Simplified IDL demo

If you have Borland Application Server 4.5, you will need to install a jar file patch to get the EJB connectivity to work. If you have AppServer 4.5.1, the patch is incorporated in the application.

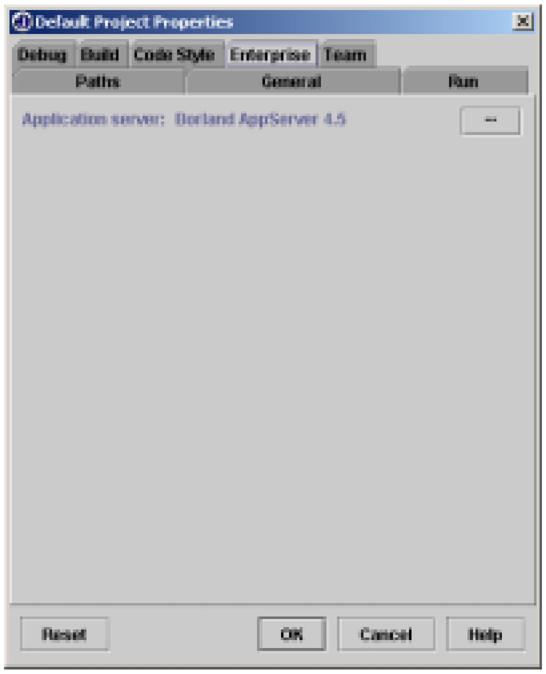
Configuring Borland AppServer 4.5 for the Patch

- Create a folder called patches in your <installAppServer>\lib folder
- Copy the delphi_bas45.jar to this location
- Restart AppServer

Configuring JBuilder

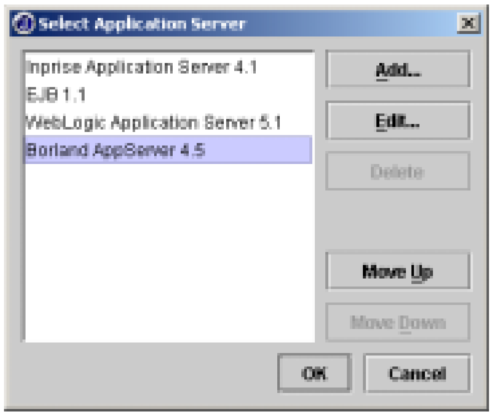
You need to have integration between JBuilder and Borland AppServer 4.5 already in place in order to complete the following steps. This is only necessary if you want to run the Container inside JBuilder.

1. Open the Default Project Properties dialog, or the Project Properties if you already have a project available.



Application server: Borland AppServer 4.5

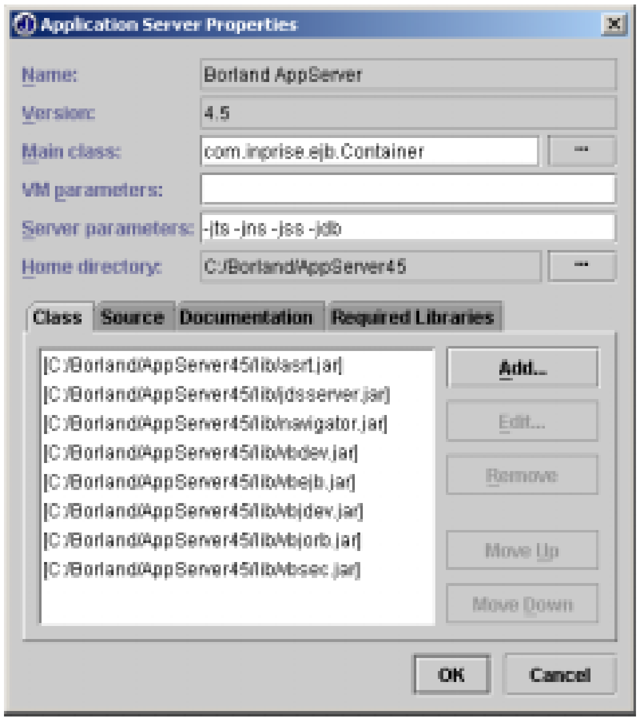
2. Select the enterprise tab and click on '…" button to change the Application Server Configuration. Select the Borland AppServer 4.5 config and click edit.



Select Application Server

Inprise Application Server 4.1
EJB 1.1
WebLogic Application Server 5.1
Borland AppServer 4.5

The Application Server properties dialog appears.

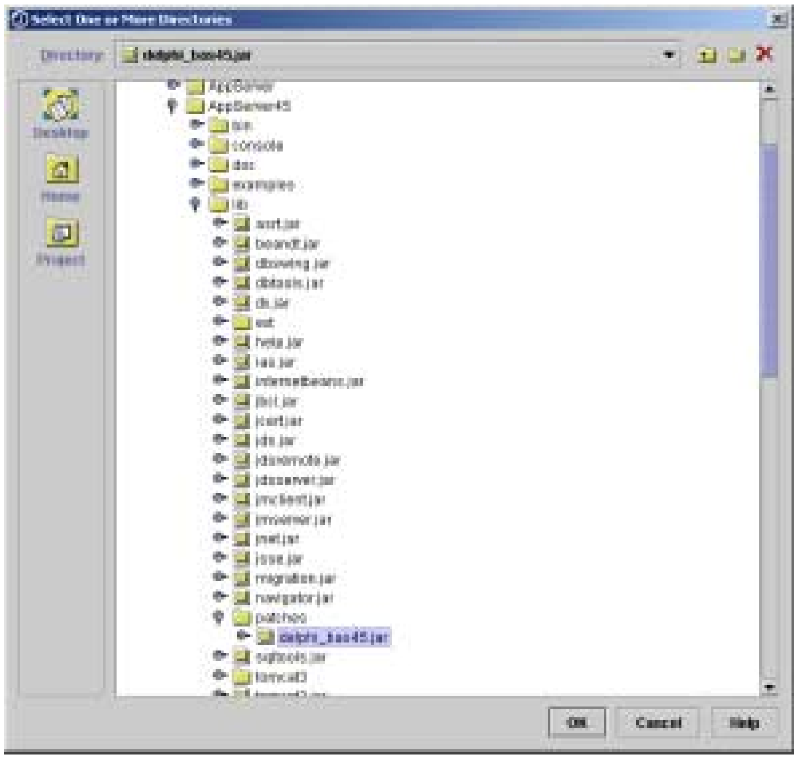


Application Server Properties

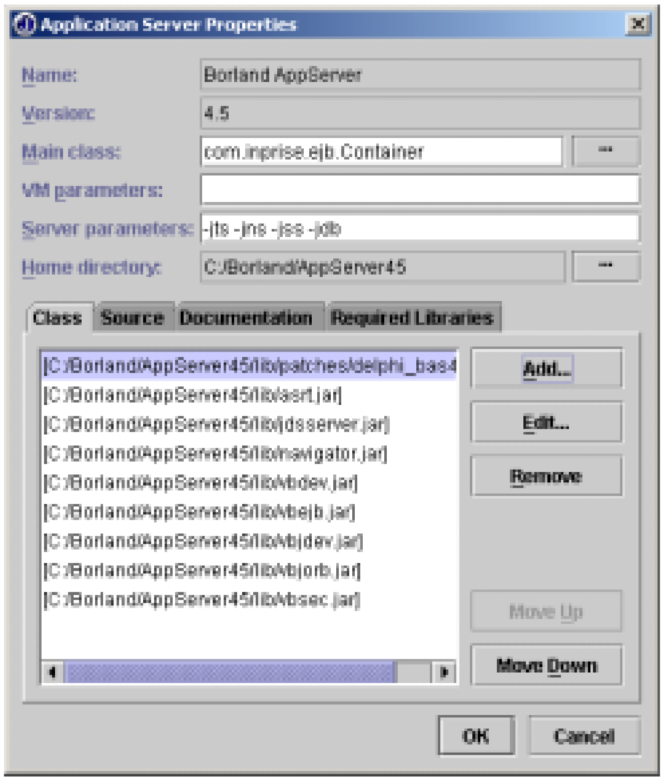
Name: Borland AppServer
Version: 4.5
Main class: com.inprise.ejb.Container
VM parameters:
Server parameters: -jts -jns -jss -jdb
Home directory: C:/Borland/AppServer45

[C:/Borland/AppServer45/lib/asrt.jar]
[C:/Borland/AppServer45/lib/jdsserver.jar]
[C:/Borland/AppServer45/lib/navigator.jar]
[C:/Borland/AppServer45/lib/vbdev.jar]
[C:/Borland/AppServer45/lib/vbejb.jar]
[C:/Borland/AppServer45/lib/vbjdev.jar]
[C:/Borland/AppServer45/lib/vbjorb.jar]
[C:/Borland/AppServer45/lib/vbsec.jar]

We need to add the patch to the configuration. Click Add and browse to the <AppServerInstall>\lib\patches folder and select the patch.



Click Ok and make sure that the jarfile is at the top of the list. (use the Move up button)


Application Server Properties

Name: Borland AppServer
Version: 4.5
Main class: com.inprise.ejb.Container
VM parameters:
Server parameters: -jts -jns -jss -jdb
Home directory: C:/Borland/AppServer45

[C:/Borland/AppServer45/lib/patches/delphi_bas4
[C:/Borland/AppServer45/lib/asrt.jar]
[C:/Borland/AppServer45/lib/jdsserver.jar]
[C:/Borland/AppServer45/lib/navigator.jar]
[C:/Borland/AppServer45/lib/vbdev.jar]
[C:/Borland/AppServer45/lib/vbejb.jar]
[C:/Borland/AppServer45/lib/vbjdev.jar]
[C:/Borland/AppServer45/lib/vbjorb.jar]
[C:/Borland/AppServer45/lib/vbsec.jar]

This is all to add the patch to JBuilder. Close the dialog(s).

Commandline creation of Simplified IDL

Go to the outputfolder of your JBuilder project, usually called classes.

For this demo I will use the Euroconverter Enterprise JavaBean. The details are described in the whitepaper “**Getting Started Creating your first Enterprise JavaBean”.** In this demo I will show the steps to create a Delphi client for this Bean, also take a look at the excellent whitepaper of Giles Davies which shows a mobile device talking to the same EJB. This whitepaper is called “**Building WAP-enabled Applications with JBuilder and Inprise Application Server**”

The commandline is:

vbj com.inprise.ejb.sidl.ejb2sidl -o ejb.idl currencyconverter\EuroConverterHome

In this example is ejb.idl the generated outputfile and currencyconverter\EuroConverterHome is the home interface from which I want to generate the IDL file.

After completion the ejb.idl looks like this:

```
// IDL file generated by: com.inprise.ejb.sidl.ejb2sidl
//                  args: -o ejb.idl currencyconverter.EuroConverterHome
//                  date: Thu Mar 01 13:01:53 GMT+01:00 2001

#include <sidl.idl>

#pragma prefix ""

// BEGIN: declaration of sequences
typedef sequence<any> Sequence_of_Any;
// END:   declaration of sequences


module currencyconverter {
  // BEGIN: forward declaration of interfaces
  interface EuroConverter;
  interface EuroConverterHome;
  // END:   forward declaration of interfaces

  // BEGIN: declaration of interfaces
  interface EuroConverter : ::sidl::javax::ejb::EJBObject {

    // JAVA: float currencyconverter.EuroConverter.hfl2float(float) throws
    //java.rmi.RemoteException;
    float hfl2float(in float arg0);

  };
  interface EuroConverterHome : ::sidl::javax::ejb::EJBHome {

    // JAVA: currencyconverter.EuroConverter currencyconverter.EuroConverterHome.create()
    // throws javax.ejb.CreateException,java.rmi.RemoteException;
    ::currencyconverter::EuroConverter create() raises
          (::sidl::javax::ejb::CreateException);

  };
  // END:   declaration of interfaces

};
```

I have adjusted the output to make it more readable.

Create a Delphi Client

In order to create the Delphi part you also need sidl.idl. It is located in the sidl example of your AppServer installation. In order to build this example you need to have VisiBroker for Delphi installed. Copy the generated **ejb.idl** file and the **sidl.idl** file to a new folder in which you want to create your Delphi Client.

Run the idl2pas util of VisiBroker for Delphi.

Idl2pas ejb.idl

After completion there are a number of generated files in your folder:

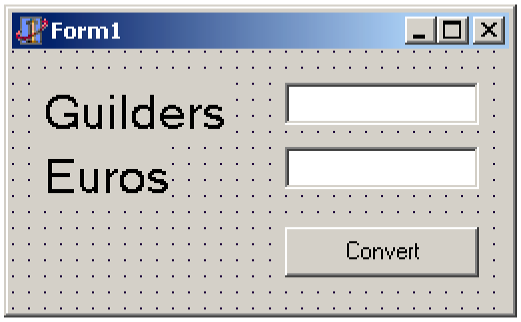
- ejb_sidl_java_lang_i.pas
- ejb_sidl_java_math_i.pas
- ejb_sidl_java_util_i.pas
- ejb_sidl_java_sql_i.pas
- ejb_sidl_javax_ejb_i.pas
- ejb_currencyconverter_i.pas
- ejb_i.pas
- ejb_sidl_java_lang_c.pas
- ejb_sidl_java_math_c.pas
- ejb_sidl_java_util_c.pas
- ejb_sidl_java_sql_c.pas
- ejb_sidl_javax_ejb_c.pas
- ejb_currencyconverter_c.pas
- ejb_c.pas
- ejb_sidl_javax_ejb_s.pas
- ejb_currencyconverter_s.pas
- ejb_sidl_javax_ejb_impl.pas
- ejb_currencyconverter_impl.pas

Start Delphi and create a new project in the same folder. Add two of the generated files to your project, they are:

- ejb_currencyconverter_i.pas
- ejb_currencyconverter_c.pas

Project, Add to project, select the files and click Ok.

Go to the Form of your Delphi App and create a simple user interface:



We only need a number of this to the code:

<u>Add the added unit to your unit</u>
File, Use unit and select both units:
Also add Corba to the uses clause.

The result will look like this:

```
implementation
Uses
   Corba,
   ejb_currencyconverter_c,
   ejb_currencyconverter_i;
```

<u>Add two variables to your unit:</u>
Just below the uses clause add the following code.

```
Var
  Home: EuroConverterHome;
  Remote: EuroConverter;
```

<u>Add a eventhandler to the OnCreate event of the Form.</u>
The event handler will look like:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  CorbaInitialize;

  Home := TEuroConverterHomeHelper.Bind('sidl/EuroConverter');
  Remote := Home._create as ejb_currencyconverter_i.EuroConverter;
end;
```

<u>Add a eventhandler to the OnClick of the button event.</u>
The event handler will look like:

```
procedure TForm1.Button1Click(Sender: TObject);
Var
  HflASText: String;
  EuroAsText: String;
  Hfl,Euro: Single;
```

```
begin
   HflASText := Edit1.Text;
   Hfl := StrToFloat(HflASText);

   Euro :=  Remote.hfl2float(hfl);

   EuroAsText := FloattoStr(Euro);
   Edit2.Text := EuroAsText;
end;
```

Before we run the example we need to make that everything is ok. In my version of idl2pas (A Beta which also includes the serverside support) I needed to make some modifications.

First I needed to add a uses clause to one of the generated files, I added the ejb_sidl_javax_ejb_c unit to interface uses clause of unit ejb_currencyconverter_c;

Second the create method of the home interface translates to a _create method in Delphi. In unit ejb_currencyconverter_c I had to change

```
function  TEuroConverterHomeStub._create :
ejb_currencyconverter_i.EuroConverter;
var
  _Output: CORBA.OutputStream;
  _Input : CORBA.InputStream;
begin
  inherited CreateRequest('_create',True, _Output);
  inherited _Invoke(_Output, _Input);
  Result := ejb_currencyconverter_c.TEuroConverterHelper.Read(_Input);
end;
```

to

```
function  TEuroConverterHomeStub._create :
ejb_currencyconverter_i.EuroConverter;
var
  _Output: CORBA.OutputStream;
  _Input : CORBA.InputStream;
begin
  inherited _CreateRequest('create',True, _Output);
  inherited _Invoke(_Output, _Input);
  Result := ejb_currencyconverter_c.TEuroConverterHelper.Read(_Input);
end;
```

Just remove the underscore in front of _create, otherwise you will get a bad operation exception.

We are now ready to test the example. Make sure your container or the AppServer is running.