



G&R

TM

GARGEN

***Java bean
generator for
encapsulating
mainframe
transactions
as Java beans***

<http://www.glink.com/glinkj/>

G&R
GALLAGHER
ROBERTSON

Microsoft, Windows, MS, MS-DOS are registered trademarks of Microsoft Corp.

IBM and PC are registered trademarks of IBM Corp.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Any other product names are trademarks of their respective owners.

Version 6.7

© Gallagher & Robertson as 1990-2006

All Rights Reserved

GALLAGHER & ROBERTSON AS, Kongens gate 23, N- 0153 Oslo, Norway

Tel: +47 23357800 • Fax: +47 23357801

www: <http://www.gar.no/glinkj/>

e-mail: glinkj@gar.no



Contents

Scope of the product	1
Introduction to Gargen	1
Using Gargen	1
Gargen and Glink	2
The Glink API	3
Licensing issues	4
Delivery	5
Installation	5
Configuration	5
Configuring Glink	5
Configuring Gargen	6
Working with Gargen.....	7
Create, modify or test a bean	7
Name your Bean	7
Name your Enterprise JavaBean	7
Test or modify your bean	8
Testing your bean	8
Log on to Glink	9
Select a session	9
The screen definition phase	10
Processing a screen	11
The generation phase	14
Generate your bean	14
Generate your Enterprise JavaBean	15
Connector Interface for Enterprise JavaBeans	15
Generate your bean, more information	16
Source files created by Gargen	17
Files created for Standard JavaBeans	17
Files created for Enterprise JavaBeans	18
Templates	19
Standard JavaBeans	19
Enterprise JavaBeans	21
Using your bean	23
Compile the source files	23
Deploy your bean	23



Scope of the product

Introduction to Gargen

Gargen is an application component generator. Its purpose is to allow you to step through an interaction with an existing application on a mainframe system, and then automatically generate a Java component (bean) that is able to repeat that interaction when used by a Java application.

Most existing mainframe applications are designed to communicate with some kind of terminal. Gargen uses a development copy of Glink for Java Professional Edition to do the communications, using the appropriate terminal type. Gargen drives this copy of Glink using the Glink for Java Application Programming Interface.

Using Gargen

The first phase of Gargen usage is the dialog definition phase, where you step through a complete functional interaction with a mainframe. At each dialog step Gargen allows you to identify the output screen for verification of the dialog step, to mark and give names to areas of the screen that you want as output from your bean, to give names to those fields that are input parameters to your bean and to enter input for the next dialog step. Data you enter into unnamed fields is regarded as constant, and is used every time you use your bean.

For simplicity and clarity Gargen drives a visible copy of Glink for Java during the interaction with the mainframe application. It is displayed as a separate pane in the Gargen window. You use the Glink screen for all data entry and retrieval, exactly as if you were using a standard copy of Glink.

In playback mode you can step quickly through the dialog defined in your bean. Gargen will pause at each interaction with the application, and you are able to decide if you want to change recognition criteria for the current screen, collect new output for your bean, change constant input fields or name new input fields that will become bean parameters.

When you have reached the end of your functional interaction, it is very efficient to return the dialog to a known screen (e.g. a menu showing the function that you have encapsulated as a bean), because this will allow your bean to be reused without all the dialog steps necessary to log-on and reach the menu.

Gargen and Glink

Gargen uses the Glink API for host communication and forms processing. You can hand-program your own JavaBeans for host communication using the Glink API.

Programming of host communication JavaBeans can be time-consuming, because of the trivial programmatical details of checking that the expected screen has been received, of selecting the data on the screen that you would like to be an output from your JavaBean and identifying the input that you want to enter, as constants for every execution of the bean, or as variables to be taken from the input parameters to your bean.

You can automate programming of these host communication JavaBeans using Gargen.

Gargen itself is a good example of an application using the Glink API. Gargen is in control of the user interface, but uses the Glink API to drive a copy of Glink to do the interaction with the host application while recording the necessary information that a JavaBean will need in order to repeat the same function.

Gargen allows you to step through a host communications sequence, and generates the source code of a JavaBean or Enterprise JavaBean that uses the Glink API for host communication and forms processing.

The JavaBean that Gargen generates may be used from a Java program on your workstation, from an applet within your browser, from a Java servlet on a Web server or from a Java application on a J2EE compliant Application Server.

Gargen generates an example that shows how to use the JavaBean. Gargen also generates an example of how you could deploy and use the Enterprise JavaBean in an Application Server environment.

The Glink API

Gargen is delivered with a developer's license for the Glink for Java Professional Edition, which includes use of the Glink for Java Application Programming Interface. A Java application using the API can connect to any mainframe supported by Glink, and use Glink's line or screen mode capability to interact with the mainframe applications.

The Glink API is implemented as a set of Java classes. There are two versions of Glink. The Standard Glink for Java classes are in `glink.jar` and an optimized version is in `glinkee.jar`.

To use the Glink API from a Java application, simply add the full path of the `glink.jar` or `glinkee.jar` file to the Java class path.

The optimized API is the same as the standard, except that the screen display calls are ignored, and it can only be used from a Java application where the user interface is provided by the application using the Glink API.

Use the `glink.jar` file when developing an application, even if you supply your own user interface. You can then see the host dialog in a background Glink window, and quickly spot unexpected situations. When the application is completed you can turn off the screen display.

If you have completely replaced the user interface, then you can deploy the optimized API `glinkee.jar` for efficient run time usage.

Gargen is itself an application that uses the Glink API. It uses the `glink.jar` class library so that Gargen can use the Glink screen display, and you can follow the dialog that your bean is exchanging with the mainframe.

Gargen produces source code for a JavaBean that uses the Glink API. The source code may be modified to add extra functionality. The Glink API documentation is found in the HTML file:

```
gargen/help/api/apidoc/index.html
```

Licensing issues

There are three different licenses for use of Glink for Java. The Standard Edition license allows Glink to be used as a terminal emulator that simply displays the screens the mainframe sends (somewhat facelifted). The Glink Professional Edition license enables the Glink API for up to five simultaneous sessions. For heavy-duty server usage the Glink Enterprise Edition license enables use of as many simultaneous sessions as are licensed.

Gargen is itself an application that uses the Glink API, and includes a license enabling the Glink Professional Edition for development purposes.

Gargen produces source code for a JavaBean that uses the Glink API. When the bean is deployed in production, the Glink for Java server used by the production system must be licensed for Glink Professional or Enterprise Edition.

Delivery

Gargen is delivered as a set of InstallAnywhere executables for various platforms.

Installation

The install package can be delivered with or without a Java Virtual Machine (JVM). A JVM 1.4.2 or later is recommended.

In addition to the Gargen application, the install package contains a development copy of Glink for Java Professional Edition, and the Administration program, which you use to configure the host sessions that you will access from Glink.

To install Gargen, just run the install utility. After the installation has completed, insert your `licenses` file in the `gargen\config` directory. Contact your distributor if you don't have a license file.

Configuration

Configuring Glink

Gargen is delivered with a copy of Glink and its administration program. The installation includes sample configurations, and (assuming you have Internet access) you can test these connections immediately after installation, to confirm the delivery. The Glink log-on name and password are both initially set to **admin**.

Your next step is to configure the mainframe sessions you want to access with Gargen. The Glink administration program is used for configuration. You can copy/paste the working configurations that we deliver, modifying them to connect to your own mainframes and applications. You can test your own mainframe configurations using Glink, before you begin to use Gargen. This should save you a great deal of development time.

Gargen is installed locally, and used from your workstation during development. If you are already using Glink for Java and have a Glink for Java server running somewhere, Gargen can be configured to use your existing server to reach the mainframes and applications already configured on your server. Just modify the **configserver** parameter in the `Glink.ini` file.

This presumes that your server is licensed to allow use of the API (Glink for Java Professional or Enterprise Edition).

Configuring Gargen

Gargen itself needs no additional configuration. However, Gargen uses some template files when generating the source for the JavaBeans or Enterprise JavaBeans. It also uses some commands files (Windows) or shell scripts (UNIX) to compile, pack and deploy to generate the JavaBean or Enterprise JavaBean. When you are familiar with Gargen you might want to modify one or more of these template files to suit your environments.

Working with Gargen

Create, modify or test a bean

Name your Bean

Welcome to Gargen. At this point you can choose to generate a new bean by choosing a name for your bean and a package name if desired. If you choose an existing bean name you will be put into playback mode, and when you use the **Next** button you will be given the opportunity to test your bean, or to modify it, by stepping through the existing bean making changes.

There is a drop-down list that allows you to select the type of bean you want to generate. Standard JavaBeans can be used in any workstation or server environment. Enterprise JavaBeans can only be used within a J2EE compliant Application Server on which the G&R JCA compliant Connector is installed. Select the **Enterprise JavaBean** list item for more information on Enterprise JavaBeans.

All the decisions you make here as regards bean name and type can be changed after the bean specification phase, before you generate the bean. It is a simple matter to modify a Standard JavaBean, and generate it once more as an Enterprise JavaBean.

Name your Enterprise JavaBean

An Enterprise JavaBean is a special case of a bean, designed to run under control of an Application Server that complies to Sun's J2EE specification. Enterprise JavaBeans access the mainframe application via a Connector, compliant with Sun's JCA specification. Enterprise JavaBeans generated by Gargen use the G&R Connector.

The dialog definition phase of Gargen is exactly the same as when defining a standard **JavaBean**, but **Enterprise JavaBeans** require additional configuration parameters.

The **EJB's JNDI name** is the name that the bean will be identified by in the Java Naming and Directory Interface.

The **Connector's resource ref name** and the **Connector's real JNDI name** define the Connector Interface.

Select the J2EE compliant application server in the Build target drop-down list. Please contact G&R if your application server is not in the list.

Test or modify your bean

You have selected an existing bean. You can test or modify it. Gargen has requested the properties of your bean to find out which parameters it needs from you in order to test or playback your bean. In either case you must supply values for the input parameters if any. Place the cursor in the value cell and type the input followed by the **Enter** key.

The **Test** button launches a Gargen test application, which calls your bean. It displays the Glink window so you can watch the dialog with the host application. The input is submitted to your bean, and the results returned from your bean are presented in a table, each entry in the format "header" and "value". The headers are the names you gave to the output fields. The values are the content. See the section entitled '**Testing your bean**'.

The **Next** button lets you modify the selected bean. It initiates the bean definition phase in playback mode where Gargen pauses for each screen. You can modify the screen recognition criteria, the named screen output areas, the named input fields or the constant input fields. When you reach the last dialog step defined in the bean, playback mode is reset, and you enter the normal Screen definition phase. You can add new dialog steps, or select the **Done** button.

Testing your bean

If you choose to test your bean, the following information is displayed while the test is running:

Glink is now stepping through the application screens you have encapsulated in your bean. For each screen, Glink checks to see if any output in the screen should be saved as output from the bean.

To go to the next screen, Glink enters the input parameters you have supplied. There are two types of input parameters, fixed and variable.

Fixed input fields have not been given a name, and the value you entered when stepping through the screens is used.

Variable input fields have been given a name by you. The input values for these fields must be supplied as input parameters to your bean before you test. In other words, they can vary with each execution of the bean, producing different results.

Finally, when Glink has reached the end of the screens in your function, Gargen displays the output from your bean.

Log on to Glink

Gargen will now start a visible copy of Glink for Java.

Enter the Glink log-on name and password that gives access to the target application. If you have not yet configured any Glink users or administrators, then use the default **admin** as both name and password.

If you are in playback mode the log-on name and password that you used when initially generating the bean will be supplied as defaults. You can change these; e.g. if you are modifying a development bean, ready for delivery to a production system with another log-on.

Note that Enterprise JavaBeans running under an Application Server skip this part of the dialog, and obtain a session from the J2EE Connection Factory. You can leave your test system log-on in place.

Select a session

The applications available to your Glink log-on name are listed below.

Select the application that has the function you want to encapsulate as a bean.

If you are in playback mode the session you used when initially generating the bean will be pre-selected. You can change this; e.g. if you are modifying a development bean, ready for delivery to a production system.

Note that Enterprise JavaBeans running under an Application Server skip this part of the dialog, and obtain a session from the J2EE Connection Factory. You can leave your test session in place.

Please note that Gargen supports only forms mode applications. If you have VT or other character mode applications configured as available for this Glink log-on name they will be listed but cannot be used.

Select the **Logon** button to start Glink as a frame in the Gargen window, and enter the screen definition phase.

The screen definition phase

For each screen (form) the mainframe application sends you can mark areas that identify the screen, mark output areas on the screen whose contents are to be returned by the bean, name input fields that must be supplied as parameters by the caller of the bean and finally enter necessary input data.

The dialog box for the screen definition phase is divided into three tabbed panes and a Glink pane:

Identify screen pane

In this pane you can mark one or more screen areas to identify the screen, or none if no screen verification is needed.

Name areas pane

In this pane you can mark screen areas that contain data that the bean should collect and return to the caller when executed. Each area must be given a unique name (key) for reference.

Name input fields pane

In this pane you can name one or more of the input fields. When you name an input field, it becomes a variable input field. The values for all the variable input fields defined are delivered as parameters when executing the bean, giving different results for different input values. For example, an input field for the account number could be defined as a variable field and the bean's task would be to get the balance for the account number specified.

Glink pane

In the Glink pane you enter data in the input fields required by the host application to present the next screen. There are two types of input data fields: constant and variable.

Constant data fields are input fields without a name, but with required input data. The bean remembers the data entered and uses it each time the bean is executed.

Variable data fields are input fields with a name. The bean does not remember data entered in these fields during the definition phase; it is supplied as parameters when the bean is executed.

Processing a screen

You have received a new screen from the application. At this point you can perform the tasks described in the Screen definition phase, using the tabbed panes, or terminate it by selecting the **Done** button.

Playback mode

If you have selected an existing bean you start in playback mode. In playback mode you can modify any parameters previously defined, such as the Glink log-on name, password, session name and any of the screen definitions.

In playback mode you are required to supply values for the input parameters as if you were calling the bean. Then you can step through the screens you have defined. For each screen received you will see the screen index number and the total number of screens defined. In the Glink pane the input parameters you have supplied are entered in the input fields. In the dialog panes, the screen definitions are listed. Select the **Next screen** button to save any modifications and go to the next screen. When you reach the last screen you can continue and define more screens if needed.

If you choose to terminate before you reach the last screen you get the option of removing the remaining screen definitions, including the current one.

Identify the current screen

You identify the current screen by marking one or more areas on the screen whose combined contents uniquely identify the screen. The bean uses this information to check that the expected screen has been received. You can skip the identification, but then your bean will not recognize unexpected screens.

In the Glink pane mark an area with the cursor and then use the **Add** button to select the marked area. Glink supports two modes of marking: line-by-line and rectangular. Choose the one you prefer using the **Edit** menu in the Glink pane. If the application can return this form with an error message in some area of the screen, then you should mark that area too so that identification fails if an error message is present there.

Mark and name screen areas

If the current screen contains data to be returned to the caller of your bean, you must identify the areas on the screen that contain the output. Each area must be given a unique name (key) for reference.

In the pane containing the Glink screen mark a screen area with the cursor and then use the **Add** button to add the marked area to the list box. In the name field in the list box, type the name of the marked area followed by the Enter key.

The marked areas are returned to the caller of your bean in a hash table. In this table each key and its corresponding value is a string. The name you give here is the key, and the content of the screen area is the value returned.

The key can be used by the calling application as a header for the returned data. For example, if you have marked an area that contains an address, it may be convenient to choose address as the key.

You can also supply additional information in the key. Let's say that you have two screens in your interaction, both with areas for which you want to use address as the header, and therefore the key. The key must be unique, but you could adopt a key convention and name the first field address:1 and the second field to address:2. The calling application would just display the part preceding the colon.

If you have marked a rectangle, you might adopt a convention to include the width of the rectangle in the key; e.g. name the rectangle address:w=23, where w=23 is information to the calling application, used to unpack the data.

Name input fields

Some of the input fields may be constants, to be used every time your bean reaches this screen. Other input fields may vary for each call to your bean, and will be supplied as input parameters to the bean at run time. This pane lists all the fields for the current form. Each field is listed with its index number, content as received from the mainframe and in playback mode, the name if already defined.

To name a field, select the field in the list box, place the cursor in the name cell and type the name followed by the Enter key. If the same input value is needed for several screens, use the same name each time.

When you name an input field, it becomes a variable input field. The values for all the variable input fields defined are supplied as parameters when executing the bean, giving different results for different input values.

Enter input data

In the Glink pane you enter the necessary data in the input fields. If you enter data in an input field that you have given a name (variable input data), the data will be sent to the host application but not recorded by the bean. If you enter data in a field without a given name (constant input data), the data will be sent to the host application and recorded.

Ready for next screen

The definitions you have made for the current screen will be saved as soon as you select the **Next screen** button or do a normal transmit of the screen. In either case, the screen input data is sent to the mainframe application, and it will return the next screen for you to process.

The screen definition phase is terminated when you select the **Done** button. The current screen is not saved. If you are in playback mode and terminate before the last recorded screen is received, you get the option of removing the remaining screen definitions including the current one.

If the definition phase is terminated you can use the **Resume** button to continue. If the host session is disconnected for some reason, the session is reconnected and the definition phase restarted from the first screen.

If you enter the wrong input data and receive an unexpected screen, you can navigate to the correct screen by first leaving the definition phase, then navigating with Glink to the correct screen and finally using the **Resume** button to continue the screen definition phase.

The generation phase

Generate your bean

You selected the **Done** button to leave the definition phase and you are in the process of generating a JavaBean. At this point you can change the bean name parameters, the bean will be generated when you save it.

You can modify the bean and package names if you would like to save the bean under another name or in another package. You can also change the bean format. Simply select the Enterprise JavaBean list item to generate an Enterprise JavaBean.

A standard JavaBean can be used by desktop applications, by applets running within a browser or by server-side applications running as servlets on any Web server supporting servlets. If you want your bean to use the G&R Connection Pool Manager select GCPM as the build target.

Connection Pool Manager (GCPM)

The G&R Connection Pool Manager (GCPM) is a component of Glink for Java Enterprise Edition that offers mainframe connection pool management in server environments. Clients, such as Servlets or Java Server Pages, request mainframe connections from named pools and are given fully operational GlinkApi objects in a known session state, freeing them from the details of configuring and instantiating GlinkApi objects. When a client no longer needs a connection it returns it to GCPM, which places it back in its pool where it becomes available for the next client.

Generate your Enterprise JavaBean

You selected the **Done** button to leave the definition phase and you are in the process of generating an Enterprise JavaBean. At this point you can change the bean name parameters; the bean will be generated when you save it.

You can modify the bean and package names if you would like to save the bean under another name or in another package. You can change the JNDI name by which the bean will be identified in the Java Naming and Directory Interface, and you can change the Connector interface fields. You can also change the bean format. Select the JavaBean list item to generate a standard bean.

An Enterprise JavaBean can only be used within a J2EE compliant application server, with the G&R JCA compliant Connector installed. Select your application server in the Build target drop-down list. Please contact G&R if your application server is not in the list.

Connector Interface for Enterprise JavaBeans

The EJB generator assumes that a G&R Connector is deployed in the Application Server, and it requests two names related to the connector:

- Connector's resource ref name
- Connector's real JNDI name

These names are defined by the Application Server system administrator and should be entered here to identify the Enterprise JavaBean generated by Gargen for the Application Server.

Examples (typical names in a Weblogic environment):

- eis/MyGcnx
- eis/MyGcnx.JNDI_NAME

Generate your bean, more information

You may have disconnected from the application, or still be connected and able to resume the interaction by selecting the **Resume** button to return and define further dialog steps.

To make reuse of your bean efficient you should return to a known state in the application dialog before finishing without disconnecting. For example, leave the application positioned in the menu that shows the function you intend to encapsulate in the bean. When the bean is reused it will recognize that Glink is started and already connected. It then skips dialog steps until it recognizes the current application screen. Only the final dialog steps are executed to obtain a new result from the application.

Select the **Save** button to save your bean. In later Gargen sessions you can then test and modify it. Select the **Exit** button to quit. Select the **Restart** button to return to the start of the bean definition phase.

Source files created by Gargen

Files created for Standard JavaBeans

You are in the process of generating a standard JavaBean. Gargen generates four files when saving a standard bean. If **MyBean** is the bean name specified, the following files will be generated in the directory `gargen\ggen\bean`:

MyBean.java, **MyBean.ser**, **MyBeanMakeJar.cmd**, **MyBeanTester.java**,
MyBeanTester.cmd

MyBean.java is your generated bean source and has the necessary code to perform the task you have stepped through with Gargen.

MyBean.ser contains the data for each screen you processed with Gargen. The data includes areas to be recognized for screen identification, areas to be collected for bean output, the content of constant input fields and the location of variable input fields whose content is to be supplied as bean input parameters.

MyBeanMakeJar.cmd is a batch file for compiling the source file generated. In addition it generates a JAR file that includes the necessary resource files together with the bean class file. Before you try to run **MyBeanMakeJar.cmd**, you should verify that you have installed a Java JDK, and that the path to the Java compiler matches the installation directory of the JDK.

MyBeanTester.java is an example of a Java application that uses your Bean. The "start" method shows how to use the bean. In addition, the tester application includes a class for entering the input parameters for the bean and a class for displaying the output parameters from the bean.

MyBeanTester.cmd is a batch file that compiles the tester program, generates a JAR file and finally starts the tester program.

Templates are used as input for the generation of the files described. These templates may be modified to suit your own setup.

Files created for Enterprise JavaBeans

You are in the process of generating an Enterprise JavaBean. Gargen generates a number of files when saving an Enterprise JavaBean. If **MyBean** is the name of the bean, the following files are generated in the directory `gargen\ggen\ejb`:

**MyBean.cmd, MyBean.sh, MyBean.java, MyBean.ser,
MyBeanAppServer.xml, MyBeanEJB.java, MyBeanEJBjar.xml,
MyBeanHome.java, MyBeanServlet.java, MyBeanWeb.xml,
MyBeanApplication.xml, MyBeanBuild.xml**

MyBean.cmd/MyBean.sh are sample batch/script files for compiling, packing and deploying your EJB in an Application Server. Before running them, you must set the appropriate environment variables for the application server to be used. These environment variables are specified for the JOnAS application server in the file `<gargen>/jonasenv.bat(.sh)`, for the Weblogic application server in the file `<gargen>/wlsenv.bat(.sh)`, for Oracle AS in `<gargen>/oraclenv.bat(.sh)`, and for WebSphere in `<gargen>/webspherenv.bat(.sh)`.

MyBean.java is the Remote interface of your Enterprise JavaBean.

MyBean.ser is a text file containing the data for each screen you processed with Gargen. The data includes areas to be recognized for screen identification, areas to be collected for bean output, the content of constant input fields and the location of variable input fields whose content is to be supplied as bean input parameters.

MyBeanAppServer.xml is a sample Application Server specific deployment descriptor for the Enterprise Javabean, pairing up with `ejb-jar.xml`. Not applicable for WebSphere.

MyBeanEJB.java is the generated Enterprise JavaBean class.

MyBeanEJBjar.xml is the standard J2EE EJB deployment descriptor; the build script will rename this to `ejb-jar.xml` during the build.

MyBeanHome.java is the EJB's Home interface.

MyBeanServlet.java is an example of a Java servlet that uses your generated EJB. If you have a Web server that supports Java servlets you can test the bean from a browser using this servlet. The servlet must be added to the Web server's servlet configuration before testing.

MyBeanWeb.xml is the Web application deployment descriptor; the build script will rename this to `web.xml` during the build.

MyBeanApplication.xml is the `.ear` application deployment descriptor; the build script will rename this to `application.xml` during the build.

MyBeanBuild.xml is an ANT build script used by `MyBean.cmd/.sh` to actually build the application `.ear` file.

Templates

Templates are used as input for the generation of the files described. These templates can be modified to suit your local environment.

Standard JavaBeans

The templates used to generate Standard JavaBeans are located in the directories:

```
gargen\no\gar\ggen\bean
gargen\no\gar\ggen\bean\GCPM
gargen\no\gar\ggen\bean\Standard
```

The common files are found in the bean directory. In the subdirectories there are specific templates for standard beans that make their own connections and for beans that use the G&R Connection Pool Manager (GCPM) to obtain connections from a pool.

Gargen uses the templates to generate source files for the bean and command files for compiling and deploying it. The tags below are found in the templates and replaced with the information defined in Gargen prior to saving the bean:

<<BEANTYPE>>	Standard or GCPM
<<BEANNAME>>	Bean name
<<PACKAGE>>	Package name
<<PACKAGEDIR>>	Package name inserted in a file path using \ as name separator
<<DIRPACKAGE>>	Package name inserted in a file path using / as name separator
<<POOLNAME>>	The name of the GCPM pool from which to obtain a connection

Enterprise JavaBeans

The templates used to generate Enterprise JavaBeans are located in the directories:

```
gargen\no\gar\ggen\ejb
gargen\no\gar\ggen\ejb\JonAS XX
gargen\no\gar\ggen\ejb\Weblogic XX
gargen\no\gar\ggen\ejb\Oracle XX
gargen\no\gar\ggen\ejb\WebSphere
```

Gargen uses the templates to generate source and command files for compiling and deploying the Enterprise JavaBean. The tags below are found in the templates and replaced with the information defined in Gargen prior to saving the bean:

```
<<BEANTYPE>>      JonAS, Weblogic, Oracle or WebSphere
<<BEANNAME>>      Bean name
<<LBEANNAME>>     Bean name in lower case
<<PACKAGE>>       Package name
<<PACKAGEDIR>>    Package name inserted in a file path using
                  \ as name separator
<<DIRPACKAGE>>    Package name inserted in a file path using
                  / as name separator
<<JNDINAME>>      EJB's JNDI name
<<CONRSCREF>>     Connector's resource reference name
<<CONREALJNDI>>   Connector's real JNDI name
```

You must modify the environment files to suit your environment:

```
gargen\jonasenv.bat (.sh)
gargen\wlsenv.bat (.sh)
gargen\oraclenv.bat (.sh)
gargen\webspherenv.bat (.sh)
```



Using your bean

Compile the source files

Use the batch file `BeanName.cmd (.sh)` that was automatically created to compile the bean. Read the notes given in the batch file. If the compilation fails, you might need to modify the batch file to match your system.

If you are using a Java development environment, you may of course compile your bean with your own tools. Make sure that the `glink.jar` file is inserted into the Java class path for the project in question.

Deploy your bean

A standard bean generated by Gargen can be called by a Java application on a workstation, a Java applet running within a browser, or it can be called by a servlet running on a Web server. In the following text we assume that the name of your bean is **MyBean** and the package name is **mypackage**.

To use the bean from a Java application, you need to include both the **BeanName.jar** file and the **glinkee.jar** (`glink.jar`) file in the Java class path.

The `gargen/ggen/bean/MyBeanMakeJar.cmd` is used to generate the `BeanName.jar` file. An example on how to use the bean from an application is shown in the `gargen/ggen/bean/MyBeanTester.java` file.

The `gargen/ggen/bean/MyBeanTester.cmd` file compiles and runs the `MyBeanTester` application.

If you have generated an Enterprise JavaBean to be used on a J2EE compliant Application Server, see the `gargen/ggen/bean/MyBean.cmd` file for instructions on how to deploy the Enterprise JavaBean.