Accessing an enterprise bean

Davide Lissoni Mat.179878
13/10/2016

1.Introduction:

The requirements for this assignment were to develope a java EE application and a relative Client used to call the bean method developed in the enterprise application.
The java Enterprise application had to be deployed using Wildlfy (version 9.0.1. Final), an application server that implements the Java EE specification.

In particular the enterpprise bean has to contain a method used to return a string containing current date and time. The Client instead (a simple java application), will call the method present in the bean twice and then print out its output.

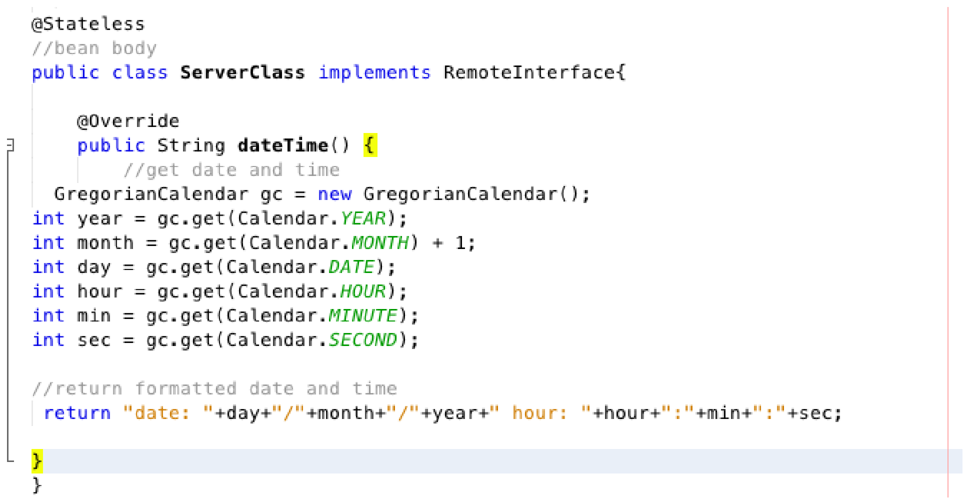
The program requirements seems to be very simple, but the core of this assignment was centered on the wildfly installation and to deploy an enterprise bean on it, since it was the first time that we had to handle with this framework.

2.Implementation:

The first step for the right implementation of this project was to install wildfly, to create an admin and a simple user on it, and to making Wildfly accessible from remote.
The guide followed in order to do the above topics, are the slides proposed by the professor, and, in my opinion, the only remakable feature fot this report is that, in order to make Wildfly accessible from remote, there is the need to modify the *standalone.xml* file by creating an interface accessible from any address and setting it as default interface.

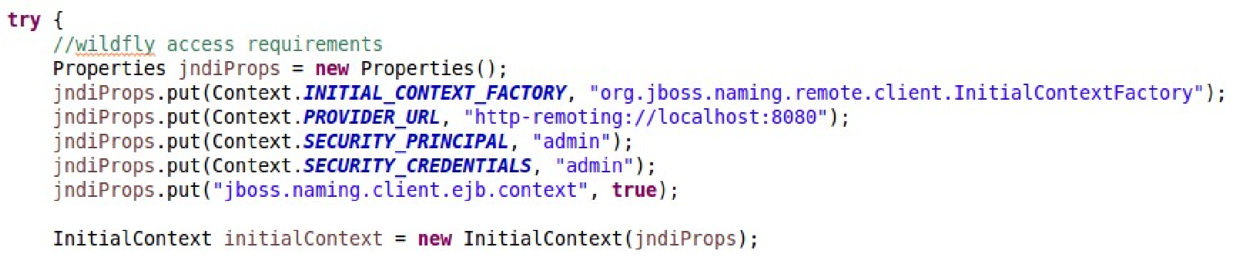
The first part of the project consists in a Java Enterprise application composed by a single stateless java enterprise bean, called ServerSide. This bean exposes only a simple String method dateTime(), whose simply gets current date and time, assembles a string containg these values and finally return it(*Figure n.1*).

The instance has to implements a remote interface containing the specification of the method implemented on the bean, in order to allow comunication between Client and bean.
The interface needs to import javax.ejb.Remote to designate itself as the remote business interface of the bean.
The interface must also contain the *@Remote* annotation in order to allow access from remote.



```
@Stateless
//bean body
public class ServerClass implements RemoteInterface{

    @Override
    public String dateTime() {
        //get date and time
  GregorianCalendar gc = new GregorianCalendar();
int year = gc.get(Calendar.YEAR);
int month = gc.get(Calendar.MONTH) + 1;
int day = gc.get(Calendar.DATE);
int hour = gc.get(Calendar.HOUR);
int min = gc.get(Calendar.MINUTE);
int sec = gc.get(Calendar.SECOND);

//return formatted date and time
 return "date: "+day+"/"+month+"/"+year+" hour: "+hour+":"+min+":"+sec;

}
}
```

Figure n.1



```
try {
    //wildfly access requirements
    Properties jndiProps = new Properties();
    jndiProps.put(Context.INITIAL_CONTEXT_FACTORY, "org.jboss.naming.remote.client.InitialContextFactory");
    jndiProps.put(Context.PROVIDER_URL, "http-remoting://localhost:8080");
    jndiProps.put(Context.SECURITY_PRINCIPAL, "admin");
    jndiProps.put(Context.SECURITY_CREDENTIALS, "admin");
    jndiProps.put("jboss.naming.client.ejb.context", true);

    InitialContext initialContext = new InitialContext(jndiProps);
```

Figure n.2

The second part of the project, the Java client, is a java application that, first of all, needs to implement the remote access to the stateless EJB, using Properties object (*Figure n.2*) and then, retrive the bean object by Context.lookup(String name), filling the String parameter, according to the bean property, in the following way:

```
initialContext.lookup(<Server app name> + "/" + <ejb-module name>+ "/" + <bean
name >+ "!" + <Remote inteface name>);
```

Finally the Client will call twice and print the results of the String dateTime() method situated on the bean just retrived.
Also the Client application needs to contain the Remote interface mentioned above. This is because since we have a remote client view, “calling any method from the remote home interface and/or remote component interface will be handled via remote method invocation(RMI)”.
Furthermore the Build path of the Client application must contain the jboss-client.jar Wildfly Library, which is located in /bin/client of the wildfly folder.

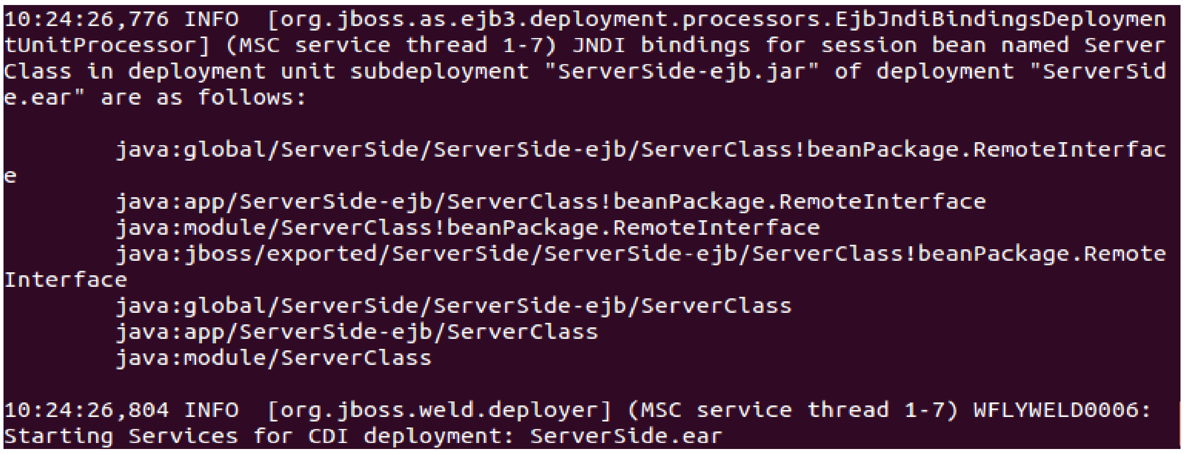
3.Deployment:

In order to develop and test this project I used a Linux machine, therefore the following commands are for running the project on a Linux machine. In any case the windows commands are pretty similar.

First of all there is the need to start the Wildfly application and to deploy its bean on it. In order to do start Wildfly digit on the terminal the following command line:

<path>/bin/standalone.sh
where <path> should be the path where Wildfly is situated on the pc.

In order to deploy the ServerSide application, generate the ear file from the Java EE project and after copy it in the *wildfly-9.0.1.Final/standalone/deployment*s folder.
The output of this operation should be similar to(*Figure n.3*):



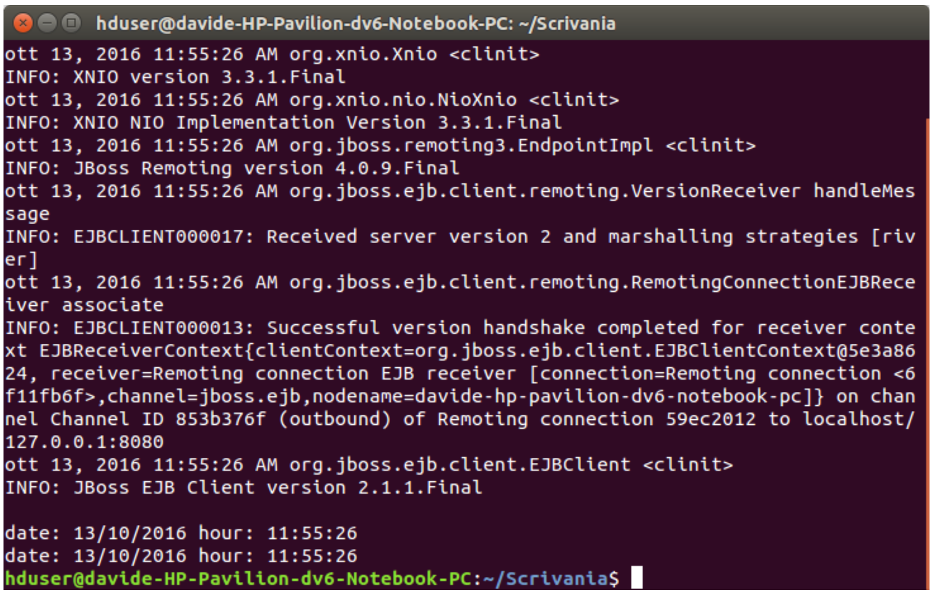
```
10:24:26,776 INFO  [org.jboss.as.ejb3.deployment.processors.EjbJndiBindingsDeploymen
tUnitProcessor] (MSC service thread 1-7) JNDI bindings for session bean named Server
Class in deployment unit subdeployment "ServerSide-ejb.jar" of deployment "ServerSid
e.ear" are as follows:

        java:global/ServerSide/ServerSide-ejb/ServerClass!beanPackage.RemoteInterfac
e
        java:app/ServerSide-ejb/ServerClass!beanPackage.RemoteInterface
        java:module/ServerClass!beanPackage.RemoteInterface
        java:jboss/exported/ServerSide/ServerSide-ejb/ServerClass!beanPackage.Remote
Interface
        java:global/ServerSide/ServerSide-ejb/ServerClass
        java:app/ServerSide-ejb/ServerClass
        java:module/ServerClass

10:24:26,804 INFO  [org.jboss.weld.deployer] (MSC service thread 1-7) WFLYWELD0006:
Starting Services for CDI deployment: ServerSide.ear
```

Figure n.3

Finally we can start the ClientRunnable.jar normally:

java -jar ClientRunnable.jar

If all the parameters are setted in the right way the output should be something like (*Figure n.4*):



```
ott 13, 2016 11:55:26 AM org.xnio.Xnio <clinit>
INFO: XNIO version 3.3.1.Final
ott 13, 2016 11:55:26 AM org.xnio.nio.NioXnio <clinit>
INFO: XNIO NIO Implementation Version 3.3.1.Final
ott 13, 2016 11:55:26 AM org.jboss.remoting3.EndpointImpl <clinit>
INFO: JBoss Remoting version 4.0.9.Final
ott 13, 2016 11:55:26 AM org.jboss.ejb.client.remoting.VersionReceiver handleMes
sage
INFO: EJBCLIENT000017: Received server version 2 and marshalling strategies [riv
er]
ott 13, 2016 11:55:26 AM org.jboss.ejb.client.remoting.RemotingConnectionEJBRece
iver associate
INFO: EJBCLIENT000013: Successful version handshake completed for receiver conte
xt EJBReceiverContext{clientContext=org.jboss.ejb.client.EJBClientContext@5e3a86
24, receiver=Remoting connection EJB receiver [connection=Remoting connection <6
f11fb6f>,channel=jboss.ejb,nodename=davide-hp-pavilion-dv6-notebook-pc]} on chan
nel Channel ID 853b376f (outbound) of Remoting connection 59ec2012 to localhost/
127.0.0.1:8080
ott 13, 2016 11:55:26 AM org.jboss.ejb.client.EJBClient <clinit>
INFO: JBoss EJB Client version 2.1.1.Final

date: 13/10/2016 hour: 11:55:26
date: 13/10/2016 hour: 11:55:26
hduser@davide-HP-Pavilion-dv6-Notebook-PC:~/Scrivania$
```

Figure n.4