# Table of Contents

Developers can integrate their own objects at the client as well as at the server. Here we will illustrate the use of objects that are integrated at the server and accessed by the client.

Any Java object/library/API can be used at the server. Compatibility to server´s JVM is a basic requirement. In addition, stateless calls should be supported. Although it would generally not be a problem to use stateful objects, the scalability of the application would be affected.

All server objects that are used by JVx in life cycle objects support stateless calls.

**Example**

We want to develop an object that enables database logging and will use this object in our application.

The log object could be implemented as follows:

Logger.java

```
/**
 * The <code>Logger</code> class is a simple log handler for databases.
 *
 * @author René Jahn
 */
public class Logger
{
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Class members
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /** the log statement. */
   private PreparedStatement psLog;

   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   // Initialization
   //~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   /**
    * Creates a new instance of <code>Logger</code> for a specific
database.
    *
    * @param pAccess the database access
    * @throws SQLException if the log statement could not be
initialized
    */
   public Logger(DBAccess pAccess) throws SQLException
   {
      psLog = pAccess.getConnection().prepareStatement("");
   }

   /**
    * Logs an error message to the database.
```

```
     *
     * @param pMessage the error message
     * @throws SQLException if it's not possible to log an error message
     */
    public void error(String pMessage) throws SQLException
    {
       psLog.setString(1, pMessage);
    }

}   // Logger
```

The object is used in the life cycle object of our MasterSession:

Session.java

```
public class Session extends Application
{
   ...
   ...

   /**
    * Gets the database logger.
    *
    * @return the logger instance
    * @throws Exception if an exception occurs during log creation
    */
   public Logger getLogger() throws Exception
   {
      Logger log = (Logger)get("logger");

      if (log == null)
      {
         log = new Logger(getDBAccess());

         put("logger", log);
      }

      return log;
   }
}    // Session
```

Now we can make a log call in our application:

```
getConnection().call("logger", "error", "This is an error message!");
```

This calls the error method of the logger object.