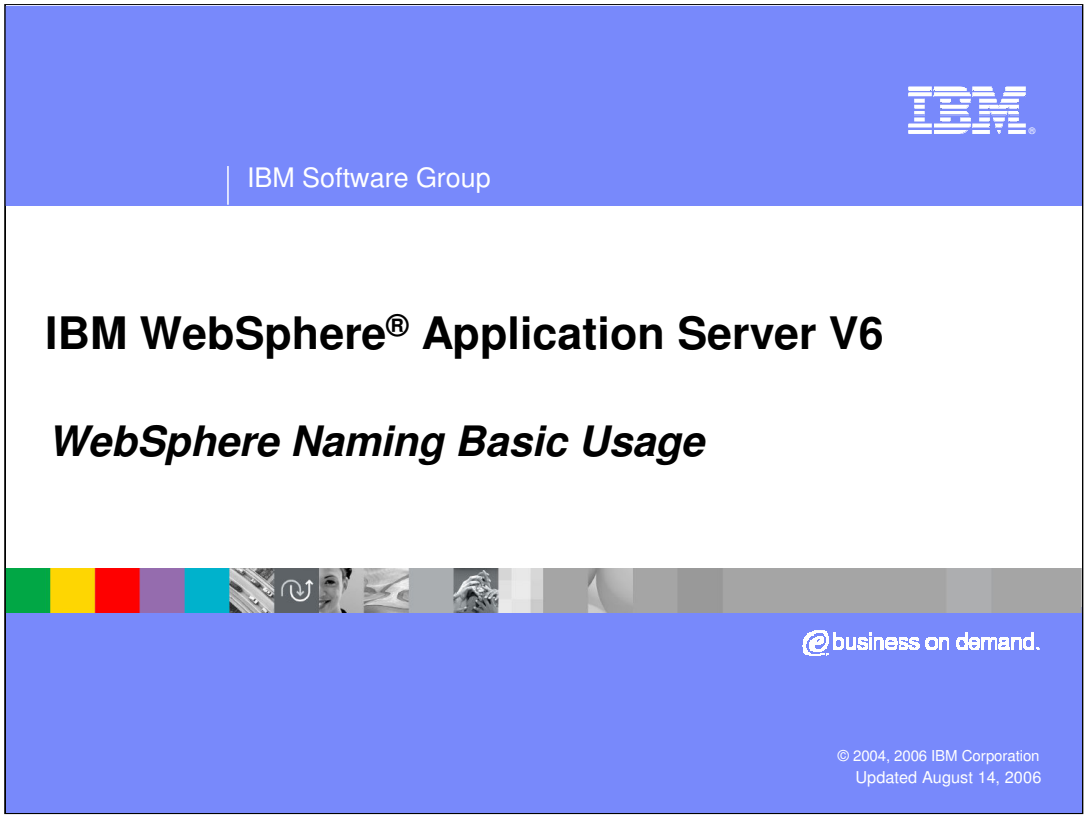
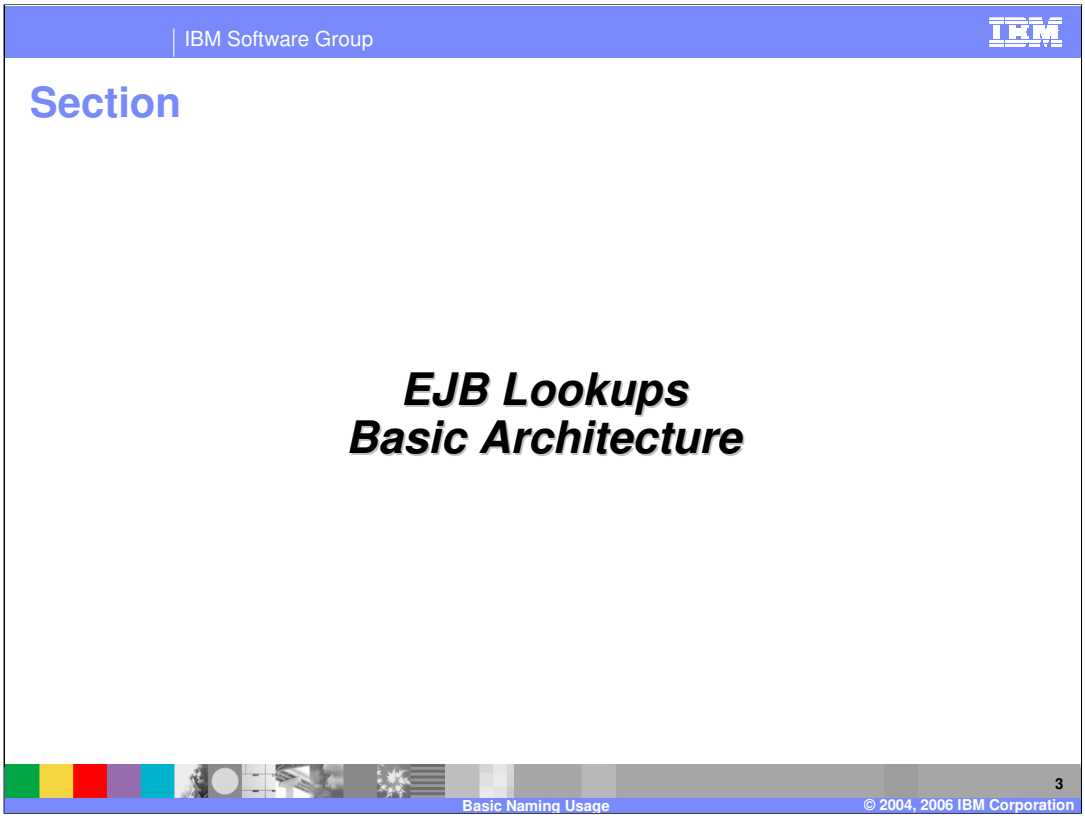


This presentation will provide a basic understanding of essential Naming functions in WebSphere Application Server Version 6. It would be helpful if you already have viewed the Naming Introduction presentation, although it is not a requirement for understanding this material. This presentation is a prerequisite to the other presentations that cover Naming Advanced Topics, Debugging, and Samples.

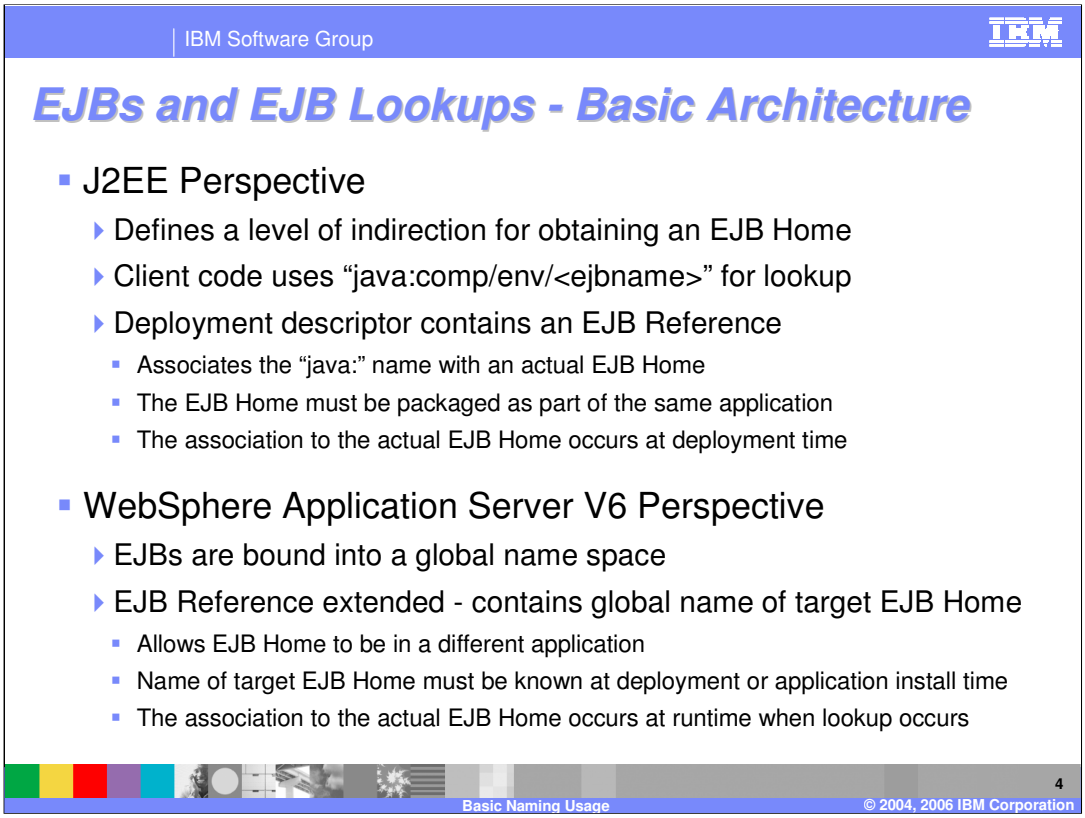
Goals

- Architectural view of Java™ 2 Enterprise Edition (J2EE) requirements and WebSphere Application Server V6 extensions
- Provide basic understanding of how to use Naming
  - Approach from a stand-alone server perspective
  - Use Enterprise JavaBeans (EJBs) as a basis for an end-to-end explanation
- Look at Naming in the full lifecycle of an EJB
  - Development
  - Deployment
  - Installation
  - Runtime
- Focus on developers' and administrators' needs
  - Show where naming related items are surfaced
  - Explain the relationships between different names
- Show how naming is used with Resources
- Show how local interface EJBs are handled

The goal of this presentation is to present the fundamental knowledge required by developers and administrators regarding the use of JNDI names and naming functionality. It approaches this by first reviewing both the J2EE and WebSphere Application Server V6 architecture that applies. Then the full lifecycle of an EJB is examined from code development through deployment, application installation and runtime operation, pointing out every place a JNDI name is used and how that name relates to the application function and to other JNDI names. There are many screen captures of the IBM Rational® Application Developer tools and WebSphere Application Server Administrative Console that show specifically where names are specified. All this is done assuming a stand-alone server environment, which eliminates many considerations that come into play in a Network Deployment environment which is covered in the Advanced Naming presentation. Once an understanding of EJBs and naming has been established, the use of naming with Resources and local interface EJBs is reviewed.

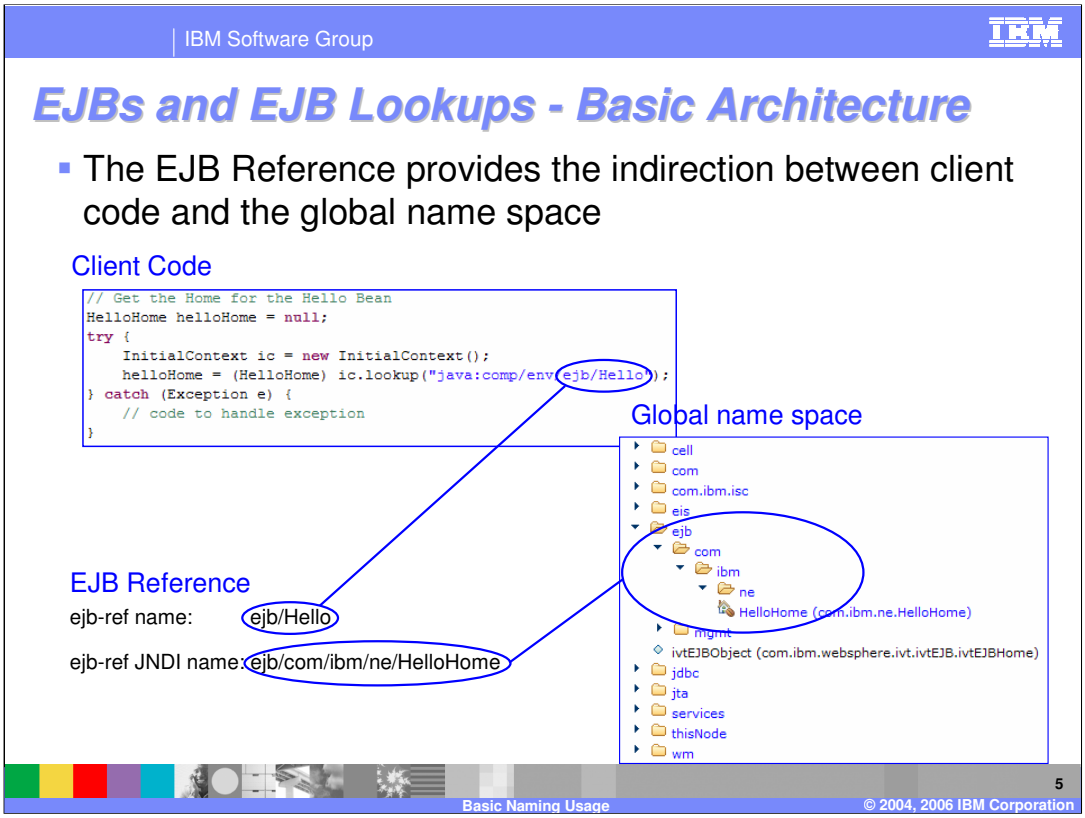


This section takes a look at the architecture for naming relative to EJB lookups, from both the J2EE perspective and the WebSphere Application Server perspective.



J2EE defines a programming model for accessing EJB Homes within a J2EE application. It introduces a level of indirection between the code that will use an EJB Home and the association to an actual instance of an EJB Home that will be used. The programming model is to code a JNDI lookup with a name in the form of “java:comp/env/<ejbname>”. The “<ejbname>” identifies an EJB Reference that is contained in the J2EE module’s deployment descriptor and identifies the interface that must be supported by the EJB Home. Within the same application, there must be an EJB module that contains an implementation of an EJB Home that supports that interface. Therefore, the association between the code using the EJB Home and the instance of the EJB Home is made during the deployment of the application.

Application Server V6 extends this model to be more flexible and to allow for later binding of the client code to the EJB Home instance. First of all, the V6 Server binds all EJB Homes into a global name space. Then, the EJB Reference in the deployment descriptor has been extended to allow for the specification of a JNDI in the Reference. At runtime, the EJB Home bound at this name in the global name space is looked up. With this model, the target EJB Home does not need to be in the same application as the client. The JNDI name of the target EJB Home needs to be known at the time the using application is deployed or when it is installed. The actual association to the EJB Home instance occurs at runtime when the lookup is done.



This slide illustrates what was just described on the previous slide. The client code does a lookup using the name “java:comp/env/ejb/Hello” which associates it with an EJB Reference named “ejb/Hello”. The EJB Reference contains the JNDI name “ejb/com/ibm/ne/HelloHome” which is used to lookup the EJB Home from the global name space.

Section

Names Specified and EJB Lookups
When, Where, What, and Why

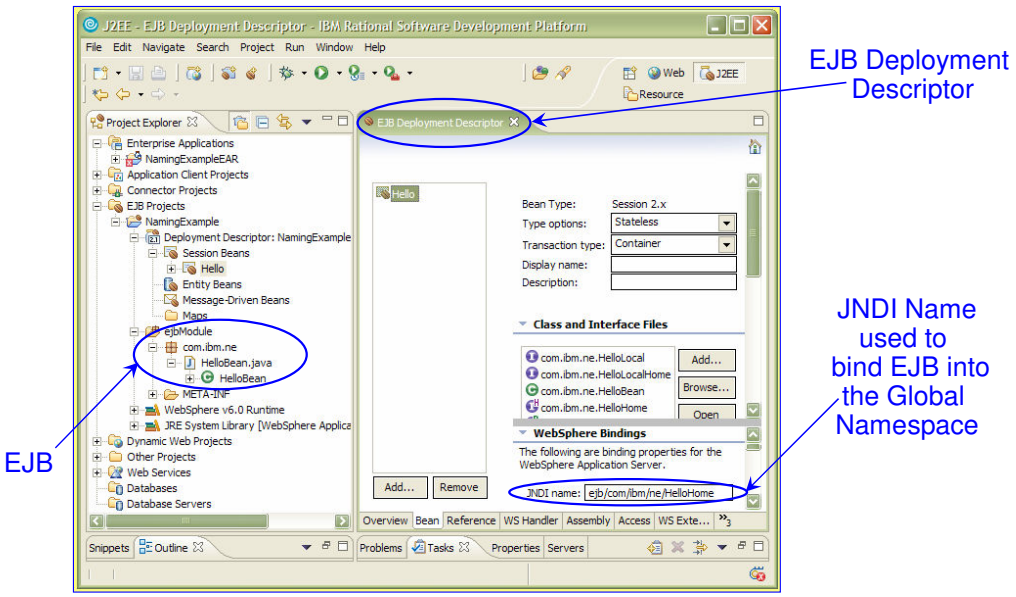
This section will walk through an end-to-end scenario examining every place that JNDI names come into play for EJB Home lookups.

What is Coming Next

- Examine EJBs and EJB Lookups
  - End-to-end considerations
    - Development
    - Deployment
    - Installation
    - Runtime
  - Single server perspective
    - Client and EJB are both in the same server (client could be a Servlet or another EJB)
    - Simplest environment - totally avoids host/port and fully qualified name issues
- Done by first looking at an EJB, then at an EJB client
  - Development in IBM Rational Application Developer – what screens contain the information
  - Artifacts in the EAR – which XML/XMI files carry the information
  - Application installation – what can be manipulated when installing
  - Runtime – what exists in a running server
- What happens in a runtime EJB lookup

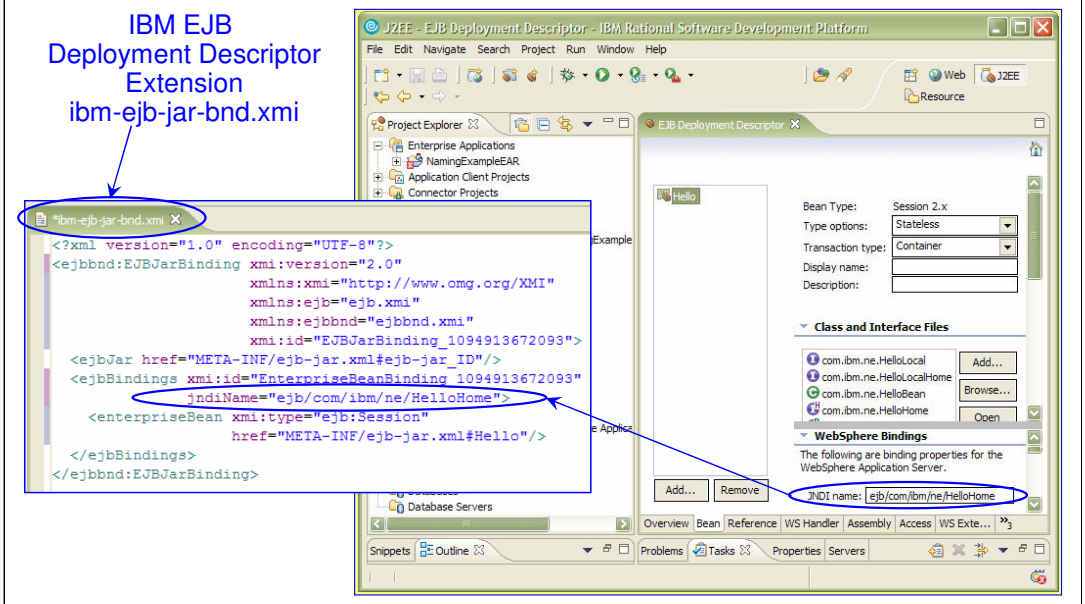
The following slides show screen captures from the IBM Rational Application Developer tool and from the WebSphere Administrative Console showing the use of JNDI names through the full lifecycle of an EJB. First the development, deployment, and installation of the target EJB is considered. Then the development, deployment, and installation of the client-side code is examined, followed by a look at both client and target at runtime. To keep this as simple as possible, it is done from a stand-alone server perspective, with the client of the EJB running in the same server (for example, a servlet or another EJB).

EJBs in Rational Application Developer

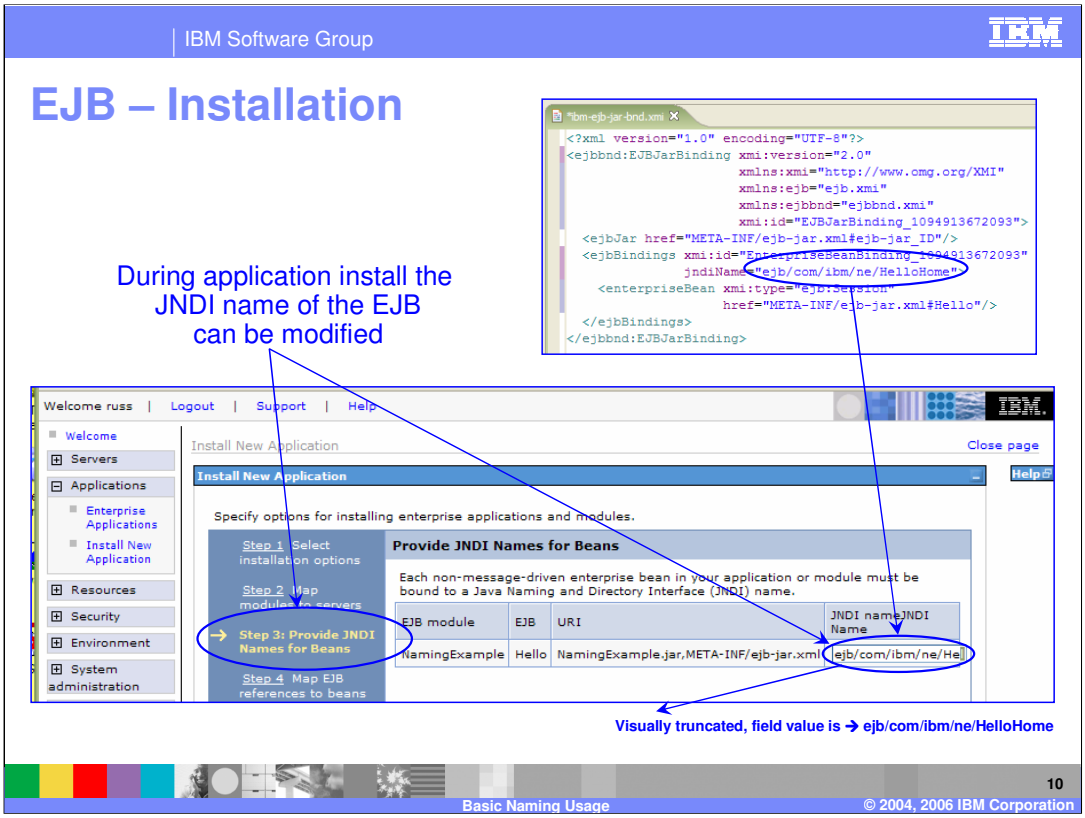


Here is a screen capture from IBM Rational Application Developer showing the EJB Deployment Descriptor for the “Hello” EJB. The JNDI name in the Deployment Descriptor will be the name by which the Hello EJB Home is bound into the global name space.

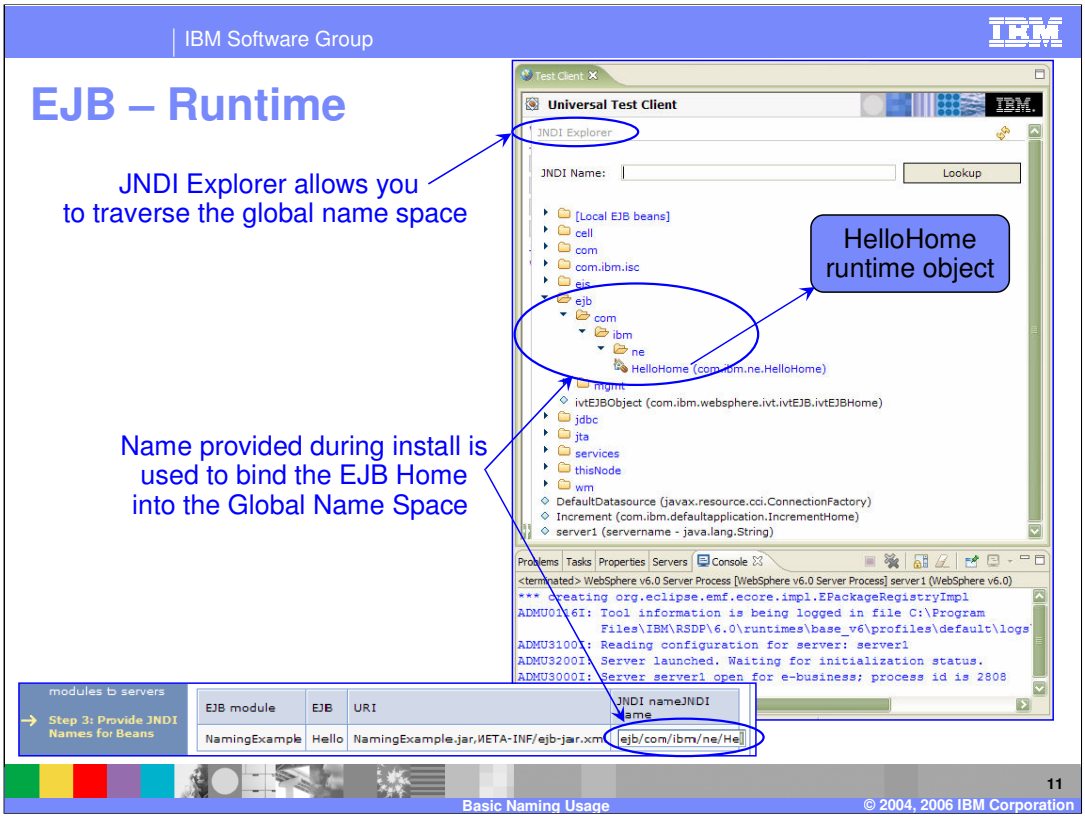
EJB – EAR Artifacts



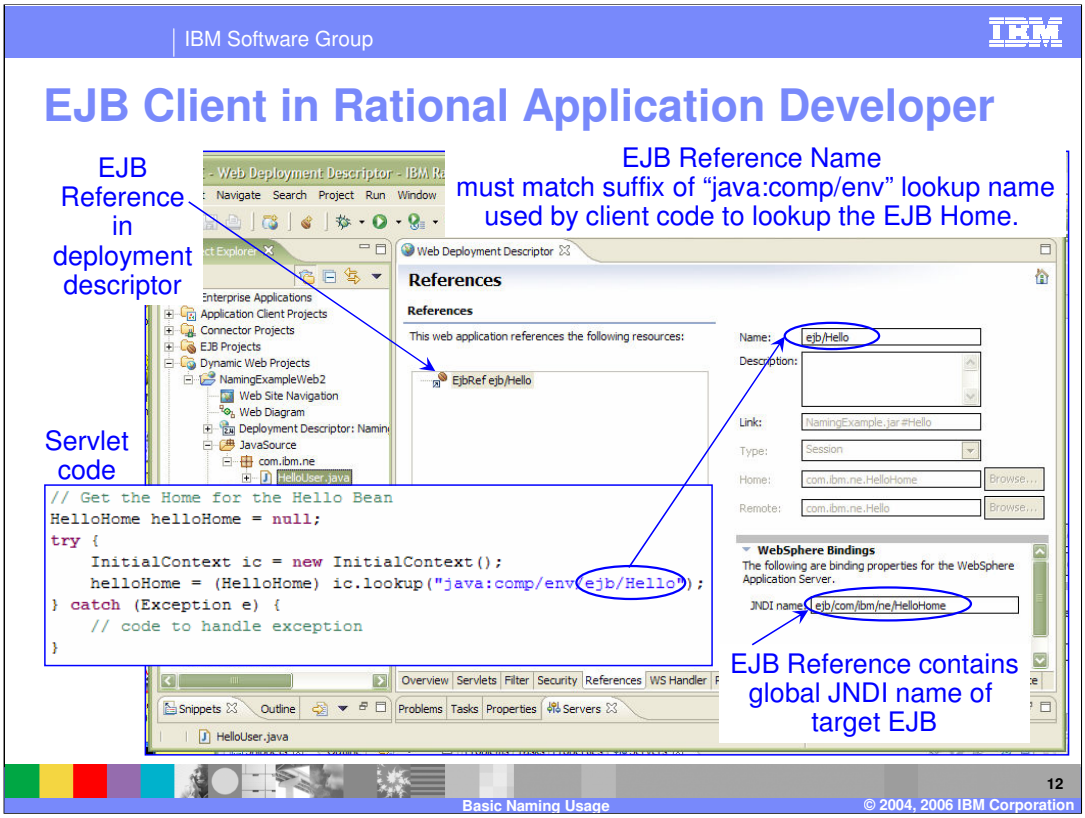
This screen capture shows that the JNDI name for the Hello EJB specified on the previous slide will be put into the ibm-ejb-jar-bnd.xmi file which will be part of the .jar file for the EJB Module.



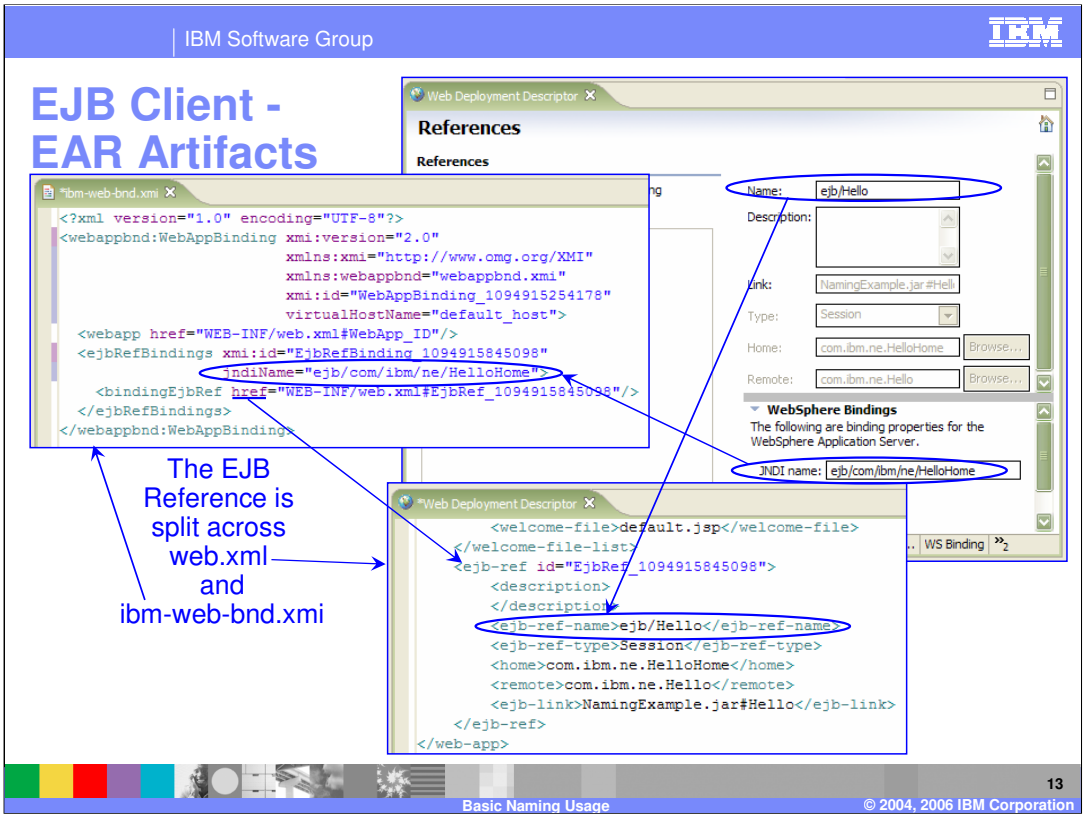
This slide shows the Administrative Console during the installation of the application containing the Hello EJB. There is an installation step “Provide JNDI Names for Beans” which displays the JNDI name found in the ibm-ejb-jar-bnd.xmi file. At this time you have the ability to accept the specified JNDI name for the target EJB, or if needed, you can change the name.



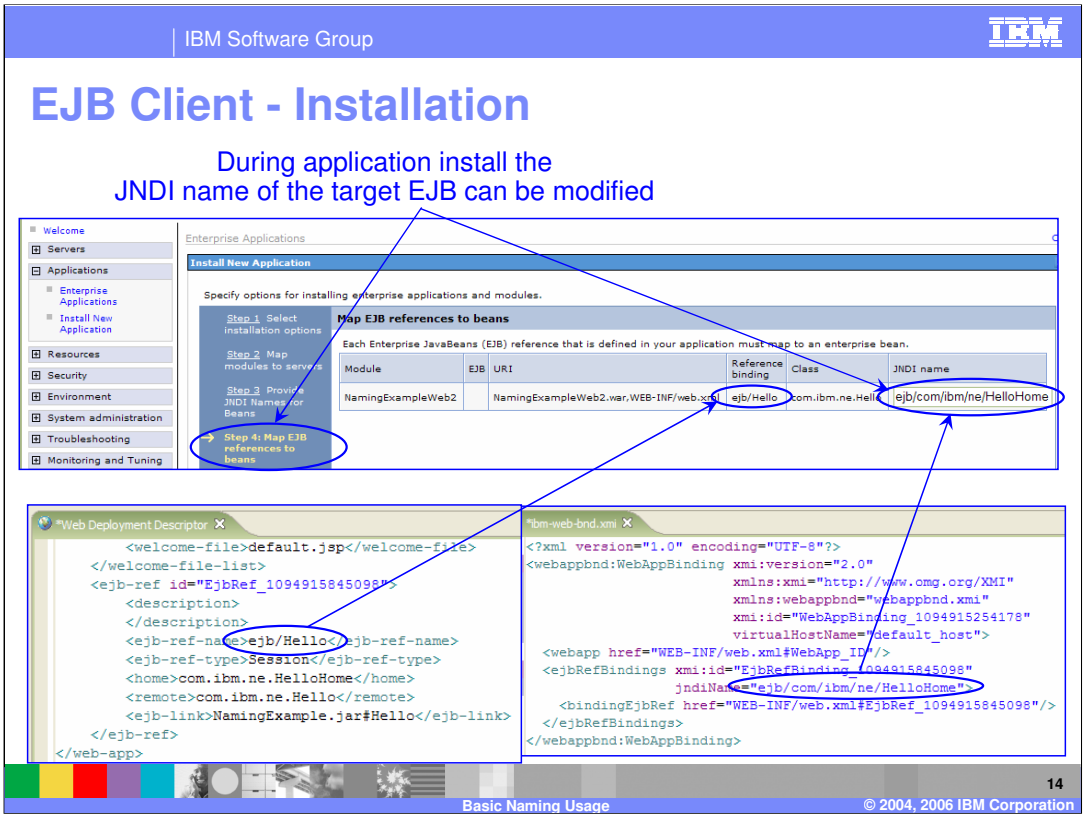
The JNDI Explorer, which is part of the IBM Rational Application Developer test client, is displayed here to show that at runtime the JNDI name of the Hello EJB specified during application install is used to bind the Hello EJB Home into the global name space.



This slide begins the story for the client code that will use the Hello EJB. There is code in the servlet which uses the name "java:comp/env/ejb/Hello" to look up the target EJB. In the Web Deployment Descriptor there is an EJB Reference named "ejb/Hello" which matches the suffix of the name used in the code. The EJB Reference also contains the global JNDI name of the target EJB. Note that it is not required to specify the global JNDI name at this point in the cycle if it is not known by the developer.



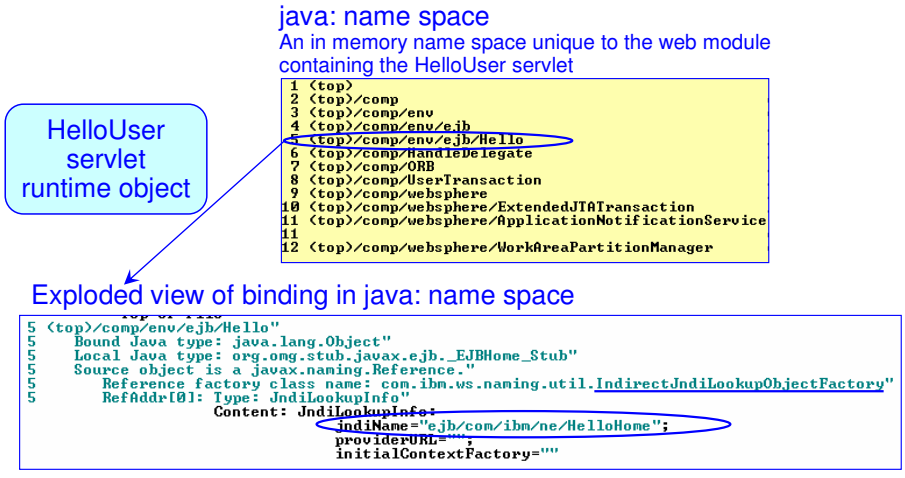
This slide shows where the information specified in the Web Deployment Descriptor is reflected in the artifacts built into the .war file. The information is split across two files, the first being the standard J2EE Web Deployment Descriptor (in web.xml) and the remainder in the ibm-web-bnd.xmi file. It is this latter file that contains the global JNDI name of the target EJB as this capability is a V6 Server extension.



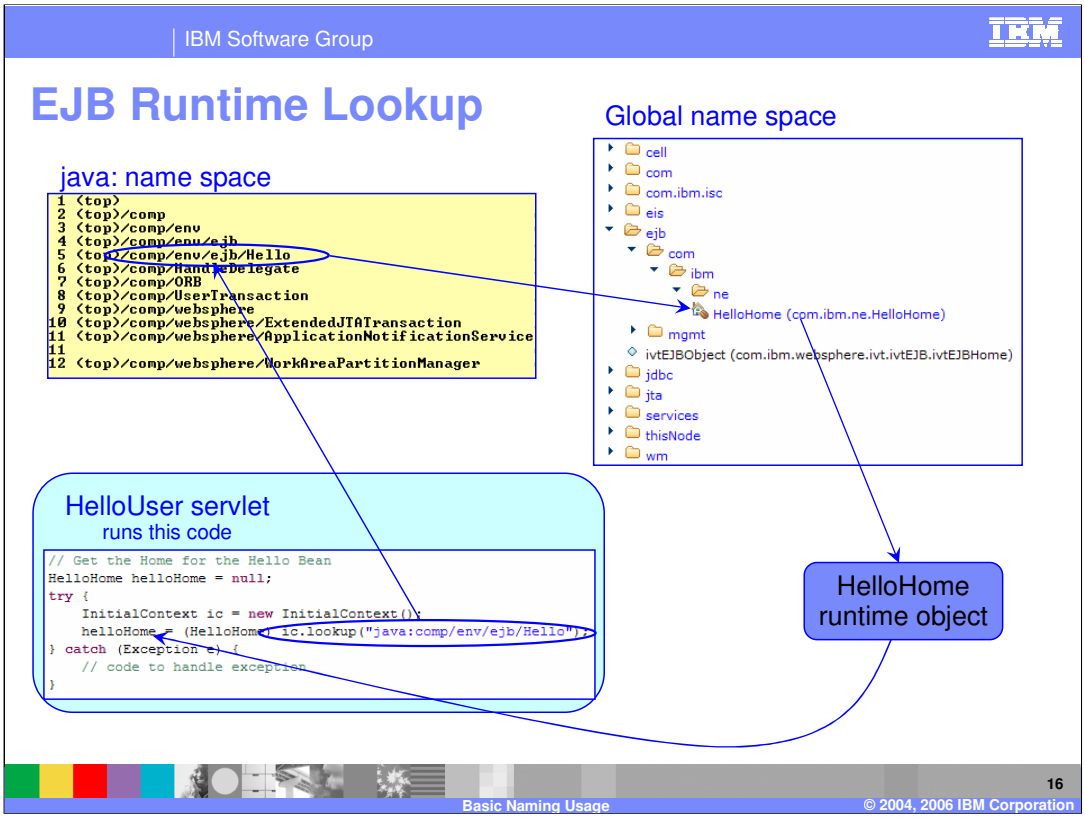
This slide shows the Administrative Console during the installation of the application which will use the Hello EJB. There is an installation step “Map EJB references to beans” which displays the name of the EJB Reference (“ejb/Hello”) and the associated global JNDI name found in the ibm-web-bnd.xmi file. At this time, you have the ability to accept the specified JNDI name for the target EJB, or if needed, you can change the name.

EJB Client - Runtime

- Client runtime consists of the servlet and “java:” name space
- “java:” name space binding is built from the EJB Reference



When the application that is the user of the Hello EJB is running in a server, there is a “java:” name space associated with the Web Module. In that name space is a binding for “comp/env/ejb/Hello”. An expanded view of the contents of that binding shows that what is bound is an IndirectJndiLookupObjectFactory along with the global JNDI name of the target EJB, “ejb/com/ibm/ne/HelloHome”. When this binding is looked up from the “java:” name space, the IndirectJndiLookupObjectFactory performs the look up of the global JNDI name and returns the result of that call.

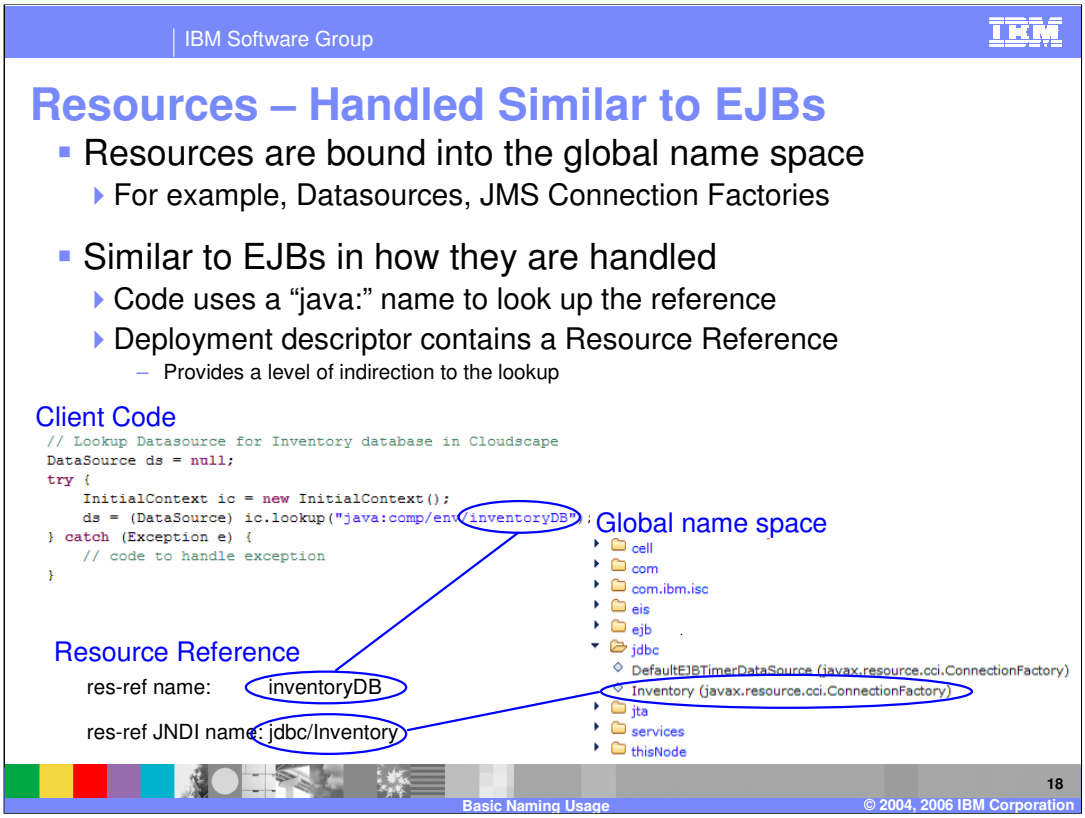


In the runtime, when the lookup is done in the code, it accesses the binding in the “java:” name space, which then accesses the global name space and returns the HelloHome runtime object.

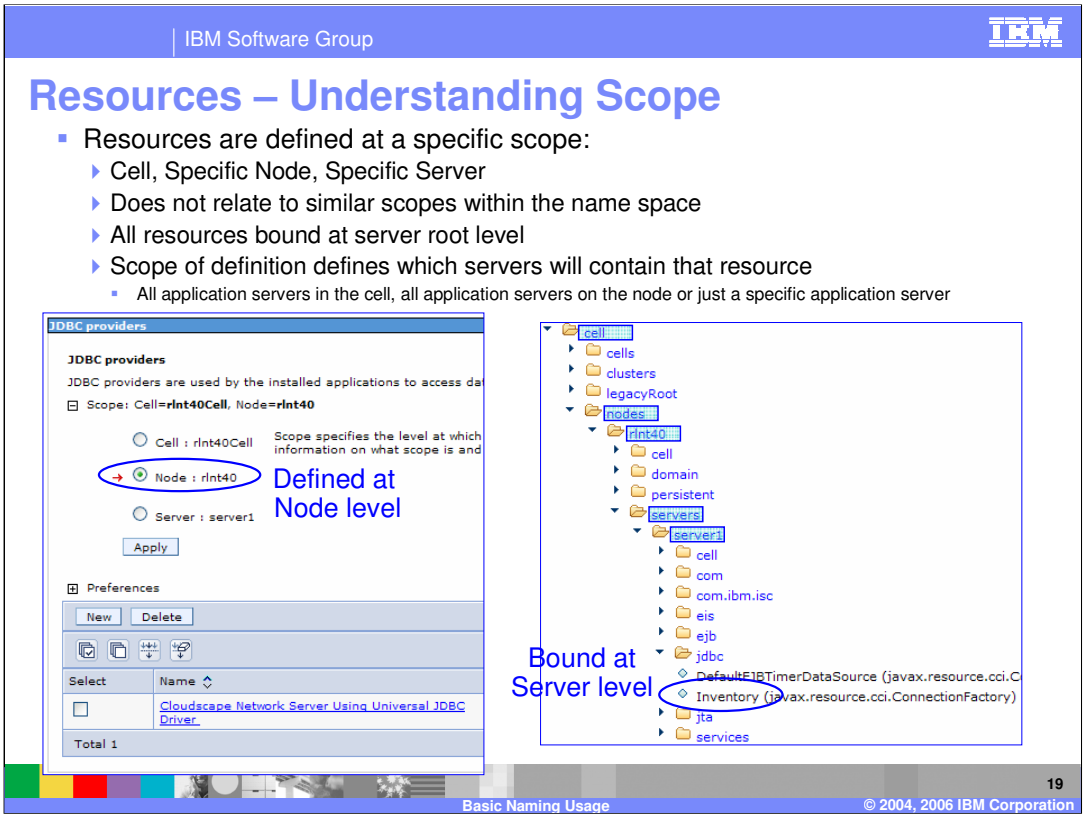
Section

Handling of Resources

Now that you understand how EJBs work with Naming, this section will address the handling of Resources.



The handling of Resources, such as JMS Connection Factories, is very similar to how EJBs are handled. The Resource is looked up using a “java:comp/env/<nameOfResource>” name, which gets associated with a Resource Reference from the Deployment Descriptor. That Resource Reference contains the JNDI name for where the target resource is bound into the global name space.

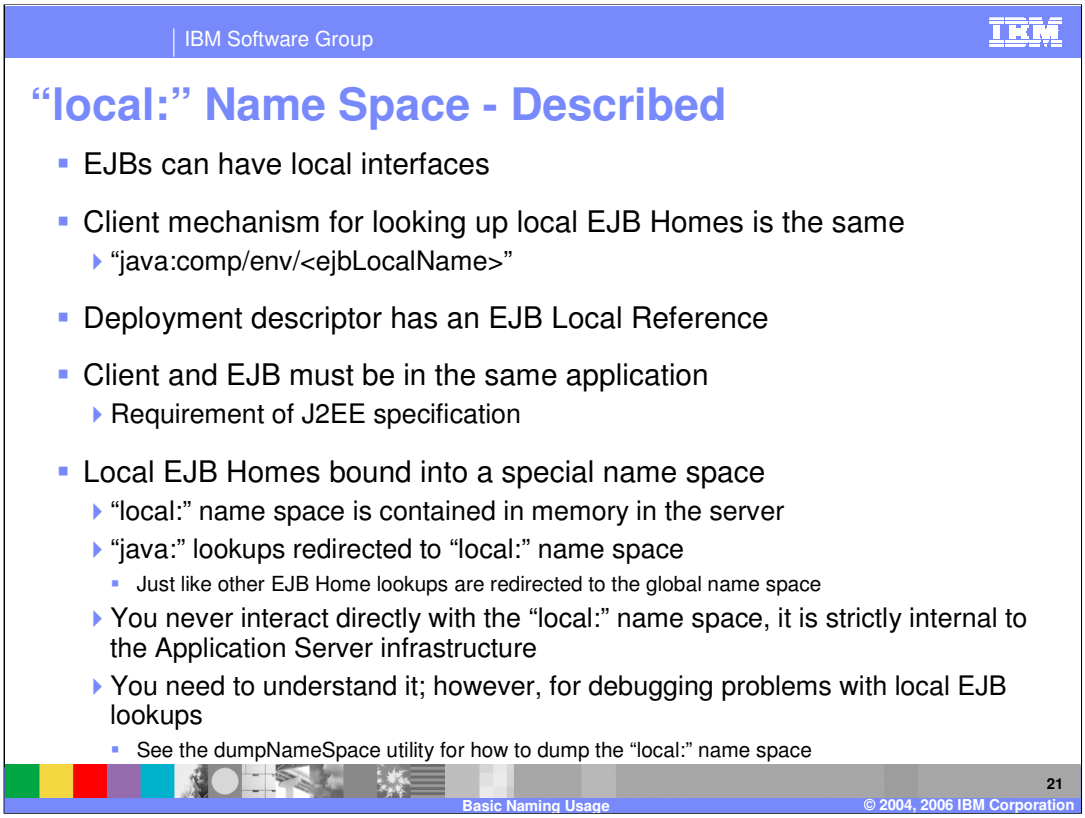


When a Resource is created by an Administrator, there must be a JNDI name associated with the Resource which is used to bind it into the global name space. During server startup, the Resources defined for that server are bound into the global name space. The purpose of this slide is to make sure there isn't confusion about what "scope" means when defining resources. A resource can be defined with a scope of cell, node or server. The structure of the global name space also has unique naming contexts at the cell, node and server level. Although having the same levels of scoping, these scopes do not relate to each other with respect to where the resource gets bound into the name space. Resources are always bound into the name space at the server level, in the server root context. Defining a resource at the node level causes that resource to be bound into the server root context for all servers in that node. Likewise, defining a resource at the cell level results in the resource being bound into the server root context for all servers in the cell.

Section

Handling of Local EJBs

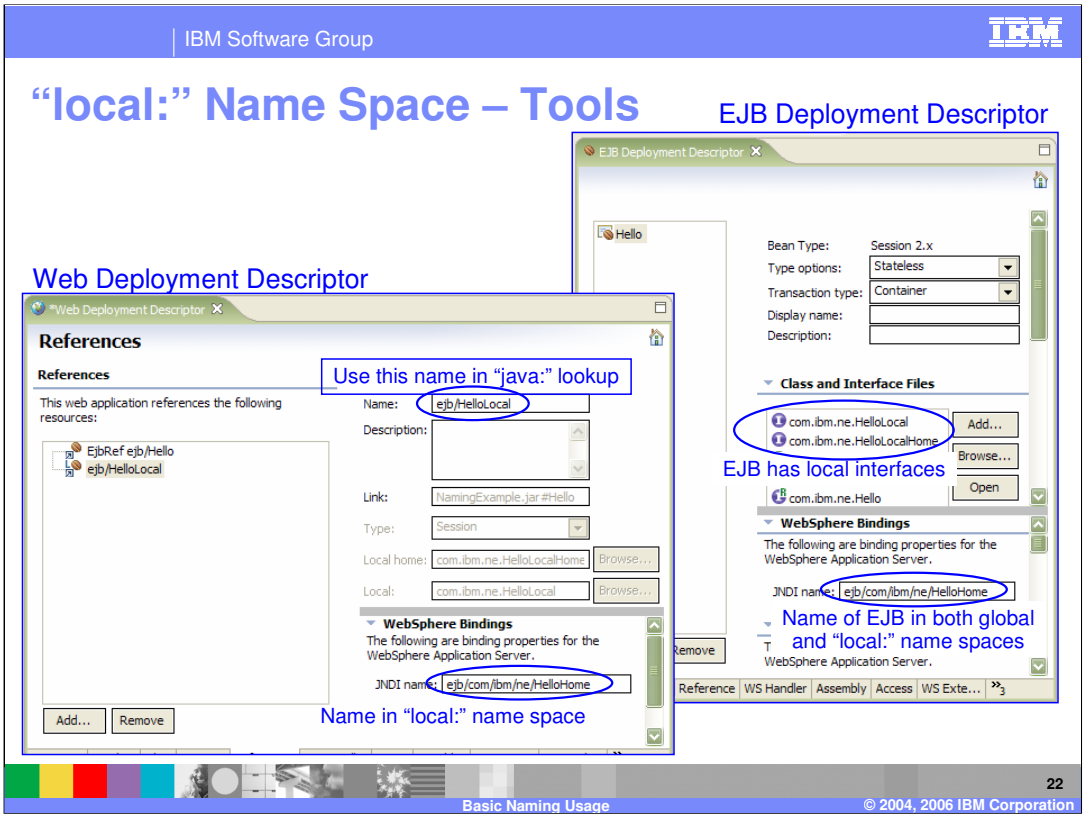
The next few slides take a look at how local interface EJBs are handled, contrasting them to remote EJBs which were previously discussed.



EJBs can have local interfaces and remote interfaces. From a programming model perspective, they work the same. The client does a lookup of a "java:comp/env/<ejbLocalName>" name which is then associated with an EJB Local Reference in the Deployment Descriptor. The local interface EJBs are not, however, bound into the global name space. Rather, each server has an internal in-memory "local:" name space used for binding local EJBs. On other respects, the runtime operations are the same in that the "java:" lookup gets redirected to the "local:" name space to find the target local EJB Home.

Local EJBs must be deployed in the same application as the client of the local EJB. Even if deployed into the same server, they cannot be in different applications.

The "local:" name space is really an internal implementation detail that you do not necessarily need to know about. However, it is good to know about it when debugging problems related to looking up local EJB Homes. The "local:" name space can be dumped using the dumpNameSpace utility which is described in the debugging presentation.



This slide shows the IBM Rational Application Developer screens where names for local EJBs are specified. The Web Deployment Descriptor for the client side defines the JNDI name to be looked up in the "local:" name space. On the EJB Deployment Descriptor, the name of the local EJB will be the same name as is used for the remote EJB in the global name space.

Summary

- Provide basic understanding of how to use Naming
  - stand-alone server environment
  - Based on usage with EJBs
- Looked at full lifecycle
  - Development, Deployment, Installation and Runtime
- Addressed from developers and administrators perspective
  - Where are names specified
  - Define relationship between different names
- Naming used with Resources
- Local EJB are handling

This presentation reviewed the basics of naming as used with remote EJBs in the stand-alone server environment. This was done by looking at the full lifecycle for both the EJB and the client of the EJB, using screen captures of IBM Rational Application Developer and the WebSphere Application Server Administrative Console to show where JNDI names are specified and how they are handled. Following this, the use of naming with Resources and local EJBs was addressed.

Template Revision: 11/02/2004 5:50 PM

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.