EoD Features and Component Dependencies in Java EE 5 / App Server 9.x

Kenneth Saks
EJB Container Lead
kenneth.saks@sun.com

Goal

- A closer look at component environment dependencies in Java EE 5 and AS 9.x
- Better understanding of :
  - > Difference between component namespace and App Server-specific(global) namespace
  - > Relationship between Java EE 5 dependency annotations, standard .xml descriptors, and vendor-specific .xml descriptors
  - > AS 9.x user experience improvements

Agenda

- Component dependencies in .xml
- Component dependencies via annotations
- AS 9.x user experience enhancements
- Q & A

Component Dependencies

- Declaration by bean developer that a component(ejb, servlet, jsp, app client) requires some data, resource, or other component.
- Examples :
  - > Using a DataSource to access a database
  - > Invoking an Enterprise JavaBean
  - > Consuming a web service
  - > Sending a JMS message
  - > Accessing a configurable property value

Advantages of declaring component dependencies

- Better deployment-time error checking
- Separation of concerns : development vs. deployment
- Portability : clear distinction between standard information and vendor-specific information
- Prevents hard-coding. Actual mappings/values can be changed without touching code.

Servlet accessing database

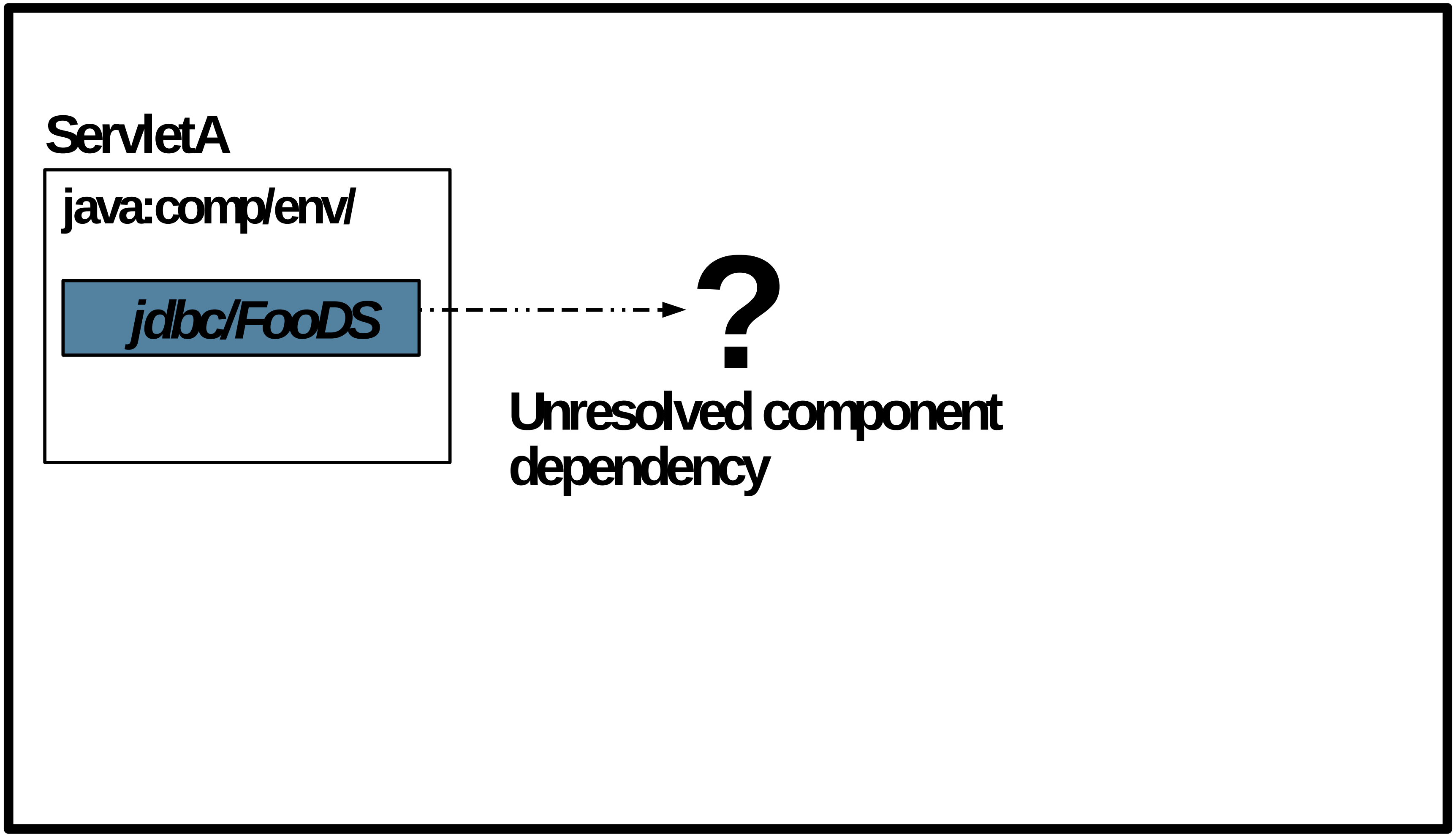
ServletA.java

```
InitialContext ic = new InitialContext();
DataSource ds = (DataSource)
    ic.lookup("java:comp/env/jdbc/FooDS");
```

web.xml

```
<resource-ref>
  <res-ref-name>jdbc/FooDS</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
</resource-ref>
```

Component namespace w/o physical resource mapping



Creating physical AppServer resources

```
% asadmin create-jdbc-resource
"  jdbc/OracleDS"   ...

% asadmin create-jdbc-resource
"  jdbc/DerbyDS"   ...

% asadmin create-jdbc-resource
"  jdbc/FooDS"   ...

% asadmin create-jms-resource
"  jms/FooQueue"   ...
```

App Server Global
JNDI Namespace

/

jdbc/OracleDS

jdbc/DerbyDS

jdbc/FooDS

jms/FooQueue

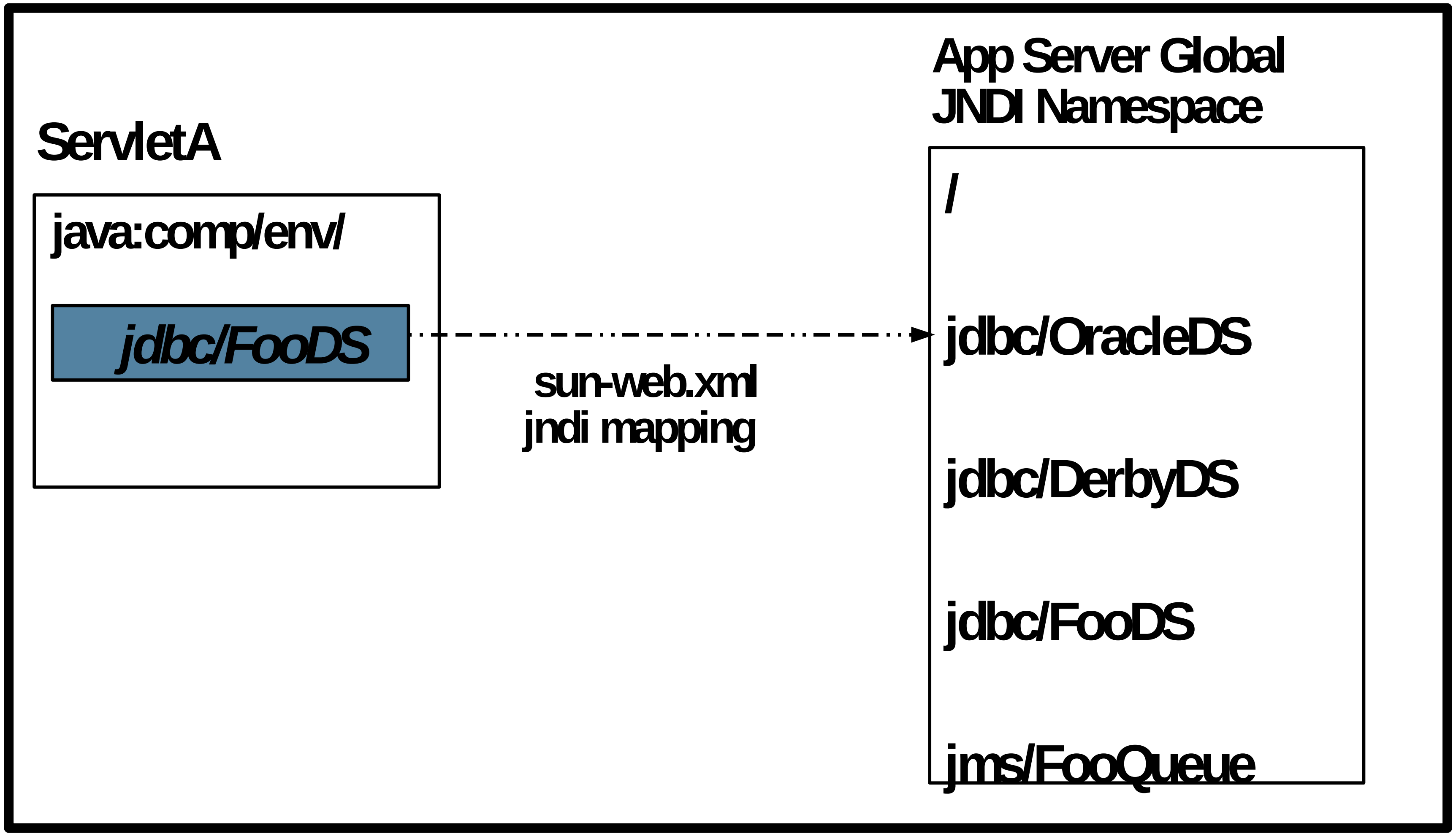
Servlet accessing database (cont.)

sun-web.xml

```
<resource-ref>
  <res-ref-name>jdbc/FooDS</res-ref-name>
  <jndi-name>jdbc/OracleDS</jndi-name>
</resource-ref>
```

- res-ref-name is the name of a component dependency and is relative to java:comp/env
- jndi-name is the name of a physical App Server resource and is relative to the root of the App Server's global namespace

Namespaces


ServletA
java:comp/env/
jdbc/FooDS
sun-web.xml
jndi mapping
App Server Global
JNDI Namespace
/
jdbc/OracleDS
jdbc/DerbyDS
jdbc/FooDS
jms/FooQueue

Notes on App Server Global Namespace

- Completely implementation-specific
  - > not covered by Java EE specification
- Direct global lookups are not portable

```
InitialContext ic = new InitialContext();
// not portable :-(
ic.lookup("jdbc/OracleDS");

// portable :-)
ic.lookup("java:comp/env/jdbc/FooDS");
```

Java EE 5 Annotations for Component Dependencies

```
@Resource(name="jdbc/FooDS")
private DataSource ds;
```

- This does TWO things :
  - Declares a component environment dependency "java:comp/env/*jdbc/FooDS*"
  - Registers the field "ds" for dependency injection

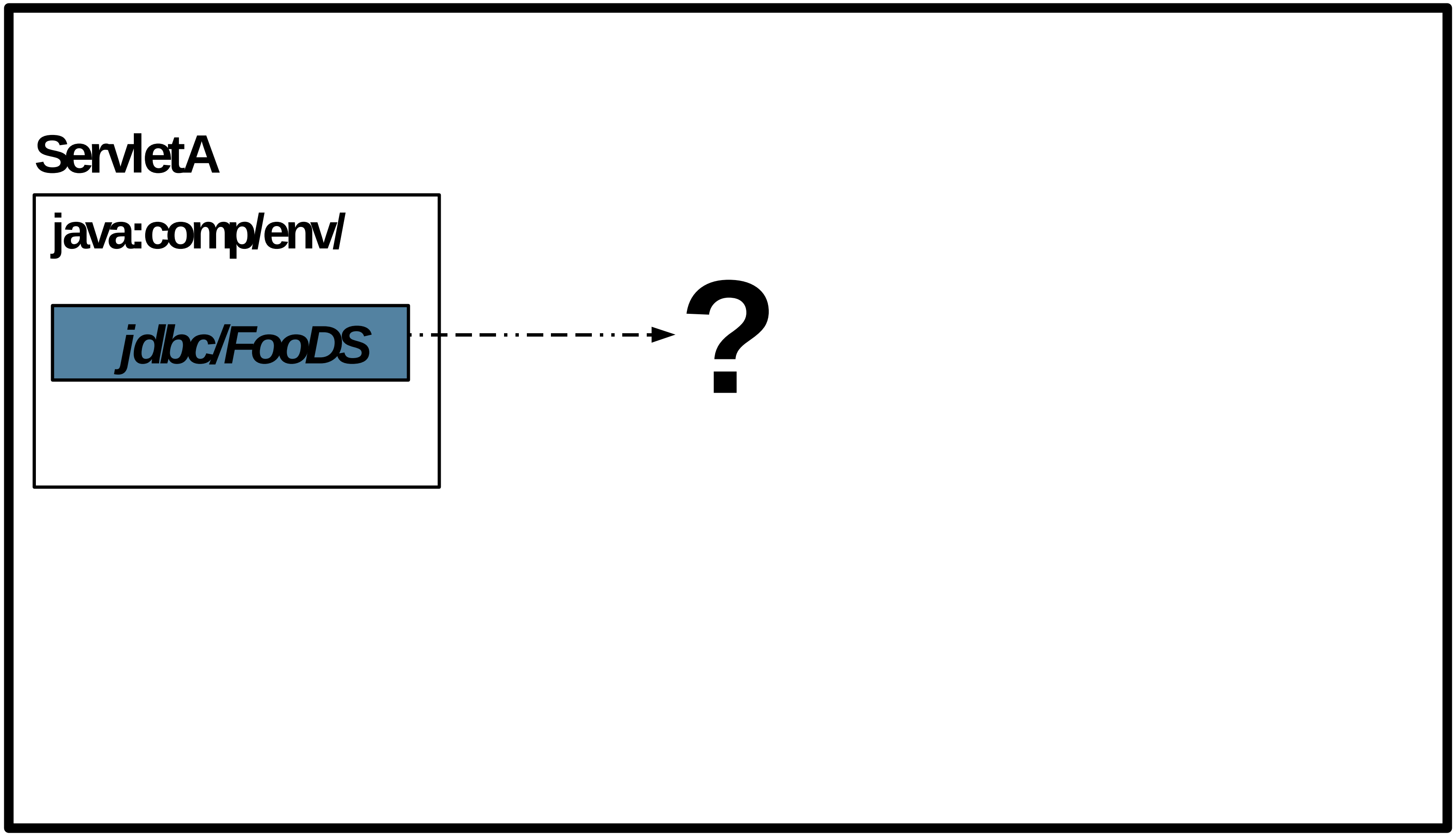
.xml representation of @Resource

```
private @Resource(name="jdbc/FooDS") DataSource ds;
```

Is equivalent to :

```
<resource-ref>
  <res-ref-name>jdbc/FooDS</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <injection-target>
    <injection-target-class>com.acme.ServletA</...>
    <injection-target-name>ds</injection-target-name>
  </injection-target>
</resource-ref>
```

Component namespace w/o mapping



dependency annotation name() defaults

- FIELD : <class-name>/<field-name>
- METHOD : <class-name>/<setter-property-name>
- TYPE : No defaulting. name() is required.

Example

@Resource private DataSource ds;

is equivalent to :

@Resource(name=” com.acme.ServletA/ds”)

private DataSource ds;

Dependencies declared by annotation

- Question : Why so much attention to java:comp/env and name() attributes for annotations? Isn't all this supposed to be hidden from the developer in Java EE 5?
- Answer :
  - > java:comp/env name might be needed to resolve the dependency.
  - > might want to do a java:comp/env lookup instead of or in addition to injection
  - > might want to override some of the dependency attributes in the deployment descriptor.

Resolving component dependencies

- Java EE 5 reduces need for standard .xml descriptors (ejb-jar.xml, web.xml, etc.)
- For maximum ease-of-use in AS 9.x, we must find similar ways to reduce need for sun-*.xml.
- Observation : In large % of cases, sun-*.xml files contain only/mostly dependency mappings.
  - “ low hanging fruit”
  - Especially true of simpler/smaller apps.

sun-*.xml ease-of-use improvements

- Simplifying dependency mapping should result in significant ease-of-use improvements for AS 9.x.
- Two main approaches for reducing need to specify resource mappings in sun-*.xml :
  - “ mapped-name”
  - use of defaults

mapped-name

- new attribute defined within many Java EE 5 annotations(e.g. @Resource, @EJB, @Stateless) and deployment descriptors.

Example :

```
// TYPE-level @Resource in Servlet
@Resource(name=” jdbc/FooDS” ,
          type=javax.sql.DataSource.class,
          mappedName=” jdbc/OracleDS” )
public class ServletA { ... }
```

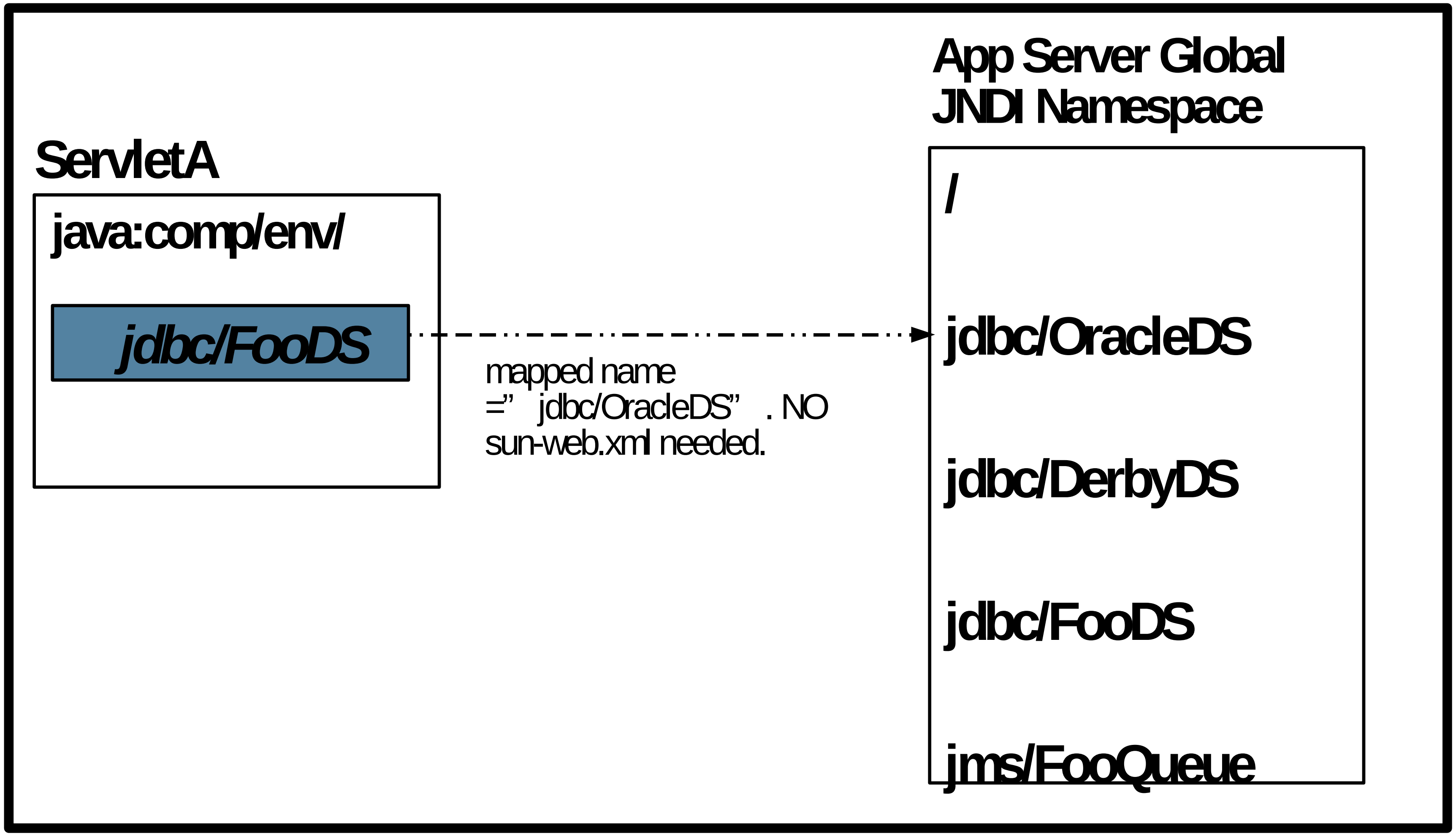
mapped-name in web.xml

```
@Resource(name=" jdbc/FooDS" ,
          type=javax.sql.DataSource.class,
          mappedName=" jdbc/OracleDS" )
public class ServletA { ... }
```

Is equivalent to :

```
<resource-ref>
  <res-ref-name>jdbc/FooDS</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <mapped-name>jdbc/OracleDS</mapped-name>
</resource-ref>
```

Namespaces



mapped name =" jdbc/OracleDS" . NO sun-web.xml needed.

mapped-name (cont.)

- mapped-name is not required to be supported by a Java EE 5 implementation
  - > value is ignored if not supported
- syntax/semantics of mapped-name values are vendor-specific
- In AS 9.x, order of precedence(lowest to highest) for dependency mappings is :
  - > annotation < standard .xml < sun-*.xml
  - > Allows any use of mapped-name to be overridden by deployer within sun-*.xml

mapped-name (cont.)

- AS 9.x will support mapped-name for the following annotations :
  - @Resource, @EJB
  - @Stateless, @Stateful, @MessageDriven
- AS 9.x will support mapped-name for the following standard .xml elements :
  - resource-ref, resource-env-ref, message-destination-ref, message-destination, ejb-ref
  - session, entity, message-driven

Remote EJB mapped-name example

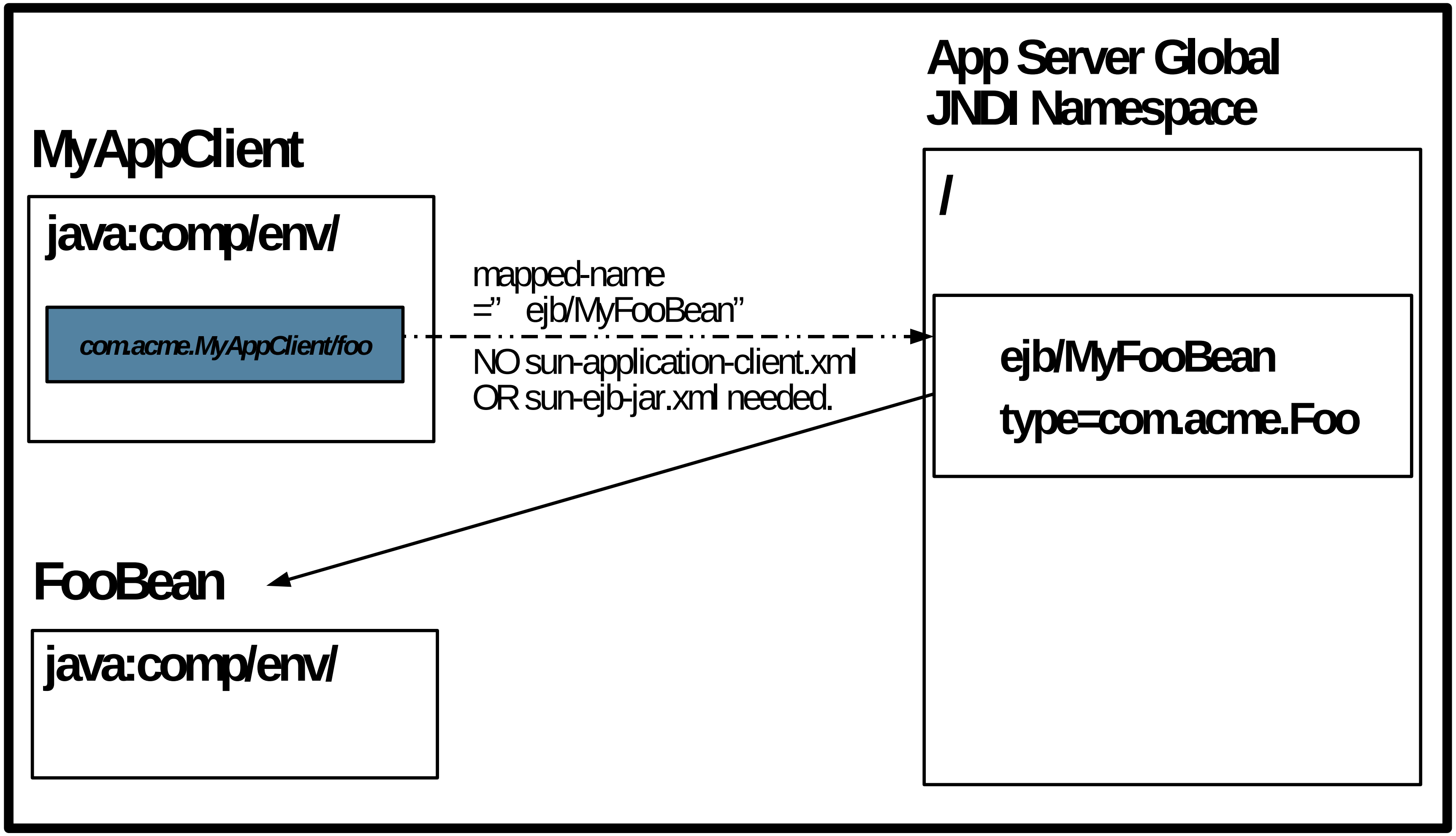
FooBean.java :

```
@Stateless(mappedName=” ejb/MyFooBean” )
public class FooBean implements Foo {
  public String hello() { return “ hello, world!\n” ; }
}
```

MyAppClient.java :

```
@EJB(mappedName=” ejb/MyFooBean” )
private static Foo foo;
```

Namespaces

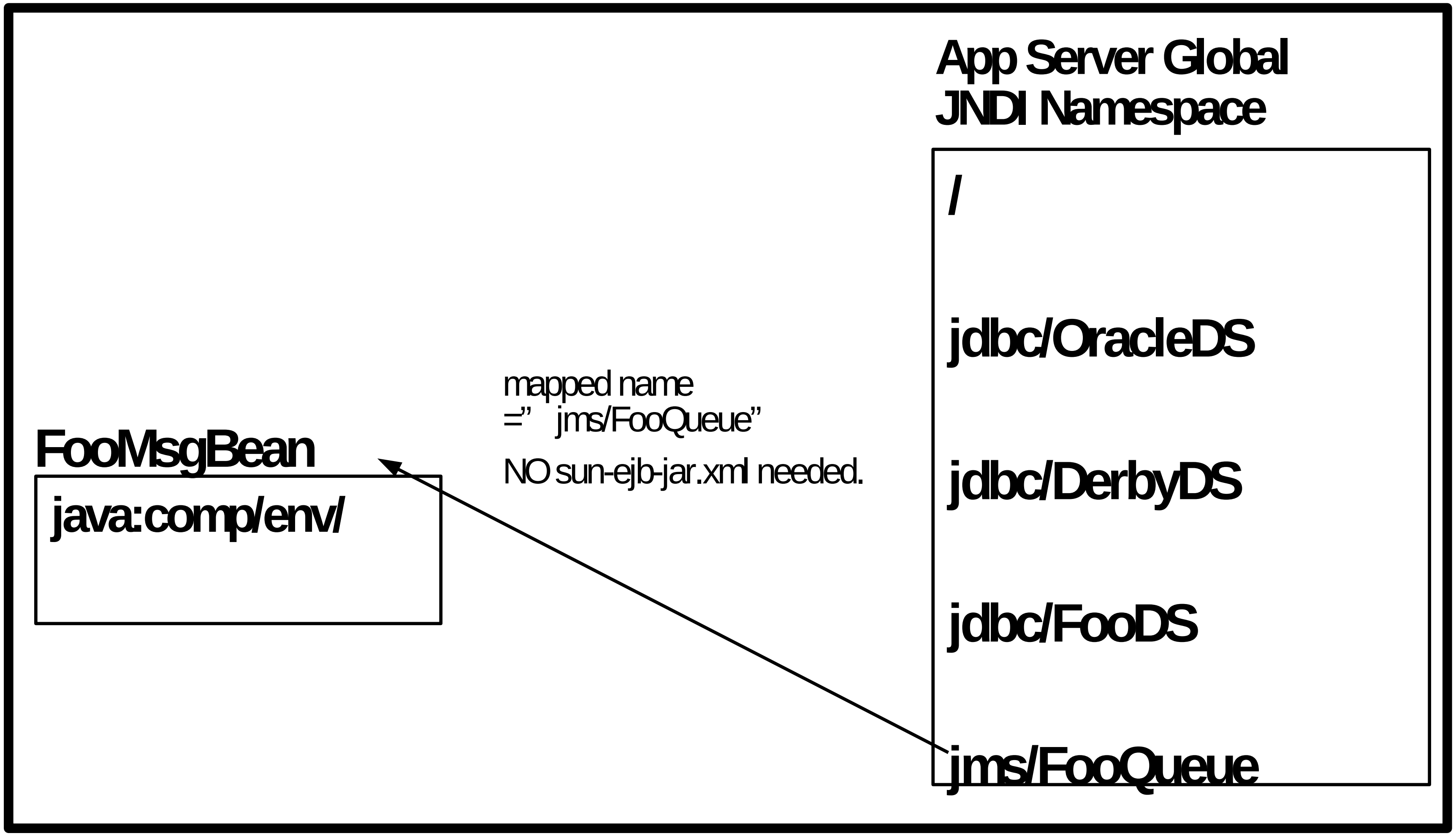

App Server Global
JNDI Namespace
MyAppClient
java:comp/env/
com.acme.MyAppClient/foo
mapped-name
="ejb/MyFooBean"
NO sun-application-client.xml
OR sun-ejb-jar.xml needed.
/
ejb/MyFooBean
type=com.acme.Foo
FooBean
java:comp/env/

MDB mapped-name example

FooMsgBean.java :

```
@MessageDriven(mappedName=”jms/FooQueue”)
public class FooMsgBean {
  public void onMessage(Message msg) { ... }
}
```

Namespaces


App Server Global JNDI Namespace
/
jdbc/OracleDS
jdbc/DerbyDS
jdbc/FooDS
jms/FooQueue
mapped name =" jms/FooQueue"
NO sun-ejb-jar.xml needed.
FooMsgBean
java:comp/env/

Component dependency defaulting

- If a component dependency that requires sun-*.xml mapping has NOT been resolved at deployment-time
  - > For Session beans and ejb-refs
    - > jndi-name is set to Home/Business interface name
  - > For everything else (@Resource, resource-ref, message-destination-ref, etc.)
    - > jndi-name is set to the resource dependency's java:comp/env name

Example : @Resource default mapping

@Resource(name=" *jdbc/FooDS*")

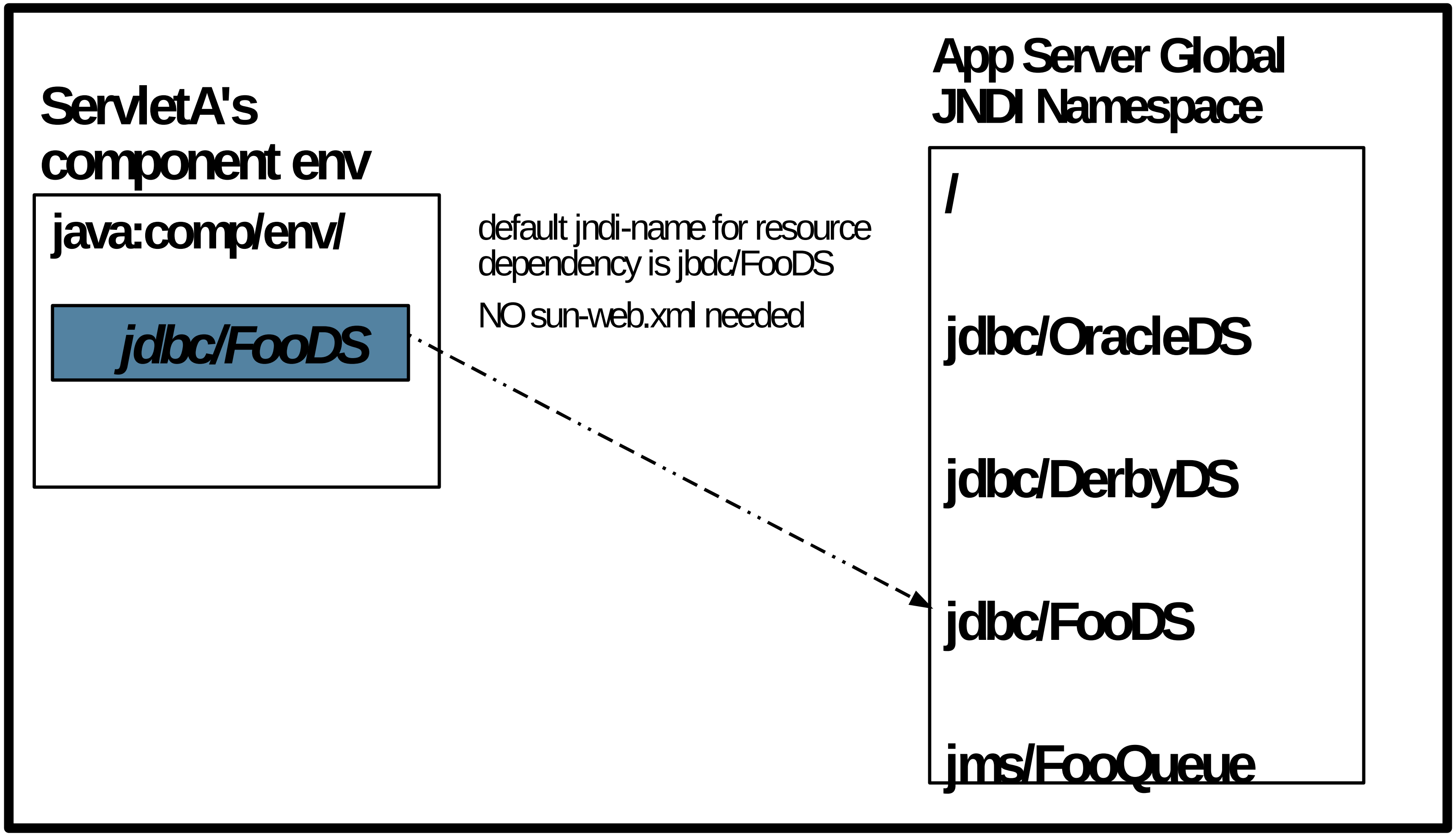
private DataSource ds;

- If the java:comp/env/jdbc/FooDS dependency is unresolved at deployment time, AS 9.x treats it as :

<u>sun-web.xml</u>

```
<resource-ref>
  <res-ref-name>jdbc/FooDS</res-ref-name>
  <jndi-name>jdbc/FooDS</jndi-name>
</resource-ref>
```

Namespaces


ServletA's component env
java:comp/env/
jdbc/FooDS
default jndi-name for resource dependency is jbdc/FooDS
NO sun-web.xml needed
App Server Global JNDI Namespace
/
jdbc/OracleDS
jdbc/DerbyDS
jdbc/FooDS
jms/FooQueue

Example : EJB jndi default

FooBean.java :

```
@Stateless(name=" FooBean" )
public class FooBean implements Foo { .. }
```

- If FooBean hasn't been assigned a jndi-name at deployment time, AS 9.x treats it as :

sun-ejb-jar.xml

```
<ejb>
  <ejb-name>FooBean</ejb-name>
  <jndi-name>com.acme.Foo</jndi-name>
</ejb>
```

Example : @EJB default mapping

MyAppClient.java :

@EJB(name=" *ejb/Foo*")

private static Foo foo;

- If the java:comp/env/ejb/Foo dependency is unresolved at deployment time, AS 9.x treats it as :

sun-application-client.xml

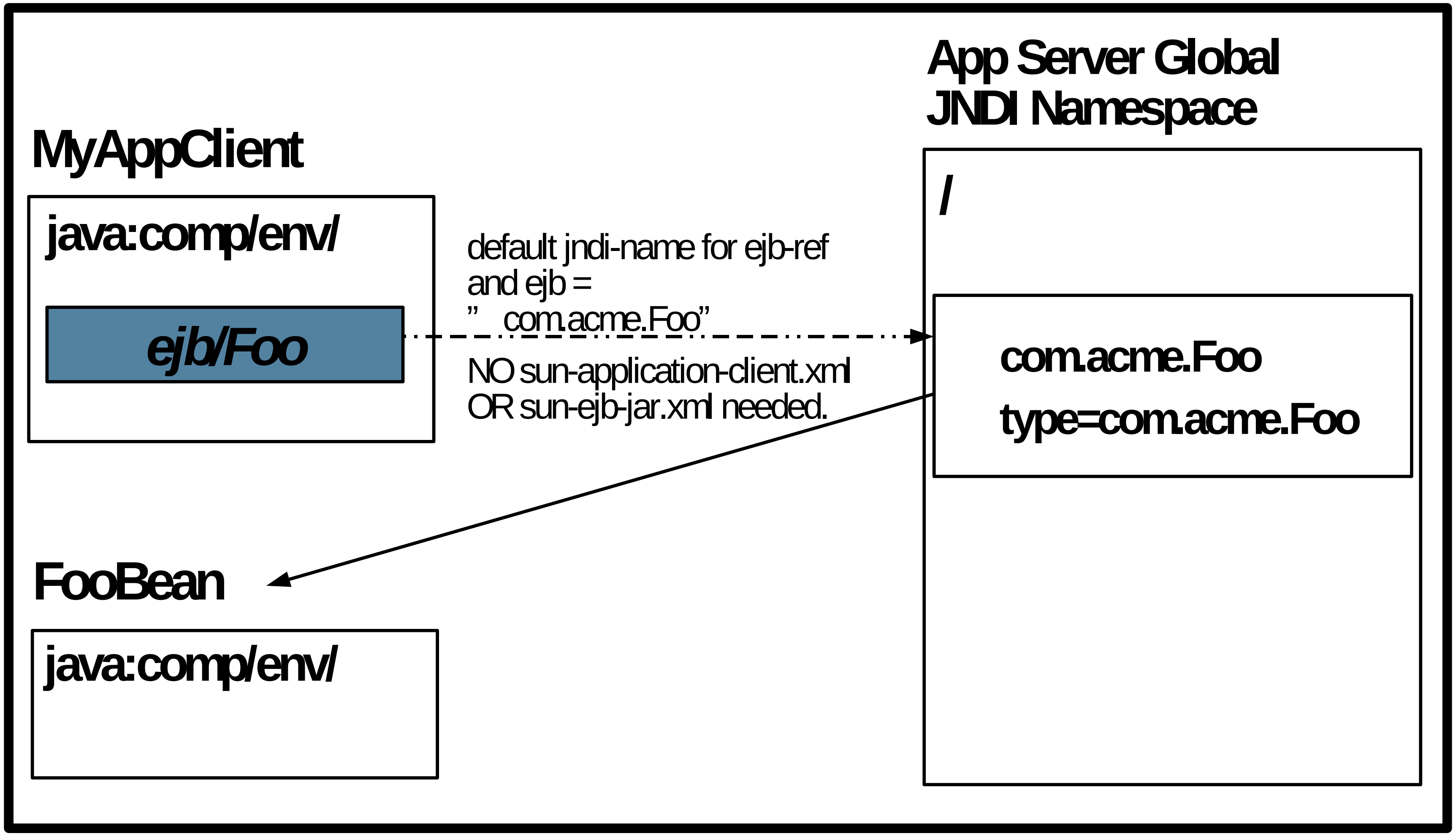
<ejb-ref>

  <ejb-ref-name>*ejb/Foo*</ejb-ref-name>

  <jndi-name>com.acme.Foo</jndi-name>

</ejb-ref>

Namespaces



App Server Global
JNDI Namespace
MyAppClient
java:comp/env/
ejb/Foo
default jndi-name for ejb-ref
and ejb =
" com.acme.Foo"
NO sun-application-client.xml
OR sun-ejb-jar.xml needed.
/
com.acme.Foo
type=com.acme.Foo
FooBean
java:comp/env/

Q & A

Component dependencies in AS 9.x

kenneth.saks@sun.com