Creating an Application Client in NetBeans IDE

PRINTABLE VERSION

This tutorial shows you how to write and deploy a simple application client that is used to access an Enterprise JavaBean (EJB). The tutorial shows you how to access EJBs built on the Java EE 5 platform (EJB 3.0) and the J2EE 1.4 platform (EJB 2.x).

The IDE enables you to create an application client as an independent project type. You can create an application client project as part of an enterprise application or as an independent standalone application.

This document uses the **NetBeans IDE 6.8** release and **GlassFish v2**. The steps outlined in this document can also be applied if you are using any 6.x version of the IDE.

Expected duration: 30 minutes

Tutorial Exercises

- Creating an Application Client as Part of an Enterprise Application
- Using a Standalone Application Client to Access an EJB
  - Accessing a Single Remote Interface
  - Accessing Two or More Remote Interfaces
- Using a Standalone Application Client to Access an EJB (EJB 2.1)
- Downloading the Solution Project

To follow this tutorial, you need the following software and resources.

Software or Resource	Version Required
NetBeans IDE	6.x, Java version
Java Development Kit (JDK)	version 6 or version 5
GlassFish application server	V2

For this tutorial you need to register a local instance of GlassFish v2 / Sun Java System Application Server with the IDE.

Prerequisites

This document assumes you have some basic knowledge of, or programming experience with, the following technologies:

- Java Programming
- NetBeans IDE

You can download a zip archive of the finished project.

Creating an Application Client as Part of an Enterprise Application

The first thing to do is create a Java EE 5 enterprise application named EnterpriseAppEE5. You will create the application client at the same time as you create the enterprise application.

Creating the Enterprise Application

1. Choose File > New Project from the main menu.
2. Select Enterprise Application from the Java EE category and click Next.
3. Type **EnterpriseAppEE5** for the project name and specify a project location.
4. Deselect the Use Dedicated Folder option, if selected.
   (For this tutorial there is little reason to copy project libraries to a dedicated folder because you will not need to share libraries with other users or projects.)
   Click Next.
5. Set the server to GlassFish v2 and set the Java EE Version to Java EE 5.
6. Select Create EJB Module and Create Application Client Module, if unselected.
   You can deselect the Create Web Application Module because you do not need a web module for this tutorial.
7. Click Finish.

Creating the Session Bean in the Enterprise Application

You now create the `SessionBean` EJB bean in the EJB module. To create the `SessionBean`, do the following:

1. Right-click the EnterpriseAppEE5-ejb EJB module in the Project window and choose New > Other to open the New File wizard.
2. From the Java EE category, select Session Bean and click Next.
3. Type **Session** for the EJB name and type **ejb** for the package.
4. Select Stateless as the Session Type and select Remote for the type of interface to create.
   (You can deselect local interface because you will only use the remote interface in this tutorial.)
   Click Finish.
   When you click Finish, the bean class `SessionBean.java` opens in the Source Editor.
5. In the source editor, right-click in the class body and choose Insert Code (Ctrl-I) > Add Business Method.
6. In the Add Business Method dialog box, type **getResult** for the name, set the Return Type to String, and select the Remote interface box. Click OK.

7. In `SessionBean.java`, modify the `getResult` method to the following:

```
public String getResult() {
    return "This is EJB 3.0 Bean";
}
```

8. Save your changes.

Calling the Session Bean From the Application Client

Now you add the code to the application client that is used to call the EJB bean. To modify the application client, do the following:

1. Expand the EnterpriseAppEE5-app-client > Source Packages > enterpriseappee5 in the Projects window and double-click `Main.java` to open the file in the Source Editor.
2. In the source editor, right-click in the class body and choose Insert Code (Ctrl-I) > Call Enterprise Bean.
3. In the Call Enterprise Bean dialog box, expand the EnterpriseAppEE5-ejb node and select `SessionBean` as the bean you want to call and select Remote as the Referenced Interface. Click OK.
   When you click OK, the following annotation is added to `Main.java` to call the session bean.

```
@EJB
private static SessionRemote sessionBean;
```

4. Modify the `main` method with the following code to generate some simple output. You will use `System.err.println` so that you can easily see the message in the Output window.

```
public static void main(String[] args) {
    System.err.println("result=" + sessionBean.getResult());
```

5. Save your changes.

Configuring the Enterprise Application

Almost everything is done now, and the last thing you need to do is to set up the application client as a module, which should be run when you run the Enterprise Application. This is probably already configured because you created the application client and the enterprise application at the same time.

To configure the enterprise application, do the following:

1. Right-click the EnterpriseAppEE5 enterprise application project node in the Projects window and choose Properties.
2. Select Run in the Project Properties dialog box.
3. Choose EnterpriseAppEE5-app-client for the Client Module URI (which should be selected by default). Click OK.

Running the Enterprise Application

You can now run the enterprise application to test the application client.

1. Right-click EnterpriseAppEE5 in the Projects window and choose Run.

When you run the project, the IDE builds and deploys the application. The following message appears in the Output window:

result = This is EJB 3.0 Bean

Using a Standalone Application Client to Access an EJB

In this exercise you create an EJB module project and then an application client project that connects to the EJB.

In this exercise you cannot use EJB annotations to link the EJB to the application client so you will use a lookup from a context. Because the code for bean lookup is different if you have more than one remote interface, this exercise is divided into two parts:

- Accessing a Single Remote Interface
- Accessing Two or More Remote Interfaces

Accessing a Single Remote Interface

In this exercise you create an EJB module with a remote interface and then create an application client that accesses the EJB.

Creating the EJB module

In this exercise you create an EJB module named EJBModule30. In this exercise you do not want to add the EJB module to an enterprise project.

1. Choose File > New Project from the main menu.
2. Select EJB Module from the Enterprise category and click Next.
3. Type **EJBModule30** for the project name and set location of the project.
4. Deselect the Use Dedicated Folder option. Click Next.
5. Ensure that the EJB module is not added to an enterprise application.
6. Set the server to GlassFish and set the Java EE Version to Java EE 5.
7. Click Finish.

Creating a Session Bean in the EJB Module

In this exercise you will create the Bean30 session bean in your EJB module. To create Bean30, do the following:

1. Right-click the EJB module in the Projects window and choose New > Other to open the New File wizard.
2. From the Enterprise category, select Session Bean and click Next.
3. Type **Bean30** for the EJB name, type **ejb** for the package, leave the Session Type as Stateless and select Remote to create a remote interface. Click Finish.
   When you click Finish, the bean class `Bean30Bean.java` opens in the Source Editor.
4. In the Source Editor, right-click in the source code and choose EJB Methods > Add Business Method from the popup window to generate a business method for the bean. In the Add Business Method dialog box, type **getResult** for the name of the method, set the Return Type to String and ensure that the Remote interface box is selected. Click OK.
5. In `Bean30Bean.java`, modify the return of the `getResult` method with the following (in bold):

```
public String getResult() {
    return "This is EJB 3.0 Bean";
}
```

6. Modify the `@Stateless` session bean annotation to change the default name used to map the bean from `Bean30Bean` to `Bean30`. You will then use `Bean30` in the application client to look up the bean. The annotation should now look like the following (changes in bold):

```
@Stateless(mappedName="Bean30")
public class Bean30Bean implements Bean30Remote {
```

7. Save your changes.

Creating the Application Client Project

You now create the application client ApplicationClientForTest that will access the EJB.

1. Choose File > New Project from the main menu.
2. Select Enterprise Application Client from the Enterprise category and click Next.
3. Type **ApplicationClientForTest** for the project name and set the location.
4. Deselect the Use Dedicated Folder option. Click Next.
5. Set the server to GlassFish and set the Java EE Version to Java EE 5.
6. Click Finish.

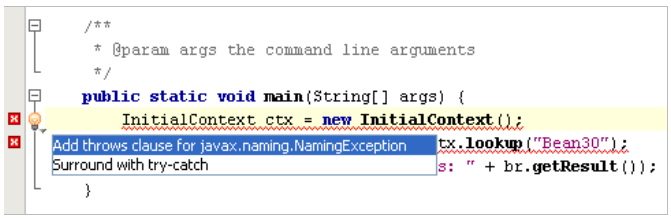
Calling the EJB from the Application Client

You now add the EJBModule30 project to ApplicationClientForTest as a library and add the code to access `Bean30`.

1. Expand the ApplicationClientForTest project node in the Projects window, right-click the Libraries node and select Add Project.
   Alternatively, you can open the ApplicationClientForTest project properties window and click Add Project in the Libraries category.
2. In the Add Project dialog box, locate and select the EJBModule30 project and click Add Project JAR Files.
   The JAR file for EJBModule30 is added to the library for ApplicationClientForTest. You can see the project libraries by expanding the Libraries node for the project in the Projects window.
3. Double-click `Main.java` in the Projects window to open the file in the Source Editor.
4. In the Source Editor, add the following code to the `main` method of `Main.java`. Again, you will use `System.err.println` so that you can easily see the message in the Output window.

```
public static void main(String[] args) {
    InitialContext ctx = new InitialContext();
    Bean30Remote br = (Bean30Remote) ctx.lookup("Bean30");
    System.err.println("EJB message is:" + br.getResult());
```

5. Fix the import statements (Ctrl-Shift-I).
6. Fix any errors in the code. You can use the suggestion feature in the IDE to help you.
   Note: In this case, the error is caused by a missing throws clause. To use the suggestion feature of the IDE, place the insert cursor in the first line of the code containing the error, underlined in red. When the suggestion bulb appears in the left margin, click the bulb and choose "Add throws clause..." to add a throws clause to the method. The IDE adds the appropriate import statements when it adds the throws clause to the method.



7. Save your changes.

Now everything should be fine, so you can deploy EJBModule30 and ApplicationClientForTest to the server by right-clicking the project node for each of the projects in the Projects window and choosing Undeploy and Deploy. After you deploy both of the projects, run the application client by right-clicking ApplicationClientForTest in the Projects window and choosing Run. In the Output window you should see the following:
EJB message is: This is EJB 3.0 Bean

Note: If you are using a firewall, you may need to disable the firewall to allow the application client to access the EJB.

Accessing Two or More Remote Interfaces

In this exercise you create a second remote interface in EJBModule30. You then modify the code in the application client so that it accesses each of the interfaces. The code used in this exercise to call the EJBs is slightly different than the code for calling a single remote interface. This exercise will help you when you want to access two or more remote interfaces from a single standalone application client.

Creating a Second Remote Interface for Bean30 EJB

In this exercise you create a second remote interface named Bean30Remote2 for Bean30 EJB.

To create the second interface, do the following:

1. Right-click the EJB module in the Projects window and choose New > Java class.
2. Type **Bean30Remote2** for the class name, type **ejb** for the package and click Finish.
   When you click Finish, the bean class `Bean30Remote2.java` opens in the Source Editor.
3. Add the `@Remote` annotation to the class to declare the class as a remote interface.
4. Modify the definition of the class to declare the class an interface and add a `getResult` method. The class should now look like the following (changes in bold):

```
@Remote
public interface Bean30Remote2 {
    String getResult2();
}
```

5. Fix the import statements. You will need to import `javax.ejb.Remote`.
6. Save your changes.

Modify Bean30 Bean to Implement the Interface

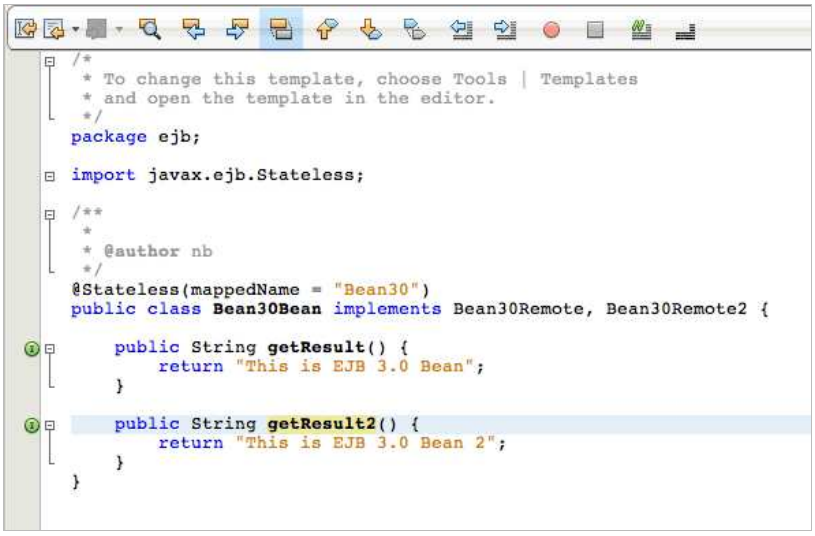
In this exercise you modify Bean30 to implement the interface Bean30Remote2.

1. In `Bean30Bean.java` implement the `getResult2` method from `Bean30Remote2.java`. The method should look like the following:

```
public String getResult2() {
    return "This is EJB 3.0 Bean 2";
}
```

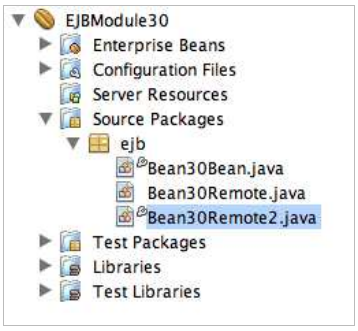
2. Modify the class definition to implement `Bean30Remote2`.

The class should now look like the following:



```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package ejb;

import javax.ejb.Stateless;

/**
 *
 * @author nb
 */
@Stateless(mappedName = "Bean30")
public class Bean30Bean implements Bean30Remote, Bean30Remote2 {

    public String getResult() {
        return "This is EJB 3.0 Bean";
    }

    public String getResult2() {
        return "This is EJB 3.0 Bean 2";
    }
}
```

And the directory structure like this:



Modifying ApplicationClientForTest

Next we modify the lookup code in the `Main` class of the application client.

1. Double-click the `main.java` class of ApplicationClientForTest in the Projects window to open the class in the Source Editor.
2. Modify the code in the `main` method to look up the second interface. In this case, when there is more than one remote interface, we have to use the full name for each of the interfaces in the lookup.

```
public static void main(String[] args) throws NamingException {
    InitialContext ctx = new InitialContext();
    Bean30Remote br = (Bean30Remote) ctx.lookup("Bean30#ejb.Bean30Remote");
    System.err.println("EJB message is:" + br.getResult());
    Bean30Remote2 br2 = (Bean30Remote2) ctx.lookup("Bean30#ejb.Bean30Remote2");
    System.err.println("EJB message 2 is:" + br2.getResult2());
```

3. Fix the imports to add an import statement for `ejb.Bean30Remote2`.
4. Save your changes.

Now redeploy EJBModule30, and run the application client. In the Output window you will see the following output:
EJB message is: This is EJB 3.0 Bean
EJB message 2 is: This is EJB 3.0 Bean 2

Summary

In this exercise we showed you how to use an application client to access EJB 3.0 Enterprise JavaBeans. We showed how to access the EJB from an application that was in an enterprise application and also from a standalone application client. We also demonstrated the difference between calling a bean with a single remote interface and beans with more than one remote interface.

Using a Standalone Application Client to Access an EJB (EJB 2.1)

If you want to work with EJB 2.X beans, the process is very similar, but not exactly the same. This exercise shows how accessing EJB 2.x beans differs from accessing EJB 3.0 beans.

Creating an EJB module EJBModule14

First create the EJB module.

1. Choose File > New Project from the main menu.
2. Select EJB Module from the Enterprise category and click Next.
3. Type **EJBModule14** for the project name and set the location of the project.
4. Deselect the Use Dedicated Folder option. Click Next.
5. Ensure that the EJB module is not added to an enterprise application.
6. Set the server to GlassFish and set the Java EE Version to J2EE 1.4.
7. Click Finish.

Creating the Session Bean in EJBModule14

You now create the `Bean14` EJB bean in the EJB module. To create the `Bean14`, do the following:

1. Right-click the EJB module in the Projects window and choose New > Other to open the New File wizard.
2. Select Session Bean from the Enterprise category and click Next.
3. Type `Bean14` for the class name, type `ejb` for the package, leave the Session Type as `Stateless` and select `Remote` to create a remote interface. Click Finish.
   When you click Finish, the bean class `Bean14Bean.java` opens in the Source Editor.
4. In the Source Editor, right-click in the source code and choose EJB Methods > Add Business Method from the popup menu to open the Add Business Method dialog box.
5. In the Add Business Method dialog box, type `getResult` for the name of the business method, set the Return Type to String, and select Remote for the interface. Click OK.
6. Modify the `getResult` method in `Bean14Bean.java` to have a return method to access the bean from other component:

```
public String getResult() {
    return "This is EJB 1.4 Bean";
}
```

7. Save your changes.

Calling the EJB from the Application Client

You now add the EJBModule14 project to ApplicationClientForTest as a library and add the code to access Bean14.

1. Expand the ApplicationClientForTest project node in the Projects window, right-click the Libraries node and select Add Project.
2. In the Add Project dialog box, locate and select the EJBModule14 project and click Add Project JAR Files.
   The JAR file for EJBModule14 is added to the library for ApplicationClientForTest. You can see the project libraries by expanding the Libraries node for the project in the Projects window.
3. Double-click `Main.java` in the Projects window to open the file in the Source Editor.
4. In the Source Editor, remove or comment out the code used to access EJBModule30 and add the following code (in bold) to the `main` method of `Main.java`:

```
public static void main(String[] args) throws NamingException {
    InitialContext ctx = new InitialContext();
    Object remote = ctx.lookup("ejb/Bean14Bean");
    Bean14RemoteHome sbrh = (Bean14RemoteHome) PortableRemoteObject.narrow(remote, Bean14RemoteHome
    Bean14Remote sbr = sbrh.create();
    System.err.println("EJB 1.4 result:" + sbr.getResult());
```

5. Fix the import statements.
6. Fix the errors in the source code. There are errors because the method needs a throws clause for `NamingException`, `CreateException` and `RemoteException` from the lookup. You can use the IDE suggestion feature to help you resolve the errors.

Now everything should be fine, so deploy EJBModule14 and ApplicationClientForTest to the server (right-click each project and choose Undeploy and Deploy) and then right-click ApplicationClientForTest and choose Run. When you run the client we should see the following in the Output window:
EJB 1.4 result: This is EJB 1.4 Bean

Downloading the Solution Project

You can download the solution to this tutorial as a project in the following ways.

- Download a zip archive of the finished project.
- Checkout the project sources from Kenai by performing the following steps:
  1. Choose Team > Kenai > Get Sources from Kenai from the main menu.
  2. In the Get Sources from Kenai dialog box, locate the Kenai Repository by clicking Browse to open the Browse Kenai Projects dialog box.
  3. Search for the **NetBeans IDE Samples Catalog**.
  4. Select the NetBeans IDE Samples Catalog and click OK.
  5. Click Browse to specify the Folder to Get and select **Samples/JavaEE/entappclient**. Click OK.
  6. Specify the Local Folder for the sources (the local folder must be empty).
  7. Click Get From Kenai.
     When you click Get From Kenai, the IDE initializes the local folder as a Subversion repository and checks out the project sources.
  8. Click Open Project in the dialog that appears when checkout is complete.

  Notes.

  - Steps for checking out sources from Kenai only apply to NetBeans IDE 6.7 and 6.8.
  - You need a Subversion client to checkout the sources from Kenai. For more about installing Subversion, see the section on Setting up Subversion in the Guide to Subversion in NetBeans IDE.

Send Us Your Feedback

Next Steps

For more information about using NetBeans IDE to develop Java EE applications, see the following resources:

- Introduction to Java EE Technology
- Getting Started with Java EE 6 Applications
- Java EE & Java Web Learning Trail

You can find more information about using EJB Enterprise Beans in the Java EE 6 Tutorial.

To send comments and suggestions, get support, and keep informed on the latest developments on the NetBeans IDE Java EE development features, join the nbj2ee mailing list.