FREE eBook

LEARNING

ejb

Free unaffiliated eBook created from
Stack Overflow contributors.

#ejb

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ejb

It is an unofficial and free ejb ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ejb.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with ejb

Remarks

This section provides an overview of what ejb is, and why a developer might want to use it.

It should also mention any large subjects within ejb, and link out to the related topics. Since the Documentation for ejb is new, you may need to create initial versions of those related topics.

Examples

Setting up EJB with JBoss AS 7.1

1. Overview

In this article we are going to discuss how to get started with Enterprise JavaBeans (EJB). We will use JBoss AS 7.1.1.Final, but you are free to use any server of your choice.

2. Maven Dependencies for Bean

In order to use EJB make sure you add the latest version of it to the dependencies section of your pom.xml file:

```
<dependency>
    <groupId>org.jboss.spec.javax.ejb</groupId>
    <artifactId>jboss-ejb-api_3.2_spec</artifactId>
    <version>1.0.0.Final</version>
</dependency>
```

Make sure you add the JBoss dependencies properly as we will be using JBoss as our application server in this tutorial. In the later part of the tutorial we will discuss in details on how to setup the maven build for the project.

3. EJB Remote

Let's first create the the Bean Interface called HelloWorldRemote.

```
public interface HelloWorldRemote {
    public String getHelloWorld();
}
```

Now we will implement the above interface and name it as `HelloWorldBean`.

```
@Stateless
public class HelloWorldBean implements HelloWorldRemote {

    public HelloWorldBean() {
```

```
    }

    @Override
    public String getHelloWorld(){
        return ("Hello World");
    }
}
```

Note the `@Stateless` notation on top of the class declaration. It denotes a stateless session bean.

4. Maven Setup for Remote Bean

In this section we will discuss how to setup maven to build and run the application on the server.

Let's look at the plugins one by one.

4.1. Compiler Plugin

The maven-compiler-plugin is used to compile the sources of our project.

Here we have used the version 2.3.1 of the plugin with the source and target JDK set to 1.7 under configuration.

We have defined these settings as properties inside tag and referring it through ${property}.

```
<version.compiler.plugin>2.3.1</version.compiler.plugin>
<!-- maven-compiler-plugin -->
<maven.compiler.target>1.7</maven.compiler.target>
<maven.compiler.source>1.7</maven.compiler.source>
```

4.2 The EJB Plugin

This plugin generates Bean file as well as the associated client jar.

We have specified the ejb version as 3.2 and the generateClient property is set to true which generates the client.

4.3 Deploying in JBoss

The jboss-as-maven-plugin is used to deploy, redeploy, undeploy or run the application in JBoss AS 7.

In this configuration we specify the build file name same as the project build filename which is by default of the form artifactid-version in our case `ejb-remote-1.0-SNAPSHOT`.

4.4 Required Maven Dependencies for EJB

jboss-javaee-6.0 defines the version of JBoss' Java EE 6 APIs we want to use.

JBoss distributes a complete set of Java EE 6 APIs including a Bill of Materials (BOM).

A BOM specifies the versions of a stack (or a collection) of artifacts. We specify this in tag so that

we always get the correct versions of artifacts. The type of this dependency itself a pom which contains the required dependencies.

```
<dependency>
    <groupId>org.jboss.spec</groupId>
    <artifactId>jboss-javaee-6.0</artifactId>
    <version>${version.org.jboss.spec.jboss.javaee.6.0}</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

4.5 Annotations

The following will get the annotations dependency:

```
<dependency>
    <groupId>org.jboss.spec.javax.annotation</groupId>
    <artifactId>jboss-annotations-api_1.1_spec</artifactId>
    <scope>provided</scope>
</dependency>
```

4.6 EJB version 3.2

In the following piece of code we get the latest version of the specifications:

```
<dependency>
    <groupId>org.jboss.spec.javax.ejb</groupId>
    <artifactId>jboss-ejb-api_3.2_spec</artifactId>
    <version>1.0.0.Final</version>
</dependency>
```

To run the above project in a JBoss server we need to first run:

```
mvn clean install
```

Then we need to deploy it to a running JBoss server by running the following maven command:

```
jboss-as:deploy
```

Now you should see the jar file being deployed in the jboss server.

Alternatively you can copy the available jar from the target folder in the project and paste it in the webapp folder of the server.

5. Setting up the Client Project

After creating the remote bean we should test the deployed bean by creating a client.

First let's discuss the maven setup for the project.

5.1 Maven Plugins used

The maven-compiler-plugin is used to compile the sources of your project.

We specified jdk 1.7 version for source and target classes.

Our a client is a Java program, to run it we use the `exec-maven-plugin` which helps to execute system and Java programs. We need to specify the executable (i.e. java), classpath, and the java class (com.baeldung.ejb.client.Client).

The classpath is left empty because the plugin includes the necessary classpath arguments based on the dependencies provided.

5.2 Maven Dependencies for EJB3 client

In order to run EJB3 client we need to include the following dependencies.

We depend on the EJB remote business interfaces of this application to run the client. So we need to specify the ejb client jar dependency. The tag with value “ejb-client” is used to specify this project’s dependency on the EJB client jar.

```
<dependency>
    <groupId>com.theopentutorials.ejb3</groupId>
    <artifactId>ejbmavendemo</artifactId>
    <type>ejb-client</type>
    <version>${project.version}</version>
</dependency>
```

The dependencies `jboss-transaction-api_1.1_spec`, `jboss-ejb-api_3.1_spec`, `jboss-ejb-client`, `xnio-api`, `xnio-nio`, `jboss-remoting`, `jboss-sasl`, `jboss-marshalling-river` have scope as runtime because these are required run time and not during compile time.

The dependencies **jboss-javaee-6.0** and **jboss-as-ejb-client-bom** under dependencyManagement have scope as import. This is used to include dependency management information from a remote POM into the current project. These remote POMs are provided by JBoss which contains the necessary dependencies for running the client.

5.3 JBoss EJB Client properties

Create a file under “src/main/resources” and name it as jboss-ejb-client.properties.

```
remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED=false
remote.connections=default
remote.connection.default.host=localhost
remote.connection.default.port = 4447
remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOANONYMOUS=false
```

6. Creating the Client Class

First we create a ClientUtility class:

```
public class ClientUtility {
    private static Context initialContext;
```

```
    private static final String PKG_INTERFACES = "org.jboss.ejb.client.naming";

    public static Context getInitialContext() throws NamingException {
        if (initialContext == null) {
            Properties properties = new Properties();
            properties.put(Context.URL_PKG_PREFIXES, PKG_INTERFACES);
            initialContext = new InitialContext(properties);
         }
        return initialContext;
    }
}
```

Now let's create the actual Client class that will consume the bean that we deployed in the server:

```
public class Client {

    //The lookup method to get the EJB name
    private static HelloWorldRemote doLookup() {
        Context context = null;
        HelloWorldRemote bean = null;
        try {
            // 1. Obtaining Context
            context = ClientUtility.getInitialContext();
            // 2. Generate JNDI Lookup name
            String lookupName = getLookupName();
            // 3. Lookup and cast
            bean = (HelloWorldRemote) context.lookup(lookupName);

        } catch (NamingException e) {
            e.printStackTrace();
        }
        return bean;
    }

    private static String getLookupName() {

         // The app name is the EAR name of the deployed EJB without .ear suffix.
         // Since we haven't deployed the application as a .ear, the app name for
         // us will be an empty string

        String appName = "";


         // The module name is the JAR name of the deployed EJB without the .jar
         // suffix.
        String moduleName = "ejb-remote-0.0.1-SNAPSHOT";


        // AS7 allows each deployment to have an (optional) distinct name. This
        // can be an empty string if distinct name is not specified.
        String distinctName = "";

        // The EJB bean implementation class name
        String beanName = "HelloWorldBean";

        // Fully qualified remote interface name
        final String interfaceName = "com.baeldung.ejb.tutorial.HelloWorldRemote";

        // Create a look up string name
        String name = "ejb:" + appName + "/" + moduleName + "/" + distinctName
```

```
                + "/" + beanName + "!" + interfaceName;

        return name;
    }
}
```

The Client class consumes the bean and outputs the result.

7. Conclusion

So we have created an EJB server and a client which consumes the service. The project can be run on any Application Server.

Read Getting started with ejb online: https://riptutorial.com/ejb/topic/5704/getting-started-with-ejb

Credits

S. No	Chapters	Contributors
1	Getting started with ejb	Community, Pritam Banerjee, RamenChef