



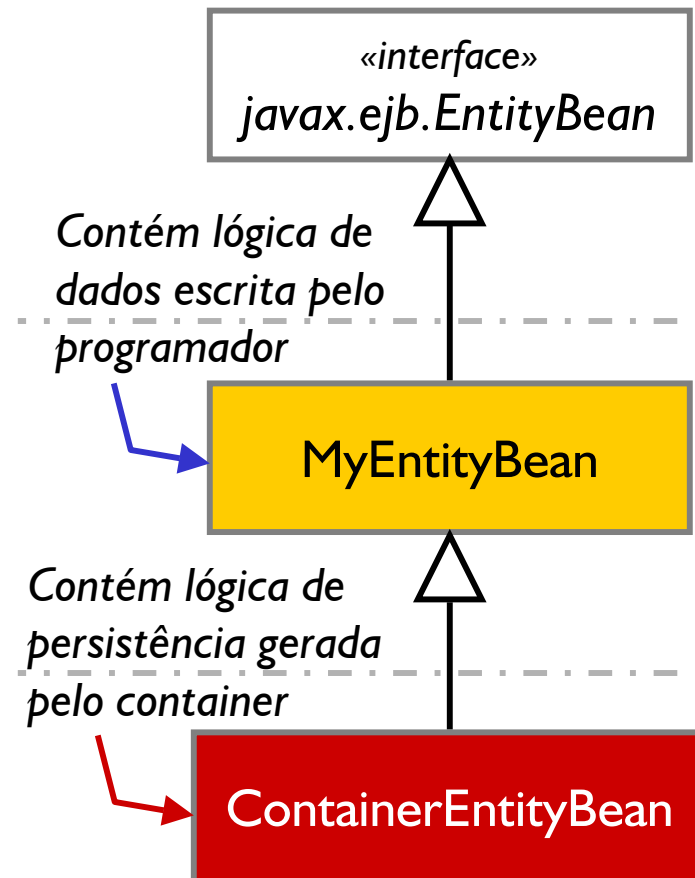
# Entity beans com CMP

*Helder da Rocha*  
[www.argonavis.com.br](http://www.argonavis.com.br)

- *Este módulo é uma continuação do módulo I I (Entity Beans).*
- *Desta vez veremos como criar beans de entidade persistentes sem escrever uma linha sequer de SQL e sem usar JDBC*
  - *O resultado: beans muito mais simples contendo apenas a lógica de negócio!*
- *CMP ainda é bastante limitado (não dá para substituir totalmente BMP ainda)*
- *Não abordaremos conceitos avançados de relacionamentos (apenas os fundamentos de CMR)*
  - *Para mais informações, consulte as referências!*

# Características de EJB CMP

- Com container-managed persistence o programador do bean não implementa lógica de persistência
  - O bean não precisa usar APIs como JDBC
  - O container se encarrega de criar as tabelas (no deployment), criar os beans e os dados, removê-los e alterá-los
- O container gera automaticamente o JDBC ao estender\* a classe do bean criada pelo programador
  - O entity bean verdadeiro é o que foi gerado: a combinação da classe implementada pelo programador e a gerada pelo container



\* Na verdade, o container tem a liberdade de utilizar outros mecanismos (delegação, por exemplo), desde que produzam o mesmo resultado

# Entity Beans CMP não tem ...

- ... campos de dados declarados
  - São implementados na **subclasse\*** (criada pelo container)
  - Programador declara seus campos no deployment descriptor (abstract persistence schema) usando `<cmp-field>`
- ... implementação de métodos de acesso
  - Programador declara todos os seus métodos de acesso (get/set) como abstract
  - Os nomes dos métodos devem ser compatíveis com os nomes dos campos declarados:  
  
Se há `getCoisa()` no bean, e no deployment descriptor deve haver:  
`<cmp-field><field-name>coisa</field-name></cmp-field>`
  - **Subclasse\*** gerada pelo container implementa os métodos (o container pode usar BMP, por exemplo)

\* ou no mecanismo alternativo implementado pelo container

- Como não há como fazer queries JDBC como entity beans CMP, há uma linguagem para fazer queries nos próprios beans: **Enterprise JavaBean Query Language** (EJB-QL)
- EJB-QL é uma linguagem com sintaxe semelhante a SQL para pesquisar entity beans. Possui
  - cláusula **SELECT**
  - cláusula **FROM**
  - cláusula **WHERE**, opcional
- EJB-QL é colocado nos deployment descriptors no contexto da declaração do método. Exemplo

```
SELECT OBJECT(obj)
FROM ProductBean obj
WHERE obj.name = ?1
```

# Métodos da interface *EntityBean* (I)

- *setEntityContext()* / *unsetEntityContext()*
  - *Mesmo comportamento de entity beans com BMP*
- *ejbFindXXX(...)*
  - *Não devem ser implementados em CMP*
  - *Use EJB-QL para definir ações de pesquisa para cada método no DD*
- *ejbSelectXXX(...)*
  - *Usados para fazer queries genéricos para o bean*
  - *É método de negócio mas não pode ser chamado pelo cliente*
  - *Declarados como abstract com lógica descrita no DD em EJB-QL*
- *ejbHomeXXX(...)*
  - *Usado para definir métodos que não são específicos a uma instância*
  - *Deve chamar um ejbSelect() para realizar as operações*
- *ejbCreate(...)* / *ejbPostCreate(...)*
  - *Chama os métodos de acesso abstratos inicializando-os com os dados que serão usados pelo container para criar novo registro*

# Métodos (2)

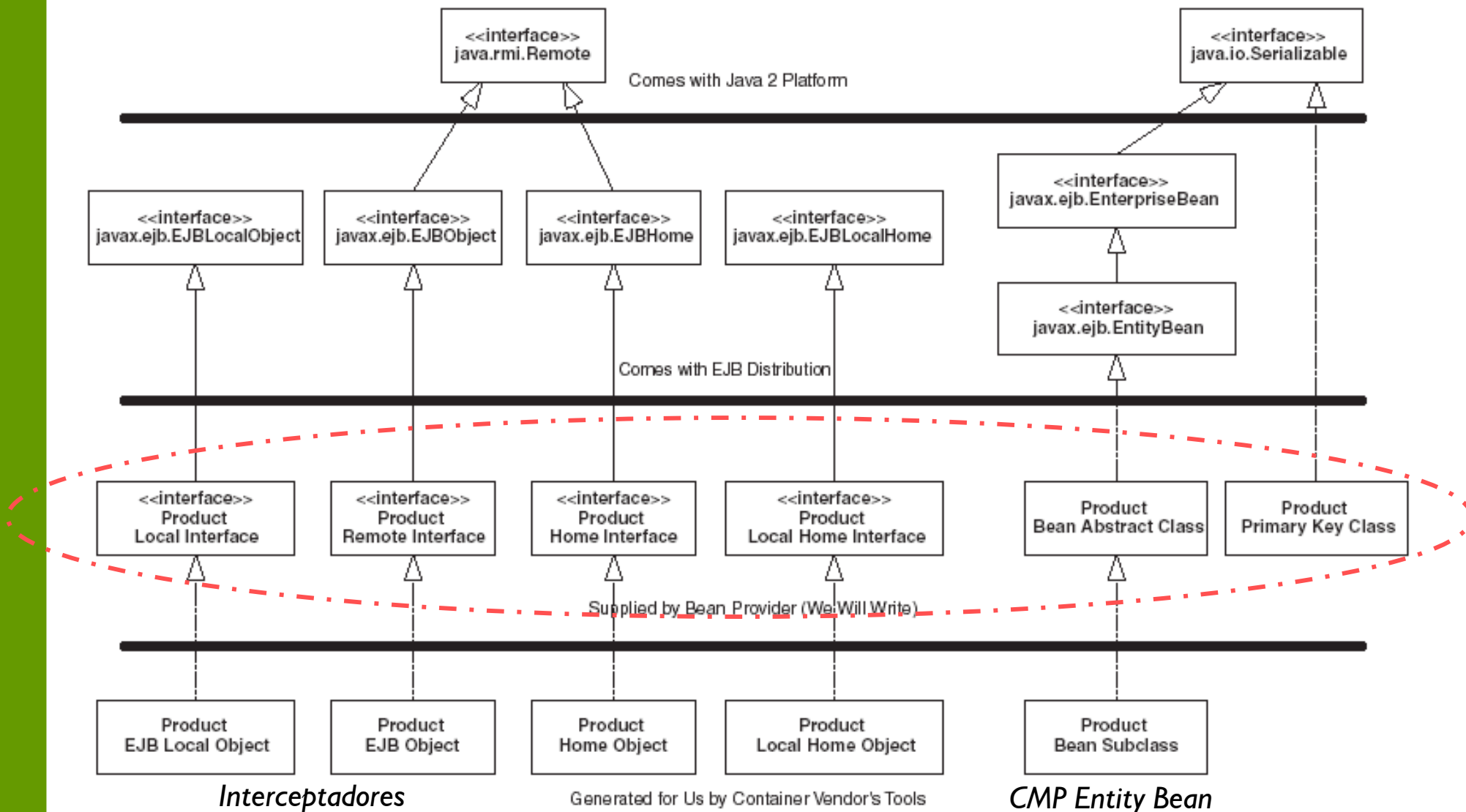
- *ejbActivate() / ejbPassivate()*
  - *Mesmo comportamento que BMP*
- *ejbLoad()*
  - *Usado para carregar dados do banco para o bean*
  - *Pode ser vazio ou conter rotinas para transformar dados recebidos*
- *ejbStore()*
  - *Usado para atualizar o banco com alterações no bean*
  - *Pode ser vazio ou conter rotinas p/ transformar dados antes do envio*
- *ejbRemove()*
  - *chamado por home.remove(), remove os dados do banco (não remove a instância - que pode ser reutilizada)*
  - *Pode ser vazio ou conter código para ser executado antes da remoção dos dados*

# Diferenças BMP-CMP

<i>Diferença</i>	<i>Container-Managed Persistence</i>	<i>Bean-Managed Persistence</i>
<i>Definição da classe</i>	<i>Abstrata</i>	<i>Concreta</i>
<i>Chamadas de acesso ao banco de dados</i>	<i>Gerada pelas ferramentas no deployment</i>	<i>Codificada pelo programador</i>
<i>Estado persistente</i>	<i>Representadas como campos persistentes virtuais</i>	<i>Codificadas como variáveis de instância</i>
<i>Métodos de acesso a campos persistentes e relacionamentos</i>	<i>Obrigatórios (abstract)</i>	<i>Não há</i>
<i>Método findByPrimaryKey</i>	<i>Gerado pelo container</i>	<i>Codificado pelo programador</i>
<i>Métodos finder customizados</i>	<i>Gerados pelo container mas programador deve escrever EJB-QL</i>	<i>Codificado pelo programador</i>
<i>Métodos select</i>	<i>Gerados pelo container</i>	<i>Não há</i>
<i>Valor de retorno de ejbCreate()</i>	<i>Deve ser null</i>	<i>Deve ser a chave primária</i>



# Exemplo CMP



**Figure 7.3** The object model for our product line.

# Exemplo: Como executar

- Diretório *cap13/mejb/*
- Configuração
  - Configure o arquivo *build.properties* com seu ambiente
  - Inicie o JBoss e o servidor de banco de dados Cloudscape
- Inicialização
  - > *ant drop.table*
  - > *ant clean*
- Deployment
  - > *ant jboss.deploy* (observe as mensagens de criação das tabelas no JBoss)
- Execução do cliente
  - > *ant run jboss.client*
  - > *ant select.all* (faz *SELECT* na tabela)

# Componentes do EJB-JAR

- *Interfaces Home e Remote e classe Primary Key*
  - *Idênticas às interfaces e classe Primary Key de um entity bean equivalente que usa BMP*
  - *PK não pode conter campos de relacionamento (CMR)*
- *Classe Enterprise bean (menor que classe BMP)*
  - *Métodos ejbLoad(), ejbStore() não possuem código de persistência*
  - *Classe é abstrata; métodos de acesso são abstratos; métodos ejbSelectXXX() são abstratos*
  - *Métodos de negócio contém apenas lógica de negócio*
  - *Métodos finder não são declarados*
- *Deployment descriptor (maior que versão BMP)*
  - *Definição da lógica de métodos e relacionamentos*
- *Arquivos de configuração do fabricante (JBoss)*
  - *Mapeamento de tipos, mapeamento de esquema*

# Enterprise Bean

```
public abstract class ProductBean implements EntityBean {
    protected EntityContext ctx;

    public abstract String getName();
    public abstract void setName(String name);
    public abstract String getProductID();
    public abstract void setProductID(String productID);
    (...)
    public void ejbStore() {}
    public void ejbLoad() {}
    public void ejbPassivate() {}
    public void ejbRemove() {}
    public void ejbActivate() {}
    public void setEntityContext(EntityContext c) {ctx = c;}
    public void unsetEntityContext() {ctx = null;}
    public void ejbPostCreate(String productID, String name,...) {}
    public void ejbCreate(String productID, String name, ...) {
        setProductID(productID);
        setName(name);
        (...)
    }
}
```

Métodos de negócio abstratos (lógica no deployment descriptor)

*ejbLoad e ejbStore são implementados pelo container*

*ejbFindBy não precisam ser declarados (lógica no deployment descriptor)*

*ejbCreate chama métodos abstratos para passar dados recebidos*

# Deployment Descriptor (I)

```
<ejb-jar>
  <enterprise-beans>
    <entity>
      <ejb-name>ProductEJB</ejb-name>
      <home>examples.ProductHome</home>
      <remote>examples.Product</remote>
      <ejb-class>examples.ProductBean</ejb-class>
      <persistence-type>Container</persistence-type>
      <prim-key-class>examples.ProductPK</prim-key-class>

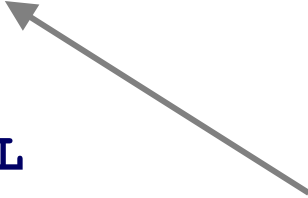
      <reentrant>False</reentrant>

      <cmp-version>2.x</cmp-version>
      <abstract-schema-name>
        ProductBean
      </abstract-schema-name>

    ...
```

# Deployment Descriptor (2)

```
...
<cmp-field>
  <field-name>productID</field-name>
</cmp-field>
<cmp-field><field-name>name</field-name></cmp-field>
<cmp-field>
  <field-name>description</field-name>
</cmp-field>
<cmp-field>
  <field-name>basePrice</field-name>
</cmp-field>
<!--
<primkey-field>productID</primkey-field>
-->
<query>
  ... declarações EJB-QL
</query>
</entity>
</enterprise-beans>
```



Se PK for um campo em vez de uma classe use este elemento em vez do <prim-key-class>

...

# <query> EJB-QL

```
<query>
  <query-method>
    <method-name>findByName</method-name>
    <method-params>
      <method-param>java.lang.String</method-param>
    </method-params>
  </query-method>
  <ejb-ql><![CDATA[ SELECT OBJECT(obj)
    FROM ProductBean obj WHERE obj.name = ?1 ]]></ejb-ql>
</query>
  ...
<query>
  <query-method>
    <method-name>findAllProducts</method-name>
  </query-method>
  <ejb-ql><![CDATA[
    SELECT OBJECT(obj) FROM ProductBean obj
      WHERE obj.productID IS NOT NULL
  ]]></ejb-ql>
</query>
```

Nome de método de Home ou Remote

Parâmetro recebido

Nome de um cmp-field

*findByPrimaryKey não precisa ser declarado!*

# JBoss deployment descriptor

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

<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS 3.0//EN"
           "http://localhost/dtd/jboss_3_0.dtd">

<jboss>
  <enterprise-beans>
    <entity>
      <ejb-name>ProductEJB</ejb-name>
      <jndi-name>product/ProductHome</jndi-name>
    </entity>
  </enterprise-beans>
  <resource-managers>
    <resource-manager>
      <res-name>jdbc/ejbPool</res-name>
      <res-jndi-name>java:/Cloudscape</res-jndi-name>
    </resource-manager>
  </resource-managers>
</jboss>
```

Nome JNDI Global

Nome do ENC java:comp/env

Nome JNDI global domínio java:



# JBoss CMP: jbosscmp-jdbc.xml

- Este arquivo é necessário se a aplicação irá se comunicar com uma fonte de dados existente com um esquema próprio

```
<jbosscmp-jdbc>
  <enterprise-beans>
    <entity>
      <ejb-name>ProductEJB</ejb-name>
      <table-name>Produto</table-name>
      <cmp-field>
        <field-name>name</field-name>
        <column-name>nome</column-name>
      </cmp-field>
      <cmp-field>
        <field-name>productID</field-name>
        <column-name>codigo</column-name>
      </cmp-field>
      <cmp-field>
        <field-name>basePrice</field-name>
        <column-name>preco</column-name>
      </cmp-field>
      ...
    </entity>
  </enterprise-beans>
</jbosscmp-jdbc>
```

*Tabela que será usada (ou criada) no banco*

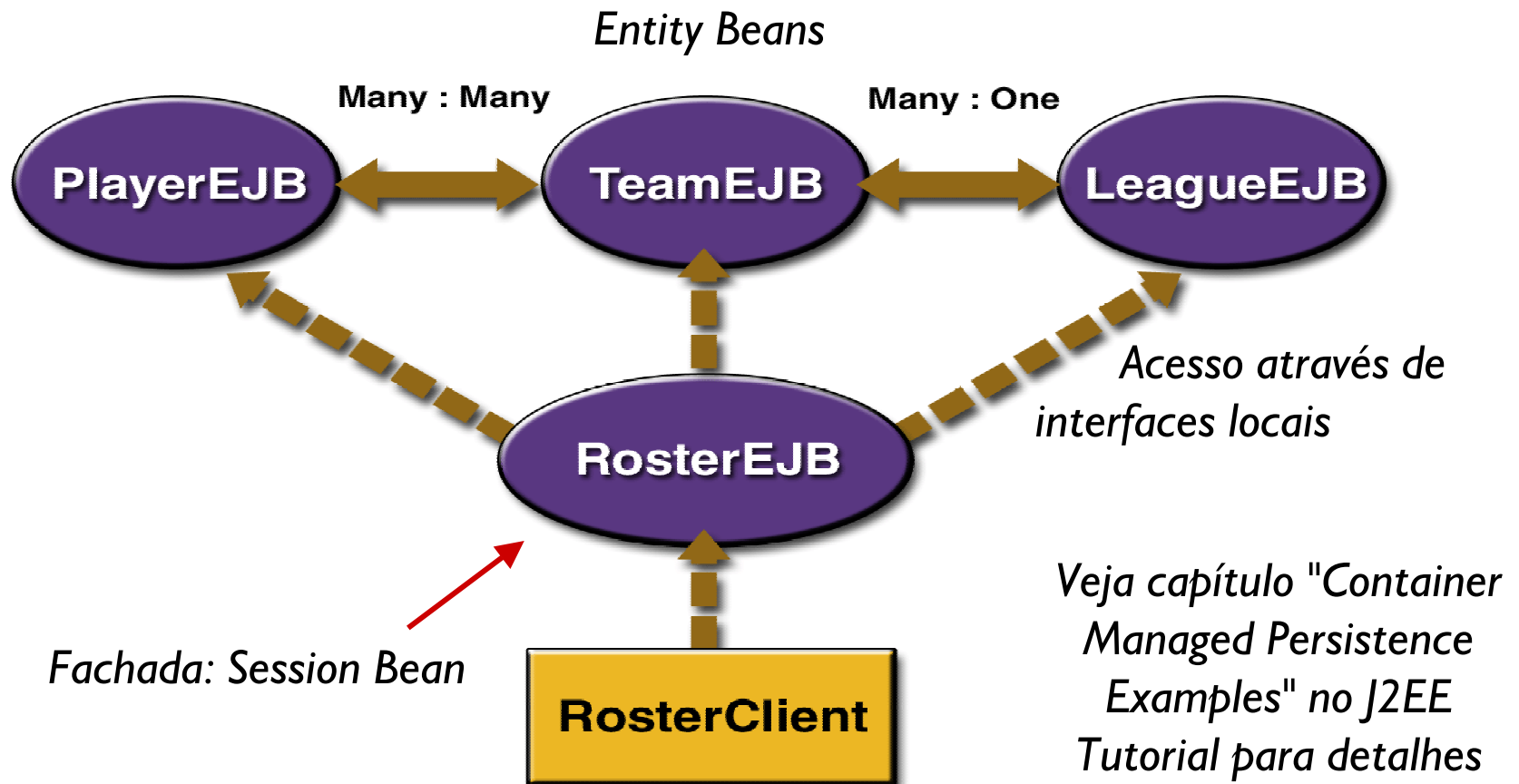
*Campos CMP* →

*Campos do banco de dados*

Pode-se ainda redefinir o mapeamento de tipos especificado em no arquivo de configuração *standardjbosscmp-jdbc.xml*

# Exemplo com relacionamentos CMR

- Este exemplo utiliza relacionamentos gerenciados pelo container (CMR) que requerem
  - Declaração de métodos de acesso ao relacionamento no Bean
  - Especificação dos relacionamentos no deployment descriptor



# Declaração de relacionamentos no Bean

## ■ *PlayerBean*

```
public abstract Collection getTeams();           // many teams
public abstract void setTeams(Collection teams);
```

## ■ *LeagueBean*

```
public abstract Collection getTeams();           // many teams
public abstract void setTeams(Collection teams);
```

## ■ *TeamBean*

```
public abstract Collection getPlayers();         // many players
public abstract void setPlayers(Collection players);
public abstract LocalLeague getLeague();         // one league
public abstract void setLeague(LocalLeague league);
```

# Relacionamentos no DD para TeamBean

```
<relationships>
  <ejb-relation>
    <description>League-Team</description>
    <ejb-relationship-role>
      <ejb-relationship-role-name>LeagueEJB</ejb-relationship-role-name>
      <multiplicity>One</multiplicity>
      <relationship-role-source>
        <ejb-name>LeagueEJB</ejb-name>
      </relationship-role-source>
      <cmr-field>
        <cmr-field-name>teams</cmr-field-name>
        <cmr-field-type>java.util.Collection</cmr-field-type>
      </cmr-field>
    </ejb-relationship-role>
    <ejb-relationship-role>
      <ejb-relationship-role-name>TeamEJB</ejb-relationship-role-name>
      <multiplicity>Many</multiplicity>
      <cascade-delete />
      <relationship-role-source>
        <ejb-name>TeamEJB</ejb-name>
      </relationship-role-source>
      <cmr-field><cmr-field-name>league</cmr-field-name></cmr-field>
    </ejb-relationship-role>
  </ejb-relation>
  ...
</relationships>
```

*campo de relacionamento*



# Como executar

- Do Sun J2EE Tutorial: RosterApp
  - Diretório *cap13/sun/*
- Configuração
  - Configure o arquivo *build.properties* com seu ambiente
- Inicialização
  - > *ant drop.table*
  - > *ant clean*
- Deployment
  - > *ant jboss.deploy* (observe as mensagens de criação das tabelas no JBoss)
- Execução do cliente
  - > *ant run jboss.client*
  - > *ant select.all* (faz *SELECT* na tabela)

- 1. Monte o exemplo RosterApp usando o deploytool da Sun (veja roteiro passo-a-passo em CMP.html)
- 2. Implemente o exemplo Account do capítulo 11 com CMP
- 3. Implemente o exemplo Enroller do capítulo 11 com CMP

## Exercício extra

- 4. O JBoss do laboratório está configurado para funcionar com o banco cloudscape. Configure-o para funcionar com o Oracle 8i\* instalado na rede
  - Copie o arquivo oracle-service.xml fornecido para o diretório deploy do JBoss
  - Faça as alterações indicadas no arquivo jbosscmp-jdbc.xml.
  - Escolha o prefixo fornecido para seus PK. Implante e rode o exercício 2 (A tabela Contas já está instalada no Oracle).

- *CMP é uma solução que permite a criação de componentes que são 100% lógica de negócio*
  - *Implementação de persistência resolvida pelo container*
  - *Desenvolvedor se concentra na modelagem e lógica de negócios*
- *Ainda não é possível, sem recorrer a extensões proprietárias, implementar qualquer modelo entidade-relacionamento usando CMP e CMR*
  - *Pode-se usar BMP em parte da aplicação*
  - *Pode-se usar extensões ao EJB-QL nos deployment descriptors (XML) e substituí-los no futuro (usando XSLT)*
  - *EJB 2.1 (2003) resolve principais limitações.*
- *Vimos só a ponta do iceberg. Separação de responsabilidades do CMP abre vários novos horizontes*
  - *Há vários design patterns relativos a CMP/CMR. Veja livro do Floyd Marinescu no CD (EJB Design Patterns) e Sun Blue Prints!*

- [1] Ed Roman. *Mastering EJB 2*. 2002
- [2] Richard Monson-Haefel. *Enterprise JavaBeans, 3rd. Edition*. O'Reilly, 2002
- [3] Dale Green, *J2EE Tutorial - Container Managed Persistence Examples*. Sun Microsystems, 2002



*helder@ibpinet.net*

***www.argonavis.com.br***