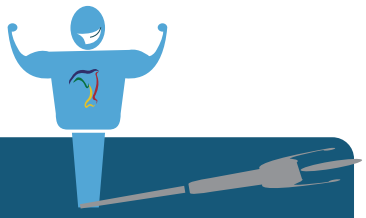
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

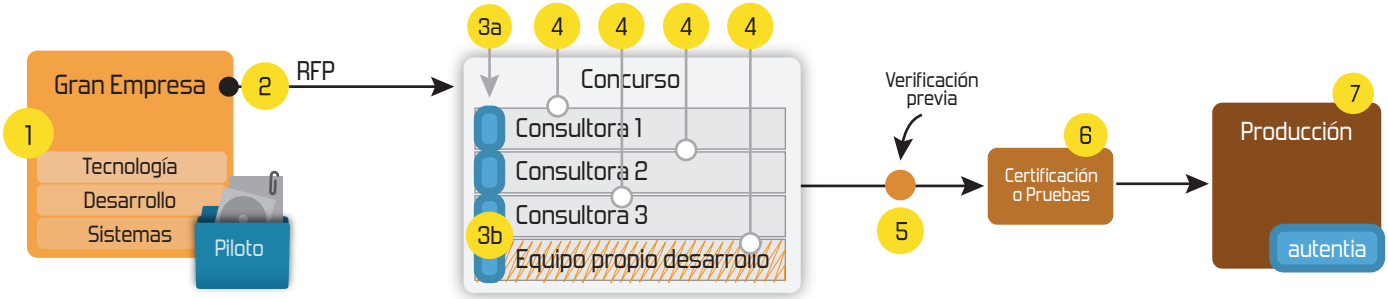
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

s en AdictosAlTrabajo: Java, J2EE, Visual C++, Linux, UML, OOP y

Table of Contents

Entity Beans en J2EE RI

Hemos comentado que a partir de ahora vamos a subir el nivel de los tutoriales .... pero pretendemos hacerlo poco a poco.

Esta vez, vamos a ampliar el tutorial anterior, realizando algunos cambios, para crear un ejemplo básico de un EJB de entidad, que desplegaremos en la implementación de referencia (RI).

El código aquí mostrado no debeis usarlo como referencia sobre buenas prácticas en el desarrollo JE22 porque en siguientes tutoriales nos centraremos precisamente en esto.... aplicar algunos patrones para ir dando forma a nuestras aplicaciones.

Básicamente, un EJB de entidad, tiene los mismos elementos que un EJB de sesión... aunque con algunos matices:

- Debemos crear un interfaz Home donde especificaremos como crear el EJB. En este caso es más importante porque probablemente sea el encargado de almacenar de modo persistente nuestro EJB.
- Un interfaz remoto, con los métodos que operan sobre los atributos privados
- Un Bean que posee los métodos obligatorios para que nuestro EJB se comporte correctamente (al implementar el interfaz EntityBean)

Hay dos modos de construir un EJB de entidad, haciendo que la persistencia la gestione el propio EJB o haciendo que la gestione automáticamente en contenedor. Hemos elegido el primer método para entender, un poco, que esta pasando por debajo cuando usemos el segundo tipo.

Realmente lo que queremos comprobar es como el contenedor gestiona los eventos ..... y crear nuestro esqueleto de referencia.

Creamos el interfaz Home del Entity Bean

Definimos un método create para inicializar en EJB y findByPrimaryKey para poder localizarlo (obligatorio). El método **findByRango** solo lo introducimos para futuros desarrollos

```
package ejbfacturas;

/**
 *
 * @author  Roberto Canales
 */

import java.rmi.*;
import javax.ejb.*;
import java.io.*;
import java.util.*;

import facturascomun.*;

/**
 * Interfaz Home para definir los métodos de creación de nuestro EJB
 *
```

```
 */
public interface iFacturaItemHome extends EJBHome
{
    // ver las diferencias entre las declaraciones del Home y del Bean
    public iFacturaItem create(Factura pFactura) throws CreateException, RemoteException;
    public Collection findByRango(int pOrigen,int pDestino) throws FinderException, RemoteExcep
    public iFacturaItem findByPrimaryKey(String pParam) throws FinderException, RemoteExceptior
}
```

Creamos el interfaz remoto

En él, definimos todos los métodos get/set para poder acceder a nuestros atributos

```
package ejbfacturas;

/**
 *
 * Interfaz remoto a nuestro EJB de Facturas
 *
 * @author  Roberto Canales
 */
import java.rmi.*;
import javax.ejb.*;
import facturascomun.*;
import java.util.*;

public interface iFacturaItem extends EJBObject
{
    public String getSConcepto() throws RemoteException;
    public void setSConcepto(String sConcepto) throws RemoteException;
    public String getSFecha() throws RemoteException;
    public void setSFecha(String sFecha) throws RemoteException;
    public String getSDireccion() throws RemoteException;
    public void setSDireccion(String sDireccion) throws RemoteException;
    public String getSTitular() throws RemoteException;
    public void setSTitular(String sTitular) throws RemoteException;
    public String getSCif() throws RemoteException;
    public void setSCif(String sCif) throws RemoteException;
    public String getIId() throws RemoteException;
    public void setIId(String iId) throws RemoteException;
    public double getDCandidad() throws RemoteException;
    public void setDCandidad(double dCandidad) throws RemoteException;
}
```

Creamos el Entity Bean

Separamos los eventos del EJB, de los métodos que realizan las lecturas y escritura, para mayor claridad.

Hemos forzado el código para la simulación de la recuperación de un elemento con clave primaria 22

```
/*
 * iFacturasItemBean.java
 *
 * Created on 20 de septiembre de 2003, 9:41
 *
 * Este EJB representa los elementos persistentes en la Base de Datos
```

```
 *
 */

package ejbfacturas;

/**
 * @author Roberto Canales
 */

import java.rmi.*;
import javax.ejb.*;
import java.sql.*;
import javax.sql.*;
import java.util.*;

import facturascomun.*;


public class iFacturaItemBean implements EntityBean
{
    private void depura(String cadena)
    {
        System.out.println("EJB iFacturaItemBean: " + cadena);
    }


    /**
    *    Declaración de miembros
    */
    private String iId;  // clave primaria de nuestra factura
    private String sTitular;
    private String sConcepto;
    private String sCif;
    private String sDireccion;
    private String sFecha;
    private double dCandidad;

    private EntityContext context = null;
    private Connection con = null;  // la usaremos para conectar a la BBDD


    /**
     * Método destinado a construir nuestro objeto
     *
     * Debe retornar la clave primaria del objeto creado
     */
    public String ejbCreate(Factura pFactura) throws CreateException
    {
        if (pFactura == null)
        {
            throw new CreateException("El Objeto retornado es nulo");
        }

        // asignamos las propiedades a los miembros de nuestra clase
        iId         = pFactura.getId();  // clave primaria de nuestra factura
        sTitular    = pFactura.getTitular();
        sConcepto   = pFactura.getConcepto();
        sCif        = pFactura.getCif();
        sDireccion  = pFactura.getDireccion();
        sFecha      = pFactura.getFecha();
        dCandidad   = pFactura.getCandidad();
```

```
        insertaFila();

        return iId;
    }

    /**
     * PosCreate del create construido
     */
    public void ejbPostCreate(Factura pFactura) throws CreateException
    {
        depura("Enviado post-create");
        // no es necesario código aunque si el método por haber escrito un create
    }

    /**
     * Método encargado de localizar un elemento por clave primaria
     *
     * Es un método obligatorio
     * Retorna la clave primaria de la entidad
     * Nunca se llama este método directamente.
     *
     */
    public String ejbFindByPrimaryKey(String pParam) throws FinderException
    {
        depura("Invocado buscar por id " + pParam);
        if (encuentroElemento(pParam) == false)
        {
              throw new ObjectNotFoundException();
        }

        return pParam;
    }

    /**
     * Metodo que verifica la existencia de un elemento concreto
     */
    private boolean encuentroElemento(String sParam)
    {
        // Este código lo usamos solamente para probar el concepto
        if(sParam.compareTo("22") == 0)
        {
            return true;  // solo nos interesa el valor cero
        }

        return false;
    }

    /**
     * Metodo que realiza la inserción
     * Lo separamos del create para desacoplar funcionalidades
     */
    void insertaFila() throws CreateException
    {
        depura("Insertamos Fila para id " + iId);
    }



    /**
     * Metodo invocado por el contenedor para reactivar la instancia
```

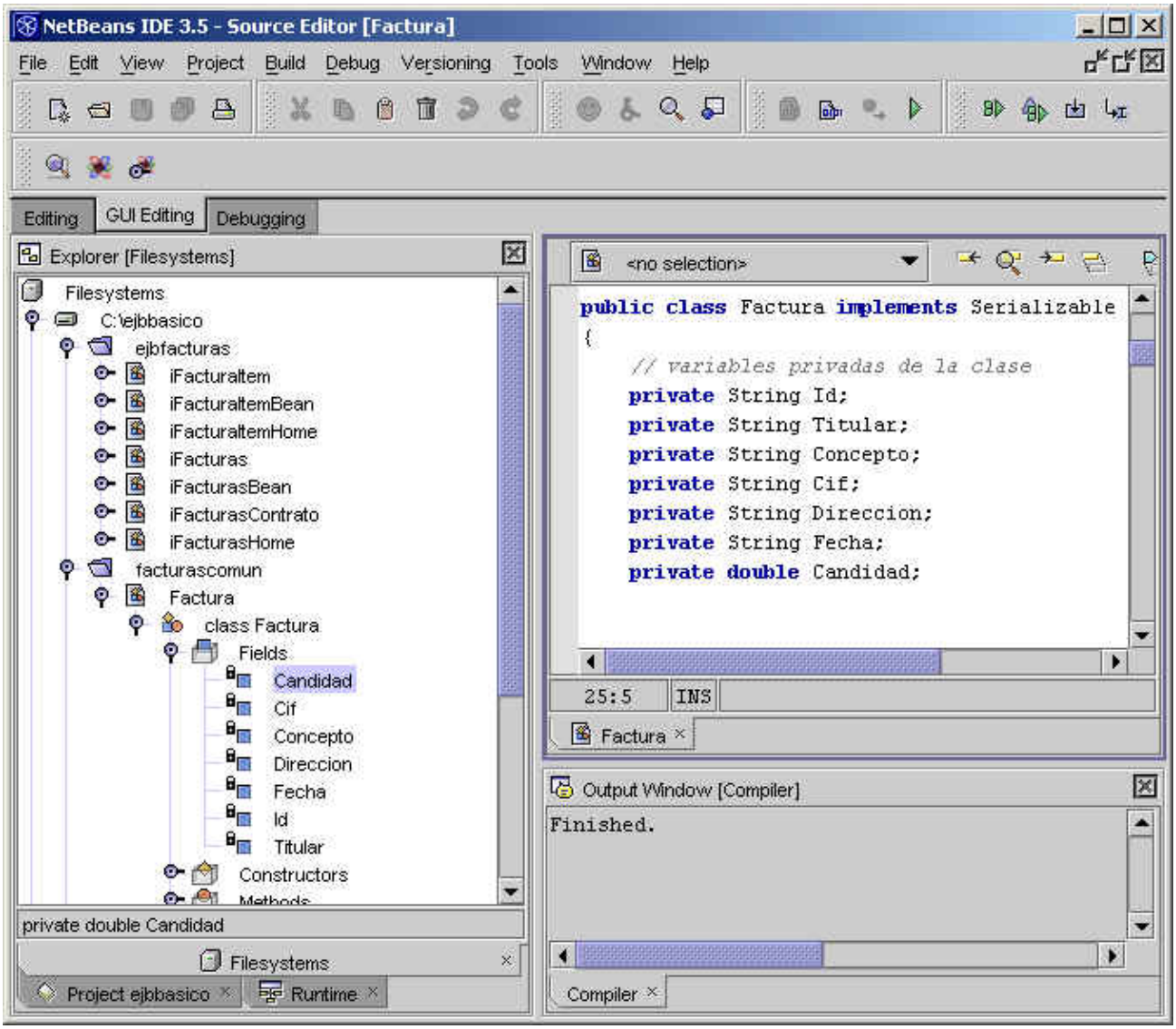
```
     */
    public void ejbActivate() {
        iId = (String)context.getPrimaryKey(); // recuperamos la clave primaria
        depura("Se activa el EJB para id " + iId);
    }


    /**
     * Metodo invocado por el contenedor para desactivar la instancia
     */
    public void ejbPassivate() {
        depura("Se pasiva el EJB para id " + iId);

        iId = null;  // eliminamos la referencia actual
    }

    /**
     * Metodo invocado por el contenedor para borrar la instancia
     */
    public void ejbRemove()  throws RemoveException {
        borraFila();
    }

    void borraFila() throws RemoveException
    {
        depura("Borramos el EJB para id " + iId);
    }

    /**
     * Metodo invocado por el contenedor para cargar de BBDD los datos de la instancia
     */
    public void ejbLoad() throws javax.ejb.EJBException, java.rmi.RemoteException {

        cargaDatos();
    }

    void cargaDatos() throws NoSuchEntityException
    {
        depura("Mandan cargar datos para ID = " + iId);

        if(iId.compareTo("22") == 0)  // simulamos la recuperacion de datos de la BBDD
        {
            sTitular        = "adictosaltrabajo"  ;
            sConcepto       = "consultoria";
            sCif            = "b64554432";
            sDireccion      = "Mi casa";
            sFecha          = "01102003";
            dCandidad       = 1000;
        }
    }


     /**
     * Metodo invocado por el contenedor para almacenar los datos cambiados
     */
    public void ejbStore() throws javax.ejb.EJBException, java.rmi.RemoteException {
        guardaDatos();
    }

    void guardaDatos() throws NoSuchEntityException
    {
```

```
        depura("Guardamos el EJB para id = " + iId);
    }


    /**
     * Metodo para localizar las facturas adecuadas
     */
    public Collection ejbFindByRango(int pOrigen,int pDestino) throws FinderException
    {
        depura("Retornamos entre el Rango: " + pOrigen + " y :" + pDestino);
        return null;
    }



     /**
     * Metodo invocado por el contenedor para pasarnos y desactivar el contexto
     */
    public void setEntityContext(javax.ejb.EntityContext entityContext) throws javax.ejb.EJBExc
            depura("Establecemos el contexto");
            this.context = entityContext; // si se nos olvida fallara el ejbActivate
    }

    public void unsetEntityContext() throws javax.ejb.EJBException, java.rmi.RemoteException {
            depura("Des-establecemos el contexto");
            context = null;
    }

    /** Metodo get para  sConcepto.
     * @return Value of property sConcepto.
     *
     */
    public String getSConcepto() {
        return sConcepto;
    }

    /** Metodo set para sConcepto.
     * @param sConcepto Nuevo valor de propiedadsConcepto.
     *
     */
    public void setSConcepto(String sConcepto) {
        this.sConcepto = sConcepto;
    }

    /** Metodo get para  sFecha.
     * @return Value of property sFecha.
     *
     */
    public String getSFecha() {
        return sFecha;
    }

    /** Metodo set para sFecha.
     * @param sFecha Nuevo valor de propiedadsFecha.
     *
     */
    public void setSFecha(String sFecha) {
        this.sFecha = sFecha;
    }

    /** Metodo get para  sDireccion.
```

```
     * @return Value of property sDireccion.
     *
     */
    public String getSDireccion() {
        return sDireccion;
    }

    /** Metodo set para sDireccion.
     * @param sDireccion Nuevo valor de propiedadsDireccion.
     *
     */
    public void setSDireccion(String sDireccion) {
        this.sDireccion = sDireccion;
    }

    /** Metodo get para  sTitular.
     * @return Value of property sTitular.
     *
     */
    public String getSTitular() {
        return sTitular;
    }

    /** Metodo set para sTitular.
     * @param sTitular Nuevo valor de propiedadsTitular.
     *
     */
    public void setSTitular(String sTitular) {
        this.sTitular = sTitular;
    }

    /** Metodo get para  sCif.
     * @return Value of property sCif.
     *
     */
    public String getSCif() {
        return sCif;
    }

    /** Metodo set para sCif.
     * @param sCif Nuevo valor de propiedadsCif.
     *
     */
    public void setSCif(String sCif) {
        this.sCif = sCif;
    }

    /** Metodo get para  iId.
     * @return Value of property iId.
     *
     */
    public String getIId() {
        return iId;
    }

    /** Metodo set para iId.
     * @param iId Nuevo valor de propiedadiId.
     *
     */
    public void setIId(String iId) {
        this.iId = iId;
```

```
    }

    /** Metodo get para  dCandidad.
     * @return Value of property dCandidad.
     *
     */
    public double getDCandidad() {
        return dCandidad;
    }

    /** Metodo set para dCandidad.
     * @param dCandidad Nuevo valor de propiedaddCandidad.
     *
     */
    public void setDCandidad(double dCandidad) {
        this.dCandidad = dCandidad;
    }
}
```

La estructura de nuestro proyecto tiene el siguiente aspecto



Invocamos desde el EJB de sesión al de entidad

Desde un metodo del EJB de sesión, ejecutamos la llamada al de Entidad.

```
     depura("Comenzamos la recuperacion de elementos");

       try
       {
           Context contexto = new InitialContext();
           depura("Hemos creado el contexto y vamos a buscar objeto");

           iFacturaItemHome miHome = null;

           Object objetoGenerico = contexto.lookup("java:comp/env/ejb/entidadfactura");

           depura("La clase original es " + objetoGenerico.getClass().getName());
           miHome = (iFacturaItemHome) PortableRemoteObject.narrow(objetoGenerico,iFacturaItem

           depura("Buscamos por ID = 22");

           iFacturaItem miFactura = miHome.findByPrimaryKey("22");
           depura("Mostramos el concepto " + miFactura.getSConcepto());

           depura("La operacion ha finalizado");
       }
       catch(Exception e)
       {
           depura("Error al probar ejb de entidad: " + e.getMessage());
       }

       return true;
```

Deberiamos hacernos una pregunta. ¿ Como la llamada a **findByPrimaryKey** retorna un **iFacturaItem**?

iFacturaItem miFactura = miHome.**findByPrimaryKey**("22");

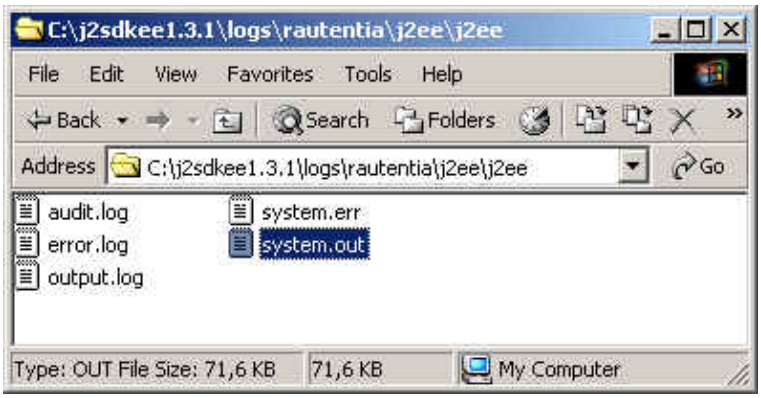
Si la funcion el el Bean .. retorna un entero ?? ...

public **String** **ejbFindByPrimaryKey**(String pParam) throws FinderException

Esto significa, que el contenedor lo hace por nosotros ....

Comprobar el funcionamieto

Si vemos la salida.... sin estar conectando realmente con una base de datos ... podemos ver que hace el contenedor de EJBs ...



Sobre todo debemos prestar atención a la última linea ..... para ver como el contenedor hace cosas por nosotros

```
EJB iFacturasBean: Comenzamos la recuperacion de elementos
EJB iFacturasBean: Hemos creado el contexto y vamos a buscar objeto
EJB iFacturasBean: La clase original es ejbfacturas._iFacturaItemHome_Stub
EJB iFacturasBean: Buscamos por ID = 22
EJB iFacturaItemBean: Establecemos el contexto
EJB iFacturaItemBean: Invocado buscar por id 22
EJB iFacturaItemBean: Se activa el EJB para id 22
EJB iFacturaItemBean: Mandan cargar datos para ID = 22
EJB iFacturasBean: Mostramos el concepto consultoria
EJB iFacturasBean: La operacion ha finalizado
EJB iFacturaItemBean: Guardamos el EJB para id = 22
```

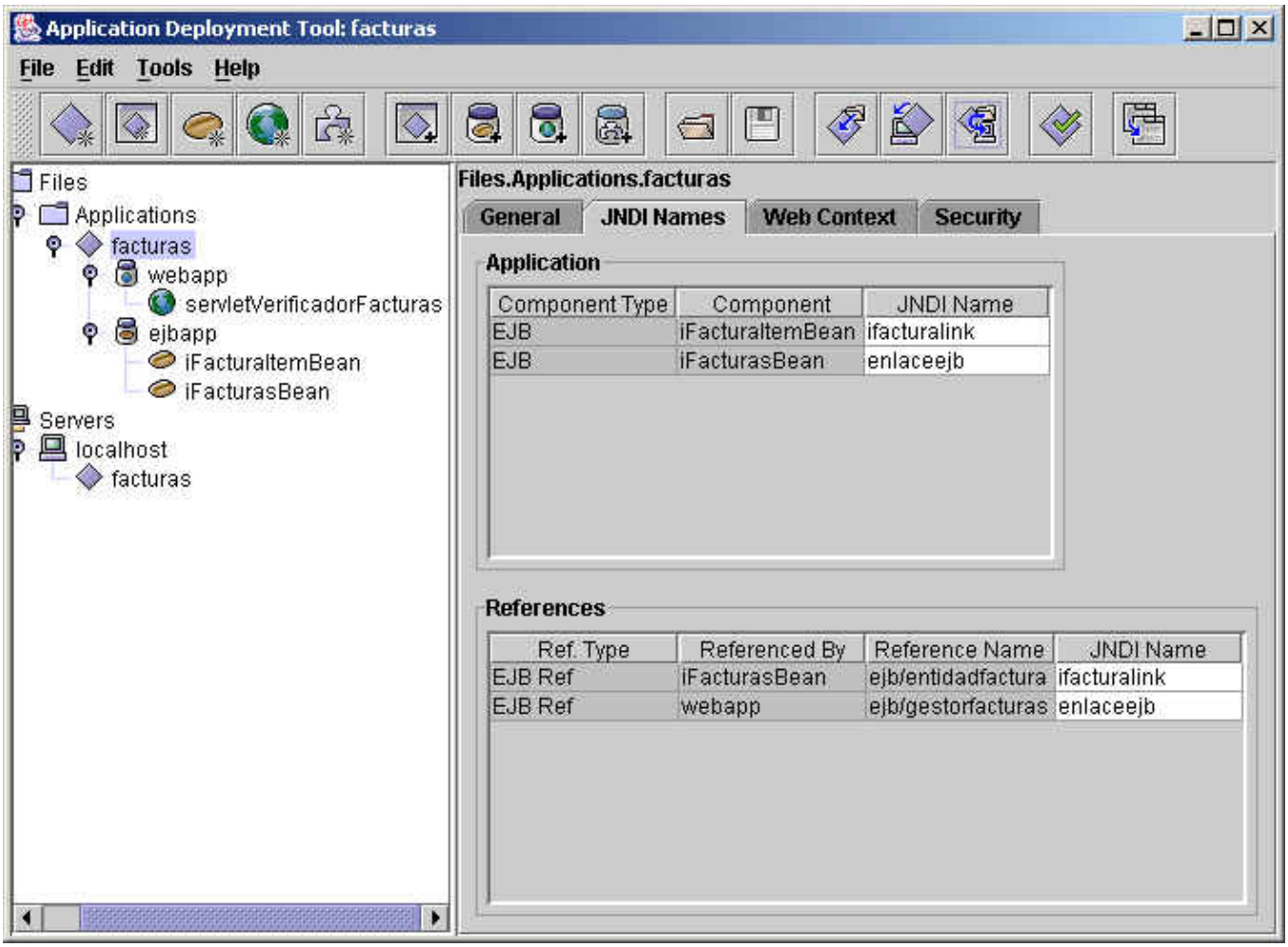
Podeis descargar el código completo

Desplegar la aplicación

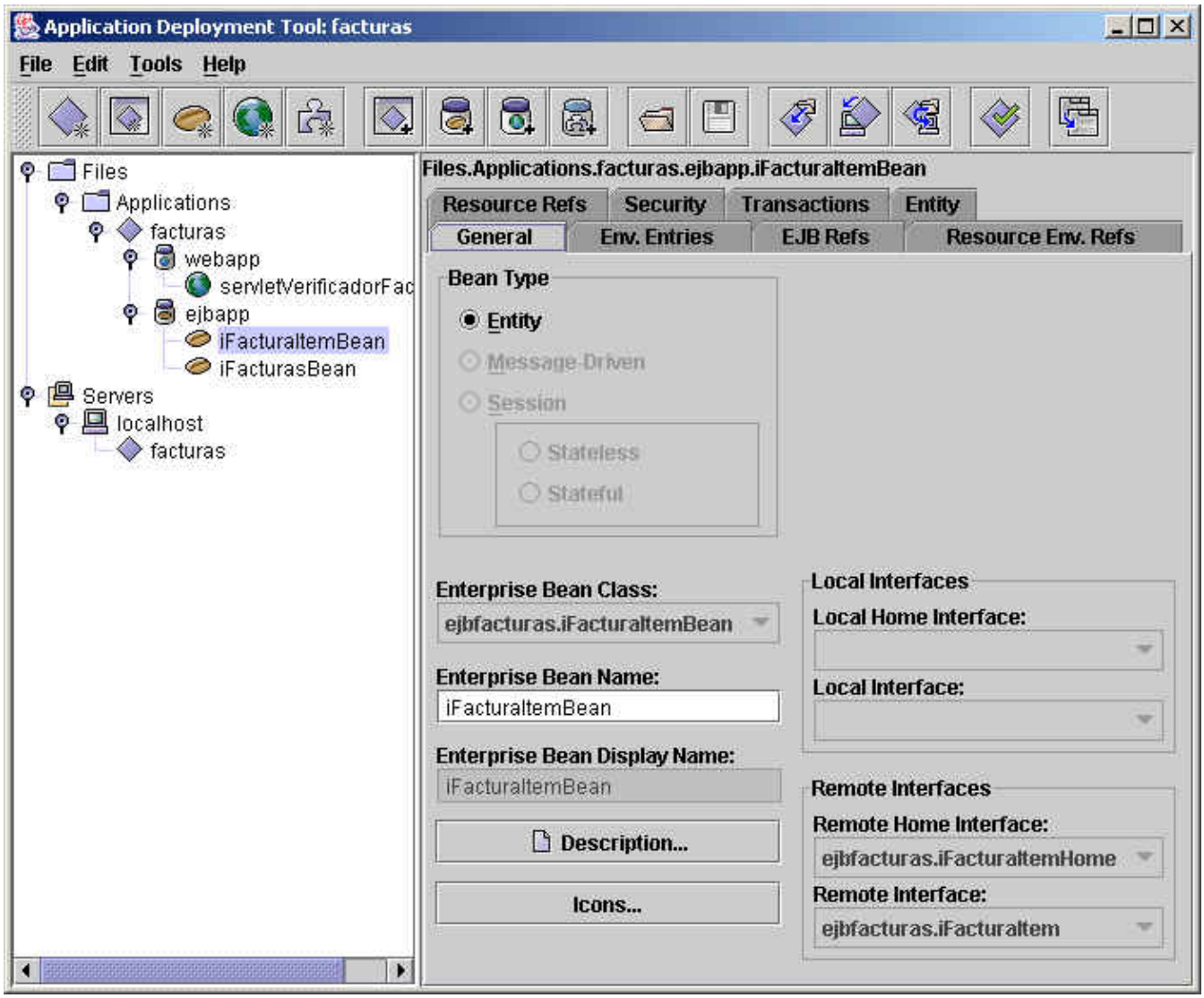
Asignamos los nombre JNDI (hemos cambiado también el de sesión).

Podemos ver que desde el EJB de sesion hemos encontrado el de entidad como:

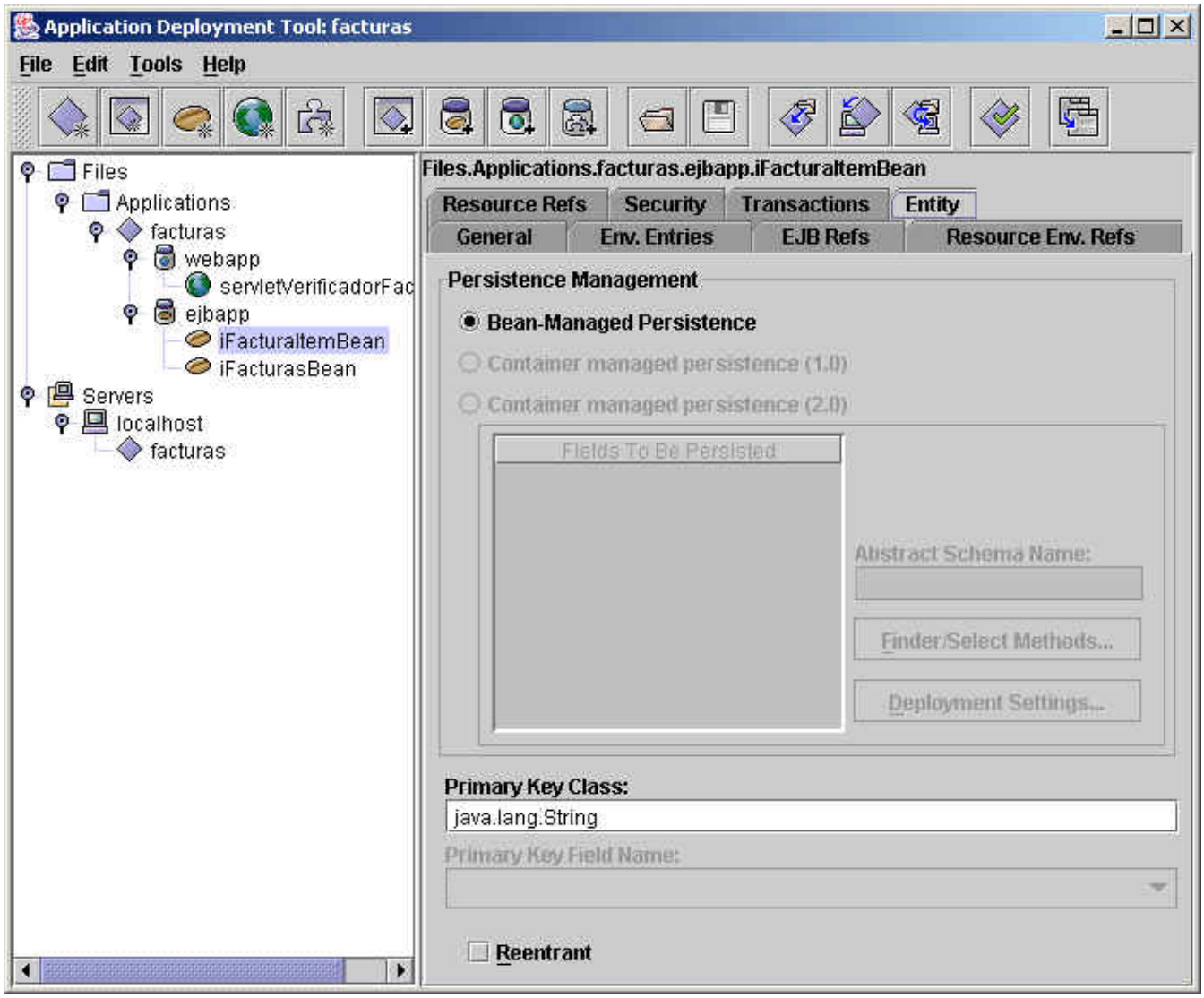
Object objetoGenerico = contexto.lookup("**java:comp/env/ejb/entidadfactura**");



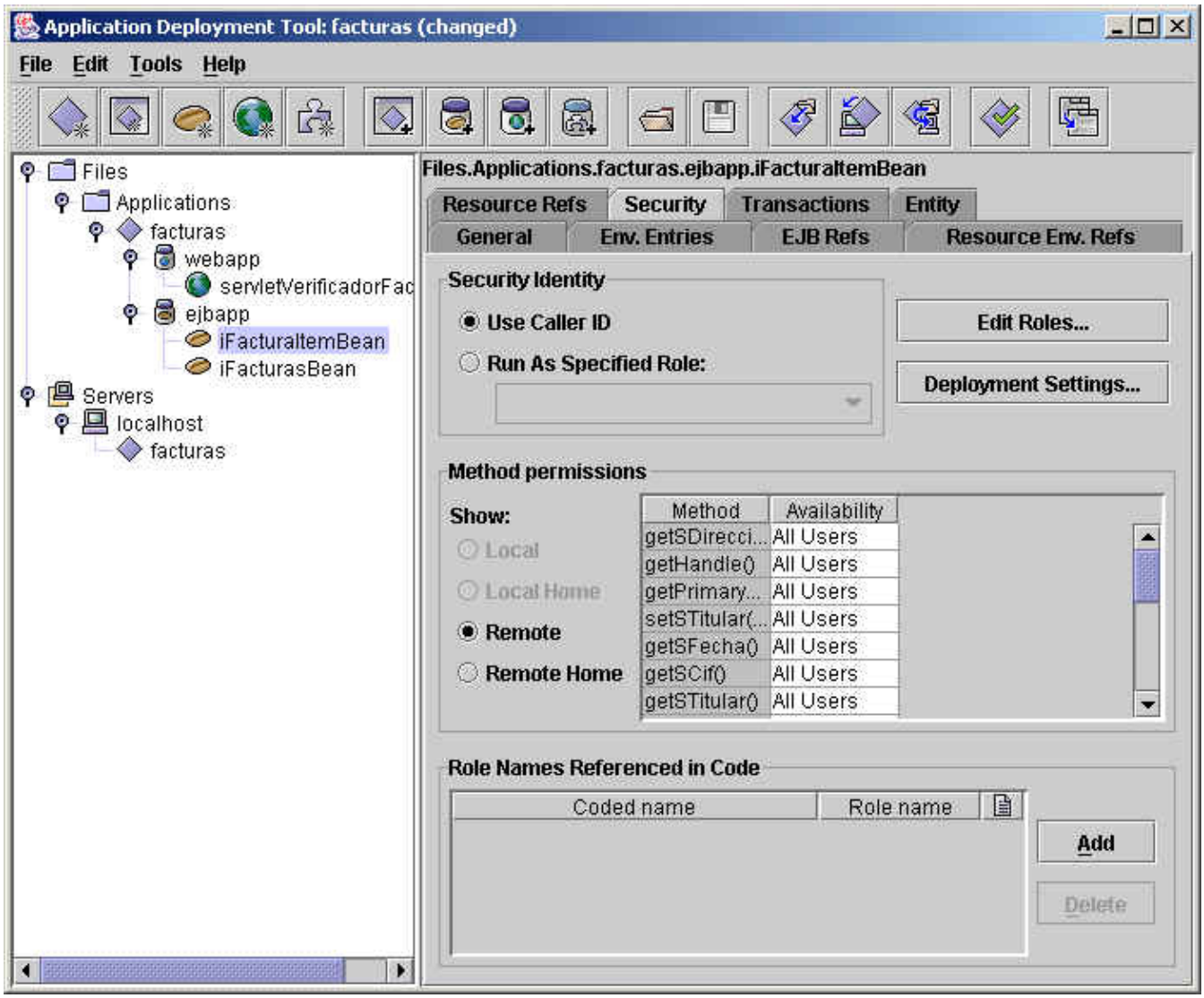
Definimos el Bean como de Entidad e identificamos sus interfaces



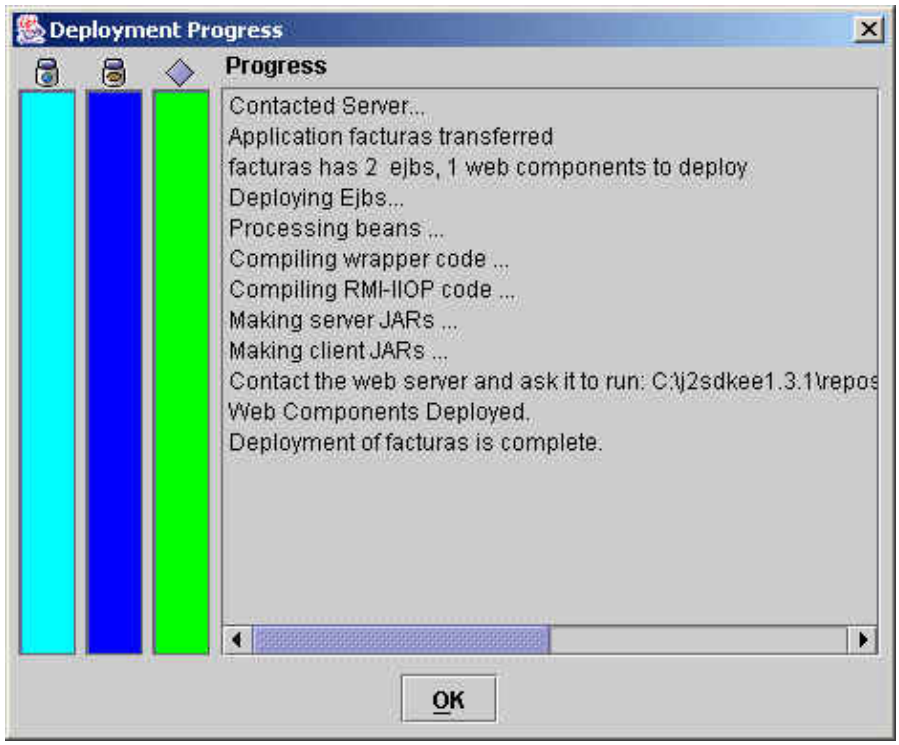
La persistencia la maneja el Bean



La seguridad de momento la dejamos a un lado (todo llegará)



Y redesplegamos



Ya estamos en marcha de nuevo ..... con un entorno más completo. A partir de ahora, analizaremos lo que hemos hecho y como mejorarlo.

Preguntas

Hay algunas preguntas que nos debemos hacer

- ¿Por qué se crea el EJB de entidad desde el de sesion?
- ¿Por qué hemos creado una clase Factura que es la que se pasa como parámetro?
- ¿La aplicación tiene un alto o bajo nivel de acoplamiento?
- ¿Si despliego los ejbs y tengo que cambiar sus nombres ... en cuantos sitios debo tocar?
- ¿Realmente estoy programando orientado a objetos o estoy complicando innecesariamente la aplicación con interfaces y objetos sin realmente obtener las ventajas de este tipo de programación?

Muchas veces se pone de moda una tecnología y se usa... sin ser muy consciente del objetivo que busca

A partir de ahora, uno de nuestros objetivos será responder a estas preguntas ....

Sobre el Autor ..

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e−mail	

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Otra implementación JDO con TJDO	Os mostramos como montar un ejemplo simple de JDO, a través de la implementación gratuita TJDO
Reingeniería JDO con Druid	Os mostramos como crear vuestras clases y descriptores JDO, de tablas existentes, con la herramienta gratuita Druid.
JDO con OJB	Os mostramos como configurar el entorno OJB de apache para construir la primera aplicación JDO
Message−Driven Beans al instante	Os mostramos como crear un EJB que consuma los mensajes JMS de una cola
Struts Jakarta	Cuando se ha trabajado creando aplicaciones Java poco a poco se va viendo la necesidad de normalizar los desarrollo. Uno de los Framework (entornos) más extendidos es Struts
Despliegue gráfico de EJBs	Os mostramos como crear y desplegar de un modo gráfico un EJB de sesión el el servidor de aplicaciones de referencia de Sun
EJB´s y Orion	Recreación de la guía paso a paso de como crear una aplicación Web con EJB´s y Servlets y su despliegue con ANT sobre Orion
CMP Entity Beans y MySql	Os mostramos como crear un Entity Bean con persistencia controlada por el servidor, configurado para usar MySql