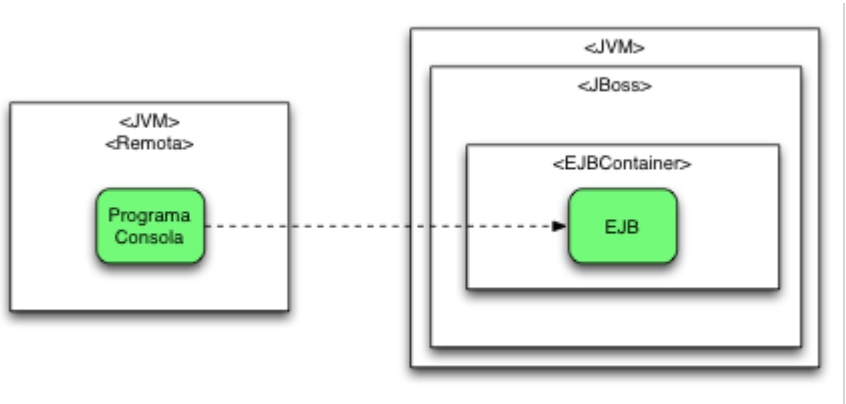
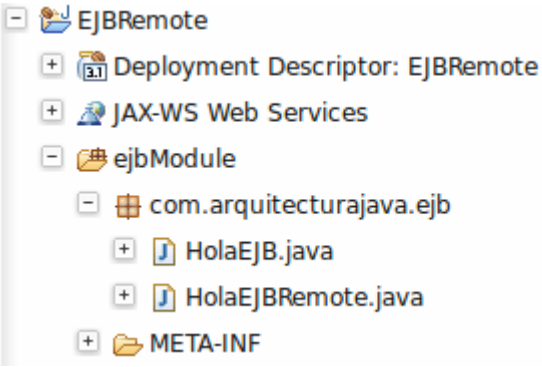
Una de las razones mas habituales para utilizar EJBs es poder acceder remotamente a ellos desde otras aplicaciones que se ejecutan en maquinas virtuales diferentes. En esta entrada vamos a desarrollar un EJB remoto al cual accedamos desde otra maquina virtual que cargue una sencilla aplicación de consola.



El primer paso que realizaremos será implementar un EJB con un interface remoto al cual podamos invocar desde un cliente.



Como podemos ver en la imagen el EJB se denomina HolaEJB y su interface HolaEJBRemote. Vamos a mostrar el código fuente de ambos.

Interface

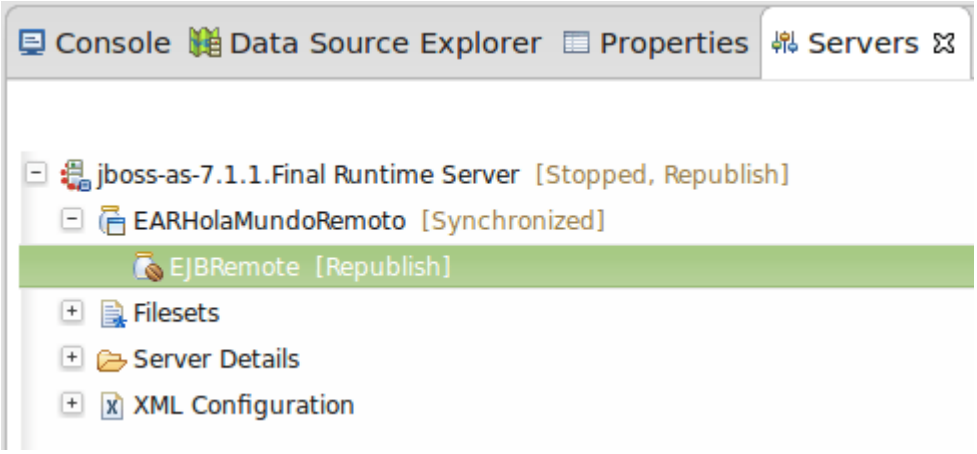
```
import javax.ejb.Remote;
```

```
@Remote
public interface HolaEJBRemote {
 public String mensaje();
}
```

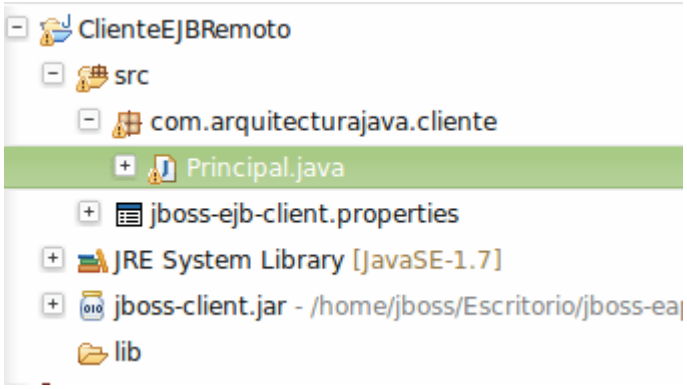
Implementación

```
import javax.ejb.Stateless;
@Stateless
public class HolaEJB implements HolaEJBRemote {

public HolaEJB() {
 }

@Override
 public String mensaje() {
 return "hola peticion remota a un ejb";
 }

}
&amp;lt;pre&amp;gt;
```

Una vez tenemos claro el interface e implementación del EJB . Lo desplegamos dentro de un EAR en nuestro servidor de aplicaciones JBoss.



Desplegado el EAR veremos en los logs de JBoss como ha quedado registrado . A partir de ese momento pasaremos a construir un cliente .Una peculiaridad que tienen los clientes remotos es que cada servidor de aplicaciones se configura de una forma distinta.Vamos a ver que elementos tiene que tener nuestro proyecto cliente.



Como podemos ver nuestro proyecto es un proyecto sencillo de consola que añade dos cosas .

- jboss-ejb-client.properties : Fichero de propiedades que configura la conexión al Servidor
- jboss-client.jar: Fichero JAR que incluye las librerías necesarias para conectarnos remotamente a jboss. Este fichero esta en la carpeta /jboss/client/bin .
  Vamos a ver el código del fichero de propiedades que vamos a usar:

```
endpoint.name=cliente

remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=
false

remote.connections=default

remote.connection.default.host=localhost

remote.connection.default.port = 4447
```

El fichero asigna varias propiedades necesarias para conectar entre las que hay que destacar el nombre del servidor “localhost” y el puerto “4447” . Una vez configurado el fichero correctamente . Podemos pasar a mostrar el código de nuestro cliente de consola.

```
import java.util.Hashtable;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

import com.arquitecturajava.ejb.HolaEJBRemote;

public class Principal {
 public static void main(String[] args) {

try {
 Hashtable propiedades = new Hashtable();
 propiedades
```

```
 .put(Context.URL_PKG_PREFIXES, "org.jboss.ejb.client.naming");

 Context context = new InitialContext(propiedades);

 HolaEJBRemote miejb=(HolaEJBRemote)
context.lookup("ejb:EARHolaMundoRemoto/EJBRemote//HolaEJB!com.arquitec
turajava.ejb.HolaEJBRemote");
 System.out.println(miejb.mensaje());
 } catch (NamingException e) {
 // TODO Auto-generated catch block
 e.printStackTrace();
 }

}

}
```

En el código se muestra como usamos JNDI para acceder al recurso remoto.Ejecutamos nuestro código y podremos ver el mensaje por consola.

```
ago 25, 2013 11:35:55 AM org.jboss.ejb.client.EJBClient <clinit>
INFO: JBoss EJB Client version 1.0.21.Final-redhat-1
ago 25, 2013 11:35:55 AM org.xnio.Xnio <clinit>
INFO: XNIO Version 3.0.7.GA-redhat-1
ago 25, 2013 11:35:55 AM org.xnio.nio.NioXnio <clinit>
INFO: XNIO NIO Implementation Version 3.0.7.GA-redhat-1
ago 25, 2013 11:35:55 AM org.jboss.remoting3.EndpointImpl <clinit>
INFO: JBoss Remoting version 3.2.16.GA-redhat-1
ago 25, 2013 11:35:55 AM org.jboss.ejb.client.remoting.VersionReceiver handleMe
INFO: EJBCLIENT000017: Received server version 2 and marshalling strategies [ri
ago 25, 2013 11:35:55 AM org.jboss.ejb.client.remoting.RemotingConnectionEJBRec
INFO: EJBCLIENT000013: Successful version handshake completed for receiver cont
hola peticion remota a un ejb
```

Hemos conseguido conectarnos a un EJB de forma remota