Table of Contents

Sowohl am Client als auch am Server können Entwickler Ihre eigenen Objekte integrieren. Wir beschreiben hier jedoch nur die Verwendung von Objekten die am Server integriert und vom Client angesprochen werden.

Am Server kann nahezu jedes beliebige Java Objekt/Bibliothek/API verwendet werden. Eine Grundvoraussetzung ist die Kompatibilität zur verwendeten JVM des Servers. Weiters sollte darauf geachtet werden, daß Stateless Aufrufe unterstützt/angeboten werden. Es ist zwar kein Problem Stateful Objekte zu verwenden, hätte aber Auswirkungen auf die Skalierbarkeit einer Applikation.

Alle Server Objekte die von JVx in Lifecycle Objekten verwendet werden, unterstützen Stateless Aufrufe.

Anwendungsbeispiel

Wir erstellen ein Objekt mit dem Datenbank Loggings durchgeführt werden können und verwenden dieses Objekt in unserer Applikation.

Das Log Objekt könnte wie folgt implementiert werden:

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
```

Das Objekt verwenden wir im Lifecycle Objekt unserer 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
```

Nun können wir in unserer Applikation einen Log Aufruf durchführen:

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

Durch den Aufruf wird die error Methode des logger Objektes aufgerufen.