**Bài 7: Container-Managed Persistent Entity Beans**

**I. Khái niệm**

1. Vòng đời của CMP

*Class.newInstance()*

*setEntityContext()*

System Exception

EJB Container



Client

**DB**



**DB**

*ejbStore()*

*ejbHome<Name>(...)*

*ejbFind<Name>(...)*

***ejbSelect<Name>(...)***

*ejbCreate(...)*

*ejbPostCreate(...)*

*ejbActivate()*

*ejbLoad()*

*<method>(...)*

*<method>(...)*

*<method>(...)*

*Object.finalize()*

*unsetEntityContext()*

*ejbRemove()*

*ejbPassivate()*

**Activate** ejbActivate() → ejbLoad()

**Passivate** ejbStore() → ejbPassivate()

***ejbSelect<Name>(...)***

2. Chi tiết hoạt động của CMP

Client

Container

Bean Instance

Database

setSessionContext()

Thêm một thực thể mới đến Pool

Phục vụ một phương thức finder

Phục vụ phương thức create

beanClass.newInsance()

Phục vụ một business method

trả về một EJB object hay một collection EJB object

bussinessMethod(args)

ejbObj.businessMethod(args)

ejbStore()

ejbPassivate()

ejbObj.remove()

hoặc

home.remove()

Phục vụ phương thức remove

home.find<METHOD>(args)

ejbFind<METHOD>(args)

tìm trong cơ sở dữ liệu

trên cơ sở args

trả về một PK hay

một collection PK

trả về EJB object mới

home.create(args)

ejbCreate(args)

chèn một hàng

vào cơ sở dữ liệu

trả về một PK

ejbPostCreate(args)

ejbObj. businessMethod(args)

hoặc

ejbHome. businessMethod(args)

cập nhật hàng thể hiện

vào cơ sở dữ liệu

Passivating một thực thể bean

ejbActivate()

đọc dữ liệu vào thực thể từ cơ sở dữ liệu

ejbLoad()

bussinessMethod(args)

Activating một thực thể bean

xóa hàng thể hiện

trong cơ sở dữ liệu

ejbRemove()

unsetEntityContext()

Loại một thực thể khỏi Pool

3. Lớp và giao diện

a) Client truy xuất từ xa

Identity mamagement

Home method

Appilcation

Logic

PK mamagement

State management

interface EJBHome

+ EJBMetaData getEJBMetaData( )  
+ HomeHandle getHomeHandle( )

+ void remove( Handle )

throws RemoveException

+ void remove(<PK> )

throws RemoveException

interface <name>Home

+ <remote> create( <args> )

throws CreateException

+ <remote> findByPrimaryKey( <PK> )

throws FinderException

+ <remote collection>

find<method>( <args> )

throws FinderException

+ <type> <name>( <args> )

throws SomeException

interface EJBObject

+ void remove( Handle )

throws RemoveException

+ EJBHome getEJBHome( )

+ Handle getHandle( )

+ Object getPrimaryKey( )

+ boolean isIdentical( EJBObject )

interface <name>

+ <ret type> business\_1( <args> )

throws SomeException

+ <ret type> business\_2( <args> )

throws SomeException

+ abstract <*name*>Bean

+ <type> *get*<*AttributeKey*>()

+ void *set*<*AttributeKey*>( <type> )

+ <type> *get*<*AttributeX*>()

+ void *set*<*AttributeX*>( <type> )

+ <type> *get*<*RelationY*>()

+ void *set*<*RelationY*>( <type> )

+ void ejbLoad( )

+ void ejbStore( )

+ void ejbActivate( )

+ void ejbPassivate( )

+ void setEntityContext( <arg> )

+ void unsetEntityContext( )

+ void ejbRemove( )

throws RemoveException

+ <PK> ejbCreate( <args> )

throws CreateException

+ <PK> *ejbFindByPrimaryKey*( <PK> )

throws FinderException

+ <PK collection>

*ejbFind*<method>( <args> )

throws FinderException

+ <PK collection>

*ejbSelect*<method>( <args> )

+ <type>ejbHome<Name>( <arg> )

throws SomeException

+ <ret type> business\_1( <args> )

throws SomeException

+ <ret type> business\_2( <args> )

throws SomeException

interface EntityBean

↑ throws RemoteException ↑ ↑ throws EJBException ↑

+ <name>PK

<type> <attributeKey>

interface java.io.Serializable

b) Client truy xuất cục bộ (local)

Identity mamagement

Home method

Appilcation

Logic

PK mamagement

State management

interface EJBLocalHome

+ void remove(<PK> )

throws RemoveException

interface <name>Home

+ <local> create( <args> )

throws CreateException

+ <local> findByPrimaryKey( <PK> )

throws FinderException

+ <local collection>

find<method>( <args> )

throws FinderException

+ <type> <name>( <args> )

throws SomeException

interface EJBLocalObject

+ void remove( Handle )

throws RemoveException

+ EJBLocalHome getEJBLocalHome( )

+ Object getPrimaryKey( )

+ boolean isIdentical( EJBLocalObject )

interface <name>

+ <ret type> business\_1( <args> )

throws SomeException

+ <ret type> business\_2( <args> )

throws SomeException

+ abstract <name>Bean

+ <type> *get*<*AttributeKey*>()

+ void *set*<*AttributeKey*>( <type> )

+ <type> *get*<*AttributeX*>()

+ void *set*<*AttributeX*>( <type> )

+ <type> *get*<*RelationY*>()

+ void *set*<*RelationY*>( <type> )

+ void ejbLoad( )

+ void ejbStore( )

+ void ejbActivate( )

+ void ejbPassivate( )

+ void setEntityContext( <arg> )

+ void unsetEntityContext( )

+ void ejbRemove( )

throws RemoveException

+ <PK> ejbCreate( <args> )

throws CreateException

+ <PK> *ejbFindByPrimaryKey*( <PK> )

throws FinderException

+ <PK collection>

*ejbFind*<method>( <args> )

throws FinderException

+ <PK collection>

*ejbSelect*<method>( <args> )

+ <type>ejbHome<Name>( <arg> )

throws SomeException

+ <ret type> business\_1( <args> )

throws SomeException

+ <ret type> business\_2( <args> )

throws SomeException

interface EntityBean

↑ throws EJBException ↑

+ <name>PK

<type> <attributeKey>

interface java.io.Serializable

4. So sánh giữa CMP và BMP

- Ưu điểm:

Thời gian hoàn thành

Tính khả chuyển (portability)

Tính linh động (flexibility)

Cải tiến hiệu suất: bằng entity caching và lazy loading.

Dễ bảo trì

- Khuyết điểm:

Khó debug

Giảm khả năng điều khiển

Phải học thêm nhiều

**II. Thiết kế**

- Ứng dụng minh họa mô phỏng business logic tạo Account và tìm kiếm Account bằng nhiều cách, sử dụng một CMP (**AccountBean**), truy xuất bằng hai cách:

* Truy xuất từ xa thông qua GUI, giao diện sẽ sử dụng một session bean (**AccountFacade**) theo design pattern Session Façade để cụ thể hóa các use-case truy xuất CMP, thông tin được lưu chuyển nhờ một đối tượng theo dessign pattern Transfer Object.
* Truy xuất cục bộ từ lớp Web, controller là các trang JSP (**CreateAccount.jsp** và **DisplayAccounts.jsp**) sẽ dùng một bean đơn giản (**InvokeEJB**) theo design pattern Business Delegate để truy xuất CMP.

- Thiết kế ứng dụng: (CMP 2.x)

Web Container

EJB Container

GUI

Presentation Tier

Bussiness

Tier

EIS

Tier

*Remote call*

Database

Client

Tier

*Domain Layer*

*Services Layer*

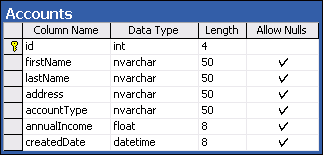
*Application Layer*

Browser

**Simple Bean**

**A. CMP**

- Cơ sở dữ liệu cho ứng dụng: **Bank**, bảng **accounts**.



Field khóa chính **id** tự tăng với Identity Seed (trị đầu tiên) bằng 1 và Identity Increment (trị tăng) bằng 1.

1. Lớp EJB

- Nhiều phương thức của lớp CMP cài đặt trong tập tin mô tả **ejb-jar.xml**. Container, bản thân là một component run-time, sẽ hiện thực các phương thức này. Vì vậy lớp CMP phải được khai báo **abstract**, gồm có:

* Các phương thức **abstract** getter/setter.
* Các phương thức callback sẽ được gọi bởi Container.
* Các phương thức Finder.
* Các phương thức Select.
* Các phương thức Home.

- Khi dùng XDoclet, không cần tạo các giao diện cũng như các DD. Thay vào đó, cần một số khai báo XDoclet cho việc tạo các giao diện và các DD. Ví dụ:

**//** khai báo **package**

**//** các khai báo **import**

**/\*\***

**\* @ejb.bean**

**\* type="CMP"**

**\* cmp-version="2.x"**

**\* name="Account"**

**\* schema="AccountBean"**

**\* jndi-name="AccountLocalJNDI"**

**\* view-type="local"**

**\* primkey-field="id";**

**\***

**\* @ejb.value-object**

**\***

**\* @ejb.persistence table-name="accounts"**

**\***

**\* @ejb.pk method-intf="Home"**

**\***

**\* @ejb.finder**

**\* signature="java.util.Collection findByFirstName( java.lang.String firstName )"**

**\* query="SELECT OBJECT(a) FROM AccountBean AS a WHERE e.firstName LIKE ?1"**

**\***

**\*/**

**public abstract class AccountBean implements EntityBean {**

**// ...**

**}**

a) Các CMP field

- CMP field dùng định nghĩa các thuộc tính lưu trữ (persistent attribute) của một Entity, chính là các thuộc tính của bảng dữ liệu. Đặc tả EJB giới hạn chúng trong:

* Các kiểu cơ bản, như **int** hoặc **double**.
* Các đối tượng có khả năng serialize, bao gồm các lớp cơ bản, như **Integer** và **String**.

Đặc tả EJB không định nghĩa cách các thuộc tính được ánh xạ đến cơ sở dữ liệu. Điều này cung cấp sự linh động cho Container khi cho phép Entity được lưu trữ trên các cơ sở dữ liệu khác nhau như cơ sở dữ liệu quan hệ, cơ sở dữ liệu XML, hệ thư mục LDAP, …

- CMP field được *khai báo trong tập tin mô tả* **ejb-jar.xml** bằng cách dùng element **<entity>**. Mỗi tên phải bắt đầu bằng *ký tự thường*. Các CMP field *không* được khai báo trong lớp Bean.

**...**

**<entity>**

**<ejb-name>Account</ejb-name>**

**...**

**<cmp-field><field-name>id</field-name></cmp-field>**

**<cmp-field><field-name>firstName</field-name></cmp-field>**

**<cmp-field><field-name>lastName</field-name></cmp-field>**

**<cmp-field><field-name>address</field-name></cmp-field>**

**<cmp-field><field-name>accountType</field-name></cmp-field>**

**<cmp-field><field-name>annualIncome</field-name></cmp-field>**

**...**

**</entity>**

**...**

- Các kiểu field được hỗ trợ trực tiếp bởi JBoss liệt kê trong bảng sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kiểu Java** | **Kiểu JDBC** | **Phương thức Set** | **Phương thức Get** | **Ghi chú** |
| boolean | BIT | setObject | getBoolean |  |
| byte | TINYINT | setObject | getByte |  |
| short | SMALLINT | setObject | getShort |  |
| int | INTEGER | setObject | getInt |  |
| long | BIGINT | setObject | getLong |  |
| float | FLOAT | setObject | getFloat |  |
| double | DOUBLE | setObject | getDouble |  |
| char | VARCHAR | setObject | getString | Chỉ trả về ký tự đầu tiên |
| java.lang.Boolean | BIT | setObject | getBoolean |  |
| java.lang.Byte | TINYINT | setObject | getByte |  |
| java.lang.Short | SMALLINT | setObject | getShort |  |
| java.lang.Integer | INTEGER | setObject | getInt |  |
| java.lang.Long | BIGINT | setObject | getLong |  |
| java.lang.Float | FLOAT | setObject | getFloat |  |
| java.lang.Double | DOUBLE | setObject | getDouble |  |
| java.lang.Character | VARCHAR | setObject | getString | Chỉ trả về ký tự đầu tiên |
| java.lang.String | VARCHAR | setObject | getString |  |
| CLOB | setCharacterStream | getCharacterStream |  |
| LONGVARCHAR | setCharacterStream | getCharacterStream |  |
| java.sql.Date | DATE | setObject | getDate |  |
| java.sql.Time | TIME | setObject | getTime |  |
| java.sql.Timestamp | TIMESTAMP | setObject | getTimestamp |  |
| java.util.Date | DATE | setObject | getTimestamp |  |
| TIME | setObject | getTimestamp |  |
| TIMESTAMP | setObject | getTimestamp |  |
| java.math.BigDecimal | DECIMAL | setBigDecimal | getBigDecimal |  |
| NUMERIC | setBigDecimal | getBigDecimal |  |
| Any | BINARY | setBytes | getBytes | Đối tượng được serialize |
| VARBINARY | setBytes | getBytes | Đối tượng được serialize |
| BLOB | setBinaryStream | getBinaryStream | Đối tượng được serialize |
| LONGVARBINARY | setBinaryStream | getBinaryStream | Đối tượng được serialize |

- Một CMP field có thể được khai báo trong XDoclet bằng cách dùng tag **@ejb-persistence**:

**/\*\***

**\* The Customer's name**

**\* @return this Customer's name**

**\* @ejb.persistence column-name="NAME"**

**\***

**\*/**

**public abstract String getName();**

Thuộc tính **column-name** cần phải có. Các phiên bản cũ của XDoclet dùng tag **@ejb.persistent-field** để khai báo một accessor nhưng từ phiên bản 1.2 tag này đã lạc hậu.

b) Các phương thức **abstract** getter/setter

- Để Container toàn quyền điều khiển việc truy xuất trị của CMP field, lớp Bean *không định nghĩa* các hàm thành viên truy xuất CMP field (các getter/setter cho các field). Thay vào đó, các getter/setter được khai báo như các phương thức **abstract**, cài đặt cụ thể sẽ do Container cung cấp. Cả hai getter/setter phải được khai báo tương ứng với mỗi CMP field. Tuy nhiên, các business method khai báo trong remote interface không nhất thiết là tất cả các accessor này mà chỉ chọn một số phương thức cần thiết.

**public abstract int getId();**

**public abstract void setId( int id );**

**public abstract String getFirstName();**

**public abstract void setFirstName( String firstName );**

**public abstract String getLastName();**

**public abstract void setLastName( String lastName );**

**public abstract String getAddress();**

**public abstract void setAddress( String address );**

**public abstract String getAccountType();**

**public abstract void setAccountType( String accountType );**

**public abstract double getAnnualIncome();**

**public abstract void setAnnualIncome( double annualIncome );**

**public abstract Date getCreatedDate();**

**public abstract void setCreatedDate( Date createdDate );**

- Các getter abstract trên thường dùng trong design pattern Transfer Object (value object, xem chương trước), trả về một transfer object chuyên chở dữ liệu cho bean. Ví dụ:

**public AccountTO toTransferObject() {**

**AccountTO to = new AccountTO();**

**to.setId( getId() );**

**to.setFirstName( getFirstName() );**

**to.setLastName( getLastName() );**

**// ...**

**return to;**

**}**

- Trong XDoclet, một cặp phương thức abstract getter/setter được khai báo như sau. Tag **@ejb.pk-field** được dành riêng cho CMP field là khóa chính

**/\*\***

**\* @ejb.interface-method**

**\* @ejb.persistence column-name="id"**

**\* @ejb.pk-field**

**\***

**\*/**

**public abstract int getId();**

**public abstract void setId( int id );**

c) Các phương thức callback

- Phương thức **ejbCreate()** khởi tạo cho các field dữ liệu của bean *thông qua các setter abstract*, không thông qua phép gán thông thường do không có khai báo dữ liệu riêng của lớp. Phương thức **ejbCreate()** trả về một trị **null**. Điều này cần thiết khi muốn thừa kế CMP rồi có những hiệu chỉnh để tạo một BMP mới.

- Nếu khóa chính là một field được cơ sở dữ liệu cấp giá trị tự tăng (auto-increment), không cần phải khởi tạo giá trị cho field đó. Nghĩa là truy vấn INSERT để chèn trị cho field này vào bảng do cơ sở dữ liệu thực hiện.

- XDoclet dùng tag **@ejb.create-method** để khai báo **ejbCreate()**, dùng cho việc tạo home interface. Các phương thức callback còn lại không cần khai báo tag của XDoclet.

**/\*\***

**\* @ejb.create-method**

**\***

**\*/**

**public AccountPK ejbCreate( String firstName, String lastName, String address,**

**String accountType, double annualIncome, Date createdDate )**

**throws CreateException {**

**// id** là field auto-increment

**this.setFirstName( firstName );**

**this.setLastName( lastName );**

**this.setAddress( address );**

**this.setAccountType( accountType );**

**this.setAnnualIncome( annualIncome );**

**this.setCreatedDate( createdDate );**

**return null;**

**}**

- Với mỗi phương thức **ejbCreate(…)** phải định nghĩa một phương thức **ejbPostCreate(...)** có cùng danh sách đối số và không có trị trả về. **ejbPostCreate()** được thực hiện bởi container sau **ejbCreate()** tương ứng với cùng một transaction context.

**public void ejbPostCreate( String firstName, String lastName, String address,**

**String accountType, double annualIncome, Date[[1]](#footnote-2) createdDate )**

**{ }**

- Các phương thức **ejbRemove()**, **ejbLoad()**, **ejbRestore()**, **ejbPassivate()**, **ejbActivate()** không cần cài đặt vì đã được container hiện thực; đôi khi không cần khai báo đủ tất cả.

**public void ejbRemove() { }**

**public void ejbActivate() { }**

**public void ejbPassivate() { }**

**public void ejbLoad() { }**

**public void ejbStore() { }**

**public void setEntityContext( EntityContext ctx ) { this.ctx = ctx; }**

**public void unsetEntityContext() { this.ctx = null; }**

d) Các phương thức Finder

- Các phương thức Finder được *khai báo trong home hoặc local home interface* của một entity bean. Với CMP, không cần và không thể cài đặt phương thức Finder trong lớp CMP, mà sẽ được Container thực hiện tự động hoặc thực hiện với truy vấn EJB-QL định nghĩa cho phương thức đó.

- Với việc tìm một hàng dữ liệu đơn (ví dụ tìm theo khóa chính), phương thức Finder trả về một tham chiếu đến một thực thể entity đơn, hoặc **null**. Nếu không tìm thấy kết quả, Finder sẽ ném **ObjectNotFoundException**; nếu kết quả chứa hơn một hàng dữ liệu, Finder sẽ ném **FinderException**. Một trị **null** được trả về nếu truy vấn trả về một kết quả và trị của kết quả này là **null**.

- Với việc tìm nhiều hàng dữ liệu (ví dụ tìm theo tên), phương thức Finder luôn trả về một **java.util.Set** hay **java.util.Collection[[2]](#footnote-3)** các tham chiếu đến thực thể. **Set** hoặc **Collection** này có thể rỗng nếu kết quả truy vấn là rỗng. Kết quả cũng có thể là **null** nếu truy vấn trả về trị **null**. Một Collection kết quả cũng có thể chứa trị trùng nếu không dùng truy vấn **DISTINCT**.

- Các phương thức Finder *được định nghĩa trong tập tin mô tả* **ejb-jar.xml** bằng cách dùng element **<query>** với ngôn ngữ EJB-QL (EJB Query Language). Đối số cho các phương thức này là tùy trường hợp, phải khai báo trong element **<method-params>**.

**<!--** Finders **-->**

**<query>**

**<query-method>**

**<method-name>findByFirstName</method-name>**

**<method-params>**

**<method-param>java.lang.String</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT OBJECT(a) FROM AccountBean AS a WHERE a.firstName LIKE ?1]]>**

**</ejb-ql>**

**</query>**

Finder khai báo trong home interface

**<query>**

**<query-method>**

**<method-name>findByCreatedDate</method-name>**

**<method-params>**

**<method-param>java.util.Date</method-param>**

Truyền tham số cho các phát biểu EJB-QL

**</method-params>**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT DISTINCT OBJECT(a) FROM AccountBean AS a WHERE a.createdDate >= ?1]]>**

**</ejb-ql>**

**</query>**

**<query>**

**<query-method>**

**<method-name>findByAllAccounts</method-name>**

**<method-params />**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT DISTINCT OBJECT(a) FROM AccountBean AS a]]>**

**</ejb-ql>**

**</query>**

**<query>**

**<query-method>**

**<method-name>findByMinMaxAnnualIncome</method-name>**

**<method-params>**

**<method-param>java.lang.Double</method-param>**

**<method-param>java.lang.Double</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT DISTINCT OBJECT(a) FROM AccountBean AS a WHERE**

**a.annualIncome BETWEEN ?1 AND ?2]]>**

**</ejb-ql>**

**</query>**

- Element **<abstract-schema-name>** mô tả quan hệ của nhiều CMP (trong trường hợp trên là **AccountBean**), được dùng trong truy vấn của EJB-QL. Phân đoạn **<![CDATA[** *EJB-QLquery* **]]>** của XML dùng bảo đảm tính well-formed của tập tin XML. Chú ý khác biệt cơ bản của EJB-QL với SQL là EJB-QL thao tác trên đối tượng.

- Container thường có khả năng giới hạn khi thực hiện các phương thức Finder, như chỉ tự động thực hiện phương thức **findByPrimaryKey( <PK> )**. Với các truy vấn phức tạp, container cần được tăng cường bằng ngôn ngữ EJB-QL, tuy nhiên ngôn ngữ này cũng vẫn còn nhiều hạn chế. Tùy nhà cung cấp application server, CMP sẽ được hỗ trợ bằng nhiều mở rộng khác nhau, trong tài liệu này, ta xem xét những hỗ trợ mở rộng do JBoss cung cấp.

Nếu truy vấn cho các phương thức Finder không thể chỉ định bởi EJB-QL do giới hạn của ngôn ngữ truy vấn này, element **<ejb-ql>** sẽ để trống. Định nghĩa của query sẽ được cung cấp trong **jbosscmp-jdbc.xml** với sự hỗ trợ mở rộng truy vấn của JBoss. JBoss cung cấp nhiều hỗ trợ mở rộng truy vấn[[3]](#footnote-4) như:

* Ngôn ngữ JBoss-QL: định nghĩa rỗng trong **ejb-jar.xml** và định nghĩa bằng element **<jboss-ql>** trong **jbosscmp-jdbc.xml** với ngôn ngữ truy vấn mở rộng JBoss-QL. Ví dụ (dùng trong ví dụ minh họa):

trong **ejb-jar.xml**:

**<!--** for JBoss-QL overridden **-->**

**<query>**

**<query-method>**

**<method-name>findByTopAccounts</method-name>**

**<method-params>**

**<method-param>int</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql />**

**</query>**

và trong **jbosscmp-jdbc.xml**:

**<query>**

**<query-method>**

**<method-name>findByTopAccounts</method-name>**

**<method-params>**

**<method-param>int</method-param>**

**</method-params>**

**</query-method>**

**<jboss-ql>**

**<![CDATA[SELECT OBJECT(a) FROM AccountBean AS a**

**ORDER BY a.annualIncome DESC LIMIT ?1]]>**

**</jboss-ql>**

**</query>**

* Declared SQL: định nghĩa rỗng trong **ejb-jar.xml** và định nghĩa bằng element **<declared-sql>** trong **jbosscmp-jdbc.xml** chứa các element khai báo truy vấn SQL. Ví dụ:

**<query>**

**<query-method>**

**<method-name>ejbSelectWithValueOver</method-name>**

**<method-params>**

**<method-param>double</method-param>**

**</method-params>**

**</query-method>**

**<declared-sql>**

**<select>**

**<distinct />**

**<ejb-name>Order</ejb-name>**

**<alias>o</alias>**

**<additional-columns>, SUM(p.PRICE\*l.QUANTITY)</additional-columns>**

**</select>**

**<from>**

**LEFT JOIN LINEITEM l ON (o.ORDER\_ID = l.FK\_ORDER\_ID)**

**INNER JOIN PRODUCT p ON (l.FK\_PRODUCT\_ID = p.PRODUCT\_ID)**

**</from>**

**<where>**

**<![CDATA[o.ORDER\_ID = o.ORDER\_ID GROUP BY ORDER\_ID HAVING**

**SUM(p.PRICE\*l.QUANTITY) > {0}]]>**

**</where>**

**<order>SUM(p.PRICE\*l.QUANTITY) DESC</order>**

**</declared-sql>**

**</query>**

cho phát biểu truy vấn:

**SELECT o.ORDER\_ID, SUM(p.PRICE\*l.QUANTITY)**

**FROM ORDER\_DATA o**

**LEFT JOIN LINEITEM l ON (o.ORDER\_ID = l.FK\_ORDER\_ID)**

**INNER JOIN PRODUCT p ON (l.FK\_PRODUCT\_ID = p.PRODUCT\_ID)**

**WHERE o.ORDER\_ID = o.ORDER\_ID**

**GROUP BY ORDER\_ID**

**HAVING SUM(p.PRICE\*l.QUANTITY) > ?**

**ORDER BY SUM(p.PRICE\*l.QUANTITY) DESC**

EJB-QL không cung cấp **JOIN**, cũng có thể giải quyết bằng cách tạo data view trong cơ sở dữ liệu rồi dùng CMP với data view như với một bảng bình thường.

* Dynamic Queries: truy vấn động, định nghĩa rỗng trong **ejb-jar.xml** và định nghĩa rỗng dùng element **<dymamic-ql>** trong **jbosscmp-jdbc.xml**. Phương thức dynamic này dùng 2 đối số: một chuỗi JBoss-QL sẽ được *cung cấp trong code* và một mảng các Object chứa đối số cho chuỗi JBoss-QL đó.

- Khi dùng XDoclet, các phương thức Finder cần khai báo trong **ejb-jar.xml** sẽ được khai báo đầu lớp Bean với tag **@ejb.finder** và các thuộc tính như **signature** (tiêu đề của phương thức Finder), **query** (truy vấn bằng EJB-QL), dùng định nghĩa rõ phương thức đó.

**/\*\***

**\* ...**

**\* @ejb.finder**

**\* signature="java.util.Collection findByFirstName( java.lang.String firstName )"**

**\* query="SELECT OBJECT(a) FROM AccountBean AS a WHERE e.firstName LIKE ?1"**

**\***

**\* ...**

**\*/**

e) Các phương thức Select

- Các phương thức Select cung cấp cơ chế cho phép một entity bean tự thực hiện các truy vấn. Các phương thức Select không bị ràng buộc giống như các Finder như chỉ trả về một thực thể thuộc lớp bean; chúng có thể trả về tham chiếu đến một thực thể bất kỳ, một trị từ một CMP field, hoặc một tham chiếu đến một thực thể từ CMR field, một **Set** hoặc **Collection**.

- Các phương thức Select được *khai báo trong lớp bean* như các phương thức **abstract** với tiếp đầu ngữ bắt buộc là **ejbSelect**. Đồng thời chúng được *định nghĩa trong tập tin mô tả* **ejb-jar.xml** bằng cách dùng element **<query>** với EJB-QL. Do các phương thức Select không liên kết với một home hoặc local home interface, kiểu tham chiếu được trả về có thể phải chỉ định bằng cách dùng element **<result-type-mapping>**, mặc định là **Local**.

- Các phương thức Select chỉ dùng bên trong lớp bean bởi các business method hay các home method. Vào thời điểm cài đặt, container sẽ sinh ra phương thức cụ thể sẽ được dùng bởi thực thể bean.

- Ví dụ:

định nghĩa trong **ejb-jar.xml** bằng EJB-QL.

**<!--** ejbSelect **-->**

**<query>**

**<query-method>**

**<method-name>ejbSelectAccountType</method-name>**

**<method-params>**

**<method-param>java.lang.Integer</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT a.accountType FROM AccountBean AS a WHERE a.id = ?1]]>**

**</ejb-ql>**

**</query>**

**<query>**

**<query-method>**

**<method-name>ejbSelectAllAnnualIncome</method-name>**

**<method-params />**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT a.annualIncome FROM AccountBean AS a]]>**

**</ejb-ql>**

**</query>**

khai báo **public abstract** trong lớp bean, ném ra **FinderException** cho trường hợp tìm kiếm thất bại.

**//** Abstract select methods - used EJB-QL

**public abstract String ejbSelectAccountType( Integer accountId ) throws FinderException;**

**public abstract Collection ejbSelectAllAnnualIncome() throws FinderException;**

và dùng trong các phương thức Home (xem phần sau).

- XDoclet khai báo phương thức Select ngay trên khai báo của phương thức Select đó trong lớp bean với tag **@ejb.select** và các thuộc tính dùng định nghĩa rõ phương thức đó.

**/\*\***

**\* @ejb.select**

**\* query="SELECT a.annualIncome FROM AccountBean AS a"**

**\***

**\*/**

**public abstract Collection ejbSelectAllAnnualIncome() throws FinderException;**

f) Các phương thức Home

- Các phương thức Home (home method) được khai báo trong home hoặc local home interface với tên phương thức **<name>**, lớp bean sẽ định nghĩa các phương thức ủy nhiệm tương ứng với tên **ejbHome<Name>**. Các phương thức **ejbHome<Name>** này sẽ triệu gọi cục bộ các phương thức Select ở trên. Chú ý các phương thức Select này là **abstract**, chỉ cài đặt bởi Container trong thời gian chạy, vì vậy cần viết code hợp lý cho **ejbHome<Name>**.

**//** Home methods - declare in home interface, call abstract select methods

**public String ejbHomeGetAccountType( AccountPK pk ) throws FinderException {**

**try {**

**return ejbSelectAccountType( new Integer( pk.getId() ) );**

**} catch ( Exception e ) { return null; }**

**}**

**public double ejbHomeGetTotalAllAnnualIncome() throws FinderException {**

**double totalAmount = 0.0;**

**Collection col = this.ejbSelectAllAnnualIncome();**

**Enumeration annual = Collections.enumeration( col );**

**while ( annual.hasMoreElements() ) {**

**Double amount = ( Double )annual.nextElement();**

**totalAmount += amount.doubleValue();**

**}**

**return totalAmount;**

**}**

- XDoclet khai báo phương thức Home ngay trên định nghĩa của phương thức Home đó trong lớp bean với tag **@ejb.home-method**, thuộc tính **view-type** cho biết phương thức Home này sẽ được khai báo trong cả hai (**both**) giao diện home hoặc chỉ một (**local** hoặc **remote**).

**/\*\***

**\* @ejb.home-method**

**\* view-type="local"**

**\***

**\*/**

**public double ejbHomeGetTotalAllAnnualIncome() throws FinderException {**

**// ...**

**}**

2. Các giao diện

- Ví dụ minh họa chỉ cần các giao diện cho truy xuất cục bộ:

+ Local interface: Local/Remote interface chứa bất kỳ business logic nào, không nhất thiết chỉ chứa các getter/setter.

**package myejb.entity.cmp;**

**public interface AccountLocal extends javax.ejb.EJBLocalObject**

**{**

**public int getId();**

**public void setId( int id );**

**public String getFirstName();**

**public void setFirstName( String firstName );**

**public String getLastName();**

**public void setLastName( String lastName );**

**public String getAddress();**

**public void setAddress( String address );**

**public String getAccountType();**

**public void setAccountType( String accountType );**

**public double getAnnualIncome();**

**public void setAnnualIncome( double annualIncome );**

**public java.util.Date getCreatedDate();**

**public void setCreatedDate( java.util.Date createdDate );**

**}**

+ Local home interface: Ngoài phương thức **create()** và các Finder, home interface còn chứa các phương thức Home, các phương thức này sẽ được client triệu gọi thông qua home interface.

**package myejb.entity.cmp;**

**import javax.ejb.\*;**

**import java.util.\*;**

**public interface AccountHomeLocal extends EJBLocalHome {**

**public AccountLocal create( int id, String firstName, String lastName,**

**String address, String accountType, double annualIncome, Date createdDate )**

**throws CreateException;**

**//** Finder method

**public AccountLocal findByPrimaryKey( AccountPK pk ) throws FinderException;**

**public Collection findByFirstName( String firstName ) throws FinderException;**

**public Collection findByCreatedDate( Date createdDate ) throws FinderException;**

**public Collection findByMinMaxAnnualIncome( Double minAmount, Double maxAmount )**

**throws FinderException;**

**public Collection findByTopAccounts( int numberOfAccounts ) throws FinderException;**

**public Collection findByAllAccounts() throws FinderException;**

**//** Home method

**public String getAccountType( AccountPK pk ) throws FinderException;**

**public int getTopID() throws FinderException;**

**public double getTotalAllAnnualIncome() throws FinderException;**

**}**

3. Deployment Desciptor

- DD **ejb-jar.xml** đã được mô tả từng phần bên trên. Nếu dùng XDoclet sẽ được tự động sinh ra cùng DD **jboss.xml**. Với CMP, **ejb-jar.xml** rất quan trọng vì mô tả trong DD nghĩa là cấu hình cho container hoạt động.

**<?xml version="1.0" ?>**

**<ejb-jar version="2.1" xmlns="http://java.sun.com/xml/ns/j2ee"**

**xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

**xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee**

**http://java.sun.com/xml/ns/j2ee/ejb-jar\_2\_1.xsd">**

**<enterprise-beans>**

**<entity>**

**<!--** CMP info **-->**

**<ejb-name>Account</ejb-name>**

**<local-home>myejb.entity.cmp.AccountHomeLocal</local-home>**

**<local>myejb.entity.cmp.AccountLocal</local>**

**<ejb-class>myejb.entity.cmp.AccountBean</ejb-class>**

**<persistence-type>Container</persistence-type>**

**<prim-key-class>myejb.entity.cmp.AccountPK</prim-key-class>**

Chỉ định sử dụng đặc tả CMP 2.x

**<reentrant>False</reentrant>**

**<cmp-version>2.x</cmp-version>**

**<abstract-schema-name>AccountBean</abstract-schema-name>**

**<!--** CMP fields **-->**

**<cmp-field><field-name>id</field-name></cmp-field>**

**<cmp-field><field-name>firstName</field-name></cmp-field>**

**<cmp-field><field-name>lastName</field-name></cmp-field>**

**<cmp-field><field-name>address</field-name></cmp-field>**

**<cmp-field><field-name>accountType</field-name></cmp-field>**

**<cmp-field><field-name>annualIncome</field-name></cmp-field>**

**<cmp-field><field-name>createdDate</field-name></cmp-field>**

**<!--**

**<primkey-field>** nếu có phải cùng nội dung với một trong các **<field-name>**

của **<cmp-field>**. Nếu PK có kiểu phức hợp thì không cần khai báo **<primkey-field>**

**<primkey-field>symbol</primkey-field>**

**-->**

**<!--** Finders **-->**

**<query>**

**<query-method>**

**<method-name>findByFirstName</method-name>**

**<method-params>**

**<method-param>java.lang.String</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT OBJECT(a) FROM AccountBean AS a WHERE a.firstName LIKE ?1]]>**

**</ejb-ql>**

**</query>**

**<query>**

**<query-method>**

**<method-name>findByCreatedDate</method-name>**

**<method-params>**

**<method-param>java.util.Date</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT DISTINCT OBJECT(a) FROM AccountBean AS a**

**WHERE a.createdDate >= ?1]]>**

**</ejb-ql>**

**</query>**

**<query>**

**<query-method>**

**<method-name>findByAllAccounts</method-name>**

**<method-params />**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT DISTINCT OBJECT(a) FROM AccountBean AS a]]>**

**</ejb-ql>**

**</query>**

**<query>**

**<query-method>**

**<method-name>findByMinMaxAnnualIncome</method-name>**

**<method-params>**

**<method-param>java.lang.Double</method-param>**

**<method-param>java.lang.Double</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT DISTINCT OBJECT(a) FROM AccountBean AS a**

**WHERE a.annualIncome BETWEEN ?1 AND ?2]]>**

**</ejb-ql>**

**</query>**

**<!--** ejbSelect **-->**

**<query>**

**<query-method>**

**<method-name>ejbSelectAccountType</method-name>**

**<method-params>**

**<method-param>java.lang.Integer</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT a.accountType FROM AccountBean AS a WHERE a.id = ?1]]>**

**</ejb-ql>**

**</query>**

**<query>**

**<query-method>**

**<method-name>ejbSelectAllAnnualIncome</method-name>**

**<method-params />**

**</query-method>**

**<ejb-ql>**

**<![CDATA[SELECT a.annualIncome FROM AccountBean AS a]]>**

**</ejb-ql>**

**</query>**

**<!--** for JBoss-QL overridden **-->**

**<query>**

**<query-method>**

**<method-name>findByTopAccounts</method-name>**

**<method-params>**

**<method-param>int</method-param>**

**</method-params>**

**</query-method>**

**<ejb-ql />**

**</query>**

**</entity>**

**<session>**

**<!—**Session Bean info **-->**

**...**

**</session>**

**</enterprise-beans>**

**</ejb-jar>**

**B. Truy xuất cục bộ - JSP gọi JavaBean đơn giản**

1. JavaBean

- Từ JSP, có thể truy xuất cục bộ trực tiếp CMP. Tuy nhiên ở đây nên dùng một JavaBean:

* JavaBean giống một Transfer Object, cho phép lưu chuyển dữ liệu nhanh giữa JSP và CMP. Trong vai trò này JavaBean gọi các phương thức của local interface.
* Các business logic JSP cần gọi từ local home interface của CMP có thể mô phỏng lại trong JavaBean theo design pattern Business Delegate.

**package myejb.beans;**

**import java.sql.\*;**

**import javax.sql.\*;**

**import java.util.\*;**

**import javax.naming.\*;**

**import myejb.entity.cmp.\*;**

**public class InvokeEJB {**

**private AccountHomeLocal accountHome = null;**

**private int id;**

**private String firstName;**

**private String lastName;**

**private String address;**

**private String accountType;**

**private double annualIncome;**

**private java.util.Date createdDate;**

**public InvokeEJB() { }**

**//** getter/setter from local interface, Transfer Object pattern

**public void setId( int id ) { this.id = id; }**

**public int getId() { return id; }**

**public void setFirstName( String firstName )**

**{ this.firstName = firstName; }**

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

**public void setLastName( String lastName )**

**{ this.lastName = lastName; }**

**public String getLastName() { return lastName; }**

**public void setAddress( String address )**

**{ this.address = address; }**

**public String getAddress() { return address; }**

**public void setAccountType( String accountType )**

**{ this.accountType = accountType; }**

**public String getAccountType() { return accountType; }**

**public void setAnnualIncome( double annualIncome )**

**{ this.annualIncome = annualIncome; }**

**public double getAnnualIncome() { return annualIncome; }**

**public void setCreatedDate( java.util.Date createdDate )**

**{ this.createdDate = createdDate; }**

**public java.util.Date getCreatedDate() { return createdDate; }**

**//** business logic from local home interface, Business Delegate pattern

**public String createAccount() throws Exception {**

**if ( accountHome == null ) this.initialize();**

**try {**

**id = accountHome.getTopID() + 1;**

**} catch ( Exception e ) {}**

**Calendar calendar = Calendar.getInstance();**

**calendar.set( Calendar.AM\_PM, Calendar.AM );**

**calendar.set( Calendar.HOUR, 0 );**

**calendar.set( Calendar.MINUTE, 0 );**

**calendar.set( Calendar.SECOND, 0 );**

**createdDate = calendar.getTime();**

**AccountLocal acc = accountHome.create( id, firstName, lastName, address,**

**accountType, annualIncome, createdDate );**

**return String.valueOf( id );**

**}**

**public Collection findByAllAccounts() throws Exception {**

**if ( accountHome == null ) this.initialize();**

**return EJBColToBeanCol( accountHome.findByAllAccounts() );**

**}**

**public Collection findByMinMaxAnnualIncome( String minAmount, String maxAmount )**

**throws Exception {**

**if ( accountHome == null ) this.initialize();**

**return EJBColToBeanCol( accountHome.findByMinMaxAnnualIncome(**

**new Double( minAmount ), new Double( maxAmount ) ) );**

**}**

**public Collection findByDate( java.util.Date createdDate ) throws Exception {**

**if ( accountHome == null ) this.initialize();**

**return EJBColToBeanCol( accountHome.findByCreatedDate( createdDate ) );**

**}**

**public Collection getTopAccounts( int numberOfAccounts ) throws Exception {**

**if ( accountHome == null ) this.initialize();**

**return EJBColToBeanCol( accountHome.findByTopAccounts( numberOfAccounts ) );**

**}**

**public String getSearchAccountType( String accountNumber ) throws Exception {**

**if ( accountHome == null ) this.initialize();**

**int key = Integer.parseInt( accountNumber );**

**return accountHome.getAccountType( new AccountPK( key ) );**

**}**

**//** helper methods

**private void initialize() throws Exception {**

**InitialContext ctx = new InitialContext();**

**accountHome = ( AccountHomeLocal )ctx.lookup( "AccountLocalJNDI" );**

**}**

**private Collection EJBColToBeanCol( Collection src ) {**

**ArrayList target = new ArrayList( src.size() );**

**Iterator iter = src.iterator();**

**while ( iter.hasNext() ) {**

**AccountLocal acc = ( AccountLocal )iter.next();**

**InvokeEJB ivb = new InvokeEJB();**

**ivb.setId( acc.getId() );**

**ivb.setFirstName( acc.getFirstName() );**

**ivb.setLastName( acc.getLastName() );**

**ivb.setAddress( acc.getAddress() );**

**ivb.setAccountType( acc.getAccountType() );**

**ivb.setAnnualIncome( acc.getAnnualIncome() );**

**ivb.setCreatedDate( acc.getCreatedDate() );**

**target.add( ivb );**

**}**

**return target;**

**}**

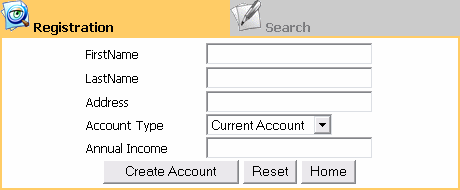
**}**

2. JSP

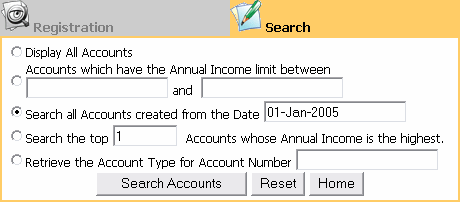
a) Form HTML front-end

- Các trang form HTML front-end dùng nhận dữ liệu từ người dùng thông qua browser. Gồm:

* Trang chính **index.html**, cũng là trang nhận dữ liệu nhập để tạo account. Validation form nhập phía client bằng Javascript dùng regular expression. Khi nhấn nút “Create Account”, trang này sẽ gọi JSP **CreateAccount.jsp**.



* Trang **SearchAccounts.html** nhận dữ liệu nhập cho mỗi yêu cầu tìm kiếm khác nhau. Validation form nhập phía client bằng Javascript dùng regular expression. Trang có dùng một control Calendar bằng Javascript hỗ trợ cho việc nhập ngày tháng. Khi nhấn nút “Search Accounts”, trang này sẽ gọi JSP **DisplayAccounts.jsp**.



b) JSP

- Các trang JSP (controller+view) nhận dữ liệu từ form HTML front-end tương ứng, tạo và chuyển dữ liệu cho JavaBean bằng các action chuẩn (jsp:useBean, jsp:setProperty), gọi các phương thức cần thiết (sẽ được JavaBean ủy nhiệm đến

**C. Truy xuất từ xa – từ GUI gọi Façade Session Bean**

- Phần client minh họa có 2 tab:

* Tab Registration tạo một account mới, gọi phương thức **create()** của home interface.
* Tab Search tìm account, sẽ gọi:

+ Các phương thức Finder của home interface, gồm: phương thức **findByPrimaryKey()** do container thực hiện tự động. Các phương thức **findByFirstName()** (theo kiểu LIKE) và **findAll()** được định nghĩa trong **ejb-jar.xml** bằng EJB-QL.

+ Khi dùng **findAll()**, phương thức **getTotalAllAnnualIncome()** sẽ tính tổng theo thuộc tính **annualIncome** bằng cách ủy nhiệm cho phương thức **ejbHomeGetTotalAllAnnualIncome()** trong lớp bean. Đến lượt phương thức này lại gọi phương thức Select **ejbSelectAllAnnualIncome()** khai báo abtract trong lớp bean và định nghĩa bằng EJB-QL trong container.

+ Click phải chọn một hàng, cho phép xóa hàng dữ liệu đó bằng phương thức **remove( primarykey )** của home interface (ta không override phương thức này).

Phần client nói chung giống phần client trong bài BMP, chỉ triển khai thêm một số thao tác trong phần tìm kiếm.

**private String[] topics = { "by FirstName (LIKE)", "by Registration ID", "find All" };**

**...**

**public void actionPerformed( ActionEvent e ) {**

**String[] tmp = new String[3];**

**while ( list.getRowCount() > 0 ) list.removeRow( 0 );**

**//** Find All

**try {**

**if ( ( ( String )jComboBox1.getSelectedItem() ).equals( topics[2] ) ) {**

**Collection c = null;**

**c = home.findAll();**

**if ( c != null ) {**

**Enumeration enumr = Collections.enumeration( c );**

**while ( enumr.hasMoreElements() ) {**

**Account acc = ( Account )enumr.nextElement();**

**tmp[0] = ( (AccountPK)acc.getPrimaryKey() ).toString();**

**tmp[1] = acc.getFirstName() + " " + acc.getLastName();**

**tmp[2] = String.valueOf( acc.getAnnualIncome() );**

**list.addRow( tmp );**

**}**

**JOptionPane.showMessageDialog( jPanel1, "Annual Income Total:\n" +**

**String.valueOf( home.getTotalAllAnnualIncome() ),**

**"Total", JOptionPane.INFORMATION\_MESSAGE );**

**}**

**return;**

**}**

**} catch ( Exception ex ) {**

**JOptionPane.showMessageDialog( jPanel1, "Database Empty!", "Message",**

**JOptionPane.INFORMATION\_MESSAGE );**

**}**

**Validate.resetValidate();**

**if ( !Validate.isNotBlank( "Keyword", jTextField5.getText() ) ) {**

**JOptionPane.showMessageDialog( jPanel1, Validate.errMessage,**

**"Error", JOptionPane.WARNING\_MESSAGE );**

**return;**

**}**

**try {**

**//** Find By FirstName (LIKE)

**if ( ( ( String )jComboBox1.getSelectedItem() ).equals( topics[0] ) ) {**

**Collection c = null;**

**c = home.findByFirstName( "%"+jTextField5.getText()+"%" );**

**if ( c != null ) {**

**Enumeration enumr = Collections.enumeration( c );**

**while ( enumr.hasMoreElements() ) {**

**Account acc = ( Account )enumr.nextElement();**

**tmp[0] = ( (AccountPK)acc.getPrimaryKey() ).toString();**

**tmp[1] = acc.getFirstName() + " " + acc.getLastName();**

**tmp[2] = String.valueOf( acc.getAnnualIncome() );**

**list.addRow( tmp );**

**}**

**}**

**}**

**//** Find By ID

**if ( ( ( String )jComboBox1.getSelectedItem() ).equals( topics[1] ) ) {**

**AccountPK key = new AccountPK( Integer.parseInt( jTextField5.getText() ) );**

**Account acc = null;**

**acc = home.findByPrimaryKey( key );**

**if ( acc != null ) {**

**tmp[0] = key.toString();**

**tmp[1] = acc.getFirstName() + " " + acc.getLastName();**

**tmp[2] = String.valueOf( acc.getAnnualIncome() );**

**list.addRow( tmp );**

**}**

**}**

**} catch ( Exception ex ) {**

**JOptionPane.showMessageDialog( jPanel1, "Account not found!", "Message",**

**JOptionPane.INFORMATION\_MESSAGE );**

**}**

**}**

**...**

**III. Triển khai trên JBoss 4.x**

- Chuẩn bị **jboss.xml**:

**<?xml version="1.0"?>**

**<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS 4.0//EN"**

**"http://www.jboss.org/j2ee/dtd/jboss\_4\_0.dtd">**

**<jboss>**

**<enterprise-beans>**

**<entity>**

**<ejb-name>Account</ejb-name>**

**<jndi-name>AccountJNDI</jndi-name>**

**</entity>**

**</enterprise-beans>**

**</jboss>**

- Chuẩn bị **sqlserver-ds.xml**:

**<?xml version="1.0" encoding="UTF-8"?>**

**<datasources>**

**<local-tx-datasource>**

**<jndi-name>Bank</jndi-name>**

**<connection-url>**

**jdbc:microsoft:sqlserver://venus:1433;databaseName=Bank**

**</connection-url>**

**<driver-class>com.microsoft.jdbc.sqlserver.SQLServerDriver</driver-class>**

**<user-name>sa</user-name>**

**<password></password>**

**</local-tx-datasource>**

**</datasources>**

- Chuẩn bị **sqlserver-ds.xml**:

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE jbosscmp-jdbc PUBLIC "-//JBoss//DTD JBOSSCMP-JDBC 3.2//EN"**

**"http://www.jboss.org/j2ee/dtd/jbosscmp-jdbc\_3\_2.dtd">**

**<jbosscmp-jdbc>**

**<defaults>**

**<datasource>java:/Bank</datasource>**

**<datasource-mapping>MS SQLSERVER</datasource-mapping>**

**</defaults>**

**<enterprise-beans>**

**<entity>**

**<ejb-name>Account</ejb-name>**

**<table-name>accounts</table-name>**

**<cmp-field>**

**<field-name>id</field-name>**

**<column-name>id</column-name>**

**</cmp-field>**

**<cmp-field>**

**<field-name>firstName</field-name>**

**<column-name>firstName</column-name>**

**</cmp-field>**

**<cmp-field>**

**<field-name>lastName</field-name>**

**<column-name>lastName</column-name>**

**</cmp-field>**

**<cmp-field>**

**<field-name>address</field-name>**

**<column-name>address</column-name>**

**</cmp-field>**

**<cmp-field>**

**<field-name>accountType</field-name>**

**<column-name>accountType</column-name>**

**</cmp-field>**

**<cmp-field>**

**<field-name>annualIncome</field-name>**

**<column-name>annualIncome</column-name>**

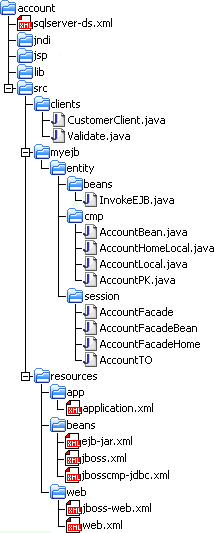
**</cmp-field>**

**</entity>**

**</enterprise-beans>**

**</jbosscmp-jdbc>**

2. Cấu trúc thư mục



1. Dùng java.util.Date, không nhất thiết phải dùng java.sql.Date. [↑](#footnote-ref-2)
2. Với Java 1.1, dùng java.util.Enumeration [↑](#footnote-ref-3)
3. Xem: Mark Fleury, Scott Stark, Richards Norman - **JBoss® 4.0 The Official Guide** - Sams Publishing. April 20, 2005. ISBN: 0-672-32648-5. Chi tiết các element của jbosscmp-jdbc.xml xem trong tập tin %JBOSS\_HOME%\docs\dtd\jbosscmp-jdbc\_4\_0.dtd. [↑](#footnote-ref-4)