Technical Manual
for Skid Version 2.0.8

Klaus Hammermüller
klaus@ifs.tuwien.ac.at

October 6, 1999

Abstract

This paper is the technical documentation of the Java object-broker called *"Skid"* used in the *Asgaard-framework* built on pure Java components. The focus is having transparent persistent Java beans without the need of a proprietary enterprise server. *"Skid"* collaborates potentially with every SQL-server. We implemented a JDBC flatfile - driver as well as mySQL, Oracle has been also successfully tested.

Contents

1 Overview

1.1 What is Skid?

Skid is a collection of *pure Java*-classes implementing an object broker for bean-based data-objects implementing the `RemoteBean` interface for distributed applications.

Features of data objects implementing the `RemoteBean` interface:

- Transparent persistence for data objects;
- Implementation of a special kind of relationship called *"link"* between data objects to load objects at runtime which are explicit called dynamically;
- Searching for data objects about a automatically generated keyset intuitive filter classes;
- Implementation of an explicit event-log to enable push-capabilities.

Features of the object broker implementing the `RemoteBroker` interface:

- Transparent local or remote service for distributed application using the *RMI* mechanism;
- The broker can be accessed remotely using the `RemoteBroker`interface;
- Encapsulation from different database server using an flatfile-driver if no database-server is available;
- Zero administration of the *Skid*-server using the *Log*-server for Backup and service-messaging.

1.2 Limitations

Things which will be done soon:

- Implementing multi-threading to enable an multi-user environment;
- Adding more primitive Beans to the `asgaard.lang` package;
- Redesign integrating bean- and link-filter to a common mechanism.

Things which will be done by the implementation of the *Banshee*-project:

- Increacing the efficientsy of the log/backupstatements implementing an automated recover- and replication mechanism;
- Using an `zip` mechanism to tune performance;
- Implementing more complex filter-conditions explorating the needs by *Banshee* and adding the `BEAN` table as common root in all filter queries;
- Implementing an caching algorithm ensuring automated persistence.

Some other things:

- multi-language support for beans (implemented in the `banshee` package)

2 Install Skid

2.1 Distribution

The Distribution is under GNU public license Version 2 `http://www.gnu.org/copyleft/gpl.html`, there is absolutely NO WARRANTY on this software.

A complete Distribution contains on the following parts:

- packages `asgaard.lang`, `asgaard.skid`, `asgaard.skid.resource`;
- SimpleText flatfile JDBC driver;
- a mySQL JDBC driver;
- the packages of the *Log*-server `asgaard.utils.log`, `asgaard.utils.encode` and `asgaard.utils.mail`;
- the `asgaard.util.test` package;
- Suns JavaMail 1.1 classes (optional by the use of the *Log*-server).

2.1.1 Files

1. this document `Asgaard/doc/asgaard/skid`;
2. the binary distribution `bin/skid`;
3. guided by the installation `install/skid`;
4. the sources `Asgaard/src/asgaard/skid`;
5. and API documentation in `Asgaard/javadoc/asgaard/skid`.

2.1.2 Log Server

Optionally you may install the Log-server package. All binaries necessary for the logging mechanism are coming with this distribution from *Skid* and are running automatically. If you want to read more about the logging mechanism you have to have a look at the complete *Log-server* distribution.

2.1.3 Test-Scripts

Skid has a built in testing mechanism to ensure that the complete functionality is working as specified. All code for local as well as remote test-runs are coming with this *Skid* distribution. If you want to know more about the test mechanism you have to have a look at the complete *Test-script* distribution. How to run the test is described in section 3.2.

2.1.4 JDBC-driver

Skid support all level 4 JDBC driver to enable the physical storage in any SQL-Server. If no database server is available the ASCII-flatfile driver *SimpleText* (see next subsection) is used to create a local database automatically. Due performance impacts for bigger data-volumes a remote server is recommended.

This implementation has been tested with the free database-server mySQL (see `www.mysql.com`) which is available for different platforms. Also Oracle SQL server had been tested successfully in prior versions of this distribution. To support different SQL-dialects all needed SQL syntax is defined in a Ressource-class in the `asgaard.skid.ressource` package which can be maintained easily.

2.1.5 SimpleText

This is an ASCII flatfile JDBC driver from `http://www.thoughtinc.com` which is used to generate the log-file and the database if no database-server is available.

```
Copyright:   (C) 1997 THOUGHT Inc.  All rights reserved.
Copyright:   (C) 1996 Karl Moss.  All rights reserved.
             You may study, use, modify and distribute this example
             for any purpose, provided that this copyright notice
             appears in all copies.  This example is provided WITHOUT
             WARRANTY either expressed or implied.
```

2.1.6 Mail

The mail-capabilities to inform automatically about failures insist the packages `mail.jar` and `activaion.jar` from SUN's free 1.1 Mail-Distribution which is available at `http://java.sun.com/products/javamail`.

2.2 Installation

Please follow the instructions on `http://install/skid/index.html` or the following descriptions:

You must have at least JDK1.1.6 from JavaSoft or compatible Java environment at the Server you want to run Skid. In the rest of this page we assume that the Java interpreter (java) is installed correctly and available in your path. Native-code compilations of *Skid* are planned, but not implemented yet.

2.2.1 Install a database server (*optionally*)

1. Install a SQL-server, follow the instructions for that software. Additionally get a level 4 JDBC driver for that server. We choose the mySQL server.

2. Create a user. In mySQL, make sure that a separate user has been created for remote access over the net. The FAQ gives an example of this (you can set varying degrees of abilities - such as querying, inserting, deleting, creating, etc.) e.g.

   ```
   $ mysql mysql
   mysql> insert into user values ('%', 'name', password('[a password]'),
          'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y');
   mysql> quit
   $ mysqladmin reload
   ```

   See also: mySQL-Manual 6.4 Adding new user privileges to MySQL

   Note: This is a very priveleged user - you should specify 'N' to many of the above!

   This user has to be named when you start *Skid* (see command line parameters in section 3.1).

3. Create the database (in our case named "skid") in mysql (if it doesen't exist already). e.g.

   ```
   $ mysqladmin create skid # (skid is our database - name)
   ```

   The Database is created as a subdirectory from the installation path e.g. mysql/data/skid.

4. Add JDBC-Driver If you aren't using the release Java 1.1 jdk, then make sure you have downloaded and installed the latest JDBC implementation (version 1.22). See also `splash.javasoft.com/jdbc`.

5. The driver is loaded dynamically. If the driver is following the default syntax, the name of the driver and the location of the server should be enough information (see command line parameters in section 3.1). If not you may have to modify the source-code in the `asgaard.skid.Skid` class.

2.2.2 *Skid* installation

The following packages (Java-Bytecode) have to be copied to the server-machine and added to the CLASSPATH. If you habe chosen an other database than we, ensure that you are using the right JDBC-driver.

- The Skid classes which are coming in `skid.jar` and `utils.jar`;
- The `SimpleText.zip` ASCII flatfile JDBC driver;
- The `test.jar` package (*optional*: only necessary for test-runs);
- The JDBC-driver for the remote server (*optional*: if any);
- javamail 1.1 (*optional*: only nessecary with local LOG and used MAIL-OPTION): the packages `activation.jar` and `mail.jar`;

3 Running Skid

3.1 Command-Line Parameter

The remote *Skid*-server `Log` is started with a Java virtual machine like

```
java asgaard.skid.Skid
```

Command line parameter are `-hpPFLHMTDCUWmlt`:

```
-h  print this help message
-pP <propertyfile>  alternative path for the skid file
-pF <DBpath>        local directory if using a flatfile database
-pL <LOGpath>       local directory to write the LOGFILE
-pH <mail HOST>     used mailer (local log only)to
-pM <mail CC>       cc mailed statements (local log only)
-pT <mail TO>       mailed statements to (local log only)
-pS <LOGserver>     remote LOGSERVER
-pK <LOGsize>       mail if the Logfile extends <LOGsize>kb
-l<statement>       log created statements up to this level
-m<statement>       mail created statements (local log only)
    0   process no statements
    1   process security statements
    2   process error statements
    3   process warning statements
    4   process remark statements
    5   process backup statements
    6   process debug statements
-pD <JDBC-drivername> e.g. oracle.jdbc.driver.OracleDriver
-pC <connectstring> e.g. jdbc:oracle:thin:@<host>:<port>:<database>
```

```
-pU <connectuser>
-pW <connectpassword>
-t  runs the testscript
```

The parameter are not stored, so need to be specified on every start. To store the settings edit the `.property` file in the root directory of *Skid*. Additional reading of the *Log*-server are recommended.

Before starting the *Skid*-server the `rmiregistry` must be started to establish the *RMI* name-service.

3.2 Testing the Skid-Installation

To start *Skid* type

```
rmiregistry
java -cp skid.jar;utils.jar;test.jar;SimpleText.zip asgaard.skid.Skid -t
java -cp skid.jar;utils.jar;test.jar;SimpleText.zip asgaard.skid.TestSkid <host>
```

First statement starts the registry, the second one a local instance of the *Skid* server and the last does a remote test of the remote *Skid* interface. For `<host>` insert the IP-adress of the host where *Skid* is running.

3.3 Starting *Skid*

A complete command-line call could look like:

```
rmiregistry
java -cp skid.jar;utils.jar;mysql.jar;activation.jar;mail.jar
     asgaard.skid.Skid
     -pD org.gjt.mm.mysql.Driver
     -pC jdbc:mysql://<host-IP>:<port>/skid
     -pU <username>
     -pW <password>
```

4 Integrate the `BeanBroker` into sources

The use of *Skid* is implemented in one class named `BeanBroker`. For more detailed information please have a look at the `javadoc`-documentation.

4.1 Usage of `Skid` - components

```
BeanBroker broker = new BeanBroker(); // a new broker
                                      // (one per VM)

SkidBean b = new SkidBean();          // instanciates a new Bean (1)

b.setXXX();                           // sets properties
b.getXXX();                           // get properties

b.undo();                             // undo changes
                                      // if in setXXX
      SkidBean.modify();              // was implemented

b.flush();                            // stores changes explizit
```

```
BeanLink[] l = b.getLink();           // all links of this bean

SkidBean b2 = l[i].getTo();           // get the connected Bean
                                      // with the rekursion Bean/Link
                                      // all data in the DB is available (2)

SkidBeanFilter f = b2.getFilter();    // 2nd possibility to (3)
                                      // find beans

f.setFilterEntry("Vorname", "Klaus", FILTER_EQ);      // def.

SkidBean[] b3 = f.getBeans();         // get the search result

BeanLinkFilter f2 = l.getFilter();    // same thing for links

b.addEvent( b2 );                     // create an event (-> PUSH)
b.suscribeEvent();                    // suscribe a type of Event
SkidEvent[] e = b.getEvent();         // get the events (4)
```

1. If the generation of a subclass from `SkidBean` is not possible, there is the interface `RemoteBean` and the container class `SkidBeanContainer`. A special class is the `Definition`, a "lightweight-class" which carries only a name;
2. `Links` are implemented because normal handles to other objects are stored automatically (if not declared `transient`). To operate within a huge amount of objects there is the need to load only objects which are actually needed for processing an operation, *links* are an explicite mechanism to septerate different objects holding their relationships;
3. Every bean is analyzed automatically. The `getXXX` methods are used to generate (numerical) keys which are stored within a keytable. Every table carries only the new keys of the class. If only fore some general properties are asked (e.g. "name") instances from different classes may be with the resultset;
4. This "manual" event-mechanism shall avoid from an event-avalance. They are explicitely generated by objects and can be suscribed from other objects to implement an *push*-mechanism carrying relevant information to an interested user automatically.

4.2 UML - Model of the Skid-Components

A overview about all classes from package `asgaard.skid` is given by Figure 1. A short intro in the outlined classes:

- An implementation of `RemoteBean` is able to access most of the functionality and is the precondition to make an object persistent;
- The `BeanBroker` is the central manager of all beans but works transparent for the user (for use it only has to be instanciated once);
- `Skid` wrappes the `SkidBroker` as remote server;
- `SkidBeanFilter` is the carrier of a more complex search for a bean;
- `SkidLink` models the relationship between `RemoteBeans` with explicite access (no automated loading into the VM).

To create the stub classes the remote `asgaard.skid.Skid` and `asgaard.skid.SkidBean` class has to be compiled with the `rmic` compiler too.

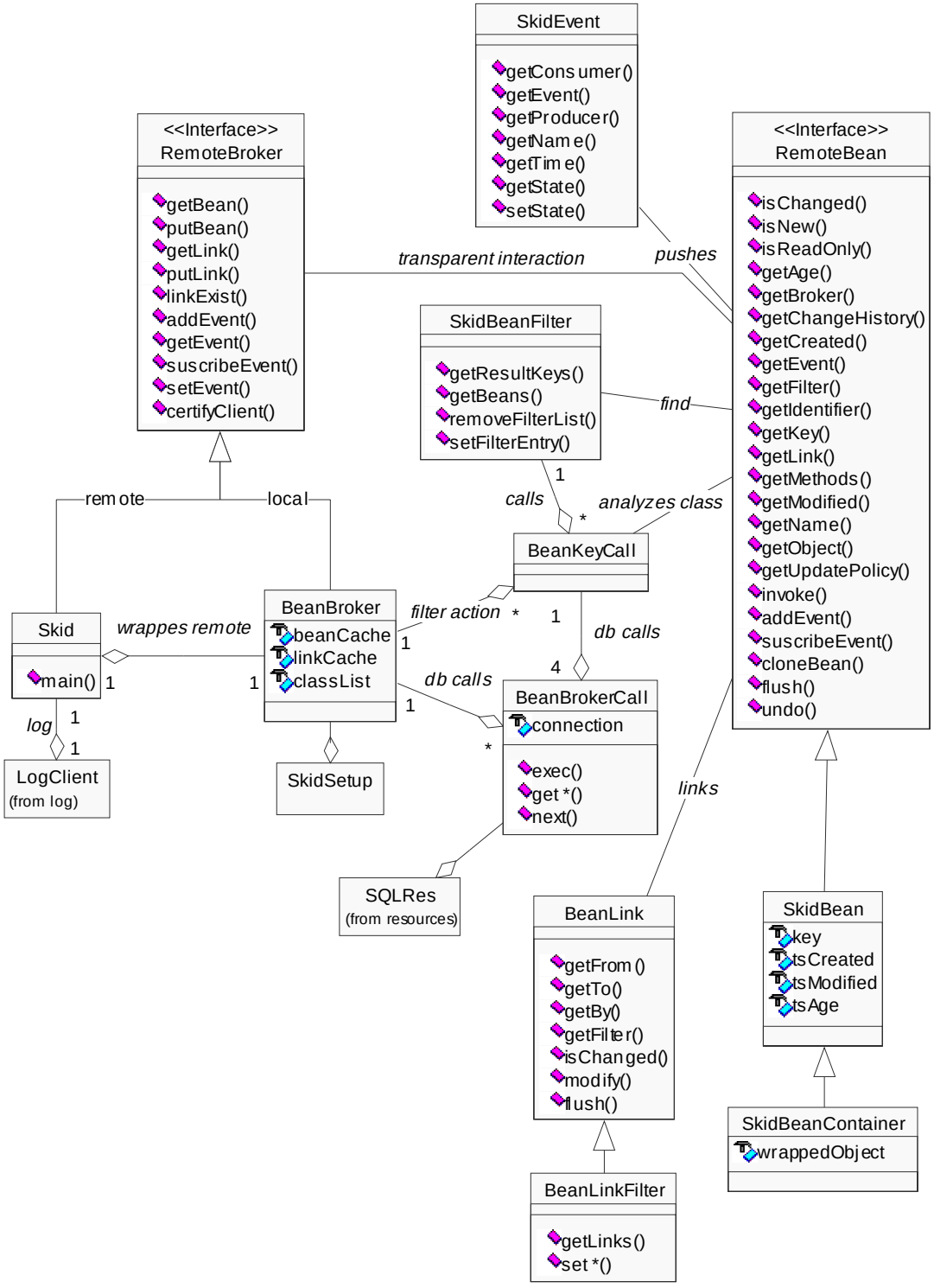

Figure 1: Class diagram from package `asgaard.skid`