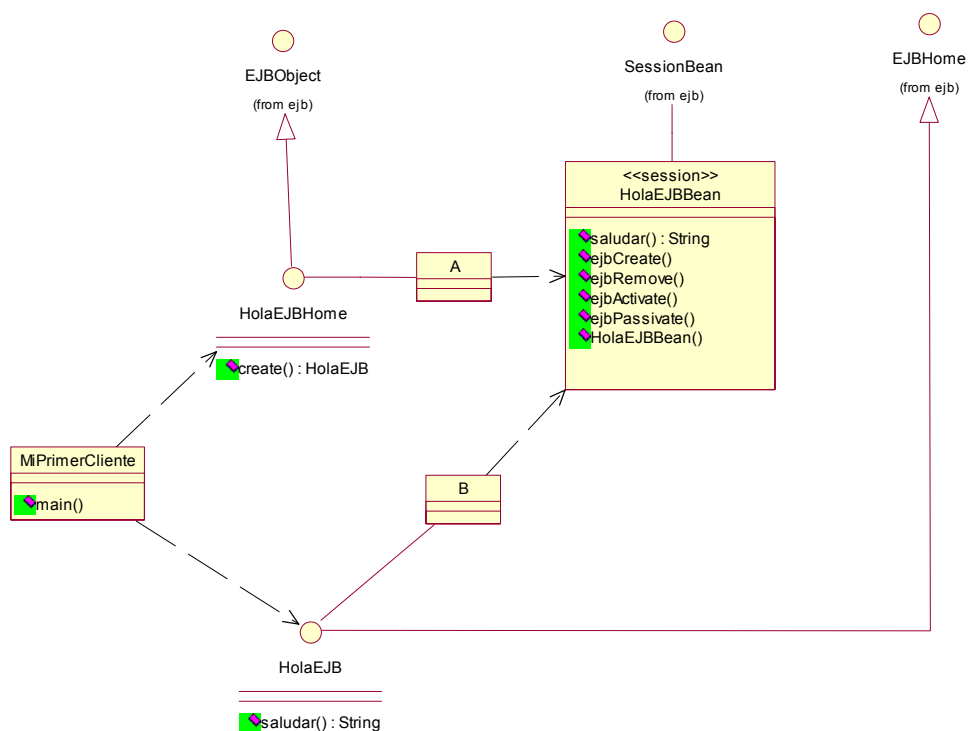
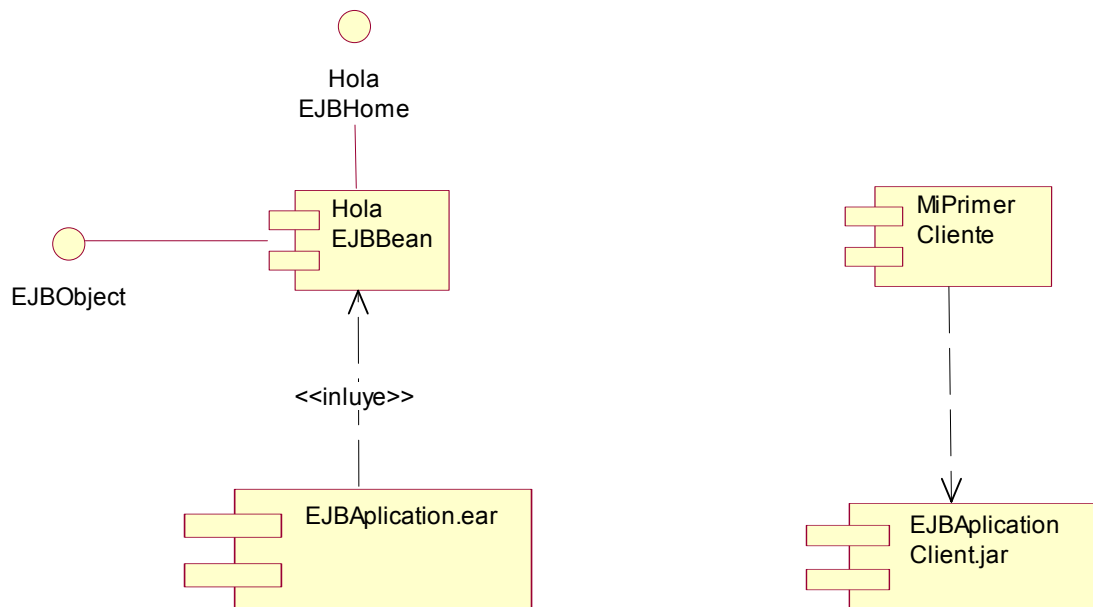


JTALLER 1: Introducción a los componentes Java



El modelo de componentes



El código a escribir y a compilar

No se olvide fijar el path y el class path para que encuentre:

1. El compilador y el intérprete java: `javac` y `java`
2. Las bibliotecas.(incluyendo `holaMundoClient.jar` el cual posteriormente se describe)

Para ello puede ejecutar el siguiente batch:

```
@echo off
cls
set J2EE_HOME=C:\j2sdkee1.3.1
set JAVA_HOME=C:\j2sdk1.4.1_01
rem CODEDIR donde tengo mi codigo
set CODEDIR=C:\julio
set classpath=.;%J2EE_HOME%\lib\j2ee.jar;%CODEDIR%\holaComponentes\holaMundoClient.jar
set path=.;%JAVA_HOME%\bin;%path%
```

La interface home: HolaEJBHome

```
package holaComponentes;

import javax.ejb.*;

public interface HolaEJBHome extends javax.ejb.EJBHome {

    public holaComponentes.HolaEJB create()
        throws javax.ejb.CreateException, java.rmi.RemoteException;

}
```

La interface remota: HolaEJB

```
package holaComponentes;

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

public interface HolaEJB extends javax.ejb.EJBObject {

    public String saludar() throws RemoteException;

}
```

El EJB de Session: HolaEJBBean

```
package holaComponentes;
import javax.ejb.*;
public class HolaEJBBean implements javax.ejb.SessionBean {
    private javax.ejb.SessionContext context;
    public HolaEJBBean(){}
    public String saludar(){
        return(" Hola te saluda un Componente Java de tipo Session ");
    }
    public void setSessionContext(javax.ejb.SessionContext aContext) {
        context=aContext;    }

    public void ejbActivate() {}
    public void ejbPassivate(){}
    public void ejbRemove() {}
    public void ejbCreate() {}
}
```

El cliente del EJB: MiPrimerCliente

(Este no lo compile aún, espérese al deployment del servidor)

```
package holaComponentes;
import javax.naming.*;
import javax.rmi.*;

public class MiPrimerCliente {
    public MiPrimerCliente() {
    }
    public static void main(String[] args) {
        try{

            Context initial = new InitialContext();
            System.out.println("Trayendo el contexto");

            Object objref= initial.lookup("JNDIHOLAMUNDO");
            System.out.println("Trayendo Interface remote");
            HolaEJBHome holaHome=(HolaEJBHome)
            javax.rmi.PortableRemoteObject.narrow(objref,HolaEJBHome.class);
            System.out.println("Creando el bean");
            HolaEJB hola = holaHome.create();
            System.out.println("Llamando el método remoto");
            System.out.println(hola.saludar());
            hola.remove();
        }
        catch(Exception ex){
            System.out.println("Mi primer error de
Componentes"+ex.getMessage());
        }
    }
}
```

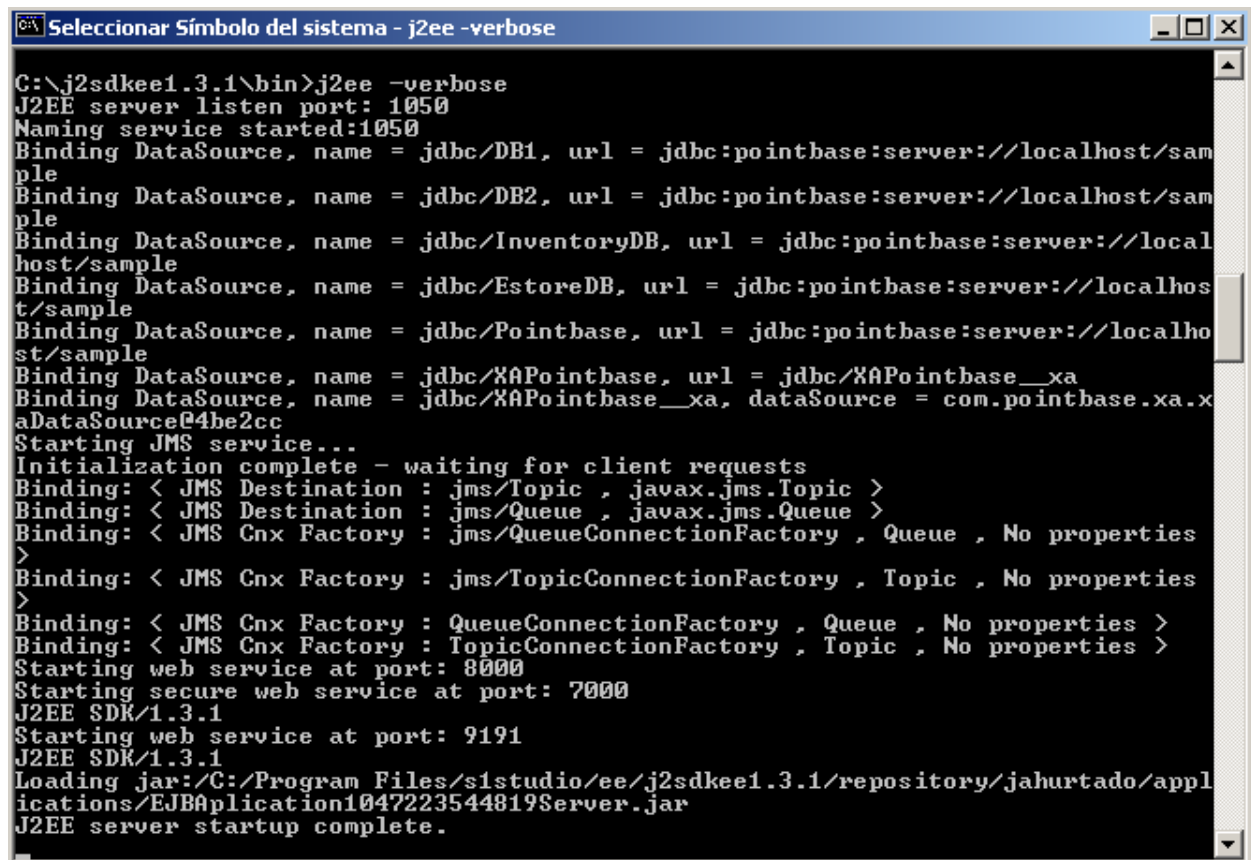
El Deployment del EJB

1. Inicie el EJB Server

Use el archivo batch que brinda el J2EE SDK para iniciar el servidor, usando la consola(símbolo de sistema) ejecutar:

C:\j2sdkee1.3.1\bin>j2ee -verbose

Y aparecerá algo así:



```
C:\j2sdkee1.3.1\bin>j2ee -verbose
J2EE server listen port: 1050
Naming service started:1050
Binding DataSource, name = jdbc/DB1, url = jdbc:pointbase:server://localhost/sample
Binding DataSource, name = jdbc/DB2, url = jdbc:pointbase:server://localhost/sample
Binding DataSource, name = jdbc/InventoryDB, url = jdbc:pointbase:server://localhost/sample
Binding DataSource, name = jdbc/EstoreDB, url = jdbc:pointbase:server://localhost/sample
Binding DataSource, name = jdbc/Pointbase, url = jdbc:pointbase:server://localhost/sample
Binding DataSource, name = jdbc/XAPointbase, url = jdbc/XAPointbase__xa
Binding DataSource, name = jdbc/XAPointbase__xa, dataSource = com.pointbase.xa.xaDataSource@4be2cc
Starting JMS service...
Initialization complete - waiting for client requests
Binding: < JMS Destination : jms/Topic , javax.jms.Topic >
Binding: < JMS Destination : jms/Queue , javax.jms.Queue >
Binding: < JMS Cnx Factory : jms/QueueConnectionFactory , Queue , No properties >
Binding: < JMS Cnx Factory : jms/TopicConnectionFactory , Topic , No properties >
Binding: < JMS Cnx Factory : QueueConnectionFactory , Queue , No properties >
Binding: < JMS Cnx Factory : TopicConnectionFactory , Topic , No properties >
Starting web service at port: 8000
Starting secure web service at port: 7000
J2EE SDK/1.3.1
Starting web service at port: 9191
J2EE SDK/1.3.1
Loading jar:C:/Program Files/s1studio/ee/j2sdkee1.3.1/repository/jahurtado/applications/EJBApplication1047223544819Server.jar
J2EE server startup complete.
```

Pruebe si la parte web está operando a través de un browser web ingresando la siguiente dirección URL:

<http://localhost:8000/>

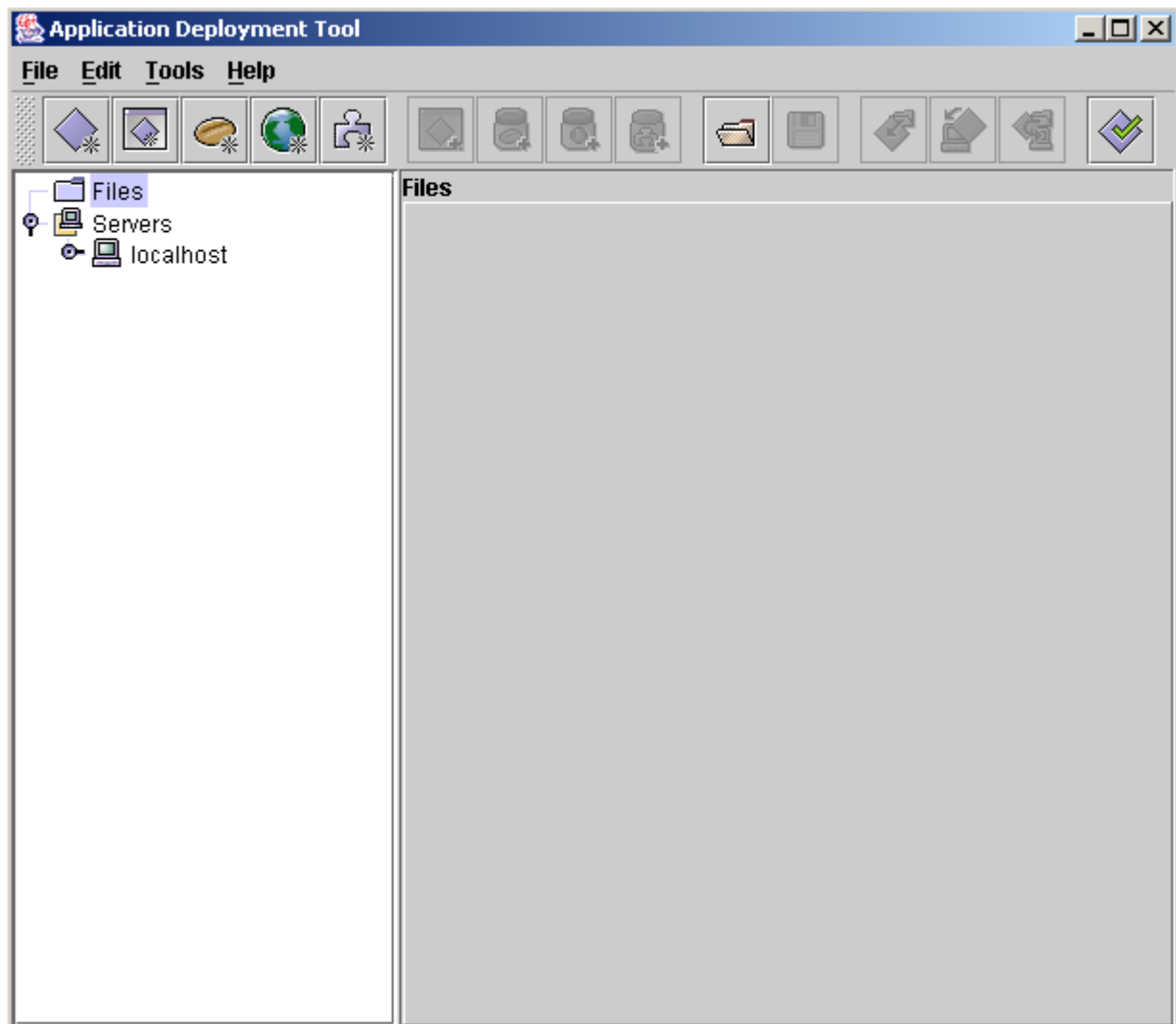
2. Despliegue del EJB

Ahora cargue la Aplicación Deployment Tool, esta le permitirá desplegar el EJB o una Aplicación web al servidor EJB en ejecución.

En una nueva consola ejecute:

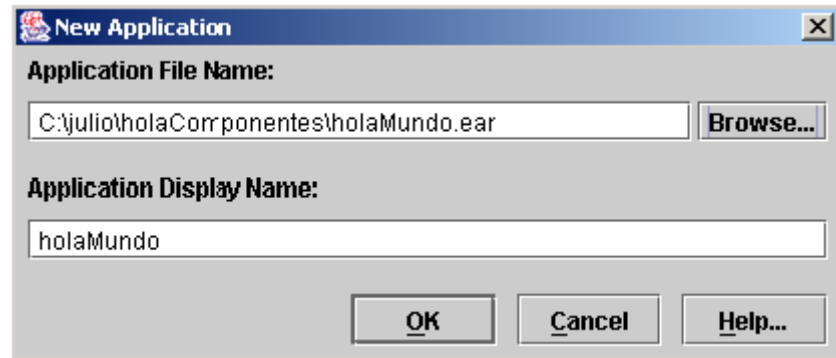
C:\j2sdee1.3.1\bin>**deploytool**

Aparecerá una ventana como sigue:



Lo primero que hay que hacer es crear la aplicación:

Para crear una aplicación, seleccione File | New | Application... del menú. En la ventana que aparece ingrese la siguiente información:



Ahora adicionemos el EJB a la aplicación, para ello seleccione File | New Enterprise Bean del menú. Después de leer el material introductorio presione el botón Next.

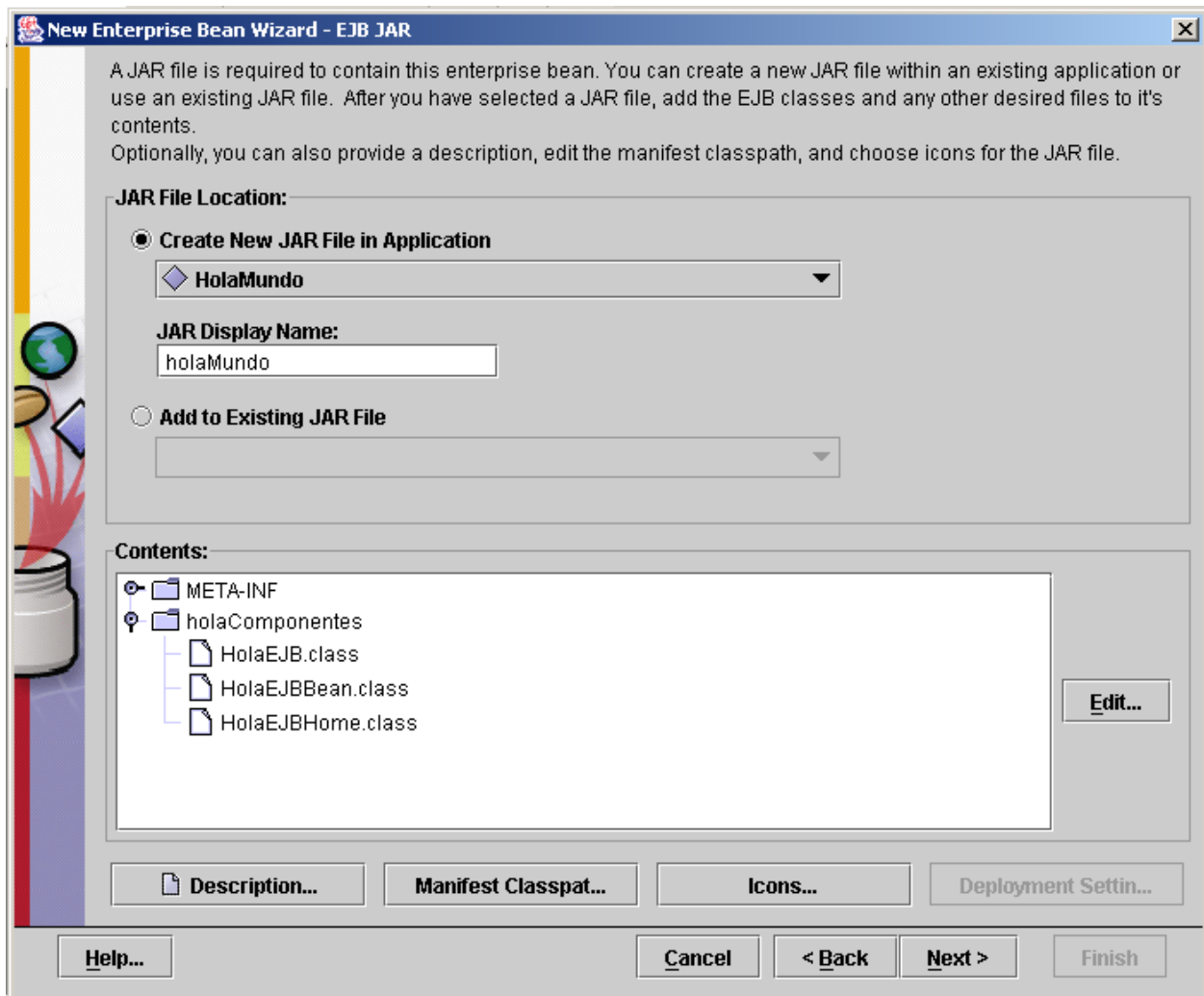
El siguiente paso consiste en crear un archivo jar y asociarlo con la aplicación creada en el paso anterior. El archivo jar contiene el EJB y sus interfaces Home y Remote. También tiene el descriptor de deployment, el cual es un archivo XML que provee los detalles de deployment para el contenedor(container). La herramienta de deployment crea este archivo a través del proceso de deployment.

Deberá crearlo como sigue:

Seleccionar la aplicación

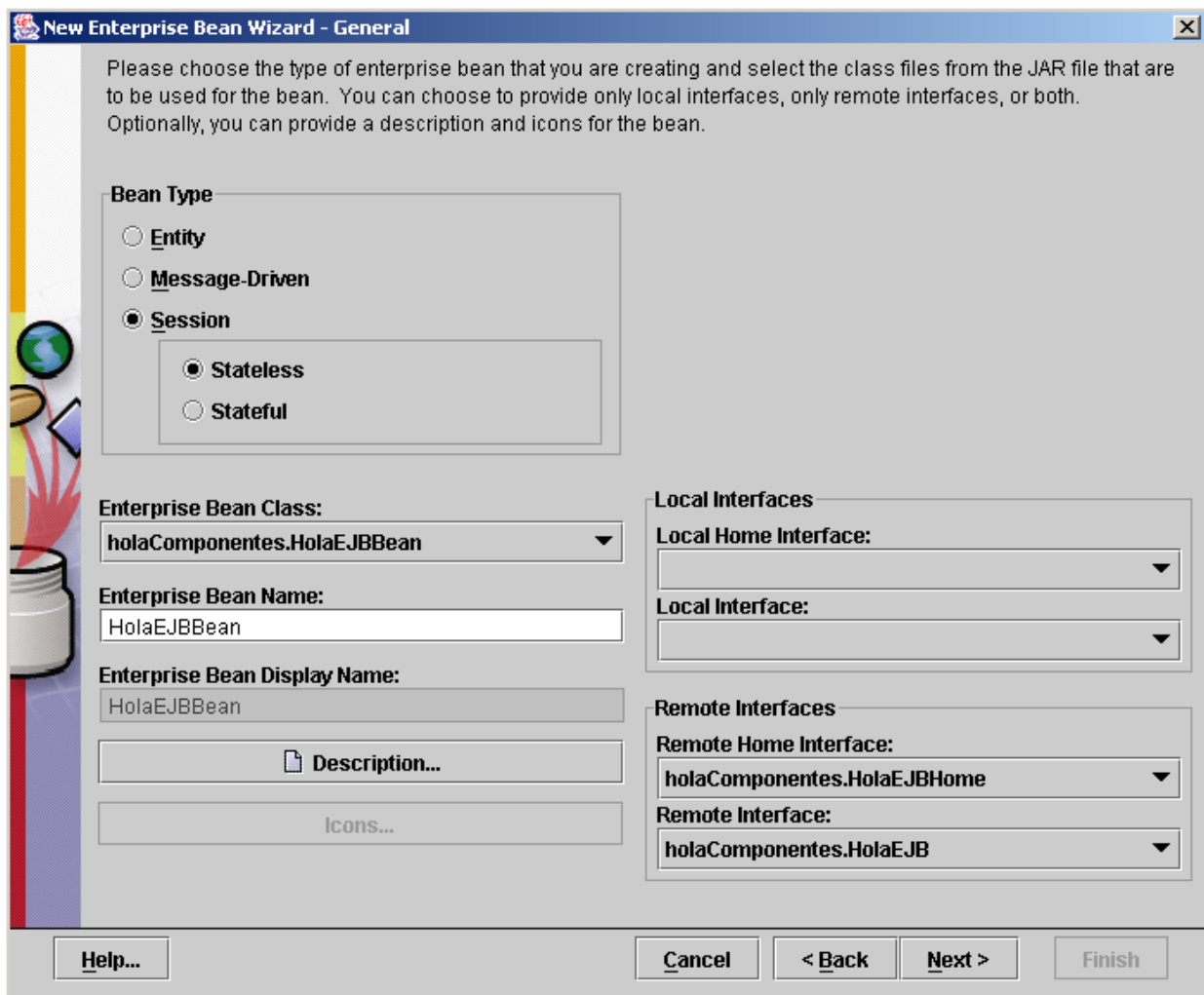
Crear el archivo .jar

Adicionar los archivos: HolaEJB.class, HolaEJBHome.class, HolaEJBBean.class



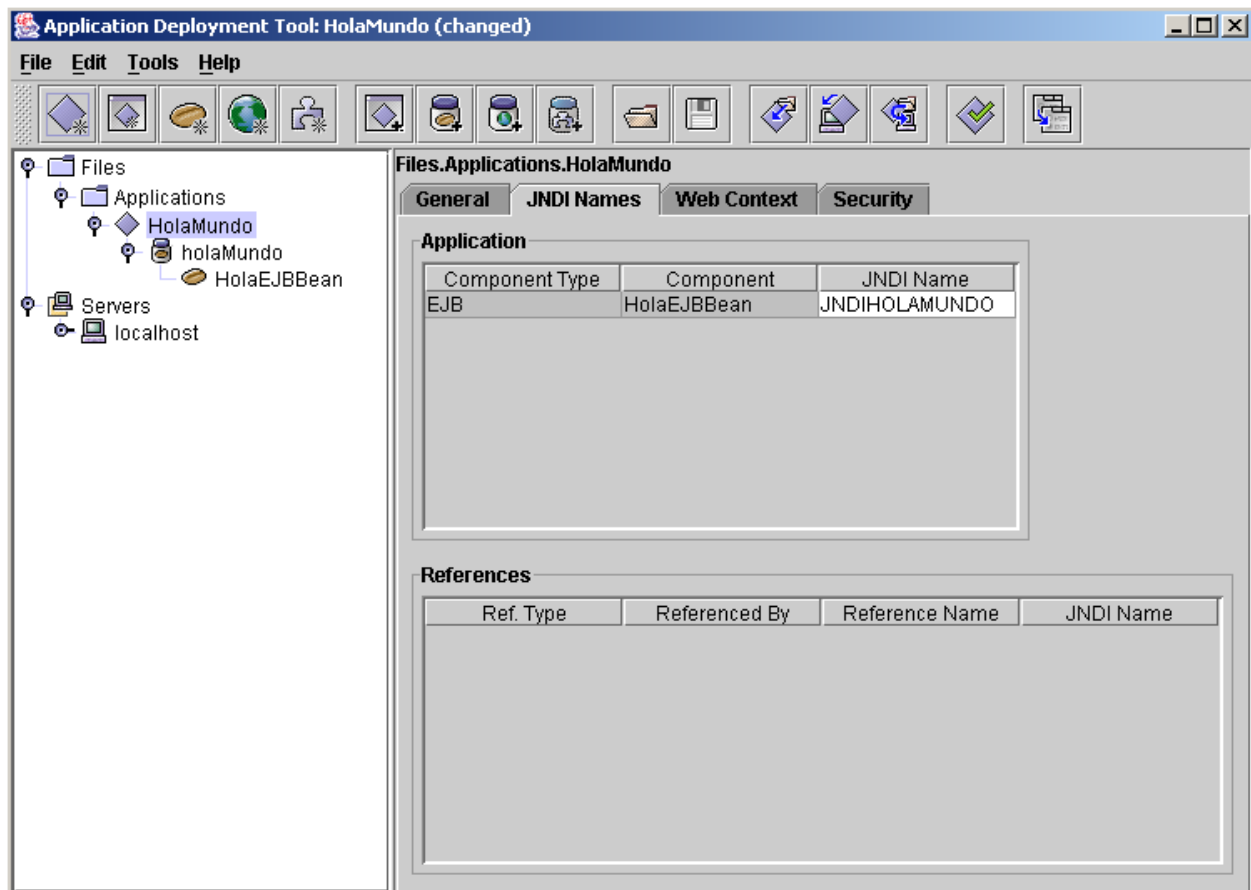
Una vez ha hecho esto presione el botón next para ir a la siguiente ventana, en esta se deberá asociar los archivos antes adicionados con el EJB, la interface Home y la interface Remote. También permitirá definir el tipo de EJB (Entity, Message-Driven, Session) y si es de Session permitirá definir el tipo Stateless o Stateful.

Seleccione tal y como se muestra en la siguiente figura:

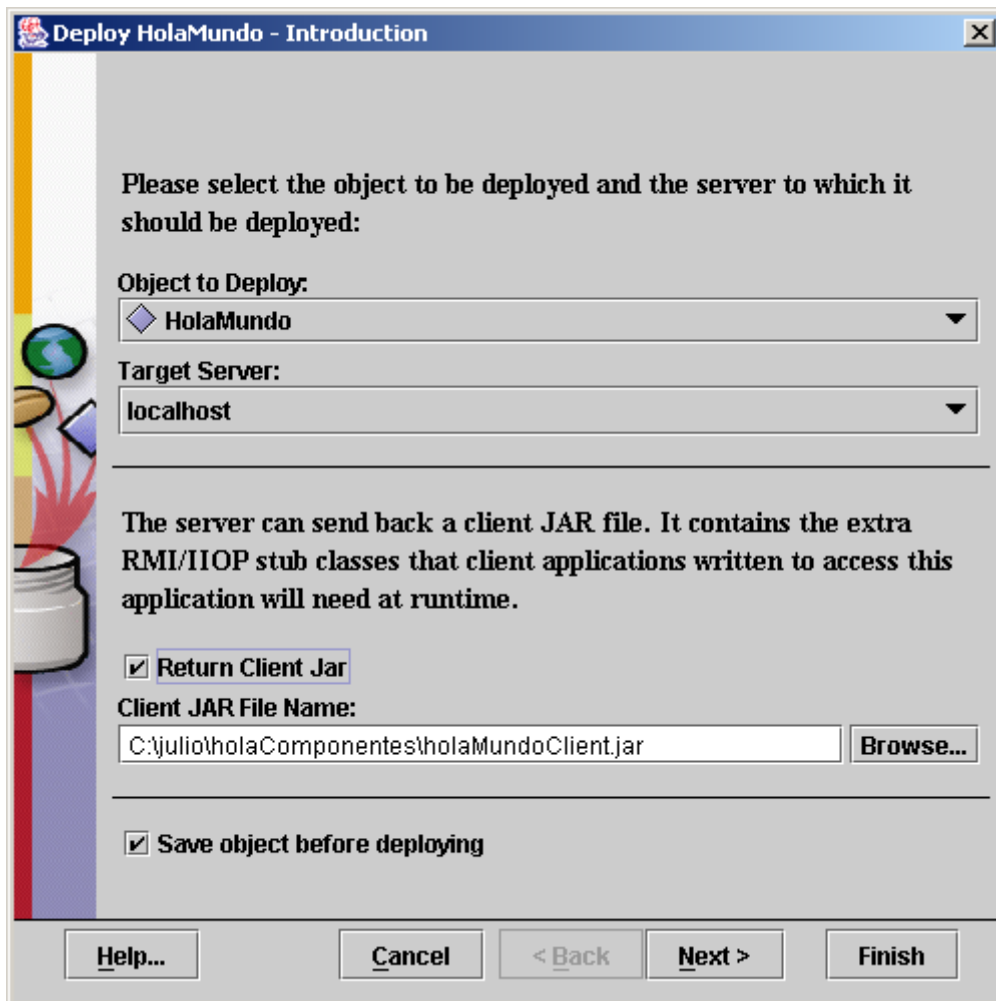


Seleccione Next y luego Finish, para completar la adición del archivo .jar a la aplicación. Esto los devuelve a la ventana principal.

Ahora hay que asegurar la identificación en el Java Naming and Directory Interface(JNDI). Para esto seleccione la aplicación y en el panel derecho seleccione la pestaña JNDI Names. En la columna de más a la derecha escriba el nombre con el que se identificará la aplicación a través de JNDI.



Ahora por fin el deployment, para ello haga clic derecho en la aplicación HolaMundo y seleccione Deploy...



Seleccione localhost en el target Server. Adicionalmente seleccione "Return Client Jar", ya que en esta jar se provee las clases proxy o skeleton que implementan las interfaces remote y home para el programa cliente.

Seleccione el botón Next y si en la siguiente ventana aún no aparece el Nombre JNDI adiciónese y presione next hasta finalizar.

EJECUCION Y PRUEBA

1. Compile el código del cliente
2. Ejecute el código del cliente de manera local
3. Ejecútelo de manera remota, para ello utilice el siguiente batch en la página cliente y copie los archivos j2ee.jar y holaMundoClient.jar a la máquina cliente.

```
@echo off
cls
set J2EE_HOME=C:\j2sdkee1.3.1
set JAVA_HOME=C:\j2sdk1.4.1_01
rem CODEDIR donde tengo mi codigo
set CODEDIR=C:\julio
set
classpath=.;%J2EE_HOME%\lib\j2ee.jar;%CODEDIR%\holaComponentes\holaMundoClient.jar
set path=.;%JAVA_HOME%\bin;%path%
set HOST=-Dorg.omg.CORBA.ORBInitialHost=J2EEHostName
java %HOST% -cp %CPATH% MiPrimerCliente
```

En lugar de **J2EEHostName** coloque el nombre del host donde se está ejecutando el servidor de aplicaciones.

EJERCICIO ADICIONAL

Construir un EJB de Session con estado, que almacene los nombres enviados por el usuario a través de una aplicación cliente y luego se los devuelva como un listado de nombres.