Università degli Studi di Trento

Web Architectures

Accessing an entity bean

Gianvito Taneburgo

October 10, 2016

1 Introduction

The following report describes the procedure that has been followed to develop a client-server application using Enterprise JavaBeans (EJB). In particular, the program shows how a client can contact and use an EJB deployed and running inside the EJB container of the Wildlfy application server.
The Implementation paragraph provides details both on the server and on the client code. Some details about Java EE and EJB have been omitted: the reader is supposed to know the benefits of the technologies, the most import parts of the specifications and the basic mechanisms that make them possible.
The Deployment paragraph, on the contrary, provides details on how to configure and run a Wildfly application server, how to compile and deploy the server EAR containing the EJB and, finally, how to use it from a client application.
Java EE and EJB specifications have been modified over the years and the process to write and deploy client-server applications relying on EJBs has changed as well. For the remainder of this report the following versions of specifications, applications and frameworks will be considered:

- Java Platform, Standard Edition: *8*
- Java Platform, Enterprise Edition: *7*
- Enterprise JavaBean: *3.2*
- Wildfly: *10.1.0.Final*
- Apache Maven: *3.3.9*

2 Implementation

The application is made up of one interface (*TimeI*) and two classes (*TimeBean* and *Client*). The interface is shared between the client and the server hosts and represents the API they can use to communicate. As required by the specifications, *TimeI* only defines the method *getDate* that a client can use to ask such information to the server. *TimeBean* is the server-specific class implementing the EJB. Accordingly to the EJB 3.2 specifications, the class implements *TimeI* as a remote interface (the method simply returns a *String* representing the current date and time). Moreover, it is annotated with *@Stateless*, to make explicit the type of the EJB, and with *@Remote(TimeI.class)* to declare the EJB remote interface it is wired to. Finally, the *Client* class contains the code necessary to use the EJB. The parameters required by the JNDI are provided with the *jndi.properties* configuration file [1] . It contains the following key-value pairs:

- *java.naming.factory.initial=org.jboss.naming.remote.client.InitialContextFactory*
  the Context [2] class to be used by the JNDI;
- *java.naming.provider.url=http-remoting://127.0.0.1:8080*
  the address of the EJB directory service;
- *java.naming.factory.url.pkgs=org.jboss.ejb.client.naming*
  the type of the naming service;
- *jboss.naming.client.ejb.context=true*
  a flag to be used when dealing with EJB;
- *java.naming.security.principal=myuser*
  the username of an account authorized on the server to interact with the EJBs;
- *java.naming.security.credentials=mypass*
  the password of the account.

In addition, the key bound to the remote EJB is supplied with a command line argument. If not, the default value (*server-ear/server-ejb/TimeBean!ejbsample.common.TimeI*) is used. *Client main* method can be run once the EJB is deployed inside the container of the application server. Its logic is straightforward: a JNDI context is built according to the *jndi.properties* that are automatically loaded by the Java virtual machine; a proxy for the EJB is created by using the input binding key; the remote method calls are executed.

3 Deployment

The *src.zip* file contains two configuration files for Widlfy and the source code of both the client and the server. The Apache Maven framework is required to easily compile and package the client JAR application and the EAR to be deployed on the application

[1] http://docs.oracle.com/javase/jndi/tutorial/beyond/env/source.html
[2] https://docs.oracle.com/javase/8/docs/api/javax/naming/Context.html

server, thus it has to be properly installed on the system.
Download the *src.zip* file and the Wildfly binary tarball [3] into a folder. All the following commands have to be executed from the download directory.
To unzip both the archives just type:

```
$ unzip src; tar -xvzf wildfly-10.1.0.Final.tar.gz
```

To temporarily set the shell variable *JBOSS_HOME*, required by Wildfly, type:

```
$ export JBOSS_HOME=./wildfly-10.1.0.Final
```

An authorized user is required to invoke the EJB. It can be created by running the Wildfly *add-user.sh* script. In order to simplify the configuration process, some files including a test user have already been provided. It is sufficient to copy them inside the Wildfly folder with the command:

```
$ cp mgmt-* $JBOSS_HOME/standalone/configuration
```

Wildlfy is now properly configured and can be executed by typing:

```
$ $JBOSS_HOME/bin/standalone.sh
```

If everything goes fine the console should print something similar to this (some lines have been omitted for clarity):

```
=========================================================================

  JBoss Bootstrap Environment

  JBOSS_HOME: ./wildfly-10.1.0.Final

  JAVA: /usr/lib/jvm/java-8-openjdk/bin/java

  JAVA_OPTS:  -server -Xms64m -Xmx512m -XX:MetaspaceSize=96M -XX:MaxMetaspaceSize=256m
    -Djava.net.preferIPv4Stack=true -Djboss.modules.system.pkgs=org.jboss.byteman
    -Djava.awt.headless=true

=========================================================================

15:41:08,280 INFO  [org.jboss.modules] (main) JBoss Modules version 1.5.2.Final
15:41:08,657 INFO  [org.jboss.msc] (main) JBoss MSC version 1.2.6.Final
15:41:08,762 INFO  [org.jboss.as] (MSC service thread 1-5) WFLYSRV0049: WildFly Full
  10.1.0.Final (WildFly Core 2.2.0.Final) starting
15:41:10,260 INFO  [org.jboss.as.server] (Controller Boot Thread) WFLYSRV0039: Creating
  http management service using socket-binding (management-http)
15:41:10,320 INFO  [org.xnio] (MSC service thread 1-2) XNIO version 3.4.0.Final
15:41:10,337 INFO  [org.xnio.nio] (MSC service thread 1-2) XNIO NIO Implementation
  Version 3.4.0.Final
...
15:41:10,424 INFO  [org.jboss.remoting] (MSC service thread 1-1) JBoss Remoting
  version 4.0.21.Final
...
```

[3] Wildfly 10.1.0.Final - Java EE7 Full and Web Distribution, 2016-08-19

```
15:41:10,695 INFO  [org.jboss.as.naming] (MSC service thread 1-6) WFLYNAM0003:
  Starting Naming Service
15:41:10,701 INFO  [org.jboss.as.mail.extension] (MSC service thread 1-3) WFLYMAIL0001:
  Bound mail session [java:jboss/mail/Default]
...
15:41:10,886 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-6) WFLYUT0012:
  Started server default-server.
15:41:10,887 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-7) WFLYUT0018:
  Host default-host starting
15:41:11,050 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-1) WFLYUT0006:
  Undertow HTTP listener default listening on 127.0.0.1:8080
...
15:41:11,414 INFO  [org.jboss.as.server.deployment.scanner] (MSC service thread 1-2)
  WFLYDS0013: Started FileSystemDeploymentService for directory
  /home/gianvito/Desktop/final_test/./wildfly-10.1.0.Final/standalone/deployments
15:41:11,594 INFO  [org.wildfly.extension.undertow] (MSC service thread 1-7) WFLYUT0006:
  Undertow HTTPS listener https listening on 127.0.0.1:8443
...
15:41:11,939 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0060:
  Http management interface listening on http://127.0.0.1:9990/management
15:41:11,940 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0051:
  Admin console listening on http://127.0.0.1:9990
15:41:11,940 INFO  [org.jboss.as] (Controller Boot Thread) WFLYSRV0025:
  WildFly Full 10.1.0.Final (WildFly Core 2.2.0.Final) started in 4073ms
  - Started 331 of 577 services (393 services are lazy, passive or on-demand)
```

The output shows the URLs of the running services, the versions of the main technologies in use and the deployment folder of the application server. Wildfly will now keep running in the console, so another terminal has to been opened in the same working directory to run new commands. The shell variable *JBOSS_HOME* has to be set again:

```
$ export JBOSS_HOME=./wildfly-10.1.0.Final
```

To compile and package the EAR containing the EJB just type:

```
$ mvn -f server/ clean install package
```

Maven will download the required dependencies, will compile the code and will build the EAR automatically by using some plugins.
In a similar way, to compile and package the JAR containing the client application just type:

```
$ mvn -f client/ clean install package
```

By default, Wildfly automatically deploys new EARs present in a specific folder (the full path can be read in the previous console log). So, to deploy the application, just copy the EAR built by Maven with the command:

```
$ cp server/server-ear/target/server-ear.ear $JBOSS_HOME/
  standalone/deployments
```

The previous terminal, with Wildfly still running, will automatically output:

```
15:53:16,773 INFO [org.jboss.as.repository] (DeploymentScanner-threads - 2) WFLYDR0001:
  Content added at location /home/gianvito/Desktop/final_test/./wildfly-10.1.0.Final/
  standalone/data/content/04/4b34d28ce43454e13a74aa3345450b3071da86/content
15:53:16,832 INFO [org.jboss.as.server.deployment] (MSC service thread 1-7) WFLYSRV0027:
  Starting deployment of "server-ear.ear" (runtime-name: "server-ear.ear")
15:53:16,995 INFO [org.jboss.as.server.deployment] (MSC service thread 1-5) WFLYSRV0207:
  Starting subdeployment (runtime-name: "server-ejb.jar")
15:53:17,229 INFO [org.jboss.weld.deployer] (MSC service thread 1-2) WFLYWELD0003:
  Processing weld deployment server-ear.ear
15:53:17,381 INFO [org.hibernate.validator.internal.util.Version] (MSC service thread
  1-2) HV000001: Hibernate Validator 5.2.4.Final
15:53:17,575 INFO [org.jboss.weld.deployer] (MSC service thread 1-5) WFLYWELD0003:
  Processing weld deployment server-ejb.jar
15:53:17,607 INFO [org.jboss.as.ejb3.deployment] (MSC service thread 1-5) WFLYEJB0473:
  JNDI bindings for session bean named 'TimeBean' in deployment unit 'subdeployment
  "server-ejb.jar" of deployment "server-ear.ear"' are as follows:

        java:global/server-ear/server-ejb/TimeBean!ejbsample.common.TimeI
        java:app/server-ejb/TimeBean!ejbsample.common.TimeI
        java:module/TimeBean!ejbsample.common.TimeI
        java:jboss/exported/server-ear/server-ejb/TimeBean!ejbsample.common.TimeI
        java:global/server-ear/server-ejb/TimeBean
        java:app/server-ejb/TimeBean
        java:module/TimeBean

15:53:17,999 INFO [org.infinispan.configuration.cache.EvictionConfigurationBuilder]
  (ServerService Thread Pool -- 67) ISPN000152: Passivation configured without an
  eviction policy being selected. Only manually evicted entities will be passivated.
15:53:18,000 INFO [org.infinispan.configuration.cache.EvictionConfigurationBuilder]
  (ServerService Thread Pool -- 67) ISPN000152: Passivation configured without an
  eviction policy being selected. Only manually evicted entities will be passivated.
15:53:18,576 INFO [org.jboss.as.clustering.infinispan] (ServerService Thread Pool
  -- 67) WFLYCLINF0002: Started client-mappings cache from ejb container
15:53:19,955 INFO [org.jboss.as.server] (DeploymentScanner-threads - 2) WFLYSRV0010:
  Deployed "server-ear.ear" (runtime-name : "server-ear.ear")
15:57:55,772 INFO [org.jboss.ejb.client] (pool-1-thread-2) JBoss EJB Client version
  2.1.4.Final
```

The output shows the EJB bindings required by the client JNDI.
Finally, run the client with:

```
$ java -jar -Dejb='server-ear/server-ejb/TimeBean!ejbsample
  .common.TimeI' client/target/client-v1.0.jar
```

The output should be something similar to this:

```
Trying to connect to EJB bound at server-ear/server-ejb/TimeBean!ejbsample.common.TimeI

Oct 10, 2016 3:57:55 PM org.xnio.Xnio <clinit>
INFO: XNIO version 3.4.0.Final
Oct 10, 2016 3:57:55 PM org.xnio.nio.NioXnio <clinit>
INFO: XNIO NIO Implementation Version 3.4.0.Final
```

```
Oct 10, 2016 3:57:55 PM org.jboss.remoting3.EndpointImpl <clinit>
INFO: JBoss Remoting version 4.0.21.Final
Oct 10, 2016 3:57:55 PM org.jboss.ejb.client.remoting.VersionReceiver handleMessage
INFO: EJBCLIENT000017: Received server version 2 and marshalling strategies [river]
Oct 10, 2016 3:57:55 PM org.jboss.ejb.client.remoting.RemotingConnectionEJBReceiver
  associate
INFO: EJBCLIENT000013: Successful version handshake completed for receiver context
  EJBReceiverContext{clientContext=org.jboss.ejb.client.EJBClientContext@6fd02e5,
  receiver=Remoting connection EJB receiver [connection=Remoting connection <1f01d682>
  on endpoint "config-based-naming-client-endpoint" <5bcab519>,channel=jboss.ejb,
  nodename=xps13]} on channel Channel ID b3cc3d3d (outbound) of Remoting connection
  31dc339b to /127.0.0.1:8080 of endpoint "config-based-naming-client-endpoint"
  <5bcab519>
Oct 10, 2016 3:57:55 PM org.jboss.ejb.client.EJBClient <clinit>
INFO: JBoss EJB Client version 2.1.4.Final

Server time: 2016/10/10 15:57:55

Server time: 2016/10/10 15:57:55
```