



# *Sistemas Distribuídos na WEB*

*(Plataformas para Aplicações Distribuídas)*

## **J2EE (Java 2 Enterprise Edition)**

**Exemplo Session Bean (com estado)**



## *Session Bean (com estado)*

- ✎ Criaremos um Bean de cesta de compras, *CartEJB*
- ✎ Você concorda que uma cesta de compras mantida no servidor não poderia ser stateless?
- ✎ O bean tem a seguinte funcionalidade:
  - Adicionar um livro
  - Remover um livro
  - Examinar conteúdo

## Session Bean (com estado)

✎ Precisamos fazer o seguinte:

- Uma interface Home (CarHome)
- Uma interface Remote (Cart)
- Uma classe de implementação (CartBean)

## Session Bean (com estado)

✎ A Remote Interface

- Define os Business Methods

```
import java.util.*;
import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Cart extends EJBObject {
    public void addBook(String title) throws RemoteException;
    public void removeBook(String title) throws BookException, RemoteException;
    public Vector getContents() throws RemoteException;
}
```

## Session Bean (com estado)

### 🔗 A Home Interface

- Define os métodos de criação que podem ser chamados pelo cliente remoto

```
import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface CartHome extends EJBHome {
    Cart create(String person) throws RemoteException, CreateException;
    Cart create(String person, String id) throws RemoteException, CreateException;
}
```

Sistemas Distribuídos 2007  
Prof. Carlos Paes

5

## Session Bean (com estado)

### 🔗 A Home Interface

- Define os métodos de criação que podem ser chamados pelo cliente remoto

```
import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface CartHome extends EJBHome {
    Cart create(String person) throws RemoteException, CreateException;
    Cart create(String person, String id) throws RemoteException, CreateException;
}
```

Sistemas Distribuídos 2007  
Prof. Carlos Paes

6

## Session Bean (com estado)

### ☀ Classe de Implementação

- Algumas regras a obedecer:
- Implementa a interface *SessionBean*
- A classe deve ser "public" (devido à introspecção do Java)
- A classe não pode ser *abstract* ou *final*

## Session Bean (com estado)

### ☀ Classe de Implementação

- Para cada método *create* na interface *Home*, a classe de implementação deve implementar um método *ejbCreate*
  - Lembre que essa classe não implementa a *Remote interface*
  - Não é chamada pelo cliente
  - Só é chamada pelo container
- Implementa os *Business Methods* da *Remote Interface*
  - Contém um construtor público sem parâmetro
  - Não deve definir o método *finalize*

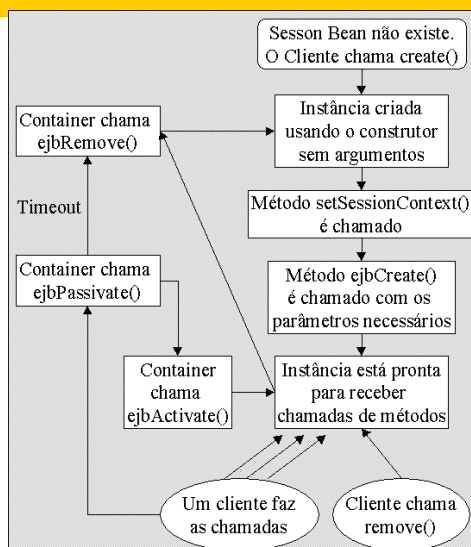
### ☀ Código Fonte

### ☀ Classe de Apoio

## Session Bean (com estado)

- ☀ Há vários métodos de callback
  - Chamados pelo container em momentos apropriados
- ☀ Foram deixados vazios porque não são necessários aqui
- ☀ Para entender quando os callbacks são chamados, examine a figura abaixo que representa o ciclo de vida de uma Stateful Session Bean

## Session Bean (com estado)



## Session Bean (com estado)

### ☛ Os Business Methods

- Examine a implementação dos Business methods acima
- Eles seriam chamados assim pelo cliente:

```
Cart shoppingCart = home.create("Duke DeEarl", "123");  
...  
shoppingCart.addBook("The Martian Chronicles");  
shoppingCart.removeBook("Alice In Wonderland");  
bookList = shoppingCart.getContents();  
...
```

## Session Bean (com estado)

### ☛ Você pode lançar suas exceções nos métodos: eles serão repassados para o cliente

- Exemplo: *BookException*
- Exceções *EJBException* indicam erros de sistema

### ☛ São colocadas dentro de uma *RemoteException* que o cliente captura

### ☛ Outras exceções (como *BookException*) não são colocadas numa *RemoteException*

## Session Bean (com estado)

### ✱ Criação de um cliente

- Vamos fazer um cliente simples (*Application Client*)

