

Tutorial: Container Managed Relationships using WebSphere Studio Application Developer 5.0

Tutorial: Container Manager Relationships using WebSphere Studio Application Developer 5.0

[Gary Flood](#)

Advisory IT Specialist, WebSphere Platform Solution Test

IBM Hursley, United Kingdom

March, 2003

© Copyright International Business Machines Corporation 2003. All rights reserved.

About this tutorial:

This tutorial introduces the concept of Container Managed Relationships (CMR) in WebSphere® Studio Application Developer 5.0. CMR specifies logical relationships between entity beans. The tutorial provides steps to develop a working example using WebSphere Studio. In addition, a test session bean is provided to highlight some of the benefits and complexities of using CMR.

Objectives:

Upon completion of this tutorial, you should be able to:

1. Understand the concept of Container Managed Relationships.
2. Use WebShere Studio to build entity beans that use CMR.
3. Understand some of the coding implications of using entity beans with CMR.

Time required:

This tutorial should take about 1 hour to complete. You can take it online, or download a PDF file of the tutorial.

Audience:

This tutorial is intended for technical users familiar with Enterprise Java Beans (EJBs) and interested in learning how to implement some of the new features in the EJB 2.0 specification using WebSphere Studio 5.0.

Prerequisites:

You should have a basic knowledge of EJBs and how to use WebSphere Studio.

System requirement:

You will need the following hardware and software:

1. WebSphere Studio Application Developer 5.0
2. DB2 7.2 fixpack 6 or later

All set? Let's begin the [tutorial](#).

About the author

Gary Flood is an advisory IT Specialist working in the WebSphere Platform Solution Test Center at IBM Hursley, United Kingdom. His areas of expertise include the design and development of distributed applications using J2EE technologies, such as EJBs and JMS. You can reach him at floodg@uk.ibm.com.

Tutorial: Container Managed Relationships using WebSphere Studio Application Developer 5.0

[Gary Flood](#)

Advisory IT Specialist, WebSphere Platform Solution Test
IBM Hursley, United Kingdom
March, 2003

© Copyright International Business Machines Corporation 2003. All rights reserved.

Introduction

The second version of the Enterprise Java Beans (EJBs) specification from Sun® Microsystems has introduced a number of improvements for entity bean persistence. One of these improvements is the introduction of container managed relationships (CMR). CMR specifies logical relationships between entity beans. These logical relationships are mapped to the underlying persistence schema. The EJB Container is then responsible for managing this mapping and the referential integrity of the relationships from the point of view of entity bean instances.

CMR Fields

An entity bean accesses a related entity bean by means of the accessor methods for its CMR fields, which are defined by the `cmr-field` tag in the deployment descriptor. Relationships between entity beans is one-to-one (1-1), one-to-many (1-M), or many-to-many (M-N). They are navigated (or traversed) in one or both directions. Accessor methods for 1-M or M-N relationships use either the Collection or Set interface (List and Map will be included in later versions of the EJB 2.0 specification).

Limitations

Only EJBs that conform to the EJB 2.0 specification can take part in CMR relationships. In addition, only EJBs that expose functionality through a local interface can take part in the relationship. Local interfaces are another new concept introduced by the EJB 2.0 specification. They provide a client interface to EJB clients operating in the same JVM for enhanced performance.

CMR Example

WebSphere® Studio Application Developer 5.0 (hereafter called WebSphere Studio) supports EJB 2.0. The following example shows how to create two entity beans, define the CMR, and map the relationship to the underlying database. The example demonstrates key features of CMR.

1. [Create and configure the Enterprise Application and EJB projects.](#)
2. [Create two entity EJBs: Cust and Order.](#)
3. [Create a relationship between the two entity EJBs.](#)
4. [View generated methods to maintain relationships in entity bean classes.](#)
5. [Generate the database and EJB to RDB mapping.](#)
6. [Generate the deploy code for the EJBs.](#)
7. [Export database definitions to DB2®.](#)

Table of contents

[Introduction](#)

[CMR Fields](#)

[Limitations](#)

[CMR Example](#)

[Create and configure the Enterprise Application and EJB projects](#)

[Create two entity EJBs](#)

[Create a relationship between two entity EJBs](#)

[View generated methods](#)

[Generate the database and EJB to RDB mapping](#)

[Generate the deploy code for the EJBs](#)

[Export database definitions to DB2](#)

[Create a test server](#)

[Test the EJBs using the Universal Test Client](#)

[CMR Testing](#)

[Install the CMR Tester](#)

[Assignment Semantics for Relationships](#)

[Cascading Deletes](#)

8. [Create a test server.](#)
9. [Test the EJBs using the Universal Test Client.](#)

Step 1: Create and configure the Enterprise Application and EJB projects

To create the EAR project and EJB project, select **File -> New -> Enterprise Application Project**.

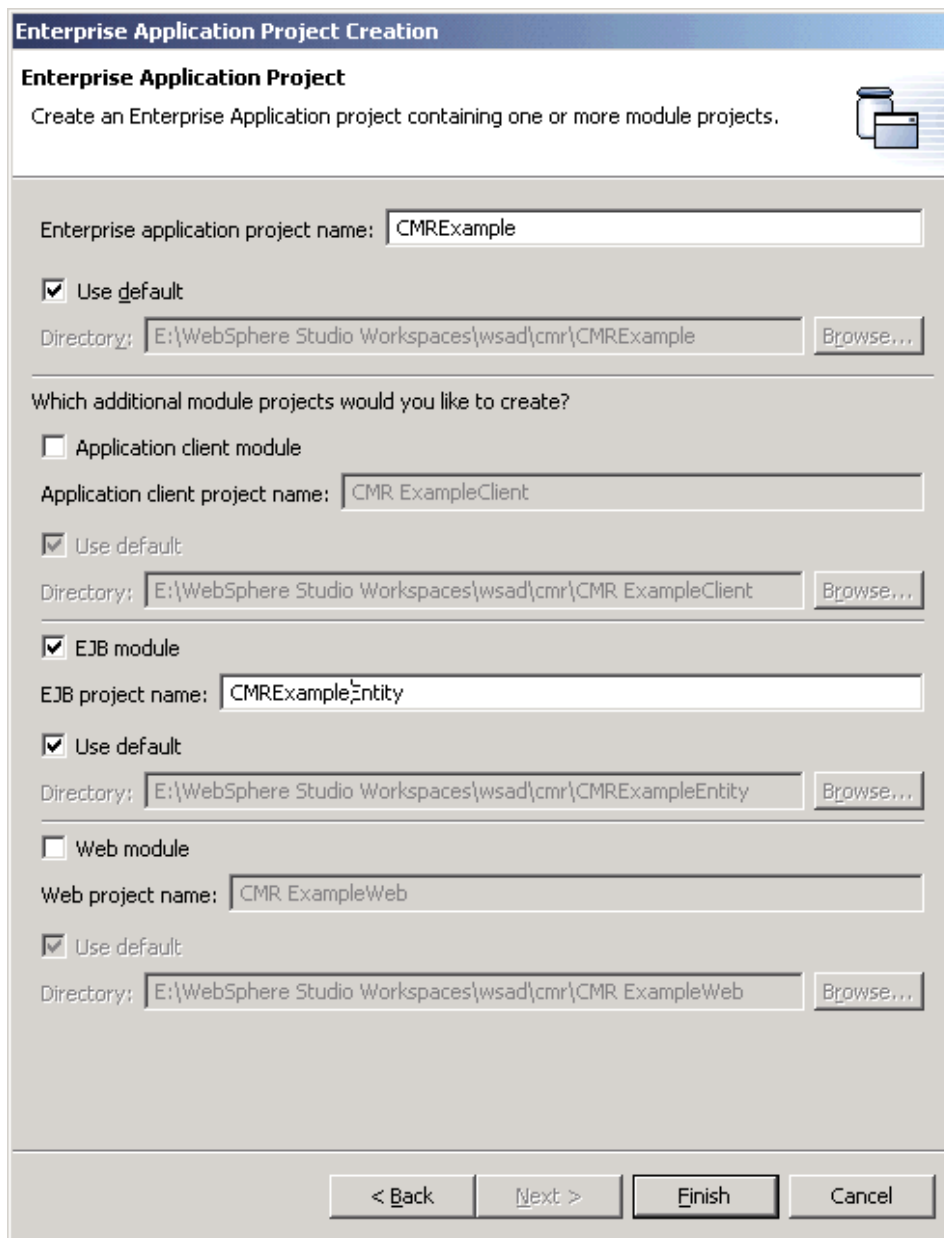
1. Select **Create J2EE 1.3 Enterprise Application Project**. This level supports EJB 2.0.

Figure 1. Enterprise Application Project Creation



2. Click **Next**.
3. Name the Enterprise Application Project, CMRExample.
4. Deselect the Application Client module.
5. Name the EJB module, CMRExampleEntity.
6. Deselect the Web Module.

Figure 2. Enterprise Application Project Creation



Enterprise Application Project Creation

Enterprise Application Project
Create an Enterprise Application project containing one or more module projects.

Enterprise application project name: CMRExample

☒ Use default
Directory: E:\WebSphere Studio Workspaces\wsad\cmr\CMRExample [Browse...](#)

Which additional module projects would you like to create?

☐ Application client module
Application client project name: CMR ExampleClient
☒ Use default
Directory: E:\WebSphere Studio Workspaces\wsad\cmr\CMR ExampleClient [Browse...](#)

☒ EJB module
EJB project name: CMRExampleEntity
☒ Use default
Directory: E:\WebSphere Studio Workspaces\wsad\cmr\CMRExampleEntity [Browse...](#)

☐ Web module
Web project name: CMR ExampleWeb
☒ Use default
Directory: E:\WebSphere Studio Workspaces\wsad\cmr\CMR ExampleWeb [Browse...](#)

< Back Next > **Finish** Cancel

7. Click **Finish**.

Step 2: Create entity beans

1. Create the Cust EJB. Select **File -> New -> Enterprise Bean**.
2. Select the **CMRExampleEntity** as the project for the new bean.
3. Select **Entity bean with container-managed persistence (CMP) fields** from the list and select **CMP 2.0 Bean** as the type. This EJB level supports CMR.
4. Enter **Cust** as the Bean name and **com.ibm.cmr** as the package.

Figure 3. Create an Enterprise Bean

Create an Enterprise Bean.

Create a 2.0 Enterprise Bean
Select the EJB 2.0 type and the basic properties of the bean.

☐ Message-driven bean
☐ Session bean
☐ Entity bean with bean-managed persistence (BMP) fields
☒ Entity bean with container-managed persistence (CMP) fields

☐ CMP 1.1 Bean ☒ **CMP 2.0 Bean**

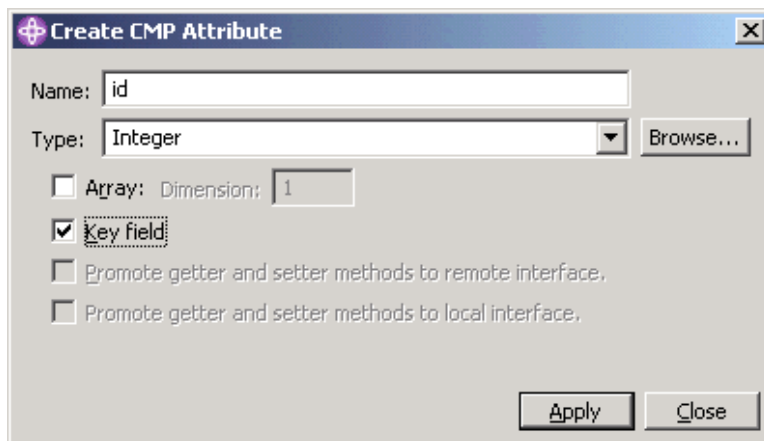
EJB project: CMRExampleEntity
 Bean name:
 Source folder:
 Default package:

5. Click **Next**.
6. Select the jndi name, **ejb/CustLocalHome**.
7. Select **Local client view**. CMR is used with local EJB interfaces in the EJB 2.0 specification.
8. In the CMP attribute listbox, click **Add...** and create the following CMP fields:

```

id - Integer (key field)
fName - String
lName - String
email - String
dob - java.sql.Date
    
```

Figure 4. Create CMP Attribute



Create CMP Attribute

Name:

Type:

☐ Array: Dimension:

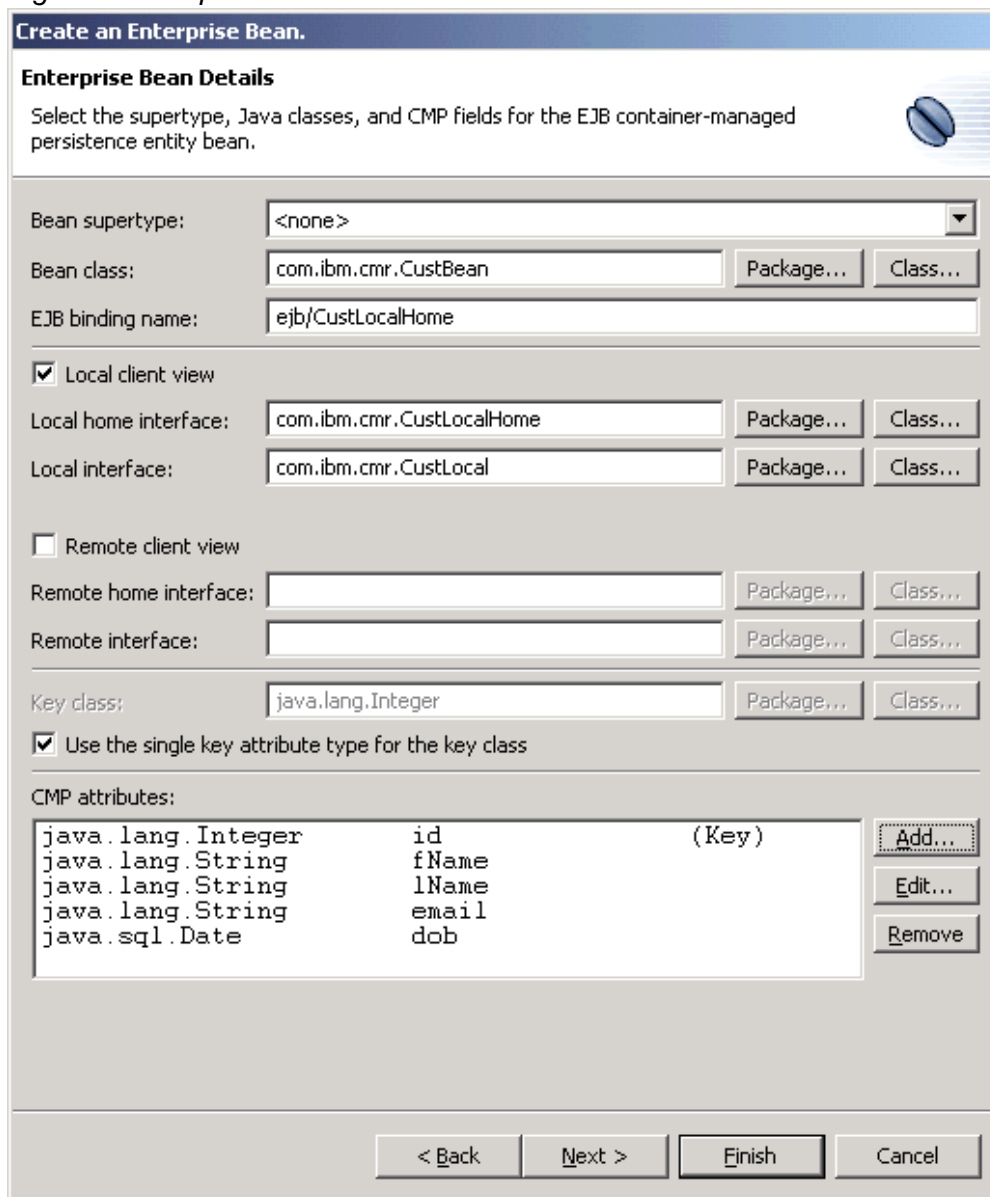
☒ Key field

☐ Promote getter and setter methods to remote interface.

☐ Promote getter and setter methods to local interface.

Your completed Enterprise Bean Details page displays as shown in Figure 5.

Figure 5. Enterprise Bean Details



Create an Enterprise Bean.

Enterprise Bean Details

Select the supertype, Java classes, and CMP fields for the EJB container-managed persistence entity bean.

Bean supertype:

Bean class:

EJB binding name:

☒ Local client view

Local home interface:

Local interface:

☐ Remote client view

Remote home interface:

Remote interface:

Key class:

☒ Use the single key attribute type for the key class

CMP attributes:

java.lang.Integer	id	(Key)
java.lang.String	fName	
java.lang.String	lName	
java.lang.String	email	
java.sql.Date	dob	

9. Click **Finish**.
10. Using the same procedure, create the Order EJB.
11. Create this EJB in the same EJB project as the Cust EJB. Otherwise, you cannot set up a relationship between them.
12. Select the jndi name, **ejb/OrderLocalHome**.
13. Specify the CMP fields:

```
id - Integer (key field)
ship dt - java.sql.Date,
cost - Integer
```

Step 3: Create a relationship between the two entity EJBs

1. From the J2EE navigator, expand the CMRExampleEntity project and open the EJB Deployment Descriptor editor.
2. From the Relationships 2.0 panel, click **Add...**
3. Select the two EJBs that will participate in the relationship, Order and Cust. The relationship name defaults to Order-Cust.

Figure 6. Add Relationship

Add Relationship

Relationship

Create an association between two enterprise beans.

Order-Cust

Order

Cust

Source EJB:

Cust

Order

Relationship name: Order-Cust

Description: Relationship indicating Customer for an Order

< Back Next > Finish Cancel

4. Click **Next**.
5. On the Relationship Roles page, you can specify the:
 - o Nature of the relationship between the EJBs, namely the cardinality, 1-1, 1-M, M-1 or M-N.
 - o The direction of the relationship, such as do you want to obtain the customer for an order instance or orders for a customer instance, or both.
 - o Whether or not there are cascading deletes from the parent entity to the child entity.

For the example, each order is placed by an individual customer and a customer can potentially have multiple orders. Therefore, the relationship cardinality for Order-Cust is M-1.

The Relationship Roles panel displays the relationship from both the UML viewpoint and the EJB specification viewpoint. Initially, the relationship is setup as a 1-1 relationship. The UML view shows *One* at the end of each link and the EJB view shows the multiplicity set to *One*.

6. To indicate that the customer can have many orders, change the Multiplicity on the Order EJB to *Many*.

The UML view shows that the Order entity is the *Many* side of relationship. The CMR fields representing the relationship are defined. The CMR field *order*, which is added to the Cust EJB, is set to *Collection* because it can contain multiple Order instances.

The foreign key checkbox on the Order EJB is set and is not editable because each Order points to one customer. You can change the name of this field from *order* to *orders* to clarify the cardinality. The CMR field type added to the Order EJB is also greyed out because it automatically sets to the related entity type in the local interface for Cust.

7. You have completed the Relationship Roles panel as shown in Figure 7.

Figure 7. Relationships Role

Add Relationship

Relationship Roles
Create the relationship roles between two enterprise beans.

UML view:

Order-Cust

Order (Many) orders

Cust (One) cust

EJB specification view:

Source EJB: Order

Role name: cust

Multiplicity: Many

☒ Navigable

CMR field referencing Cust: cust

CMR field type: [dropdown]

☐ Cascade delete

☒ Foreign key

Source EJB: Cust

Role name: order

Multiplicity: One

☒ Navigable

CMR field referencing Order: orders

CMR field type: java.util.Collection

☐ Cascade delete

☐ Foreign key

< Back Next > Finish Cancel

8. Click **Finish**.
9. Select the source panel of the Deployment Descriptor and scroll down. You should see the following XML snippet defining the relationship between Order and Cust:

```
<relationships>
<ejb-relation>
```

```

<description>Relationship indicating Customer for and Order</description>
<ejb-relation-name>Order-Cust</ejb-relation-name>
<ejb-relationship-role>
<ejb-relationship-role-name>cust</ejb-relationship-role-name>
<multiplicity>Many</multiplicity>
<relationship-role-source>
<ejb-name>Order</ejb-name>
</relationship-role-source>
<cmr-field>
<cmr-field-name>cust</cmr-field-name>
</cmr-field>
</ejb-relationship-role>
<ejb-relationship-role>
<ejb-relationship-role-name>order</ejb-relationship-role-name>
<multiplicity>One</multiplicity>
<relationship-role-source>
<ejb-name>Cust</ejb-name>
</relationship-role-source>
<cmr-field>
<cmr-field-name>orders</cmr-field-name>
<cmr-field-type>java.util.Collection</cmr-field-type>
<cmr-field>
</ejb-relationship-role>
</ejb-relation>
</relationships>

```

10. Save the changes to the Deployment Descriptor.

Step 4: View generated methods to maintain relationships in entity bean classes

Let's take a look at the generated code changes.

1. Return to the J2EE navigator and expand the **com.ibm.cmr** package.
2. Open the **CustBean.java** file.

When you created the EJB, the getter and setter methods were generated for the CMP fields. You also now have getter and setter methods for the Container Managed Relationship you just created.

```

/**
 * This method was generated for supporting the relationship role named orders.
 * It will be deleted/edited when the relationship is deleted/edited.
 */
public abstract java.util.Collection getOrders();
/**
 * This method was generated for supporting the relationship role named orders.
 * It will be deleted/edited when the relationship is deleted/edited.
 */
public abstract void setOrders(java.util.Collection anOrders);

```

Similarly, if we open the OrderBean.java file we'll see the the CMR accessor methods for the relationship with Cust.

```

/**
 * This method was generated for supporting the relationship role named cust.
 * It will be deleted/edited when the relationship is deleted/edited.
 */

```

```
public abstract com.ibm.cmr.CustLocal getCust();  
/**  
 * This method was generated for supporting the relationship role named cust.  
 * It will be deleted/edited when the relationship is deleted/edited.  
 */  
public abstract void setCust(com.ibm.cmr.CustLocal aCust);
```

Note the attribute type for setCust is com.ibm.cmr.CustLocal.

Step 5: Generate the database and EJB to RDB mapping

1. Right hand click on the **CMRExampleEntity** project and select **Generate -> EJB to RDB mapping**.
2. Select **create a new backend** folder and click **Next**.
3. Select top down as the mapping method. This automatically generates the database schema and mapping.
4. Set the database name to CMR and the schema to CMRSchema.
5. Ensure generate DDL is selected.

Figure 8. EJB to RDB Mapping

EJB to RDB Mapping

Create new EJB/RDB Mapping

Select Top Down Mapping Options:

Target Database:
DB2 Universal Database V7.2

Database name:
CMR

Schema name:
CMRSHEMA

☒ Generate DDL

☐ WebSphere 3.x Compatible

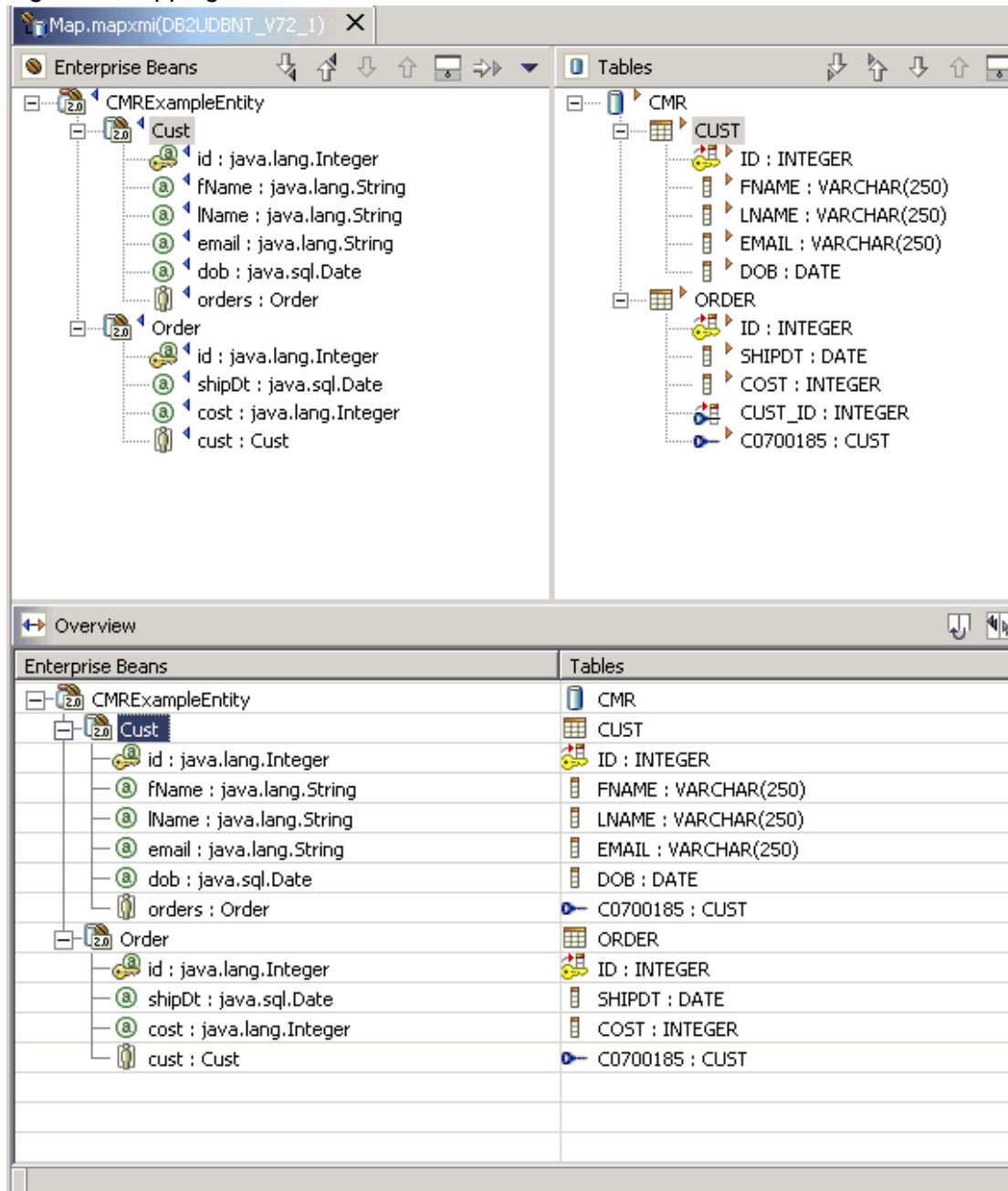
Advanced Options for Inheritance:
Click next to go the Advanced Options Page.

< Back Next > Finish Cancel

6. Click **Finish**.
7. The Mapping Editor is displayed. If you expand the twisties, you see the mappings.

The right hand side of the view displays the generated DB2 tables and fields. You see a foreign key field in the Order table that was created based on the EJB relationship. If you select the foreign key field in the top panel, you see that it is mapped to both the orders, CMR field in the Cust EJB and the cust CMR field in the Order EJB as shown in Figure 9.

Figure 9. Mapping Editor

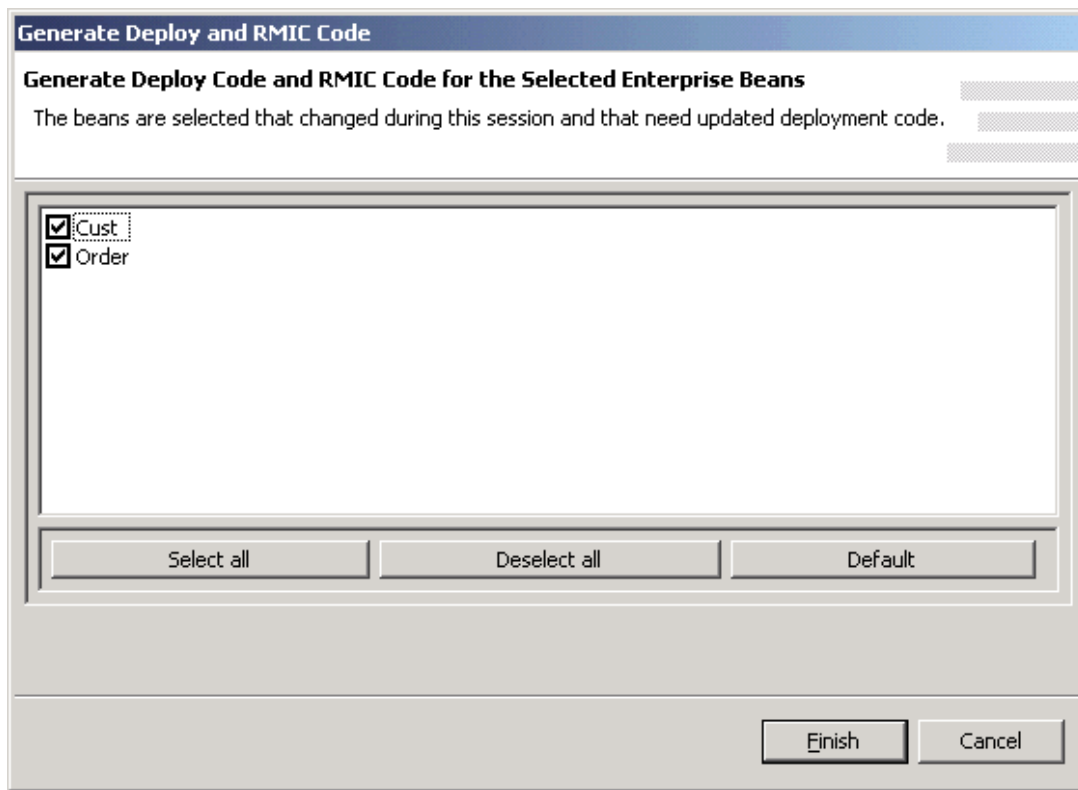


8. Close the Mapping Editor.

Step 6: Generate the deploy code for the EJBs

1. Right hand click on the **CMRExampleEntity** project and select **Generate Deploy and RMIC Code...**
2. Select both EJBs and click **Finish**.

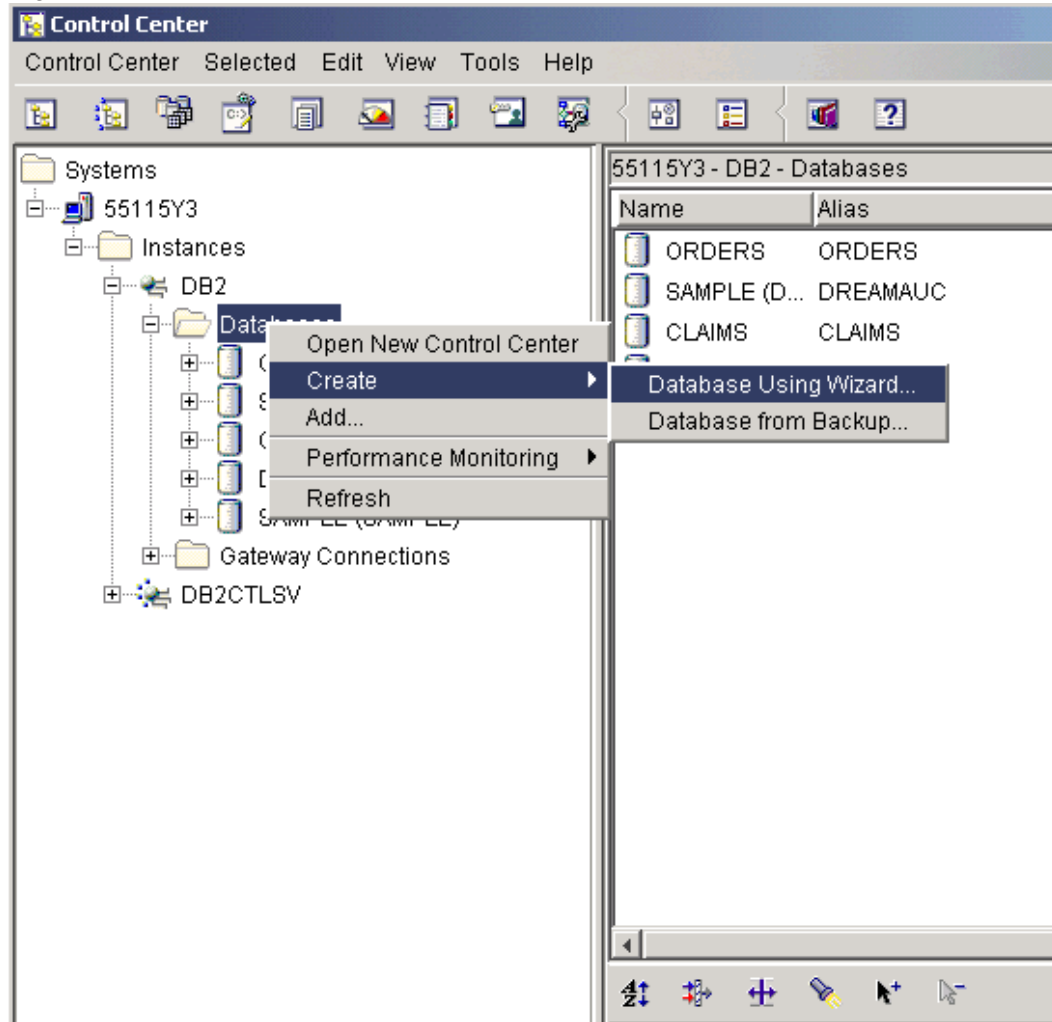
Figure 10. Generate Deploy and RMIC Code



Step 7: Export database definitions to DB2

1. From the DB2 Control Center, use the database wizard to create a database.

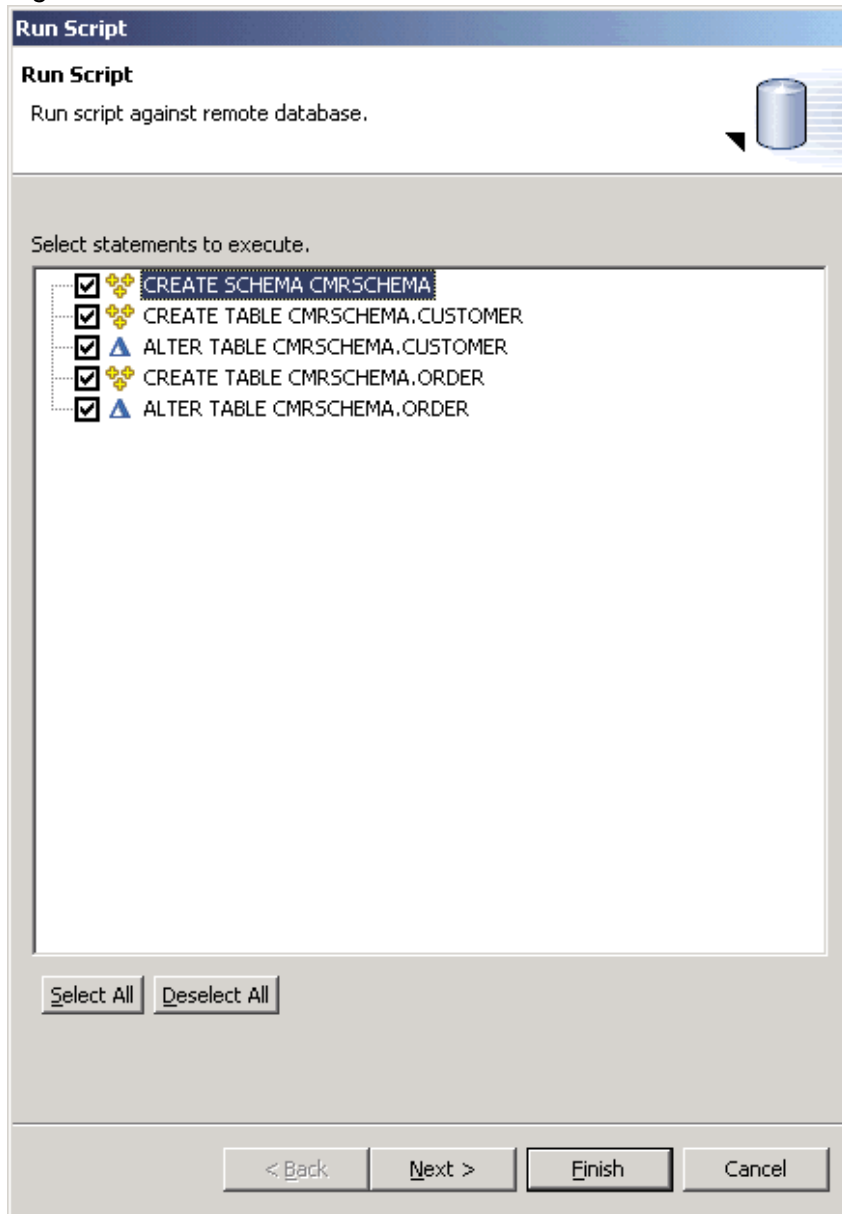
Figure 11. DB2 Control Center





2. Enter the database name as **CMR** and click **Finish**.
3. Returning to WebSphere Studio, under the **CMRExampleEntity** project, expand **ejbmodule**, **META-INF**, **backends**, **DB2UDBNT_V72_1**.
4. Right hand click on the **table.ddl** file.
5. Select **Run on Database Server...** The SQL to be executed is shown in Figure 12.

Figure 12. SQL to be executed



6. Select **Next** and **Next** again.
7. Specify the database name as **CMR**. Leave the rest of the options as defaults. If your DB2 installation is remote, you need to use the **IBM DB2 NET DRIVER**.

Figure 13. Database Connection

New

Database Connection

Establish a JDBC connection to a database.

Connection name: CMRConnection

Database: CMR

User ID:

Password:

Database vendor type: DB2 Universal Database V7.2

JDBC driver: IBM DB2 APP DRIVER

Host:

(Optional) Port number:

Server name:

Database Location: Browse...

JDBC driver class: COM.ibm.db2.jdbc.app.DB2Driver

Class location: E:\SQLLIB\java\db2java.zip Browse...

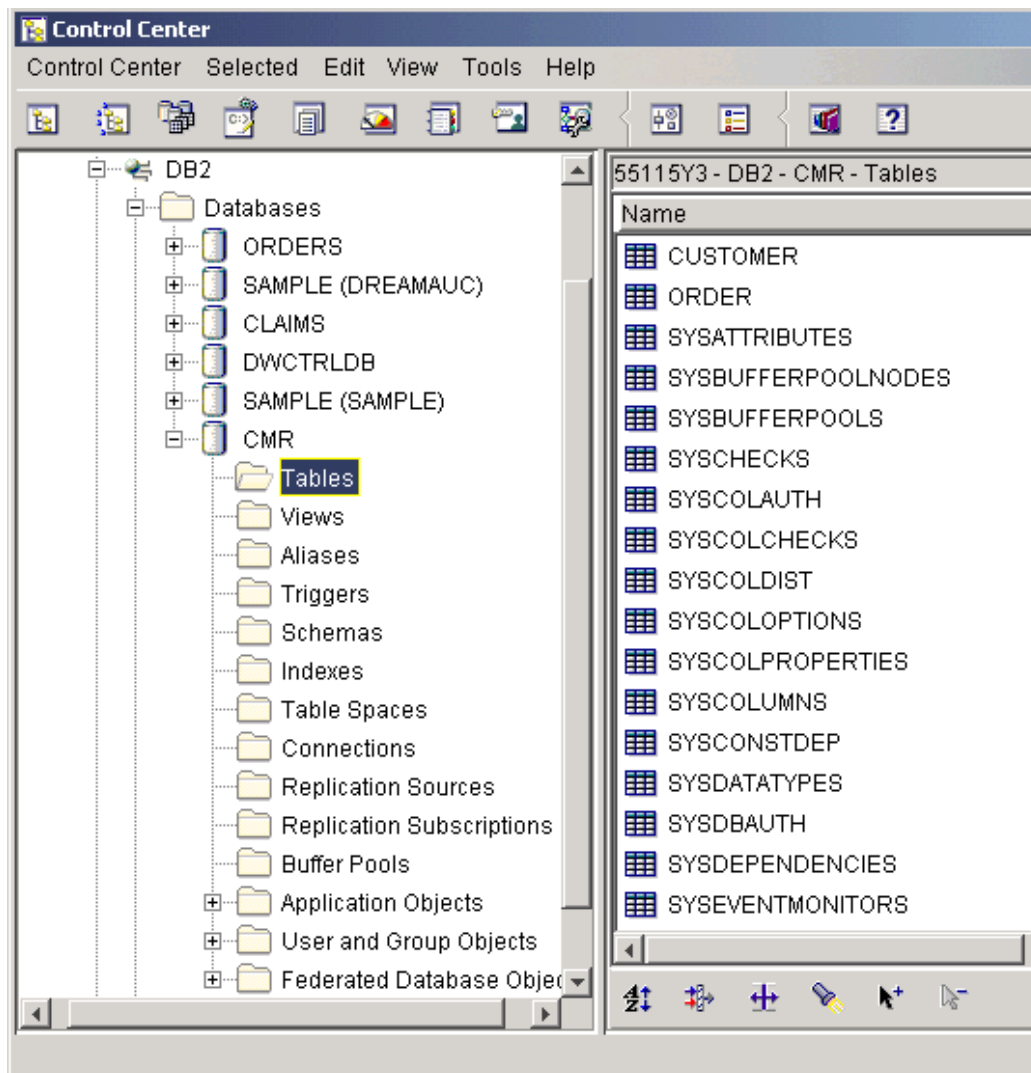
Connection URL: jdbc:db2:CMR

Filters...

Finish Cancel

8. Click **Finish**.
9. The database is updated with the new schema and table definitions. To confirm, switch back to the DB2 Control Center and expand the CMR database and select **Tables** as shown in Figure 14.

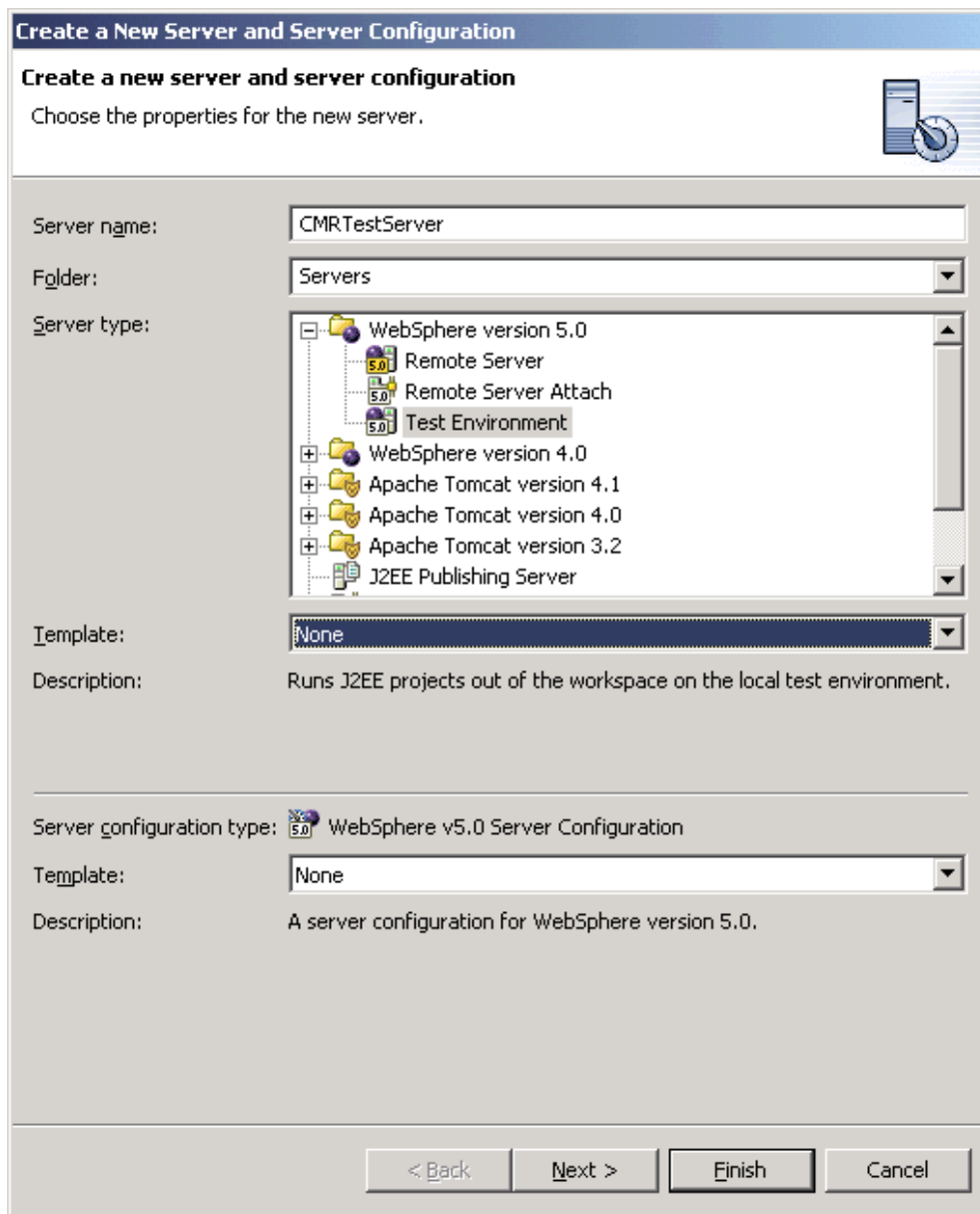
Figure 14. DB2 Control Center



Step 8: Create a test server

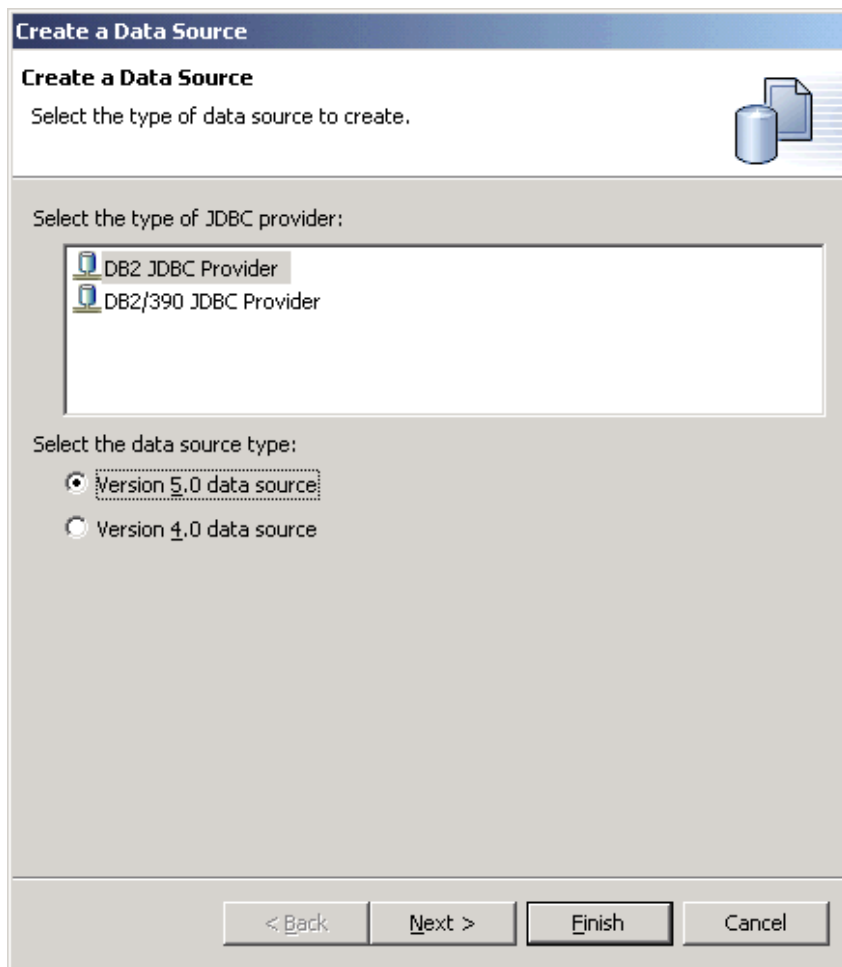
1. Change to the Server perspective and right hand click on the **Server Configurations** panel.
2. Select **New -> Server and Server Configuration**.
3. Set the server name to CMRTTestServer.

Figure 15. Create a New Server



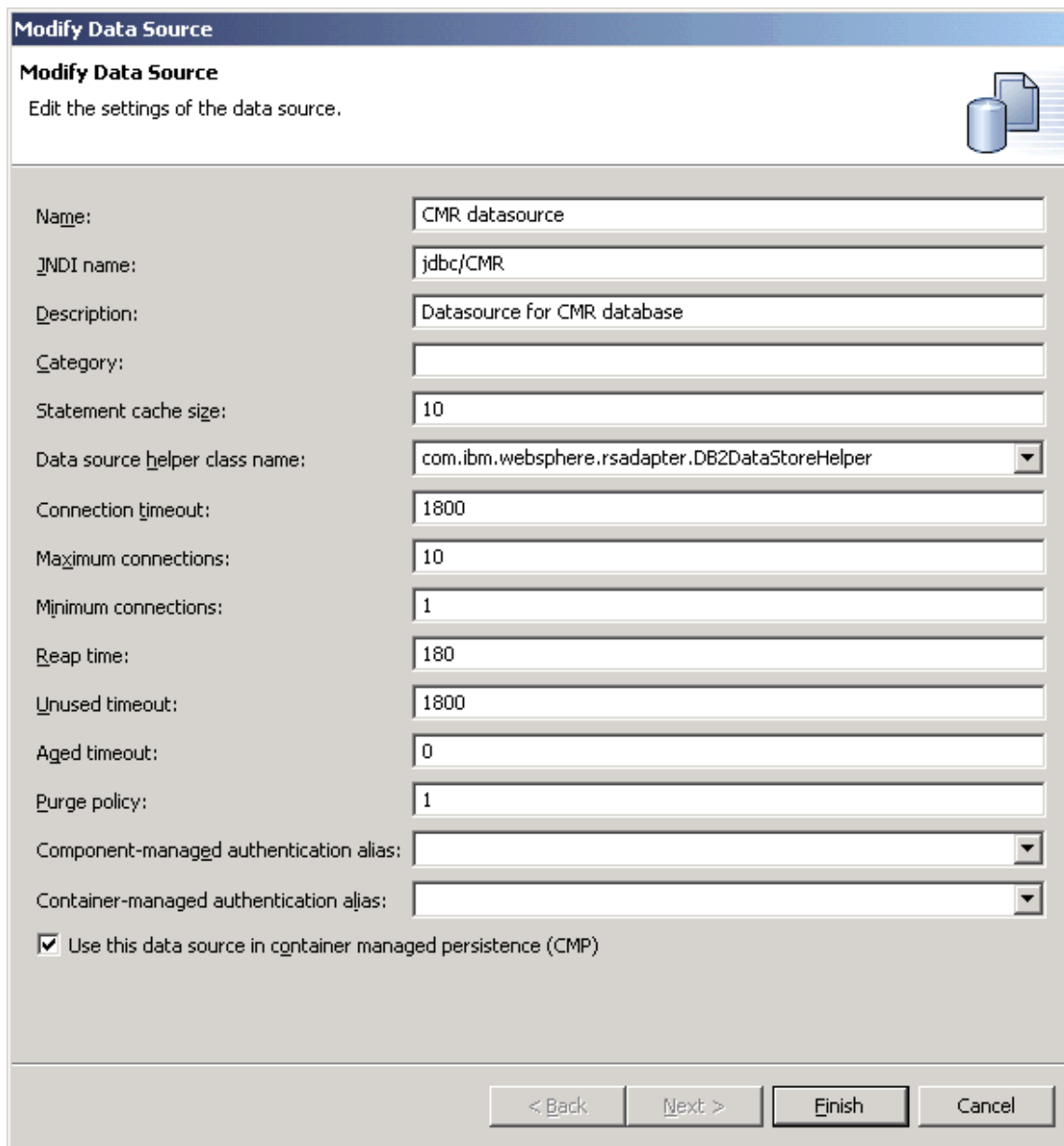
4. Click **Finish**.
5. Double click on the newly created server. Switch to the **Data Source** tab. In the JDBC provider list, ensure the **Default DB2 JDBC** provider is selected.
6. In the Data source panel, click the **Add** button.
7. In the Create data source panel, select **DB2 JDBC Provider** as the provider type, and **Version 5.0** as the data source version. This version supports CMR.

Figure 16. Create a Data Source




8. Click **Next**.
9. Set the datasource properties as shown in Figure 17.

Figure 17. Modify Data Source



Modify Data Source

Edit the settings of the data source.



Name: CMR datasource

JNDI name: jdbc/CMR

Description: Datasource for CMR database

Category:

Statement cache size: 10

Data source helper class name: com.ibm.websphere.rsadapter.DB2DataStoreHelper

Connection timeout: 1800

Maximum connections: 10

Minimum connections: 1

Reap time: 180

Unused timeout: 1800

Aged timeout: 0

Purge policy: 1

Component-managed authentication alias:

Container-managed authentication alias:

☒ Use this data source in container managed persistence (CMP)

< Back Next > Finish Cancel

10. Click **Next**.
11. In the Resource Properties pane, set the database name to CMR.
12. Click **Finish**.
13. Save the server configuration.
14. At this point, you need to link the EJBs to this datasource. Open the deployment descriptor in the EJB project.
15. Scroll down to the JNDI - CMP Factory Connection Binding panel and in the JNDI name field enter jdbc/CMR.
16. Set the Container authorization type to Per_Connection_Factory.

Figure 18. Connection Binding

JNDI - CMP Factory Connection Binding

Binding on the JAR level will create a "default" CMPFactory for CMP beans.

JNDI name: jdbc/CMR

Container authorization type: Per_Connection_Factory

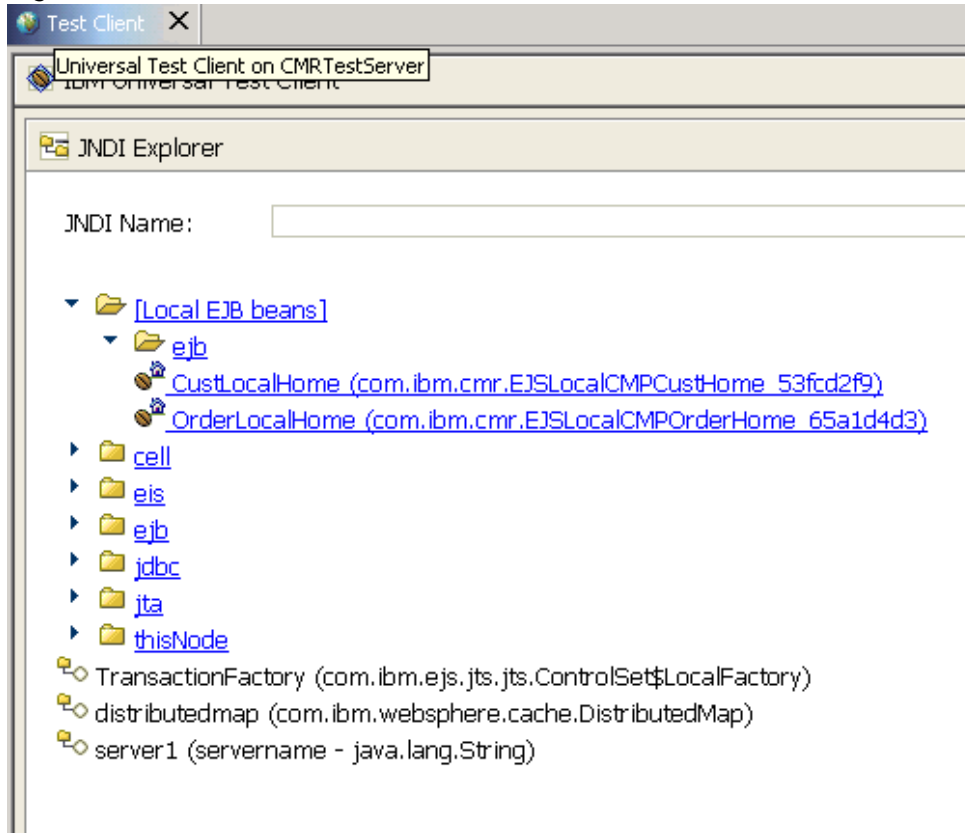
Remove

17. Save and close the deployment descriptor.

Step 9: Test the EJBs using the Universal Test Client

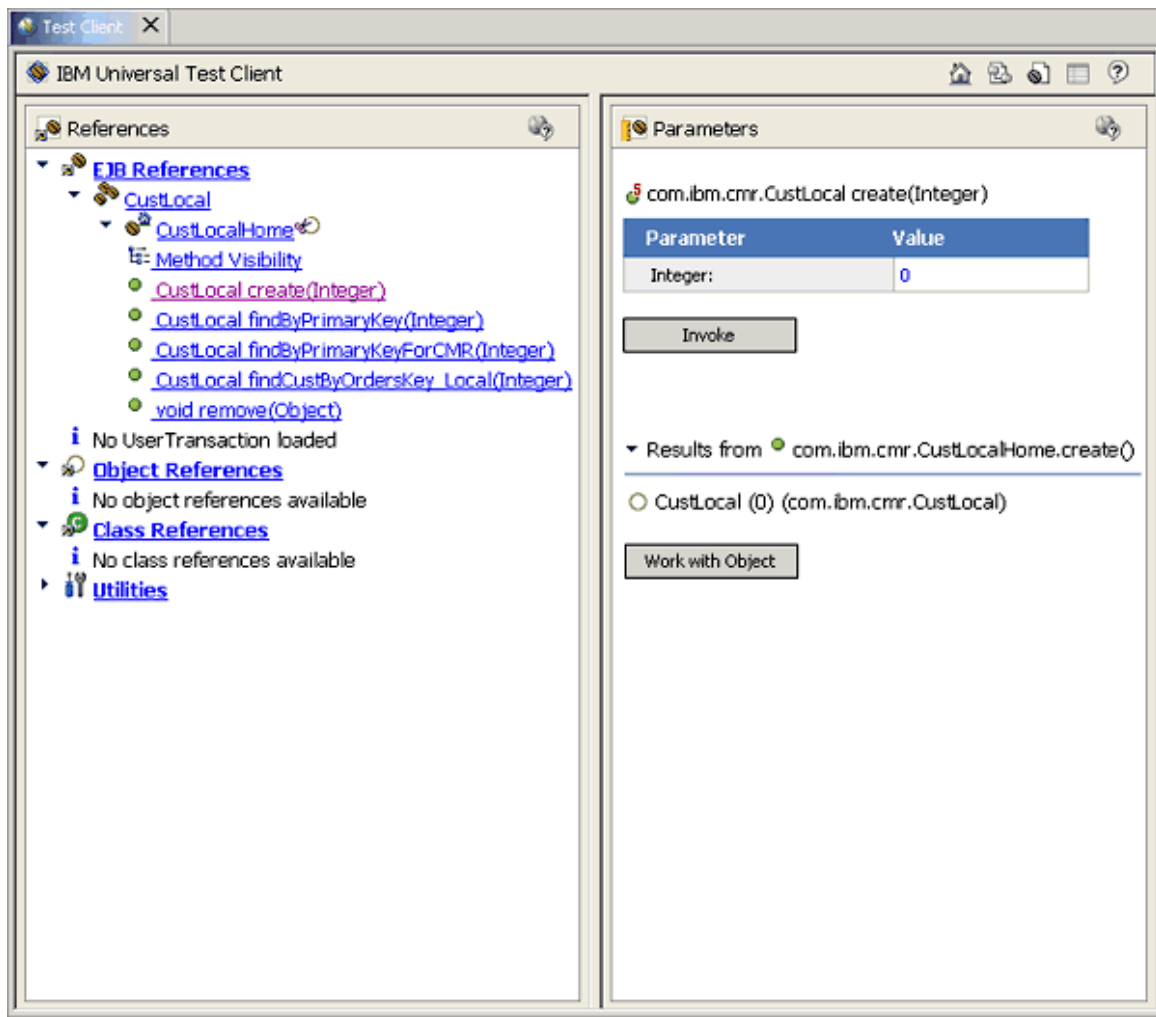
1. Switch back to the Server perspective, and right hand click on the **CMRTestServer**.
2. Select **Add -> CMRExample**.
3. Right hand click on the server from the Server tab and select **Start**.
4. When the console displays the message, "Server server1 open for e-business Return to the server tab", right hand click on the server and select **Run universal test client**.
5. Select **JNDI Explorer**, then from the Explorer, expand the local EJBs. You see the Cust and Order EJBs as shown in Figure 19.

Figure 19. Universal Test Client



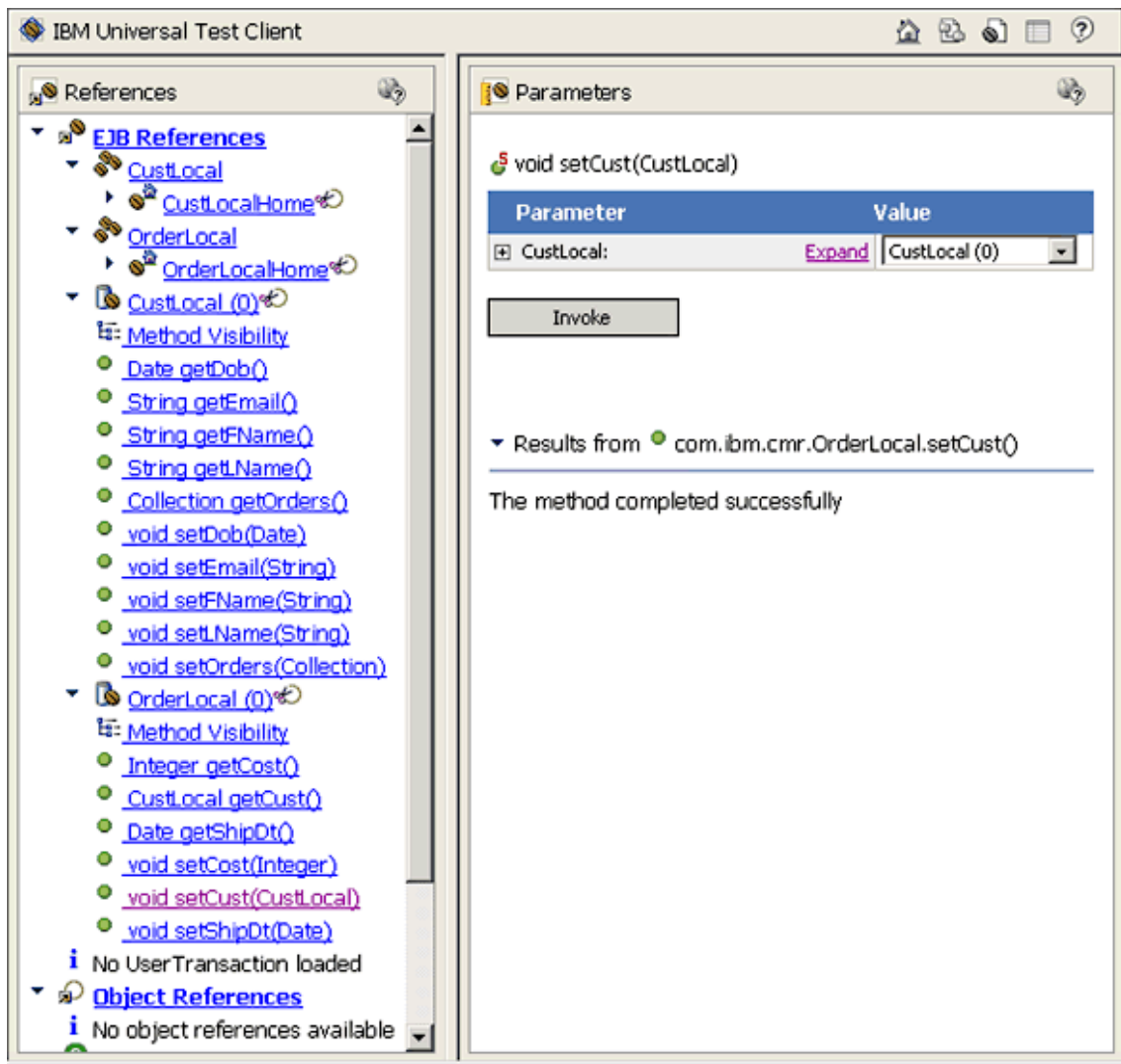
6. Select the **CustLocalHome** to go to the bean page.
7. In the bean page, expand the **CustLocal** EJB to reveal the home methods.
8. Select and invoke the **create(Integer)** method.

Figure 20. Create integer method



9. Click the **Work with Object** button.
10. In the same way, return to the JNDI Explorer in the Universal Test Client and select the **Order** EJB. Again, expand the **OrderLocal** and use its home interface create(Integer) method to create an Order. Click the **Work with Order** button.
11. From the available Order methods, select the **setCust(CustLocal)** method.
12. In the drop down list, you see the Cust that was created earlier. Select **Cust** and click the **Invoke** button.

Figure 21. Successful completion of method



As shown in Figure 21, the generated CMR assessor updated the Order foreign key to point to the correct customer. The container located the value of the Cust that corresponds to the foreign key in the Order table, and performed the SQL update using this value.

13. *Congratulations*, you have completed this tutorial. If you want to test the results, continue to the next section, "CMR Testing".

CMR Testing

With the available source code is an additional EJB jar file, [CMRTestSession.jar](#). This jar file contains a session bean that allows you to test entity beans by manipulating customers and orders through the CMR fields and the collection API. Four of these methods demonstrate assignments and deletes on entity instances involved in CMR. The full working solution, along with the tester, is also supplied as [CMRExample.ear](#).

Installing the CMR Tester

To install the session bean:

1. Select **File -> Import -> EJB JAR file**.
2. Specify the location of the jar file and import it into the existing EJB Project, CMRExampleEntity.
If you import the jar into a new EJB project, update the new project's build path to point to the existing CMRExampleEntity project, and update the new project's dependencies to point to the existing CMRExampleEntity project.
3. Click **Finish**.

4. When prompted to add the new project to EAR project, click **Ok**.
5. You can invoke the methods of the test session bean using the Universal Test Client described in [Step 9](#). In the JNDI Explorer, the CMPTester bean is found by expanding the EJB twisty and not the `[local ejb beans]`.

Assignment Semantics for Relationships

The result of an assignment operation using CMR fields is complex and depends on the cardinality of the relationship (1-1, and so on) and the direction of the relationship (unidirectional or bi-directional).

For entity instances that are associated with one and only one instance of another entity (entities involved in a 1-1 or the many side of a 1-M relationship), if an object currently associated with one entity instance is used as the input to a set assessor method on another entity, the original entity loses its association to the object. The object is now associated to the new entity.

This is seen from the `transferAnOrder()` method. It takes an order that is associated with one customer, and uses the `setCust(CustLocal newCust)` method to transfer the order to a new customer. The old customer is no longer associated with the order.

For entity instances associated with many instances of another entity, using the set assessor method has the effect of removing the object's association with all current instances, and establishing a new relationship with each entity represented by the object collection passed to the method. Each entity originally associated with the object has its association (or foreign key) set to null, and each entity in the collection has its association updated to the new object. This is seen in part in the `transferOrders()` method. This method is attempting to transfer all of the orders belonging to one customer to another customer. However, if you do this blindly by simply using the CMR setter field on the customer entity, `newCust.setOrders(oldCust.getOrders())`, then the effect loses the associations between any existing orders and the new customer.

Therefore, this line is only used in the instance where the new customer has no current orders. However, you see that the line has the effect of changing the relationship for each order in the collection, which was originally associated with the old customer. If the new customer already has associations with existing orders that you want to preserve, you can use the iterator to add the collection of orders associated with the old customer to the collection of orders associated with the new customer, `newOrders.addAll(oldCust.getOrders())`.

As shown in the code examples, it is possible to manipulate the relationships both through the entity object itself (using the setter and getter CMR fields), and through the collection API using the iterator's `add()`, `addAll()`, `remove()`, and `removeAll()` methods. You must be careful not to modify the contents of a collection by any means other than the iterator while an iterator is open upon the collection. To demonstrate this, the `clearOrdersForCust()` method has two examples of illegal modifications. The first attempts to call the entity's `remove()` method while an iterator is open upon a collection that contains the entity. The second attempts to remove the order from the collection, again while the iterator is open on the collection.

Cascading Deletes

When defining the relationship between the Cust and Order entities ([Step 3](#)), you did not select cascade delete in the Relationship Roles panel. Consequently, when you run the `deleteCust()` method in the `CMRTestSession`, the result after the deletion of the customer is removing the customer from the database and setting the `custid` foreign key to null. If you now edit the role through the deployment descriptor for the `CMRExampleEntity`, and select the Cascade delete option and rerun the `deleteCust()` method, the Orders associated to the customer are removed.

Conclusion

This tutorial explained the concepts of Container Managed Relationships and demonstrated how to develop entity beans that use CMR using WebSphere Studio Application Developer 5.0. It also provided examples that demonstrate assignments and deletes on entity instances involved in CMR.

Related information

- [Enterprise Java Beans 2.0 specification](#)

[Top of page](#)

Download

To access the download files, please see the [Web version](#) of this tutorial.

DB2, IBM, and WebSphere are trademarks or registered trademarks of IBM Corporation in the United States, other countries, or both.

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

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

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

[IBM copyright and trademark information](#)