G2 Bean Builder

User's Guide

Version 2015

G2 Bean Builder User's Guide, Version 2015
December 2015

Gensym Corporation
52 Second Avenue
Burlington, MA 01803 USA
Telephone: (781) 265-7100
Fax: (781) 265-7101

Part Number: DOC056-1200

Contents

Preface

Describes this guide and the conventions that it uses.

About this Guide

This guide describes the basic features of G2 Bean Builder and explains how you can use these features to create applications.

Software Requirements

For information about the software requirements and installation of this version of G2 Bean Builder, refer to the `readme.html` file supplied with G2 JavaLink, which includes this and other important instructions.

Audience

This document assumes that you are familiar with the features and syntax of the Java programming language, and that you know how to develop Java applications. It also assumes that you are familiar with the syntax of G2.

Conventions

This guide uses the following typographic conventions and conventions for defining system procedures.

Typographic

Convention Examples	Description
g2-window, g2-window-1, ws-top-level, sys-mod	User-defined and system-defined G2 class names, instance names, workspace names, and module names
history-keeping-spec, temperature	User-defined and system-defined G2 attribute names
true, 1.234, ok, “Burlington, MA”	G2 attribute values and values specified or viewed through dialogs
Main Menu > Start KB Workspace > New Object create subworkspace Start Procedure	G2 menu choices and button labels
conclude that the x of y ...	Text of G2 procedures, methods, functions, formulas, and expressions
new-argument	User-specified values in syntax descriptions
<u>*text-string*</u>	Return values of G2 procedures and methods in syntax descriptions
File Name, OK, Apply, Cancel, General, Edit Scroll Area	GUIDE and native dialog fields, button labels, tabs, and titles
File > Save Properties	GMS and native menu choices
workspace	Glossary terms

Convention Examples	Description
`c:\Program Files\Gensym\`	Windows pathnames
`/usr/gensym/g2/kbs`	UNIX pathnames
`spreadsh.kb`	File names
`g2 -kb top.kb`	Operating system commands
`public void main()` `gsi_start`	Java, C and all other external code

Note Syntax conventions are fully described in the *G2 Reference Manual*.

Procedure Signatures

A procedure signature is a complete syntactic summary of a procedure or method. A procedure signature shows values supplied by the user in *italics*, and the value (if any) returned by the procedure underlined. Each value is followed by its type:

g2-clone-and-transfer-objects
(*list*: class item-list, *to-workspace*: class kb-workspace,
delta-x: integer, *delta-y*: integer)
-> *transferred-items*: g2-list

Related Documentation

G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer's Guide*
- *G2 System Procedures Reference Manual*

- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*
- *Telewindows User's Guide*
- *G2 Gateway Bridge Developer's Guide*

G2 Utilities

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*
- *G2 GUIDE User's Guide*
- *G2 GUIDE/UIL Procedures Reference Manual*

G2 Developers' Utilities

- *Business Process Management System Users' Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*
- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

Bridges and External Systems

- *G2 ActiveXLink User's Guide*
- *G2 CORBALink User's Guide*
- *G2 Database Bridge User's Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User's Guide*
- *G2 Java Socket Manager User's Guide*
- *G2 JMSLink User's Guide*
- *G2 OPCLink User's Guide*
- *G2 PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*
- *G2 CORBALink User's Guide*
- *G2 WebLink User's Guide*

G2 JavaLink

- *G2 JavaLink User's Guide*
- *G2 DownloadInterfaces User's Guide*
- *G2 Bean Builder User's Guide*

G2 Diagnostic Assistant

- *GDA User's Guide*
- *GDA Reference Manual*
- *GDA API Reference*

Customer Support Services

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone, by fax, and by email.

To obtain customer support online:

➔ Access G2 HelpLink at `www.gensym-support.com`.

You will be asked to log in to an existing account or create a new account if necessary. G2 HelpLink allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.
- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

To obtain customer support by telephone, fax, or email:

➔ Use the following numbers and addresses:

	Americas	Europe, Middle-East, Africa (EMEA)
Phone	(781) 265-7301	+31-71-5682622
Fax	(781) 265-7255	+31-71-5682621
Email	`service@gensym.com`	`service-ema@gensym.com`

To obtain G2 JavaLink specific customer support by e-mail:

➔ Use the following address:

`javalink-service@gensym.com`

Introduction

Introduces the G2 Bean Builder.

Introduction

The G2 Bean Builder is a Java-based wizard that uses G2 JavaLink to create a Java Bean from a G2 class.

You use the G2 Bean Builder to build a Java Bean in its own JAR file by either:

- Providing the data through the wizard.
- Entering the data as command-line options.

Please refer to the *G2 JavaLink User's Guide* for further in-depth information on G2-Java communication and the mapping of G2 types and classes to Java.

On the Windows platform, the G2 Bean Builder can package Java Beans as ActiveX controls for use with third-party ActiveX development environments, such as Visual Basic. Note that platform restrictions may apply.

This feature uses an ActiveX packager from Sun Microsystems that creates OLE type library information and win32 registry information for a selected Java Bean, which allows OLE/COM containers to correctly analyze and present a Bean. For example, a Bean's properties are correctly presented in a Visual Basic property sheet.

The two sources of the ActiveX packager available from Sun Microsystems are:

- Java Bean to ActiveX Bridge
- Java Plug-in

Previous versions of the G2 Bean Builder have used the packager distributed with Sun Microsystems' Java Bean to ActiveX Bridge. However, the G2 Bean Builder now uses the packager included with Sun's Java Plug-in, which is included with JDK.

Java Beans

A Java Bean is:

"... a reusable platform-neutral software component that can be visually manipulated in a software development tool "- Java Beans Specification 1.0

Java Bean components are called Beans. Beans may be thought of as segments of code, which, in addition to their designed function, can also "plug-and-play" with other Beans.

Adopting Beans as the unit of software allows the Bean developer to create the basic functional units. The user then assembles these units, using a visual Bean-editing environment to create the final application.

G2 Java Beans

A G2 Java Bean/ActiveX control created by the G2 Bean Builder for a specific G2 class is a reusable visual component that you can use within a Java Beans- or ActiveX-compliant development environment. The resulting Bean also exposes the interfaces of the original G2 class so that third-party development environments such as Visual Basic or Visual Café can present methods and attributes of the original G2 class at development time. For the Bean to be useful, it must be configured to access an instance of the original G2 class within a G2 server running on the network.

Once the Bean is connected to a G2 object:

- Its attribute values are synchronized with the current values of the G2 object.
- Any changes to attributes made to the Bean are reflected as attribute changes on the G2 object.
- Any changes to attributes made to the G2 object are reflected as attribute changes on the Bean.

- Any method that is exported from the original G2 class and called on the Bean results in a synchronous method call on the appropriate G2 method. Any G2 type or class can be returned from a G2 method call to the Bean.

Note Any events defined on the G2 object are also reported to the Bean. Exposed events must be defined by using the G2 Event Listener Support Module. For more information, see the *G2 JavaLink User's Guide*.

- If Gensym's Telewindows2 Toolkit is installed and the bean has been configured to use the G2 icon, then the generated Bean displays the current icon of the connected G2 object.

Differences between G2 and Java

There are differences as well as similarities between Java and G2. The following sections describe a number of these differences and how to overcome them.

Inheritance Model

The Java class inheritance model does not support multiple inheritance, whereas G2 does. To circumvent this difference, G2 JavaLink maps G2 classes to Java interfaces, which do support multiple inheritance. JavaLink creates Java *implementation* classes from the original G2 class to implement multiple interfaces, thereby mapping G2's multiple inheritance in the Java environment.

Accessing Attributes

Using Java Beans, you obtain attribute values by using methods known as getters, and you modify attributes by using methods known as setters.

For example, suppose you have a G2 class named tank with a temperature attribute. The resulting Bean would have a getter called *getTemperature*, which returns the temperature attribute value for a given tank, whose definition would look like this:

```
public int getTemperature(){
   return temperature;
}
```

For example, you could use this getter like this:

```
 if(tank1.getTemperature() > 10)....
```

You use a similar mechanism for setting the value of the attribute like this:

```
public void setTemperature(int temp){
   if((temp > 0) && (temp < 100)){
      temperture = temp;
   }else ...

}
```

Notice that the setter performs bounds checking for the attribute. That way, all the calling code has to be concerned with is setting the attribute value; the setter takes care of error checking and so on.

Events

An event is an asynchronous signal that a source object sends to a target object to notify the target that some specific behavior has occurred.

In the Java event model, a "source" can send a single event to any number of "listeners." A source is an object that generates events, and a Listener is an object that implements the appropriate listener interface so it can receive events. Sources implement standard methods so that listeners can request notification of events. After a listener registers with a source, it gets called any time an event of the requested type occurs.

For example, when an attribute value of a G2 object changes, the `propertyChangeEvent` is generated for all objects that implement the `ItemListener` interface.

To export events to external Java Beans or ActiveX controls, you must use the G2 Event Listener Support Module (`g2evlis.kb`), as described in the *G2 JavaLink User's Guide*.

2

Installing the G2 Bean Builder

Describes installation and distribution of the G2 Bean Builder.

Introduction

This chapter describes how to check the installation of the G2 Bean Builder and troubleshoot the installation. It also describes what you need to do when distributing Java Beans and ActiveX controls that the G2 Bean Builder has created.

Software Requirements

For information about the software requirements for this version of G2 Bean Builder, see the `readme-javalink.html` file supplied with G2 JavaLink.

Installing the Files

For information about installation, refer to the `readme-javalink.html` file, supplied with G2 JavaLink, which includes installation and other important instructions.

Checking the Installation

To verify the G2 Bean Builder installation, enter the following command on a command line:

```
java com.gensym.beanbuilder.G2BeanBuilder +v
```

This command provides a simple though not exhaustive self-test facility. The application should display a message that shows the result of the self test and the version of G2 Bean Builder. For example:

```
Gensym G2 Bean Builder Version 1.2 Rev. 5

Self Test Passed
```

Minimum Requirement for Distributing Java Beans

When you distribute a packaged Java Bean that you created with the G2 Bean Builder, the following requirement must be met for the packaged Java Bean to work:

Java Development Kit (JDK) 1.3, which includes the Java Plug-in 1.3, or Java Runtime Environment (JRE) 1.3.

Note For the latest software requirements, see the `readme-javalink.html` file included with G2 JavaLink.

File Locations

The G2 Bean Builder uses the property named `com.gensym.class.user.repository`, which is defined in the JavaLink properties file `.com.gensym.properties`, to indicate where to create JAR files.

For example, the property might be defined as follows, where *g2-install-dir* is your G2 installation directory:

```
com.gensym.class.user.repository=g2-install-dir\javalink\classes
```

When the G2 Bean Builder creates the JAR file, it appends `\jars` to the specified path and uses the resulting path as the destination directory for its JAR files. In the example above, it would create JAR files in this directory of your G2 installation directory:

```
\javalink\classes\jars
```

You can also specify the JAR file location by using the:

- G2 Bean Builder Dir type-in box.
- `-dir` command-line option.

If you use either of these options, the G2 Bean Builder creates JAR files in the specified directory.

The G2 Bean Builder automatically creates ActiveX controls in the `activex` directory within the JAR directory, for example:

```
\javalink\classes\jars\activex
```

In order for the ActiveX components to function in environments such as Visual Basic, you must copy the `beans.ocx` file located in the `JavaPlugin\bin` directory to this `activex` directory.

For further information about the properties file and the JavaLink file location, see the *G2 JavaLink User's Guide*.

Troubleshooting

This section provides help troubleshooting the installation.

Running the Beans Development Kit (BDK)

You use Sun's Beans Development Kit (BDK) V1.1 to test Java Beans created from G2 Bean Builder. You run BDK from the following batch file:

```
../BDK/beanbox/run.bat
```

The batch files look like this on each platform:

- Windows:

  ```
  if "%OS%" == "Windows_NT" setlocal
  set CLASSPATH=classes;..\lib\methodtracer.jar;..\infobus.jar
  java sun.beanbox.BeanBoxFrame
  ```

- UNIX:

  ```
  #!/bin/sh
  export CLASSPATH
  CLASSPATH=classes
  java sun.beanbox.BeanBoxFrame
  ```

Unfortunately, the batch file overrides the *CLASSPATH* so that JavaLink classes are not visible to the Bean Box application. Assuming you attempt to load *ATestClassBean* from *C:\bt\mg\java\jars\ATestClassBean.jar*, the Bean Box prints in its console window the following exception message:

```
WARNING: Could not instantiate bean "com.gensym.classes.modules.
jgidemo.ATestClassBean" from JAR "C:\bt\mg\java\jars\ATestClassBean.
jar" BeanBox caught exception java.lang.SecurityException: No access
through getResource() to .class in 1.1 while processing: LoadJar msg:
No access through getResource() to .class in 1.1
```

You can resolve this problem by editing the *run.bat* file to ensure that all required JavaLink classes are on the Bean Box's *CLASSPATH*. Because the JavaLink installation ensures that the default *CLASSPATH* points to the correct JavaLink classes, the following change to line 2 normally solves the problem:

- Windows:

```
if "%OS%" == "Windows_NT" setlocal
set CLASSPATH=%CLASSPATH%;classes..\lib\methodtracer.jar;
   ..\infobus.jar
java sun.beanbox.BeanBoxFrame
```

- UNIX:

```
#!/bin/sh
export CLASSPATH
CLASSPATH=$CLASSPATH;classes
java sun.beanbox.BeanBoxFrame
```

Using the ActiveX Packager

The G2 Bean Builder uses the packager supplied with Sun's Java Plug-in to create ActiveX controls from Java Beans that have been created for G2 objects. When the G2 Bean Builder starts up, it searches for the ActiveX Packager Java startup class, *sun.beans.ole.Packager*, from the current *CLASSPATH*. If the Packager cannot be found, the G2 Bean Builder does not give the user the option of creating an ActiveX control. To ensure the Packager can be found, you should install it correctly along with Sun's Bean Development Kit.

This release of JavaLink uses the packager supplied with Sun's Java Plug-in 1.3, which is included with JDK 1.3.

To ensure that the G2 Bean Builder can find the Packager, you need to add the Java Plug-in's Java classes directory to the *CLASSPATH* as follows, assuming the Java Plug-in has been installed at *c:\Program Files\JavaSoft\JRE\1.3*:

```
set CLASSPATH=%CLASSPATH%;c:\Program Files\JavaSoft\JRE
   \1.3\lib\jaws.jar
```

In addition, you need to add the Java Plug-in's Java `bin` directory to the `PATH` as follows:

```
set PATH=%PATH%;c:\Program Files\JavaSoft\JRE\1.3\bin
```

The G2 Bean Builder places ActiveX registration (`.reg`) and type library (`.tlb`) files generated for the G2 Bean in a directory called `activex`, which is automatically created below the Java Bean JAR directory specified while building the Bean.

For example, assume the JAR is generated in this directory of your G2 installation directory:

```
\javalink\classes\jars
```

The ActiveX control files would be created in:

```
\javalink\classes\jars\activex
```

Packager Notes

- The registration file created contains absolute references to the location of the Java Bean JAR and other files. If you move any of these files, you must edit the registration file appropriately. To edit a `.reg` file entered into the Windows registry, double-click the file from a file view or use `regedit` from a DOS prompt.
- If you use the Packager to re-create the same ActiveX control, the `.reg` and `.tlb` files are overwritten. Any edits you have made to these files must be reentered.
- If an ActiveX control is previously registered and you rerun the Packager to re-create the same ActiveX Control, the Packager maintains the same GUID for the ActiveX control.
- The Packager uses a common control file called `beans.ocx` that is normally placed in `c:\Program Files\JavaSoft\JRE\1.3\bin`, assuming the Packager is installed in `c:\Program Files\JavaSoft\JRE\1.3\bin`. You must copy this file to any `activex` directory that the G2 Bean Builder creates, or the generated ActiveX controls will not work. For example, if `beans.ocx` is not found, Visual Basic would report that the generated G2 ActiveX Bean control has not been registered, when a user attempts to draw the control on a VB form.
- When switching to a new version of the Packager, you must re-create your G2 beans, using the G2 Bean Builder. If you have copies of the `beans.ocx` file from the previous version of the Packager, you must replace these with the newer version of the `beans.ocx` file supplied with Sun's Plug-in, which is found in the `\bin` directory in the Plug-in's installation directory.

Running the Wizard

Describes how to use the G2 Bean Builder wizard.

Introduction

You use the G2 Bean Builder utility by running a wizard, which provides a graphical user interface (GUI) to the utility. Alternatively, you can run the utility without the wizard.

Running the G2 Bean Builder

Before running the G2 Bean Builder, you must:

- Launch G2.
- Load the KB containing the classes for which you require Java interfaces.

You can run the G2 Bean Builder with or without a graphical user interface.

To run the G2 Bean Builder with a GUI:

➔ From a command window, execute this command:

```
java com.gensym.beanbuilder.G2BeanBuilder
```

or

➔ Launch the utility from the Start menu, by choosing the following from your Gensym G2 program group:

G2 JavaLink > G2 Bean Builder

To run G2 Bean Builder without a GUI:

➔ From a command window, execute this command:

```
java com.gensym.beanbuilder.G2DownloadInterfaces
    -class MY-CLASS
```

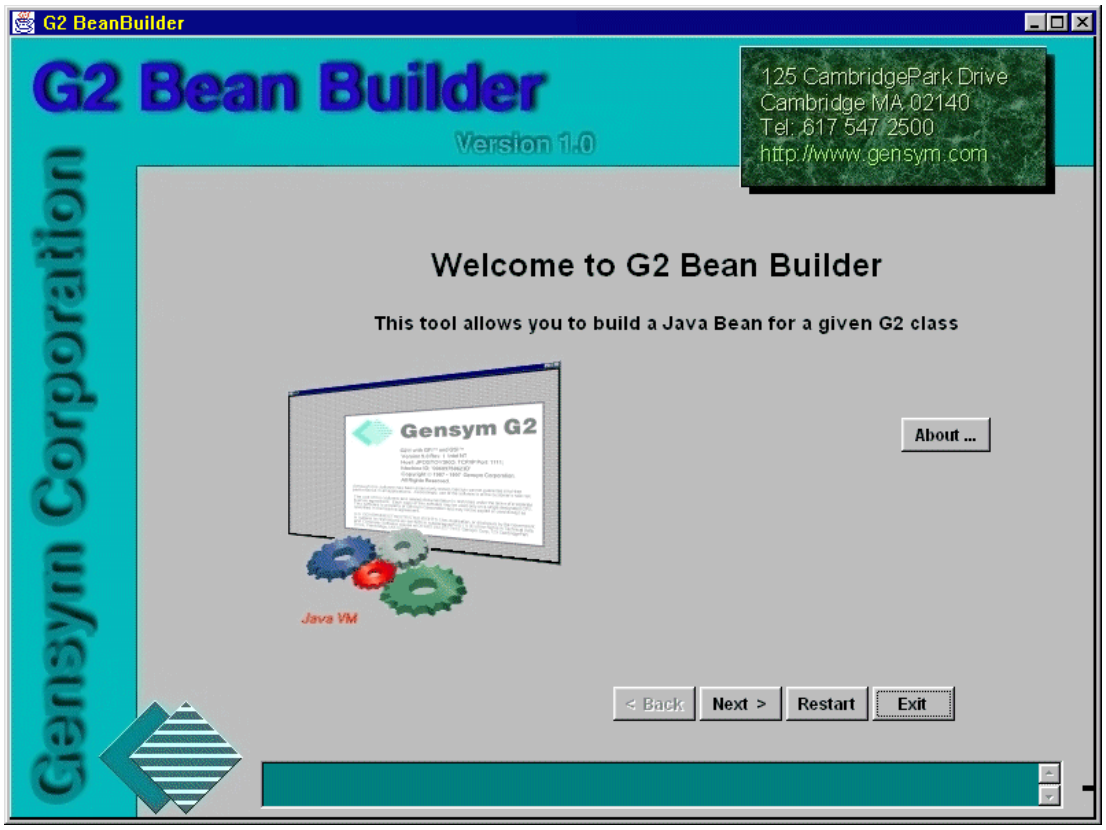
This command runs G2 DownloadInterfaces on *MY-CLASS* without a GUI.

The remaining instructions assume you have started the G2 Bean Builder with a GUI.

For information on running the G2 Bean Builder without a GUI, see Using the Command-Line Options.

Viewing the Welcome Panel

Once you start the G2 Bean Builder, the Welcome panel appears:



This panel displays the version of G2 Bean Builder currently running and provides these buttons:

Clicking this button:	Does this:
About	Displays the About screen that shows the current version of G2 JavaLink.
Back	Displays the previous panel. This button is initially inactive, because the Welcome panel is the first panel.
Next	Displays the next panel. Some panels cause this button to be inactive until you enter all of the information on that panel.

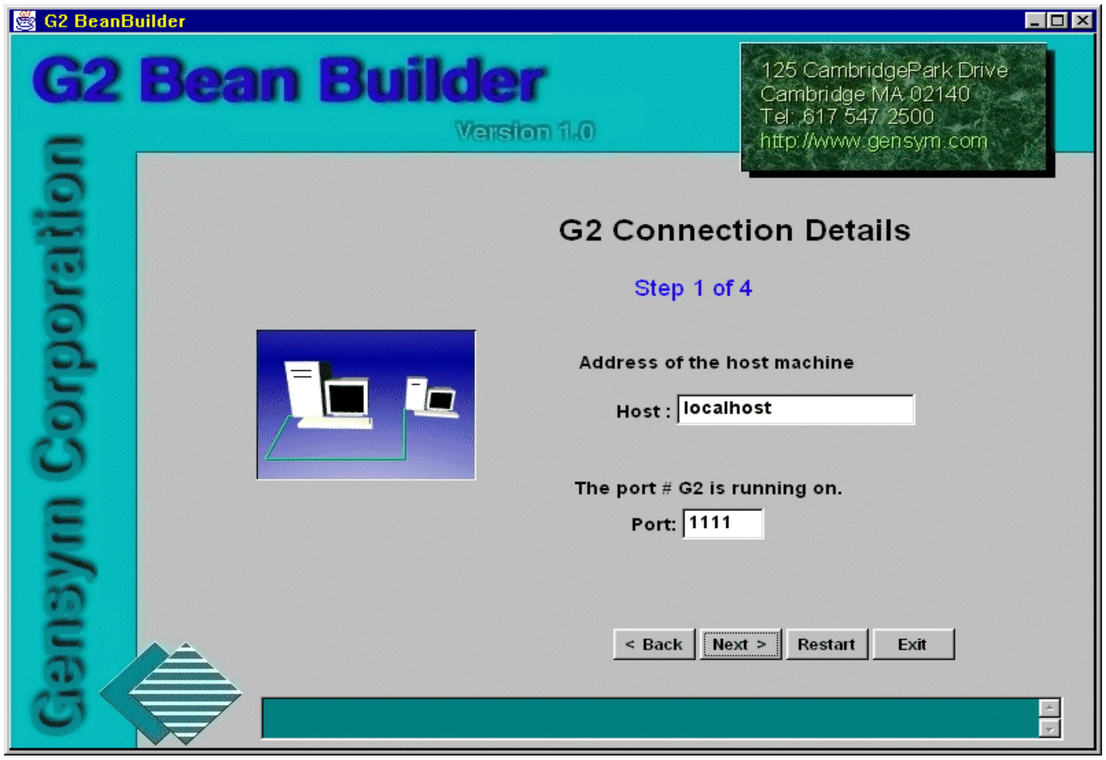
Clicking this button:	Does this:
Restart	Aborts the current build and restarts the session.
Exit	Presents a confirmation dialog asking you to confirm exiting from the G2 Bean Builder.

The four navigation buttons and the Help button are common to all of the G2 Bean Builder panels.

Click Next to advance to the G2 Connection Details panel.

Configuring the G2 Connection Details

The G2 Connection Details panel configures the connection between the G2 Bean Builder and a G2 host:



Address of the host machine

Host : localhost

The port # G2 is running on.

Port: 1111

This panel has two specific controls:

- Host field
- Port field

Configure the Host and Port, then click Next to advance to the G2 Class Name panel.

Specifying the Host

The Host field holds the name or IP address of the machine running the target G2.

By default, the default Host is localhost, unless you started the G2 Bean Builder with the *-host* command-line option, as described in Using the Command-Line Options.

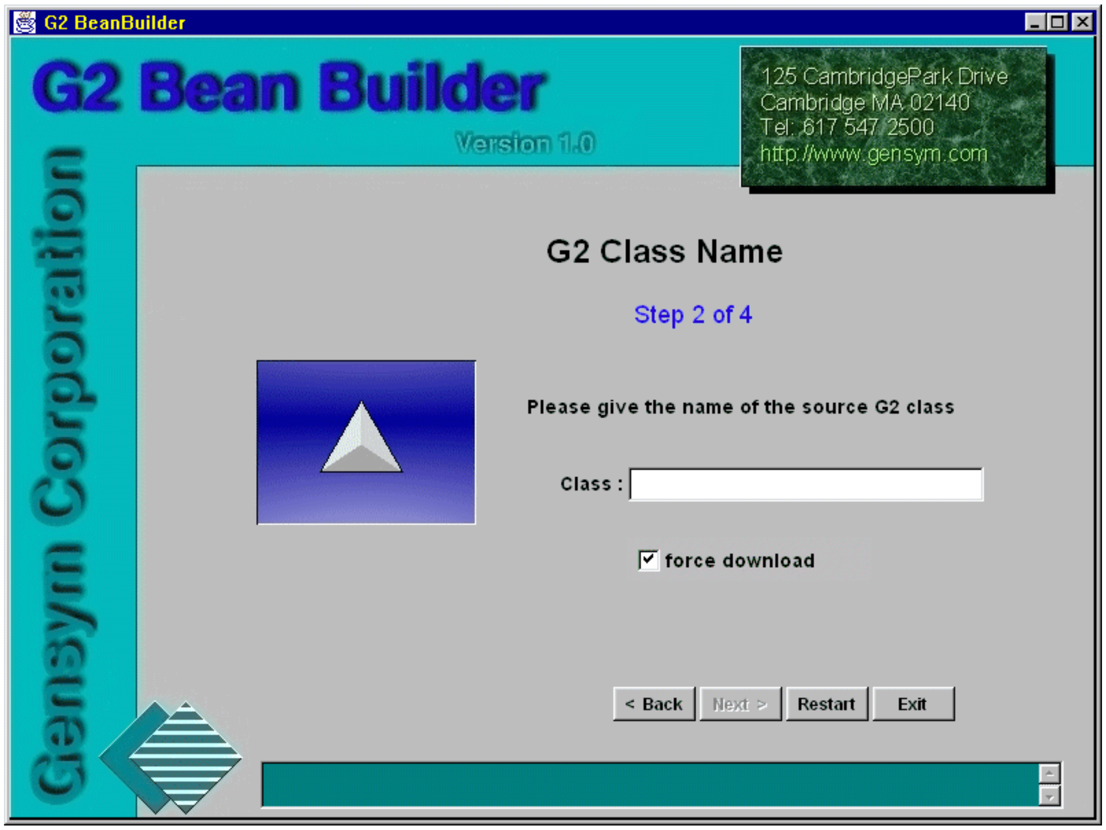
Specifying the Port

The Port field holds the TCP/IP port number at which G2 is running on the target machine.

By default, the default Port is 1111, unless you started the G2 Bean Builder with the *-port* command-line option, as described in Using the Command-Line Options.

Configuring the G2 Class Name

The G2 Class Name panel lets you specify the G2 class for which to create a Java Bean:



G2 Class Name

Step 2 of 4

Please give the name of the source G2 class

Class :

[✓] force download

This panel contains two specific controls:

- Class field
- Force download checkbox

Configure these controls, then click Next to advance to the Iconic Representation panel.

Entering the G2 Class

The Class field holds the G2 class for which you want to build a Java Bean. When entering the G2 class through the GUI, case does not matter but you must use hyphens if they appear in the G2 class name.

By default, the Class field is blank, unless you started the G2 Bean Builder with the `-class` command-line option, as described in Using the Command-Line Options.

The Next button is disabled unless this field has a value.

Forcing an Interface Download

The force download checkbox allows interfaces to be created during development so that they pre-exist at deployment time. You should use this option when a G2 class or its methods have changed in some way. Forcing the download guarantees that the Java interface classes are up-to-date.

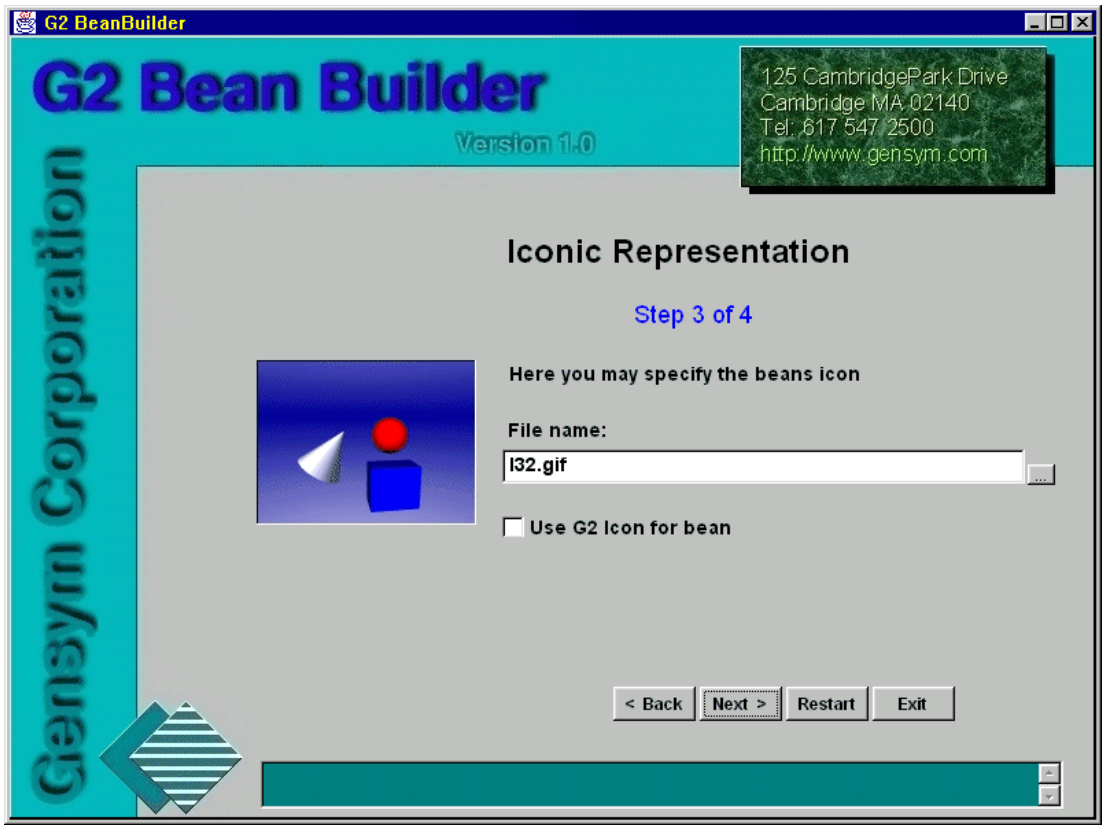
When the option is not checked, JavaLink only creates the interfaces that it requires, reusing any that were previously downloaded.

You use this feature when using the G2 Bean Builder to “repackage” previously packaged G2 classes, perhaps for creating a JAR file with a new file name or for changing the graphics of the icon.

By default, the force download checkbox is true. You can disable the force download option by unchecking the checkbox.

Configuring the Iconic Representation

The Iconic Representation panel allows you to select an image to use as the iconic representation for the Bean:



Iconic Representation

Step 3 of 4

Here you may specify the beans icon

File name: I32.gif

[ ] Use G2 Icon for bean

Typically, a Java Bean has associated with it 4 icon files, one each of size 16x16 and 32x32 pixels, both in monochrome and in color. These files are stored within the JAR file. The Bean's `beaninfo` file returns the relevant icon when prompted.

By contrast, the G2 Bean Builder stores one 32x32 image as raw data within the `beaninfo` file. You can use a wide range of graphics file types to specify this image. The source graphics file need not be 32x32, because the G2 Bean Builder automatically scales and stores the image at this resolution. The IDE automatically dithers monochrome images, so there is no need to explicitly store them.

This panel has two specific controls:

- File name field
- Use G2 icon for bean checkbox

Configure these controls, then click the Next button to advance to the Bean Build Information panel.

Entering the File Name

The File name field holds the name of a valid graphics file from which to create an icon for the Bean.

By default, the File name field defaults to `I32.gif`, which is located in the `images` directory of the G2 Bean Builder product directory. If you started the G2 Bean Builder with the `-iconfile` command-line option, as described in Using the Command-Line Options, the File name field defaults to the file name you specified in the command-line argument.

To specify a new file name, click the button next to the field to display a file dialog from which to choose the source image for the Bean's icon.

The File name field is disabled if the Use G2 icon for bean checkbox is true.

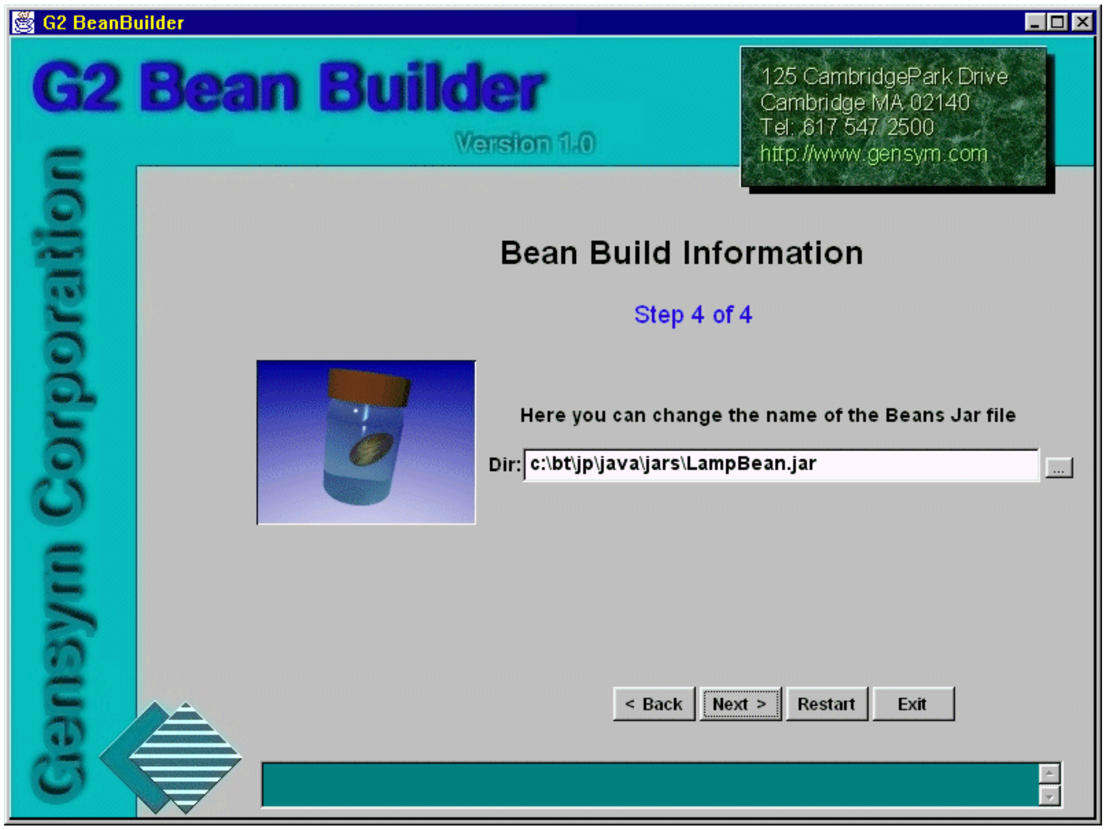
Using the G2 Icon for the Bean

The Use G2 icon for bean checkbox, when checked, causes the G2 Bean Builder to use the G2 icon of the specified class as the image for the Bean. The image is automatically rescaled.

By default, this checkbox is true, unless you started the G2 Bean Builder with the `-notg2icon` command-line option, as described in Using the Command-Line Options.

Configuring the JAR File

The Bean Build Information panel allows you to specify the name and location of the JAR file to create:



Bean Build Information

Step 4 of 4

Here you can change the name of the Beans Jar file

Dir: c:\bt\jp\java\jars\LampBean.jar

Configure the JAR file name and directory, then click the Next button to advance to the next panel. Clicking Next displays one of two panels:

- When running on a Windows platform with Sun's Java Plug-in installed, clicking the Next button advances to the ActiveX generation panel.
- Otherwise, clicking the Next button advances to the Summary panel.

Entering the Directory and File Name

The Dir field holds the name and location of the JAR file to create.

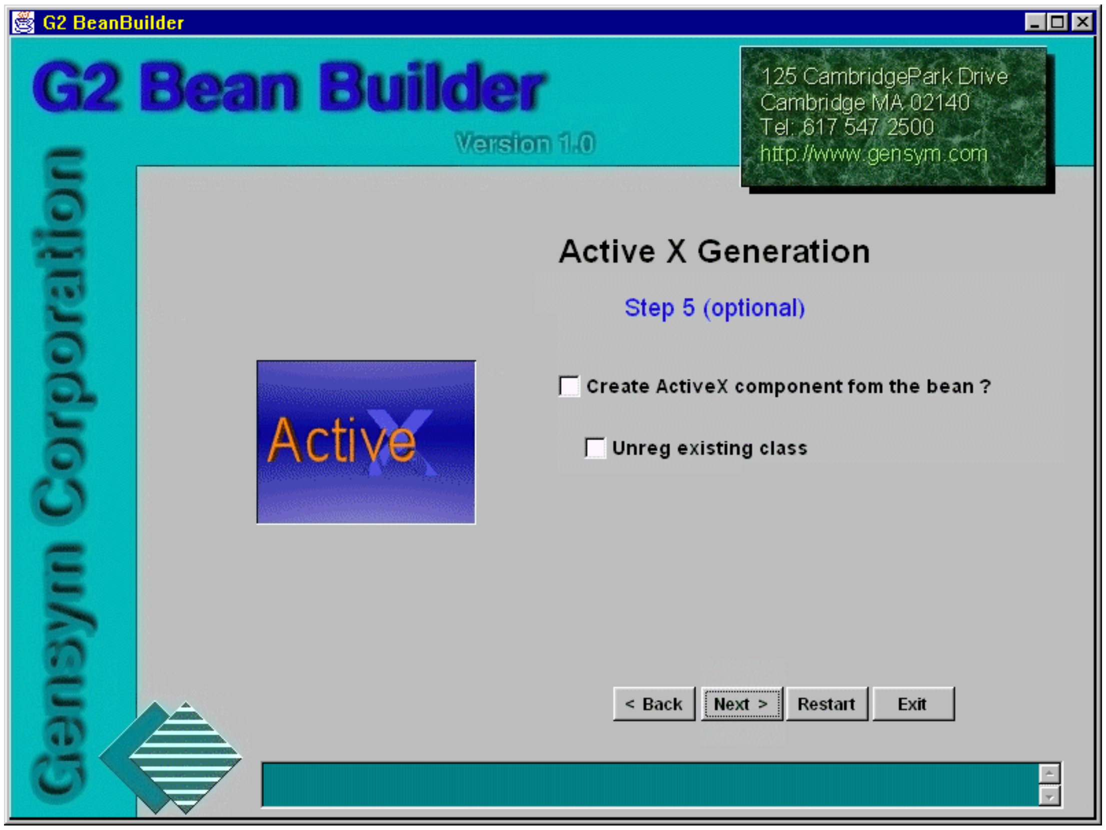
The path component of this field defaults to the path indicated by the value of the `com.gensym.class.user.repository` property with `\jars` appended. This property is defined in the JavaLink properties file, `.com.gensym.properties`. If you started the G2 Bean Builder with the `-dir` command-line option, as described in Using the Command-Line Options, the path component of the Dir field defaults to the JAR file directory you specified in the command-line option.

The file name component of this field defaults to the name of the G2 class with `Bean.jar` appended to the name. If you specified the `-jarfile` command-line option, the file name component of the Dir field defaults to the file name you specified in the command-line option.

To specify a new file name, click the button next to the text field to display a file dialog from which to choose the directory location of the JAR file to create.

ActiveX Generation Panel

The Active X Generation panel allows you to generate an ActiveX control from the Bean, in addition to creating the JAR file:



Active X Generation

Step 5 (optional)

☐ Create ActiveX component fom the bean ?

☐ Unreg existing class

This panel has two specific controls:

- Create ActiveX component from the bean? checkbox
- Unreg existing class checkbox

Configure these options, then click the Next button to advance to the Summary panel.

Creating an ActiveX Component from the Bean

This checkbox, if checked, causes the G2 Bean Builder to generate an ActiveX component from the Bean.

This option defaults to false, unless you started the G2 Bean Builder with the *`-activex`* command-line option, as described in Using the Command-Line Options.

Unregistering Existing Class

This checkbox, if checked, determines how the G2 Bean Builder deals with the existence of duplicate ActiveX controls:

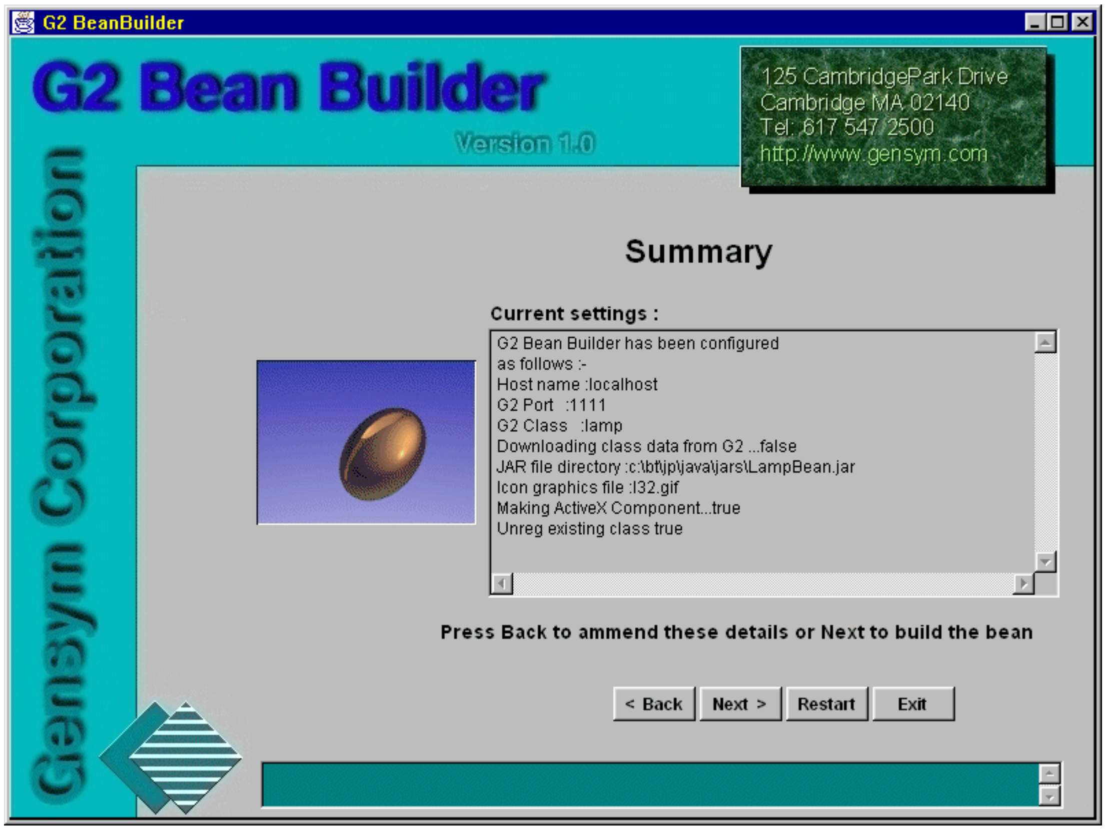
- If an ActiveX component of the same name as currently registered exists, then the G2 Bean Builder unregisters it first.
- If an ActiveX component of the same name has been registered previously, Sun's ActiveX Packager uses the existing GUID of the previous registration.

For the location of the ActiveX files, see Installing the G2 Bean Builder.

This option defaults to false, unless you started the G2 Bean Builder with the *`-unreg`* command-line option, as described in Using the Command-Line Options.

Summary Panel

The Summary panel displays all of the values you entered in each of the previous panels:



```
G2 Bean Builder has been configured
as follows :-
Host name :localhost
G2 Port  :1111
G2 Class  :lamp
Downloading class data from G2 ...false
JAR file directory :c:\bt\jp\java\jars\LampBean.jar
Icon graphics file :l32.gif
Making ActiveX Component...true
Unreg existing class true
```

At this point, you do not need to configure any more information for the utility to begin building the requested Java Bean. To change any of the values you entered, click Back.

Clicking the Back button at this panel returns to one of two previous panels:

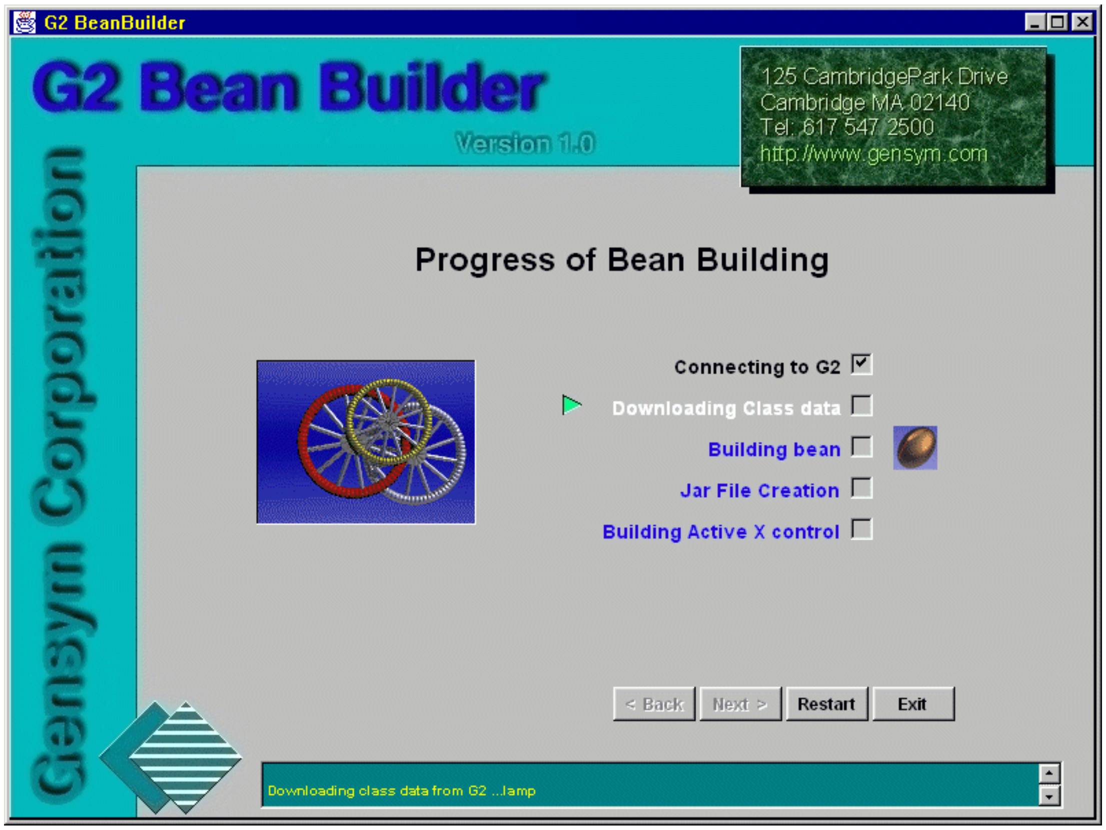
- If the GUI is being run on a Windows platform with Sun's Java Plug-in installed, clicking Back returns to the ActiveX Generation panel.
- Otherwise, clicking the Back button returns to the Bean Build Information panel.

Verify the current settings, then click Next to advance to the Progress of Bean Building panel where the G2 Bean Builder immediately starts creating the Bean.

Note The summary information pertaining to ActiveX is only visible if the G2 Bean Builder is running on a Windows platform with Sun's Java Plug-in installed and you selected the option to create an ActiveX component on the ActiveX Generation panel.

Progress of Bean Building Panel

The Progress of Bean Building panel displays progress as the bean is created:



Progress of Bean Building

Connecting to G2 ✓
Downloading Class data ☐
Building bean ☐
Jar File Creation ☐
Building Active X control ☐

Downloading class data from G2 ...lamp

This panel has the following specific controls, which indicate the progress of each stage of building the Bean:

- Connecting to G2
- Downloading Class data
- Building bean
- Jar File Creation
- Building Active X control

As each stage begins, the color of the checkbox label changes and the status bar at the bottom updates. As each stage is completed, the color changes again and the checkbox representing that stage is checked, if it completed successfully. The example above indicates that the G2 Bean Builder successfully connected to G2 and that it is currently downloading class data, as indicated by the marker.

During the bean building process, both the Back and Next navigation buttons are disabled.

When the download process is complete, the system automatically advances to the Finished panel.

Connecting to G2

During this stage, the G2 Bean Builder attempts to establish a connection with the specified G2. If the connection attempt fails, the error is reported to the wizard and no further building takes place.

Downloading Class Data

Once a connection has been established, the G2 Bean Builder attempts to extract information regarding the target G2 class.

If the specified class is not found, the application informs the user and downloading stops. Otherwise, all of the relevant data is obtained from the target G2 for G2 Bean Builder to complete its task.

Building the Bean

During this stage, the G2 Bean Builder creates and compiles a number of Java files to create the Bean. When the Bean is used in an IDE, the object has an appearance. The G2 Bean Builder either automatically extracts the G2 iconic representation for the object class and uses it for the Java Beans representation or uses the specified icon. You can see a preview of the Bean's icon in the icon preview next to the checkbox associated with this stage.

Jar File Creation

During this stage, the G2 Bean Builder creates all the relevant files and places them in a JAR file.

Building the ActiveX Control

During this phase, the G2 Bean Builder creates an ActiveX control from the Bean. This process leaves the Bean untouched; however, it adds a number of OLE Java bridge files to the JAR file.

This phase places the `.reg` and `.tlb` files in the following location:

jar-directory`\activex`

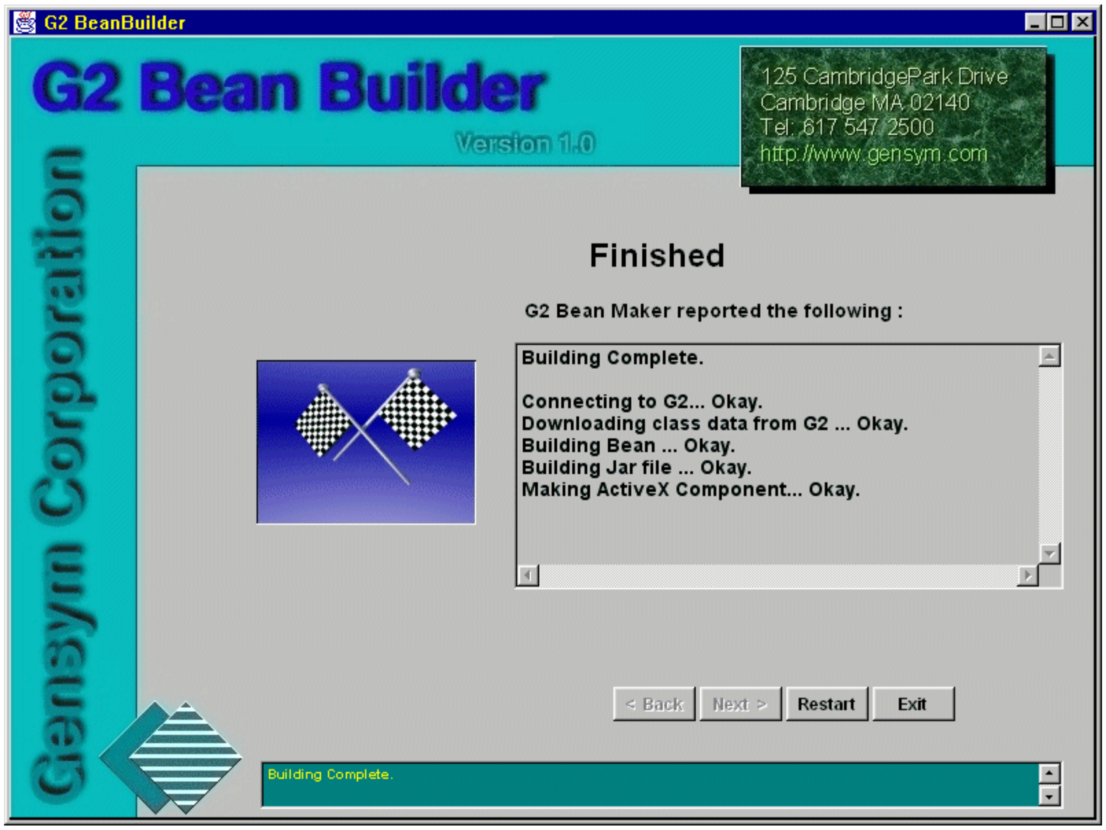
For example:

g2-install-dir`\javalink\classes\jars\activex`

Note The creation of an ActiveX control is only visible if the G2 Bean Builder is running on an Windows platform with Sun's Java Plug-in installed and you selected the option to create an ActiveX component on the ActiveX Generation panel.

Finished Panel

The Finished panel displays a report of the build, detailing the results of the build:



G2 Bean Maker reported the following :

```
Building Complete.

Connecting to G2... Okay.
Downloading class data from G2 ... Okay.
Building Bean ... Okay.
Building Jar file ... Okay.
Making ActiveX Component... Okay.
```

Building Complete.

Note The result of making an ActiveX Component is only visible if the G2 Bean Builder is running on a Windows platform with Sun's Java Plug-in installed and you selected the option to create an ActiveX component on the ActiveX Generation panel.

Using the Command-Line Options

Describes how to use the G2 Bean Builder wizard.

Introduction

You can start the G2 Bean Builder by using one or more of the following command-line options:

- *-host*
- *-port*
- *-class*
- *-force*
- *-iconfile*
- *-notg2icon*
- *-dir*
- *-jarfile*
- *-activex*
- *-unreg*
- *+g*
- *+t*
- *+v*

-host

Specifies the name or IP address of the machine on which the target G2 is running. If you do not use this command-line option, the utility uses localhost as the default value.

To use the `-host` option:

➔ *java com.gensym.beanbuilder.G2BeanBuilder*
   -host [*machine-name* | *IP-address*]

where *machine-name* and *IP-address* can be a machine name or an IP address. For example:

```
java com.gensym.beanbuilder.G2BeanBuilder
   -host mc1
```

or

```
java com.gensym.beanbuilder.G2DownloadInterfaces
   -host 1.2.3.4
```

If you use this command-line option when using the wizard, then the host you enter appears automatically in the G2 Connection Details panel, as described in Specifying the Host.

-port

Specifies the port number of the machine on which the G2 process is running. If you do not use this command-line option, the utility uses port 1111 as the default value.

To use the `-port` option:

➔ *java com.gensym.beanbuilder.G2BeanBuilder*
   -port port-number

where *port-number* is the G2 port number. For example:

```
java com.gensym.beanbuilder.G2BeanBuilder
    -port 1112
```

If you use this command-line option when using the wizard, then the port number you enter appears automatically in the G2 Connection Details panel, as described in Specifying the Port.

-class

Specifies the class for which you wish to create a Bean.

To use the `-class` option:

➔ *java com.gensym.beanbuilder.G2BeanBuilder*
   -class classname

where *classname* is the class from which to create a Bean. When specifying *classname* and not using the GUI, you must use all capital letters and include hyphens, unless the G2 class explicitly contains lower-case letters. When specifying the *-classes* command-line option, separate the class names with spaces. For example:

```
java com.gensym.beanbuilder.G2BeanBuilder
    -class MY-CLASS
```

If you use this command-line option when using the wizard, then the class name you enter appears automatically in the G2 Class Name panel, as described in Entering the G2 Class.

If you are not using the graphical user interface, this option is required when launching the G2 Bean Builder, so that the utility knows which class to use to create the Bean.

-force

Specifies whether to force an interface download. The default for this option is false.

To use the `-force` option:

➔
```
java com.gensym.beanbuilder.G2BeanBuilder
    -force
```

If you use this command-line option when using the wizard, then the force download checkbox is checked automatically when the G2 Class Name panel appears, as described in Forcing an Interface Download.

-iconfile

Specifies a 32x32 color image to represent the Bean when viewing it in an IDE. The default for this flag is *`I32.gif`*, which is located in the G2 JavaLink *`beanbuilder.jar`* file.

To use the `-iconfile` option:

➔
```
java com.gensym.beanbuilder.G2BeanBuilder
    -iconfile filename
```

where *filename* is the name of the image file to use for the Bean's icon. For example:

```
java com.gensym.beanbuilder.G2BeanBuilder
    -iconfile c:\images\myimage.gif
```

If you use this command-line option when using the wizard, then the file name you specify appears automatically in the Iconic Representation panel, as described in Entering the File Name.

-notg2icon

Specifies that the default icon to use to represent the Bean is *`I32.gif`*, located in the G2 JavaLink *`beanbuilder.jar`* file, or the image file specified by the *`-iconfile`* command-line option.

To use the `-notg2icon` option:

➔
```
java com.gensym.beanbuilder.G2BeanBuilder
    -notg2icon
```

If you use this command-line option when using the wizard, then the Use G2 icon for bean checkbox is set to false in the Iconic Representation panel, as described in Entering the File Name.

-dir

Specifies the JAR file home directory to use when generating the Bean. The default value is the path indicated by the value of the `com.gensym.class.user.repository` property with `\jars` appended. This property is defined in the JavaLink properties file, `.com.gensym.properties`.

To use the `-dir` option:

➔
```
java com.gensym.beanbuilder.G2BeanBuilder
    -dir directory
```

where *directory* is the path name to use for the JAR file. For example:

```
java com.gensym.beanbuilder.G2BeanBuilder
    -dir c:\jars\
```

If you use this command-line option when using the wizard, then the directory you specify appears automatically in the Bean Build Information panel, as described in Entering the File Name.

-jarfile

Specifies the JAR file name to use when generating the Bean. The default value is the name of the G2 class with `Bean.jar` appended to the name.

To use the `-jarfile` option:

➔
```
java com.gensym.beanbuilder.G2BeanBuilder
    -jarfile filename
```

where *filename* is the file name to use for the JAR file. For example:

```
java com.gensym.beanbuilder.G2BeanBuilder
    -jarfile MyBean.jar
```

If you use this command-line option when using the wizard, then the directory you specify appears automatically when the Bean Build Information panel appears, as described in Entering the File Name.

-activex

On Windows platforms only, specifies whether to generate an ActiveX component from the Bean. The default value is false.

To use the `-activex` option:

➔
```
java com.gensym.beanbuilder.G2BeanBuilder
    -activex
```

If you use this command-line option when using the wizard, then the Create ActiveX component from the bean? checkbox is checked automatically in the Active X Generation panel, as described in Creating an ActiveX Component from the Bean.

-unreg

On Windows platforms only, specifies whether to unregister an existing ActiveX control if it is already registered. The default value is false.

To use the `-unreg` option:

➔
```
java com.gensym.beanbuilder.G2BeanBuilder
    -activex -unreg
```

If you use this command-line option when using the wizard, then the Unreg existing class checkbox is checked automatically in the Active X Generation panel, as described in Creating an ActiveX Component from the Bean.

+g

Launches G2 Bean Builder with the graphical user interface wizard. When you launch G2 Bean Builder with no command-line option, you get the GUI, by default. When you launch it with the `-class` command-line option, you do not get the GUI, unless you use the `+g` command-line option.

+t

Sets the trace output for the G2 Bean Builder utility to true. By default, tracing is not enabled. The trace output can provide information about problems encountered while downloading G2 classes.

+v

Runs the G2 Bean Builder verification test. By default, the verification test is not run when you launch the utility. For more information, see Checking the Installation.

Using G2 Bean Builder Beans

Describes how to use G2 Bean Builder beans as Java Beans and as ActiveX controls.

Introduction

The G2 Bean Builder is a tool for extracting Java Beans from existing G2 class objects as JAR files and ActiveX controls.

This chapter illustrates the use of the JAR file by using Sun Microsystem's Beanbox, while it illustrates the use of the ActiveX control by using Microsoft's Visual Basic.

Neither of these examples is intended to be an in-depth guide to using third-party software. For full details on their use, please refer to the relevant product documentation.

Creating the G2 Class

To illustrate the use of the G2 Bean Builder, first you must create a G2 object from which to generate the Bean. This example uses the test class detailed below because it has a variety of attribute types.

To create a test G2 class:

1. In G2, create an object definition for the class a-test-class with the following class-specific attributes:

   ```
   a-int is an integer, initially is 34;
   a-float is a float, initially is 3.4e6;
   a-text is a text, initially is "A TEST STRING";
   a-symbol is a symbol, initially is hi-there;
   a-truth-value is a truth-value, initially is false;
   a-sequence is a sequence, initially is sequence (1, 2.3, the symbol fred,
       "This text belongs to this sequence",
   structure (fred: 1, ethal: "ethals structure text"));
   a-structure is a structure, initially is structure (attone: the symbol hi-there,
   atttwo: "A Very complex object is this")
   ```

2. Create an instance of a-test-class.
3. Name that instance a-test-class-instance.

Creating the Bean

The following instructions assume that the G2 Bean Builder is running on a Windows platform with Sun's Java Plug-in installed. The G2 Bean Builder automatically reconfigures itself on systems where ActiveX is not available so that ActiveX-related functions are not visible.

To create the test Bean:

1. Start the G2 Bean Builder, using one of the following techniques:
   - ➔ Enter the following command at the command line:

     ```
     java com.gensym.beanbuilder.G2BeanBuilder
     ```

   - ➔ From the Start menu, choose the following from your Gensym G2 program group:

     G2 JavaLink > G2 Bean Builder

2. Configure the G2 Bean Builder to connect to your G2.

   For details, see Configuring the G2 Connection Details.

3. Configure the class to download to be A-TEST-CLASS.

4 Accept the default icon representation.

5 Accept the default JAR file location.

6 Instruct the G2 Bean Builder to create an ActiveX component, as needed.

7 Continue with the G2 Bean Builder prompt to create the Bean.

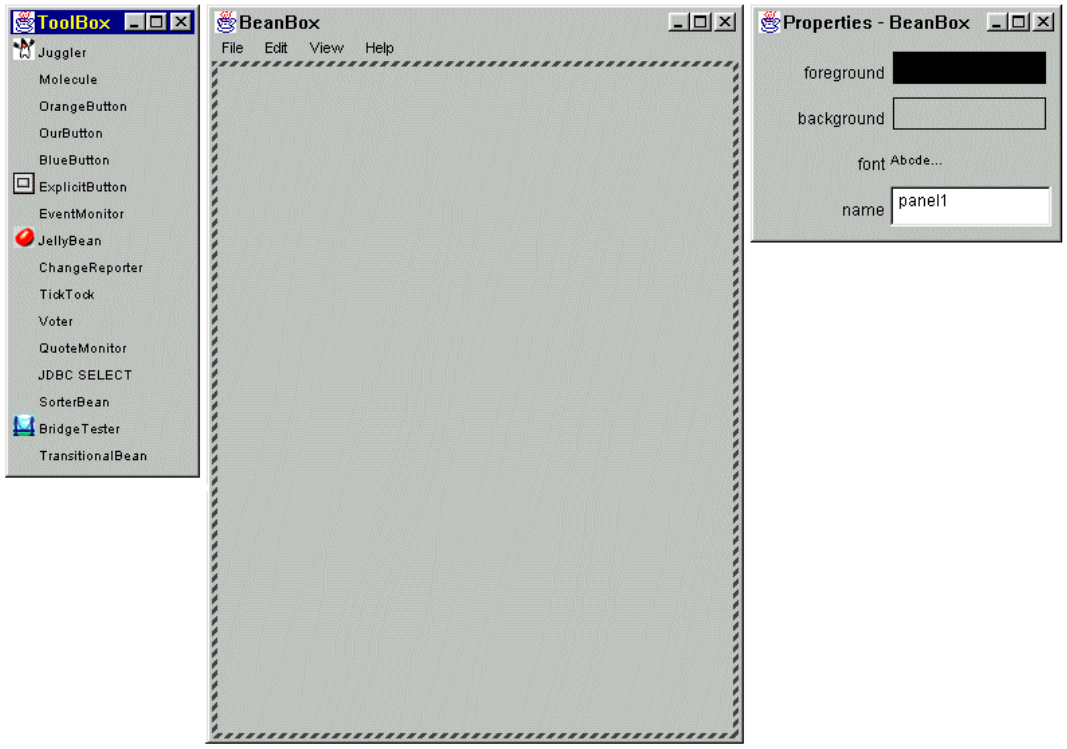
The G2 Bean Builder creates a Java Bean called *ATestClassBean* found in *ATestClassBean.jar*. If an ActiveX component was created, it is registered as "ATestClassBean Bean Control."

Using the Bean in the Beans Development Kit (BDK)

Sun Microsystem's Beans Development Kit (BDK) is a pure Java application, whose only dependency is the Java Development Kit (JDK) 1.2.1 or later. The BDK provides:

- Support for the JavaBeans APIs.
- A test container, called the "BeanBox," used to test Bean behavior.
- Sample Beans complete with their source code.
- The JavaBeans Specification.
- A tutorial.

This figure shows the BDK:



The above figure shows, from left to right:

- The ToolBox, a palette of beans.
- The BeanBox, an area for positioning beans and configuring their events.
- The property sheet, an area for configuring the properties of beans.

Viewing the Bean in the Bean Box

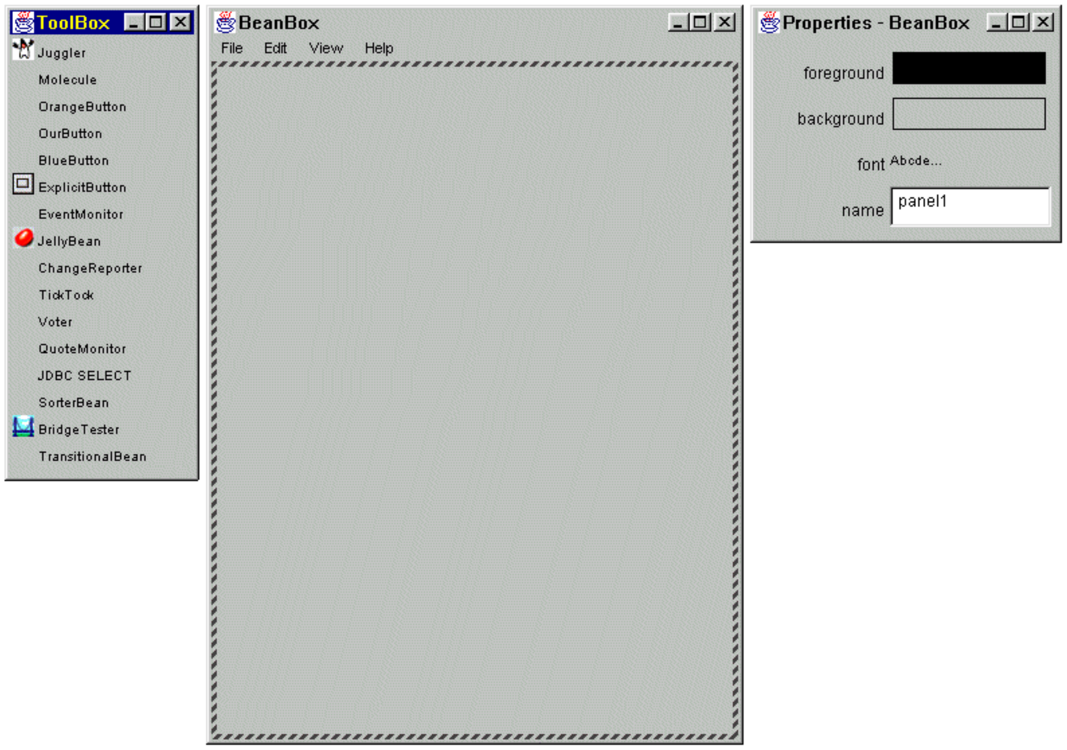
To view the Bean in the Bean Box, you must load its associated JAR file.

To view the Bean in the Bean Box:

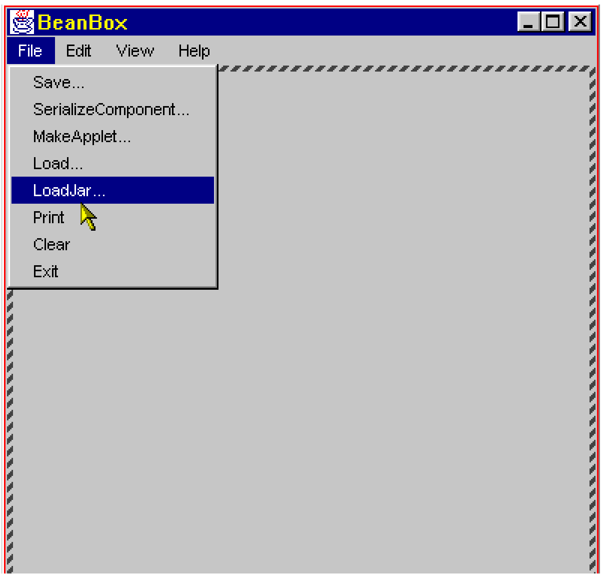
1 In a command window, change to the BDK BeanBox product directory and enter this command:

```
run
```

The BDK BeanBox appears:

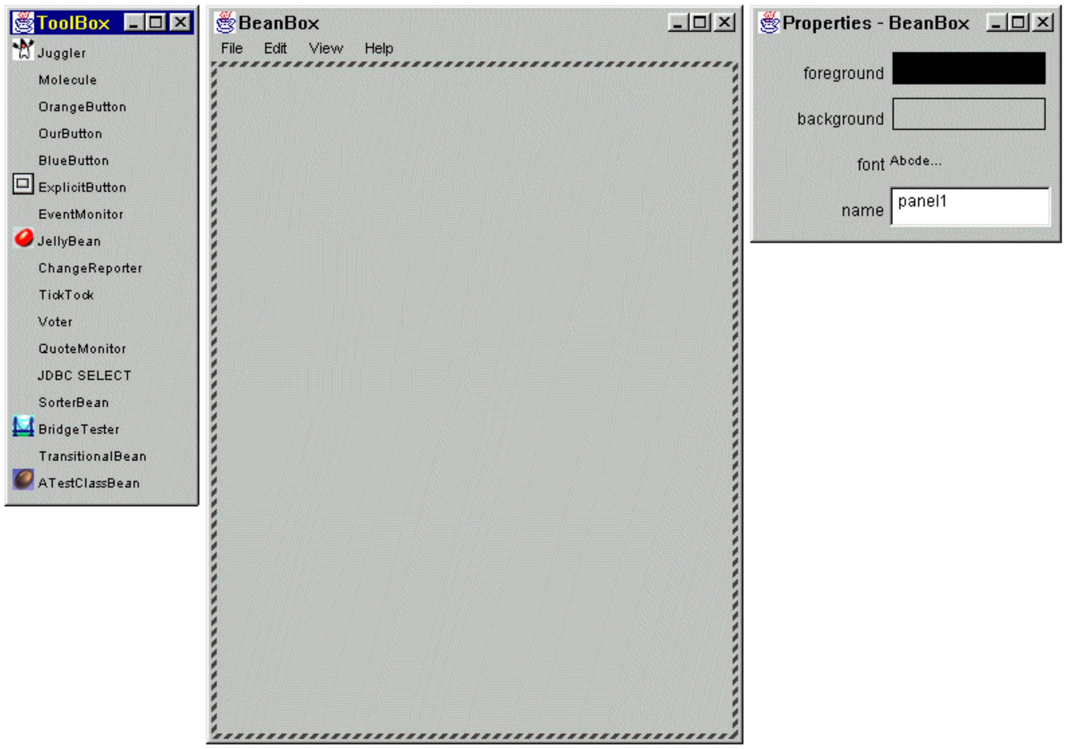


2 Choose LoadJar from the File menu on the main BeanBox composition window:



3 Use the dialog box to select and load `ATestClassBean.jar`.

When loaded, the Bean is visible at the end of the ToolBox palette:



Creating an Instance of the Bean

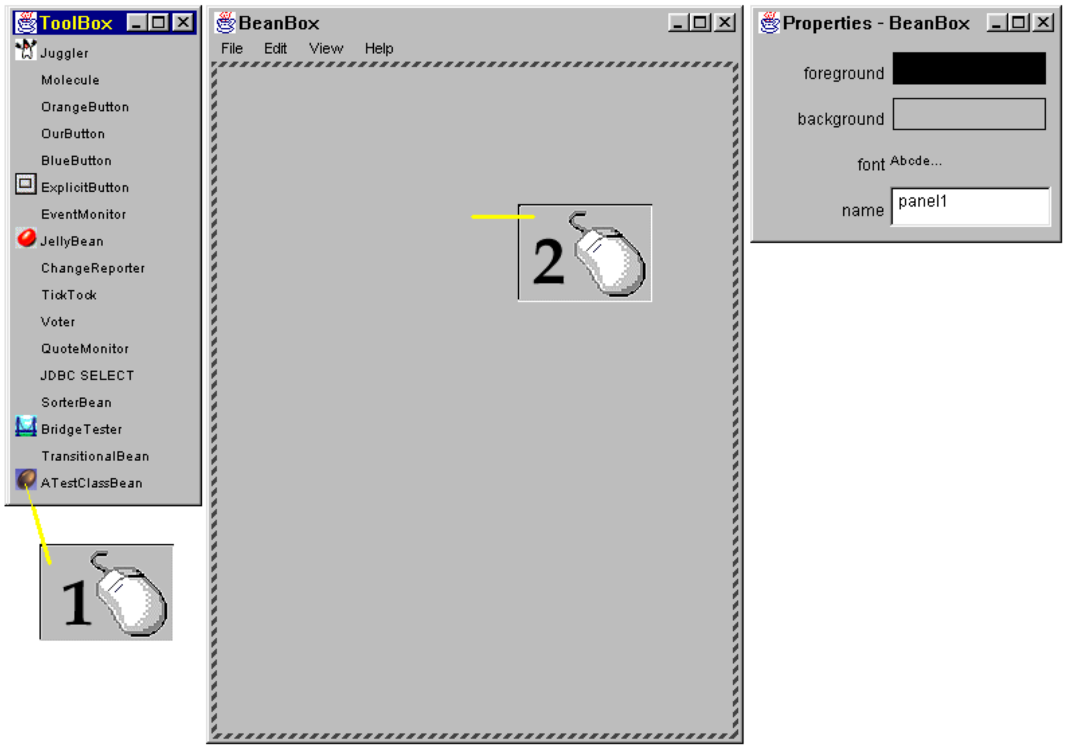
To create an instance of the Bean:

1 Click the ATestClassBean icon on the ToolBox palette.

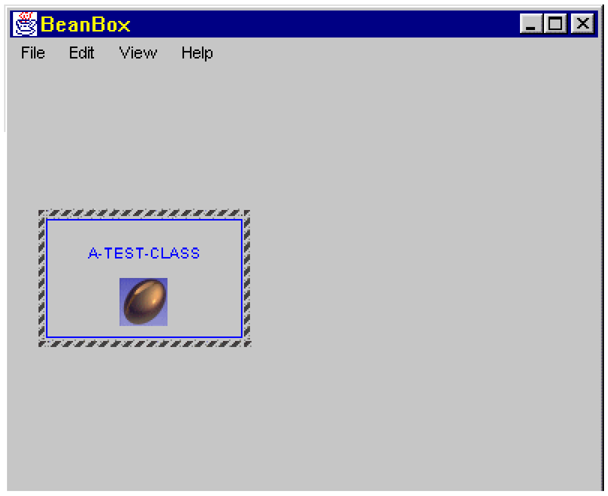
The cursor changes to a +.

2 Click anywhere in the BeanBox composition window to create the Bean.

This figure shows the two steps required to create an instance of the Bean:

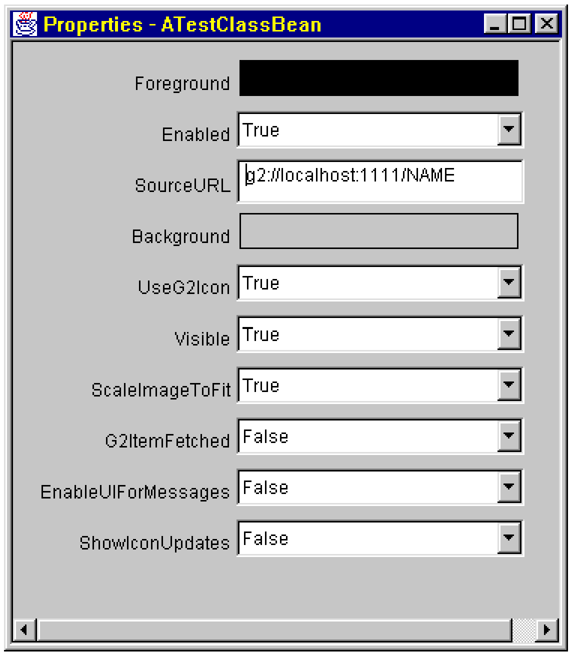


In this figure, the ATestClassBean is surrounded by a box indicating that it is currently selected:



Configuring the Bean

This figure shows the property sheet for ATestClassBean, which includes the SourceURL and G2ItemFetched properties, described below:



Foreground	
Enabled	True
SourceURL	g2://localhost:1111/NAME
Background	
UseG2Icon	True
Visible	True
ScaleImageToFit	True
G2ItemFetched	False
EnableUIForMessages	False
ShowIconUpdates	False

The SourceURL Property

The SourceURL property specifies the G2 object that the selected Bean represents.

The URL references an existing G2 object resident in a G2 running on the indicated port of a specified machine. It takes the form:

```
g2://<host>:<portnumber>/<G2objectname>
```

For example:

```
g2://localhost:1111/A-TEST-CLASS-INSTANCE
```

The G2ItemFetched Property

The G2ItemFetched property is a boolean flag to indicate that the Bean is to fetch the item specified by the SourceURL.

Assuming the SourceURL is set to a valid G2 object, setting the flag to true causes the Bean to connect to the specified G2 and probe the indicated object to obtain all of its properties, which are defined in the class-specific attributes of the object class definition.

Note G2 must be running for the item to be fetched.

The property sheet displays only properties of certain types, for example, `String`, `int`, `float`. For more information, see the BDK user documentation from Sun.

For this reason, the definition of the G2 attributes determines whether or not the corresponding Bean property appears in the property sheet. You must define the G2 attribute properly for it to appear in the property sheet.

For example, the following G2 attribute definition makes the property accessible from the BDK's BeanBox property sheet as a boolean:

```
g2-item-fetched is a truth-value, initially is false
```

The following G2 attribute definition, exported as an object because this G2 attribute accepts any type or item, creates a Java Object rather than a native type, so it does not appear in the property sheet:

```
x-offset initially is 23
```

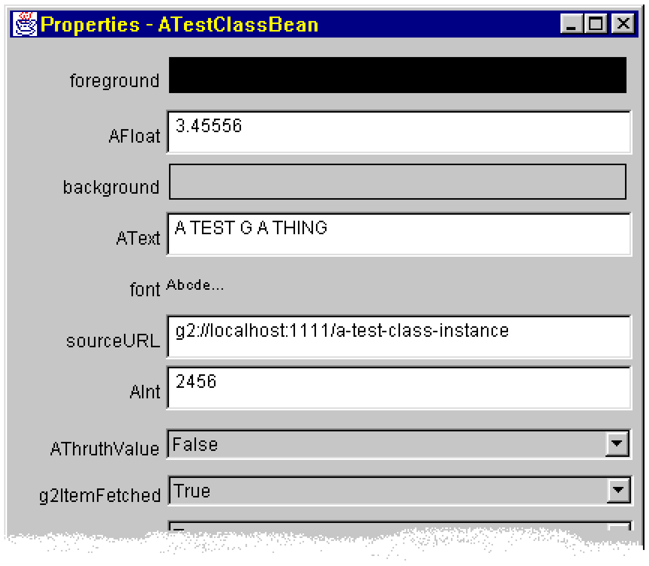
To access this property via the property sheet, it should be redefined to be "strongly" typed as an integer. G2 can then export the definition for this attribute as a Java `int` rather than as a generic object.

For example:

```
x-offset is an integer, initially is 23
```

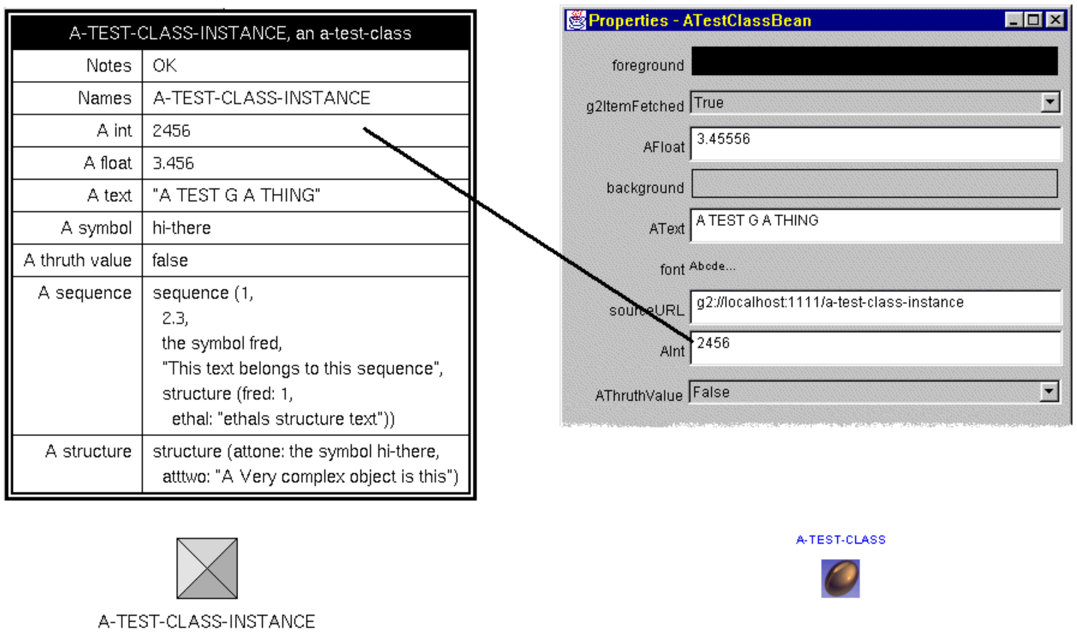
Note When a G2 class definition is changed, it should be reexported to Java, using the G2 Bean Builder or G2 DownloadInterfaces utilities, to ensure that the new definition is available to Java.

Once a correct SourceURL is specified and G2ItemFetched is set to true, then the exported Bean connects to a-test-class-instance in the G2 at `localhost:1111`. Reselecting ATestClassBean in the BeanBox automatically displays the current values of the exported attributes of a-test-class-instance, as follows:


Properties - ATestClassBean

foreground	■
AFloat	3.45556
background	
AText	A TEST G A THING
font	Abcde...
sourceURL	g2://localhost:1111/a-test-class-instance
AInt	2456
AThruthValue	False
g2ItemFetched	True

At this point the Bean is "aware" of its specified G2 object and vice versa, as this figure shows:



A-TEST-CLASS-INSTANCE, an a-test-class	
Notes	OK
Names	A-TEST-CLASS-INSTANCE
A int	2456
A float	3.456
A text	"A TEST G A THING"
A symbol	hi-there
A thruth value	false
A sequence	sequence (1, 2.3, the symbol fred, "This text belongs to this sequence", structure (fred: 1, ethal: "ethals structure text"))
A structure	structure (attone: the symbol hi-there, atttwo: "A Very complex object is this")

Properties - ATestClassBean

foreground	■
g2ItemFetched	True
AFloat	3.45556
background	
AText	A TEST G A THING
font	Abcde...
sourceURL	g2://localhost:1111/a-test-class-instance
AInt	2456
AThruthValue	False

If you change an attribute value of the G2 object, the change appears in the property sheet of the Bean and vice versa.

Note It is necessary to reselect ATestClassBean in the BeanBox window for the property sheet to be updated when attributes are changed in G2.

The EnableUIForMessages Property

The EnableUIForMessage boolean property controls whether or not the Bean reports exceptions to the user via a GUI dialog box. Some IDE's such as Visual Basic do not catch and report exceptions. In such environments, an exception may at best go unreported, or in the worst case hang the system. Java IDE's tend to catch and report exceptions. If this property is set to true and the IDE supports it, then the Bean causes the display of a dialog box containing the exception message, as well as throwing the exception.

The UseG2Icon Property

The UseG2Icon boolean property controls whether or not the icon displayed by the Bean is the same as the G2 icon for the item specified by SourceURL or is the image that was specified as the Beans icon during the construction of the Bean.

The ScaleImageToFit Property

The ScaleImageToFit boolean property controls whether or not the icon image for the Bean, as controlled by the UseG2Icon property, is scaled to fit within the bounds of the Bean. If set to true, the image is scaled in size so it fits within the bounds of the Bean. If set to false and the UseG2Icon property is true, then the image is displayed within the bounds of the Bean starting at 0,0 (top left). If the UseG2Icon property is set false, the image is displayed centrally within the bounds of the Bean. If the image is larger than the bounds of the Bean, it is cropped.

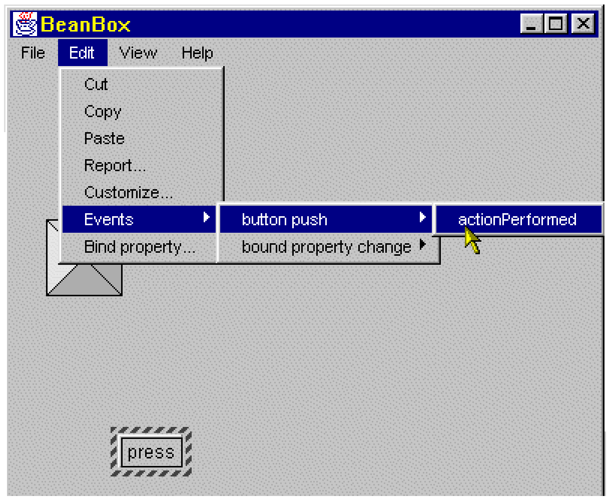
The ShowIconUpdates Property

The ShowIconUpdates boolean property controls whether or not the icon of the Bean reflects changes that take place in the G2 icon for the specified item.

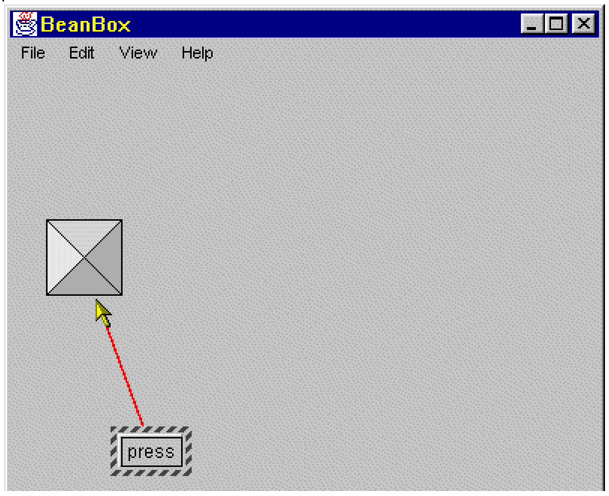
Note This property is applicable only if the UseG2Icon property has been set to true and Telewindows2 Toolkit has been installed.

Interacting with Other Beans

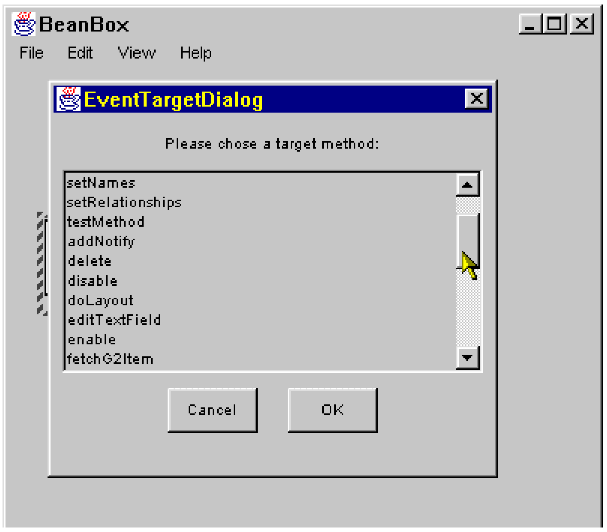
The following figure shows the BeanBox composition window with a BlueButton added from the Toolbox palette and the actionPerformed button push event has been selected from the Edit menu:



Here the action is being "tied to" the test Bean:



When the two Beans are connected, the EventTargetDialog window appears and displays the available methods of the test Bean:



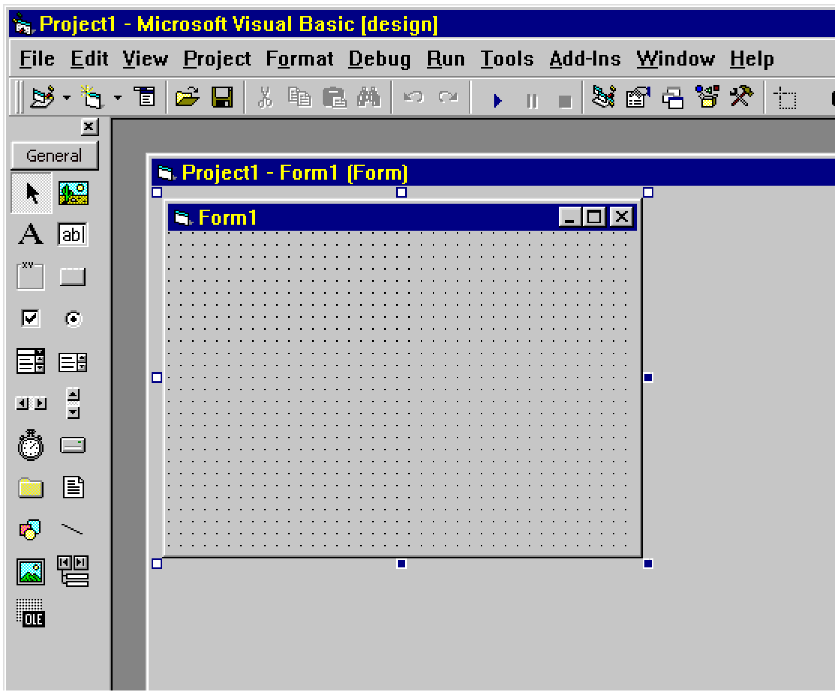
Events for the object defined in G2 are also available in Java and so appear in this list. For more information, see the chapter on writing G2 Bean events in the *G2 JavaLink User's Guide*.

Selecting one of these events causes the selected method to be called when the button is pressed.

For a detailed example, see the BDK tutorial supplied with BDK1.1.

Using the Bean as an ActiveX Control in Visual Basic

The following figure shows a Visual Basic 5.0 project:


Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Tools Add-Ins Window Help
General
Project1 - Form1 (Form)
Form1

The palette on the left holds a number of standard Visual Basic controls.

Importing Beans into Visual Basic

To use the Bean as an ActiveX control, you import the Bean as a custom control into Visual Basic.

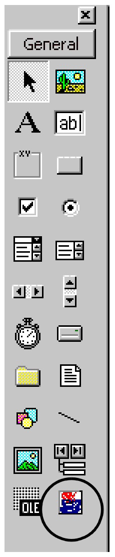
To import the Bean as an ActiveX control into Visual Basic:

1 From Visual Basic, choose Components from the Project menu:

2 Select ATestClassBean Bean Control.

3 Accept the dialog.

After importing the component, ATestClassBean appears on the Toolbox, as follows:



You can work with the imported Bean in the same way as you would with any other ActiveX component.

Creating an Instance of the ActiveX Control

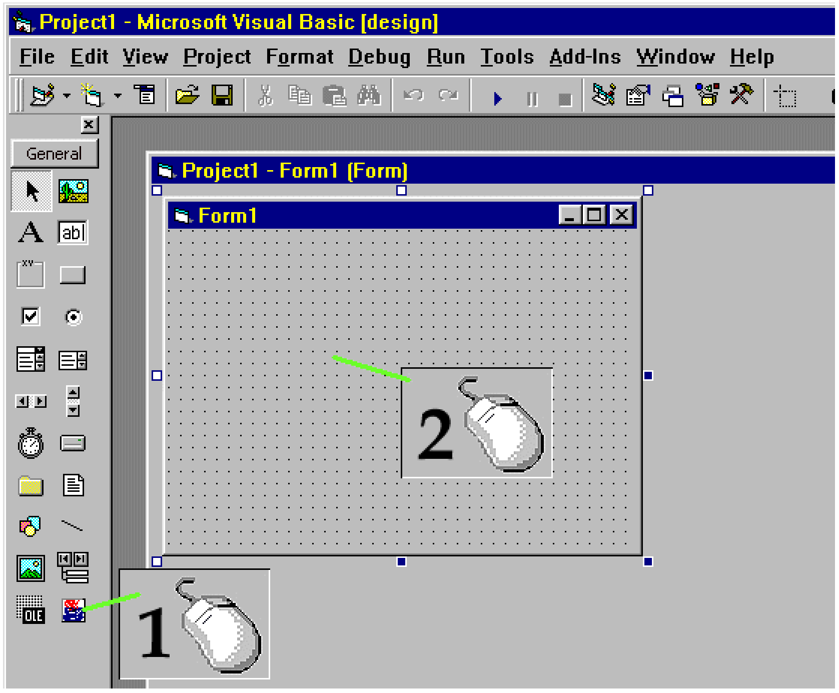
To create an instance of the ActiveX control:

1 Click the ATestClassBean icon in the Toolbox.

The cursor changes to a + when positioned over the form.

2 Draw a box on the Form to create an instances of the control.

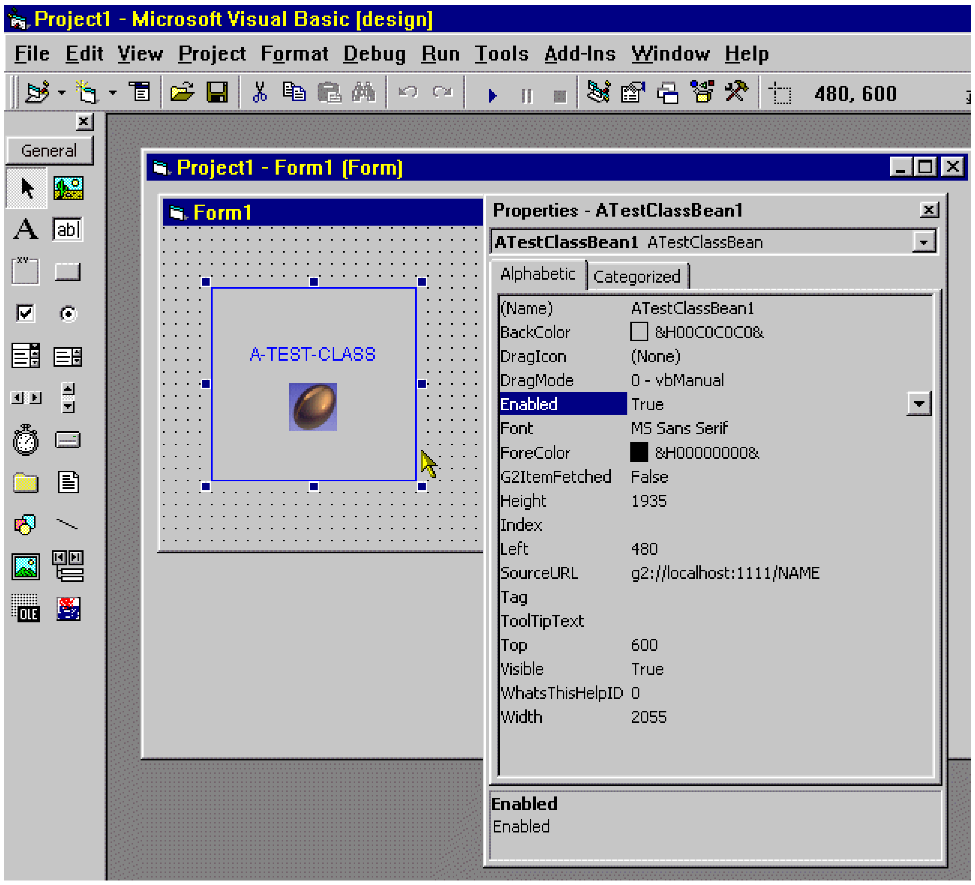
This figure shows the two steps required to create an instance of the control:


Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Tools Add-Ins Window Help
General
Project1 - Form1 (Form)
Form1
2
1

If Visual Basic reports that the generated G2 ActiveX Bean control has not been registered, when a user attempts to create and draw the control on a VB form, the most likely cause is that the file *beans.ocx* is not found.

Note The Packager uses a common control file called *beans.ocx* that is normally placed in *c:\Program Files\JavaSoft\JRE\1.3\bin\*, assuming the Java Plug-in is installed at *c:\Program Files\JavaSoft\JRE\1.3\*. This file must be copied to any ActiveX directory created by the G2 Bean Builder, or the generated ActiveX controls fail to function.

This figure shows A-TEST-CLASS Bean on the Form and its properties as displayed in the Properties window:


Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Tools Add-Ins Window Help
480, 600
General
Project1 - Form1 (Form)
Form1
A-TEST-CLASS
Properties - ATestClassBean1
ATestClassBean1 ATestClassBean
Alphabetic
Categorized

(Name)	ATestClassBean1
BackColor	&H00C0C0C0&
DragIcon	(None)
DragMode	0 - vbManual
Enabled	True
Font	MS Sans Serif
ForeColor	&H00000000&
G2ItemFetched	False
Height	1935
Index	
Left	480
SourceURL	g2://localhost:1111/NAME
Tag	
ToolTipText	
Top	600
Visible	True
WhatsThisHelpID	0
Width	2055

Enabled
Enabled

For information about configuring the properties of the Bean, see Configuring the Bean.

Index