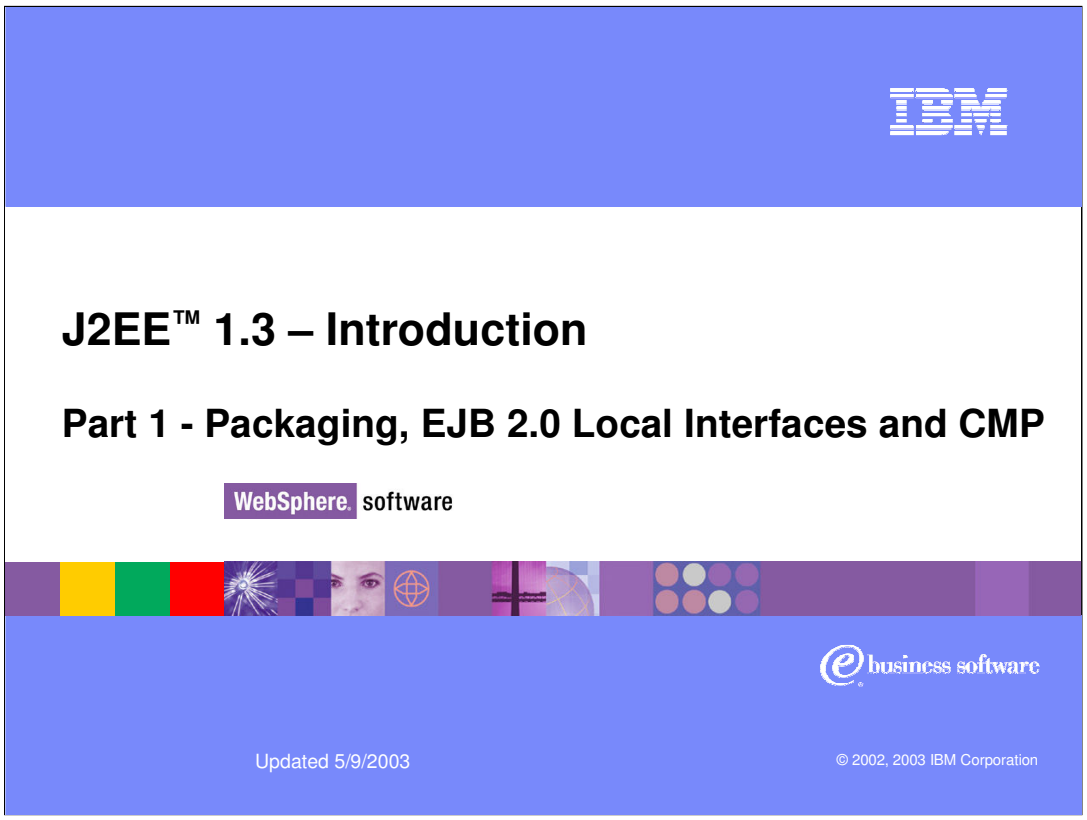
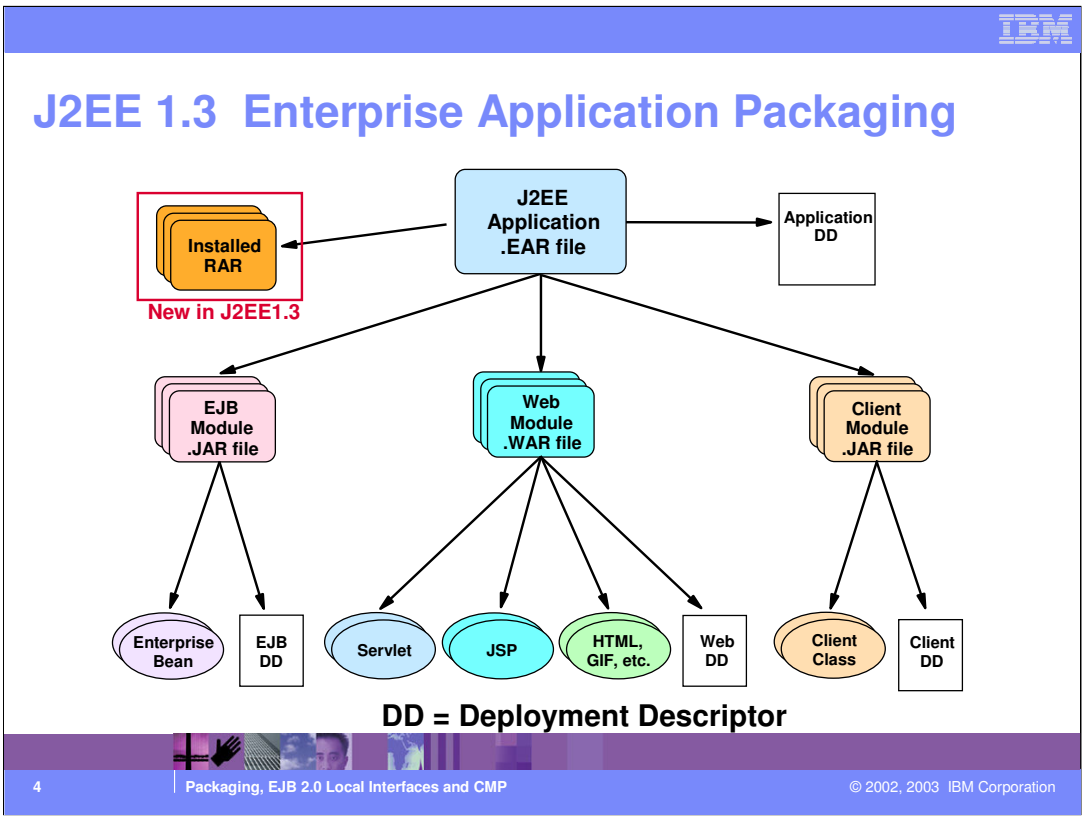

IBM
J2EE™ 1.3 – Introduction
Part 1 - Packaging, EJB 2.0 Local Interfaces and CMP
WebSphere. software
e business software
Updated 5/9/2003
© 2002, 2003 IBM Corporation

J2EE 1.3 Topics

- **J2EE 1.3 Packaging**
- **EJB 2.0:**
  - **EJB 2.0 Interoperability**
  - **New type of Interface: Local Interfaces**
  - **New Persistence Manager to handle Container-Managed Persistence and Relationships**
  - EJB Query Language (EJB QL)
  - New type of bean: Message-driven Bean
  - EJB Home Methods
  - Dependent Values
- **Value-Add Features beyond EJB 2.0 specification**
- **J2EE 1.3 aspects of Web Components**
  - Servlets 2.3
  - HTTP Session Topics
  - JSP 1.2

J2EE 1.3 Packaging and EJB Interoperability



A J2EE application is packaged in an Enterprise Archive, a file with a .EAR extension.

The application has a Deployment Descriptor, shown here as DD, allowing configuration to a specific container's environment when deployed.

The application can include one or more modules.

J2EE components are grouped in modules, and each module has its own deployment descriptor.

EJB modules group related EJBs in a single module, and are packaged in Java Archive (JAR) files.

Note that there is only one deployment descriptor for all of the EJBs in the module. Previously, in WebSphere® 3.5, each Enterprise bean had its own deployment descriptor.

Web modules group Servlet class files, JSPs, HTML files and images.

They are packaged in Web Application Archive (WAR) files.

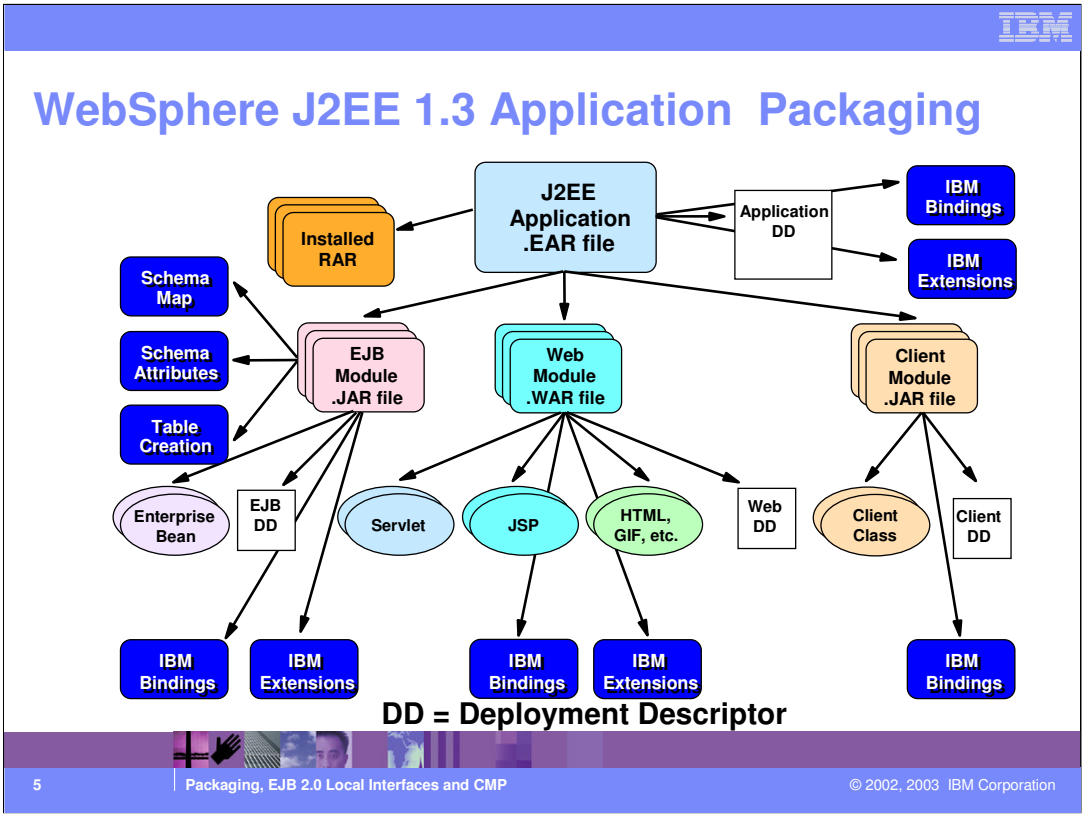
Application client modules are packaged in Java Archive (JAR) files.

New to J2EE 1.3 is the packaging of Resource Adapters (RAR) as part of the EAR file.

RARs only being used by these applications can now be packaged with the EAR file.

In J2EE 1.2, the developer would have to send the EAR file and RAR separate.

Will still need to install the RAR on the Node.



This page shows the J2EE Application EAR file enhanced with the IBM Bindings and Extensions.

These adapt the generic J2EE application to the IBM WebSphere 4.0 Application Server environment.

Most J2EE server vendors have their own proprietary extensions to the specification.

The schema map, attributes, and table creation code provide the setup for entity Enterprise beans to store their values in a database.

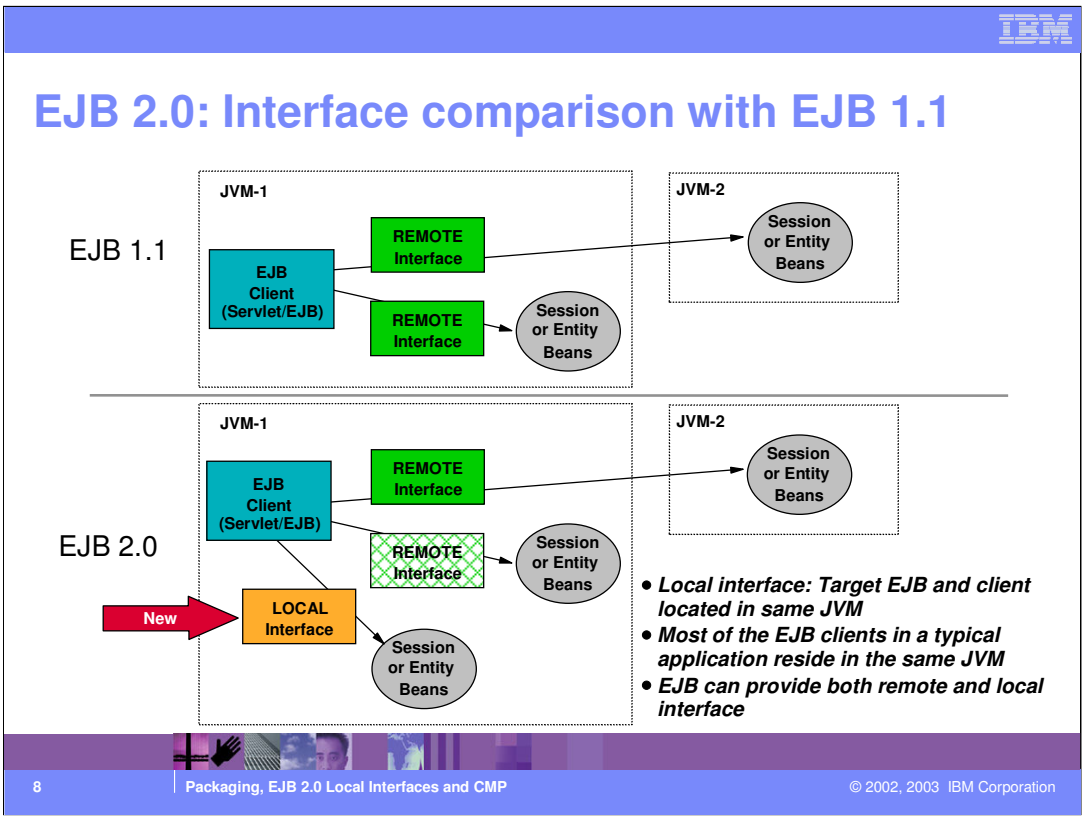
These are for entity beans that use Container Managed Persistence, or CMP.

All of these objects in the Application EAR file are packaged by the Application Assembly Tool.

EJB 2.0 Multi-vendor Interoperability

- **EJB 2.0 supports interoperability among EJB containers from different vendors**
- **Areas of interoperability addressed by the specs:**
  - Remote method invocation
    - Data type mapping
    - HandleDelegate - allows portable serialization/deserialization of EJBObject and EJBHome
  - Transaction Interop
  - Naming
    - CORBA CosNaming APIs support
  - Security
    - CSIv2

EJB 2.0 Local Interface



EJB client for the remote interface can be:

•Java client

•EJB or Servlet within the same or different JVM (App Server).

EJB client for Local interface can only be EJB or Servlet within the same JVM (App Server).

In a typical application, about 60 to 80 % of the calls to an EJB are from clients that are local within the same App Server.

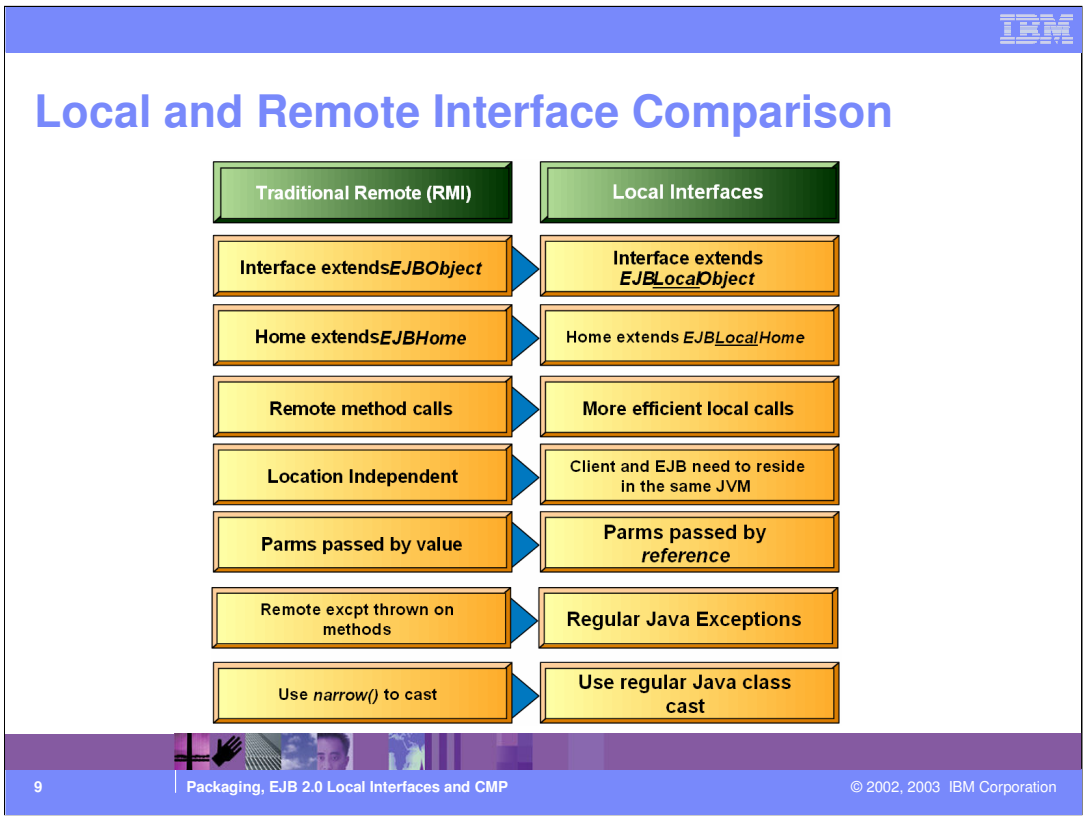
Local Interfaces are the foundation for New Container Managed Relations (CMR) among entity beans.

Interfaces are defined in Deployment Descriptor.

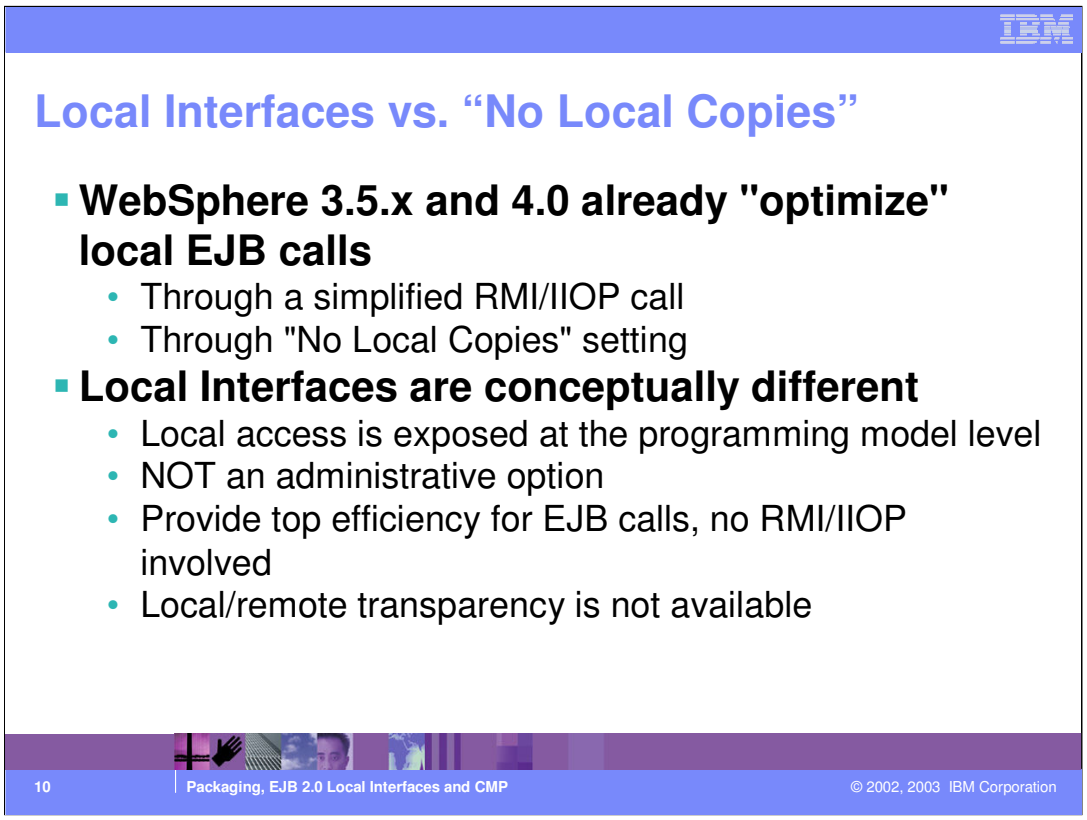
Clients specify EJB Local Reference or EJB Reference (remote) as needed.

Advantages:

•Better Performance since there is no overhead of remote method call, marshaling, de-marshalling, etc.

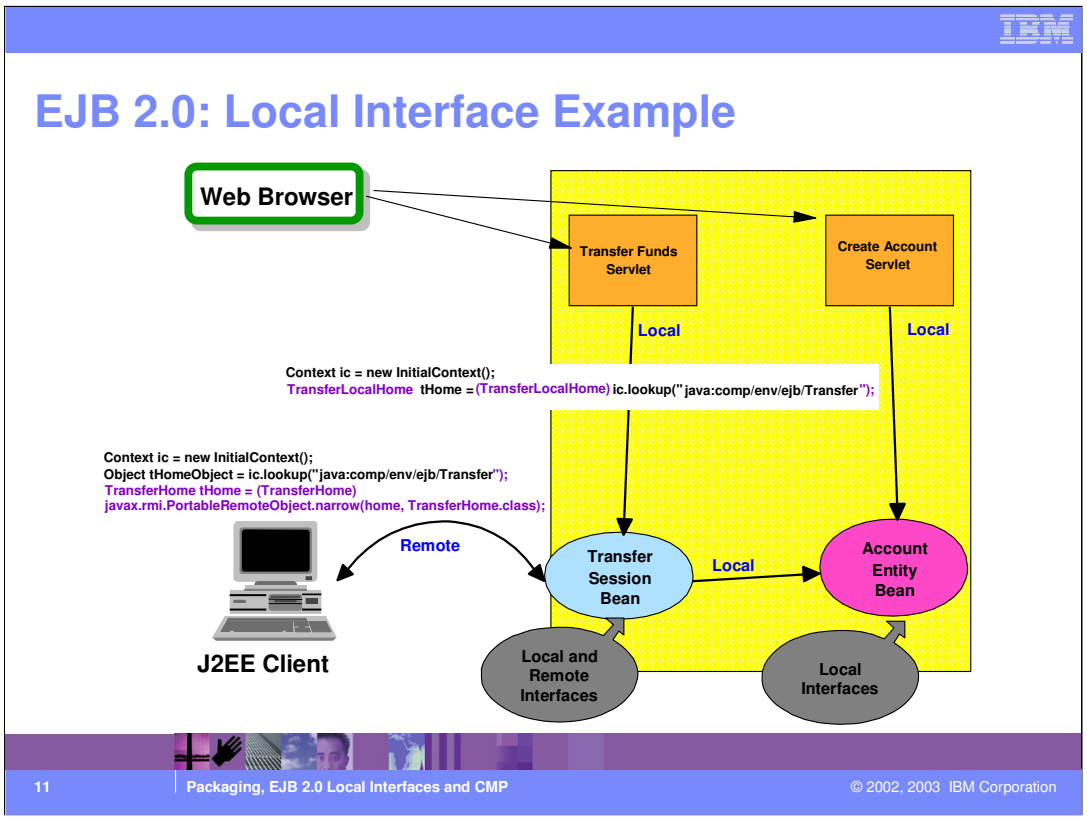


This chart summarizes the differences between local and remote interfaces. It's important to notice that, while an EJB may offer both local and remote interfaces, the choice of using local or remote in the EJB client is NOT transparent. The client code will clearly reflect which interface is going to be used.



It's also interesting to notice that WebSphere 3.5.x and 4.x offers performance optimizations that are similar to the those that Local Interfaces bring about (whenever the EJB client and the EJB coexist in the same JVM, WebSphere App Server would optimize the call path and use a simplified RMI/IIOP stack, leading to better performance - also you could turn on "Local Copies" and have parameters passed by copy in an app server). Although they are conducive to similar performance advantages, Local Interfaces are technically very different from the performance optimizations provided natively by WebSphere.

With Local Interfaces, there is no RMI/IIOP being invoked. The choice of using Local vs. Remote interfaces is an application design one, not an administrative option.



•Here is an example of using Local and Remote interfaces.

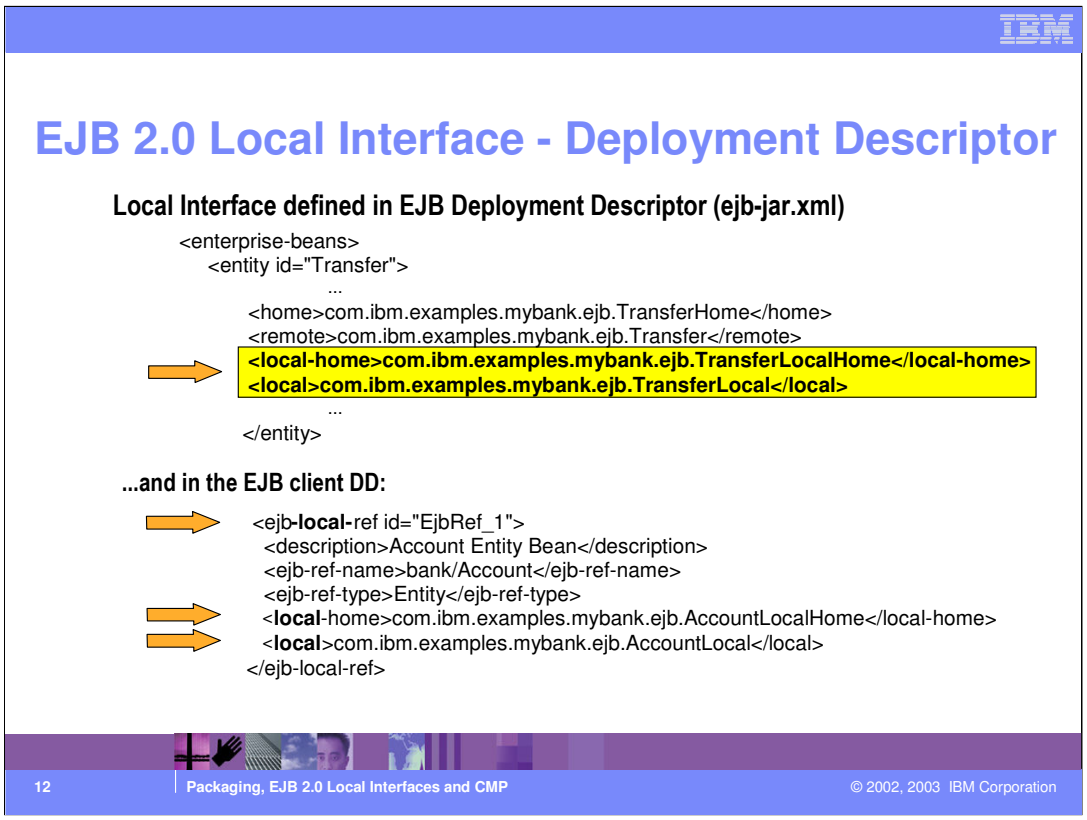
•Notice that the Transfer bean offers both local and remote interfaces, while the Account bean offers Local interfaces only.

•The servlets, since they can coexist in the same app server as the EJBs, can choose whether to use local or remote interfaces. In this case they are using local interfaces - but that's not a general design recommendation as there may be situations where you would choose otherwise.

•The J2EE client can ONLY use remote interface since it will never run in the same JVM as the EJBs.

•The bottom line is that while Local Interfaces are conducive to better performance and to a more granular Entity EJB object model, they also "tie the hands" of the deployer, in the sense that the EJB clients have to be installed on the same app server as the EJB they talk to.

•The chart also shows the different coding style and the obvious differences in the casting mechanism.



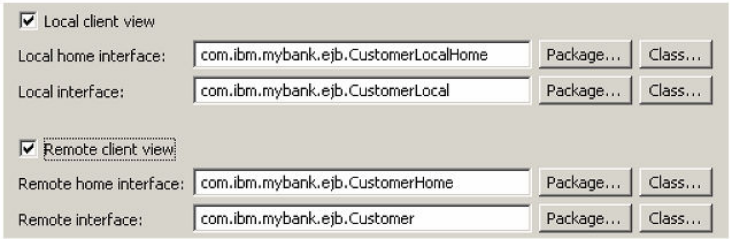
•The bean provider has to make it clear to the "outside world" whether a bean is offering local, remote, or both interfaces. The DD carries new elements <local-home> and <local> to signify the presence of local interfaces.

•On the other hand, the EJB client developer has to specify the use of local or remote interfaces in the deployment descriptor. This indication will allow the application deployer to install the EJBs and their clients correctly (i.e. in the same JVM if the client uses local interfaces).

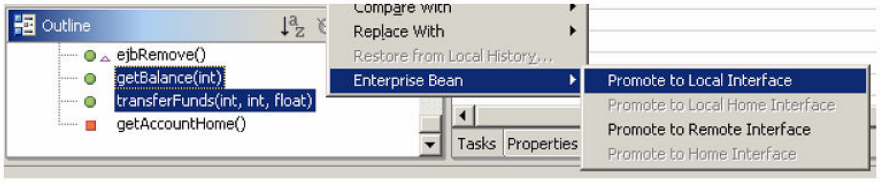
•That's why we have a new <ejb-local-ref> element in the deployment descriptors of Web modules and EJB modules (in addition to the existing <ejb-ref> which designates the intention to use remote interfaces).

Tooling Support for Local/Remote Interfaces

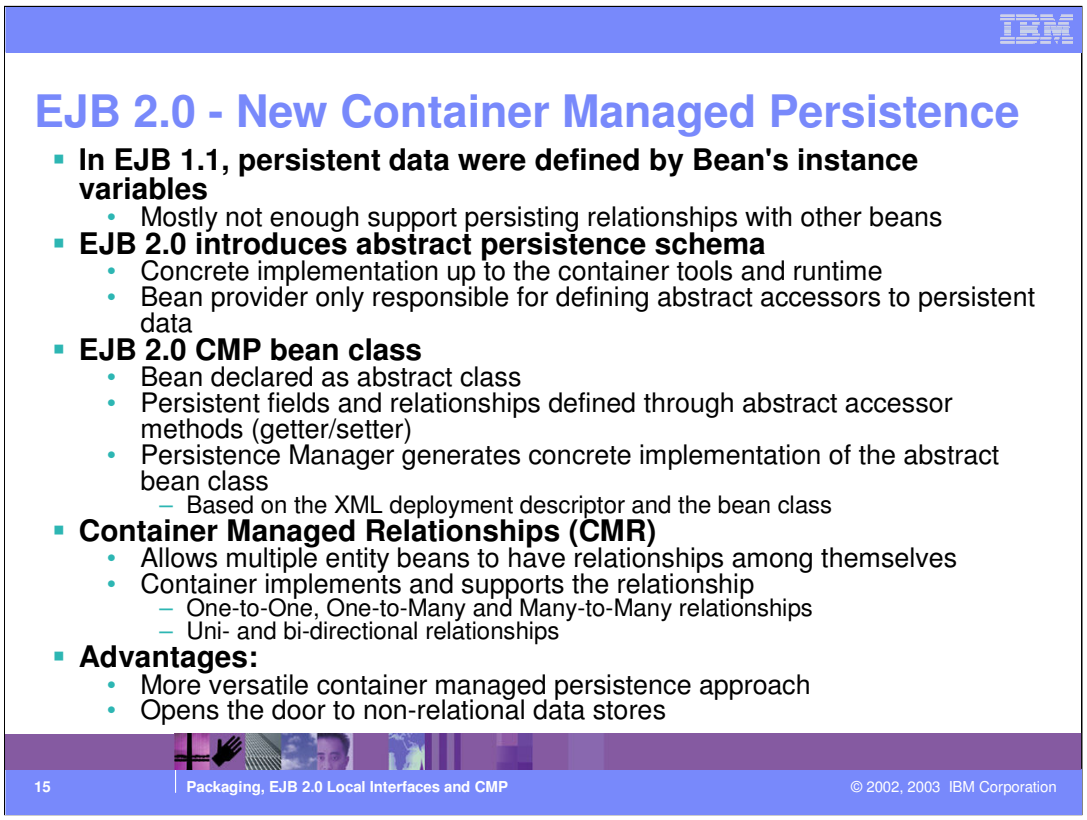
Local and/or Remote Interfaces specified when enterprise bean created with WebSphere Studio Application Developer



Promote methods to Local or Remote existing interfaces

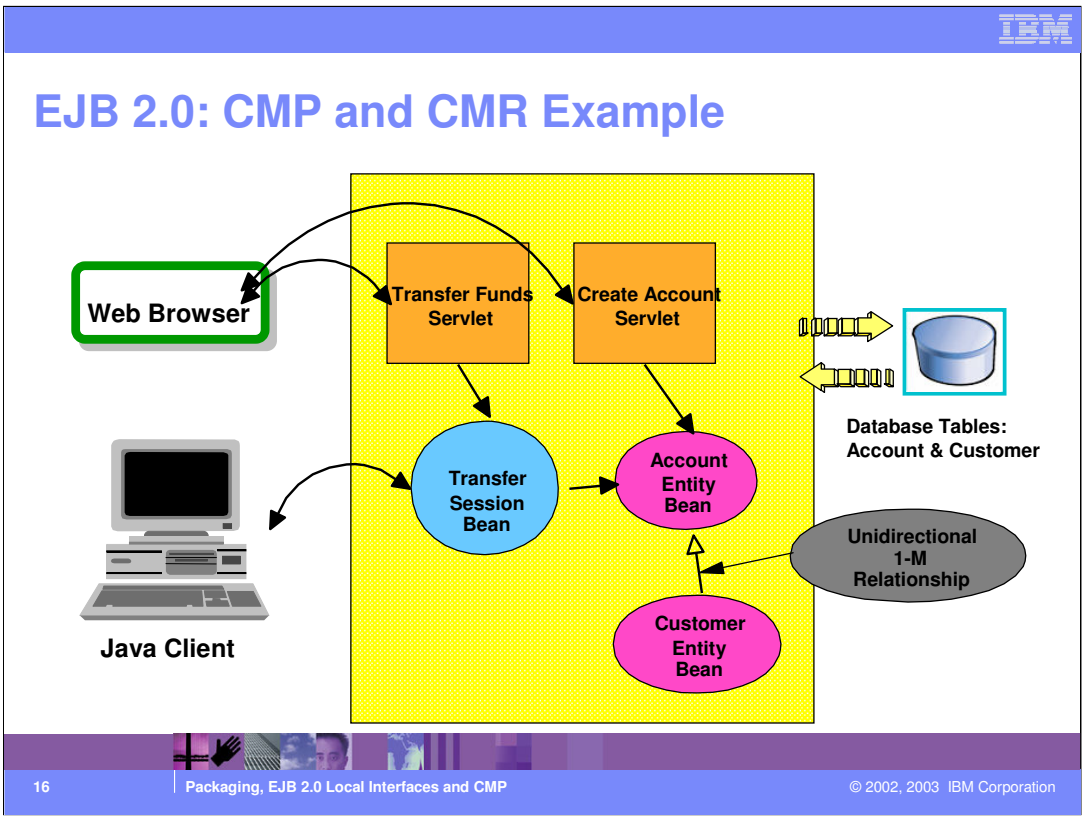


EJB 2.0 Container-Managed Persistence Support

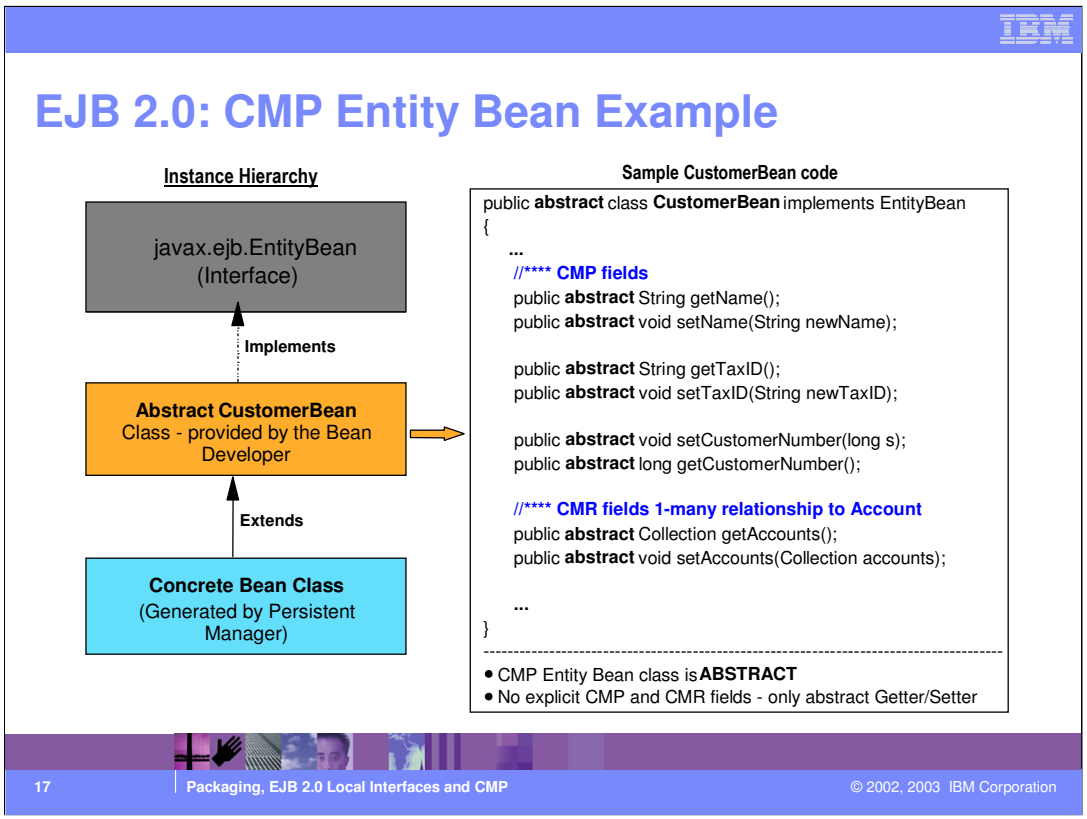


•In EJB 2.0, it's up to the container to implement the concrete persistence model. The bean developer only defines the accessor abstract methods.

•This new approach offers very powerful perspectives on persistence - like the total independence of CMP beans from the underlying persistence store, which may not even be implemented by a relational db.



This chart shows a simple unidirectional relationship: the customer EJB "knows" its accounts, but not vice-versa (the account doesn't provide any behavior to go back to the customer).



•Here is an example of CMP 2.0 EJBs. Notice the abstract nature of the EJB and of the accessor methods.

•It's up to the container tooling and runtime to provide for a suitable implementation of those abstract methods and ultimately for a concrete bean class that can be instantiated.

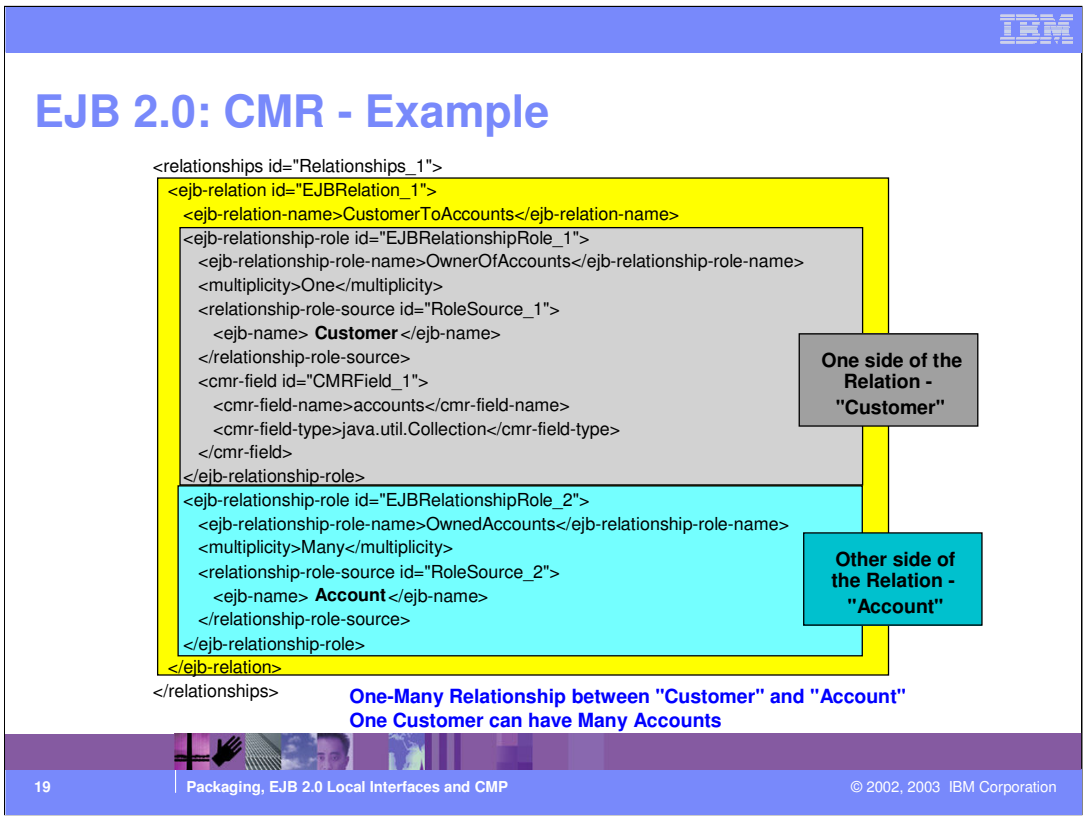
EJB 2.0 CMP and CMR - Deployment Descriptor

- **CMP/CMR fields defined in EJB Deployment Descriptor (ejb-jar.xml)**

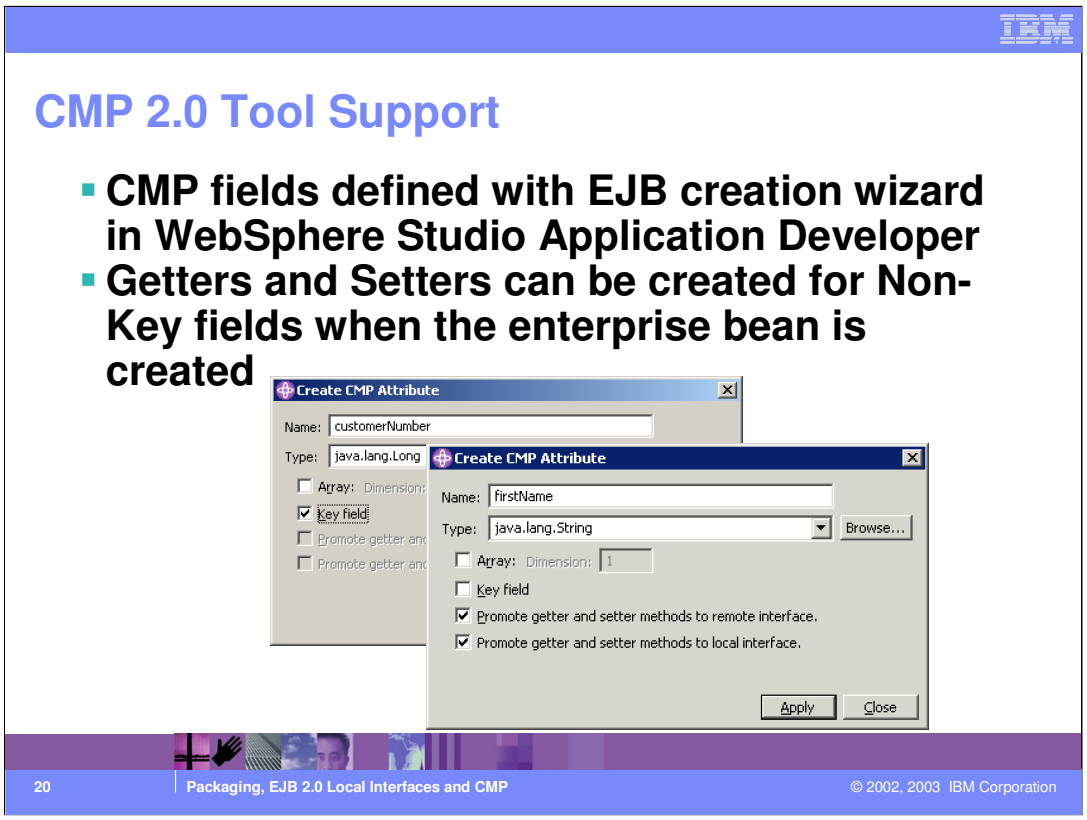
```
<enterprise-beans>
   <entity id="Customer">
      <ejb-name>Customer</ejb-name>
      ...
      <abstract-schema-name>Customer</abstract-schema-name>
      <cmp-field id="Customer_number">
        <field-name>customerNumber</field-name>
      </cmp-field>
      <cmp-field id="Customer_name">
        <field-name>name</field-name>
      </cmp-field>
      <cmp-field id="Customer_taxID">
        <field-name>taxID</field-name>
      </cmp-field>
      ...
   </entity>
   ...
</enterprise-beans>
```

CMP fields

•From a Depl Descr. standpoint, not much has changed - you still declare CMP fields as you used to do, this time by extracting the field name from the "getter and setter" accessor methods.



•A completely new element in the DD (<relationships>) defines EJB relationships. A relationship can be unidirectional (like in the example) or bi-directional: in the latter case, you would have to define two <cmr-field> elements (one for the source and one for the target - notice that we only have one <cmr-field> in our example).



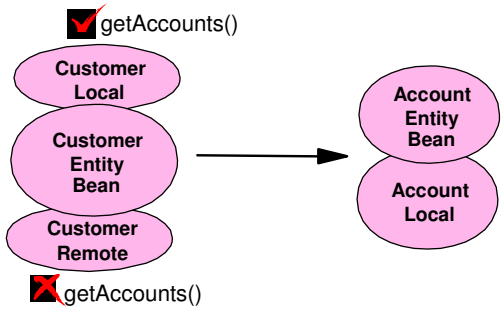
•WebSphere Studio Application Developer v5 fully supports the creation of 2.0 EJBs.

•When creating CMP entity beans with Application Developer, you can specify the Name of the field, the type, and whether getters and setters should be promoted to the remote or local interfaces.

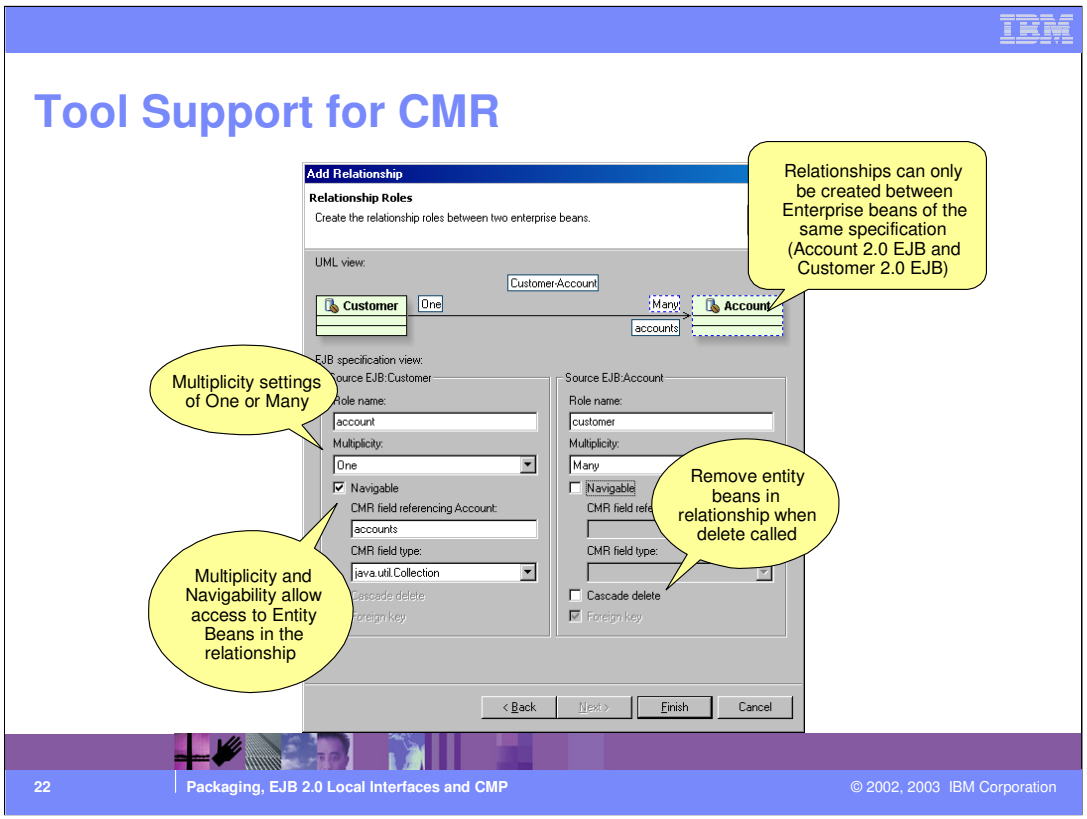
•When creating the attributes, it is recommended that you use a type of the java.lang package. This will make each attribute an object and offers more options when dealing with collections of attributes.

CMR and Local Interfaces

- **"Target" EJB in a relationship must provide Local Interfaces**
  - In bi-directional relationships, both EJBs must have Local Interfaces
- **CMR accessor methods deal with Local Interfaces**
  - Must **not** be exposed on Remote Interfaces
  - Remote clients ca not get directly to related objects
    - Bean provider must add business logic to enable remote accessibility



The requirement of local interfaces for Container-Managed Relationships is required for performance reasons, especially in one-to-many or many-to-many relationships.



•WebSphere Studio Application Developer fully supports EJB 2.0 and allows you to create Container-Managed Relationships between two entity beans. The relationship can be set up from within the EJB Deployment Descriptor editor.

•The two entity beans in a relationship must be created prior to running the wizard.

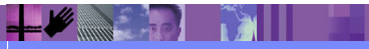
•If an entity bean does not have local interfaces, it will only be able to have unidirectional relationships with itself to other entity beans. Other entity beans will not be able to have a relationship to it.

•For access the other bean in a relationship, it is recommend to make the name of the CMR field plural if the relationship is one-to-many or many-to-many.

•The type of the CMR field can be java.util.Collection or java.util.Set. Depending on how the information in the relationship is to be used will determine whether to use Collection or Set.

Summary

- J2EE 1.3 specification supports adding Resource Adapter (RAR) files to EAR files
- Local Interfaces are a design decision which offers benefits in certain situations
- CMP in EJB 2.0 specification leaves implementation details to the Container and reduces requirement of tight integration with datastore
- CMR allows for interaction between entity beans corresponding to realistic business relationships



Trademarks and Disclaimers

References in this document to IBM products or services do not imply that IBM intends to make them available in every country. The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

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

Lotus, Domino, Freelance Graphics, and Word Pro are trademarks of Lotus Development Corporation and/or IBM Corporation in the United States and/or other countries.

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 trademarks of Microsoft Corporation in the United States, other countries, or both. 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. Other company, product and service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. 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.