

Building Jaguar CTS components in PowerBuilder

What this chapter covers:

- **The PowerBuilder component model**
- **Building a Jaguar component in PowerBuilder**
- **Deploying a component into Jaguar from PowerBuilder**

At this point we have covered a lot of material on the Jaguar CTS application server. The introduction of new Jaguar concepts and the basic administration of the server are all important for laying the groundwork for understanding how to build a distributed application. In the next two chapters, we will build a Hello World application to demonstrate how a basic PowerBuilder/Jaguar application is developed. Using this simple example, we will also explore how to use the wizards to speed development.

6.1 *PowerBuilder as a component model*

Jaguar supports several different component models, including PowerBuilder. The PowerBuilder component model is based on the custom-class user object, more commonly known as the nonvisual user object (NVO). The custom-class object is used to build processing objects that have no visual properties. The NVO is the building block of a distributed application in which all the application functionality and data access is written. It has functions and events, collectively known as methods and variables which are also known as attributes. The NVO represents an object class and is used to group related processes and encapsulate business and systems functions, promoting true-code reusability.

The custom-class user object is a very simple object derived from the PowerBuilder NonVisualObject system object. It has no visual properties and contains only two events: the constructor and the destructor. It also supports the most basic set of PowerBuilder functions:

- `ClassName`
- `GetContextService`
- `GetParent`
- `PostEvent`
- `TriggerEvent`
- `TypeOf`

6.1.1 *Building a custom-user object*

The nonvisual object class is created in the User Object painter. Only a custom-class user object can be deployed as a Jaguar component. To create a new NVO, choose File | New from the main menu or click on the New toolbar option. The New Wizard dialog box will open, as shown in figure 6.1. Select the Object tab and click on the Custom Class icon.

This will launch the new User Object painter as illustrated in figure 6.2. As you can see, the development environment has undergone a radical change in PowerBuilder 7.

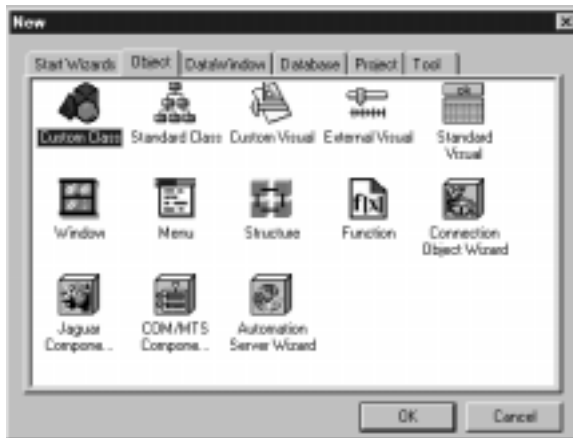


Figure 6.1
New User Object wizard

As you step through the development of a user object, you'll see some of the new features highlighted.

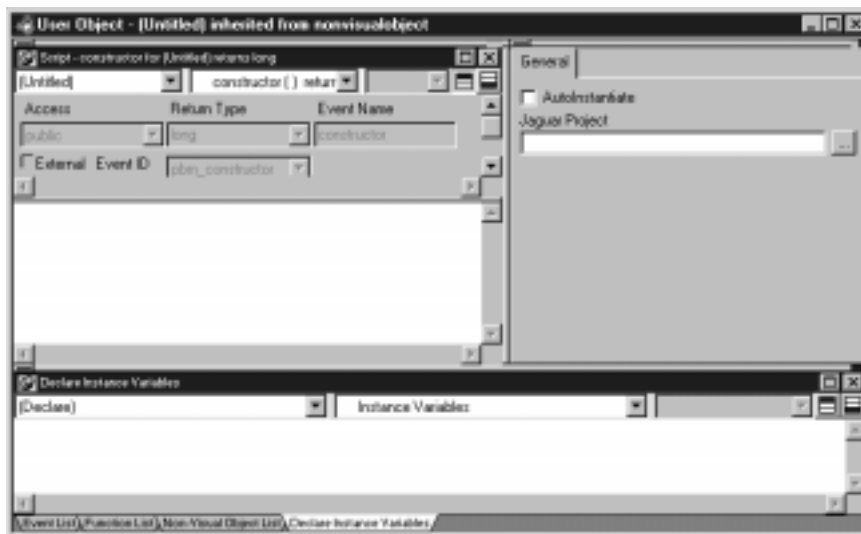


Figure 6.2 **User Object painter**

The custom-class object can be deployed as a Jaguar component as long as none of the guidelines established by Jaguar and CORBA 2.2 IDL are broken. To insure that the object will be compatible with the Jaguar server, check the Jaguar Validation option on the main menu under the Design menu item. When Jaguar validation is turned on, the object will be inspected every time it is saved. Any code that the Jaguar server does not support will be flagged. The restrictions Jaguar places on object development are covered in chapter 9, which provides more details on Jaguar component development.

To build a Jaguar component in PowerBuilder, follow these steps:

- 1 Create a new custom-class user object or inherit from an existing one.
- 2 Add the appropriate methods and variables to the object.
- 3 Set up the Jaguar component project.
- 4 Deploy the Jaguar component.

Three new events have been added to PowerBuilder to support Jaguar components. Jaguar calls these events at various stages of the component instance's life cycle. Table 6.1 lists the events that need to be added.

Table 6.1 Events to be added

Event Name	Event ID
Activate	Pbm_component_activate
CanBePooled	Pbm_component_cانبepooled
Deactivate	Pbm_component_deactivate

The Activate and Deactivate events take the place of the constructor and destructor events when using instance pooling and building a stateless component. The CanBePooled event is used to tell the Jaguar server whether or not the component instance can be pooled. This event is called when the Pooling component property is disabled. If the Pooling component property is enabled, the component instance is pooled and the event is not invoked. These events are covered in more detail in chapters 8 and 9.

For the developer building Jaguar components, there is a built-in wizard that makes creating a Jaguar component even easier. The wizard walks the developer through all the steps involved with creating a custom class user object and the Jaguar Component project that is used to deploy the object to the Jaguar server. The wizard also adds the events activate, and deactivate to the NVO automatically.

6.2 The Jaguar component wizards

Every PowerBuilder application requires a PowerBuilder library (PBL) and an application object. A Jaguar component is no different. In fact each Jaguar component is a separate PowerBuilder application. When building a Jaguar component, the application object will not require any code, as it is only used for specifying the library search path and telling the PowerBuilder environment which application is being worked on. In order to work on objects in PowerBuilder, you must specify a particular application object. The application object library path must contain all of the PowerBuilder libraries that hold the objects that are being developed so that you can save changes.

A Jaguar component wizard has been provided that will create the application object and initial PBL as well as the Jaguar component (NVO) and the Jaguar component project used to deploy the component. From the PowerBuilder main menu, choose File | New or click on the New icon on the tool bar (see figure 6.3) to open the New dialog box, as shown in figure 6.4. Click on the Start Wizards tab and select the Jaguar Component.

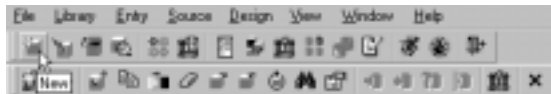


Figure 6.3
The PowerBuilder main menu

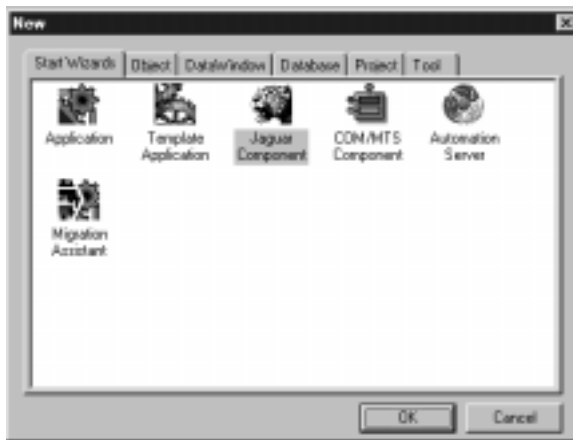


Figure 6.4
The New dialog box

The wizard displays the steps that will be taken to build a Jaguar component application. Before continuing, make sure that you know the Jaguar server location and have a valid user ID and password that is a member of the Jaguar Admin role.

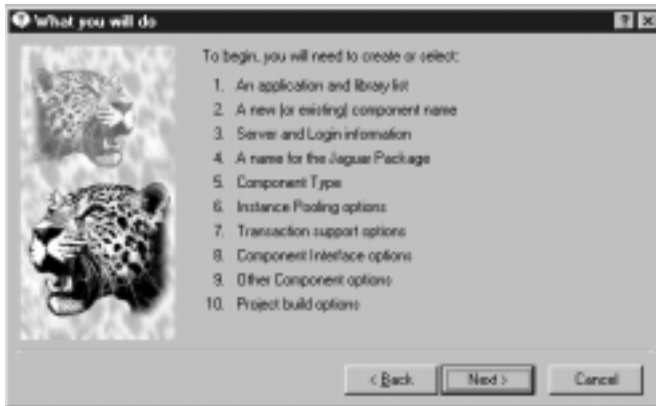


Figure 6.5
Jaguar wizard steps

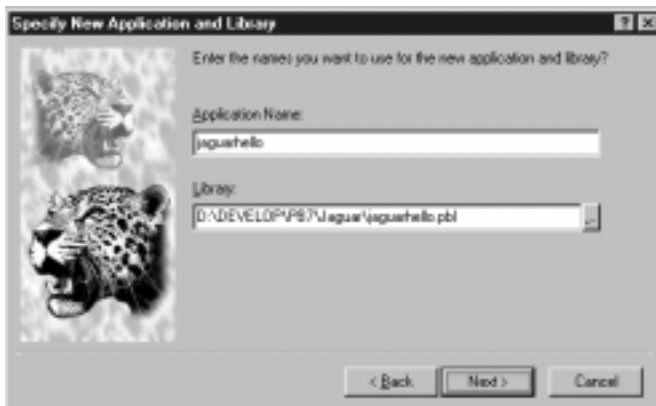


Figure 6.6
Application name and library

The first thing the wizard will require is the name of the application object and the name and location of the PowerBuilder library (PBL). Type in the information as shown in figure 6.6. The next step will be to adjust the library search path. Any existing PowerBuilder libraries can be added to the library list now (see figure 6.7). The library search path can be modified later using the File | Library List option in the Library Painter.

The Specify New Component dialog box prompts the developer for the component name and a description, as depicted in figure 6.8. The component name specified on the



Figure 6.7
Library search path

window in figure 6.8 is the name of the nonvisual object, as it will appear in the PowerBuilder PBL. A different name can be defined for the Jaguar component (see figure 6.9). This allows PowerBuilder developers to use the naming conventions that they are most familiar with, while implementing a separate naming convention for the Jaguar objects. In this example we let the Jaguar component and the PowerBuilder NVO use the same name.



Figure 6.8
Component name

The connection information required for the Jaguar server in which the component will be deployed is specified in the Specify Server Information dialog box, illustrated in



Figure 6.9
Jaguar component name



Figure 6.10
Jaguar server information

figure 6.10. In order for PowerBuilder to connect to the Jaguar server, the server must have an IIOP listener setup on the port defined in the dialog box. Jaguar CTS has an IIOP listener setup on port 9000 by default when it is installed. When the PowerBuilder development environment and the Jaguar CTS server are not on the same machine, Jaguar must have the IIOP listener configured to listen on a host other than localhost. Remote machines cannot access the Jaguar server without changing the listener's default host name. This is covered in chapter 3.

**Tip**

When specifying the Login information for the Jaguar server in the wizard, make sure that the login ID is a member of the Admin Role, otherwise the wizard cannot deploy the component to the Jaguar server.

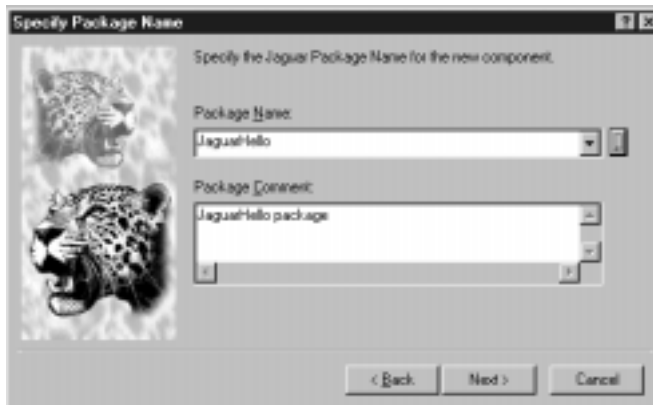


Figure 6.11
Package name

All components in Jaguar are grouped in packages. You cannot install a component on Jaguar outside of a package. Packages are covered in detail in chapter 4. You can place the component being built with the wizard in a new package or in a package that already exists on the server. You can use the small button next to the Package name field to browse packages installed on the Jaguar server (see figure 6.11). Only packages installed on the Jaguar server defined in the Specify Server Information dialog box in figure 6.10 will be listed. If the Jaguar server has not been started or is not listening on the specified port the developer can continue with the wizard but will not be able to verify the package name.

The Component type is specified next. A Jaguar component can be defined as one of three types:

- standard component
- shared component
- service component

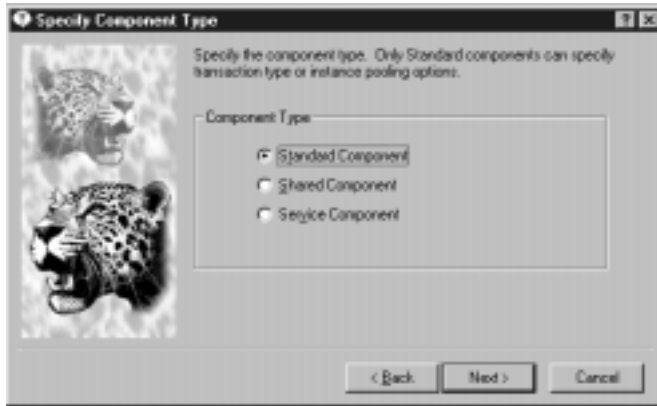


Figure 6.12
Component type

Select the *standard component* type (see figure 6.12). Most components will be standard components. The shared component is similar to the PowerBuilder Shared Object. A Jaguar shared object forces the component to have only a single instance active in the Jaguar kernel. This allows components to share information, cache data, and store information on state. The Service components are special Jaguar components that run in the background and perform various processing tasks.



Figure 6.13
Instance pooling options

The instance pooling and transaction support options are specified in the next two dialog boxes, as shown in figures 6.13 and 6.14. When a component supports instance pooling, the actual object instance is not destroyed but placed in a pool for reuse by other client applications. Instance pooling is covered in chapter 8. Transaction support deter-

mines whether or not a component will use the Jaguar Transaction Manager and how the component behaves when it is accessed by other components that have ongoing transactions. These options are covered in detail in chapter 11. The auto demarcation/deactivation option is enabled for stateless Jaguar components and is covered in detail in chapter 8.

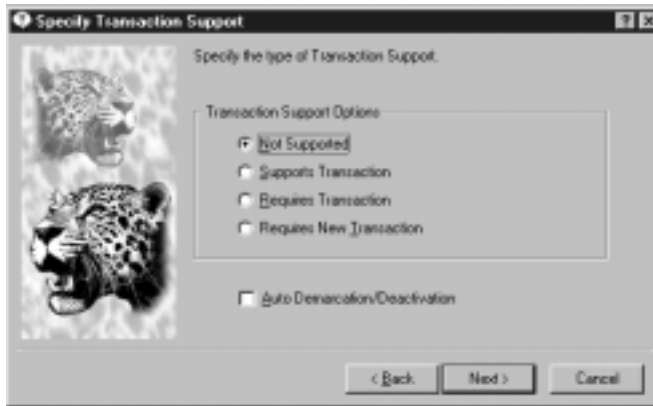


Figure 6.14
Transaction support

A component's interface describes how client applications and other Jaguar components can interact with it. Jaguar exposes all of the public functions on an object as part of the interface. This information is stored in the Jaguar repository as IDL modules, which is covered in chapter 4. All of the exposed functions must adhere to the CORBA IDL specification. Client applications and other Jaguar components can invoke any of the functions exposed on the interface. A PowerBuilder component can choose to expose user events and public instance variables as part of its interface in addition to public functions (see figure 6.15).

I do not recommend exposing variables or events as part of the public interface. Access to attributes and events can be provided using method calls. Exposing variables breaks the object-oriented rule of encapsulation. Events are typically used to respond to system events and provide a way to extend and override code easily. CORBA IDL does not support events, so events are made public by creating a function in the proxy that maps to the event. The generated function uses a "name mangling" technique so that the event and the function are mapped correctly. This is transparent to PowerBuilder clients and components, but exposing events as part of the interface on a Jaguar component affects the IDL and how components and clients written in other languages call component methods that are actually events.

CORBA IDL does not support passing NULL values as arguments. PowerBuilder components supporting NULL values will have their interfaces altered. An extra argument will be added to all public methods indicating which values have a NULL value. This is transparent to PowerBuilder clients and components, but the change to the interface is visible in the IDL and therefore must be managed by components and clients written in other languages.

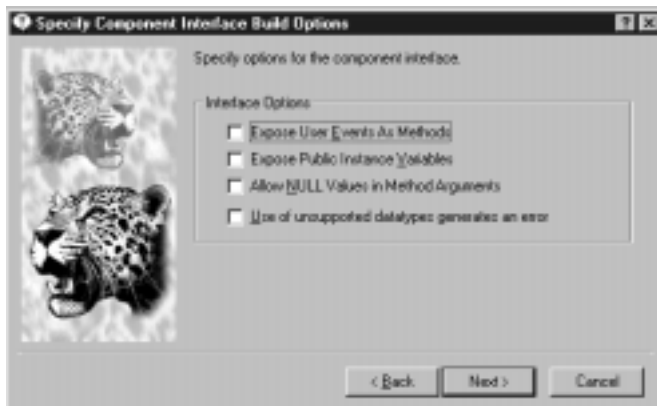


Figure 6.15
Interface options

During development there are two very useful options that can make component development easier. They are remote debugging and live editing (see figure 6.16). Enabling remote debugging on a component allows the developer to use the new remote debugger to walk through server-side code. Anyone who has written a Distributed PowerBuilder application will quickly see the value of this option. The debugger is covered in chapter 13.



Tip

Make sure that remote debugging is turned off when moving objects into production. Leaving remote debugging on can impact performance.

Live editing is another useful feature in development. When Live editing is enabled, the PowerBuilder object is deployed to the Jaguar server specified in the Specifying Server Information dialog box (see figure 6.10) whenever the object is saved. This frees

the developer from having to constantly perform the step of launching the project painter and deploying the object manually. This option should only be used when the project specifies the Jaguar server used for development. In order to avoid prematurely deploying objects to a production server, build a separate project to install components with this option turned off.

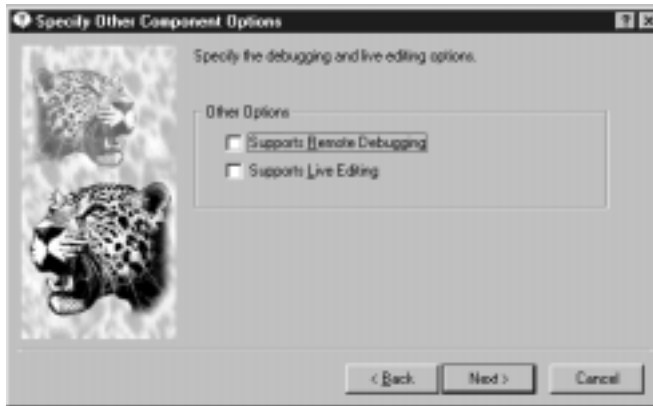


Figure 6.16
Debugging and live editing

The project name is defined in the Specify Project dialog box, depicted in figure 6.17 on p. 142. The project is where most of the information gathered by the wizard is stored. You use the project to actually deploy the object on the Jaguar server. You can use a project to deploy several components at once, but they must all be installed to the same package. You can add additional components to the project in the project painter. A naming convention for projects might be useful to indicate which servers and packages the project is deploying to. For example:

```
p_<ServerName>_<PackageName>
```

Using this convention, a project that deploys components to the JaguarHello package on the JagDev server might be named p_JagDev_JaguarHello.

The final step before finishing with the wizard is to specify the library options used to deploy the PowerBuilder objects to Jaguar (see figure 6.18 on p. 142). The PowerBuilder objects must be installed on the Jaguar server machine so that they can be instantiated. The PowerBuilder objects are shipped to the Jaguar server as PowerBuilder dynamic libraries (PBD). The PBD option is the only deployment option available, the machine code (DLL) option is not supported. This actually makes the most sense as it allows the PowerBuilder objects to be used on any platform. A developer has the option of shipping

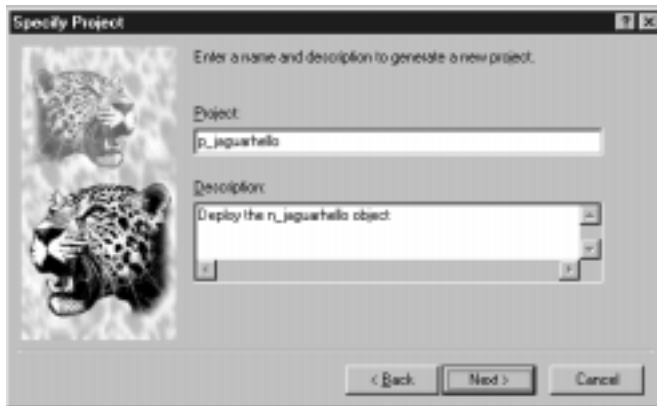


Figure 6.17
Project name

a separate PBD for each PBL in the library search path of the application object or consolidating all the objects into a single PBD. In addition, the developer can choose to include un-referenced objects in the consolidated PBD. This is important when the component uses DataWindow objects, which would not be included in the PBD that is deployed to the Jaguar server if you disable this option. An alternative to using the Include Unreferenced Objects option is to specify the objects in the PowerBuilder resource (PBR) file.

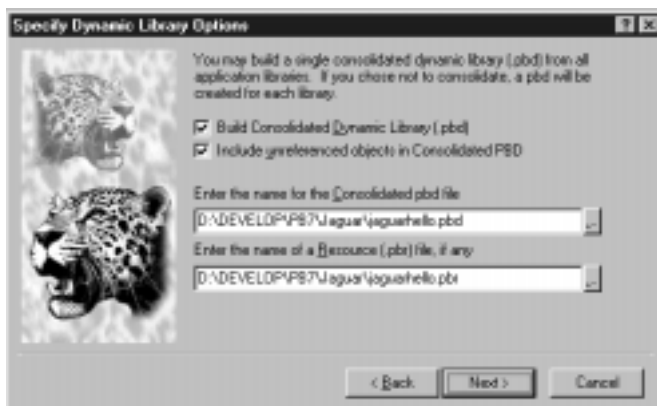


Figure 6.18
Dynamic library options

Once the library options are defined, the Jaguar component is ready to be created. Review the characteristics and then click on the Finish button, as shown in figure 6.19.



Figure 6.19
The final step

The wizard creates the JaguarHello.pbl and a JaguarHello.pbr file. The JaguarHello.pbl contains the following objects:

- application object (jaguarhello)
- project object (p_jaguarhello)
- user object (n_jaguar_hello)

6.3 Building a Hello World object

In this chapter we will build the Jaguar object for the classic Hello World example. In the next chapter we will build a client to access this object. The Hello World example will help demonstrate all the basic concepts involved with building a Jaguar application. The rest of the book will explore the more advanced features used to build more complicated objects capable of delivering data and handling transactions.

The Hello World example involves the following steps:

- 1 Building a Jaguar component
- 2 Deploying the Jaguar component
- 3 Generating a Jaguar proxy for the component
- 4 Building a Jaguar client
- 5 Testing the application

The overview of the Hello World application is illustrated in figure 6.20. The figure shows the application running over a network. This example can also be run on a single machine.

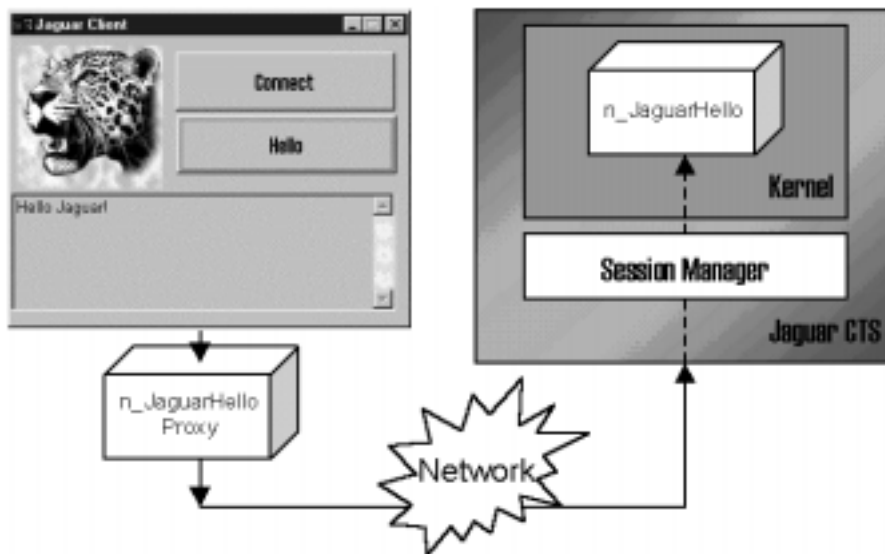


Figure 6.20 Hello World overview

To start, you will need to create a PBL, application object, and a NVO. I recommend using the Jaguar Component wizard. Once you generate the PBL and the objects with the wizard, you are ready to begin coding methods and functionality. You can build the Jaguar Hello World object (`n_JaguarHello`) using the Jaguar component wizard or using the custom-class option from the New dialog box. Both options will end up in the User Object painter, where you will write the object. The rest of this example will assume that you used the wizard.

Before continuing, let's take a look at a new PowerBuilder 7 feature—the To Do List. The To Do list consists of the steps that a developer must perform in order to complete a specific task. The To Do list, shown in figure 6.21, was generated by the Jaguar wizard. To open the To Do list, select the Window | To Do List option from the main menu or click on the To Do List toolbar icon (see figure 6.22).

The wizard has already built the project to deploy the Jaguar component, so we can check that off of our list. You will not need a PBR file so we can cross that off the list as

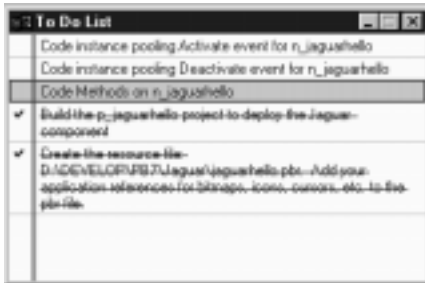


Figure 6.21
To Do list



Figure 6.22
To Do list toolbar icon

well. All that is left to do is code your Hello World method. To write the Hello World method you will need to open the User Object painter.

The easiest way to open the User Object painter is by clicking on the object `n_JaguarHello` in the library painter. You can also open the user object painter by choosing `File | Open` from the main menu or by clicking on the `Open` toolbar option (see figure 6.23)



Figure 6.23
Open toolbar icon



Tip

Double-clicking on the code methods task in the To Do list will launch the User Object painter and open the appropriate object.

When using the menu or toolbar option, you'll see the `Open` dialog box, as illustrated in figure 6.24. The developer can choose any PowerBuilder object type from the `Object Type` drop-down list box. In this example we will select the `User Objects` option. After selecting the `User Objects` from the `Object Type` list, you will see all the user objects in the library highlighted in the `Application Libraries` list in the top panel. Click on the object that you wish to open and hit the `OK` button. This will open up the object in the appropriate painter.



Figure 6.24
Open Painter dialog box



Figure 6.25 The User Object painter

Once in the User Object painter, you will notice the new interface that has been introduced in PowerBuilder 7 (see figure 6.25). To add a new function, choose the Insert | Function option from the main menu, or click on the Function List tab and right-click to open the popup menu. Choose the Add option from the popup menu, as shown in figure 6.26



Figure 6.26
Function popup menu

A third method is to choose Functions from the drop-down list box above the Access label in the Script sheet. After choosing Functions, select the New Function option from the drop-down list box above the Return Type label (see figure 6.27). A new function can also be added by right-clicking on the area above the script editor where the drop-down list boxes are contained. This will open a popup menu that contains options to add a new function, event, or parameter as well as delete the function.

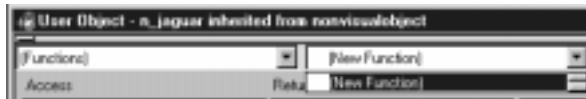


Figure 6.27
Adding a new function

Once you've added a new function, type the name of the function and specify the return type as a string, as shown in figure 6.28 on p. 148. The function name in this example is of_Hello. The function accepts no arguments and returns a string.

In the script portion of the screen add the following code:

```
RETURN "Hello Jaguar!"
```

After coding the script for the Hello World function, save the object. We have finished building the n_JaguarHello object.

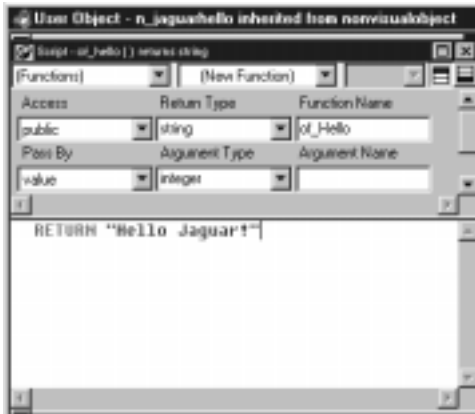


Figure 6.28
The Hello World function

6.4 Deploying the Jaguar object

Once you have finished writing your Hello World object you will need to deploy it to the Jaguar server. If you used the wizard option to build the Jaguar object, you've already built the Jaguar Component project. If you built the component using the custom class option instead of the wizard, you will need to build a Jaguar Component project now. See the section called *Building a Project using the Project Wizard*, at the end of this chapter.

If you enabled the live-editing option, the Jaguar object has already been installed on the Jaguar server. When this option is not implemented, the Jaguar component is deployed through the project. Open the project object in the project painter, and click the build option (see figure 6.29). That is it! You've deployed the Jaguar component on the Jaguar server specified in the project properties.



Tip

The Jaguar server must be started and listening on the specified port using IIOP to successfully deploy objects. In addition, you must supply the login user ID and password if you have not selected the Save Login option. If a build fails because the Jaguar server is not available, the message box, shown in figure 6.30, is displayed with the COMM_FAILURE. If the login is incorrect, a similar message box will display with a NO_PERMISSION error, as pictured in figure 6.31.



Figure 6.29
The project

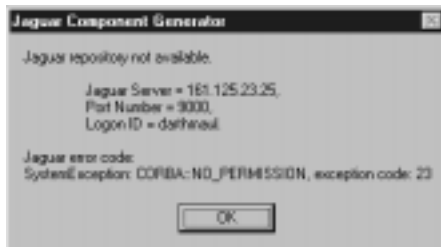


Figure 6.30
Login ID is not authorized to deploy components

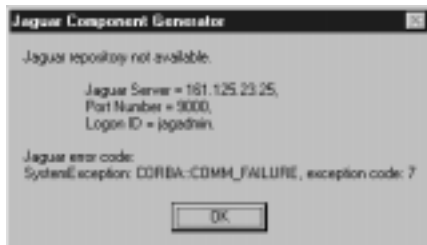


Figure 6.31
Jaguar Server is not available

After you have completed the build, log onto the Jaguar server, using the Jaguar Manager, to verify that the package and component have been successfully deployed. The package and the component are shown in the Jaguar Manager under the Installed Packages folder of the Jaguar server, as illustrated in figure 6.32 on p. 150. Congratulations, you have just built and deployed your first Jaguar object.

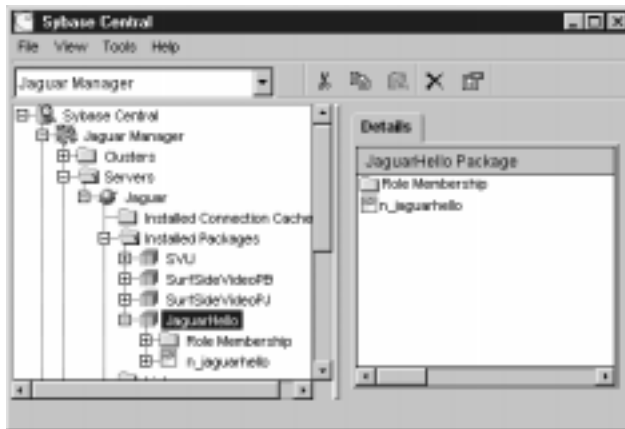


Figure 6.32
The component installed on the Jaguar server

6.5 Adding another Jaguar component

In order to build additional Jaguar components without creating a new PBL or application object each time, you can use the Jaguar Component Wizard on the Object tab, as shown in figure 6.33. This wizard is used to generate only the NVO and the project object, placing them in an existing PBL.

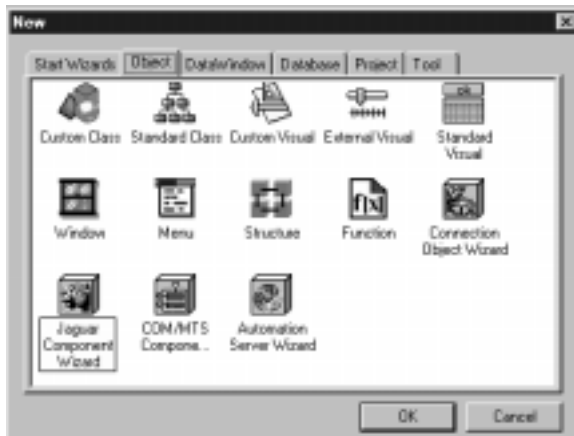


Figure 6.33
Jaguar component object wizard

The Jaguar Component Wizard will prompt the developer for the PBL in which the new object should be created (see figure 6.34). After this step, the Jaguar Component Wizard will continue opening dialog boxes, starting with the one that asks for the com-

ponent name and properties, shown in figure 6.8. It continues from this point using the same set of dialog boxes as the Jaguar Component Wizard accessed from the Start Wizards tab.



Figure 6.34
Specifying the library

6.6 Building a project using the project wizard

To build a project to deploy a component to Jaguar for a custom-class object, select File | New from the main menu to open the New dialog box. Select the Project tab, and choose the Jaguar Component Wizard, as shown in figure 6.35. This wizard will prompt you for the PBL in which the generated project will be placed (see figure 6.34) and the name of the project (see figure 6.17).

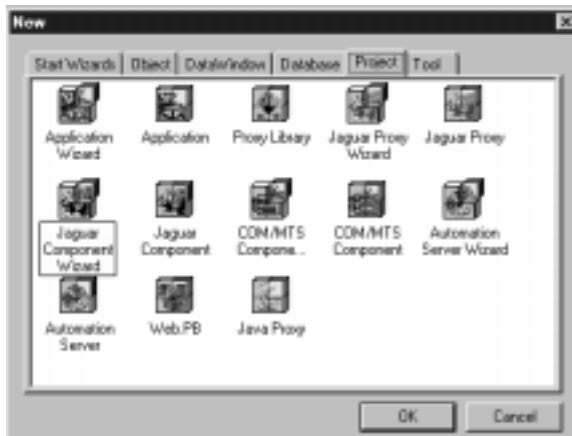


Figure 6.35
Jaguar component project wizard

After this step, the Jaguar component wizard will continue opening dialog boxes, starting with the one that asks for the server information, shown in figure 6.10. It continues from this point using the same set of dialog boxes as the Jaguar Component Wizard accessed from the Start Wizards tab, skipping the Project Name dialog box in figure 6.17.

6.7 Summary

This chapter has covered the basics on how to build a simple object in PowerBuilder and deploy it to a Jaguar server. After reading this chapter, you should understand how to use the Jaguar wizards in PowerBuilder to build and deploy a Jaguar component. In the next chapter you will build a client application in PowerBuilder that will access your Jaguar component.