Especialista en Aplicaciones y Servicios Web con Java Enterprise

Enterprise JavaBeans

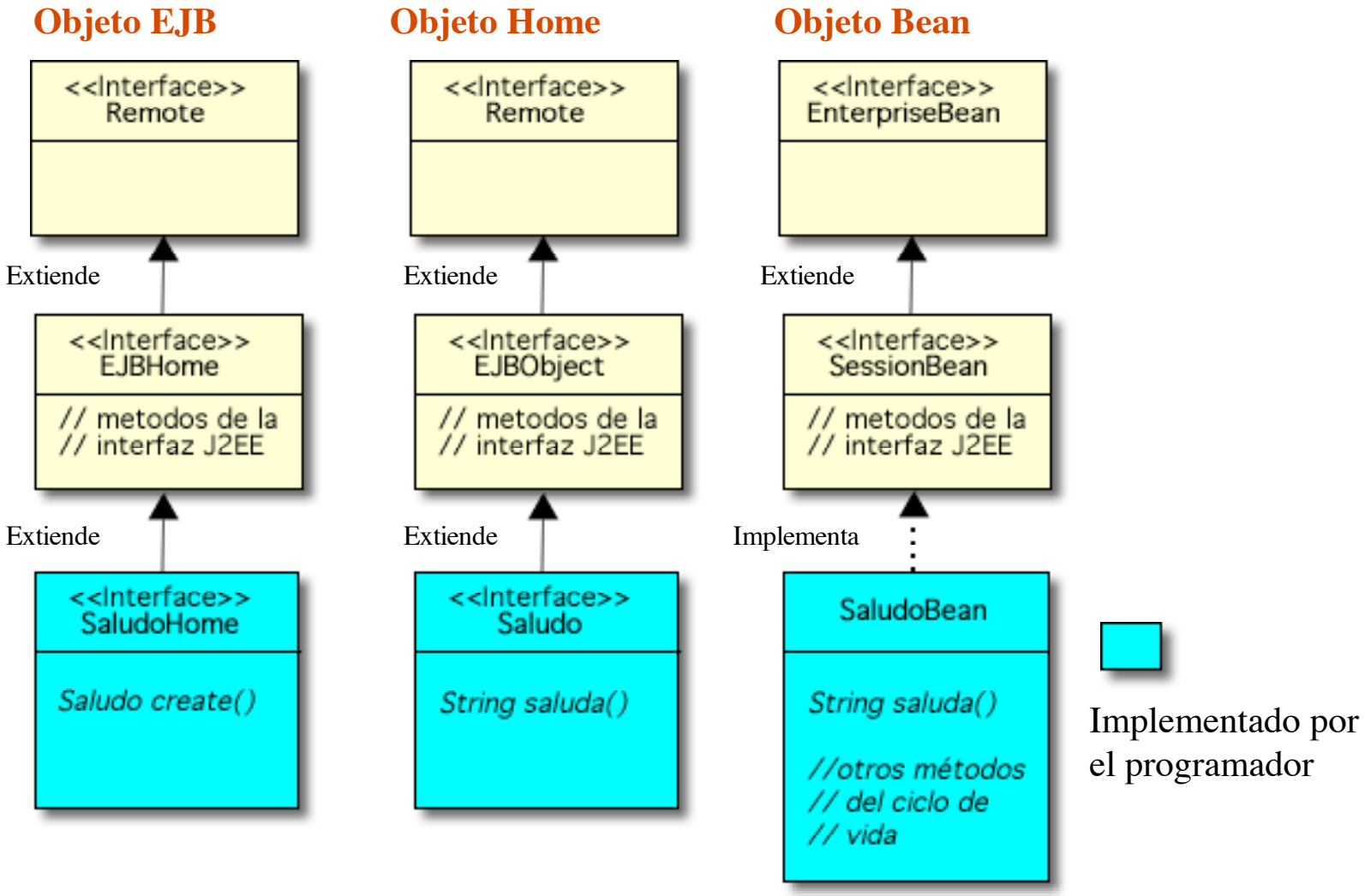
Sesión 2:

Implementación de EJBs de sesión

Implementación componente EJB

- Un componente EJB [Saludo] se define mediante tres objetos:
- Objeto componente [Saludo.java]: define los métodos del componente accesibles desde el cliente
- Objeto home [SaludoHome.java]: define los métodos de creación y borrado de componentes accesibles desde el cliente (*fábrica* de componentes)
- Objeto bean [SaludoBean.java]: define la implementación de los métodos del componente y el código de inicialización y borrado de los componentes

Diagrama UML de un enterprise bean



Desarrollo de un bean de sesión

- 1. Escribir las interfaces componente y home
- 2. Escribir la clase bean con la implementación de los métodos de negocio
- 3. Crear el descriptor de despliegue ejb-jar.xml
- 4. Crear el fichero EJB JAR
- 5. Desplegar el bean en el contenedor
- 6. Usar el bean desde los clientes

1. Escribir la interfaz componente

- La interfaz componente debe extender la interfaz `javax.ejb.EJBObject`
- En la interfaz se deben definir:
  - Métodos de negocio del bean

1. Clase Saludo.java

```
package es.ua.jtech.ejb.beans;

import javax.ejb.EJBObject;
import java.rmi.RemoteException;

public interface Saludo extends EJBObject {
   public String saluda() throws RemoteException;
   public SaludoTO getSaludo(int numDatos) throws RemoteException;
}
```

1. Escribir la interfaz home

- La interfaz componente debe extender la interfaz `javax.ejb.EJBHome`
- En la interfaz se deben definir:
  - Métodos de creación del bean

1. Clase SaludoHome.java

```
package es.ua.jtech.ejb.beans;

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

public interface SaludoHome extends EJBHome {
   public Saludo create() throws RemoteException, CreateException;
}
```

2. Escribir la clase bean

- La clase bean debe implementar la interfaz `javax.ejb.SessionBean`, `EntityBean` o `MessageBean`
- En la clase se deben implementar:
  - Funciones de la interfaz ejb relacionadas con su ciclo de vida (`ejbActivate`, `ejbPassivate`, ...)
  - Funciones de la interfaz componente: métodos de negocio
  - Funciones de la interfaz home: creación del bean

```
public class SaludoBean implements SessionBean {

   private static final long serialVersionUID = 1L;
   private String[] saludos = { "Hola, que tal?", "Cuanto tiempo sin verte",
         "Que te cuentas?", "Me alegro de volver a verte" };

   public String saluda() {
      int random = (int) (Math.random() * saludos.length);
      return saludos[random];
   }

   public SaludoTO getSaludo(int numDatos) {
      int random = (int) (Math.random() * saludos.length);
      String saludo = saludos[random];

      Date fecha = new Date();
      ArrayList<Integer> datos = new ArrayList<Integer>();

      for (int i = 0; i < numDatos; i++) {datos.add(i);}

      SaludoTO miSaludo = new SaludoTO(saludo, fecha, datos);
      return miSaludo;
   }

   public void setSessionContext(SessionContext arg0) throws EJBException {}
   public void ejbCreate() throws EJBException {}
   public void ejbRemove() throws EJBException {}
   public void ejbActivate() throws EJBException {}
   public void ejbPassivate() throws EJBException {}

}
```

Clase SaludoTO

```
package es.ua.jtech.ejb.beans;
import java.io.Serializable;
import java.util.Date;
import java.util.List;

public class SaludoTO implements Serializable {
   private static final long serialVersionUID = 1L;
   private String mensaje;private Date fecha;
   private List<Integer> datos;

   public SaludoTO(String mensaje, Date fecha, List<Integer> datos) {
      this.mensaje = mensaje;
      this.fecha = fecha;
      this.datos = datos;
   }
   public String getMensaje() {
      return this.mensaje;
   }
   public Date getFecha() {
      return this.fecha;
   }
   public List<Integer> getDatos() {
      return this.datos;
   }
}
```

3. Fichero de despliegue

- Define las características del bean que debe gestionar el contenedor EJB
- Fichero XML: `ejb-jar.xml`
- Las más importantes:
  - Transacciones
  - Seguridad
  - Nombre del bean
  - Nombre JNDI del bean
- El nombre JNDI del bean se define en un fichero descriptor no estándar que depende del contenedor (`weblogic-ejb-jar.xml`)

3. Fichero ejb-jar.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
        <session>
            <ejb-name>SaludoBean</ejb-name>
            <home>es.ua.jtech.ejb.beans.SaludoHome</home>
            <remote> es.ua.jtech.ejb.beans.Saludo</remote>
            <ejb-class> es.ua.jtech.ejb.beans.SaludoBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
        </session>
    </enterprise-beans>
</ejb-jar>
```

3. Fichero weblogic-ejb-jar.xml

- Nombre JNDI remoto

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-ejb-jar xmlns="http://www.bea.com/ns/weblogic/90"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.bea.com/ns/weblogic/90
http://www.bea.com/ns/weblogic/90/weblogic-ejb-jar.xsd">
    <weblogic-enterprise-bean>
        <ejb-name>SaludoBean</ejb-name>
        <jndi-name>SaludoBean</jndi-name>
   </weblogic-enterprise-bean>
</weblogic-ejb-jar>
```

4. Estructura EJB JAR

- El fichero EJB JAR es un fichero JAR que contiene la siguiente estructura:

```
sesion1-beans/.../beans/Saludo.class
sesion1-beans/.../beans/SaludoBean.class
sesion1-beans/.../beans/SaludoHome.class
sesion1-beans/META-INF/ejb-jar.xml
sesion1-beans/META-INF/weblogic-ejb-jar.xml
```

5. Desplegar el bean

- Herramientas específicas del servidor de aplicaciones
- Se suele usar
  - Ant
  - Consola de administración del servidor de aplicaciones
  - Opción integrada en el entorno de desarrollo

6. Usar el bean en un cliente

- Obtener el objeto EJBHome (realmente un stub del objeto EJBHome) mediante JNDI
  - Obtener el contexto inicial
  - Realizar loockup
  - Realizar narrowing
- Mediante el objeto home obtener un objeto EJBObject (realmente un stub del objeto EJBObject)
- Invocar un método de negocio a través del objeto EJBObject

Interfaces locales

- ¿Tiene sentido usar interfaces remotas cuando se está accediendo al enterprise bean desde la misma máquina virtual (en una aplicación Web) por ejemplo?
- Interfaz local:
  - Se mantiene el EJB Object que hace de "cortafuegos". Pero ahora ya no es remota.
- Interfaces locales:
  - SaludoHomeLocal.java
  - SaludoLocal.java

Implementación de interfaces locales

```
package es.ua.jtech.ejb.beans;

import javax.ejb.EJBLocalObject;

public interface SaludoLocal extends EJBLocalObject {
   public String saluda();
   public SaludoTO getSaludo(int numDatos);
}
```

```
package es.ua.jtech.ejb.beans;

import javax.ejb.CreateException;
import javax.ejb.EJBLocalHome;

public interface SaludoLocalHome extends EJBLocalHome {
   public SaludoLocal create() throws CreateException;
}
```

Descriptores de despliegue locales

ejb-jar.xml

```
<enterprise-beans>
    <session>
        <ejb-name>SaludoBean</ejb-name>
        <home>es.ua.jtech.ejb.beans.SaludoHome</home>
        <remote>es.ua.jtech.ejb.beans.Saludo</remote>
        <local-home>es.ua.jtech.ejb.beans.SaludoLocalHome</local-home>
        <local>es.ua.jtech.ejb.beans.SaludoLocal</local>
        <ejb-class>es.ua.jtech.ejb.beans.SaludoBean</ejb-class>
        <session-type>Stateless</session-type>
        <transaction-type>Container</transaction-type>
    </session>
</enterprise-beans>
```

weblogic-ejb-jar.xml

```
<weblogic-enterprise-bean>
    <ejb-name>SaludoBean</ejb-name>
    <jndi-name>SaludoBean</jndi-name>
    <local-jndi-name>SaludoBeanLocal</local-jndi-name>
</weblogic-enterprise-bean>
```

Acceso a la interfaz local

- Para acceder a la interfaz local se usa el nombre JNDI local del bean
- Se accede al contexto JNDI local
- No es necesario hacer un narrow del objeto Home ni del objeto interfaz, basta con un casting
- En el ejercicio veremos un ejemplo de acceso desde una página JSP y un bean.

¿Preguntas?