Java 2 Enterprise Edition

Session Beans

Helder da Rocha
www.argonavis.com.br

Sumário

- *Esta apresentação aborda os principais conceitos sobre Session Beans*
- *Demonstração interativa será feita através dos exemplos do capítulo 10*
  - *Demonstração de Stateful Session Beans (Stateless Session Beans já foram vistos no capítulo 1)*

Session Beans

- São objetos de processo de negócio
  - Implementam lógica de negócio, algoritmos, workflow
  - Representam ações
- Uma das principais diferenças entre Session Beans e Entity Beans é o seu escopo de vida
  - Um Session Bean dura uma sessão (do cliente)
- Sessão
  - tempo que o browser está aberto
  - tempo que um outro bean usa o session bean
  - tempo que uma aplicação remota está aberta
- Objetos transientes
  - Não tem seu estado armazenado em meio persistente

Tipos de Session Beans

- Clientes travam um diálogo com um bean (o diálogo é a interação entre um cliente e um bean)
  - Consiste de uma ou mais chamadas entre cliente e bean
  - Dura um processo de negócios para o cliente
- Os dois tipos de session beans modelam tipos diferentes de diálogos
  - **Stateful** Session Beans modelam diálogos consistem de várias requisições onde certas requisições podem depender do estado de requisições anteriores
  - **Stateless** Session Beans modelam diálogos que consistem de apenas uma requisição

Stateless Session Beans

- *Como stateless session beans não mantém informação de estado do diálogo, todas as instâncias do mesmo bean são equivalentes e indistiguíveis*
  - *Não importa que chamou o bean no passado*
  - *Qualquer instância disponível de um session bean pode servir a qualquer cliente*
- *Session Beans podem ser guardados em um pool, reutilizados e passados de um cliente para outro em cada chamada*

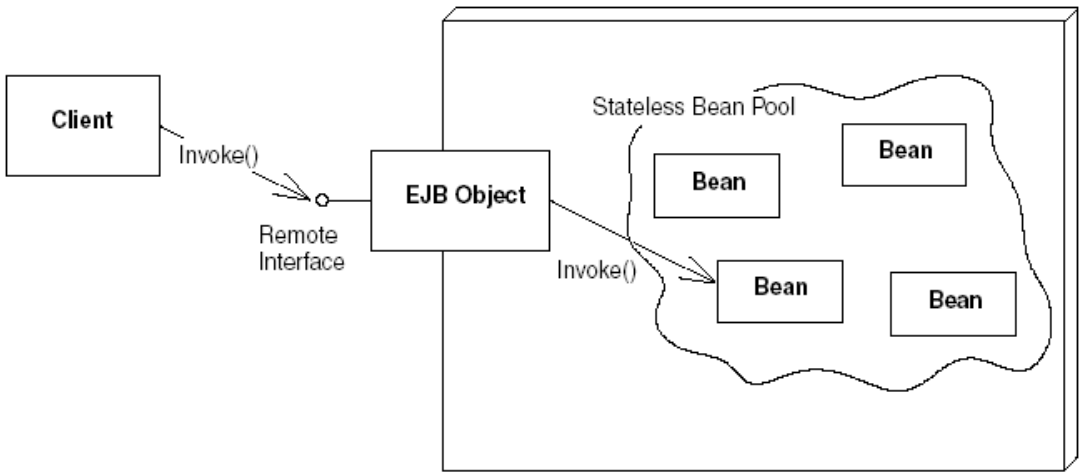

Figure 4.1 Stateless session bean pooling.

Fonte: [1]

Stateful Session Beans

- Quando um cliente chama um método de um bean, ele está iniciando um diálogo
  - O estado do diálogo deve ser mantido para a próxima requisição
  - Container não pode fazer o mesmo tipo de pooling que faz com stateless session beans
- Passivação e ativação
  - Solução para o problema de pooling
  - Dados do bean são armazenados em meio persistente durante a passivação e recuperados na ativação
  - Permite manter poucas instâncias no ar e vários clientes
  - Estratégia comum: LRU - Least Recently Used - Se container precisar de recursos, beans menos usados serão passivados. Logo que receberem uma requisição, serão reativados
- Objetos são serializados, portanto, fazem parte do estado do diálogo apenas objetos e variáveis não transientes

Passivação

- *Antes da passivação, método ejbPassivate() é chamado*

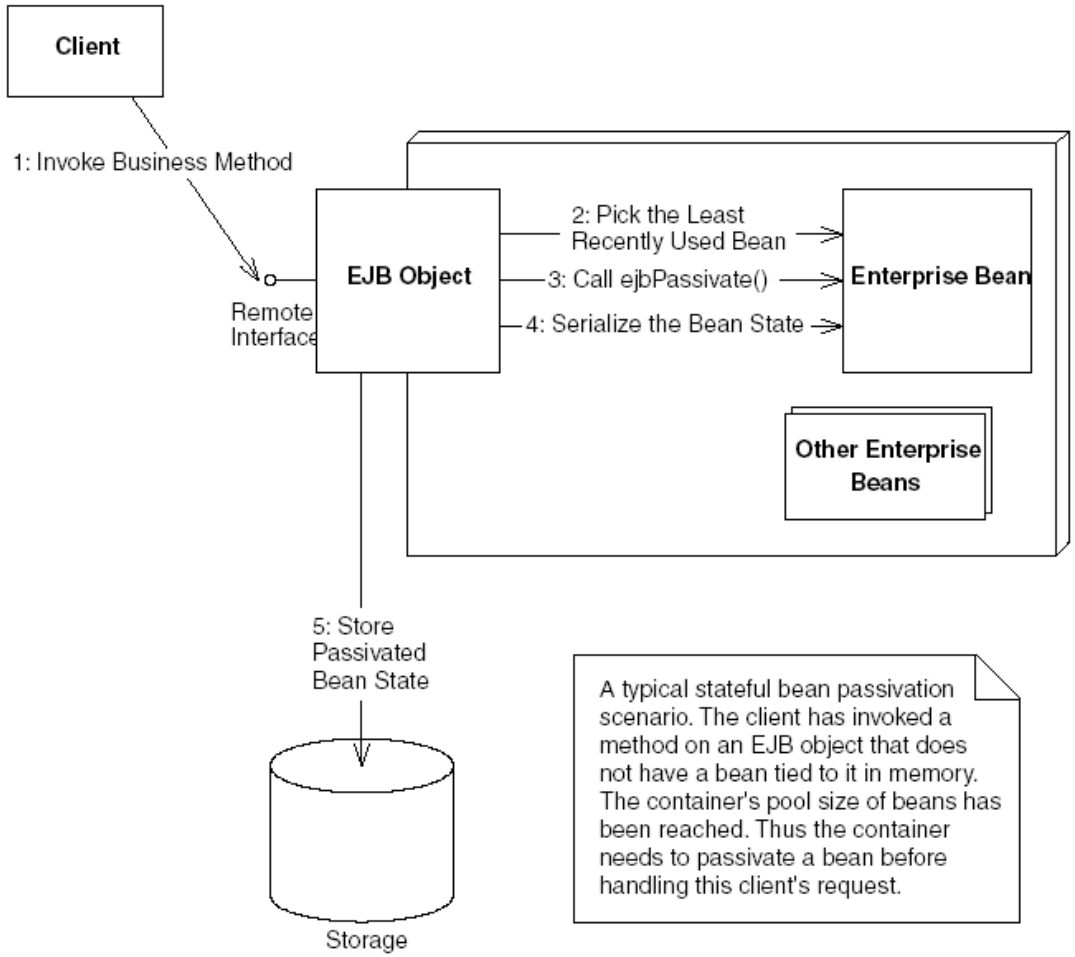

Figure 4.2 Passivation of a stateful bean.

Fonte: [1]

Ativação

- *ejbActivate()* é chamado logo após a ativação

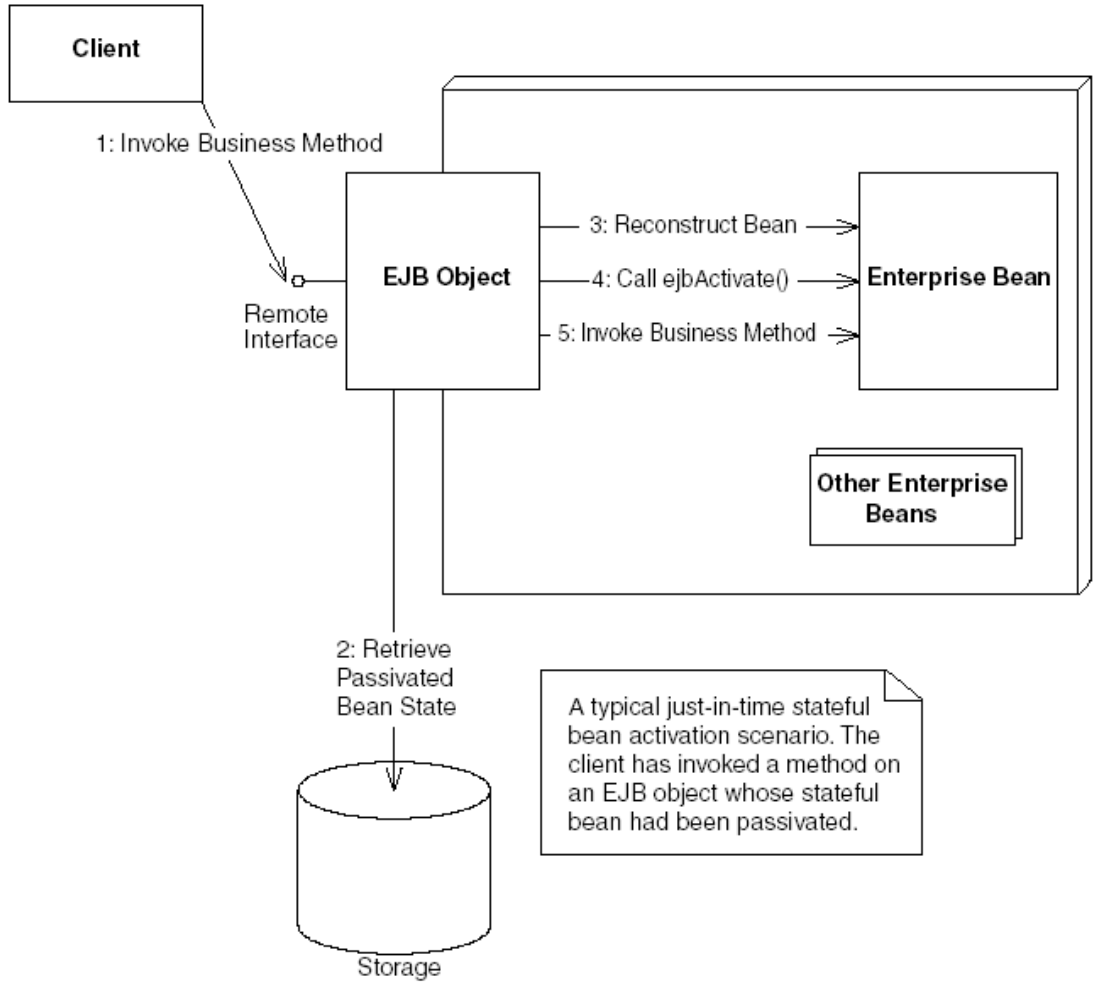

Figure 4.3 Activation of a stateful bean.

Fonte: [1]

Métodos da interface SessionBean

- *void setSessionContext(SessionContext ctx)*
  - *Associa bean com contexto da sessão*
  - *O contexto pode ser usado para obter referências para o interceptador e home do bean, se necessário*
- *void ejbCreate()*
  - *Stateless: só pode haver um, sem argumentos*
  - *Stateful: pode haver vários,com argumentos diferentes*
  - *Realiza a inicialização do bean*
- *void ejbRemove()*
  - *Chamado antes de liberar recursos e remover o bean da memória*
- *void ejbPassivate()*
  - *chamado antes de fazer o swap do bean para o disco*
  - *Session: não utiliza*
- *void ejbActivate()*
  - *chamado depois que estado do bean é recuperado do disco*
  - *Session: não utiliza*

Exemplos de Stateful Session Bean

- Nós já vimos exemplos de stateless session beans em exemplos anteriores
- Veja exemplo em cap10
  - mejb2: CountBean (comentado)
  - sun: CheckerBean e CartBean
- Para executar use o Ant (configure build.properties)
  - `ant jboss.deploy` (cria o JAR e copia para o deploy no JBoss)
  - `ant run.jboss.client` (executa um cliente de aplicação)
- Execução
  - Para demonstrar o efeito ejbActivate() e ejbPassivate(), o bean pool foi reduzido artificialmente (usando jboss.xml) para uma capacidade máxima de 2 beans
  - Como o cliente cria mais de dois beans, o container terá que passivar e depois ativar os beans (veja na saída da execução do JBoss quando cada método é chamado)

Interfaces Remote e Home

```
package examples;

public interface Count extends javax.ejb.EJBObject {

  public int count() throws java.rmi.RemoteException;

}
```

```
package examples;

public interface CountHome extends javax.ejb.EJBHome {

  Count create(int val)
        throws java.rmi.RemoteException, CreateException;

  Count create()
        throws java.rmi.RemoteException, CreateException;

}
```

Enterprise JavaBean

```
package examples;
public class CountBean implements javax.ejb.SessionBean {
    private SessionContext sessionContext;
    public int val;

    public int count() {
        System.out.println("count() chamado");
        return ++val;
    }
    public void ejbCreate(int val) throws CreateException {
        this.val = val;
        System.out.println("ejbCreate(val) chamado");
    }
    public void ejbCreate() throws CreateException {
        this.val = 0;
        System.out.println("ejbCreate() chamado");
    }
    public void ejbRemove() {
        System.out.println("ejbRemove() chamado");
    }
    public void ejbActivate() {
        System.out.println("ejbActivate() chamado");
    }
    public void ejbPassivate() {
        System.out.println("ejbPassivate() chamado");
    }
    public void setSessionContext(SessionContext ctx) {
         this.sessionContext = ctx;
    }
}
```

Deployment Descriptor

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

<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>CountEJB</ejb-name>
      <home>examples.CountHome</home>
      <remote>examples.Count</remote>
      <ejb-class>examples.CountBean</ejb-class>
      <session-type>Stateful</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

Ciclo de vida: Stateless Session Bean

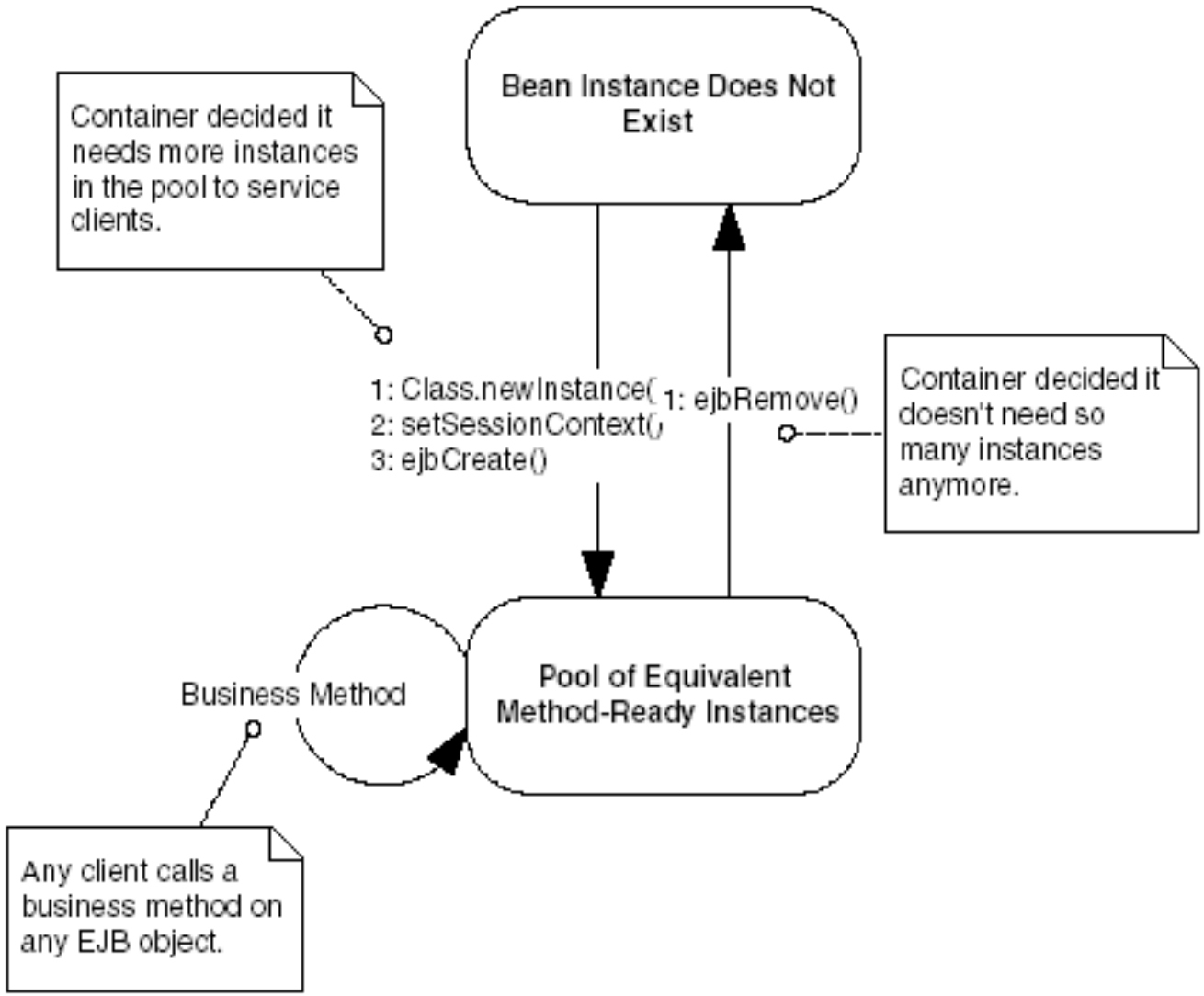

Figure 4.4 The life cycle of a stateless session bean.

Fonte: [1]

Ciclo de vida: Stateful Session Bean

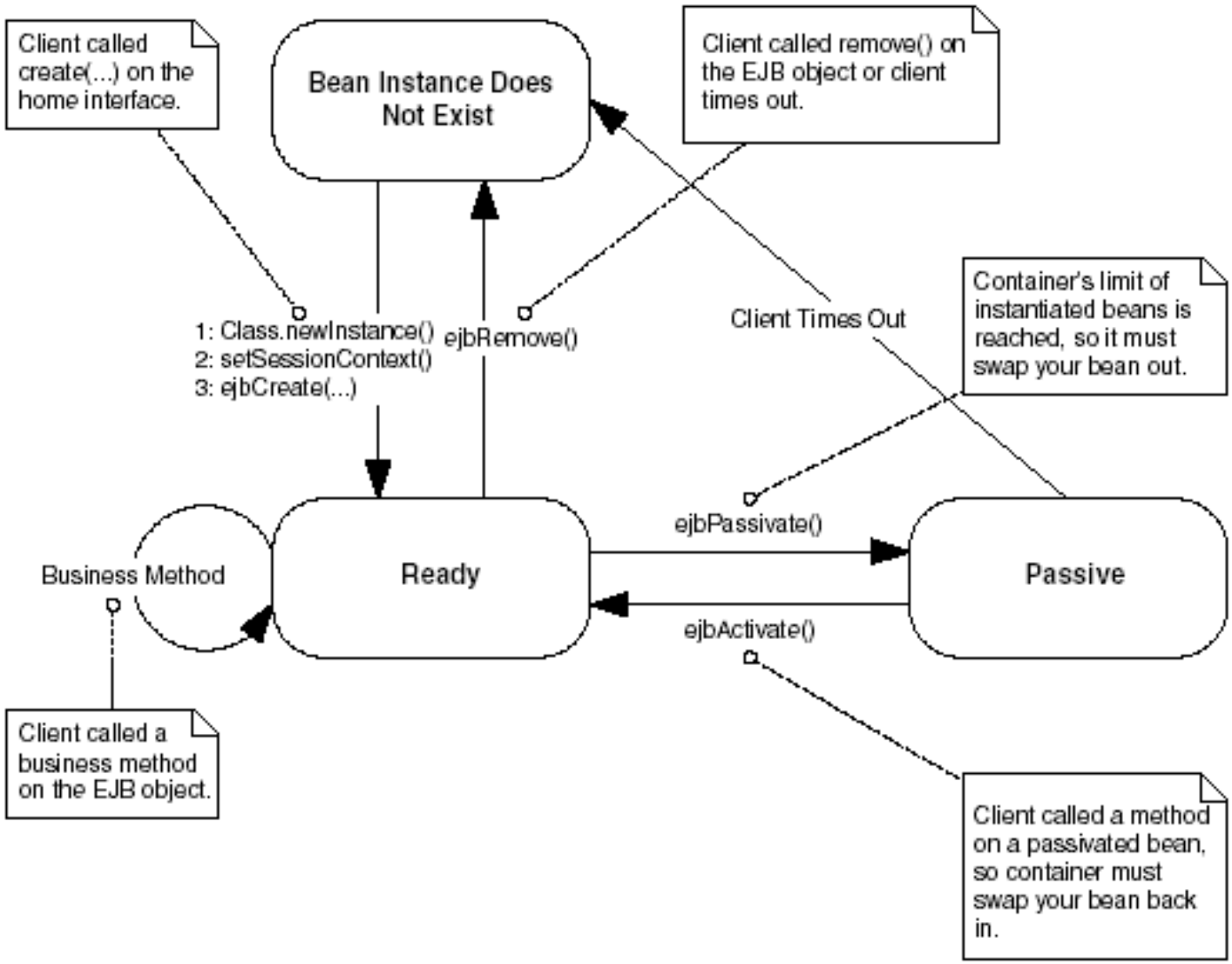

Figure 4.5 Life cycle of a stateful session bean.

Fonte: [1]

Exercício

- 1. Comparação Stateful-Stateless
  - Remova o método create() com argumentos do exemplo do CountBean (deixando apenas o sem argumentos). Adapte o cliente para que ele use apenas create() e não create(valor)
  - Execute a aplicação
  - Mude, no deployment descriptor, o bean para Stateless e rode a aplicação novamente.
- 2. Veja cap10/exercicio
  - A partir dos arquivos NomesBase.java e Nomes.java (este uma interface Remote), construa um Stateful Session Bean (Home, Bean, ejb-jar.xml) que funcione com o cliente NomesClient fornecido.
  - Utilize o jboss.xml fornecido

Fontes

[1] *Ed Roman, Mastering EJB 2, 2002, Capítulo 4.*
[2] *Dale Green. Session Beans. J2EE Tutorial, Sun, (veja CD)*

helder@ibpinet.net

www.argonavis.com.br