# Introduction to Entity Beans

## Relationships

## ejb-jar.xml

```
<?xml version="1.0"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD
Enterprise JavaBeans 2.0//EN" "http://java.sun.com/dtd/ejb-
jar_2_0.dtd">

<ejb-jar>
 <enterprise-beans>
  ...
 </enterprise-beans>
 <relationships>
   ...
 <relationships>
 <assembly-descriptor>
  ...
 </assembly-descriptor>
</ejb-jar>
```

## ejb-jar.xml

```
<enterprise-beans>
 <entity>
    <ejb-name>CustomerEJB</ejb-name>
    <home>com.titan.customer.CustomerHomeRemote</home>
    <remote>com.titan.customer.CustomerRemote</remote>
    <ejb-class>com.titan.customer.CustomerBean</ejb-class>
    <persistence-type>Container</persistence-type>
    <prim-key-class>java.lang.Integer</prim-key-class>
    <reentrant>False</reentrant>
    <cmp-version>2.x</cmp-version>
    <abstract-schema-name>Customer</abstract-schema-name>
    <cmp-field><field-name>id</field-name></cmp-field>
    <cmp-field><field-name>lastName</field-name></cmp-field>
    <cmp-field><field-name>firstName</field-name></cmp-field>
    <cmp-field><field-name>hasGoodCredit</field-name>
        </cmp-field>
    <primkey-field>id</primkey-field>
    <security-identity><use-caller-identity/></security-identity>
  </entity>
```

## ejb-jar.xml

```
  <entity>
        <ejb-name>AddressEJB</ejb-name>
        <local-home>com.titan.address.AddressHomeLocal
              </local-home>
        <local>com.titan.address.AddressLocal</local>
        <ejb-class>com.titan.address.AddressBean</ejb-class>
        <persistence-type>Container</persistence-type>
        <prim-key-class>java.lang.Integer</prim-key-class>
        <reentrant>False</reentrant>
        <cmp-version>2.x</cmp-version>
        <abstract-schema-name>Address</abstract-schema-name>
        <cmp-field><field-name>id</field-name></cmp-field>
        <cmp-field><field-name>street</field-name></cmp-field>
        <cmp-field><field-name>city</field-name></cmp-field>
        <cmp-field><field-name>state</field-name></cmp-field>
        <cmp-field><field-name>zip</field-name></cmp-field>
        <primkey-field>id</primkey-field>
        <security-identity><use-caller-identity/></security-identity>
  </entity>
</enterprise-beans>
```

## ejb-jar.xml

```
<relationships>
  <ejb-relation>
     <ejb-relation-name>Customer-HomeAddress</ejb-relation-name>
     <ejb-relationship-role>
           ...
     </ejb-relationship-role>
     <ejb-relationship-role>
           ...
     </ejb-relationship-role>
   </ejb-relation>
</relationships>
```

## ejb-jar.xml

```
<ejb-relationship-role>
 <ejb-relationship-role-name>   Customer-has-a-Address
     </ejb-relationship-role-name>
     <multiplicity>One</multiplicity>
     <relationship-role-source>
          <ejb-name>CustomerEJB</ejb-name>
     </relationship-role-source>
      <cmr-field>
          <cmr-field-name>homeAddress</cmr-field-name>
     </cmr-field>
</ejb-relationship-role>
<ejb-relationship-role>
     <ejb-relationship-role-name>Address-belongs-to-Customer
     </ejb-relationship-role-name>
     <multiplicity>One</multiplicity>
     <relationship-role-source>
          <ejb-name>AddressEJB</ejb-name>
     </relationship-role-source>
</ejb-relationship-role>
```

```
<assembly-descriptor>
  <security-role>
    <description> ... </description>
    <role-name>everyone</role-name>
  </security-role>
  <method-permission>
    <role-name>everyone</role-name>
    <method>
      <ejb-name>CustomerEJB</ejb-name>
      <method-name>*</method-name>
    </method>
    <method>
      <ejb-name>AddressEJB</ejb-name>
      <method-name>*</method-name>
    </method>  </method-permission>
  <container-transaction>
    <method>
      <ejb-name>CustomerEJB</ejb-name>
      <method-name>*</method-name>
    </method>
    <method>
      <ejb-name>AddressEJB</ejb-name>
      <method-name>*</method-name>
    </method>

    <trans-attribute>required</trans-attribute>
  </container-transaction>
</assembly-descriptor>
```

## Example - 1

1. AddressBean

```
package com.titan.address;
import javax.ejb.EntityContext;
public abstract class AddressBean implements javax.ejb.EntityBean
{
  private static final int IDGEN_START =
                                    (int)System.currentTimeMillis();
  private static int idgen = IDGEN_START;

  public Integer ejbCreate(String street, String city,
        String state,  String zip ) throws javax.ejb.CreateException  {
    setId(new Integer(idgen++));
    setStreet(street);
    setCity(city);
    setState(state);
    setZip(zip);
    return null;
  }
  public void ejbPostCreate(String street, String city,
                            String state,  String zip)   { }
```

## Example - 2

1. AddressBean

```
  // persistent fields
  public abstract Integer getId();
  public abstract void setId(Integer id);
  public abstract String getStreet();
  public abstract void setStreet(String street);
  public abstract String getCity();
  public abstract void setCity(String city);
  public abstract String getState();
  public abstract void setState(String state);
  public abstract String getZip();
  public abstract void setZip(String zip);

  // standard call back methods
  public void setEntityContext(EntityContext ec){}
  public void unsetEntityContext(){}
  public void ejbLoad(){}
  public void ejbStore(){}
  public void ejbActivate(){}
  public void ejbPassivate(){}
  public void ejbRemove(){}
}
```

## Example - 3

2. CustomerRemote

```
package com.titan.customer;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.naming.NamingException;
public interface CustomerRemote extends javax.ejb.EJBObject
{
  public void setAddress(String street, String city,
                         String state, String zip)
    throws RemoteException;

  public void setAddress(AddressDO address)
    throws RemoteException;

  public AddressDO getAddress() throws RemoteException;

  public Name getName() throws RemoteException;
  public void setName(Name name) throws RemoteException;

  public boolean getHasGoodCredit() throws RemoteException;
  public void setHasGoodCredit(boolean flag) throws RemoteException;
}
```

## Example - 4

3a. AddressDO

```
package com.titan.customer;
public class AddressDO implements java.io.Serializable {

  private String street;
  private String city;
  private String state;
  private String zip;

  public AddressDO(String street, String city,
              String state, String zip )  {
    this.street = street;
    this.city = city;
    this.state = state;
    this.zip = zip;
  }
  public String getStreet()  {     return street;  }
  public String getCity()  {     return city;  }
  public String getState()  {     return state;  }
  public String getZip()  {     return zip;  }
}
```

## Example - 5

3b. Name

```
package com.titan.customer;
public class Name implements java.io.Serializable
{
  private String lastName;
  private String firstName;

  public Name(String lname, String fname)  {
    lastName = lname;
    firstName = fname;
  }

  public String getLastName()   {
    return lastName;
  }

  public String getFirstName()   {
    return firstName;
  }
}
```

## Example - 6

4. CustomerBean

```
package com.titan.customer;
import com.titan.address.*;
import javax.naming.*;
import javax.ejb.*;
public abstract class CustomerBean implements javax.ejb.EntityBean {

// metodi di creazione
  public Integer ejbCreate(Integer id) throws CreateException
     { this.setId(id); return null;}
  public void ejbPostCreate(Integer id){}

 // abstract accessor methods
  public abstract Integer getId();
  public abstract void setId(Integer id);
  public abstract String getLastName();
  public abstract void setLastName(String lname);
  public abstract String getFirstName();
  public abstract void setFirstName(String fname);
  public abstract boolean getHasGoodCredit();
  public abstract void setHasGoodCredit(boolean flag);
```

## Example - 7

4. CustomerBean

```
// persistent relationships

  public abstract AddressLocal getHomeAddress();
  public abstract void setHomeAddress(AddressLocal address);

// standard call back methods

  public void setEntityContext(EntityContext ec){}
  public void unsetEntityContext(){}
  public void ejbLoad(){}
  public void ejbStore(){}
  public void ejbActivate(){}
  public void ejbPassivate(){}
  public void ejbRemove(){}
```

## Example - 8

4. CustomerBean

```
// business methods

  public Name getName()  {
    Name name = new Name(getLastName(),getFirstName());
    return name;
  }
  public void setName(Name name) {
    setLastName(name.getLastName());
    setFirstName(name.getFirstName());
  }
```

## Example - 9

4. CustomerBean

```
  public void setAddress(String street, String city,
                         String state, String zip)   {
    AddressLocal addr = this.getHomeAddress( );
    try {
      if (addr == null)
      { // Customer doesn't have an address yet. Create a new one.
        InitialContext cntx = new InitialContext( );
        AddressHomeLocal addrHome =
          (AddressHomeLocal)cntx.lookup("AddressHomeLocal");
        addr = addrHome.create(street, city, state, zip);
        this.setHomeAddress(addr);
      } else {// Customer already has an address. Change its fields
        addr.setStreet(street);
        addr.setCity(city);
        addr.setState(state);
        addr.setZip(zip);
      }
    }
    catch (NamingException ne) { throw new EJBException(ne); }
    catch (CreateException ce) { throw new EJBException(ce); }
  }
```

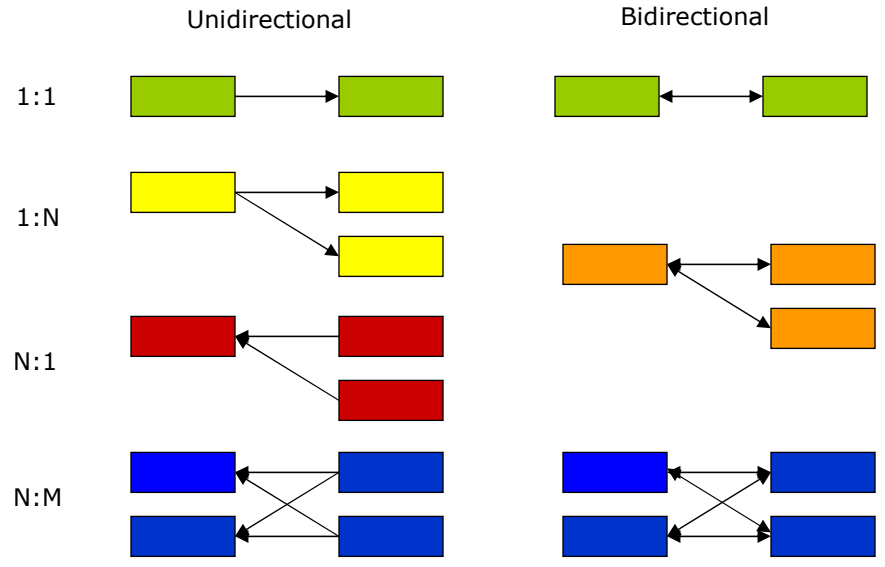
## Example - 10

4. CustomerBean

```
public AddressDO getAddress()   {
    AddressLocal addrLocal = this.getHomeAddress();
    if (addrLocal == null) return null;
    String street = addrLocal.getStreet();
    String city = addrLocal.getCity();
    String state = addrLocal.getState();
    String zip = addrLocal.getZip();
    AddressDO addrValue = new AddressDO(street,city,state,zip);
    return addrValue;
  }

  public void setAddress(AddressDO addrValue)   {
    String street = addrValue.getStreet();
    String city = addrValue.getCity();
    String state = addrValue.getState();
    String zip = addrValue.getZip();
    setAddress(street,city,state,zip);
  }
}
```

## Multiplicity and Directionality



## One-to-One Unidirectional

```
<ejb-relationship-role>
 <ejb-relationship-role-name>   Customer-has-a-Address
     </ejb-relationship-role-name>
     <multiplicity>One</multiplicity>
     <relationship-role-source>
          <ejb-name>CustomerEJB</ejb-name>
     </relationship-role-source>
      <cmr-field>
          <cmr-field-name>homeAddress</cmr-field-name>
     </cmr-field>
</ejb-relationship-role>
<ejb-relationship-role>
     <ejb-relationship-role-name>Address-belongs-to-Customer
     </ejb-relationship-role-name>
     <multiplicity>One</multiplicity>
     <relationship-role-source>
          <ejb-name>AddressEJB</ejb-name>
     </relationship-role-source>
</ejb-relationship-role>
```

## One to One Unidirectional

```
public class CustomerBean implements javax.ejb.EntityBean    {
   ...
   public abstract AddressLocal getHomeAddress();
   public abstract void setHomeAddress(AddressLocal address);
    ...
}
```

```
public class AddressBean implements javax.ejb.EntityBean    {
   ...
}
```

## One-to-One bidirectional

```
<ejb-relationship-role>
 <ejb-relationship-role-name>   Customer-has-a-Address
     </ejb-relationship-role-name>
     <multiplicity>One</multiplicity>
     <relationship-role-source>
          <ejb-name>CustomerEJB</ejb-name>
     </relationship-role-source>
      <cmr-field>
          <cmr-field-name>homeAddress</cmr-field-name>
     </cmr-field>
</ejb-relationship-role>
<ejb-relationship-role>
     <ejb-relationship-role-name>Address-belongs-to-Customer
     </ejb-relationship-role-name>
     <multiplicity>One</multiplicity>
     <relationship-role-source>
          <ejb-name>AddressEJB</ejb-name>
     </relationship-role-source>
     <cmr-field>
          <cmr-field-name>homeCustomer</cmr-field-name>
     </cmr-field>
</ejb-relationship-role>
```

## One to One Bidirectional

```
public class CustomerBean implements javax.ejb.EntityBean    {
   ...
   public abstract AddressLocal getHomeAddress();
   public abstract void setHomeAddress(AddressLocal address);
    ...
}
```

```
public class AddressBean implements javax.ejb.EntityBean    {
 ...
   public abstract CustomerRemote getHomeCustomer();
   public abstract void setHomeCustomer(CustomerRemote customer);
    ...

}
```

## One-to-Many unidirectional

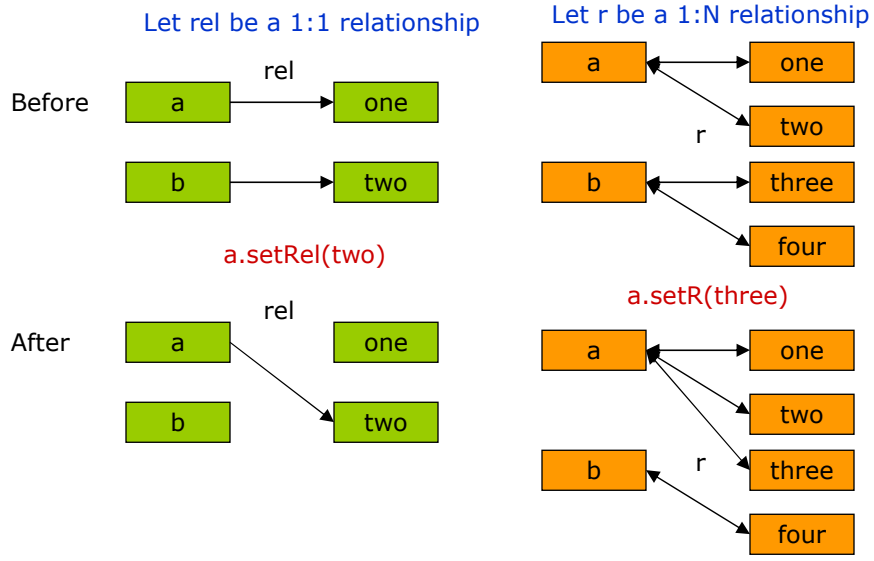
```
<ejb-relationship-role>
   <ejb-relationship-role-name>Company-Employs-Employees
   </ejb-relationship-role-name>
   <multiplicity>One</multiplicity>
   <relationship-role-source>
      <ejb-name>Company</ejb-name>
   </relationship-role-source>
   <cmr-field>
     <cmr-field-name>employees</cmr-field-name>
     <cmr-field-type>java.util.Collection</cmr-field-type>
   </cmr-field>
</ejb-relationship-role>
<ejb-relationship-role>
   <ejb-relationship-role-name>Employees-WorkAt-Company
   </ejb-relationship-role-name>
   <multiplicity>Many</multiplicity>
   <relationship-role-source>
        <ejb-name>Employee</ejb-name>
   </relationship-role-source>
</ejb-relationship-role>
```

or java.util.Set

## Many-to-Many bidirectional

```
<ejb-relationship-role>
   <ejb-relationship-role-name>Students-EnrollIn-Courses
   </ejb-relationship-role-name>
   <multiplicity>Many</multiplicity>
   <relationship-role-source>
      <ejb-name>Student</ejb-name>
   </relationship-role-source>
   <cmr-field>
      <cmr-field-name>courses</cmr-field-name>
      <cmr-field-type>java.util.Collection</cmr-field-type>
   </cmr-field>
</ejb-relationship-role>
<ejb-relationship-role>
   <ejb-relationship-role-name>Courses-HaveEnrolled-Students
   </ejb-relationship-role-name>
   <multiplicity>Many</multiplicity>
   <relationship-role-source>
      <ejb-name>Course</ejb-name>
   </relationship-role-source>
   <cmr-field>
       <cmr-field-name>students</cmr-field-name>
       <cmr-field-type>java.util.Collection</cmr-field-type>
   </cmr-field>
</ejb-relationship-role>
```

## Watch out for side effects!




## Cascade-delete

```
<ejb-relationship-role>
   <ejb-relationship-role-name>order-spawns-shipments
   </ejb-relationship-role-name>
   <multiplicity>One</multiplicity>
   <cascade-delete/>
   <relationship-role-source>
       <ejb-name>Order</ejb-name>
   </relationship-role-source>
   <cmr-field><cmr-field-name>order</cmr-field-name></cmr-field>
</ejb-relationship-role>
```

Deleting "a" also one,two e three are canceled!