Using the EntireX Wrapper for Enterprise JavaBeans

This chapter covers the following topics:

Generation Process

To generate the Enterprise JavaBeans (EJB) source code, use the *EntireX Workbench*. This can be done interactively with the UI of the *EntireX Workbench*. The generation is controlled by the following properties:

Default Properties for the IDL File

Description	Data Type	Default Value
Broker ID	String	Localhost:1971
Server class	String	RPC
Service class	String	SRV1
Service name	String	CALLNAT
Package name	String	IDL file name without extension
Package name prefix	String	empty string

Generated Files

During the generation process for each library with the name *<libname>*, the following interfaces and classes source files are created in the subdirectory *EJB* of the home directory of the *<name>.idl* file.

- The interfaces are created in the subdirectory:

  <Package Name Prefix><Package Name><file.separator>interfaces.

  They will be a component of the package: *<Package Name Prefix><Package Name>.interfaces*

Naming Conventions	Description
EJB<libname>.java	The remote interface.
EJB<libname>Home.java	The home interface.

- The EJB classes will be created in the subdirectory:

  <Package Name Prefix><Package Name><file.separator>ejb

  They will be a component of the package: *<Package Name Prefix><Package Name>.ejb*

Naming Conventions	Description
EJB<libname>Bean.java	The enterprise bean class.
<libname><innerclassname>.java	Serializable classes for Software AG IDL groups/structs.
<libname><progname>Input.java	Serializable holder class for all IN and IN/OUT parameters of the program.
<libname><progname>Output.java	Serializable holder class for all OUT and IN/OUT parameters of the program.
<libname>Mapper.java	Mapper class.

To build the JAR files for the different application servers, we generate an Ant script which uses the XDoclet tool. This file will be generated in the *EJB* subdirectory:

<Package Name Prefix><Package Name>.xml

If the package prefix/name contains dots, subdirectories will be created, for example: *abc.def.library* will become *abc/def/library/...*

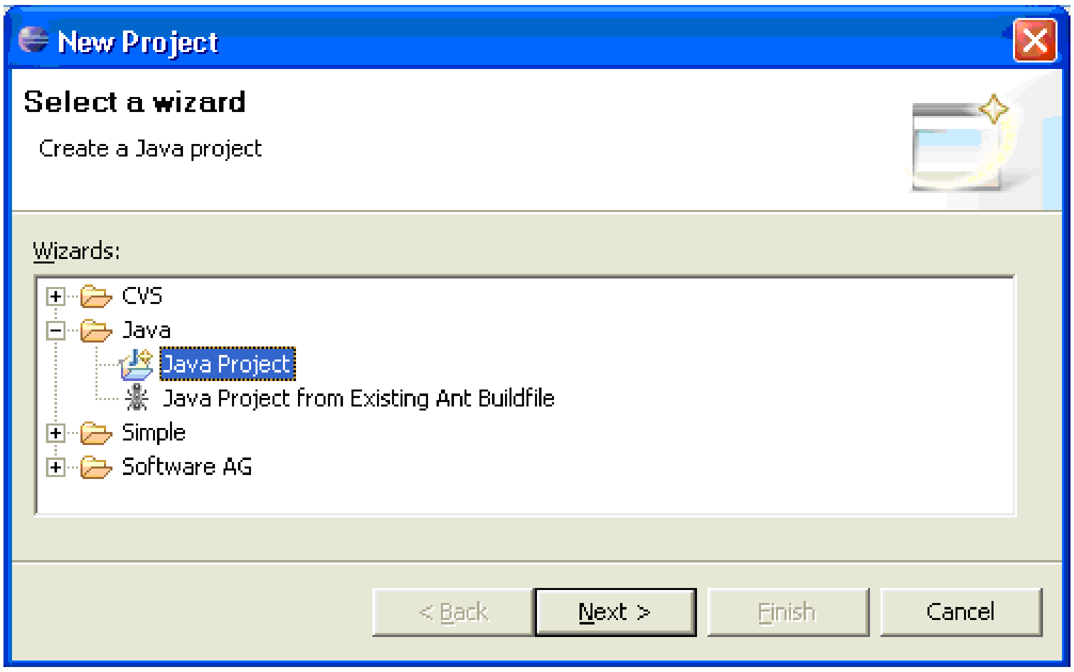
Using the Wrapper for EJB Interactively

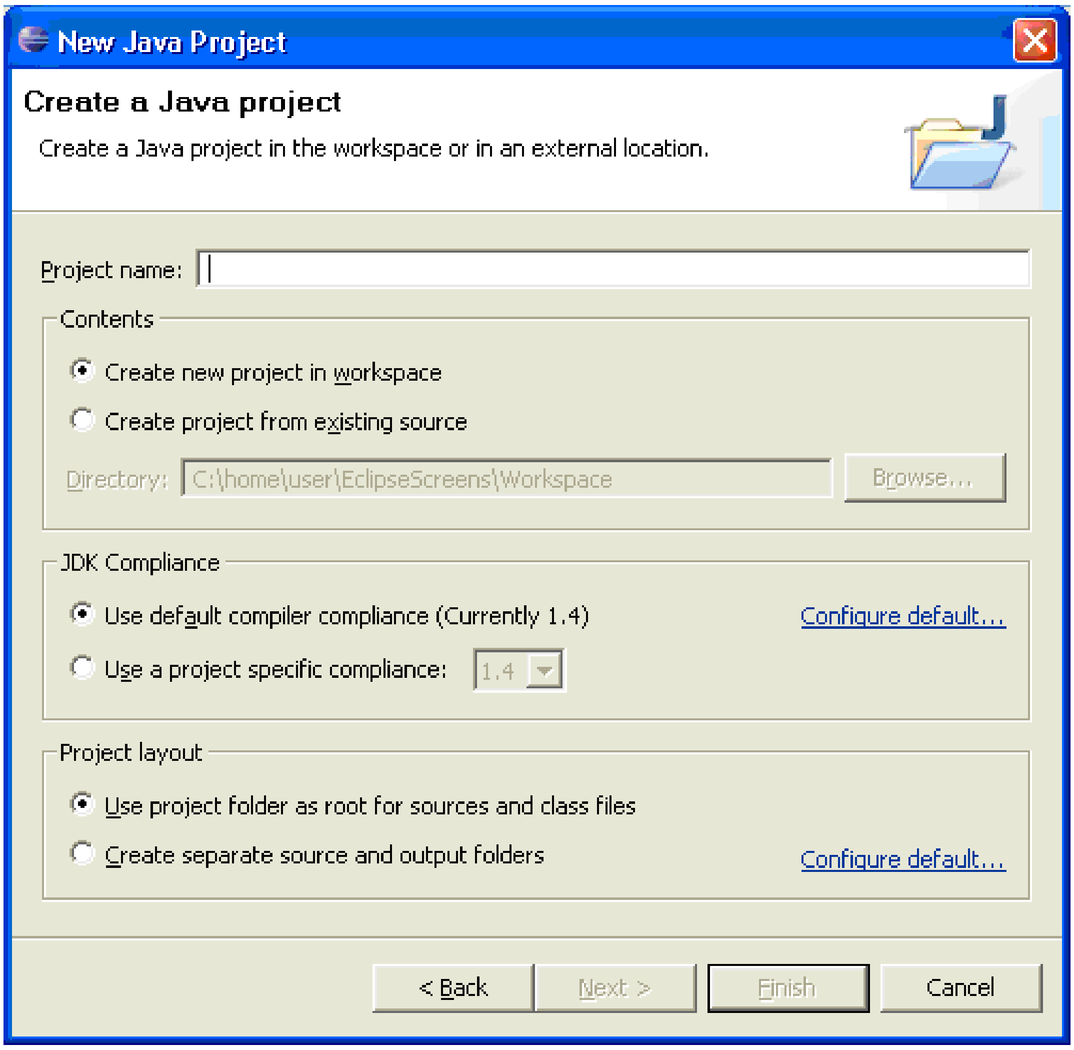
- Using the Wrapper for EJB Functions
- Setting/Modifying EntireX Enterprise JavaBeans Preferences
- Setting/Modifying IDL File Properties

Using the Wrapper for EJB Functions

To use the Wrapper for EJB functions

1. Open the *EntireX Workbench*.
2. Create a new Java Project (e.g.: "EJB"), using **File** > **New** > **Project**.




New Java Project

Create a Java project

Create a Java project in the workspace or in an external location.

Project name:

Contents

(●) Create new project in workspace

( ) Create project from existing source

Directory: C:\home\user\EclipseScreens\Workspace

JDK Compliance

(●) Use default compiler compliance (Currently 1.4)

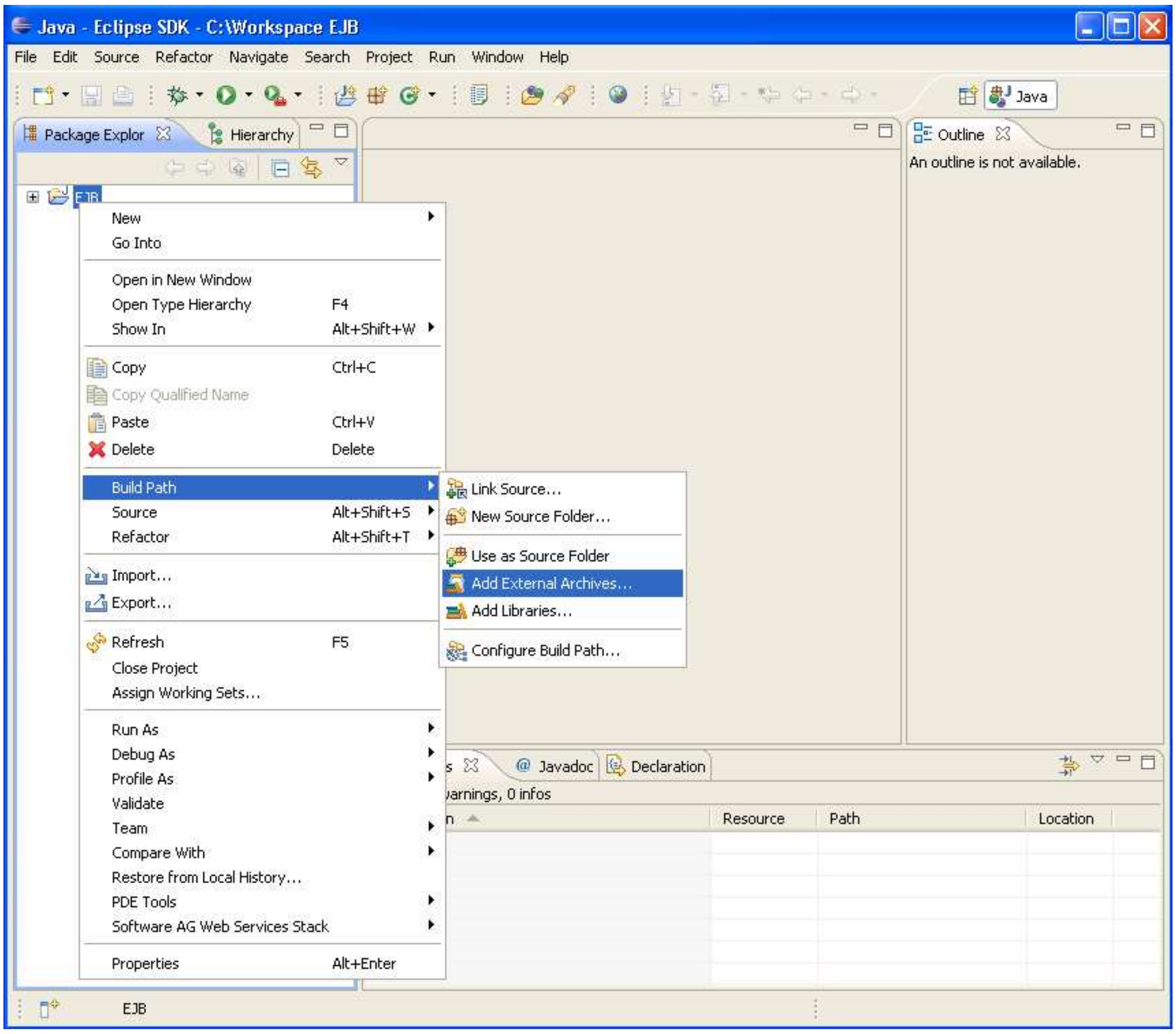
( ) Use a project specific compliance: 1.4

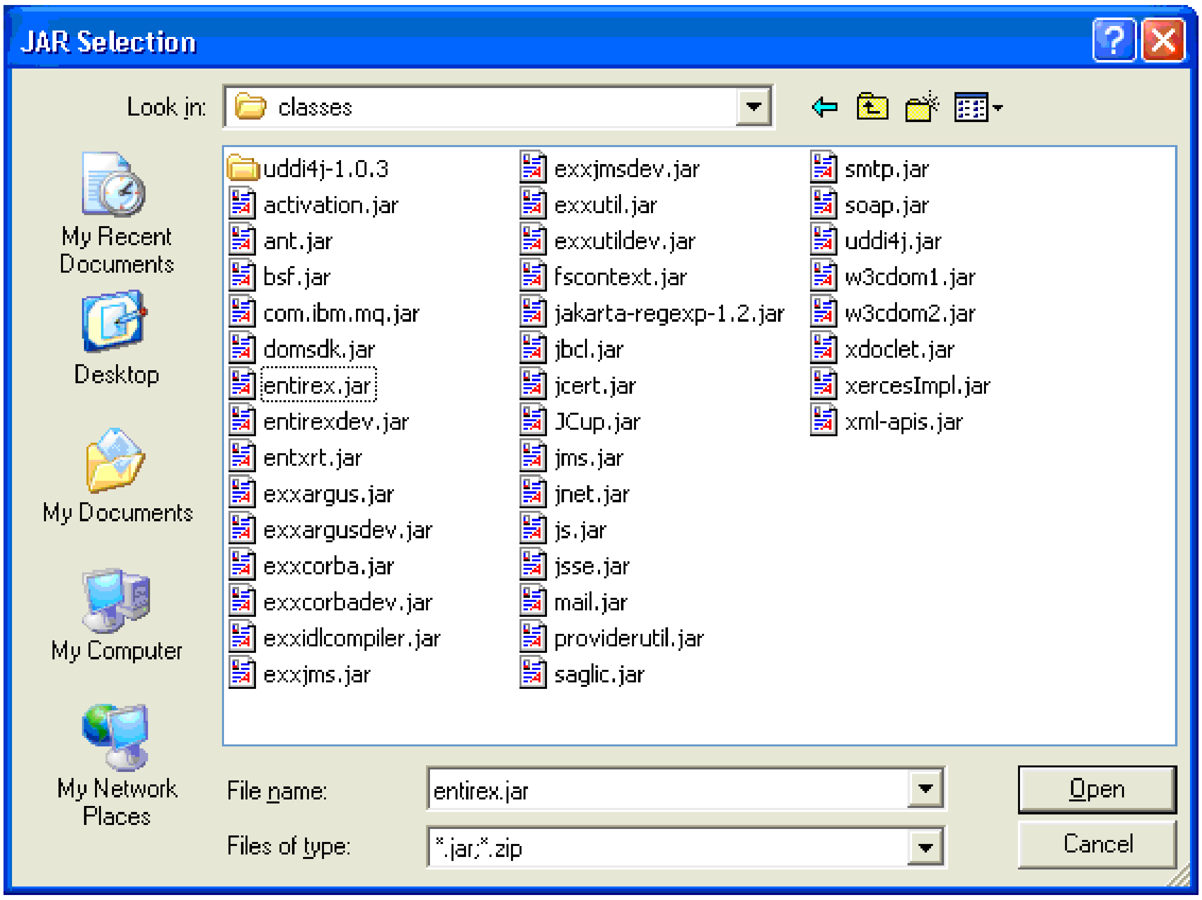
Project layout

(●) Use project folder as root for sources and class files

( ) Create separate source and output folders

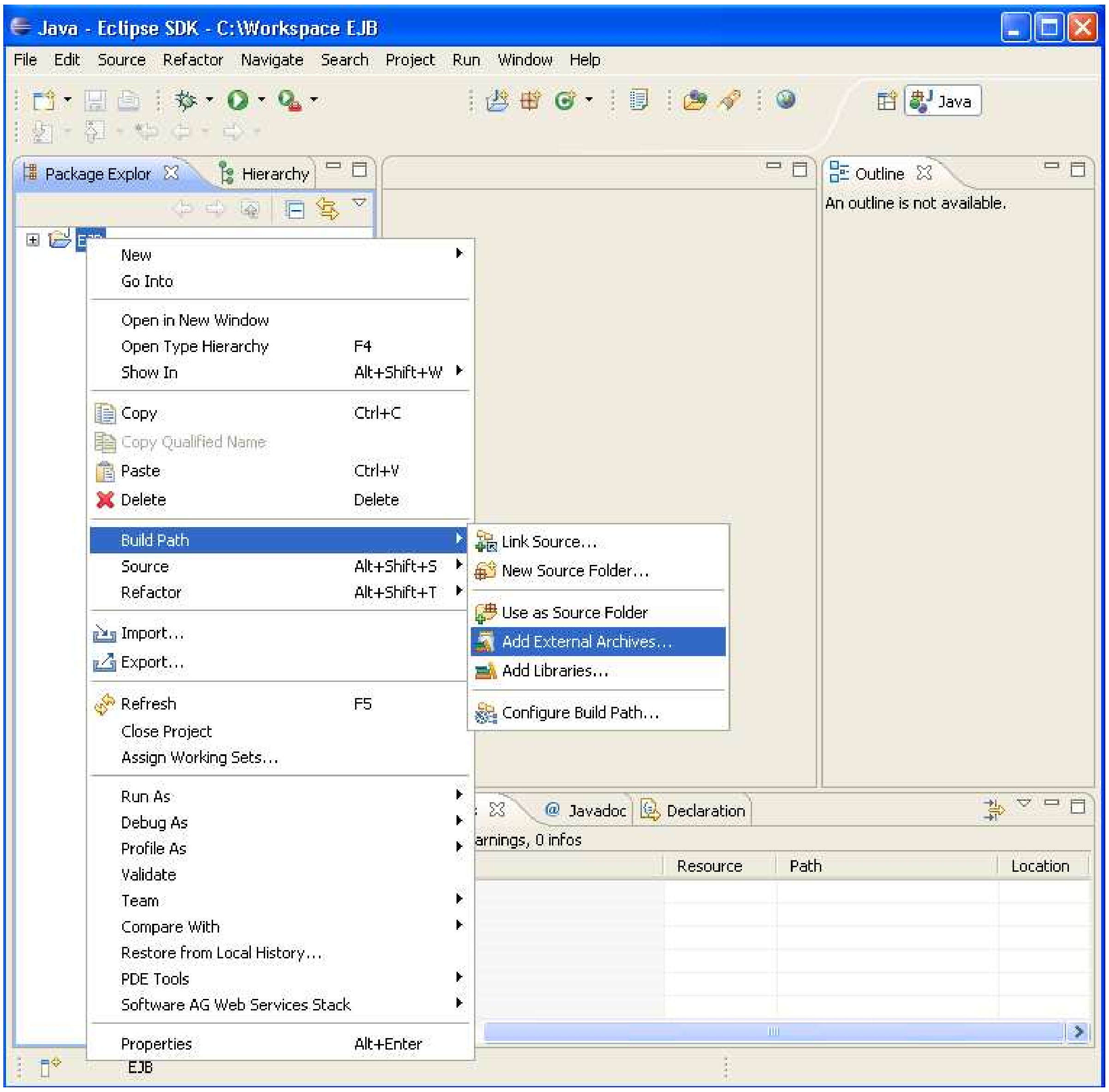
3. Add the EntireX classes to the build path of the project (*entirex.jar*).

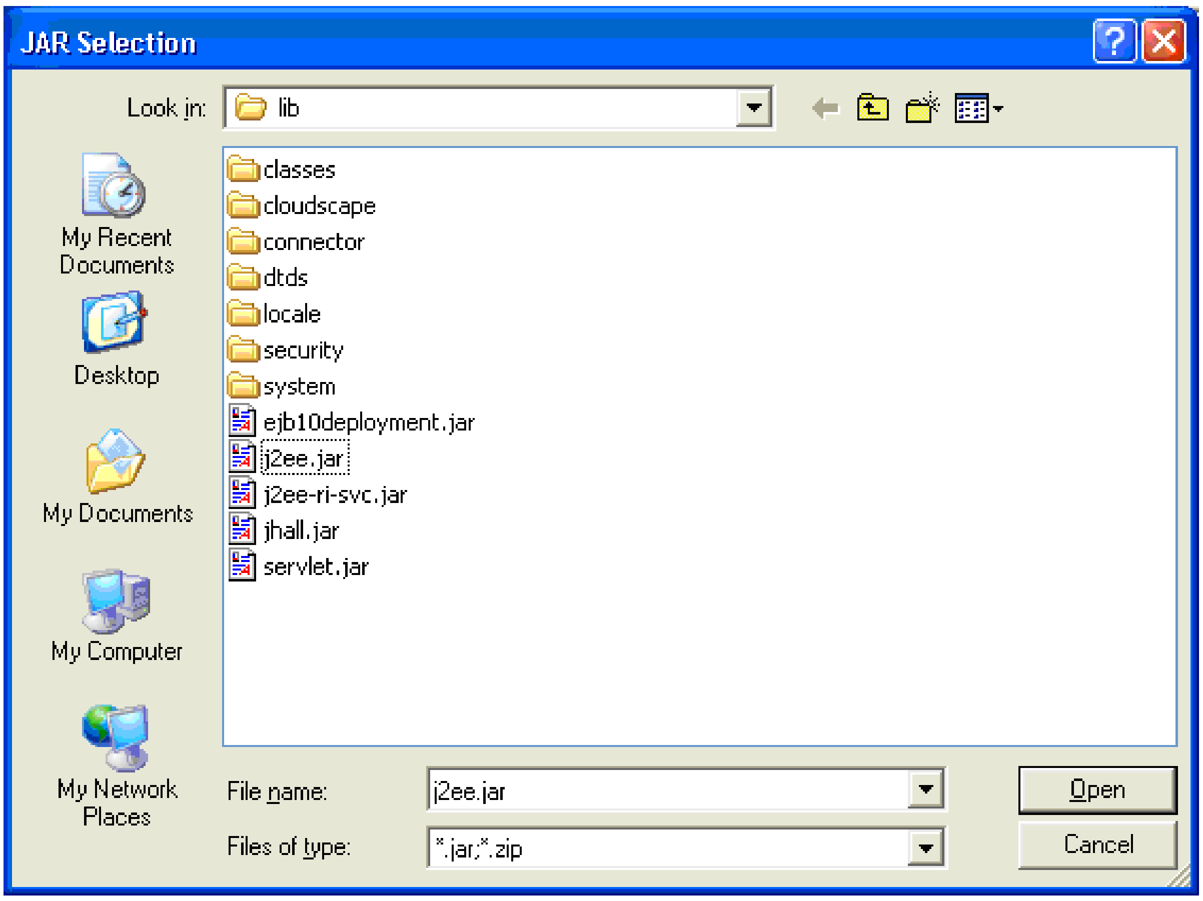

Java - Eclipse SDK - C:\Workspace EJB
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explor
Hierarchy
Outline
An outline is not available.
EJB
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Run As
Debug As
Profile As
Validate
Team
Compare With
Restore from Local History...
PDE Tools
Software AG Web Services Stack
Properties Alt+Enter
Link Source...
New Source Folder...
Use as Source Folder
Add External Archives...
Add Libraries...
Configure Build Path...
Javadoc
Declaration
varnings, 0 infos
Resource
Path
Location
EJB



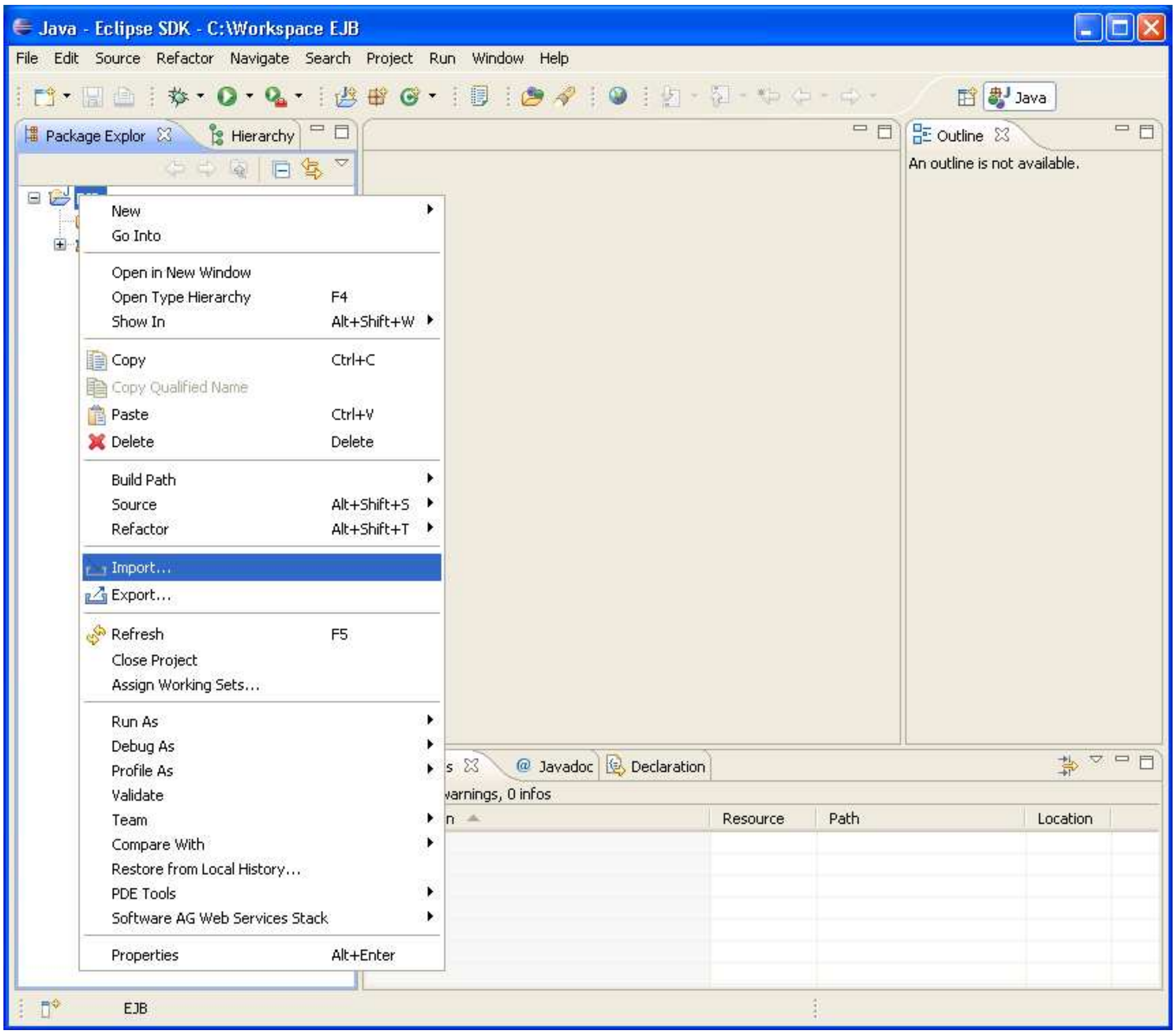
File name: entirex.jar
Files of type: *.jar;*.zip

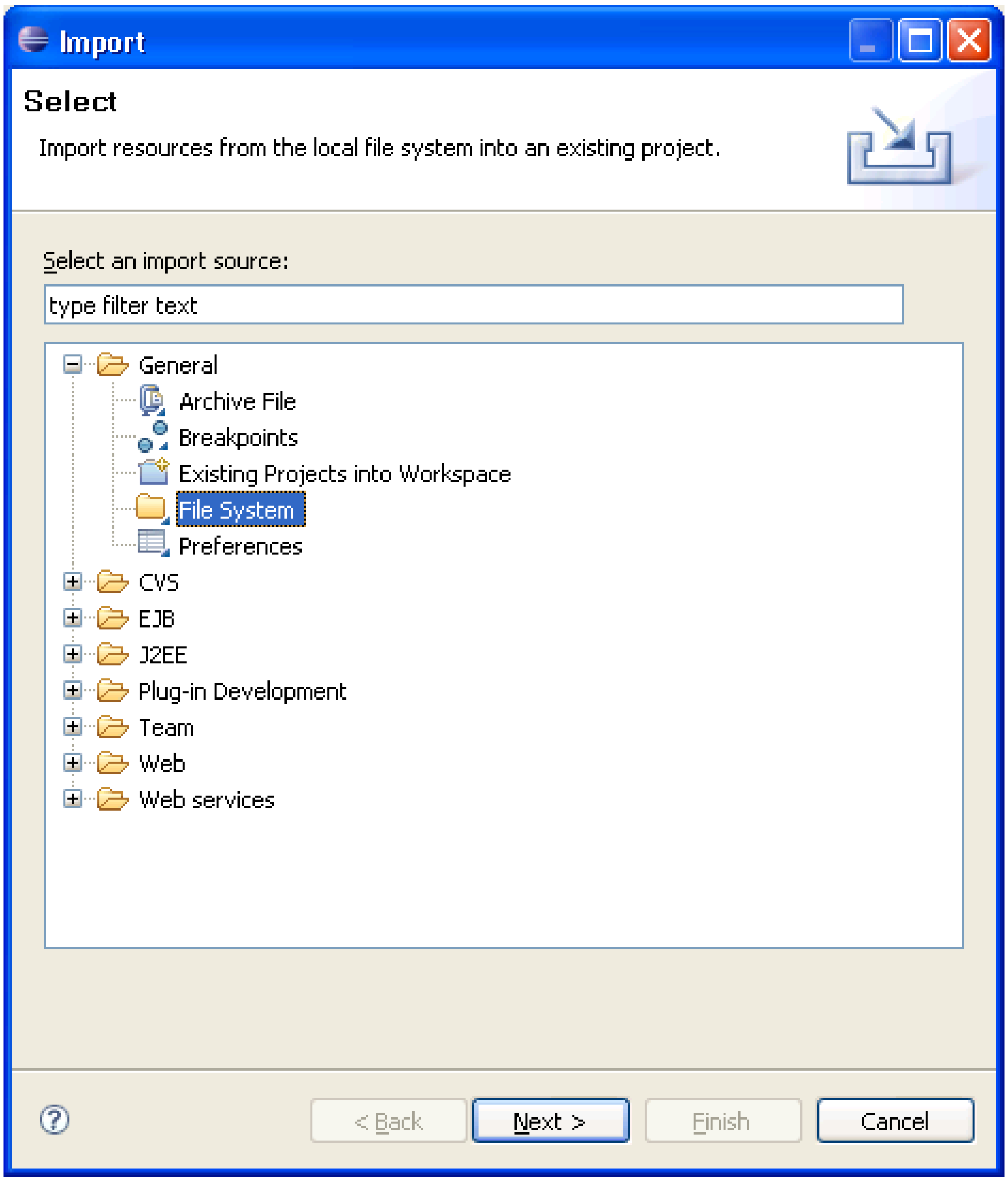
4. Add the Enterprise JavaBeans classes to the build path of the project (e.g.: *j2ee.jar*).

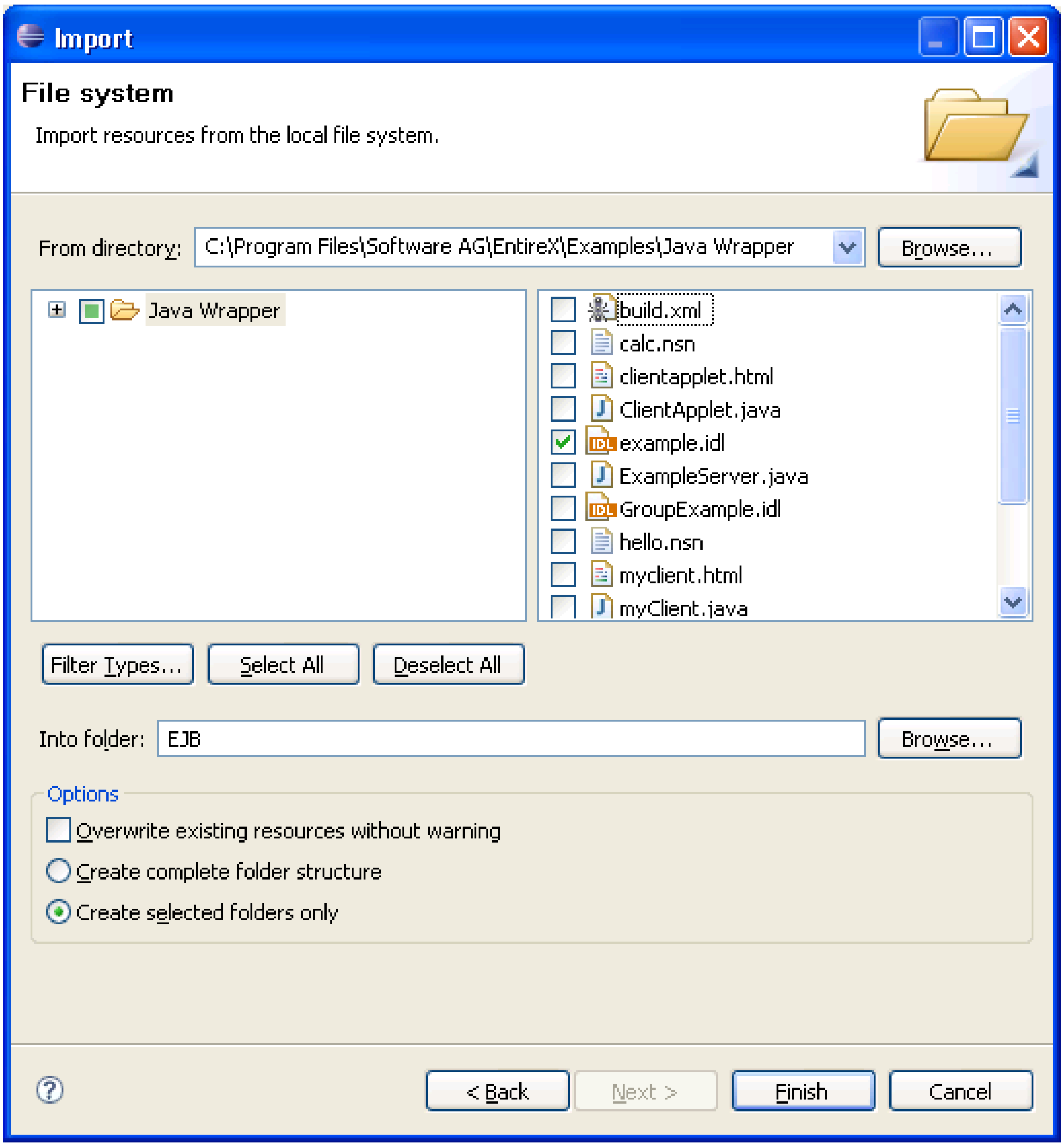

Java - Eclipse SDK - C:\Workspace EJB
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Explor
Hierarchy
Outline
An outline is not available.
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Run As
Debug As
Profile As
Validate
Team
Compare With
Restore from Local History...
PDE Tools
Software AG Web Services Stack
Properties Alt+Enter
Link Source...
New Source Folder...
Use as Source Folder
Add External Archives...
Add Libraries...
Configure Build Path...
Javadoc
Declaration
arnings, 0 infos
Resource
Path
Location
EJB



5. Import the IDL file into the project.


Java - Eclipse SDK - C:\Workspace EJB
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explor
Hierarchy
Outline
An outline is not available.
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Assign Working Sets...
Run As
Debug As
Profile As
Validate
Team
Compare With
Restore from Local History...
PDE Tools
Software AG Web Services Stack
Properties Alt+Enter
Javadoc
Declaration
warnings, 0 infos
Resource
Path
Location
EJB


Import
Select
Import resources from the local file system into an existing project.
Select an import source:
type filter text
General
Archive File
Breakpoints
Existing Projects into Workspace
File System
Preferences
CVS
EJB
J2EE
Plug-in Development
Team
Web
Web services
< Back
Next >
Finish
Cancel


Import

File system

Import resources from the local file system.

From directory: C:\Program Files\Software AG\EntireX\Examples\Java Wrapper

Java Wrapper

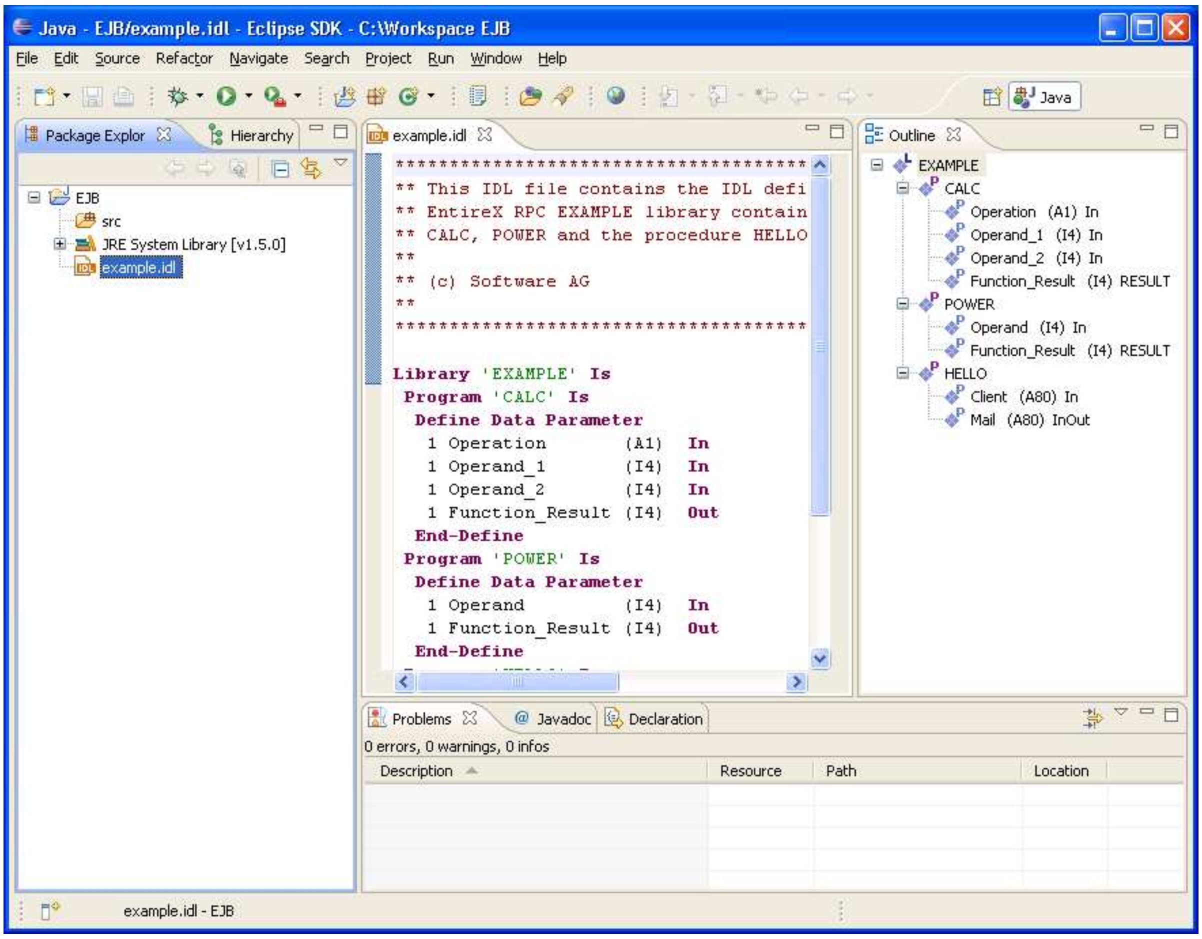
- [ ] build.xml
- [ ] calc.nsn
- [ ] clientapplet.html
- [ ] ClientApplet.java
- [✓] example.idl
- [ ] ExampleServer.java
- [ ] GroupExample.idl
- [ ] hello.nsn
- [ ] myclient.html
- [ ] myClient.java

Into folder: EJB

Options

- [ ] Overwrite existing resources without warning
- ( ) Create complete folder structure
- (●) Create selected folders only

6. Activate the *example.idl* file.


Java - EJB/example.idl - Eclipse SDK - C:\Workspace EJB
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package Explor
Hierarchy
EJB
src
JRE System Library [v1.5.0]
example.idl
Outline
Problems
Javadoc
Declaration
example.idl - EJB

```
****************************************
** This IDL file contains the IDL defi
** EntireX RPC EXAMPLE library contain
** CALC, POWER and the procedure HELLO
**
** (c) Software AG
**
****************************************


Library 'EXAMPLE' Is
 Program 'CALC' Is
  Define Data Parameter
   1 Operation        (A1)   In
   1 Operand_1        (I4)   In
   1 Operand_2        (I4)   In
   1 Function_Result  (I4)   Out
  End-Define
 Program 'POWER' Is
  Define Data Parameter
   1 Operand          (I4)   In
   1 Function_Result  (I4)   Out
  End-Define
```

```
EXAMPLE
  CALC
    Operation  (A1) In
    Operand_1  (I4) In
    Operand_2  (I4) In
    Function_Result  (I4) RESULT
  POWER
    Operand  (I4) In
    Function_Result  (I4) RESULT
  HELLO
    Client  (A80) In
    Mail  (A80) InOut
```

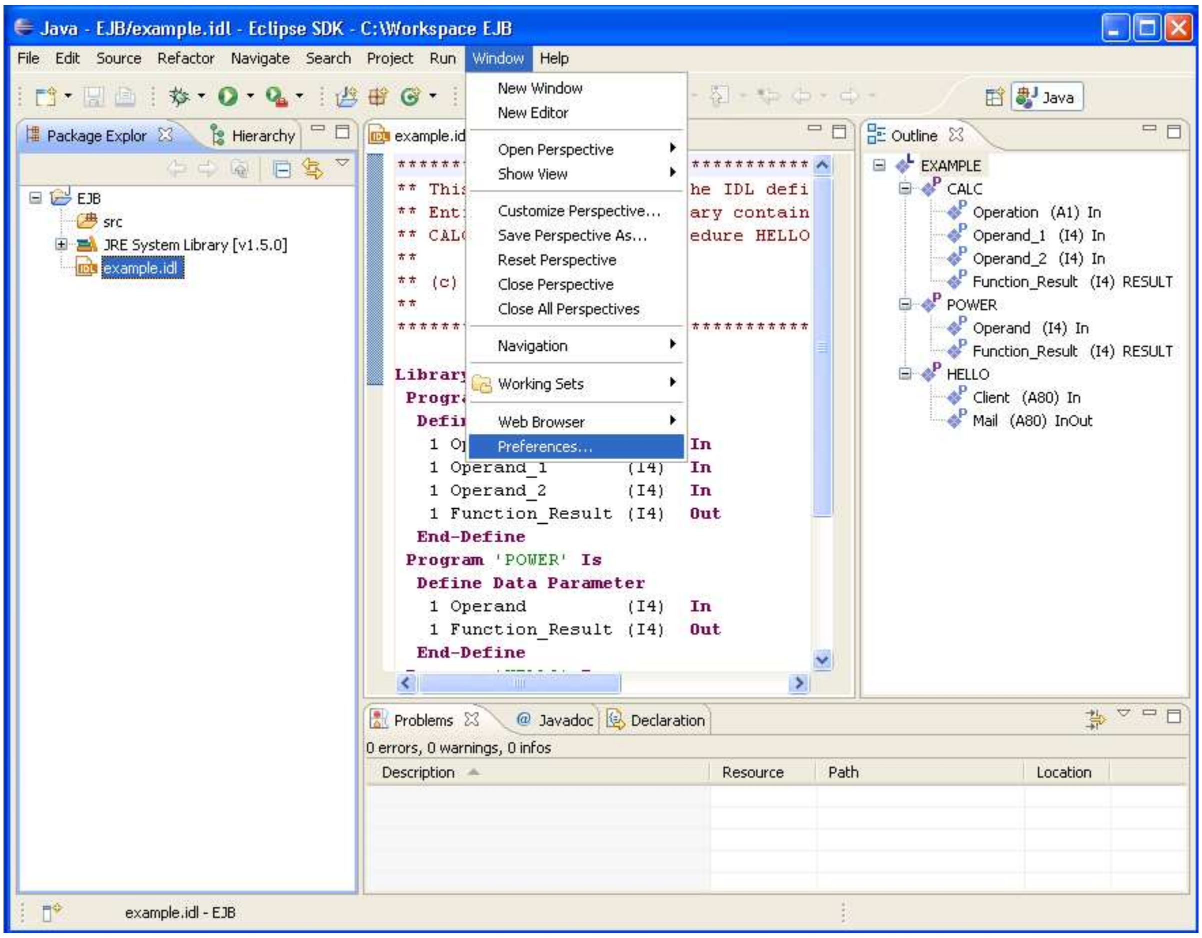
0 errors, 0 warnings, 0 infos

Description	Resource	Path	Location

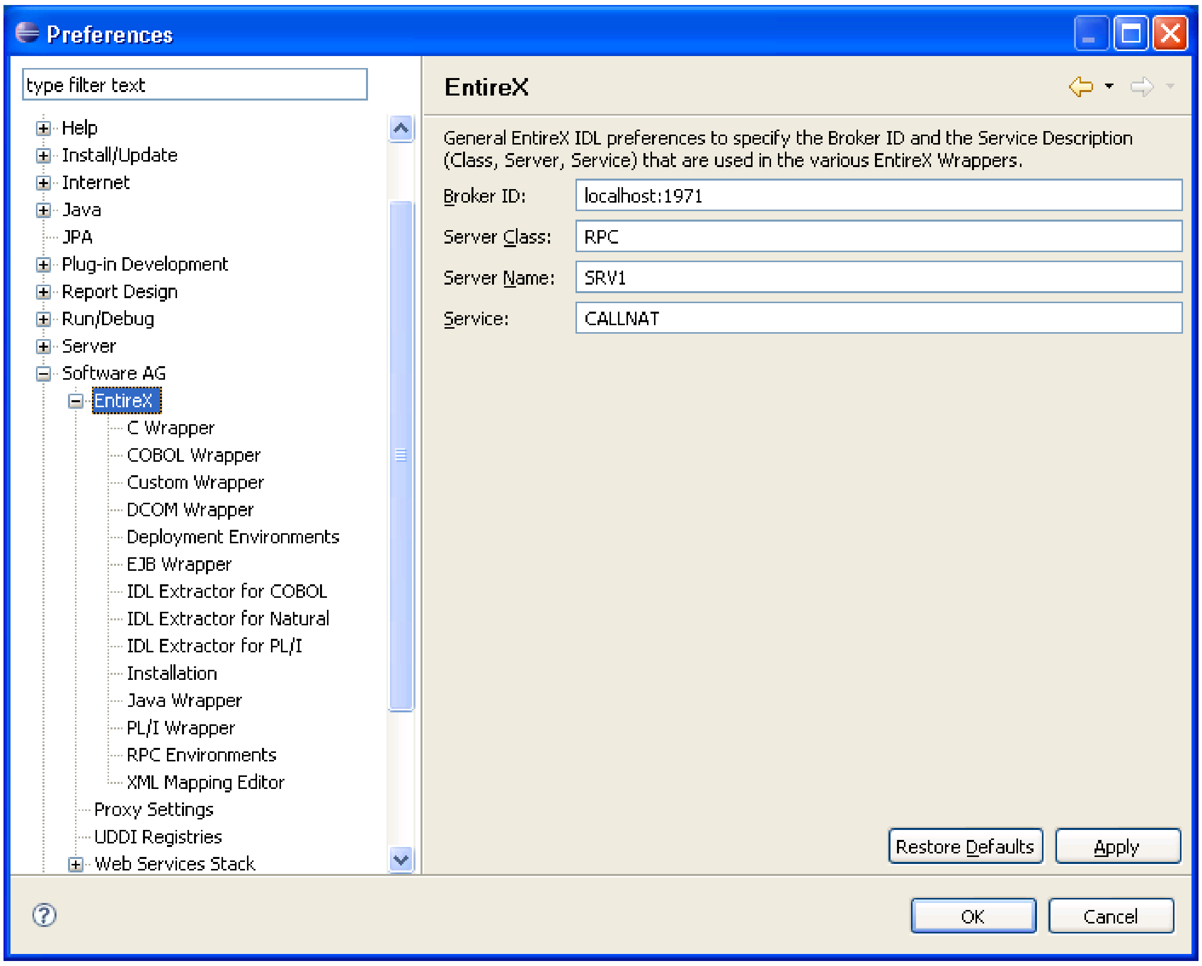
Setting/Modifying EntireX Enterprise JavaBeans Preferences

To set/modify the preferences

1. Display the Preferences window.



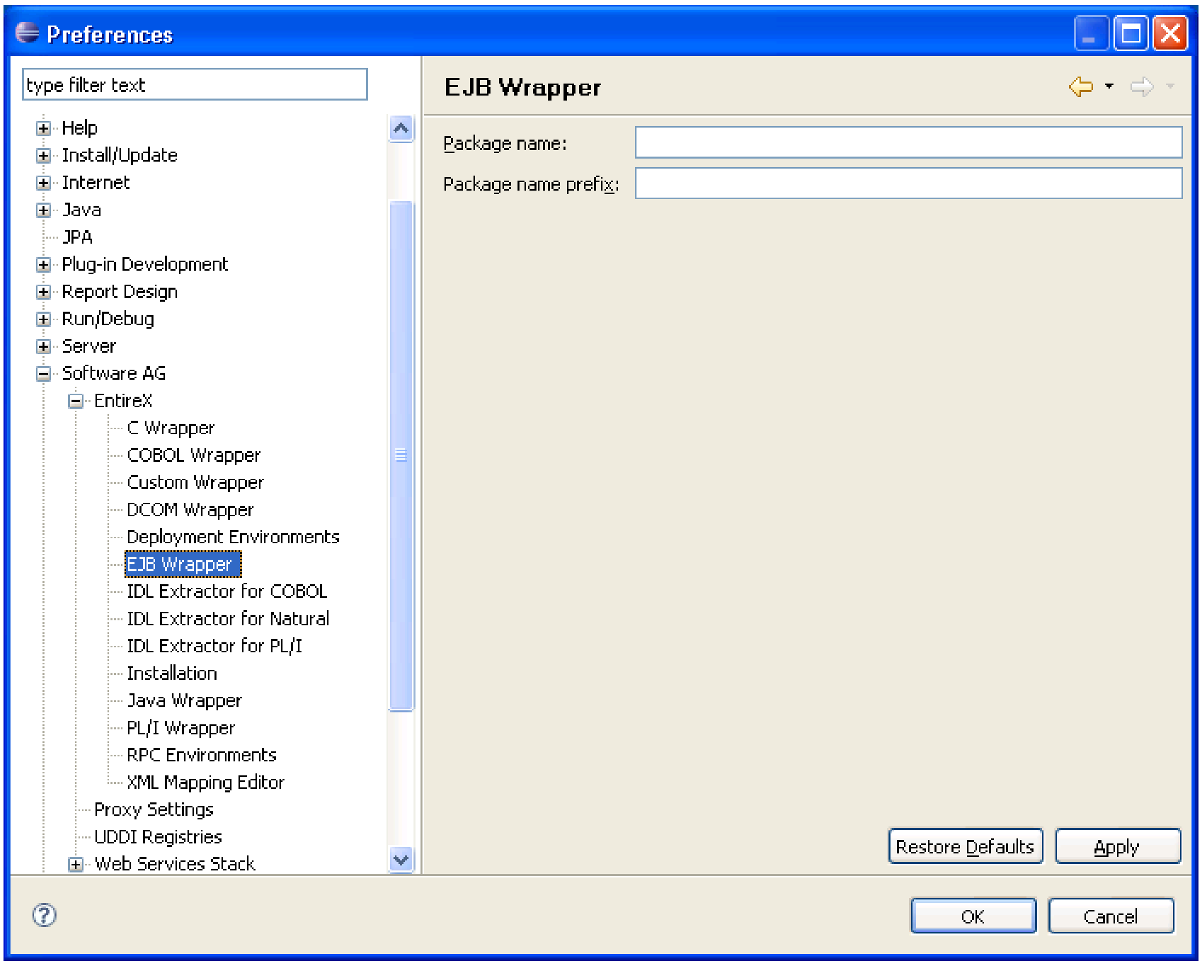
```
    1 Operand_2        (I4)  In
    1 Function_Result  (I4)  Out
  End-Define
 Program 'POWER' Is
  Define Data Parameter
    1 Operand          (I4)  In
    1 Function_Result  (I4)  Out
  End-Define
```



EntireX

General EntireX IDL preferences to specify the Broker ID and the Service Description (Class, Server, Service) that are used in the various EntireX Wrappers.

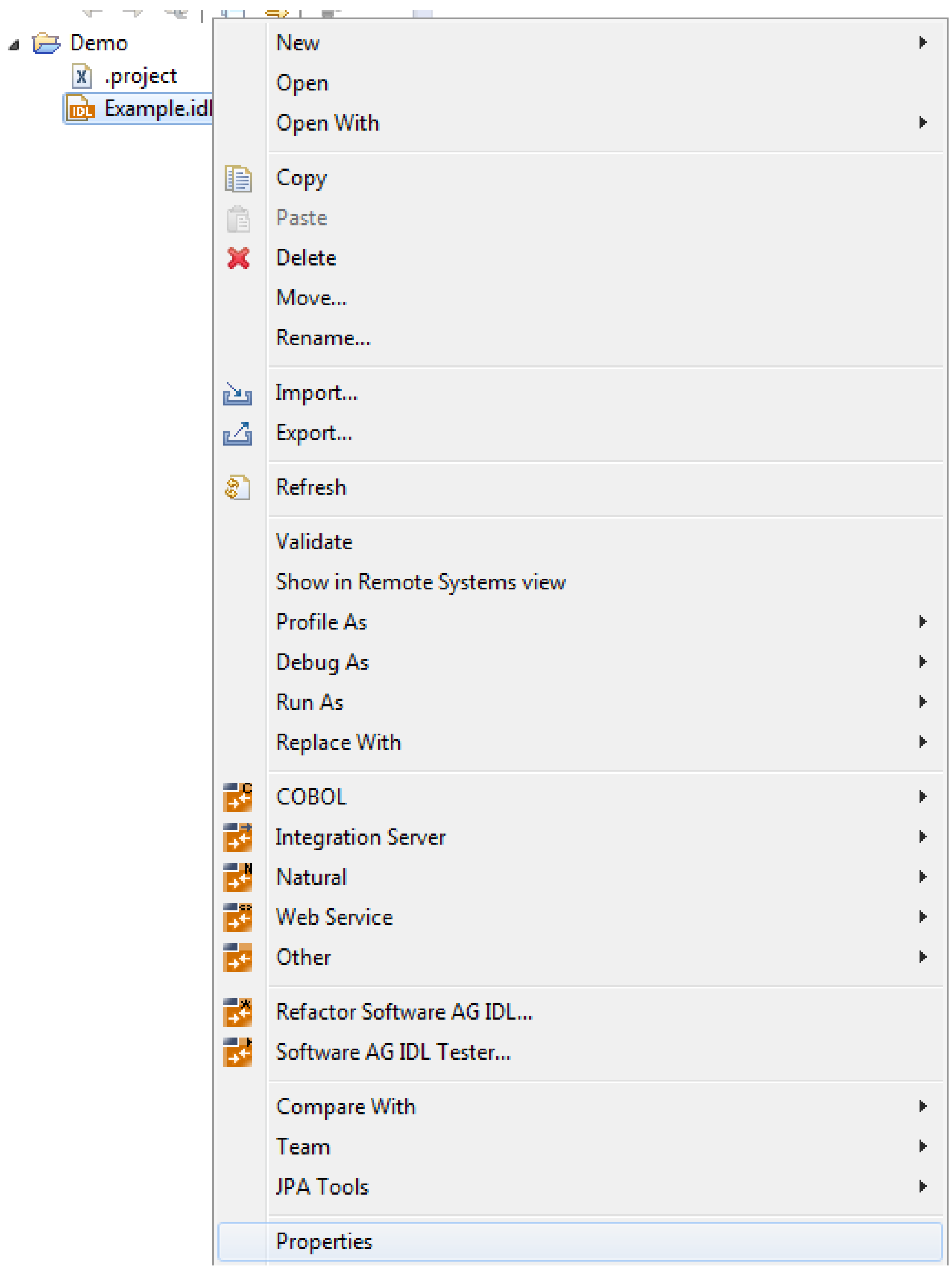
Broker ID:	localhost:1971
Server Class:	RPC
Server Name:	SRV1
Service:	CALLNAT



Setting/Modifying IDL File Properties

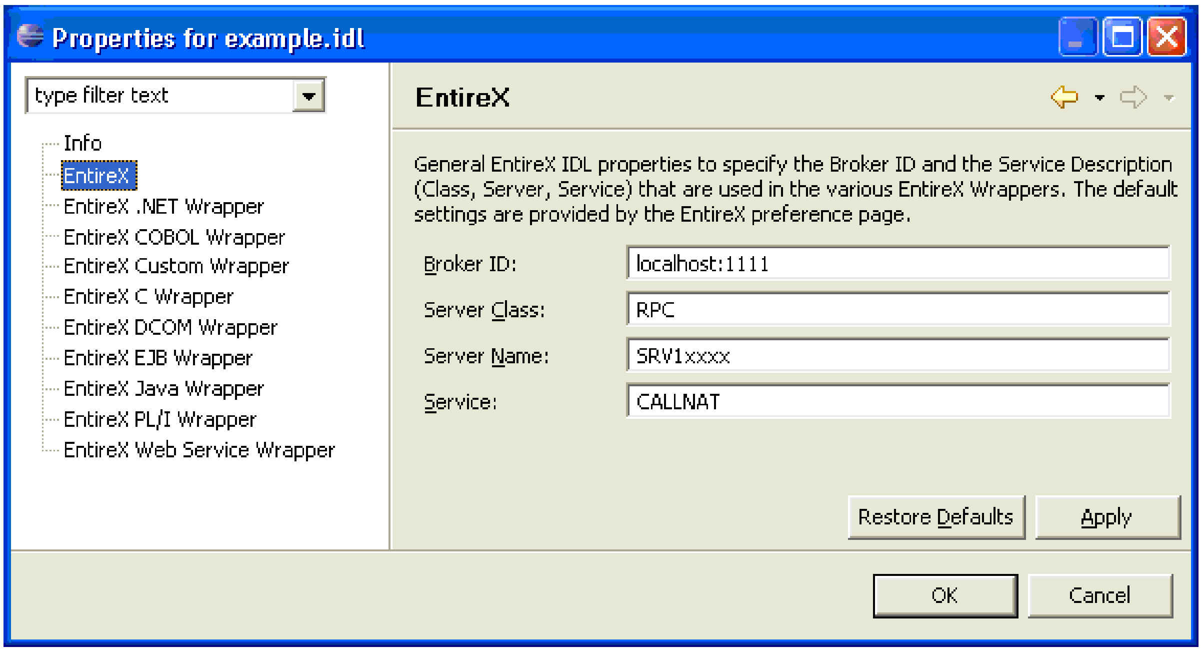
To set/modify the properties

1. Display the Properties window; use either the pop-up menu or, from the **File** menu, choose **Properties**.



2. In the **Properties** window, choose the **EntireX** tab.

   Here it is possible to modify the common default properties for the IDL file.

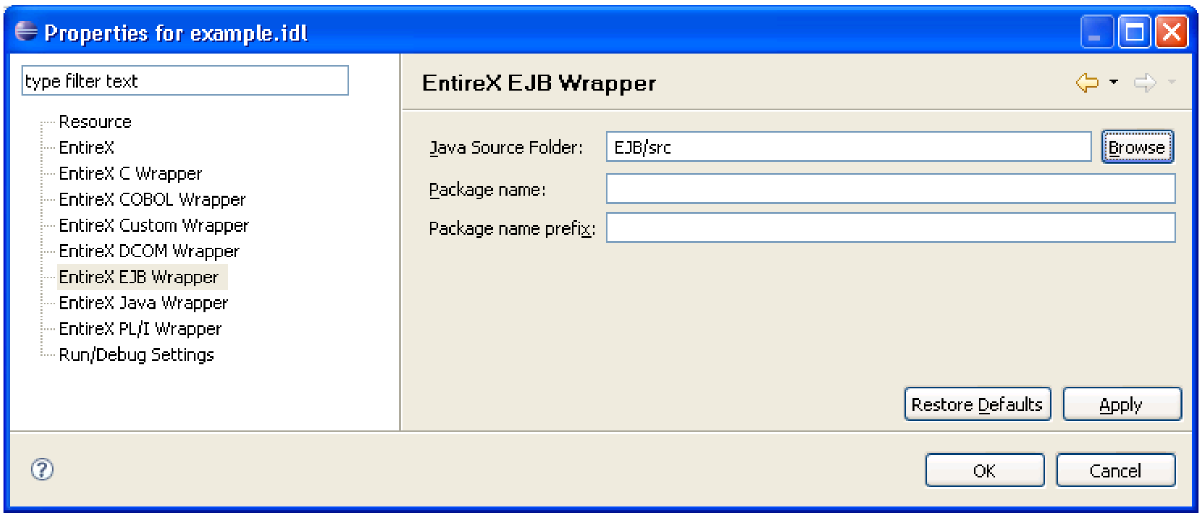


General EntireX IDL properties to specify the Broker ID and the Service Description (Class, Server, Service) that are used in the various EntireX Wrappers. The default settings are provided by the EntireX preference page.

Broker ID: localhost:1111
Server Class: RPC
Server Name: SRV1xxxx
Service: CALLNAT

3. Choose the **EntireX Wrapper for Enterprise JavaBeans** tab.

   Here it is possible to modify the defaults (see table *Default Properties for the IDL File*, below).

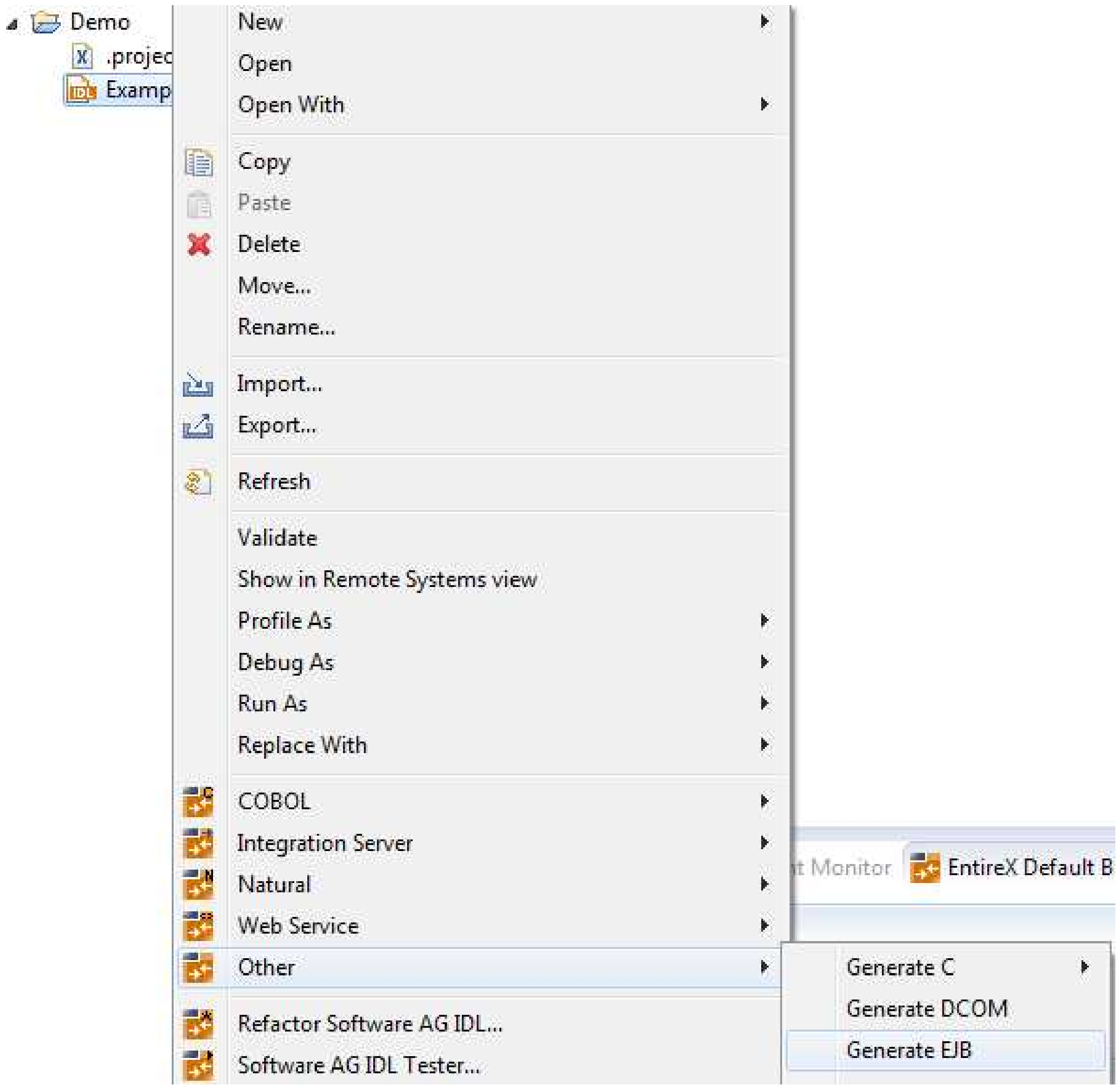


Java Source Folder: EJB/src
Package name:
Package name prefix:

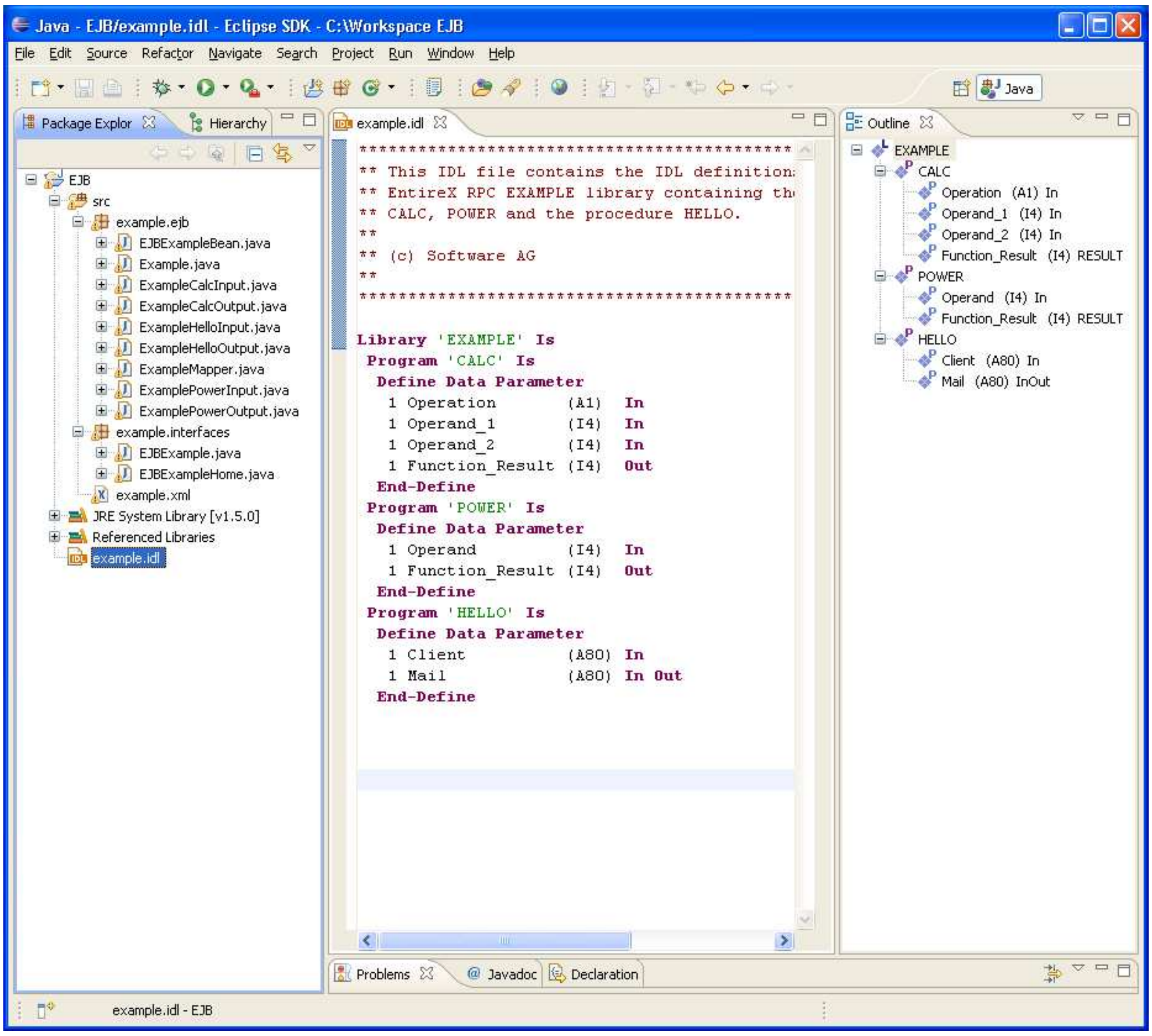
4. Confirm the entries with **OK**.

To generate the EJB sources

1. From the context menu, choose **Other > Generate EJB**.



2. All generated EJB sources are in the subdirectory EJB of the home directory of the IDL file.



```
*********************************************
** This IDL file contains the IDL definition
** EntireX RPC EXAMPLE library containing th
** CALC, POWER and the procedure HELLO.
**
** (c) Software AG
**
*********************************************

Library 'EXAMPLE' Is
 Program 'CALC' Is
  Define Data Parameter
   1 Operation       (A1)  In
   1 Operand_1       (I4)  In
   1 Operand_2       (I4)  In
   1 Function_Result (I4)  Out
  End-Define
 Program 'POWER' Is
  Define Data Parameter
   1 Operand         (I4)  In
   1 Function_Result (I4)  Out
  End-Define
 Program 'HELLO' Is
  Define Data Parameter
   1 Client          (A80) In
   1 Mail            (A80) In Out
  End-Define
```

Generated Classes and Interfaces

The workbench generates the following interfaces and classes from the Software AG IDL file sources.

Home Interface

The generated home interfaces *EJB<libname>Home.java* contain the create methods. These methods can be called by an EJB client.

create()

```
EJB<libname> create()
```

Calling this method creates an EJB with the default settings.

create(String user)

```
EJB<libname> create(String user)
```

where: String user is the Broker user name

Calling this method creates an EJB with the default settings except for the user name.

create(String user, String password)

```
EJB<libname> create(String user,String password)
```

where: String user is the Broker user name and String password the Broker user password

Calling this method creates an EJB with the default settings except for the user name and the password.

Remote Interface

The generated remote interfaces *EJB<libname>.java* contain the customer methods. For each RPC program a customer method is generated with the following naming conventions.

```
<libname><program>Output <program> (<libname><program>Input input)
```

where: Output is the output object and Input is the input object

These methods can be called by an EJB client.

EJB Class

The generated EJB class *EJB<libname>Bean.java* extends *<libname>Mapper.java* class. Bean implements the EJB session bean and controls the EntireX RPC communication.

Mapper Class

The generated *<libname>Mapper.java* class extends the Java RPC client stub *<libname>.java*. It implements a method for each RPC Program. Methodname is *<program>*. The method has one parameter, an instance of *<libname><progname>*Input.class.The return value of the method is an instance of *<libname><progname>*Output.class.

Group Classes

The group classes are serializable representations of the inner classes of the groups of the RPC library, filename is *<libname><progname><innerclassname>.java*. The class contains all parameters of the group.

Struct Classes

The struct classes are serializable representations of the inner classes of the structs of RPC library, filename is *<libname><innerclassname>.java*. The class contains all parameters of the struct.

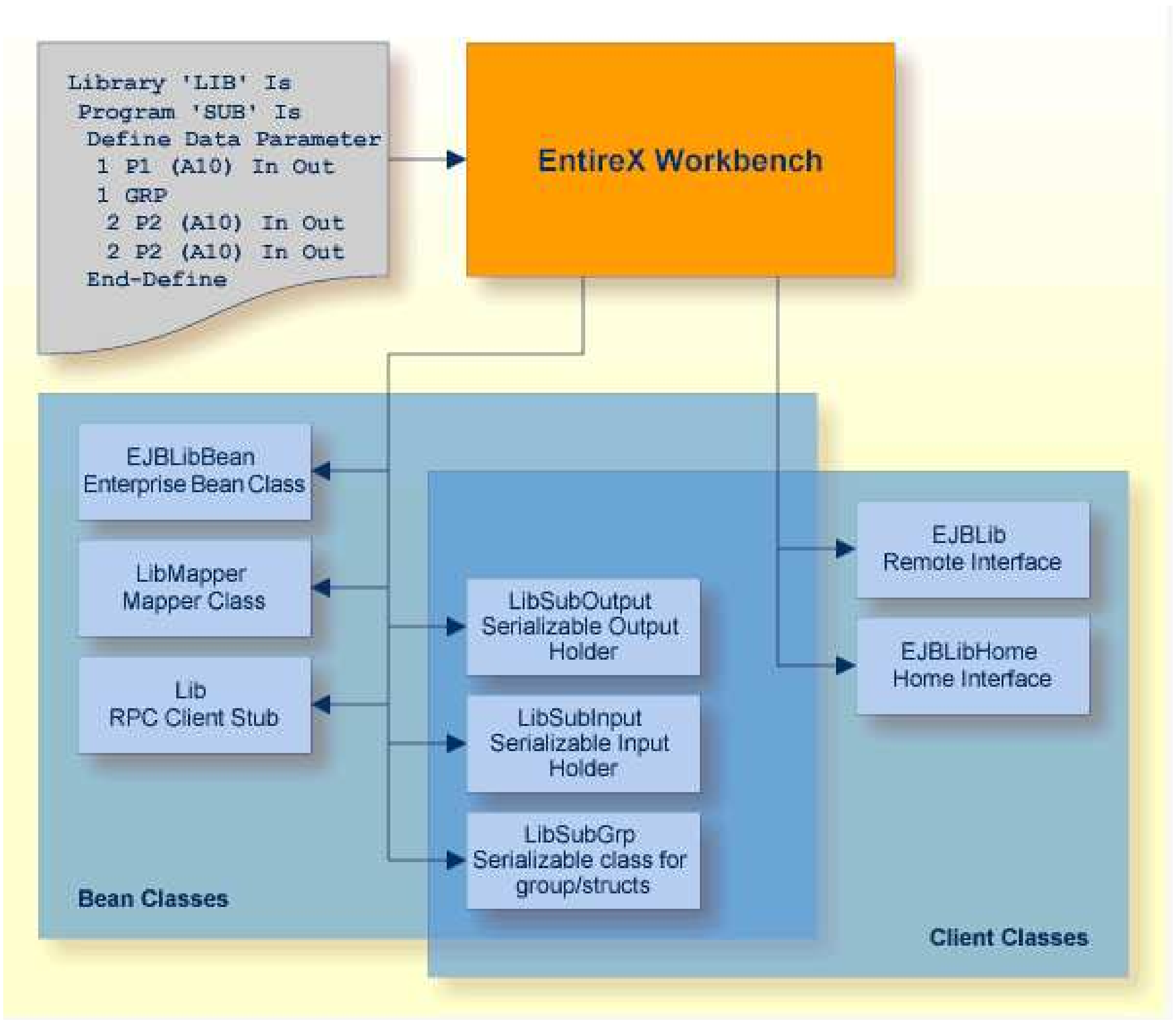
Input/Output Classes

These classes are serializable representations of all in, out, in/out parameters of RPC program. The filename of the input classes is *<libname><progname>Input.java*, it holds all in and in/out parameters of the program.The filename of the output classes is *<libname><progname>Output.java*, it holds all out and in/out parameters of the program.

Java RPC Client Stub

The generated Java RPC Stub *<libname>.java* extends com.softwareag.entirex.aci.RPCService class. It is the connecting link between the EJB and the EntireX server.

Example



```
Library 'LIB' Is
 Program 'SUB' Is
  Define Data Parameter
   1 P1 (A10) In Out
   1 GRP
    2 P2 (A10) In Out
    2 P2 (A10) In Out
  End-Define
```

EntireX Workbench
EJBLibBean Enterprise Bean Class
LibMapper Mapper Class
Lib RPC Client Stub
LibSubOutput Serializable Output Holder
LibSubInput Serializable Input Holder
LibSubGrp Serializable class for group/structs
EJBLib Remote Interface
EJBLibHome Home Interface
Bean Classes
Client Classes

Delivered Example

An example is delivered in directory *<drive>:\SoftwareAG\EntireX\examples\EJBWrapper* (Windows) or */opt/softwareag/EntireX\examples\EJBWrapper* (UNIX).

See the *README.TXT*.