PKUAS Developer Guide

Institute of Software

School of Electronics Engineering and Computer Science

Peking University

2006.03

License

Contents

List of Figures

Chapter 1. Introduction

PKUAS (PeKing University Application Server) is designed and implemented by Software Institute, School of Electronics Engineering and Computer Science of Peking University, who has the independent intellectual property rights. PKUAS provides comprehensive supports for J2EE/EJB applications. It has passed the test of SUN's JPS (Java PetStore) 1.3 and conformance test of ECperf1.0B MP.

PKUAS implements all the functionality of J2EE 1.3 and EJB 2.0. It provides containers for four standard types of EJB: stateless session bean, stateful session bean, entity bean and message-driven bean, as well as integrating Tomcat as its web container. It also provides various common services, such as naming, security, transaction, communication, logging, etc. All of them make up a perfective running environment for EJB and Web components. Besides, PKUAS supports RMI-IIOP interoperability protocol and enables accessing EJB via Remote/Local Interfaces.

This manual aims to be a guide for the designers and developers of J2EE applications, who will learn how to develop, assemble and deploy application on PKUAS.

The readers should understand and master the basic developing techniques of J2EE/EJB applications, which will not be included in this manual. You can refer to SUN's website and documents for further information about that.

This is a step-by-step guide for readers, including prerequisites, EJB/Web module development, application's assembly and deployment, with the addition of advanced features.

Chapter 2. Prepare

The preparations before developing J2EE applications, including the installation and configuration of JDK, application server and database. For more help, please refer to the official documents of J2SE and J2EE. Besides, developers might have to choose an appropriate IDE to enhance developing efficiency.

2.1 System Requirement

- Pentium class PC running Windows98(1st or 2nd Edition)/NT4.0(NT4.0 with Service Pack 5)/2000/ME/XP or Unix class (Linux,Solaris)
- 48 MB of physical RAM minimum (256 MB recommended)
- 70 MB available hard disk space(200 MB recommended)
- Java 2 SDK 1.4 or later version

2.2 Datasource configuration

PKUAS supports most mainstream databases, including MySql, Oracle, SQL Server, Cloudscape and etc. In fact, all JDBC-compatible databases can be used with PKUAS. Here is a quick guide to how to configure data-sources in PKUAS (taking MySql for example).

- Download MySql 3.23 or later version from http://www.mysql.com/ and finish the installation.
- Type ”mysql” in console to enter mysql interaction interface (use correct user name and password, if necessary. Here, we use ”test” for them both in the guide)
- Use ”create database” command to create a database. (EJBTEST, for example)
- Configure the created database in services.xml

The snippet of the configuration file:

2.1

```
<SERVICE CLASS="pku.as.datasvc.PoolManager"
    NAME="JdbcPool:name=jdbc/asdemo" >
    <ATTRIBUTE NAME="ExpirationTime" VALUE="300"/>
    <ATTRIBUTE NAME="MinCapacity" VALUE="5"/>
    <ATTRIBUTE NAME="URL"
        VALUE="jdbc:mysql://MysqlServerAddress/EJBTEST"/>
    <ATTRIBUTE NAME="User" VALUE="test"/>
    <ATTRIBUTE NAME="Password" VALUE="test"/>
    <ATTRIBUTE NAME="DriverClassName"
        VALUE="org.gjt.mm.mysql.Driver" />
    <ATTRIBUTE NAME="MaxCapacity" VALUE="30"/>
</SERVICE>
```

Figure 2.1 Datasource configuration Sample

The above ”JdbcPool:name=jdbc/asdemo” specifies the name of the data source. ”JdbcPool:name=” is the prefix. The following ”jdbc/asdemo” is the database string, which will be used by the developers while developing J2EE applications. You can substitute it with anything you like. ”jdbc:mysql://MysqlServerAddress/EJBTEST” is used when establishing JDBC connections. Substitute ”MysqlServerAddress” with the actual location of Mysql. ”EJBTEST” is the name of the database to be used.

For detailed configuration, please refer to PKUAS User Manual.

Chapter 3. EJB Module Development

Developing EJB module on PKUAS fully conforms to the J2EE/EJB specification. You pack all the enterprise bean classes, home and remote interfaces, the helper classes and the deployment descriptors into a jar file as the release unit. For more details, there are a lot of books and documents about J2EE/EJB development you can refer to. Especially, here are some issues you must pay attention to.

- Each EJB JAR archive comprises of one EJB at least. single JAR or enterprise archive (EAR for short) made of several jars can be deployed onto PKUAS. The pkuas-ejb.xml providing necessary deployment information is mandatory for the JAR to be PKUAS compatible. You can get more explanation in the following section 5.3.2.
- As for developing Container-Managed Persistence Entity Bean (CMP for short), you can refer to chapter 7.2.5.

Chapter 4. WEB Module Development

PKUAS integrates Tomcat as its Servlet engine (current integrated Tomcat version is 4.1.31). The package structure of a PKUAS web module (WAR) is basically the same as a stand-alone Tomcat web application, with the addition of a pkuas-web.xml deployment descriptor file. Figure 4.1 shows the package structure of a typical PKUAS web module.

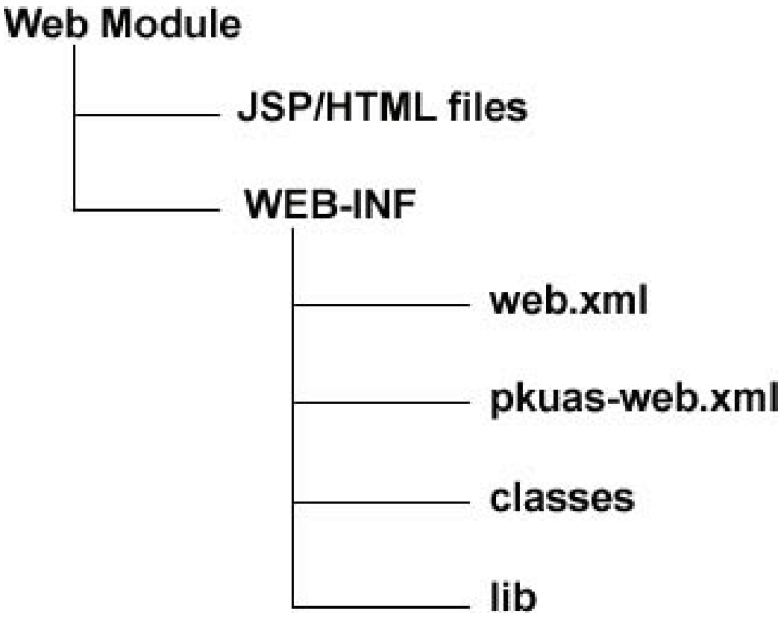

Figure 4.1 PKUAS web module directory structure

Web modules can be deployed in one of two ways:

- As a stand-alone web application:
  - Copy the root directory of the web module to the application directory of the PKUAS directory tree;
  - Or pack the web module into a WAR file (using the 'jar' utility) and put the WAR file into the 'application' directory of the PKUAS directory tree.

Additionally, if the web module is dependent upon any EJB modules, then the neccessary EJB's interface class files as well as the stubs generated by PKUAS will have to be copied to the web module's 'classes' directory or packaged into 'jar' files under the 'lib' directory. This step is neccessary in order to allow the web module's runtime ClassLoader to find the interface and stub class files.

- As a part of an enterprise application:
  - Pack the web module into a WAR file and pack the WAR file into the final EAR file of the enterprise application. This will be discussed in further detail in the next chapter. 5

Chapter 5. Application Assembly and Deployment

Application Assembly is the process of combining EJB module and WEB module, and configuring it as a complete application system; Application Deployment is the process of putting the application system which has been assembled into its running environment and making any necessary configuration to ensure the runnable of the application system. Naturally saying, assembly and deployment are two separated processes, but they always work together in actual application development, as there is no obvious boundary between them. So we treat them as one whole thing

In addition, we could consider an EJB module as an application which has only one EJB module when deploying EJB modules on PKUAS directly. So we consider EJB module as a special application, introduce it as a common enterprise application.

5.1 Necessary modification to EJB module's description file

We may make necessary configuration to the packed EJB module, which is modifying or adding something in ejb-jar.xml, during application assembly and deployment. For example, we add the following lines in ejb-jar.xml of EJB Order: 5.1:

The lines added above are mostly about transaction and security which will be introduced in detail in section 7.1 and section 7.2.

After modifying ejb-jar.xml, we should pack the EJB again to update this file in the package, to ensure it is the modified EJB that in the latest deployment.

5.2 Necessary modification to Web module's description file

In the same way, we should make some modification and addition to the description file of the Web module during application assembly and deployment. In PKUAS, the main description file of the Web module is web.xml which is defined in the J2EE and JSP/Servlet specification; In addition, there is some information specific to PKUAS about Web module in pkuas-web.xml, which will be introduced later.

```
<enterprise-beans>
    ......
    <security-role-ref>
        <description> test purpose </description>
        <role-name> admin </role-name>
        <role-link> administrator </role-link>
    </security-role-ref>
    ......
</enterprise-beans>
<assembly-descriptor>
    <security-role>
        <description> test purpose </description>
        <role-name> administrator </role-name>
    </security-role>
    <method-permission>
        <role-name>administrator</role-name>
        <method>
            <ejb-name>Order</ejb-name>
            <method-name>findAllOrders</method-name>
        </method>
    </method-permission>
    <container-transaction>
        <method>
            <ejb-name>Order</ejb-name>
            <method-name>create</method-name>
        </method>
        <trans-attribute>Required</trans-attribute>
    </container-transaction>
</assembly-descriptor>
```

Figure 5.1 Security Configuration Sample

5.2.1 web.xml

There is only a few complementarities should be done in the file, as the majority of web.xml has been done in Web module's developing. $< ejb-ref >$element is used in web.xml to describe the information of EJB which is referenced, for example: 5.2:

Here, we describe an EJB named CartEJB, which is a Session Bean, whose Home Interface is

com.sun.j2ee.blueprints.shoppingcart.cart.ejb.ShoppingCartHome
and Remote Interface is

com.sun.j2ee.blueprints.shoppingcart.cart.ejb.ShoppingCart.

The name of referenced EJB is CartEJB. In actual developing, the referenced EJB

```
<ejb-ref>
    <ejb-ref-name>ejb/cart/Cart</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>
      com.sun.j2ee.blueprints.shoppingcart.cart.ejb.ShoppingCartHome
    </home>
    <remote>
      com.sun.j2ee.blueprints.shoppingcart.cart.ejb.ShoppingCart
    </remote>
    <ejb-link>CartEJB</ejb-link>
</ejb-ref>
```

Figure 5.2 EJB Ref Sample

may be defined in another file; Generally speaking, reference is given as the form of the jar pack's name plus EJB's name, which is separated by #. For example: 5.3:

```
    <ejb-link>EmployeeRecord</ejb-link>
    <ejb-link>../products/product.jar\#ProductEJB</ejb-link>
```

Figure 5.3 EJB Reference Sample

5.3 Writing Deployment Descriptors

Deployment descriptor contains the application's configurable information. The application deployed in PKUAS needs related deployment descriptor.

- application.xml：is defined by J2EE Specification, describes the application's architecture.
- pkuas-ejb.xml：is a particular descriptor needed by PKUAS, used to describe application's detailed information.
- pkuas-web.xml：is a particular descriptor needed by PKUAS, used to describe web application's detailed information.
- ejb-jar.xml：is defined by J2EE specification, describes EJB module's deployment information.

The four deployment descriptors will be introduced in the following chapter.

5.3.1 application-xml

Application.xml describes the static information of application itself and its modules. It mainly contains:

- Application's information
  - Application's name$< display - name >$
  - Application's description$< description >$
- EJB module's information
  Every EJB module can contain more than one EJB, here we just introduce the physical file (*.jar) related with EJB module.
- Web module's information
  Every web module contains an application which can be accessed through web, here we introduce the physical file (*.war) and the path when the application starts up (the information in URL).

Next figure illustrates an application.xml belongs to an application named eshop.
5.4

5.3.2 pkuas-ejb.xml

Pkuas-ejb.xml and ejb-jar.xml must be in the META-INF subdirectory of the jar file, they describe the information related with service vendor PKUAS. Some content of pkuas-ejb.xml is the same as ejb-jar.xml, it also contains the EJB architecture and assembly information in ejb-jar.xml; meanwhile, it can bind JNDI name with EJB, provide security role mapping and so on.

The architecture of whole xml file information is:

- $< pkuas >$: The only outermost element of the xml file, other elements are its sub-elements.
- $< enterprise - beans >$: It describes the deployment information of every EJB, contains EJB foundational deployment information and outer dependency of EJB: the usage of other EJB, data source, environment variable.

```
<?xml version="1.0" encoding="ISO8859_1"?>
<application>
  <display-name>eshop</display-name>
  <description>Application description</description>
  <module>
    <ejb>Customer.jar</ejb>
  </module>
  <module>
    <ejb>Order.jar</ejb>
  </module>
  <module>
    <ejb>OrderLineItem.jar</ejb>
  </module>
  <module>
    <ejb>Product.jar</ejb>
  </module>
  <module>
    <ejb>ShoppingCart.jar</ejb>
  </module>
  <module>
    <ejb>ShoppingCartLineItem.jar</ejb>
  </module>
  <module>
    <web>
      <web-uri>eshop.war</web-uri>
      <context-root>eshop</context-root>
    </web>
  </module>
</application>
```

Figure 5.4 application.xml Sample

- $< security - realm >$：The information of description about application security.
- $< utility >$：The information of application's stub/skeleton package.
- $< evolution >$：Whether or not support online evolvement.

Next we'll show particular description in every part:

1．enterprise-beans

- enterprise-beans：Contains one or more $< ejb >$, each $< ejb >$ contains:
  - ejb-name：EJB name which is an identity in CMP. In an application, two EJB with the same ejb-name is banned in PKUAS.

- ejb-type：The type of EJB, it's only can be Session, Entity or MessageDriven.
- jndi-name/local-jndi-name：The name bound with EJB in JNDI, map to remote EJB and local EJB. If an EJB offers remote and local interface, you should give both of them.
- clustered：Whether it uses cluster or not, the value only can be true or false.
- resource-ref：Resource reference, declares manageable resource: data source, JavaMail Session, class factory and so on, and describes the outer resource information used by this EJB.
  - res-ref-name：Resource name used in source code.
  - jndi-name：Resource's JNDI name used in actual environment.
  - res-type：The type of resource, its value only can be one of those:
    - javax.sql.DataSource
    - javax.jms.QueueConnectionFactory
    - javax.jms.TopicConnectionFactory
    - javax.mail.Session
  - principal-name：User name which used to access the resource.
  - principal-password：Password which used to access the resource.

  Please refer to the api of the JavaMail about the configuration of javamail.
- resource-env-ref：Resource environment reference, used to simplify the configuration of resource environment without authentication in Servlet 2.4. It declares the manageable object related with resource, for example, environment variable, resource-ref variable. Resource environment reference offers a way to search and access manageable object with JNDI.
  - resource-env-ref-name：The name of environment source used in source code.
  - resource-env-ref-type：The type of resource offered by environment,

it can be class or interface defined in JAVA language.

* jndi-name：The JNDI name of resource used in actual environment.

– ejb-ref and ejb-local-ref：Other EJB's information used or depended on by this EJB, the difference between them is whether referenced EJB is remote or local.

* ejb-ref-name：The name used in source code.

* jndi-name：The EJB's JNDI name in actual deployment.

– About Message Driven Bean:

* max-message：This attribute is about Message Driven Bean consuming JMS message. Its value generally is between 10 and 20.

* mdb-user：The user name used in accessing Message Driven Bean.

* mdb-password：The password used in accessing Message Driven Bean.

* mdb-client-id：The client Id in accessing Message Driven Bean.

Those tags are just needed when you describing a Message Driven Bean.

The description information of EJB ShoppingCart in ShoppingCart.jar: 5.5

The description information of EJB Product in Product.jar: 5.6

The description information about a Message Driven Bean: 5.7

2．Security-realm：The application in each jar package has its own security domain information. Deployed in PKUAS, the application can use authentication and authorization offered by PKUAS; when an application is deployed, only the method whose access need be controlled and the mapping between user and role should be announced. Defined in ejb-jar.xml which contains its EJB, the security definition of methods in EJB announces the security roles that can call these methods. The element $< security - realm >$ in pkuas-ejb.xml defines security role used by application and the mapping between physical user and security role.

• realm-name：The security domain used by application, using JNDI name. The

```
<?xml version="1.0" encoding="UTF-8"?> <pkuas>
    <enterprise-beans>
        <ejb>
            <ejb-name>cart</ejb-name>
            <ejb-type>Session</ejb-type>
            <jndi-name>CartHome</jndi-name>
            <ejb-ref>
                <ejb-ref-name>ejb/Order</ejb-ref-name>
                <jndi-name>OrderHome</jndi-name>
            </ejb-ref>
            <ejb-ref>
                <ejb-ref-name>ejb/OrderLineItem</ejb-ref-name>
                <jndi-name>OrderLineItemHome</jndi-name>
            </ejb-ref>
        </ejb>
    </enterprise-beans>
</pkuas>
```

Figure 5.5 Session EJB ShoppingCart

```
<?xml version="1.0" encoding="ISO8859_1"?> <pkuas>
    <enterprise-beans>
        <ejb>
            <ejb-name>product</ejb-name>
            <ejb-type>Entity</ejb-type>
            <jndi-name>ProductHome</jndi-name>
            <resource-ref>
                <res-ref-name>jdbc/ejbPool</res-ref-name>
                <jndi-name>jdbc/Product</jndi-name>
                <res-type>javax.sql.DataSource</res-type>
                <principal-name>root</principal-name>
                <principal-password />
            </resource-ref>
        </ejb>
    </enterprise-beans>
</pkuas>
```

Figure 5.6 Entity EJB Product

application that contains EJB and Web use the same security domain which is asked to be exclusion.

- mapping：The mapping from user name to role name.
  - user-name：Physical user name.
  - role-name：The role that user map in this application.

```
<?xml version="1.0" encoding="UTF-8"?> <pkuas>
    <enterprise-beans>
        <ejb>
            <ejb-name>Schedule</ejb-name>
            <ejb-type>MessageDriven</ejb-type>
            <max-message>20</max-message>
            <destination-jndi-name>
                jms/ScheduleQueue
            </destination-jndi-name>
            <ejb-local-ref>
                <ejb-ref-name>entiry/assign</ejb-ref-name>
                <ejb-ref-type>Entity</ejb-ref-type>
                <jndi-name>hr/spoffice</jndi-name>
            </ejb-local-ref>
        </ejb>
    </enterprise-beans>
</pkuas>
```

Figure 5.7 Message Driven EJB

The relationship between physical user and role is many-many mapping. It means that a user can have more than one role at a time; more than one user also can have the same role. Users are defined in PKUAS security service, so application can not define user only for its own using.

The security domain information of ShoppingCart.jar package in eshop: 5.8

```
<security-realm>
    <realm-name>jaas:/eshop</realm-name>
    <mapping>
        <user-name>alice</user-name>
        <role>manager</role>
    </mapping>
    <mapping>
        <user-name>alice</user-name>
        <role>administrator</role>
    </mapping>
</security-realm>
```

Figure 5.8 Security Domain Information

3．utility

- module-name：The name of jar package file which contains stub and skeleton.

- If pkuas-ejb.xml does not contain $< utility >$, when deploying this application, PKUAS will generate stub/skeleton automatically.

4．evolution：Used to point out whether support online evolution or not, it can be true (support) or false (nonsupport). The default value is nonsupport.

5.3.3 pkuas-web.xml

Pkuas-web.xml can be found in every war package, describing some information of the web application deployed in PKUAS, mostly about outer resource reference. The information architecture of the whole xml file:

pkuas-web: The only outermost element of the xml file, other elements are its sub-elements.

- resource-ref : The same as pkuas-ejb.xml.
- resource-env-ref : The same as pkuas-ejb.xml.
- login-config：Used by single-sign-on.
  - auth-method
  - form-longin-config
    - form-login-page：The page announced for user login.
    - form-error-page：The page announced for failure of authentication.

5.3.4 ejb-jar.xml

Ejb-jar.xml is a most important component of ejb-jar file defined in EJB specification. It is a xml format deployment descriptor, offers all EJB architecture and application assembly information which is optional. Please refer to EJB reference for the detail content.

5.4 Application Packing

Packing is the process of putting the EJB, Web module and the deployment description file into one file which could run in PKUAS. So far, PKUAS supports three types

of application: EJB Application (which contains EJB only and deployed in the JAR form), Web Application(which contains Web pages JSP/Servlet and JavaBean, etc.) and Enterprise Application(which contains EJB and Web module). We shall discuss the packing process of these three types of application separately.

1. The process of packing Enterprise Application: enterprise application is deployed on PKUAS in the EAR form. A complete EAR file should contain:

- A JAR file with one or more EJBs: every JAR is an EJB module, it could contain one or more EJBs, which consists of class files, description file, pkuas-ejb.xml and ejb-jar.xml.
- A WAR file of one or more Webs: every WAR is a Web module, which can be accessed through browser independently.
- Assembly and deployment description file: the description and configuration information of the whole application consists of application.xml (defined in J2EE specification), which is in the META-INF directory.

The structure of an EAR is shown in the figure below: 5.9：

The process of application packing is shown as below:

- Create a new directory to put the file before packing. Suppose the file name is eshop.
- Put the EJB andWeb module which is to be packed into EAR file in the directory created above.
- Create a new directory named META-INF in eshop directory, warning: case sensitive.
- Put application.xml in eshop/META-INF directory.
- Packing with the jar command: in command prompt, enter the eshop directory and execute command:
  jar cvf eshop.ear *.*.
  This means packing the whole application into eshop.ear.

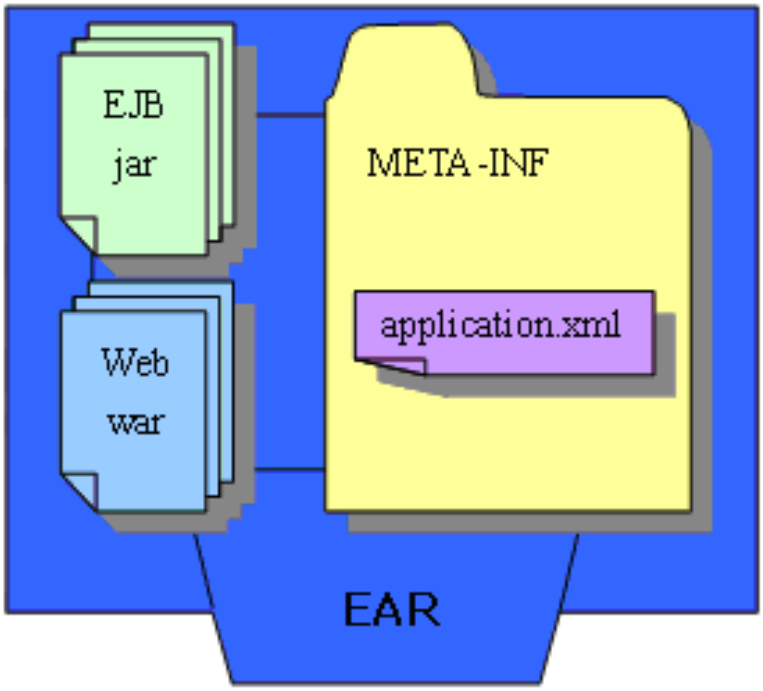

Figure 5.9 The structure of EAR

Application packed could be deploy into PKUAS.

2.The process of packing EJB application: EJB application is deployed on PKUAS in the JAR form. A complete JAR file should contain:

- One or more EJBs: every JAR is an EJB module, contains one or more EJBs, JAR contains the class files which EJB needed.
- Assembly and deployment description file: the ejb-jar.xml which is the description file of EJB module and pkuas-ejb.xml which contains the description and configuration information specific to PKUAS about the whole application should put in the directory META-INF.

The structure of a JAR is shown in the figure below: 5.10：

3. The process of packing EJB application: similar to the EAR one.

5.5 Application Deployment

The application will be deployed should be packed and put in PKUAS application deployment directory. It means you should put the .ear file in the directory named

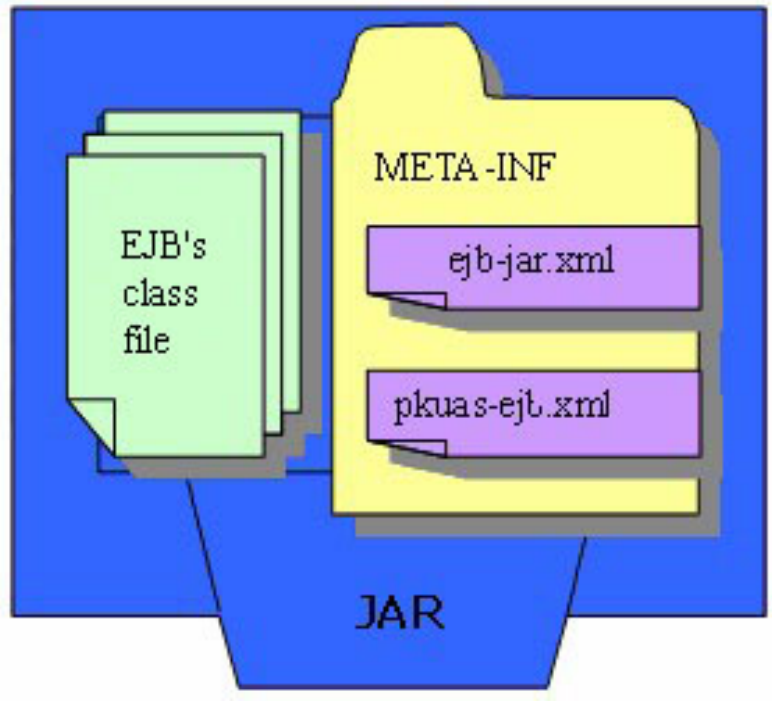

Figure 5.10 The structure of JAR

applications in the PKUAS installment directory. When the PKUAS starts up next time, this application can be start up automatically.

Chapter 6. Application Boot

Currently, applications deployed on PKUAS will be started when the Application Server boots. Applications can also be started by using the PKUAS management console. If the application contains Message Driven EJBs, then you must specify the message destinations this application requires in the server configuration file `services.xml`. The Application Server will then create these destinations when it boots, and also binds them to the names specified in the configuration file in the naming system. If the application contains Entity EJBs, the data sources it uses must be specified in the `services.xml` configuration file. Details of how this can be done can be found in the relevant sections in the PKUAS User Guide.

PKUAS also provides the mechanism for auto application deployment. The server will monitor the `application` directory, if it discovers new applications, or that existing applications have been changed, it will automatically deploy or redeploy the applications.

Chapter 7. Advance

7.1 Transcation

When developing the EJB component, the developer can use the transaction service provided by the application server to improve the running reliability of components. It makes the operation which uses the transaction service ACID properties. According to the J2EE specification, there are two ways to access the transaction: one is the programmatic way; the other is the declarative way. You can decide which way according to your actual developing requirement. Refer to the relative chapters of Mastering EJBII to get the concrete principals.

7.1.1 Declarative

Session Bean, Entity Bean and MessageDriven Bean can all use the declarative way to visit transaction service. The component developer only need to describe the requirement in every deployment descriptor of the component and do not need to care about the concrete codes of the application server. For example, there is a component named Order (Order is an entiry bean) in eshop application. The developer hopes the create() method of the Order component to run in the transaction environment. That is, start transaction before the execution of create() method and close transaction after the completion of create() method. So the support of transaction service ensures the reliability and consistency of the database operations in the create method. The developer can do it by the ejb-jar.xml of the EJB Order:

In the assembly-descriptor figure 7.1 of ejb-jar.xml file, you can set the create() method of Order EJB with the Required transaction attribute. There are six different transaction attributes: Required,RequiresNew,Supports,Mandatory,NotSupported and Never. For every transaction attribute, application server will take different strategies to satisfy the requirements of users. See figure 7.2:

From the figure 7.2 you can see that the meaning of Required attribute is if the client

```
<assembly-descriptor>
      <method-permission>

      </method-permission>
      <container-transaction>
            <method>
                  <ejb-name>Order</ejb-name>
                  <method-name>create</method-name>
             </method>
             <trans-attribute>Required</trans-attribute>
      </container-transaction>
</assembly-descriptor>
```

Figure 7.1 Command

TRANSACTION ATTRIBUTE	CLIENT'S TRANSACTION	BEAN'S TRANSACTION
Required	none	T2
	T1	T1
RequiresNew	none	T2
	T1	T2
Supports	none	none
	T1	T1
Mandatory	none	error
	T1	T1
NotSupported	none	none
	T1	none
Never	none	none
	T1	error

Figure 7.2

does not start a transaction, then the server will start a new transaction before calling the method of the bean; if the client does start a transaction, then the transaction will be propagated to the callee and the method of the bean will execute in the client transaction context. You can get the meaning of other transaction attributes from the table. Client can set the proper transaction attributes in the corresponding ejb-jar.xml file according to their requirements. The declarative transaction in deployment descriptor supports the declaration of asterisk character and the same method name with different parameters. You can refer to the relative chapters in Mastering EJBII and EJB specification for details.

7.1.2 Progmmatic

The Standalone client, Web client, Session Bean and MessageDriven Bean can also use the programmatic way to access transaction service. Client can get an implementation of the JTA interface javax.transaction.UserTransaction by string "java:comp/UserTransaction" with the help of naming service. Then client can call begin() method of UserTransaction to start a transaction, commit() method to commit a transaction and rollback() method to rollback a transaction. There are also other methods which can be used to visit and control a transaction. The workflow in Enterprise bean is shown in figure 7.3:

```
Context ctx = new InitialContext(); UserTransaction utx=
    (UserTransaction)ctx.lookup("java:comp/UserTransaction");
utx.begin(); Connection conn = getConnection();
PreparedStatement ps =
    conn.prepareStatement("insert into accounts values(?,?,?)");
ps.setString(1,"123"); ps.setString(2,"456"); ps.setDouble(3,789);
ps.executeUpdate();
utx.commit();
```

Figure 7.3 Procedure

As to Session Bean and MessageDriven Bean, you can use EJB context to get the UserTransacton interface through the naming service. See figure 7.4：

```
protected SessionContext ctx;

UserTransaction utx = this.ctx.getUserTransaction();

utx.begin();
//do something
utx.commit();
```

Figure 7.4 Getting Interface Sample

7.2 Security

7.2.1 Security Mechanic

EJB Specification defines integrated security mechanism in order to control users' access to every EJB method. EJB Specification does not agree that EJB developer hard-codes security control into the EJB programs, but encourages you to declare the security access permission in the deployment descriptor according to the specification, so that EJB can be flexibly deployed to all kinds of operation environment. In order to make the deployers' work easy, application assembler can define security roles. A security role is a collection of access permission seted for assumed users. Application assembler can define every EJB method's access permission with security roles in detail. When deployers deploy EJB, they only need to simply map the users or groups to the security roles.

7.2.2 Define Security Role

$< security - role >$ define the security roles

$< role - name >$name the roles

$< description >$describe the roles (optional)

The defined security role is effective in the whole ejb-jar file, i.e. all the EJB in the same ejb-jar file can use these security roles.

For example7.5：ejb-jar.xml

7.2.3 Method Permission

After defining security roles, you can designate the methods that can be invoked by the relevant roles with $< method - permission >$ element.

$< method - permission >$contains the elements below:

- $< role - name >$： the role's name
- $< method >$： the method authorized to call

```
...
<assembly-descriptor>
   <security-role>
      <description>
        This role is defines right of administrator
      </description>
      <role-name>admin</role-name>
   </security-role>
   <security-role>
      <description>
        This role defines rights of normal employee
      </description>
      <role-name>employee</role-name>
   </security-role>
    ...
</assembly-descriptor>
```

Figure 7.5 ejb-jar.xml Sample

- $< ejb - name >$：the component where the method is
- $< method - name > * < /method - name >$：the method's name

For example7.6：

7.2.4 Define User

The roles defined in the third section need to be maped to the server's users, so first we must create users in the server. Note that PKUAS is different from the most of servers. It does not have the concept of group. This is mainly due to this consideration: security roles themselves are abstract user group, a security role can be maped to more than one user. For the most of applications, it can meet the requirement by delegating the roles to the users. PKUAS implements user authentication by JAAS mechanism. At present, the default login module provided by the server is pku.as.securety.login.UsersRolesLoginModule, which authenticates user name and password. PKUAS provides a command line to create the users. It can implement user management through run useradmin.bat (Windows Platform) or useradmin.sh（Linux/Unix/Solaris Platform）which are under PKUAS's installation directory.

```
...
 <method-permission>
  <role-name>employee</role-name>
  <method>
    <ejb-name>ShoppingCart</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>
<method-permission>
  <role-name>employee</role-name>
  <method>
    <ejb-name>ShoppingCartLineItem</ejb-name>
    <method-name>findByPrimaryKey</method-name>
  </method>
<method-permission>
  <role-name>admin</role-name>
  <method>
    <ejb-name>Product</ejb-name>
    <method-name>*</method-name>
  </method>
</method-permission>
```

Figure 7.6 ejb-jar.xml Sample

7.2.5 Assign Roles to Users

After creating the roles and the users, in order to map different roles to relevant users, you must add $< security - realm >$ element in application.xml.

$< security - realm >$contains the elements below:

- $< realm - name >$： the name of the security realm
- $< mapping >$： map the role to the user

For example7.7：

```
...
 <security-realm>
  <realm-name>jaas:/eshop</realm-name>
  <mapping>
    <user-name>alice</user-name>
    <role>manager</role>
  </mapping>
  <mapping>
    <user-name>alice</user-name>
    <role>administrator</role>
  </mapping>
  <mapping>
    <user-name>admin</user-name>
    <role>administrator</role>
  </mapping>
  <mapping>
    <user-name>pkuas</user-name>
    <role>tomcat</role>
  </mapping>
</security-realm>
...
```

Figure 7.7 application.xml Security Domain

CMP Application Development Document

PKUAS 2005 CMP tools process the entity EJB compliant with CMP 2.0 specification, not other types of EJB.

It supports both remote and local interface of entity bean, but you need to pay attention that the relationship between entity EJBs must be constructed on local interface.

Until now,all data types CMP field supported contian:

java.lang.Integer，

float，

java.lang.Float，

double，

java.lang.Double，

short，

java.lang.Short，

byte，

java.lang.Byte，

long，

java.lang.Long，

boolean，

java.lang.Boolean，

java.util.String，

java.sql.Date，

java.util.Date，

java.sql.Timestamp，

java.math.BigDecimal，

java.lang.Object,

Others are not supported until now;

Both relation table and foreign table are supported in object-relation mapping process, but many-to-many relationship can only be represented by relation table.

Database products supported already includes:

Oracle（9i or higher），（already passed JPS1.3，JONAS test）

MS SQLSERVER（2000 or higher），

MySQL（4.0.15 or higher）；

MS SQLSERVER and MySQL have not passed the strict test，only pass JPS1.3 test。

.1 CMP abstract persistence schema

Refer to CMP part of EJB 2.0 specification;

.2 CMP Deployment Descriptor

Every entity bean needs to write deployment description ejb-jar.xml which will be based on specification, this file contains the information of the entity bean deployed, including: EJB name, class name of local home interface/local interface /entity bean, EJB type, primary key; most importance, the abstract persistence schema part of EJB deployment description, which contains CMP field and CMR field description of the entity EJB. Refer to EJB 2.0 specification for more about ejb-jar.xml.

- display-name：the name which EJB displys
- ejb-name：the name of EJB
- local-home：the fully-qualified name of the enterprise bean’s local home interface.
- local：EJB local interface class
- remote：EJB remote interface class
- home：EJB remote home interface class
- ejb-class：EJB Bean class
- persistence-type：persistence type（must be Container for CMP）

- cmp-version：CMP version，must be 2.x (default)
- prim-key-class：EJB primary key class
- reentrant：EJB reentrant
- cmp-field：CMP EJB CMP field
- query：query method description
- method-name：method name
- method-params：method parameters
- ejb-ql：query method
- abstract-schema-name：bean's abstract schema name for EJBQL
- relationships: relationships
- ejb-relation-name：the name of EJB relation
- ejb-relationship-role-name：the name of EJB relationship role
- multiplicity：multiplicity of the relationship of EJB instances, "One" or "Many"
- ejb-name：source EJB name of the relationship
- cmr-field-name：CMR domain in relationship，corresponding to EJB
- cascade-delete：cascade delete or not

$< relationships >$ describe more than one $< ejb - relation >$. $< ejb - relationship - role >$ represents the direction of a relationship, because a relationship can only be a 2-side relationship at most, a $< ejb - relation >$ can have two roles at most, described by the element $< ejb - relationship - role >$. $< ejb - relationship - role >$ points out the source EJB name of relationship, CMR domain of target EJB and multiplicity of the relationship, shown in figure7.5

Every EJB application containing a CMP bean needs a deployment description named "database.xml".

database.xml described the mapping between CMP entity EJB and the persistence information of database table in the jar package.

Attribute name is an optional database element,which can be ”oracle”(default)、”mysql”、”sqlserver”.

The information of every EJB is recorded in a corresponding module element of the deployment description, including:

- ejb-name：the name of CMP entity EJB
- ejb-table-name：the name of corresponding EJB table
- refname: the name of EJB referenced database
- reftype：EJB referenced database type
- cmp-entity-bean-field-name：the name of cmp entity bean field
- db-field-name：database field name that EJB CMP domain mapped to
- relation-tables：relation tables information
- relation-name：relation name（must corresponding to one of EJB-relation-name in ejb-jar.xml）
- relation-target-ejb-name：the name of relation target EJB
- relation-table-name：the name of relation table(foreign key table)
- relationships: relationships between entity EJB
- ejb-relation：ejb relation
- ejb-relation-name：the name of EJB relation name(must corresponded to one of ejb-relation-name in ejb-jar.xml)
- ejb-relationship-role：EJB relationship role
- ejb-relationship-role-name：the name of EJB relationship role(must corresponded to one of ejb-relationship-role-name in ejb-jar.xml)
- key-fields：key fields mapping between entity EJBs in relationship

- field-name：the name of entity EJB field
- column-name：column name mapped by field-name

Database.xml sample can refer to figure10.

when recording the key mapping in the $< key-fields >$ of the $< ejb-relation >$, you need notice that, the 2 roles of the relationship must map their own primary key to the corresponding relation table field. They can not be null. That is to say，the $< key - fields >$ can't be null in the circs of relation table.

The above example is about the relation table, when it comes to foreign key table, database.xml needs some modifications:

1. It is the name of foreign key table at the relation-table-name label , not the relation table name, that is to say, the database table of the entity EJB in relation is chosen to be foreign key table to record entity EJB relation, the principle is that : When in a one-to-one relation, you can choose either; choose the "many" one when in a one-to-many relation.

2. relation-table must be wrote in this way: $< relation - tabletype = "FK" FKTable = "Source"(or"Target") >$ "type" attribute must be "FK" to indicate it's a foreign key table. If the table of entity EJB is used as foreign key table, "FKTable" attribute is "Source", "Target" otherwise. If both beans in relation is the same entity EJB, the table is the foreign key table, it is "Source".

3.In $< ejb - relation >$, when mapping two roles of primary key foreign key field in $< key - fields >$, the entity EJB acting as foreign key table, $< key - field >$ can be null, but the other needs to map the primary key of the entity EJB acting as foreign key table to the corresponding field in the foreign key table.

4. if primary key domain and primary key class is not defined in ejb-jar.xml, they will be generated automatically by the system, meanwhile, if primary key of the entity EJB needs to be mapped to foreign key table, $< field - name >$ in $< key - fields >$ must be pk+bean name, for example "pkAccount", Account is the entity bean name.

Database.xml foreign key sample can refer to figure 12.

You need to write deployment description file pkuas-ejb.xml and application.xml for ejb application, refer to PKUAS development document.

All these files can be generated by PKUAS deploy tools.

.3 Package

In PKUAS, application is packed twice, first enery EJB is packed as a JAR file, then all EJB JAR package and the deployment description file packed as a bigger JAR (or EAR) package. Entity EJB JAR package includes: the entity EJB class(two interface, one EJB class and utility class(optional)) and a EJB deployment description file ejb-jar.xml, and database.xml(in META-INF). Others description files ,like pkuas-ejb.xml, will not give unnecessary details.

The structure of single EJB JAR package can refer to figure 14：

The structure of entire EJB application JAR (or EAR) package: refer to figure15

The instruction of detail pack procedure can refer to PKUAS deployment management tools. We only deal with CMP part here. The package can be deployed on PKUAS directly.

.4 Create Database Table

PKUAS can generate table for application automatically, if the table is not existed while developing, PKUAS can be used to generate script to create tables.

For example, after a CMP application named ”XXX” is successfully deployed, in repository/work/localhost directory of PKUAS directory, there will be a file named XXX_SQL.txt. It contains the script to create table in database. After running it at corresponding database client, the tables will be created.

```
<?xml version="1.0"  encoding="UTF-8"?> <ejb-jar>
<enterprise-beans>
    <entity>
        <display-name>Order</display-name>
        <ejb-name>Order</ejb-name>
        <local-home>myentity.Order.OrderHome</local-home>
        <local>myentity.Order.Order</local>
        <ejb-class>myentity.Order.OrderBean</ejb-class>
        <persistence-type>Container</persistence-type>
        <prim-key-class>java.lang.Integer</prim-key-class>
        <primkey-field>orderId</primkey-field>
        <reentrant>False</reentrant>
        <cmp-field><field-name>orderId</field-name></cmp-field>
        <cmp-field><field-name>orderValue</field-name></cmp-field>
        <query>
            <query-method>
                <method-name>ejbFindOrderIDBetween</method-name>
                <method-params>
                <method-param>int</method-param>
                <method-param>int</method-param>
                </method-params>
                </query-method>
            <ejb-ql>
                select orID from orderdb where orID &gt; ? and orID &lt; ?
            </ejb-ql>
        </query>
        <resource-ref>
            <res-ref-name>jdbc/ejbPool</res-ref-name>
            <res-type>javax.sql.DataSource</res-type>
            <res-auth>Container</res-auth>
        </resource-ref>
    </entity>
    <entity>
        <display-name> LineItem </display-name>
        <ejb-name> LineItem </ejb-name>
        <local-home>myentity.Order. LineItemHome </local-home>
        <local>myentity.Order. LineItem </local>
        <ejb-class>myentity.Order. LineItemBean</ejb-class>
        <persistence-type>Container</persistence-type>
        <prim-key-class>java.lang.Integer</prim-key-class>
        <primkey-field> lineItemId </primkey-field>
        <reentrant>False</reentrant>
        <cmp-field><field-name>lineItemId </field-name></cmp-field>
        <cmp-field><field-name> productId</field-name></cmp-field>
    </entity>
</enterprise-beans>
```

(Cont.)9

Figure 8 Deployment Descriptor Sample

(Cont.)8

```
<relationships>
    <ejb-relation>
        <ejb-relation-name>Order- LineItem</ejb-relation-name>
        <ejb-relationship-role>
            <ejb-relationship-role-name>
                order-has-lineItems
            </ejb-relationship-role-name>
            <multiplicity>One</multiplicity>
            <relationship-role-source>
                <ejb-name> Order </ejb-name>
            </relationship-role-source>
        </ejb-relationship-role>
        <ejb-relationship-role>
            <ejb-relationship-role-name>
                lineItems -of-order
            </ejb-relationship-role-name>
            <multiplicity>Many</multiplicity>
            <relationship-role-source>
                <ejb-name> LineItem </ejb-name>
            </relationship-role-source>
            <cmr-field>
                <cmr-field-name>order</cmr-field-name>
            </cmr-field>
        </ejb-relationship-role>
    </ejb-relation>
</relationships>
 </ejb-jar>
```

Figure 9 Deployment Descriptor Sample

```
<database  name = "oracle">
    <module refname="jdbc/ejbPool" reftype="javax.sql.DataSource">
        <ejb-name> Order </ejb-name>
        <ejb-table-name> OrderTB</ejb-table-name>
        <maps>
            <map>
                <cmp-entity-bean-field-name>
                    orderID
                </cmp-entity-bean-field-name>
                <db-field-name> orID </db-field-name>
            </map>
            <map>
                <cmp-entity-bean-field-name>
                    orderValue
                </cmp-entity-bean-field-name>
                <db-field-name> orValue </db-field-name>
            </map>
            <relation-tables>
                <relation-table>
                    <relation-name>
                        Order- LineItem
                    </relation-name>
                    <relation-target-ejb-name>
                        LineItem
                    </relation-target-ejb-name>
                    <relation-table-name>
                        relationtb1
                    </relation-table-name>
                </relation-table>
            </relation-tables>
        </maps>
    </module>
    <module refname="jdbc/ejbPool" reftype="javax.sql.DataSource">
        <ejb-name> LineItem </ejb-name>
        <ejb-table-name> LineItemTB</ejb-table-name>
        <maps>
            <map>
                <cmp-entity-bean-field-name>
                    lineItemId
                </cmp-entity-bean-field-name>
                <db-field-name> liId </db-field-name>
            </map>
            <map>
                <cmp-entity-bean-field-name>
                    productId
                </cmp-entity-bean-field-name>
                <db-field-name> proId </db-field-name>
            </map>
```

(Cont.)13

Figure 10 database.xml Sample

(Cont.)10

```
            <relation-tables>
                <relation-table>
                    <relation-name>
                        Order- LineItem
                    </relation-name>
                    <relation-target-ejb-name>
                        Order
                    </relation-target-ejb-name>
                    <relation-table-name>
                        relationtb1
                    </relation-table-name>
                 </relation-table>
            </relation-tables>
        </maps>
    </module>
    <relationships>
        <ejb-relation>
        <ejb-relation-name>
            Order- LineItem
        </ejb-relation-name>
        <ejb-relationship-role>
        <ejb-relationship-role-name>
            order-has- lineItems
        </ejb-relationship-role-name>
        <key-fields>
            <key-field>
                <field-name> orderId </field-name>
                <column-name> orderId </column-name>
            </key-field>
        </key-fields>
        </ejb-relationship-role>
        <ejb-relationship-role>
            <ejb-relationship-role-name>
                lineItems -of-order
            </ejb-relationship-role-name>
            <key-fields>
                <key-field>
                    <field-name> lineItemId </field-name>
                    <column-name> lineItemId </column-name>
                </key-field>
            </key-fields>
        </ejb-relationship-role>
        </ejb-relation>
    </relationships>
</database>
```

Figure 11 database.xml Sample

```
<database  name = "oracle">
<module refname="jdbc/ejbPool"
    reftype="javax.sql.DataSource">
    <ejb-name> Order </ejb-name>
    <ejb-table-name> OrderTB</ejb-table-name>
    <maps>
        <map>
            <cmp-entity-bean-field-name>
                orderID
            </cmp-entity-bean-field-name>
            <db-field-name> orID </db-field-name>
        </map>
        <map>
            <cmp-entity-bean-field-name>
                orderValue
            </cmp-entity-bean-field-name>
            <db-field-name> orValue </db-field-name>
        </map>
        <relation-tables>
            <relation-table type = "FK"   FKTable = " Target ">
            <relation-name> Order- LineItem </relation-name>
            <relation-target-ejb-name>
                LineItem
            </relation-target-ejb-name>
            <relation-table-name>
                LineItemTB
            </relation-table-name>
            ( LineItemTB is a foreign key table.)
            </relation-table>
        </relation-tables>
    </maps>
</module>
<module refname="jdbc/ejbPool" reftype="javax.sql.DataSource">
    <ejb-name> LineItem </ejb-name>
    <ejb-table-name> LineItemTB</ejb-table-name>
    <maps>
        <map>
           <cmp-entity-bean-field-name>
              lineItemId
           </cmp-entity-bean-field-name>
           <db-field-name> liId </db-field-name>
        </map>
        <map>
           <cmp-entity-bean-field-name>
              productId
           </cmp-entity-bean-field-name>
           <db-field-name> proId </db-field-name>
        </map>
```

(Cont.)13

Figure 12 database.xml foreign key table Sample

(Cont.)12

```
        <relation-tables>
           <relation-table type = "FK"   FKTable = " Source " >
                <relation-name> Order- LineItem </relation-name>
                <relation-target-ejb-name>
                    Order
                </relation-target-ejb-name>
                <relation-table-name>
                    LineItemTB
                </relation-table-name>
           ( LineItemTB is a foreign key table.)
           </relation-table>
        </relation-tables>
    </maps>
</module>
<relationships>
<ejb-relation>
    <ejb-relation-name>Order- LineItem</ejb-relation-name>
    <ejb-relationship-role>
        <ejb-relationship-role-name>
            order-has- lineItems
        </ejb-relationship-role-name>
        <key-fields>
            <key-field>
                <field-name> orderId </field-name>
                <column-name> orderId </column-name>
            </key-field>
        </key-fields>   (bean not used as foreign key table)
    </ejb-relationship-role>
    <ejb-relationship-role>
        <ejb-relationship-role-name>
            lineItems -of-order
        </ejb-relationship-role-name>
        <key-fields/>   (bean used as foreign key table)
    </ejb-relationship-role>
</ejb-relation>
</relationships>
</database>
```

Figure 13 database.xml foreign key table Sample

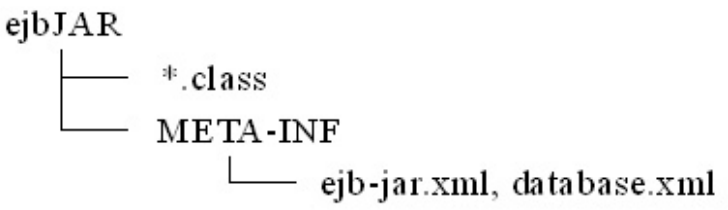

Figure 14 EJB JAR Structure

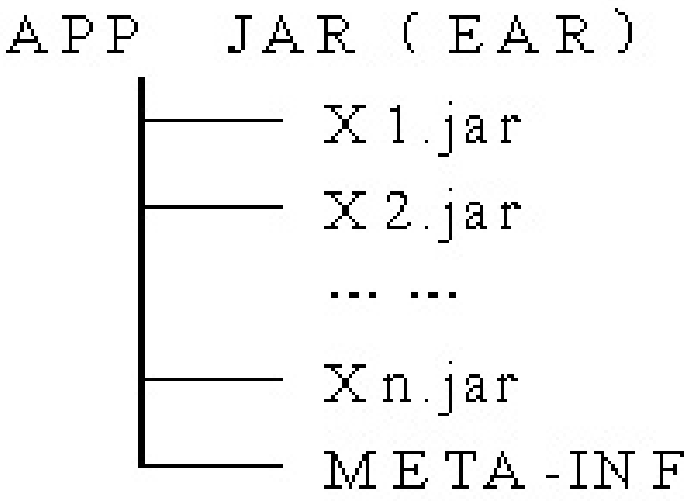

Figure 15 Application JAR（EAR）Structure

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document ”free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ”Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ”you”. You accept the license if you copy, modify or distribute the work in a way

requiring permission under copyright law.

A ”Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ”Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ”Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ”Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ”Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ”Transparent” is called ”Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ”Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ”Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section ”Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ”Acknowledgements”, ”Dedications”, ”Endorsements”, or ”History”.) To ”Preserve the Title” of such a section when you modify the Document means that it remains a section ”Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies.

If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. H. Include an unaltered copy of this License. I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title

of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section Entitled ”Endorsements”. Such a section may not be included in the Modified Version. N. Do not retitle any existing section to be Entitled ”Endorsements” or to conflict in title with any Invariant Section. O. Preserve any Warranty Disclaimers. If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled ”Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents,

unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ”History” in the various original documents, forming one section Entitled ”History”; likewise combine any sections Entitled ”Acknowledgements”, and any sections Entitled ”Dedications”. You must delete all sections Entitled ”Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ”aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Docu-

ment specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.