Especialista en Aplicaciones y Servicios Web con Java Enterprise

Enterprise JavaBeans

Sesión 4:

Beans de sesión con estado

Beans de sesión con estado

- La única diferencia con los beans sin estado es que definen campos (variables de instancia) en los que se almacenan datos proporcionados en las llamdas del cliente
- Se suelen usar para implementar métodos de negocios que requieren multiples pasos de ejecución
- El estado del bean dura el tiempo de sesión y no es persistente
- Ejemplo típico: carrito de la compra

El bean Cart

- Bean de sesión con estado que implementa un carrito de la compra en una librería
- Los clientes podrán llamar a los métodos:
  - `addBook(String title)`
  - `removeBook(String title)`
  - `ArrayList[Strings] getBookTitles()`
  - `double getOrderPrice()`

La clase CartBean

- Implementa la interfaz `SessionBean`
- Creación de instancias: `ejbCreate(args)`
- Uso del bean: los métodos de negocio del bean:

```
void addBook(String title)
void removeBook(String title)
ArrayList[Strings] getBookTitles()
double getOrderPrice()
```

Requisitos de la clase Bean

- Implementa la interfaz SessionBean
- La clase se define como public
- No puede definirse como abstract ni final
- Implementa uno o más métodos ejbCreate (corresponden a los que se definen en la interfaz home)
- Implementa los métodos remotos definidos en la interfaz componente del bean
- Vamos a verlo en detalle, aplicado al bean CartBean

Los campos del bean

- Guardan el estado del cliente
- Se inicializan en el método ejbCreate y pueden copiarse en ellos los argumentos del método
- Se modifican en los métodos de negocio
- En el caso de CartBean son:

```
String customerName
String customerId
Collection contents
```

Los métodos ejbCreate

- Los llama el contenedor EJB cuando el EJBHome recibe una petición de creación de una nueva instancia
- Se definen también en la interfaz home
- Inicializan el estado del bean
- Requisitos:
  - Deben ser `public` y no pueden ser `static` ni `final`
  - El tipo de retorno debe ser `void`
  - Si el bean permite acceso remoto los argumentos deben ser tipos legales RMI

Los métodos de negocio

- La única diferencia con los métodos de negocio de los EJB sin estado es que pueden acceder a los campos del bean, y modificarlos

CartBean (1)

```
public class CartBean implements SessionBean {
   private static final long serialVersionUID = 1L;
   String customerName;
   String customerId;
   Collection<BookVO> contents;
   double precio;

   public void ejbCreate(String person) throws CreateException {
      if (person == null) {
         throw new CreateException("Null person not allowed.");
      } else {
         customerName = person;
      }
      customerId = "0";
      contents = new ArrayList<BookVO>();
      System.out.println("CartBean creado");
   }

   public void ejbCreate(String person, String id) throws CreateException {
      if (person == null) {
         throw new CreateException("Null person not allowed.");
      } else {
         customerName = person;
      }
      contents = new ArrayList<BookVO>();
      System.out.println("CartBean creado");
   }
```

CartBean (2)

```
public void addBook(BookVO book) throws BookException {
      contents.add(book);
      System.out.println("Añadido libro: " + book.getTitle());
   }

   public void removeBook(BookVO book) throws BookException {
      boolean result = false;
      String title = book.getTitle();
      Iterator it = contents.iterator();
      while (it.hasNext()) {
         BookVO bookIt = (BookVO) it.next();
         if (title.equals(bookIt.getTitle())) {
            it.remove();
            result = true;
            System.out.println("Borrado libro: " + book.getTitle());
            break;
         }
      }
      if (result == false) {
         throw new BookException(title + " not in cart.");
      }
   }

   public ArrayList<BookVO> getAllBooks() {
      return (ArrayList<BookVO>) contents;
   }
```

CartBean (3)

```
public double getOrderPrice() {
      double price = 0.0;
      Iterator it = contents.iterator();
      while (it.hasNext()) {
         BookVO book = (BookVO) it.next();
         price = price + book.getPrice();
      }
      return price;
   }

   public CartBean() {}

   public void ejbRemove() {}
   public void ejbActivate() {}
   public void ejbPassivate() {}
   public void setSessionContext(SessionContext sc) {}
}
```

Interfaces

CartHome.java

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

public interface CartHome extends EJBHome {

    public Cart create(String person) throws RemoteException, CreateException;
    public Cart create(String person, String id) throws RemoteException,
                                                        CreateException;
}
```

Cart.java

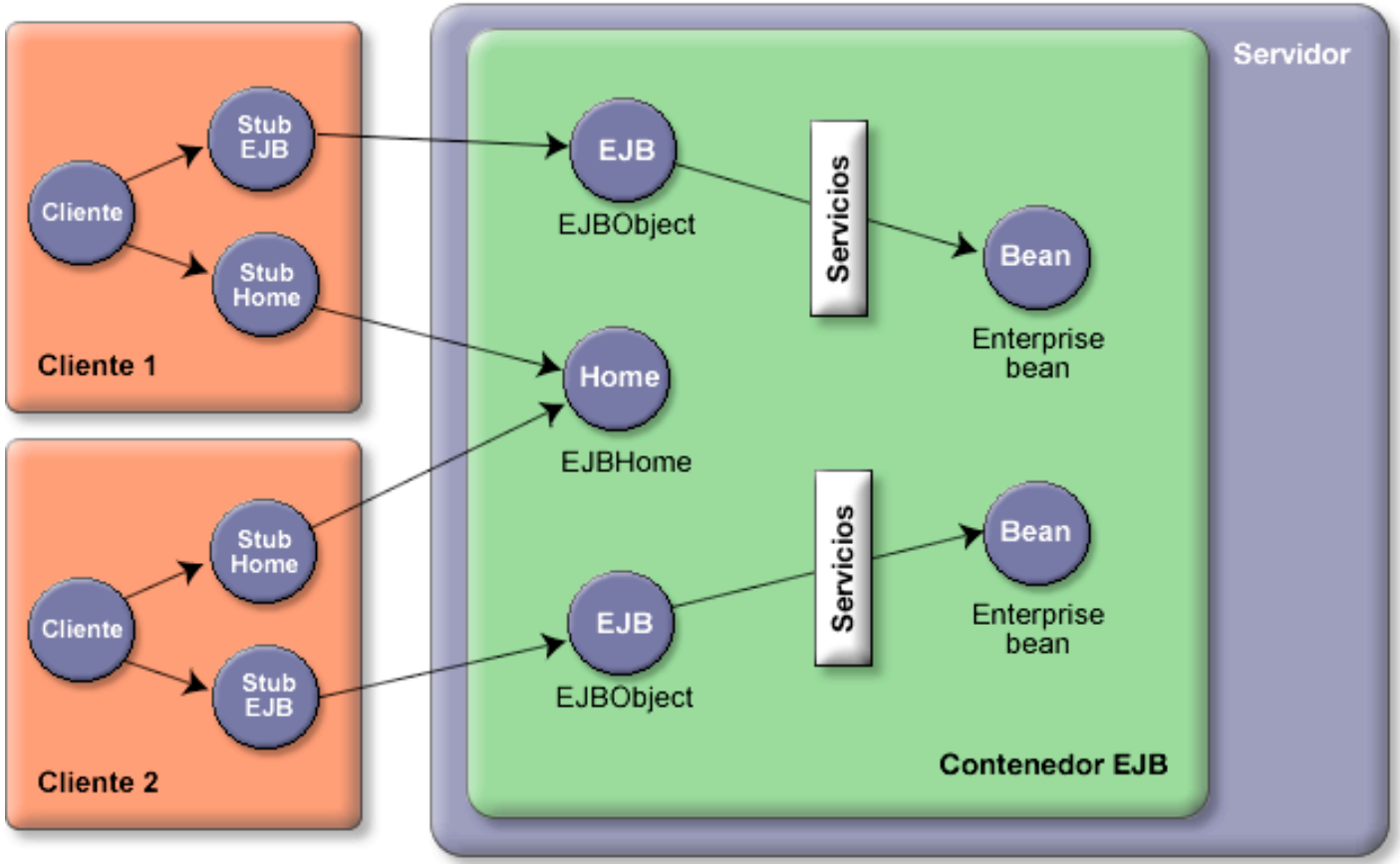
```
import java.util.*;
import javax.ejb.EJBObject;
import BookVO;

import java.rmi.RemoteException;

public interface Cart extends EJBObject {

   public void addBook(BookVO book) throws BookException, RemoteException;
   public void removeBook(BookVO book) throws BookException, RemoteException;
   public ArrayList<BookVO> getAllBooks() throws RemoteException;
   public double getOrderPrice() throws RemoteException;
}
```

Arquitectura beans de sesión con estado



¿Preguntas?