

---

*GemStone*<sup>®</sup>

***GemBuilder***<sup>™</sup> ***for Java***<sup>™</sup>  
***Tools***

September 2002



Version 2.0 for GemStone/S 6.0.1

Send your comments about this manual to [docs@gemstone.com](mailto:docs@gemstone.com)

---

## IMPORTANT NOTICE

This documentation is furnished for informational use only and is subject to change without notice. GemStone Systems, Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in this documentation. The documentation, or any part of it, may not be reproduced, displayed, photocopied, transmitted or otherwise copied in any form or by any means now known or later developed, such as electronic, optical or mechanical means, without written authorization from GemStone Systems, Inc. Any unauthorized copying may be a violation of law.

The software installed in accordance with this documentation is copyrighted and licensed by GemStone Systems, Inc. under separate license agreement. This software may only be used pursuant to the terms and conditions of such license agreement. Any other use may be a violation of law.

Copyright © GemStone Systems, Inc. 1994-2002. All Rights Reserved.

Use, duplication, or disclosure by the Government is subject to restrictions set forth in the Commercial Software - Restricted Rights clause at 52.227-19 of the Federal Acquisitions Regulations (48 CFR 52.227-19) except that the government agency shall not have the right to disclose this software to support service contractors or their subcontractors without the prior written consent of GemStone Systems, Inc.

## Trademarks

**GemStone**, **GemBuilder**, **GemConnect**, and the GemStone logo are trademarks or registered trademarks of GemStone Systems, Inc. in the United States and other countries.

**Java** and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

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

**Sun**, **Sun Microsystems** and **Solaris** are trademarks or registered trademarks of Sun Microsystems, Inc. All **SPARC** trademarks, including **SPARCstation**, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation is licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

**Microsoft**, **MS**, **Windows** and **Windows NT** are registered trademarks of Microsoft Corporation in the U.S.A. and other countries.



**Netscape** is a registered trademark of Netscape Communications Corporation in the United States and other countries.

Other company or product names mentioned herein may be trademarks or registered trademarks of their respective owners. Specifications are subject to change without notice. All terms mentioned in this documentation that are known to be trademarks or service marks have been appropriately capitalized. GemStone cannot attest to the accuracy of this information. Use of a term in this documentation should not be regarded as affecting the validity of any trademark or service mark.

**Patents**

Facets is covered by U.S. Patent Number 6,256,637 “Transactional virtual machine architecture” and Patent Number 6,360,219 “Object queues with concurrent updating”. Facets may also be covered by one or more pending United States patent applications.





---

## About This Documentation

This documentation describes GemBuilder® for Java™ 2.0 Tools, an integrated environment for programming GemStone® Smalltalk applications. These tools consist of the following:

- A *GemStone Browser* for examining, creating, and modifying GemStone classes and methods.
- A *Workspace* for running small pieces of GemStone Smalltalk.
- An *Inspector* for examining and modifying the state of GemStone objects.
- A *Debugger* for examining and modifying an execution stack created by GemStone Smalltalk execution.
- A *Class Version Browser* for examining a class history, inspecting instances, migrating instances, deleting versions, and moving versions to another class history.
- A *Breakpoint Browser* for enabling, disabling, and removing breakpoints in methods, as a debugging aid examining a class history, inspecting instances, migrating instances, deleting versions, and moving versions to another class history.

## Assumptions

To make use of the information in this documentation, you need to be familiar with the GemStone server and with the GemStone Smalltalk programming language as described in the *GemStone/S Programming Guide*. That book explains the basic concepts behind the language and describes the most important GemStone kernel classes.

This documentation assumes that the GemStone system has been correctly installed on your host computer as described in the GemStone System Administrator's Guide and that your system meets the requirements listed in your *GemBuilder for Java Installation Guide*.

## How This Manual is Organized

**Communicating with the Server** explains how to communicate with the GemStone object server by initiating a GemStone session.

**The Launcher** explain how to start the Tools as a standalone application or as a Java applet running within a Web browser.

**The Browsers** explains how to use the GemStone browsers and tools to create classes and methods in the GemStone server.

**Related Tools** describes how to use the GemStone workspace, inspectors, and the debugging tool with your GemStone Smalltalk code.

**Working with GemStone Classes and Methods** explains what you need to know in order to define new GemStone classes and add them to the system. It explains the difference between modifiable and invariant classes, and it shows you how to constrain instance variables for faster access.

## Other Useful Documents

- The *GemBuilder for Java Programming Guide*, which is shipped in online and printable forms with GemBuilder for Java.
- The *GemStone/S Programming Guide* describes the GemStone System and the GemStone Smalltalk language.
- If you will be acting as a system administrator, or developing software for someone else who must play this role, you should read the *GemStone System Administration Guide*.

---

---

## Technical Support

GemStone provides several sources for product information and support. GemBuilder for Java product manuals provide extensive documentation, and should always be your first source of information. GemStone Technical Support engineers will refer you to these documents when applicable. However, you may need to contact Technical Support for the following reasons:

- Your technical question is not answered in the documentation.
- You receive an error message that directs you to contact GemStone Technical Support.
- You want to report a bug.
- You want to submit a feature request.

Questions concerning product availability, pricing, keyfiles, or future features should be directed to your GemStone account manager.

When contacting GemStone Technical Support, please be prepared to provide the following information:

- Your name, company name, and GemStone/S license number
- The GemBuilder product and version you are using
- The hardware platform and operating system you are using
- A description of the problem or request
- Exact error message(s) received, if any

Your GemBuilder for Java support agreement may identify specific individuals who are responsible for submitting all support requests to GemStone. If so, please submit your information through those individuals. All responses will be sent to authorized contacts only.

For non-emergency requests, you should contact Technical Support by web form, email, or facsimile. You will receive confirmation of your request, and a request assignment number for tracking. Replies will be sent by email whenever possible, regardless of how they were received.

**World Wide Web: <http://support.gemstone.com>**

This is the preferred method of contact. The Help Request link is at the top right corner of the home page—please use this to submit help requests. This form requires an account, but registration is free of charge. To get an account, just complete the Registration Form, found in the same location. You'll be able to access the site as soon as you submit the web form.

**Email: [support@gemstone.com](mailto:support@gemstone.com)**

Please do not send files larger than 100K (for example, core dumps) to this address. A special address for large files will be provided as appropriate.

**Facsimile: (503) 629-8556**

When you send a fax to Technical Support, you should also leave a voicemail message to make sure your fax will be picked up as soon as possible.

**Telephone: (800) 243-4772 or (503) 533-3503**

We recommend you use telephone contact only for more serious requests that require immediate evaluation, such as a production system that is non-operational.

Emergency requests are handled by the first available engineer. If you are reporting an emergency and you receive a recorded message, do not use the voicemail option. Transfer your call to the operator, who will take a message and immediately contact an engineer.

Non-emergency requests received by telephone are placed in the normal support queue for evaluation and response.

## **24x7 Emergency Technical Support**

GemStone offers, at an additional charge, 24x7 emergency technical support. This support entitles customers to contact us 24 hours a day, 7 days a week, 365 days a year, if they encounter problems that cause their production application to go down, or that have the potential to bring their production application down. Contact your GemStone account manager for more details.

---

---

## GemStone Web Site

The GemStone Web site, at <http://support.gemstone.com>, provides a variety of information to help you use GemBuilder products. Use of this site requires an account, but registration is free of charge. To get an account, just complete the Registration Form, found in the same location. You'll be able to access the site as soon as you submit the web form.

The following types of information are provided at this web site:

**Help Request** is an online form that allows designated technical support contacts to submit requests for information or assistance via email to GemStone Technical Support.

**Technotes** provide answers to questions of general interest submitted by GemBuilder customers. They may contain coding examples, links to other sources of information, or downloadable code.

**Bugnotes** identify performance issues or error conditions that you may encounter when using a GemBuilder product. A bugnote describes the cause of the condition, and, when possible, provides an alternative means of accomplishing the task. In addition, bugnotes identify whether or not a fix is available, either by upgrading to another version of the product, or by applying a patch. Bugnotes are updated regularly.

**Patches** provide code fixes and enhancements that have been developed after product release. A patch generally addresses a specific group of behavior or performance issues. Most patches listed on the GemStone Web site are available for direct downloading.

**Tips and Examples** provide information and instructions for topics that usually relate to more effective or efficient use of GemStone products. Some Tips may contain code that can be downloaded for use at your site.

**Release Notes and Install Guides** for your product software are provided in PDF format.

**Documentation** for GemBuilder is provided in PDF format. This is the same documentation that is included with your GemBuilder product, with the exception of the API reference files (javadocs).

**Community Links** provide customer forums for discussion of GemBuilder product issues.

Technical information on the GemStone Web site is reviewed and updated regularly. We recommend that you check this site on a regular basis to obtain the latest technical information for GemStone products. We also welcome suggestions and ideas for improving and expanding our site to better serve you.

## Training and Consulting

Consulting and training for all GemStone products are available through GemStone's Professional Services organization.

- Training courses are offered periodically at GemStone's offices in Beaverton, Oregon, or you can arrange for onsite training at your desired location.
- Customized consulting services can help you make the best use of GemStone products in your business environment.

Contact your GemStone account representative for more details or to obtain consulting services.

---

**Chapter 1. Connecting to the GemStone Server**

Overview . . . . .	1-1
Launching the Tools . . . . .	1-3
To Start a Standalone Launcher . . . . .	1-3
Logging in to GemStone/S . . . . .	1-3
Logging out of GemStone/S. . . . .	1-6
Administering the Server Component . . . . .	1-6
Running the Session Broker. . . . .	1-6
Effect of NetLDI Mode . . . . .	1-7
Configuration Files . . . . .	1-7
To Start the Session Broker. . . . .	1-10
To Halt the Session Broker . . . . .	1-11
Connecting to the Session Broker Gem . . . . .	1-12
To Run Multiple Session Brokers . . . . .	1-13
Maintaining the Log Directory . . . . .	1-14
Troubleshooting . . . . .	1-14
To Determine if a Session Broker Is Running . . . . .	1-14
To Restart a Session Broker. . . . .	1-15
To Locate Log Files . . . . .	1-15

---

To Enable Verbose Logging . . . . .	1-15
-------------------------------------	------

## **Chapter 2. The Launcher**

Overview . . . . .	2-1
The Launcher Buttons . . . . .	2-2
The Launcher Menus . . . . .	2-3
The File Menu . . . . .	2-3
The Edit Menu . . . . .	2-4
The Session Menu . . . . .	2-4
The Tools Menu . . . . .	2-5
The Windows Menu . . . . .	2-6

## **Chapter 3. The Browsers**

The GemStone Browser . . . . .	3-1
The Symbol List Pane . . . . .	3-2
The Classes Pane . . . . .	3-2
The Protocol Pane . . . . .	3-3
The Text Pane . . . . .	3-3
The Browser Menus . . . . .	3-3
The File Menu . . . . .	3-3
The Edit Menu . . . . .	3-3
The Session Menu . . . . .	3-4
The Dictionary Menu . . . . .	3-5
The Class Menu . . . . .	3-5
The Protocol Menu . . . . .	3-7
The Method Menu . . . . .	3-7
Specialized Browsers . . . . .	3-8
The Class Browser . . . . .	3-8
The Hierarchy Browser . . . . .	3-9
The Class Version Browser . . . . .	3-10
The Method Browser . . . . .	3-11
The Method List Browser . . . . .	3-12
The Breakpoint Browser . . . . .	3-12
Breakpoint Browser Buttons . . . . .	3-13
Breakpoint Menu . . . . .	3-14

## **Chapter 4. Related Tools**

Overview . . . . .	4-1
The Workspace . . . . .	4-2
The Inspector . . . . .	4-3
Inspecting Dictionaries and Sets . . . . .	4-5
The Debugger . . . . .	4-6
Debugger Buttons . . . . .	4-8
Debugger Menus . . . . .	4-8
Invoking the Debugger from a Workspace . . . . .	4-9
Invoking the Debugger from Your Java Client . . . . .	4-10
Common Menus and Shortcuts . . . . .	4-12
The Popup Menu . . . . .	4-12
Keyboard Shortcuts . . . . .	4-13

## **Chapter 5. Working with GemStone Classes and Methods**

About GemStone Classes . . . . .	5-1
Instance Variables Can Be Constrained . . . . .	5-1
Constraints Are Inherited . . . . .	5-1
Instances Can Be Made Invariant . . . . .	5-2
Reserved Selectors . . . . .	5-3
Optimized Selectors . . . . .	5-3
Defining a New Class . . . . .	5-4
To Define the Class . . . . .	5-4
Example of a Class Definition . . . . .	5-6
Private Instance Variables . . . . .	5-7
Subclass Creation Methods . . . . .	5-7
Compiling the Class Definition . . . . .	5-8
If an Error Occurs . . . . .	5-8
Modifying an Existing Class . . . . .	5-8
Defining Methods . . . . .	5-9
Public and Private Methods . . . . .	5-10
Saving Class and Method Definitions in a File . . . . .	5-10
Reading and Compiling a Saved File . . . . .	5-11
Sample File-out Contents . . . . .	5-11

## ***Glossary***

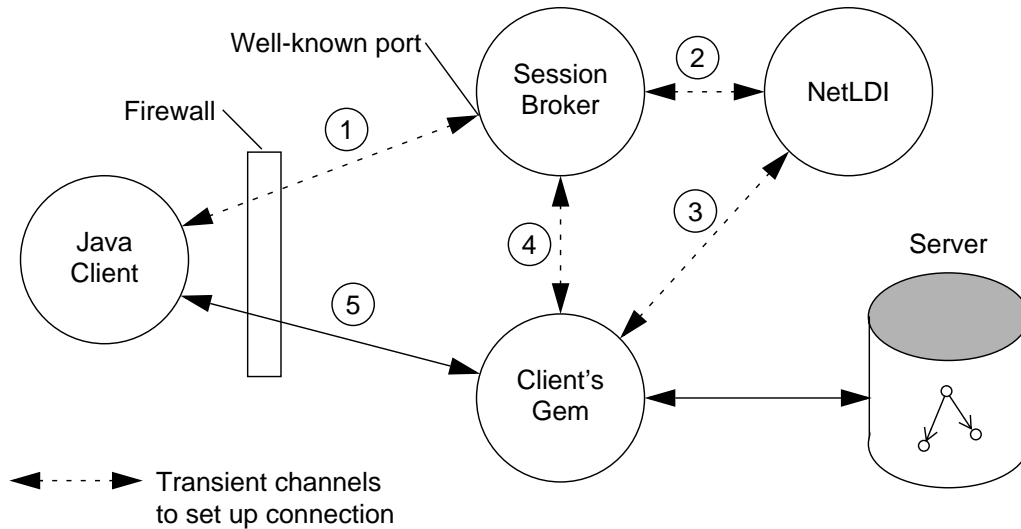
# *Connecting to the GemStone Server*

---

## **Overview**

The GemBuilder for Java Tools are Java applications that make use of the underlying GemBuilder for Java application programming interface. You can use these tools by themselves, or you can integrate them into popular Java development environments like Microsoft Visual J++, Sun's Java Workshop, and Symantec Café by following instructions in our *Installation Guide*.

All interaction with the GemStone/S server takes place through a Gem process that is created to serve your Tools session. An intermediary called the *Session Broker* is responsible for starting the Gem process (through a NetLDI network server) and for placing the Gem and the Tools in communication with each other.

**Figure 1.1 Session Broker Connecting a Client to the GemStone/S Repository**

A GemStone/S administrator starts the Session Broker from a Topaz session using a chosen TCP/IP port number. During login, the Tools use the port number and the name of the GemStone/S server to contact the Session Broker and have it start the Gem session process.

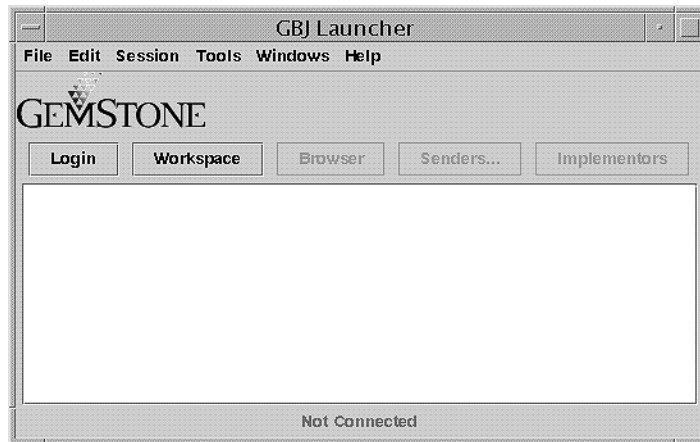
In order to log in from the Tools, you must have at hand at least the following information:

- the name of the GemStone/S server, such as gemserver60
- the name of the machine on which the Session Broker is running
- the port number at which the tools launcher is to contact the Session Broker, such as 9090 (the default)
- the GemStone/S user ID under which you will log in, and the corresponding password.

## Launching the Tools

The GemStone Launcher window lets you log in to a GemStone server, which gives you access to the GemStone Smalltalk compiler and the GemBuilder for Java Tools. Initially, only the **Login...** and **Workspace** buttons are enabled. The lower pane, the text pane, serves as a transcript for system messages.

**Figure 1.2** GemStone Tools Launcher



You can run the Launcher as a stand-alone Java application within a Web browser.

### To Start a Standalone Launcher

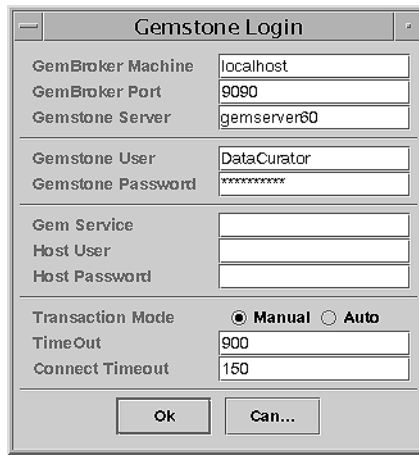
To start the Launcher as a standalone application, invoke the Java runtime system from an operating system command prompt. Your CLASSPATH environment variable must include the location of `gbjtools10.zip`, as described in the documentation for your Java system and our *Installation Guide*. For example:

```
c:\> java com.gemstone.tools.GbjLauncher
```

### Logging in to GemStone/S

After the Launcher appears, the next step ordinarily is to log in to a GemStone/S server. Click the **Login...** button in the Launcher, which opens a dialog in which you identify the GemStone/S server and provide the GemStone/S userID, password, and other relevant information (the client can provide this information programmatically). Use the mouse or Tab key (not Return) to move between fields.

**Figure 1.3 Login Dialog**



**NOTE:**

*After you log in, all tools you start will run under that GemStone/S session. To run tools under a different session, start the Launcher again.*

Parameter	Explanation
GemBroker Machine	The name or IP address of the machine where the Session Broker is running (the default "localhost" may be used if the Session Broker is on your machine)
GemBroker Port	The number of the well-known port at which the Session Broker is listening: either 9090 or the number provided by the GemStone administrator who started the Session Broker.
GemStone Server	The name of the GemStone server. The default is "gemserver51". This field can include the name of server's machine using Network Resource String (NRS) syntax; for instance, "!@handel!gemserver60". GemBuilder for Java must be installed on the server.
GemStone User	The userId under which you want to log in (must already exist in the server)
GemStone Password	The password for the GemStone userId

Gem Service (optional)	The name of the Gem service to be used. Initially, the default (not shown) is “gemnetobject”; users of the UNIX C Shell may prefer “gemnetobjcsh”. Because the Gem and Broker must be on the same machine, you should not specify a machine name as part of an NRS.
Host User	An operating system login name on the machine where the Session Broker is running. (This user name and password are required to start a separate Gem process unless your GemStone installation uses another method of providing network authentication, such as a .netrc file or running NetLDIs in guest mode with a captive account.)
Host Password	The operating system password for the Host User on the machine where the Session Broker is running
Transaction Mode	Choose <b>Automatic</b> if you are going to be committing changes to the server. Choose <b>Manual</b> if your primary activity will be reading the committed state; before making a change you want to commit, begin a transaction manually by choosing <b>Session &gt; Begin</b> in the GemStone Browser. (For more information, see the GemBuilder for Java documentation or the <i>GemStone/S Programming Guide</i> .)
Timeout	The length of time in seconds that the server will wait before it drops an idle session. If zero, the connection will not time out. The default is 900 seconds (15 minutes).
Connect Timeout	The length of time in seconds that Java client will wait for a connection request to be fulfilled by the Session Broker. The default is 30 seconds.

For information about Network Resource Syntax and the NetLDI, see the *GemStone System Administration Guide*.

## Logging out of GemStone/S

When your login succeeds, the button label is changed to **Logout**. Click this button to end your session. All browsers, inspectors, and debuggers are closed automatically, but the launcher and any workspaces remain open. The workspaces remain open so you can save text, but you can no longer run code.

## Administering the Server Component

This topic provides information to help you configure the Session Broker and troubleshoot problems that may arise.

### Running the Session Broker

A Session Broker can be run on the GemStone/S server's machine or on any other supported platform that has a network connection to the server. The primary consideration in locating a Session Broker is that the Gems it spawns will be located on the same machine. There are three parts to consider: the GemStone/S server, the Session Broker and Gems, and the Java clients. Each part can be on a separate machine, so several configurations are possible:

- The Session Broker can run on the Server's machine, and all Gems it spawns will be on that machine. This configuration is the simplest, and is suitable in most cases.
- The Session Broker can run on a client's machine, and the Gems will be spawned on the client's machine. Where necessary, each of several machines can have its own Session Broker.
- The Session Broker and Gems can run on an separate machine, offloading that activity from the server and client machines.

*NOTE:*

*Ordinarily, a GemStone/S NetLDI network server must be running on the Session Broker's machine so the broker can use it to spawn Gems.*

## Effect of NetLDI Mode

The mode in which the NetLDI is running affects the ownership of the Gems that are spawned and the network authentication requirements. In general, the effect is the same as for other parts of a GemStone/S installation:

- The default mode requires authentication by way of a host user name and password on the Session Broker's machine. This information can be provided as session parameters or by a .netrc file. The Gem is owned by the account under which the Session Broker is running.
- Guest mode makes it unnecessary to provide authentication. Under UNIX, this mode typically is combined with captive account mode, which causes the captive account to own the Gem processes. Under Windows NT, guest mode is the recommended mode; it reduces security but provides increased convenience.

## Configuration Files

You can use a configuration file to provide settings for the Session Broker to use at startup. The configuration file is optional; if you specify none, the Session Broker looks for a file in the current directory with the default name gbj.ini. If it finds no such file, it uses default values.

You can use environment variables in a configuration file if you delimit them with the character %. For example:

```
fileDirectory=%GEMSTONE%/data
```

Lines preceded by a semi-colon are comments. The following example configuration file explains each parameter:

### Example 1.1 Configuration File

---

```
; GBJ Broker Configuration Parameters
; start section --
[gbjbroker]
;
; serverPort is the port number to use for the broker's
; server socket.
; Permissible values: integer > 0.
; serverPort=9090
serverPort=9090
;
```

```
; ports are numbers to use for temporary server sockets
; in spawned gems.
; Permissible values: a sequence of integers or
; range of integers, separated by spaces or commas.
; For example: ports= 9091 9093-9094,9096
; ports=9091-9099
ports=9091-9099
;
; fileDirectory is the path name of the log file
; directory -- the location for log files in spawned gems
; and default location for the broker's log file.
; Permissible values: a valid directory path for
; the host OS.
; Delimit environment variables with %.
; fileDirectory=.
fileDirectory=.
;
; logFile is the name of the broker's log file.
; If no directory is specified, the file is placed in
; #fileDirectory.
; Delimit environment variables with %.
; The default log file name is 'gbjbroker.log'.
; Permissible values: a valid file name for the host OS.
; logFile=gbjbroker.log
logFile=gbjbroker.log
;
; If verbose is true, the broker writes detailed messages
; to the log file. If verbose is false, the broker writes
; error messages only.
; Permissible values: true or false.
; verbose=false
verbose=false
;
; gbjAdministrator and gbjAdministratorPassword are the
; GemStone username and password used by the startgbj and
; stopgbj scripts for starting and stopping a broker.
; Permissible values: valid GemStone username and
; password.
gbjAdministrator=DataCurator
gbjAdministratorPassword=swordfish
;
; gbjBrokerConnectToken is the GemStone username for
; connecting to a running broker.
; This need not be a valid GemStone username.
; If the GemStone username in a connection request
; matches #gbjBrokerConnectToken, the broker establishes
```

```
; a connection in the broker's gem.
; gbjBrokerConnectToken=GbjAdministrator
gbjBrokerConnectToken=GbjAdministrator
;
; Next are GemStone parameters for spawning gems.
; Permissible values: valid GemStone login parameters.
gemStoneName=gemserver51
username=DataCurator
password=swordfish
hostUsername=
hostPassword=
gemService=gemnetobject
;
; initialGemCount is the number of gems to spawn on
; startup. Gems are spawned as needed. If you know how
; many you will need, starting all at once may save time.
; Permissible values: an integer >= 0 and >=
; #initialGemMinimum.
; initialGemCount=1
initialGemCount=1
;
; initialGemMinimum is the minimum number of gems to
; spawn on startup. If this many gems are not created
; before the number of seconds specified by
; #initialDelay, the broker quits.
; Permissible values: an integer >= 0 and <=
; #initialGemCount.
; initialGemMinimum=1
initialGemMinimum=1
;
; initialDelay is the number of seconds to wait for
; the #initialGemMinimum gems on startup.
; The broker quits if this many seconds pass before
; #initialGemMinimum gems have been created.
; 0 means no time limit.
; Permissible values: an integer >= 0
; initialDelay=180
initialDelay=180
;
; maximumGemCount is the maximum number of gems possible.
; Permissible values: an integer >= 0 and >=
; #maximumIdleGems and >= #initialGemCount.
; Default is the value of the #STN_MAX_SESSIONS GemStone
; configuration option minus one for the broker session.
; 0 means no limit.
; Note: The default value of #STN_MAX_SESSIONS is 40.
```

```
; (See GemStone Administration Guide).
; maximumGemCount=0
;
; maximumIdleGems is the maximum number of idle gems.
; If a gem goes idle, causing the number of idle gems to
; exceed this number, the gem is terminated.
; Permissible values: an integer >= 0 and <=
; #maximumGemCount.
; Default is the value of the #maximumGemCount option.
; 0 means no limit.
; maximumIdleGems=0
; End of configuration file.
```

---

## To Start the Session Broker

You can start the Session Broker using either the startup script provided, or Topaz (GemStone's command line programming environment).

A configuration file is required if you wish to run the startup script. To do so, enter the following at the operating system command prompt:

```
C:\> startgbj gbj.ini
```

You can substitute another file name for gbj.ini if you wish to use a different configuration file. If you supply no file name, GemBuilder for Java looks for a file named gbj.ini in the current directory. If it finds no such file, the script fails.

To use the command line:

1. Log in to the GemStone/S server as any user, using Topaz (GemStone's command line programming environment).
2. To use the default settings, enter at the command line:

```
topaz 1> doit
GbjBroker new startup
%
```
3. Note the port number and the log directory that are displayed. You can minimize the window, but don't close it — that would halt the Session Broker.

To start a Session Broker from the command line using a different configuration file, enter:

```
topaz 1> doit
(GbjBroker newOnFile: 'c:\mygbj.ini') startup
%
```

The GbjBroker class also provides methods to override specific configuration settings, either from a configuration file or the defaults. To override the server port setting, for example, enter:

```
topaz 1> doit
(GbjBroker newOnFile: 'gbj.ini')
serverPort: 9080;
startup
%
```

For related information, see Configuration Files.

## To Halt the Session Broker

You can halt the Session Broker using either the shutdown script provided, or by connecting to the Session Broker as described in the next topic.

A configuration file is also required if you wish to run the shutdown script. To do so, enter the following at the operating system command prompt:

```
C:\> stopgbj gbj.ini
```

This shuts down only if all Gems are idle. If some Gems are active, use either the **-i** or **-t** option.

The **-i** option shuts down the Session Broker immediately, even if Gems are still active.

Follow **-t** with an integer representing a number of seconds. This option disables logins and then shuts down Gems as soon as they are idle, until the specified number of seconds has passed. If some Gems are still active, the Session Broker is not shut down and logins are reenabled.

An additional optional argument, **-h**, prints usage information and exits.

You can substitute another file name for `gbj.ini` if you're using a different configuration file. If you supply no file name, GemBuilder for Java looks for a file named `gbj.ini` in the current directory. If it finds no such file, the script fails.

### NOTE:

*In this release, it is not possible to resume execution of the Session Broker after pressing Control-C.*

## Connecting to the Session Broker Gem

You can connect to the Session Broker's Gem in order to shut it down, or to perform other administrative tasks. Doing so requires that you use a configuration file.

### Logging In

Log in through GemBuilder for Java with a GemStone/S username that matches the `gbjBrokerConnectToken` parameter in the configuration file. The default is `GbjAdministrator`. No password is needed.

### Shutting Down the Session Broker

Once connected to the Session Broker's Gem, you can shut it down by sending any of several messages either to the `GbjBroker` class, or to the current instance of `GbjBroker`. For example:

```
GbjBroker shutdown
```

or

```
GbjBroker current shutdown
```

The shutdown messages available are:

<code>shutdown</code>	Stops the Session Broker if all Gems are unconnected.
<code>shutdownImmediate</code>	Stops the Session Broker immediately, whether Gems are unconnected or busy, terminating all Gems.
<code>shutdownWait</code>	Logins are disabled. The Session Broker waits for all Gems to disconnect before shutting down.

shutdownWait:	Takes an integer argument specifying the number of seconds to wait for all Gems to become idle before stopping the Session Broker. Logins are disabled immediately, except for other logins to the Session Broker Gem. If the number of seconds specified elapses and some Gems are still busy, logins are reenabled and the Session Broker is not shut down. To specify that the Session Broker must wait indefinitely, supply a negative number as an argument.
shutdownCancel	Stops the shutdown process started by shutdownWait.

**NOTE:**

*In this release, it is not possible to resume execution of the Session Broker after a Control-C.*

## To Run Multiple Session Brokers

You can run more than one Session Broker on a single machine, although there is no need to do so (you can connect to any local GemStone/S server through a single Session Broker). You must be careful to use separate ports and log directories for each broker on the same machine.

1. Choose a set of unused TCP/IP ports and a log directory for the new broker.
2. Make a copy of the `gbj.ini` configuration file (in the GemBuilder for Java server directory) under another name.
3. Edit the copy to substitute the chosen ports and log directory. The lines you need to change look like these:

```
serverPort=9090
ports=9091-9099
fileDirectory=.
```

4. Follow the instructions for starting a Session Broker, logging in to a different server and using the modified configuration file.

For related information, see [To Start the Session Broker and Configuration Files](#).

## Maintaining the Log Directory

You should plan to remove old log files periodically from the file directory specified in `gbj.ini`. The files that the Session Broker and each Gem create in this directory are not deleted automatically.

- The Session Broker log, `gbjbroker.log`, continues to grow until the broker is restarted, at which time the previous log file is replaced by a new one.
- Each Gem creates a log file having the name `gbjnn.log`, where *nn* is a unique, sequential number that begins at 1 and increments until the Session Broker is restarted, at which time the number is reset to a value of 1.

*CAUTION:*

*Depending on the frequency at which clients are started, the length of time the Session Broker has been running, and the chosen degree of logging detail (verbosity), it is possible for the log files to fill the disk.*

## Troubleshooting

If you have trouble connecting to the server, first make sure the Session Broker is running and that you have the correct machine name and port number. These can be verified by reviewing information displayed at the time the broker was started or by examining the broker's log file.

You might find it helpful to examine the broker's log file for clues. If a problem persists, try restarting the broker under verbose logging so that more information is available.

## To Determine if a Session Broker Is Running

1. Go to the window in which the Session Broker was started.
2. Check the status:
  - If the last status line reads "Ready" and the prompt has not returned, the Broker is running.
  - If the prompt has returned, the Session Broker has exited.

*TIP:*

*Note the GbjBroker port number and the location of the broker log file. These items are part of the status information displayed by the startup script.*

## To Restart a Session Broker

1. Make sure the previous Session Broker was terminated cleanly. If in doubt, stop the broker as described under To Halt the Session Broker.
2. Start a new Session Broker. You can use the same port numbers.

In this release, it is not possible to resume operation of a broker that has been halted.

For related information, see To Start the Session Broker.

## To Locate Log Files

The log files for the Session Broker and the Gems are located in the directory specified by the `fileDirectory` instance variable of the GemStone GbjBroker. This variable is set by the broker configuration file, `gbj.ini`.

- If possible, check the window in which the Session Broker is running. The current value of `fileDirectory` is displayed at the time you start the Session Broker.
- If necessary, check the configuration file to determine the setting that was used. Its initial location is the GemBuilder for Java `server` directory.
- If the NetLDI is running under a captive account, the session log files (`gbjnn.log`) are created in the home directory of the captive account or as otherwise specified when the NetLDI was started.

## To Enable Verbose Logging

1. Open the file `startup.gs` (in the server directory) for editing.
2. Edit the line that controls verbose logging so it looks like this:  

```
verbose: true; "write debugging messages to the log file"
```
3. If the Session Broker is running, halt it.
4. Restart the broker.

For related information, see To Halt the Session Broker and To Start the Session Broker.

You can enable verbose logging for a particular session by choosing **File > Settings > Verbose Client** or **Verbose Server**.



# *The Launcher*

---

## **Overview**

The Launcher not only lets you log in to a GemStone/S server, but also gives access to the other Tools:

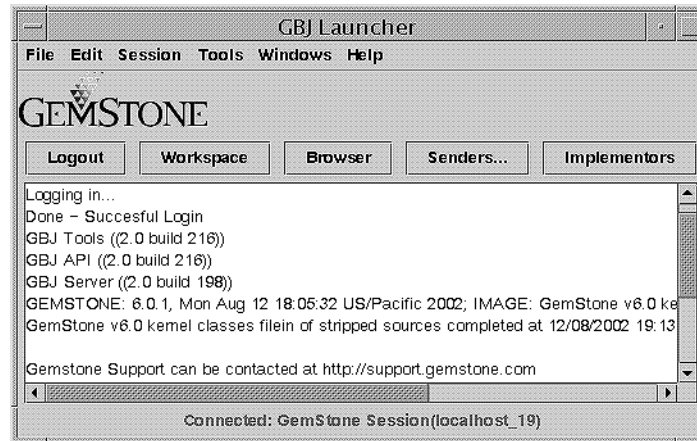
- GemStone Browsers and specialized browsers derived from it
- Workspaces
- Inspectors

The Launcher contains a text pane that serves as both a session transcript and as a workspace. The transcript records significant events, such as logins, commits, and logouts. Clicking **Help** displays version information in the transcript.

---

**Figure 2.1 Session Transcript in Launcher Text Pane**


---



## The Launcher Buttons

Five Launcher buttons provide convenient access to frequently used functions: Login/Logout, Browser, Workspace, Senders..., and Implementors... .

The **Login** button opens a dialog through which you enter session parameters and log in to the GemStone/S server, as described in *Launching the Tools*. When the login attempt succeeds, the label on this button changes to **Logout**.

The **Browser** button opens a GemStone Browser through which you can examine, create, or modify Smalltalk classes and methods in the server. For further information, see *The GemStone Browser*. This button is enabled only when you are logged in to the server.

The **Workspace** button opens a text area in which you can evaluate GemStone Smalltalk expressions or edit text. This button is always enabled, but to evaluate expressions, you must be logged in to the server.

The **Senders...** button prompts for a method name, then opens a Method List Browser showing all visible server methods that send the specified message. For further information, see "The Method List Browser" on page 37. This button is enabled only when you are logged in to the server.

The **Senders...** and **Implementors...** buttons search the methods visible in the session's symbol list. For more information about how GemStone/S uses the

symbol list for name resolution, see The Symbol List Pane and the *GemStone/S Programming Guide*.

You can use wild-card characters in the search dialog associated with **Senders...** and **Implementors...**. A "\*" matching any number of characters, and "?" matching a single character.

## The Launcher Menus

The Launcher menus provide commands for executing GemStone Smalltalk code and accessing the GemStone Tools.

### The File Menu

The File menu contains these items:

<b>Open...</b>	Requests a file name, then copies the contents into the workspace (transcript)
<b>Save As...</b>	Requests a file name, then writes the contents of the workspace (transcript) into the specified file
<b>File In...</b>	Opens a file dialog allowing you to specify a source code file. Treats the selected file as being in GemStone file-in format (see Reading and Compiling a Saved File) and files it in.
<b>Settings</b>	Opens a dialog that allows you to specify a font and font style; and allows you to set verbose client logging <b>On</b> or <b>Off</b> . Changes affect the Launcher and any tools you open subsequently.
<b>Exit</b>	Kills the Tools application, closing all open browsers, inspectors, and so forth. The GemStone session is terminated.

## The Edit Menu

The Edit menu contains these items:

<b>Cut</b>	Cuts the currently selected text from the text pane and puts it in the clipboard.
<b>Copy</b>	Copies the currently selected text from the text pane and puts it in the clipboard.
<b>Paste</b>	Pastes the text from the clipboard at the current selection position in the text pane.
<b>Select All</b>	Selects all of the text in the text pane
<b>Do It</b>	Uses the GemStone compiler to evaluate and execute the currently selected text
<b>Print It</b>	Performs a <b>Do It</b> , sends the resulting object the message <code>printString</code> , then places the resulting string after the current text pane selection
<b>Inspect It</b>	Performs a <b>Do It</b> , and then opens a GemStone Inspector on the resulting object
<b>File It In</b>	Treats the selected text as being in GemStone file-in format (see Reading and Compiling a Saved File) and files it in.

## The Session Menu

The Session menu contains these items:

<b>Login...</b>	If there is no Launcher session, opens a login dialog and lets you log in to GemStone (same as clicking the <b>Login...</b> button)
<b>Logout</b>	Logs out the Launcher session, if there is one (same as clicking the <b>Logout</b> button)
<b>Commit</b>	Commits the Launcher session
<b>Abort</b>	Aborts (rolls back) the Launcher session
<b>Begin</b>	Begins a new transaction
<b>Manual</b>	Switches the Launcher session to manual transaction mode
<b>Auto</b>	Switches the Launcher session to automatic (chained) transaction mode

## The Tools Menu

The Tools menu contains these items:

<b>Browser</b>	Opens a GemStone System Browser (same as clicking the <b>Browser</b> button)
<b>Workspace</b>	Opens a GemStone workspace in which you can execute Smalltalk expressions (same as clicking the <b>Workspace</b> button)
<b>Senders...</b>	Requests a selector name, then opens a Method List Browser showing all senders of that message (same as clicking the <b>Senders...</b> button)
<b>Implementors...</b>	Requests a selector name, then opens a Method List Browser showing all implementors of that message (same as clicking the <b>Implementors...</b> button)
<b>References...</b>	Requests a symbol and then opens a Method List Browser showing all methods that refer to that symbol, including all references to associations with the specified symbol as the key.
<b>Find Substring...</b>	Requests a string, and then opens a Method List Browser showing all methods that contain the specified string in their source code. The search is case-sensitive. If no such string is found, opens a notifier instead.
<b>Browse Class...</b>	Requests a class name, then opens a Class Browser on the class
<b>Browse Hierarchy...</b>	Requests a class name, then opens a Hierarchy Browser on the class
<b>Breakpoints</b>	Opens a Breakpoint Browser.

You can use wild-card characters in the search dialog associated with **Senders...**, **Implementors...**, **Browse Class...**, and **Browse Hierarchy...**. A "\*" matching any number of characters, and "?" matching a single character.

## The Windows Menu

The Windows menu contains these items:

<b>Hide All</b>	Hides all windows.
<b>Show All</b>	Shows all windows.
<b>Name of each open window</b>	Each open window has an entry in the menu. You can show an individual window by clicking that window in the Windows menu.

# *The Browsers*

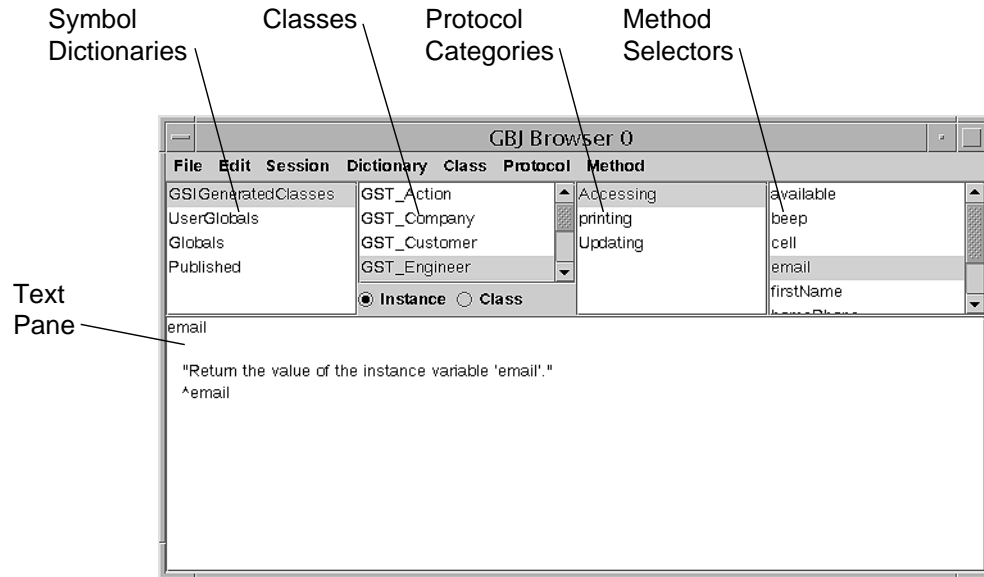
---

## **The GemStone Browser**

After you have successfully logged in to GemStone, you can open a GemStone Browser by choosing **Tools > Browser** or by clicking the **Browser** button in the Launcher. The GemStone Browser lets you create classes and methods in the server using GemStone Smalltalk.

The *GemStone Kernel Reference* provides full information about each of the GemStone kernel classes that you can examine through the GemStone Browser.

Figure 3.1 GemStone Browser



## The Symbol List Pane

The upper left pane contains a list of symbol dictionaries. In GemStone/S, the named objects to which your programs refer (the classes and root or global objects) are listed in symbol dictionaries. This list facilitates finding objects efficiently as well as sharing them among users. All of the symbol dictionaries that you can access in your current session environment are listed in the GemStone Browser's Symbol List pane.

When you select a symbol dictionary, all classes defined in that dictionary appear in the pane to the right of the Symbol List pane. (Symbols other than classes can be viewed by opening an inspector on the symbol dictionary in question.)

For further information about symbol lists, see the *GemStone/S Programming Guide*.

## The Classes Pane

When you select a class from the Classes pane, a list of its protocols appears in the Protocol pane adjacent on the right. In GemStone/S, methods are categorized by function, or *protocol*, to make them easier to browse.

## The Protocol Pane

When you select a protocol, all the message selectors in that category appear in the Method Selectors pane at the far right.

You can switch the focus of the display between instance methods or class methods by using the radio buttons provided.

## The Text Pane

When you select a method, its source code is displayed in the lower portion of the browser—the text pane. In this pane, you can edit and recompile the text of the displayed method, or execute fragments of GemStone Smalltalk code as in a workspace.

## The Browser Menus

### The File Menu

The following commands are available in the File menu:

<b>Update</b>	Updates the browser to match the current state of the Launcher's session. This action may change the contents of the browser because of changes made in other tools.
<b>Close</b>	Close the browser window.

### The Edit Menu

The Edit menu contains these items:

<b>Cut</b>	Cuts the currently selected text from the text pane and puts it in the clipboard.
<b>Copy</b>	Copies the currently selected text from the text pane and puts it in the clipboard.
<b>Paste</b>	Pastes the text from the clipboard at the current selection position in the text pane.
<b>Select All</b>	Selects all of the text in the text pane
<b>Do It</b>	Uses the GemStone/S compiler to evaluate the currently selected text

<b>Print It</b>	Performs a <b>Do It</b> , sends the resulting object the message <code>printString</code> , then places the resulting string after the current text pane selection
<b>Inspect It</b>	Performs a <b>Do It</b> , then opens a GemStone Inspector on the resulting object
<b>Compile</b>	Compiles the current text in the text pane. Use this choice to create a new class or method, or to modify existing definitions.
<b>Cancel</b>	Cancels any text entry done in the text pane, and returns the method definition, class definition, and so forth, to its unmodified form
<b>Set Break</b>	Sets a breakpoint at the current insertion point in the current method.

## The Session Menu

The Session menu contains these items:

<b>Commit</b>	Commits the Launcher session
<b>Abort</b>	Aborts (rolls back) the Launcher session
<b>Begin</b>	Begins a new transaction

If an attempted commit fails, choose **Report** in the resulting dialog to open a GemStone Inspector through which you can determine the cause of the failure. The inspector is on the result of invoking `System | transactionConflicts` in the server.

## The Dictionary Menu

The Dictionary menu contains these items:

<b>File Out As...</b>	Requests a file name, then creates a fileout (.gs file) containing the classes and methods in the selected symbol dictionary.
<b>Inspect</b>	Opens a GemStone Inspector on the currently selected symbol dictionary.
<b>Add...</b>	Requests a new dictionary name, then adds a new symbol dictionary by that name to your symbol list.
<b>Rename As...</b>	Requests a new dictionary name, then renames the selected symbol dictionary to that name.
<b>Remove...</b>	After confirmation, removes the selected symbol dictionary from your symbol list. <i>WARNING: Do not try to remove the Globals dictionary. The Globals dictionary defines the GemStone kernel classes.</i>
<b>Find Class...</b>	Requests a class name, then selects the matching symbol dictionary and class entry in the class list.

## The Class Menu

The Class menu is active only when a class has been selected. The following commands are available from the Class menu. (Defining a New Class explains how to define a new GemStone Smalltalk class to add to the currently selected symbol dictionary.)

<b>File Out As...</b>	Requests a file name, then creates a fileout (.gs file) containing the selected class and its methods
<b>Browse &gt; Class</b>	Opens a Class Browser on the selected class
<b>Browse &gt; Hierarchy</b>	Opens a Class Hierarchy Browser on the selected class.
<b>Browse &gt; Versions</b>	Opens a Class Version Browser on the selected class. The browser shows how many versions of that class exist.
<b>References To &gt; Instance Variable</b>	Opens a selection dialog allowing you to choose from a list of instance variables of the selected class. Choose one and click <b>OK</b> or press Return, and a Method List Browser opens showing all methods that refer to the selected instance variable.

<b>References To &gt; Class Variable</b>	Opens a selection dialog allowing you to choose from a list of class variables of the selected class. Choose one and click <b>OK</b> or press Return, and a Method List Browser opens showing all methods that refer to the selected class variable.
<b>References To &gt; Class</b>	Opens a Method List Browser showing all methods that refer to the selected class.
<b>Show &gt; Hierarchy</b>	Lists the superclasses of the selected class and their instance variables. For example, if the current class is <code>PositionableStream</code> , the hierarchy list appears as follows: Object () Stream () PositionableStream (itsCollection position) ReadStream () WriteStream () The names in parentheses in this example show that <code>PositionableStream</code> has two named instance variables.
<b>Show &gt; Definition</b>	Displays the class definition of the selected class in the text pane.
<b>Show &gt; Comment</b>	Displays the class comment of the currently selected GemStone-supplied class
<b>Move to...</b>	Lets you choose another symbol dictionary, then moves the selected class to that dictionary.
<b>Remove</b>	After confirmation, removes the selected class from its associated symbol dictionary <i>CAUTION: To avoid inadvertently removing or modifying a GemStone kernel class, use the DataCurator account for all administrative tasks except those that require SystemUser privileges, such as upgrading the repository.</i>
<b>New Template</b>	Presents a class definition template in the text pane.

## The Protocol Menu

To help you organize your work, GemStone Smalltalk methods that are functionally similar are grouped together into protocol categories with descriptive names, such as *initializing*, *accessing*, or *updating*. These categories are descriptive only; they do not affect the operation of GemStone Smalltalk in any way.

The following items are available in the Protocol menu. (Until you've selected a method category, Add is the only menu item enabled.)

<b>Add...</b>	Requests a protocol name, then adds the protocol to the selected class.
<b>Rename As...</b>	Requests a protocol name, then renames the selected protocol to that name.
<b>Remove...</b>	Removes the selected protocol (after confirmation, if it contains any methods).
<b>Find Method...</b>	Provides a list of all method names in the selected class, then updates the protocol and method panes to show the selected protocol and method.

## The Method Menu

The Method menu allows you to file out a single method, reorganize methods into different categories, and remove a method. Items in the Method menu are available only when a method has been selected.

<b>File Out As...</b>	Requests a file name, then creates a fileout (.gs file) containing the selected method.
<b>Browse Method</b>	Opens a GemStone Method Browser on the selected method.
<b>Senders</b>	Opens a GemStone Method List Browser showing all senders of the selected method (same as clicking the <b>Senders...</b> button in the Launcher and entering the name of the method).
<b>Implementors</b>	Opens a GemStone Method List Browser showing all implementors of the selected method (same as clicking the <b>Implementors...</b> button in the Launcher and entering the name of the method).

<b>Messages...</b>	After you choose one of the messages sent by the selected method, opens a GemStone Method List Browser showing all implementors.
<b>Move to...</b>	Requests the name of a protocol to which the selected method should be moved, then moves the method. If no protocol exists by the name supplied, this menu item adds the protocol and then does the move.
<b>Remove...</b>	After confirmation, removes the selected method.
<b>View &gt; Class Only</b>	Displays protocols and methods for the selected class only. The selected view persists when you select a new class.
<b>View &gt; Up To Root</b>	Displays protocols and methods for the selected class and all superclasses up to, but not including, the class at the root of the hierarchy (typically Object). The selected view persists when you select a new class.
<b>View &gt; Including Root</b>	Displays protocols and methods for the selected class and all superclasses, including the class at the root of the hierarchy (typically Object). The selected view persists when you select a new class.
<b>New Template</b>	Presents a new method template in the text pane.

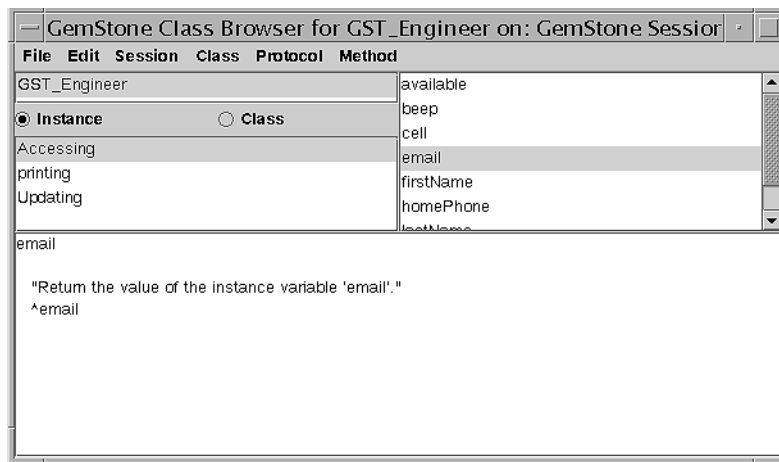
## Specialized Browsers

GemBuilder for Java Tools implement several specialized browsers based on the GemStone Browser: the Class Browser, Hierarchy Browser, Class Version Browser, and Method List Browser.

### The Class Browser

The GemStone Class Browser displays a single GemStone class. The menu items supported by the browser are the same as those supported by the System Browser except the Dictionary menu is not available. Open this browser by choosing **Class > Browse > Class** in the GemStone Browser or by choosing **Tools > Browse Class...** in the GemStone Launcher.

You can use wild-card characters in searches, with "\*" matching any number of characters, and "?" matching a single character.

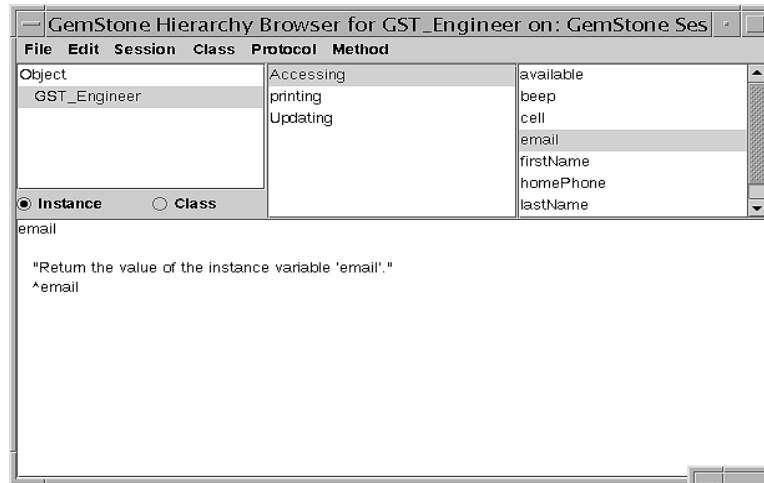
**Figure 3.2 Class Browser**

## The Hierarchy Browser

The GemStone Hierarchy Browser displays the superclass chain of a target class plus all subclasses of the target class. For example, choosing **Class > Browse > Hierarchy** when class Company is selected allows you to browse the hierarchy for that class, which happens to be a direct subclass of Object. The menu items supported by the browser are the same as those supported by the System Browser except the Dictionary menu is not available.

You can use wild-card characters in searches, with "\*" matching any number of characters, and "?" matching a single character.

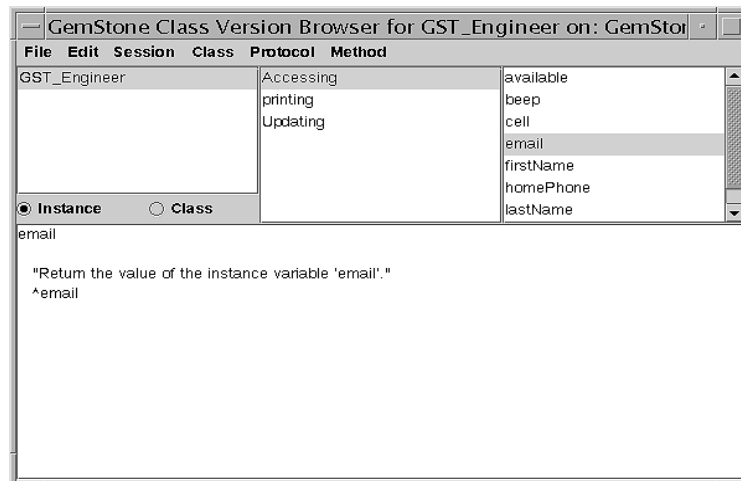
Figure 3.3 Class Hierarchy Browser



## The Class Version Browser

The GemStone Class Version Browser displays the versions of a class in the class history of the target class. For example, doing **Class > Browse Versions** when class **Engineer** is selected in the GemStone Browser shows two versions exist. You can select either version in the left pane. The browser displays the definition of the selected version in the text pane.

Figure 3.4 Class Version Browser



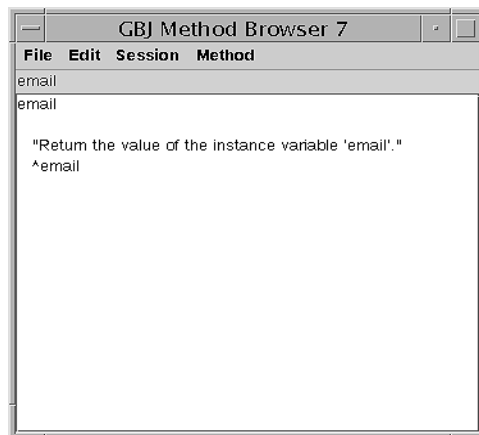
The menu items supported by the Class Version Browser are generally the same as those supported by the System Browser. The Class menu has been extended, and several of the menu items have slightly different semantics. The additional and altered items in the Class menu are these:

<b>Class &gt; Inspect Instances</b>	Opens a GemStone Inspector on all instances of the currently selected class version. Because this operation can be lengthy, you are asked to confirm your intent.
<b>Class &gt; Migrate Instances</b>	After you choose a target class version, migrates all instances of the selected version to the target version.
<b>Class &gt; Move to...</b>	Requests the name of a target class, then moves the selected class version to the class history of the target class.
<b>Class &gt; Remove...</b>	After confirmation, removes the selected class version from its class history.

## The Method Browser

The Method Browser displays an individual method. For example, invoking the Method browser on `Engineer > firstName`: shows its definition.

**Figure 3.5** Method Browser



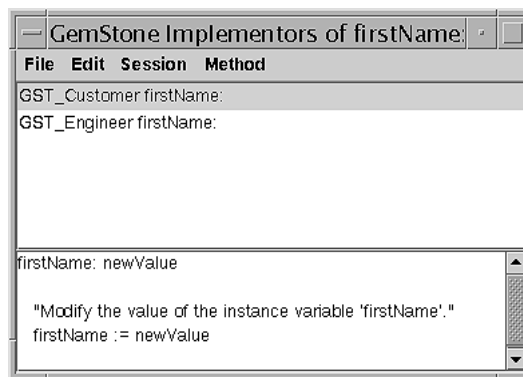
The supported menu items are the same as those for the other browsers except the Dictionary, Class and Protocol menus are not available.

## The Method List Browser

The Method List Browser displays a list of methods defined in various classes. The browser is used to display senders and implementors of various messages. For example, invoking the menu item Implementors on `firstName:` lets you browse each implementation of that message. The same list can be obtained by clicking the Implementors... button in the GemStone Launcher.

You can use wild-card characters in searches, with "\*" matching any number of characters, and "?" matching a single character.

**Figure 3.6 Implementors of a Particular Method**



## The Breakpoint Browser

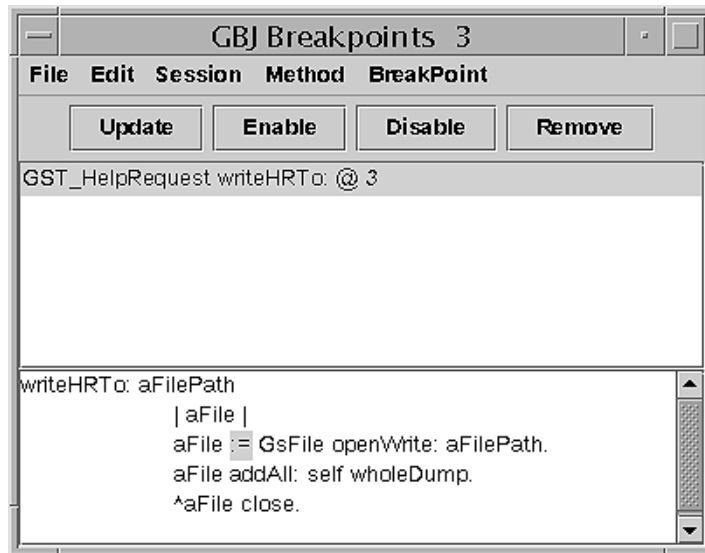
The Breakpoint Browser displays a list of methods for which breakpoints have been specified. Open it using the **Tools > Breakpoints** menu item in the GemStone Launcher. You can then select a breakpoint and enable, disable, or remove it.

To set a breakpoint:

1. Open any tool that allows you to browse method source code.
2. Place the insertion point before the point at which you wish execution to halt.
3. Select **Edit > Set Break**.

When a breakpoint is encountered, a notifier allows you to open a debugger. See The Debugger.

The Breakpoint Browser appears as shown below:

**Figure 3.7 Viewing Breakpoints**

The top pane contains a list of all breakpoints set in GemStone Smalltalk code. The bottom pane shows the source for the selected breakpoint, highlighting the message-send at which execution will stop when the breakpoint is encountered.

### Breakpoint Browser Buttons

Four buttons provide convenient access to the main functions of the Breakpoint Browser.

The **Update** button updates the Breakpoint Browser in case breakpoints have been set in another tool since the browser was opened.

The **Remove** button removes the selected breakpoint.

The **Enable** button enables the selected breakpoint, causing execution to stop when it is reached.

The **Disable** button disables the selected breakpoint without removing it.

## Breakpoint Menu

The Breakpoint Browser offers one new menu, the Breakpoint menu. It contains these items:

<b>Remove Break</b>	Removes the currently selected breakpoint from the browser and the system. This is the same as clicking the <b>Remove</b> button.
<b>Enable Break</b>	Enables the currently selected breakpoint, causing execution to halt when it is reached. This is the same as clicking the <b>Enable</b> button.
<b>Disable Break</b>	Disables the currently selected breakpoint, causing execution to continue uninterrupted when it is reached, but leaving the breakpoint in both the system and the browser for possible future use. This is the same as clicking the <b>Disable</b> button.
<b>Remove All Breaks</b>	Removes all breakpoints from the browser and the system.
<b>Enable All Breaks</b>	Enables all breakpoints, causing execution to halt when they are reached.
<b>Disable All Breaks</b>	Disables all breakpoints, causing execution to continue uninterrupted when they are reached, but leaving them in both the system and the browser for possible future use.

---

## Overview

Three additional tools provide assistance in working with GemStone Smalltalk:

- *Workspaces* are text areas in which Smalltalk expressions can be evaluated. You can also use a workspace to file in code previously saved from GemStone/S.
- *Inspectors* let you examine GemStone/S objects, including the result of evaluating an expression in a workspace.
- The *Debugger* lets you inspect the state of Smalltalk execution at the time of an error.

In addition, certain menus and keyboard shortcuts are common to all the tools.

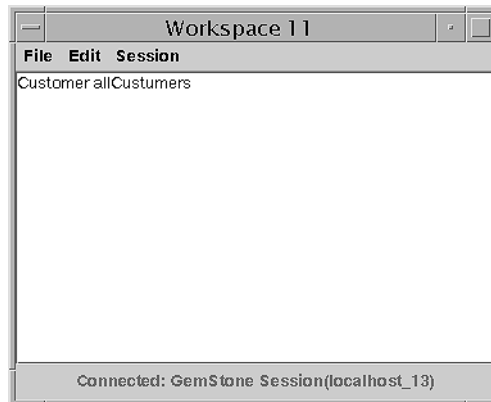
## The Workspace

The GemStone Workspace provides a text area in which expressions can be evaluated. For example, you can evaluate the expression

```
Customer allCustomers
```

by highlighting the expression and then choosing **Edit > Print It** or a similar menu item.

**Figure 4.1** A Workspace



The menu items available in the Workspace menu are similar to those in the Browser and Launcher:

<b>File &gt; Open...</b>	Requests a file name, then copies the contents into the workspace.
<b>File &gt; Save As...</b>	Requests a file name, then writes the contents of the workspace into the specified file.
<b>File &gt; File In...</b>	Opens a file dialog allowing you to specify a source code file. Treats the selected file as being in GemStone/S file-in format (see Reading and Compiling a Saved File) and files it in.
<b>File &gt; Close</b>	Closes the workspace.
<b>Edit &gt; Cut</b>	Cuts the currently selected text from the text pane and puts it in the clipboard.
<b>Edit &gt; Copy</b>	Copies the currently selected text from the text pane and puts it in the clipboard.

---

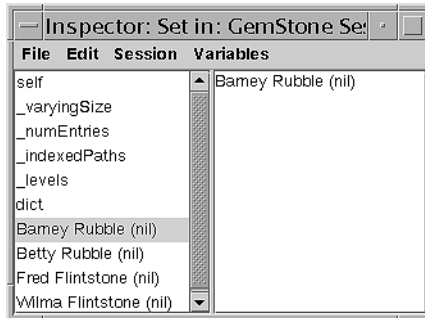
<b>Edit &gt; Paste</b>	Pastes the text from the clipboard at the currently selected position in the text pane.
<b>Edit &gt; Select All</b>	Selects all of the text in the text pane.
<b>Edit &gt; Do It</b>	Uses the GemStone/S compiler to evaluate the currently selected text.
<b>Edit &gt; Print It</b>	Performs a <b>Do It</b> , sends the resulting object the message <code>printString</code> , and then places the resulting string after the current text pane selection.
<b>Edit &gt; Inspect It</b>	Performs a <b>Do It</b> , then opens a GemStone Inspector on the resulting object.
<b>Edit &gt; File It In</b>	Treats the selected text as though it is in GemStone/S file-in format, and tries to file it in to GemStone/S.
<b>Session &gt; Commit</b>	Commits the Launcher's session.
<b>Session &gt; Abort</b>	Aborts the Launcher's session.
<b>Session &gt; Begin</b>	Begins a new transaction.

## The Inspector

When you select a GemStone Smalltalk expression and execute **Inspect It** from the GemStone menu, a GemStone Inspector opens. This inspector contains two panes, which enable you to examine and modify the values of instance variables in the GemStone/S object returned as the result of the expression.

The left pane lists `self`, which refers to the inspected object, together with the names or indexes of the object's instance variables. (If the object is an instance of a Dictionary, the left pane lists the keys). When you select an item in the left pane, the right pane shows the value of that item. For example, invoking an Inspector on the expression `Customer allCustomers` produces this inspector:

Figure 4.2 An Inspector



You can use the right pane to execute GemStone Smalltalk expressions that contain names appearing in the left pane. For example, if the inspected object had an instance variable named *count* whose value was 5, then executing the expression `count * 5` in the right pane returns the value 25.

The menu items supported by the GemStone Inspector are shown below.

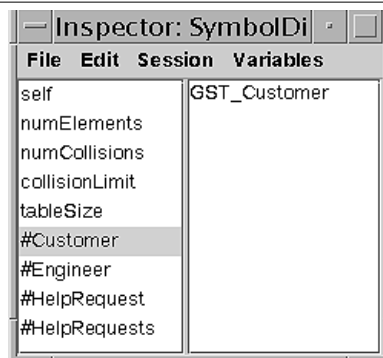
<b>File &gt; Update</b>	Updates the inspector to match the current state of the Launcher's session.
<b>File &gt; Close</b>	Closes the inspector window.
<b>Edit &gt; Cut</b>	Cuts the currently selected text from the text pane and puts it in the clipboard.
<b>Edit &gt; Copy</b>	Copies the currently selected text from the text pane and puts it in the clipboard.
<b>Edit &gt; Paste</b>	Pastes the text from the clipboard at the current selection position in the text pane.
<b>Edit &gt; Select All</b>	Selects all of the text in the text pane.
<b>Edit &gt; Do It</b>	Uses the GemStone/S compiler to evaluate the currently selected text.
<b>Edit &gt; Print It</b>	Performs a <b>Do It</b> , sends the resulting object the message <code>printString</code> , and then places the resulting string after the current text pane selection.
<b>Edit &gt; Inspect It</b>	Performs a <b>Do It</b> , then opens a GemStone Inspector on the resulting object.

<b>Edit &gt; Compile</b>	Compiles the current text in the text pane. The text in the text pane is evaluated, and the resulting object is assigned to the selected slot of the object being inspected.
<b>Edit &gt; Cancel</b>	Cancels any text entry done by the user in the text pane, and return the inspector text pane to its unmodified form (which may be to show the <code>printString</code> of the selected slot).
<b>Session &gt; Commit</b>	Commits the Launcher's session.
<b>Session &gt; Abort</b>	Aborts the Launcher's session.
<b>Session &gt; Begin</b>	Begins a new transaction.
<b>Variables &gt; Inspect</b>	Opens a new inspector on the selected item from the inspector list pane.
<b>Variables &gt; More</b>	Retrieves more of the object's indexable fields. By default, the first 100 are retrieved. Each pick of this menu item doubles the number of fields that are retrieved.

## Inspecting Dictionaries and Sets

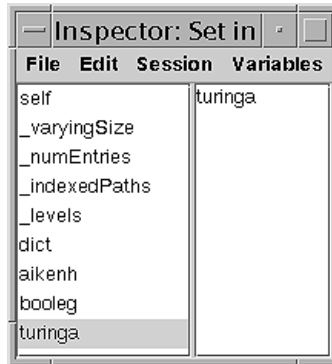
When you inspect a Dictionary and select an association in the left pane, its value is shown in the right pane. For example, opening the inspector on the `HelpRequestSys` dictionary and selecting `#Customer` shows the value, the `Customer` class.

**Figure 4.3** Inspecting a Dictionary



When you inspect the Set returned by evaluating `Engineer allEngineers`, the elements (the engineers' login names) are shown in the left pane.

**Figure 4.4** Inspecting a Set



## The Debugger

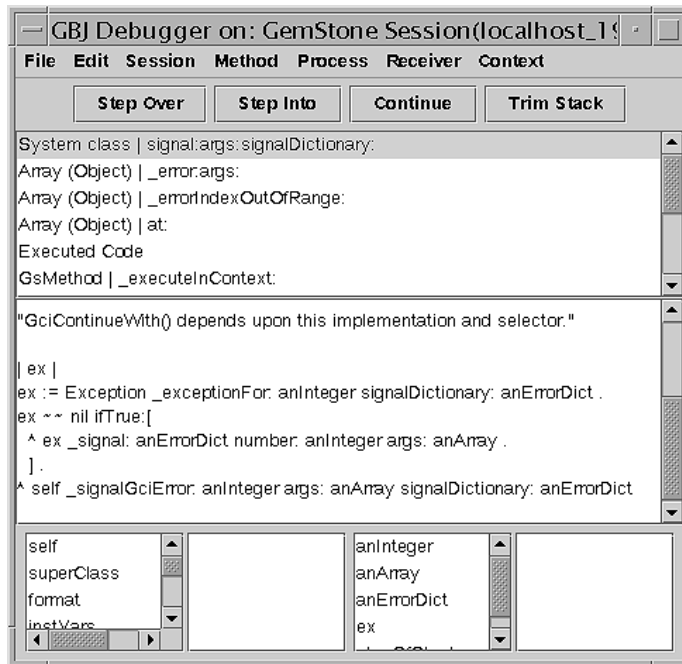
The Debugger can be invoked either from an error notifier or by encountering a breakpoint while running code from a workspace.

For example, the following sequence results from evaluating the expression

```
Array new at: 1
```

with **Do It** in a workspace. When the error notifier appears, clicking the **Debug** button brings up a Debugger.

Figure 4.5 Debugger



The debugger consists of six panes. At the top is the *stack pane*, which shows the stack as it was when execution halted due to an error or breakpoint. Select a specific message in the stack, and the *source pane* below shows the source code associated with it.

The bottom four panes are actually two inspectors. The inspector on the left lists the receiver and instance variables of the selected message. Selecting one causes its value to appear in the pane directly to the right. The inspector on the right lists the variables referred to in the selected message. Selecting one causes its value to appear in the rightmost pane.

## Debugger Buttons

Four debugger buttons provide convenient access to common debugging functions.

**Step Over** steps process execution over the next message-send highlighted in the source pane. Control then returns to the debugger with the next step point highlighted.

**Step Into** steps process execution into the next message-send highlighted in the stack pane. Control typically returns to the debugger with a new message on top of the stack. Execution is halted at the beginning of the new method in the source pane.

**Continue** continues executing the process and closes the debugger.

**Trim Stack** trims messages from the stack, starting at the top and trimming until the selected message is reached. The next step point is at the beginning of the new top of the stack.

## Debugger Menus

The debugger File, Edit, Session, and Method menus are all similar to those in other tools. The Process menu contains these items:

<b>Step Over</b>	Steps process execution over the next message-send highlighted in the source pane. Control then returns to the debugger with the next step point highlighted. This is the same as clicking the <b>Step Over</b> button.
<b>Step Into</b>	Steps process execution into the next message-send highlighted in the stack pane. Control typically returns to the debugger with a new message on top of the stack. Execution is halted at the beginning of the new method in the source pane. This is the same as clicking the <b>Step Into</b> button.
<b>Explain</b>	Prints the original descriptive error text in the source pane.
<b>Continue</b>	Continues executing the process and closes the debugger. This is the same as clicking the <b>Continue</b> button.

<b>Trim Stack</b>	Trims messages from the stack, starting at the top and trimming until the selected message is reached. The next step point is at the beginning of the new top of the stack. This is the same as clicking the <b>Trim Stack</b> button.
<b>Copy Stack</b>	Creates a text representation of the stack, as seen in the stack pane, and places it in the clipboard.

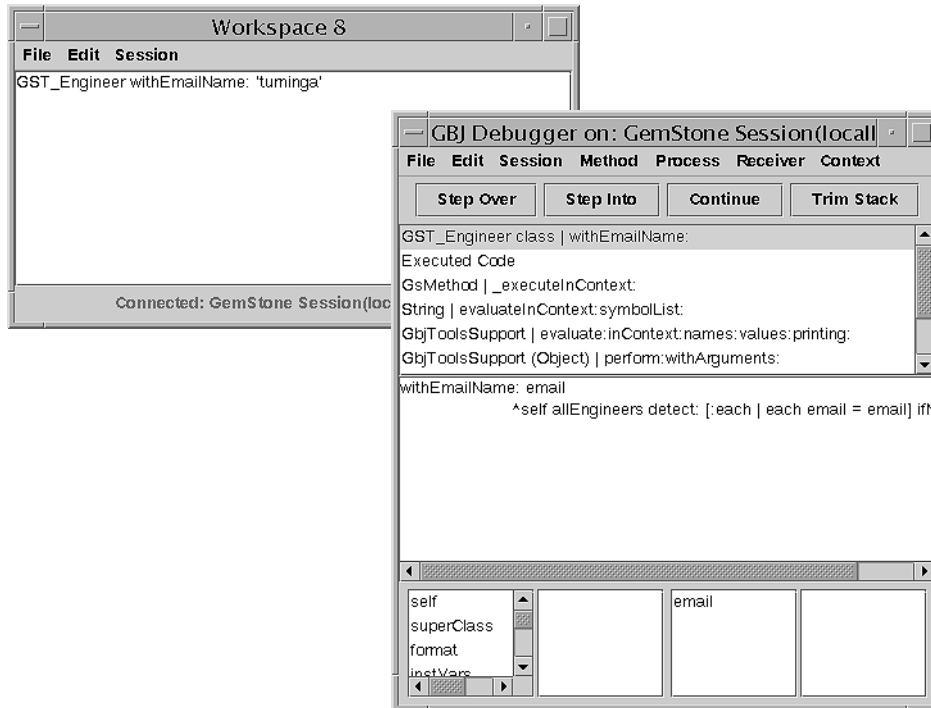
The Receiver and Context menus have one item each:

<b>Inspect</b>	Opens an Inspector on the value selected in the leftmost bottom pane, in the case of the Receiver menu, or the bottom pane second from the right, in the case of the Context menu.
----------------	--

## Invoking the Debugger from a Workspace

The ordinary way to invoke the debugger is to set a breakpoint in your Smalltalk code, then invoke that code by evaluating an expression in the workspace. For example:

1. Set the breakpoint:
  - a. Open any source code browser on the appropriate class, such as `Engineer`.
  - b. Select the desired protocol and method.
  - c. Place the insertion point before the place where you want execution to halt, then choose **Edit > Set Break**. (To view existing breakpoints, see The Breakpoint Browser.)
2. Open a workspace and enter an expression that invokes code where the breakpoint is set. Select the expression and choose **Edit > Do It**.
3. When the breakpoint notifier appears, click **Debug**. (**Continue** continues the current process without opening the debugger. **Terminate** ends the current process.

**Figure 4.6 Debugging Engineer | withEmailName:**

## Invoking the Debugger from Your Java Client

To invoke the debugger from your running application, you must set a breakpoint within Smalltalk code that is called within a Java `try` block. When GemStone/S encounters the breakpoint, it returns an exception to the client. The catch block uses the exception as an argument to open an instance of `GbjDebugger`. For example, this sequence stops at a breakpoint in `Engineer | email:firstName:lastName:phone:`

1. Find the Java code that calls the Smalltalk code in which the breakpoint will be set. Open a `GbjDebugger` in the catch block for that code.

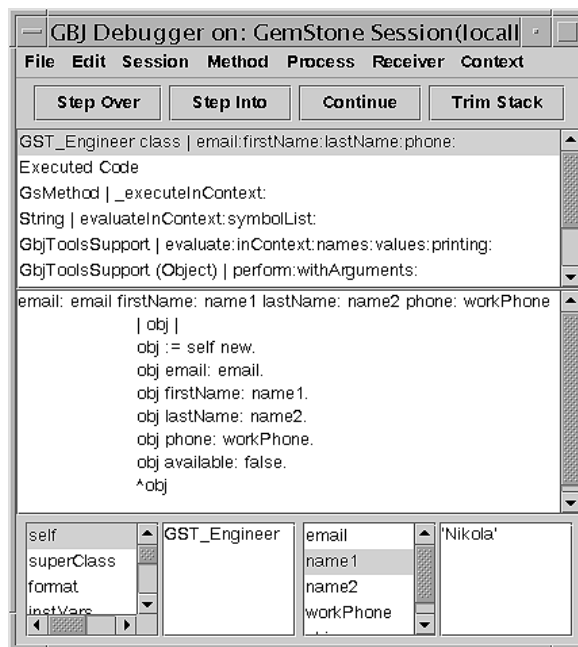
```
try {
    eng = (Engineer) engClass.perform(
        "email:firstName:lastName:phone:", args, 4);
} catch (GbjEventException e) {
    new com.gemstone.tools.GbjDebugger(e);
}
```

2. Have the application open a `GbjLauncher` so you can set the breakpoint, which must be set within the session in which the application is running. This example opens the Launcher only when the system property `mydebug` is true. The object `session` is an instance of `GbjSession`.

```
if (Boolean.getBoolean("mydebug"))
    new GbjLauncher(session);
```

3. Start the application. When the Launcher appears, open a source code browser and set the breakpoint as explained in the previous topic. For our example, the command line is
4. Run the part of the application that calls the Smalltalk code where the breakpoint is set. When the breakpoint is encountered, the debugger will open.

**Figure 4.7 Debugger Invoked from a Running Application**



You can step through the Smalltalk execution as desired. Because the client Java thread is not suspended, however, it continues execution and may encounter a `NullPointerException` on a object that was not returned from GemStone/S.

## Common Menus and Shortcuts

All text panes in all GemBuilder for Java tools have a popup menu allowing you to perform common operations. The popup menu differs slightly, depending on which tool the text pane belongs to.

To access the popup menu, place the cursor in a text pane and click the right mouse button. To dismiss the popup menu without selecting an item, move the cursor off the menu and click the left mouse button.

### The Popup Menu

The popup menu items are:

<b>Cut</b>	Cuts the selected text and places it in the clipboard.
<b>Copy</b>	Copies the selected text and places it in the clipboard.
<b>Paste</b>	Pastes the text from the clipboard at the current insertion point.
<b>Select All</b>	Selects all the text in the pane.
<b>Do It</b>	Uses the GemStone/S compiler to evaluate the currently selected text.
<b>Print It</b>	Performs a <b>Do It</b> , sends the resulting object the message <code>printString</code> , and then places the resulting string after the current text pane selection.
<b>Inspect It</b>	Performs a <b>Do It</b> , then opens a GemStone Inspector on the resulting object.
<b>Compile</b>	Compiles the selected source code. This menu item is available only in those tools for which it is relevant.
<b>Cancel</b>	Cancels any text you entered in the text pane, returning it to its unmodified form. This menu item is available only in those tools for which it is relevant.
<b>Set Break</b>	Sets a breakpoint at the current insertion point in the current method. This menu item is available only in those tools for which it is relevant.

## Keyboard Shortcuts

The following keyboard shortcuts are available:

<b>Edit &gt; Cut</b>	Control-X
<b>Edit &gt; Copy</b>	Control-C
<b>Edit &gt; Paste</b>	Control-V
<b>Edit &gt; Select All</b>	Control-A
<b>Edit &gt; Do It</b>	Control-D
<b>Edit &gt; Print It</b>	Control-P
<b>Edit &gt; Inspect It</b>	Control-I
<b>Edit &gt; Compile</b>	Control-S
<b>Session &gt; Commit</b>	Control-Shift-S



# *Working with GemStone Classes and Methods*

---

## **About GemStone Classes**

### **Instance Variables Can Be Constrained**

To speed GemStone Smalltalk's indexed associative access for efficient querying, you can constrain the value of an instance variable to contain only specified kinds of objects. (Class variables can also be constrained.)

Constraining a variable means that its value is always either an instance of the specified class, a subclass thereof, or nil.

Constraints can be circular. You can constrain an instance variable to be an instance of its own class. You can also constrain instance variables of two classes to each hold instances of the other.

### **Constraints Are Inherited**

Constraints, like instance variables, are inherited by subclasses: when you define a subclass, its inherited instance variables by default bear the same constraints as specified in the superclass from which they are inherited.

However, inherited instance variables can be further constrained in a subclass. In this case, the instance variable's new constraint must be a subclass of that specified by the inherited constraint.

To further constrain inherited instance variables, specify the name of the inherited variable and its new constraint in the argument to the `constraints:` keyword in the class definition template. For example, suppose you have defined a class `Employee` with instance variables named `jobTitle` and `department` that are constrained to be `Strings`. You can now create a subclass of `Employee` named `FormerEmployee` and constrain the inherited variables `jobTitle` and `department` to be `InvariantStrings`. `FormerEmployee`'s new instance variables can be constrained or not, as you require, and its other inherited instance variables retain whatever constraints were set in the superclass that defined them.

## Instances Can Be Made Invariant

The definition of the class can specify that all instances of the class are invariant. An invariant object can be modified only during the transaction in which it is created. After the transaction is committed, you can no longer modify its instance variables, nor the size or class of the object. You can specify invariance for all instances of a class by providing the argument `true` to the `instancesInvariant:` keyword in the class definition template. Class-level invariance is useful for supporting literals in methods and in other limited situations, but it is generally more cumbersome than object-level invariance.

Any object can be made invariant by sending it the message `immediateInvariant`. This mechanism protects objects from being modified and can be useful for maintaining the integrity of your repository. After `immediateInvariant` is sent to an object, you can no longer modify its instance variables, nor the size or class of the object. The effect of the `immediateInvariant` message is not reversible. The message `isInvariant` returns `true` if the receiver is invariant; `false` otherwise.

## Reserved Selectors

A small number of selectors are reserved for sole use by the GemStone kernel classes. Do not create new methods with these selectors. If you do, the Smalltalk compiler will not execute your code.

and:	ifTrue:ifFalse:	to:do
downTo:by:do:	isNil	untilFalse
downTo:do:	notNil	untilTrue
ifFalse:	or:	whileFalse:
ifFalse:ifTrue:	timesRepeat:	whileTrue:
ifTrue:	to:by:do:	

## Optimized Selectors

Certain selectors are optimized in Smalltalk kernel classes. Redefining an optimized selector in the class for which it is optimized has no effect; the same primitive method will be called and your redefinition will be ignored. However, you can redefine them in other classes, such as those comprising your application.

Class	Selector
SmallInteger	+
SmallInteger	-
SmallInteger	*
SmallInteger	>=
SmallInteger	=
Any kernel class	==
Any kernel class	~~
Any kernel class	_class
Any kernel class	isKindOf:
Any kernel class	_disableProtectedMethod
Any kernel class	_gsReturnNoResult

## Defining a New Class

This topic explains how to define a class in GemStone and gives an example. It also discusses several related topics.

### To Define the Class

1. Open a GemStone Browser if one is not already open.
2. Make sure no classes are selected in the class list, and, in the Symbol List pane, select the dictionary in which you wish to refer to the new class. The browser responds by displaying a template:

```
NameOfSuperclass subclass: 'NameOfClass'  
  instVarNames: #('instVarName1' 'instVarName2')  
  classVars: #('ClassVarName1' 'ClassVarName2')  
  classInstVars #('ClassInstVarName1'  
    'ClassInstVarName2')  
  poolDictionaries: #[]  
  inDictionary: aDictionary  
  constraints: #[]  
  instancesInvariant: false  
  isModifiable: false
```

3. Replace *NameOfSuperclass* with the name of your new class's immediate superclass. You cannot create subclasses of certain GemStone kernel classes. A note to that effect is included in the descriptions of those classes in the *GemStone Programming Guide*.
4. Replace *NameOfClass* with the name of the new class. By convention, the first character in each GemStone class name is capitalized.
5. Replace *instVarNameX* with the names of any instance variables, or delete all the text within the parentheses if your new class has no instance variables. A class can define up to 255 instance variables to which you can refer by name in that class's methods.
6. Replace *classVarNamesX* with the names of any class variables, or delete all the text within the parentheses if your new class has no class variables. Replace *classInstVarNames* with the names of any class instance variables, or delete all the text within the parentheses if your new class has no class instance variables.
7. Fill in the brackets after the `poolDictionaries:` keyword with any pool dictionaries that you want the class to access.
8. Examine the name of the dictionary after the `inDictionary:` keyword. Unless you replace the inserted text with the name of another symbol dictionary to which you have access, your new class is defined in that dictionary.

9. Fill in the brackets after the `constraints:` keyword with any constraints you wish to specify for one or more of the instance variables. If your class is a collection subclass and you wish to constrain its elements, simply type the name of the constraint class inside the brackets. For example, to constrain a Bag subclass `BagOfEmployees` to contain only instances of the class `Employee`, type:

```
constraints: #[Employee]
```

10. However, if you wish to constrain a named instance variable, type the name of the variable and the constraint class as a pair, separated by a comma, each within its own set of square brackets, also separated by commas. Preface the instance variable name with a `#`. For example, to constrain the instance variables name and address of the class `Employee` to be a `String` and an instance of the class `Address`, respectively, type:

```
constraints: #[ #[ #name, String], #[ #address, Address] ]
```

11. After the `instancesInvariant:` keyword, specify whether instances of the class are modifiable. The default value is `false`. Change this to `true` if you wish instances to be invariant.
12. After the `isModifiable:` keyword, specify whether the structure of the class can be modified. The default value is `false`. Change this to `true` if you wish the class to be invariant.

## InstVars

Named instance variables are variables whose name is shared by all instances of a class and all instances of its subclasses. Each instance, however, holds a distinct value for the variable

## ClassVars

A class variable is a variable whose name and value are shared by a class, all of its instances, its subclasses, and all of their instances. Both class and instance methods of the class and its subclasses can refer to the variable.

## ClassInstVars

A class instance variable is a variable whose name and value are shared by a class and all of its instances.

## PoolDicts

Pool dictionaries are special-purpose storage structures that enable any arbitrary group of classes and their instances to share information

## Example of a Class Definition

### Example 5.1

---

```
Object subclass: 'Engineer'  
  instVarNames: #('firstName' 'lastName' 'email' 'phone' 'homePhone' 'beep'  
    'cell' 'available')  
  classVars: #( 'AllEngineers')  
  classInstVars: #()  
  poolDictionaries: #[]  
  inDictionary: Published  
  constraints: #[]  
  instancesInvariant: false  
  isModifiable: true
```

---

Here is an explanation of each line:

```
Object subclass: 'Engineer'
```

Engineer is a direct subclass of Object.

```
instVarNames: #('firstName' 'lastName' 'email' 'phone' 'homePhone'  
'beep' 'cell' 'available')
```

The Engineer class has eight named instance variables: firstName, lastName, email, phone, homePhone, beep, cell, and available.

```
classVars: #( 'AllEngineers')
```

This class has one class variable, AllEngineers.

```
classInstVars: #()
```

This class has no class instance variables.

```
poolDictionaries: #[]
```

This class need not access any pool dictionaries.

```
inDictionary: Published
```

Engineer resides in the global Published dictionary.

```
constraints: #[]
```

No constraints have been placed on the instance variables or the class variable.

```
instancesInvariant: false
```

When instances of the class are created, their values will be modifiable even after they have been committed to the repository.

```
isModifiable: true)
```

This class is modifiable; instance variables can still be added, removed, and constrained, and class or class instance variables can be added. As long as the class itself remains modifiable, however, no instances of it can be created.

## Private Instance Variables

In addition to the private methods discussed earlier, you also see private instance variables occasionally in the GemStone kernel classes. For example, the GemStone Bag class defines four private instance variables that are used by the object manager and primitives to implement features of UnorderedCollections, such as adding indexing structures that allow efficient querying. The names of private instance variables are symbols beginning with an underscore (`_`). Private instance variables cannot be modified or constrained when defining subclasses. An attempt to place a constraint on a private instance variable generates an error.

## Subclass Creation Methods

You can choose from a variety of subclass creation messages, depending on the type of class you want to create. For example, if you wish to create a byte subclass, replace the initial keyword `subclass:` with the keyword `byteSubclass:.` If the superclass is not a subclass of String, instances of the new class store and return SmallIntegers in the range 0–255.

Similarly, if you wish to create an indexable subclass, replace the initial keyword `subclass:` with the keyword `indexableSubclass:.` Instances of the new class are represented as pointer objects.

For complete descriptions of the different kinds of classes, see the section describing class storage format in the *GemStone Programming Guide*.

If you wish to set the class history of your new class explicitly, you can include the keyword `newVersionOf:` in the class definition template or any subclass creation message, after `instancesInvariant:.` If the argument to this keyword is a class, this method creates the new class as a new version of that class, and the two classes share a class history. In this way, you can make one class a new version of another even if they do not have the same name.

If the argument to the `newVersionOf:` keyword is nil, the new class is created with a new class history.

If you do not include the `newVersionOf:` keyword, the compiler checks to see if another class having the same name already exists. If it does, the new class is compiled as a new version of

the other class and shares its class history. If it does not, the new class is created with a new class history.

For more discussion of class histories, see the section describing class histories in the *GemStone Programming Guide*.

## Compiling the Class Definition

After you have edited the class template to your satisfaction, choose **Edit > Compile**. When you do this, the GemStone Smalltalk compiler evaluates the class definition template in the browser's text pane as a subclass creation message. For this reason, make sure you don't have any extraneous text in the text pane when you choose **Compile**.

## If an Error Occurs

If the class definition does not compile properly, an error message is inserted in the browser's text pane. Delete the error message, then edit your subclass creation message and try to compile the class definition again.

If the new class object cannot be created due to a run-time error, an error notifier window is displayed. Edit the class creation message and try again.

GemStone supports modification of the schema when you already have instances of invariant classes populating your repository and you discover that you must redefine some of your classes.

## Modifying an Existing Class

If you select an existing class, then modify and save the class definition, you are creating a new version of the class and all of its subclasses. The browser attempts to recompile all methods from the previous version into the new version. Methods that fail to recompile are presented in a method list browser, from which you can correct the errors. If the class has subclasses, they are also versioned and their methods recompiled.

Versioning a class does not cause its instances to be migrated to the new class. They are still instances of the old class. You can migrate some or all instances of one version of a class to another version when you need to.

For more information on migrating instances, see the section on class histories in the *GemStone Programming Guide*.

To create a new version of a class, select the class's name in the browser to bring up its definition in the text pane. Edit the definition and select **Edit > Compile**. Whenever you create a class with the same name as a class that already exists in one of your symbol dictionaries, the

new class is automatically created as the latest version of the existing class and it automatically shares the same class history. Instances created after the redefinition have the new class's structure and access the new class's methods. Instances that were created earlier have the old class's structure and access the old class's methods, but they can be migrated to the new class.

Let's assume that you have a class named `Vendor` with instance variables for `id`, `name`, `address`, `city`, `state`, `zipCode`, and `phone`. Five of `Vendor`'s instance variables are constrained to be instances of class `String`, and one (`zipcode`) is constrained to be a `SmallInteger`. Suppose that you decide that the class needs an additional instance variable named `fax` to represent the `Vendor`'s fax number.

To do this, you can define a new version of the class `Vendor` to include the new instance variable. Keeping the same name as the old class ensures that it shares the same class history as the previous version.

After you compile the class definition, the new class is named `Vendor`, and all of the original instance and class methods are copied to the new class. Any existing instances will still belong to the original class and may have to be migrated to the new class.

## Defining Methods

*NOTE:*

*You can modify only methods for which you have write authorization—for example, methods that you have written for your own classes. You cannot modify any GemStone kernel class method—that is, any method that is defined for one of the predefined classes supplied with the GemStone system.*

To define a new method, select the appropriate category and either select no method or execute **Method > New Template**. The following template is displayed for you:

### Example 5.2

```
message selector and argument names
  "comment stating purpose of message"

  | temporary variable names |
  statements
```

Replace the template text as indicated. After you have completed the method definition, choose **Edit > Compile**. If the new method compiles successfully, it is inserted into the list of methods for the selected category and class. Otherwise, the first compiler error encountered is inserted into the text at the point of error and becomes the new text selection.

## Public and Private Methods

Most of the methods for GemStone kernel classes are public; that is, they are supported methods that appear in the manual. Along with these public methods, you will also see methods whose selector begins with an underscore character (`_`). These are private methods that are implemented by system developers to support the public protocol. Unlike the public methods, private methods are not supported and are subject to change; they are reserved for use by GemStone developers.

Because Smalltalk is an open system, private methods are listed along with the public methods for the associated category. As with public methods, you can display the text of private methods in the GemStone Browser's text pane. Although it may be interesting to review the definition of private methods, observe the following caution:

*CAUTION:*

*Private methods are subject to change at any time. Do not depend on the presence or specific implementation of these private methods when creating your own classes and methods.*

## Saving Class and Method Definitions in a File

It's often useful to store the GemStone Smalltalk source code for your classes and methods in ordinary files. This process is called filing out source code. Such files make it easy to:

- transport your code to other GemStone systems,
- perform global edits and recompilations,
- produce paper copies of your work, and
- recover code that would otherwise be lost if you are unable to commit.

As noted earlier, the GemStone Browser provides File Out menu items so you can file out methods and class definitions.

## Reading and Compiling a Saved File

- Open a GemStone workspace, or click on the Launcher.
- Select **File > File In**.
- Select the file in the resulting file dialog.
- Click **Open**.

Saved files are written by GemStone as sequences of commands such as those understood by Topaz, the command line-oriented version of the GemStone programming environment.

### Sample File-out Contents

This example shows a class definition in Topaz format.

#### Example 5.3

---

```
! Class 'Address'
!
run
Object subclass: 'Address'
    instVarNames: #( 'street' 'zip')
    classVars: #()
    poolDictionaries: #()
    inDictionary: UserGlobals
    constraints: #[[#street, String],
                 #[#zip, Integer]]
isInvariant: false
%
!
! Instance Category 'Updating'
!
category: 'Updating'
method: Address
street: newValue

    "Modify the value of the instance variable5 'street'."
    street:= newValue
%
method: Address
zip: newValue

    "Modify the value of the instance variable 'zip'."
    zip:= newValue
%
```

---

```

!
! Instance Category 'Accessing'
!
category: 'Accessing'
method: Address
street

    "Return the value of the instance variable 'street'."
    ^street
%
method: Address
zip
%

    "Return the value of the instance variable 'zip'."
    ^zip
%

```

---

GemStone's filing out and filing in facilities are intended mainly for saving and restoring classes and methods without manual intervention. If this is all you want to do, then you don't need to understand the Topaz commands involved. However, it is also possible to create custom files that include commands to commit transactions and to create and manipulate objects other than classes and methods. If you want to perform such tasks, you'll need a full description of the Topaz commands. See the *Topaz Programming Environment* manual for your system.

The file-in mechanism, however, has certain limitations. It cannot execute the full set of Topaz commands. Instead, it is restricted to the following subset:

category:	method:
classmethod	printit
classmethod:	removeAllMethods
commit	removeAllClassMethods
doit	run
method	

GemStone file-in mechanism acknowledges the presence of the following commands by adding notes to the transcript, but it does not execute them:

display	omit
expectvalue	output
level	remark
limit	status
list	time

If GemStone encounters any other Topaz commands, it stops reading the file and displays an error notifier.

The file in mechanism does not display execution results, either. Instead, it appends information to the transcript about the files it reads and the classes and categories for which it compiles methods.



---

## *Glossary*

---

### **Applet**

A Java program that is meant to be run in the context of a Java-compatible browser, rather than as a stand-alone program. Another kind of Java program, called an *application*, can run independently of a browser.

### **Debugger**

In this release, the debugger enables you to inspect a stack trace resulting from code execution on the server.

### **Filing Out**

The process of storing source code for GemStone Smalltalk classes and methods in an ordinary file. The files are written as a sequence of commands understood by Topaz, the GemStone command-line programming environment.

### **Firewall**

A gatekeeper computer that protects a local network by filtering traffic to and from an external network, such as the Internet.

**Gem**

A GemStone server process that provides access to clients. Currently, each session connects to the server through a dedicated Gem process.

**Inspector**

A window that lets you examine the values of a variety of GemStone objects and modify them when appropriate.

**Local Tools Buffer**

In this release, the Edit menu items Cut, Copy, and Paste use a buffer that is local to that instantiation of the Tools.

**NetLDI**

A GemStone network server process, which provides informational services and spawns other processes for GemStone clients.

**Session Broker**

A GemStone server object that supervises the connection of a Java client to the server by connecting the client to a Gem process. A broker is an instance of the GemStone class GbjBroker.

**Stone**

A GemStone process that oversees other server processes; it is also known as the repository monitor.

---

## *Index*

---

### **A**

Abort 2-4, 3-4, 4-3  
Add 3-5  
applet GlosGlossary-1

### **B**

Begin 3-4, 4-3  
Breakpoint Browser 3-12  
    opening 2-5  
breakpoints 3-12, 4-6  
    buttons 3-13  
Browser button 2-2, 3-1  
browsers, specialized 3-8  
browsing  
    classes 2-5  
    hierarchy 2-5  
    references 2-5

### **buttons**

    Browser 2-2, 3-1  
    Implementors 2-2  
    in Debugger 4-8  
    Login 2-2  
    Logout 1-6, 2-2  
    Senders 2-2  
    Workspace 2-2

### **C**

Cancel 3-4  
class  
    creating in GemStone 5-4  
    saving source code for 5-10  
Class Browser 2-5, 3-5, 3-8  
    description 3-8  
class instance variable 5-5  
Class menu 3-5  
class variable 5-5

Class Version Browser 3-5, 3-10  
description 3-8  
CLASSPATH 1-3  
Close 3-3  
Commit 2-4, 3-4, 4-3  
Compile 3-4, 5-8, 5-9  
configuration file 1-7  
shutdown and 1-11  
constraints 5-1  
Continue 4-8  
Copy 2-4, 3-3, 4-2  
Cut 2-4, 3-3, 4-2

## D

Debugger GlosGlossary-1  
debugger 4-1, 4-6  
Dictionary menu 3-5, 3-8  
dictionary, inspecting 4-5  
Do It 3-3

## E

Edit menu 2-4, 3-3  
environment variables 1-3  
error during class compilation 5-8  
executing code 2-4

## F

File menu 2-3, 3-3  
File Out As 3-5  
filing in code 2-4, 4-3  
filing out GlosGlossary-1  
filing out source code 5-10  
Find Class 3-5  
firewall GlosGlossary-1

## G

Gem GlosGlossary-2  
Gem Service 1-5  
GemBroker 1-4  
GemStone Browser 2-2, 3-8  
description 3-1

## H

Hierarchy Browser 2-5, 3-5, 3-8  
description 3-9

## I

Implementors 2-5, 3-7  
button 2-2  
Inspect 3-5  
Inspect It 3-4  
inspecting objects 2-4  
Inspector 2-4, 4-3  
inspector 3-2, 4-1, GlosGlossary-2  
description 4-3  
instance variables 5-1  
defined 5-5  
inspecting 4-4  
private 5-7  
instancesInvariant: 5-2

## K

keyboard shortcuts 4-13

## L

Launcher 1-6, 3-1, 3-8  
description 2-1  
using 1-3  
local tools buffer GlosGlossary-2

log files 1-14  
  locating 1-15  
Logging, verbose 1-15  
Login 2-4  
Login button 2-2  
login process 1-2  
Logout 1-6, 2-4  
  button 2-2

## M

Method Browser 3-7  
  description 3-11  
Method List Browser 2-2, 2-5, 3-7,  
  3-8  
  description 3-12  
Method menu 3-7  
method, defining a 5-9

## N

NetLDI 1-1, 1-5, 1-6, 1-15,  
  GlosGlossary-2

## O

optimized selectors 5-3

## P

Paste 2-4, 3-3, 4-3  
pool dictionaries 5-5  
popup menu 4-12  
Print It 2-4, 3-4  
Process menu 4-8  
Protocol menu 3-7

## R

Remove 3-5  
Rename As 3-5  
reserved selectors 5-3

## S

Select All 2-4, 3-3, 4-3  
selectors  
  optimized 5-3  
  reserved 5-3  
Senders 2-5, 3-7  
Senders button 2-2  
Session Broker 1-1, 1-4,  
  GlosGlossary-2  
  connecting to Gem 1-12  
  starting 1-10  
  startup script 1-10  
  stopping 1-11  
Session menu 2-4, 3-4  
Set Break 3-4  
set, inspecting 4-6  
shortcuts, keyboard 4-13  
Step Into 4-8  
Step Over 4-8  
Stone GlosGlossary-2  
string, searching for 2-5  
subclass, creating 5-7  
symbol dictionaries 3-2, 5-8

## T

Timeout 1-5  
Tools menu 2-5  
Transaction Mode 1-5, 2-4  
Transaction mode  
  automatic 2-4  
  manual 2-4

transaction, beginning 2-4  
Trim Stack 4-8  
Troubleshooting 1-14

## **U**

Update 3-3

## **W**

wildcards 2-3, 2-5, 3-8, 3-12  
workspace 2-1, 2-5, 4-1  
Workspace button 2-2