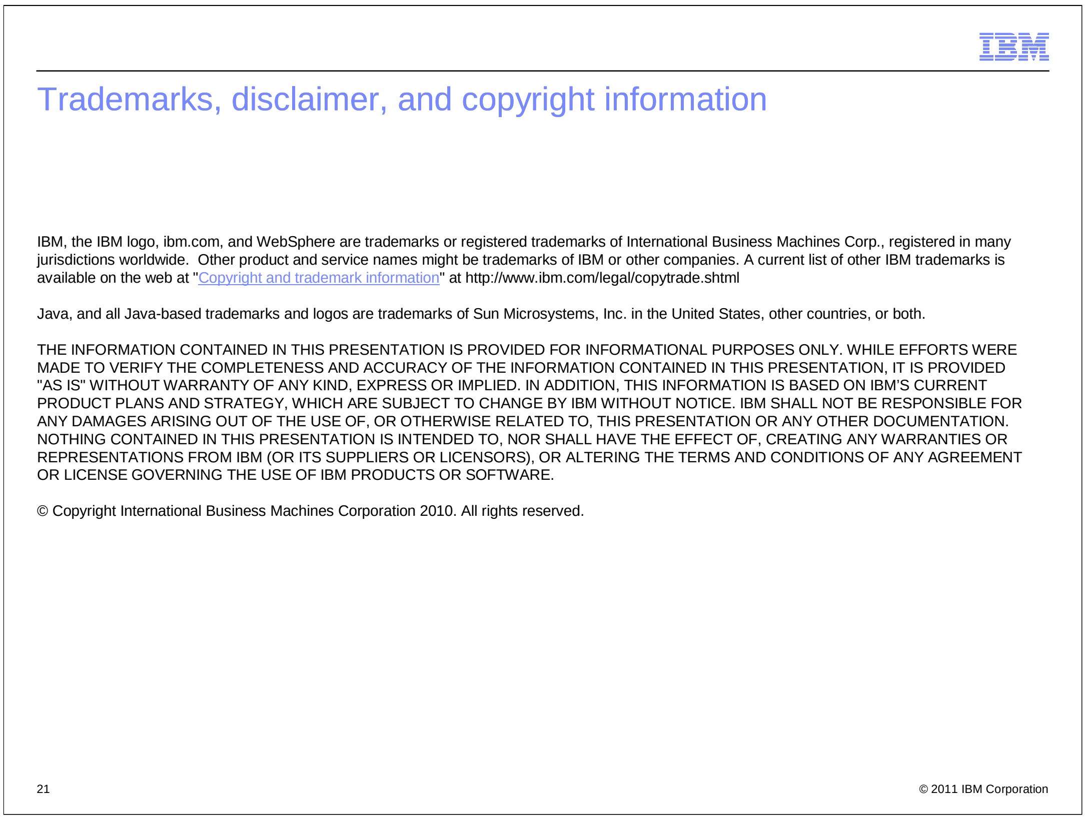
Click to send email feedback:

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

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

20 OSGi Blueprint extensions © 2011 IBM Corporation

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



Trademarks, disclaimer, and copyright information