Software Architektur
Butterfly Alarmanlage

Stefan Frings

03.10.2011

Inhaltsverzeichnis

1 Allgemeines

1.1 Entwicklungsumgebung

Die Butterfly Alarmanlage wird mit Netbeans in Java entwickelt, auf Basis von JSF 2.0.

Als Datenbank wird MySQL eingesetzt.

Die Applikation wird auf einen Tomcat Server installiert. Theoretisch kann auch jeder andere Webserver verwendet werden, der Java EE 6 unterstützt.

Da das Web-Interface von Anfang an zur Grundfunktion gehören sollte, wurde Software ausgewählt, die sich seit vielen Jahren im Web-Umfeld bewährt hat.

Dass letztendlich überwiegend Produkte von Oracle eingesetzt werden, hat sich eher zufällig ergeben.

Besonders hervorzuheben ist die modulare Struktur der Hardware-Schnittstelle. Sie ermöglicht das Hinzufügen weiterer Hardware (Anschlussboxen) mit minimalem Entwicklungsaufwand.

1.2 Quelltext-Dokumentation

Die Quelltexte der Butterfly Alarmanlage sind durch Javadoc Kommentare umfangreich dokumentiert, wie auch die öffentlich zugängliche bfUtilities Library dokumentiert ist. Diese Kommentare lassen sich mit dem Javadoc Compiler zu einer übersichtlichen HTML Dokumentation zusammenfassen.

1.3 Verzeichnis-Struktur

Die Quelltexte sind zusammen mit der Dokumentation und den benötigten Bibliotheken in einem Projektverzeichnis zusammengefasst, das folgende Struktur hat:

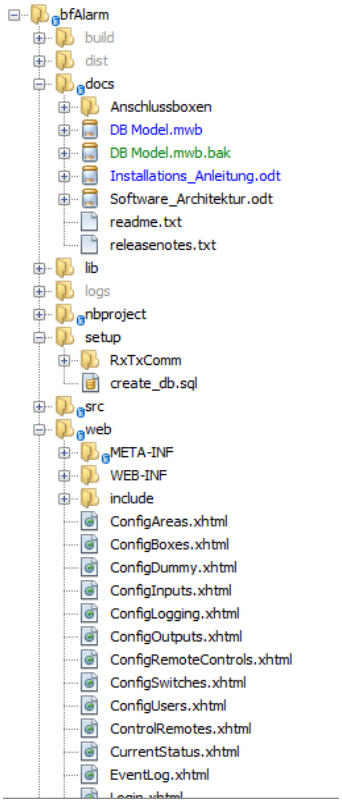
Im docs Verzeichnis befindet sich die Dokumentation, also dieses Dokument, sowie die Betriebsanleitung und kleinere Textdateien mit Notizen des Entwicklers.

Im lib Verzeichnis befinden sich die benötigten Bibliotheken. Wobei die bfUtilities Bilbliothek vom gleichen Autor stammt. Deren Quelltexte sind als Open-Source veröffentlicht.

Im setup Verzeichnis befinden Dateien, die zur Installation der Applikation benötigt werden. Wobei Java, MySQL und Tomcat separat downgeloaded werden müssen.

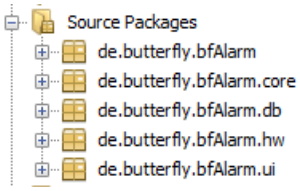
Im src Verzeichnis befinden sich die Java Quelltexte. Alle Klassen dieser Applikation befinden sich im Haupt-Paket de.butterfly.bfALarm, was sich entsprechend der standarisierten Java Namenskonvention in der Verzeichnis-Struktur widerspiegelt.

Im web Verzeichnis befinden sich statische Dateien für den Webserver, also Bilder, und Stylesheets, aber auch die Vorlagen zur Erzeugung dynamischer Seiten.



1.4 Java Packages

Die Java Quelltexte sind in fünf Pakete sortiert.



Das Paket de.butterfly.bfAlarm enthält die Startup Klasse, die Abläufe beim Starten und Beenden des Programms regelt, sowie allgemein verwendete Klassen, z.B. für Ereignis-Protokollierung.

Die Kern-Funktionen der Applikation befinden sich in dem Paket de.butterfly.bfAlarm.core. Der Kern der Alarmanlage arbeitet mit Threads, die im Hintergrund den Status der Eingänge abfragen und ggf. Alarm auslösen.

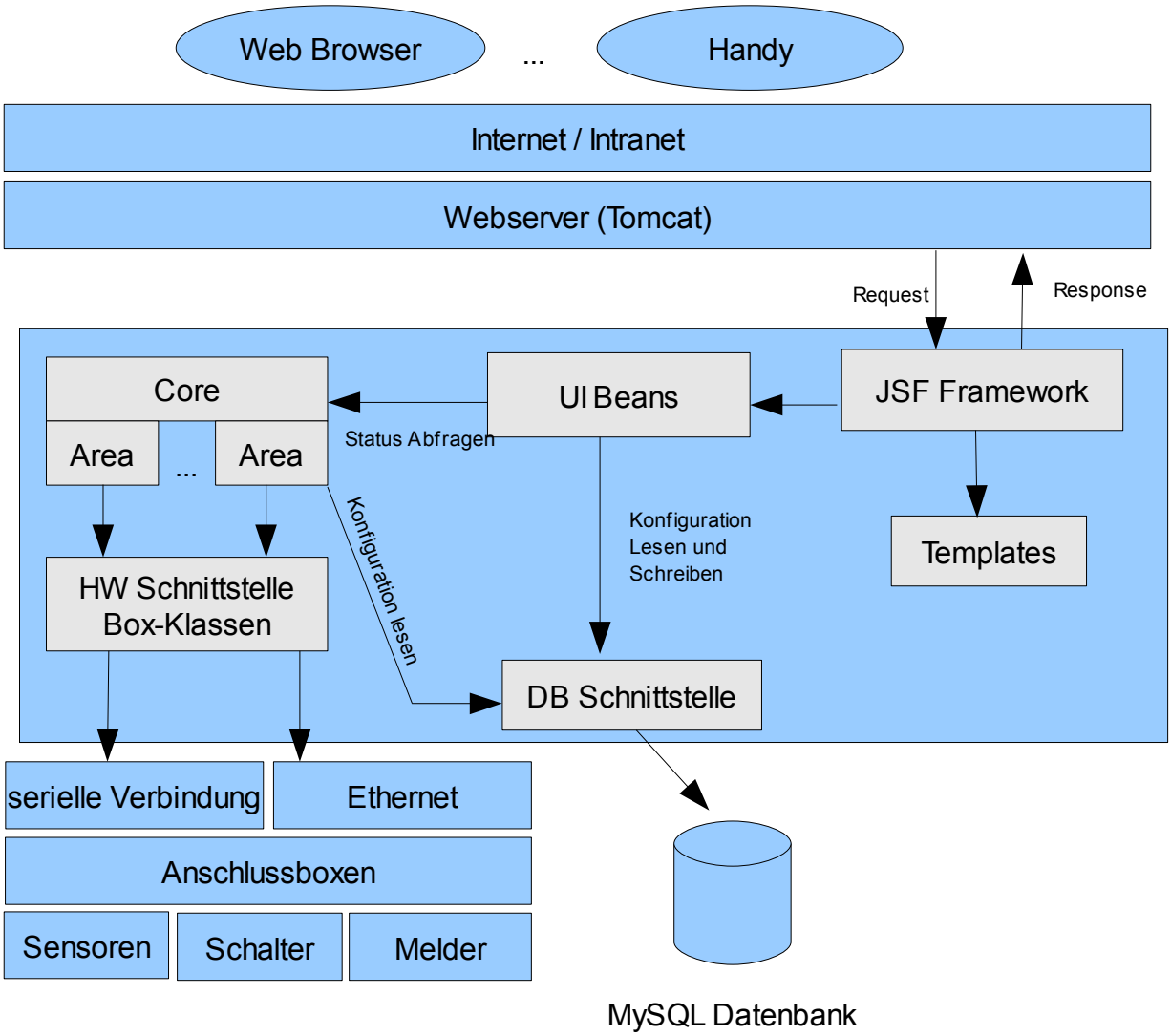
Im Paket de.butterfly.bfALarm.db befinden sich die Klassen für Zugriff auf die Datenbank.

Das Paket de.butterfly.bfAlarm.hw enthält Klassen zur Kommunikation mit der Hardware – den Anschlussboxen.

Das Paket de.butterfly.bfAlarm.ui enthält JSF Beans für die Darstellung der dynamischen Webseiten.

2 Architektur

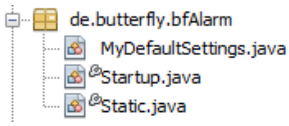
2.1 Übersicht



Die Pfeile zeigen an, wer sich mit wem verbindet:

- Der Web Server ruft das JSF Framework auf, welches mit Hilfe von Templates Webseiten erzeugt.
- Das JSF Framework fragt den aktuellen Zustand der Anlage über die UI Beans ab.
- Die UI Beans enthalten auch Aktions-Methoden für Benutzereingaben.
- Die UI Beans kommunizieren mit der Datenbank, um Konfigurationen zu lesen oder zu speichern. Sie kommunizieren auch mit dem Kern der Alarmanlage, um dessen Status abzufragen.
- Der Kern holt sich seine Konfiguration aus der Datenbank. Entsprechend der Konfiguration baut der Kern seine Areas auf, die in einem folgen Kapitel noch detaillierter dargestellt sind.
- Diese Areas kommunizieren über die HW Schnittstellenklassen mit der Hardware. Auch diese Klassen sind in einem folgenden Kapitel detaillierter dargestellt.

2.2 Ablaufsteuerung



Die Ablaufsteuerung beim Starten und Stoppen übernimmt die Startup Klasse.

Beim Starten initialisiert sie den Verbindungs Pool zur Datenbank. Dann initialisiert Sie den Logger, der Meldungen in Textdateien schreibt. Diese Dateien diesen primär der Untersuchung von Störungen.

Zum Schluss wird der Kern gestartet.

Die Web Oberfläche arbeitet Ereignisorientiert (reverse of Control). Deren Servlets werden bei Bedarf vom Webserver aufgerufen; darum enthält die Startup Klasse keine entsprechenden Aufrufe.

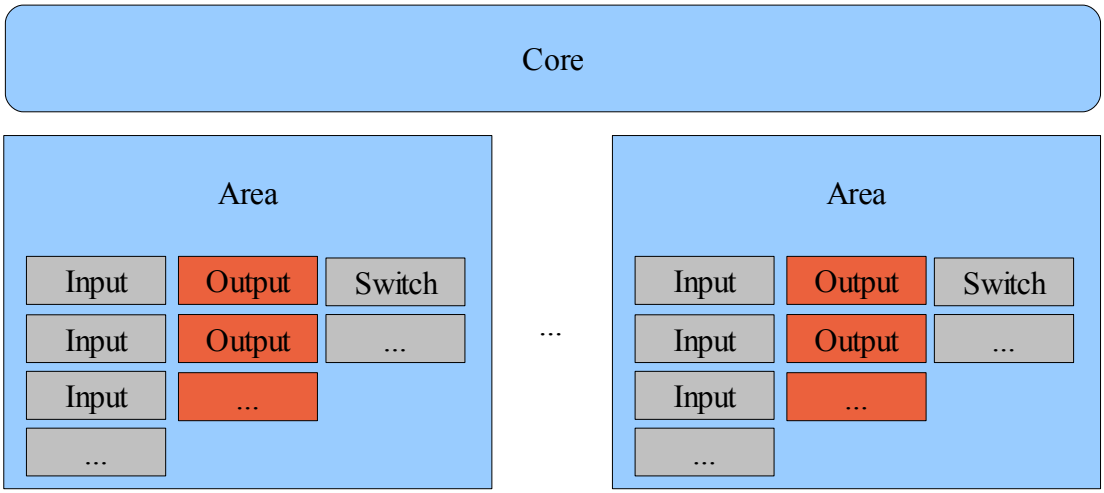
Beim Stoppen werden diese Komponeten in umgekehrter Reihenfolge beendet.

Die Static Klasse enthält die wenigen Statischen Ressourcen, die vom ganzen Programm verwendet werden. Zum Beispiel eine Referenz auf die Konfigurations-Klasse und eine Referenz auf den Datenbank-Verbindungspool.

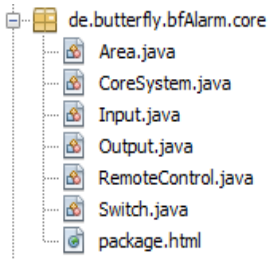
Die Klasse MyDefaultSettings enthält Vorgabewerte für die allgemeine Konfigurationstabelle der Applikation. Im Wesentlichen sind dies die Einstellungen zum Logging, sowie Textfragmente von Fehlermeldungen (zwecks Internationalisierung).

2.3 Funktionaler Kern

Der funktionale Kern ist die eigentliche Alarmanlage, also der Teil, der für die ständige Überwachung der Eingänge verantwortlich ist, sowie für das Auslösen von Alarm-Meldungen.



Dieser Teil befindet sich im Paket de.butterfly.bfAlarm.core, in den folgenden Klassen:



Die Klasse CoreSystem beschreibt den funktionalen Kern der Alarmanlage. Der Kern erzeugt für jeden konfigurierten Bereich ein Area Objekt und schließt diese auch wieder, wenn der Kern beendet wird. Der Kern enthält einen Thread, der stündlich veraltete Ereignisprotokolle löscht.

Die Klasse Area beschreibt einen zu überwachenden Bereich mit seinen Sensor-Eingängen, Melder-Ausgängen und Scharf-Schaltern. Jeder Bereich hat einen eigenen Thread, der zyklisch alle Eingänge Abfragt und ggf. einen Alarmauslöst.

Die Klasse Input beschreibt einen Sensor-Eingang. Die poll() Methoden der Input Klassen werden zyklisch von der Area Klasse aufgerufen, um den jeweiligen Status des angeschlossenen Sensors abzufragen.

Die Klasse Output beschreibt einen Melder-Ausgang. Deren Methoden startAlarm() und stopAlarm() werden von der Area Klasse aufgerufen, wenn ein Alarm beginnt oder endet. Ein Timer Thread sorgt dafür, dass der Alarm ggf. verzögert wird und dass er nach der eingestellten Dauer wieder endet.

Die Klasse Switch beschreibt einen Scharf-Schalter. Die Area Klasse ruft deren Methode poll() auf, um den Status des Schalters abzufragen.

Jeder Bereich kann beliebig viele Inputs, Outputs und Switche haben. Normalerweise hat ein Bereich viele Eingänge, einen Ausgang und eventuell einen Scharf-Schalter.

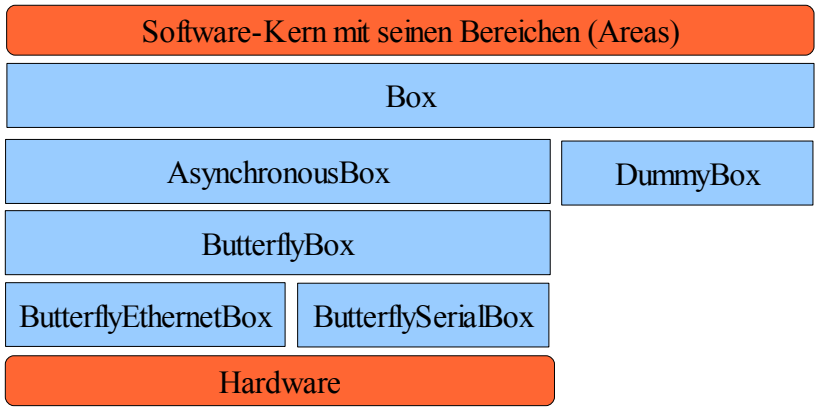
Weitere Klassen in diesemPaket:

Die Klasse RemoteControl repräsentiert einen Fernsteuer-Ausgang, der direkt über die Web-Oberfläche ein/aus geschaltet wird. Sie steht in keiner Beziehung zu den obigen fünf Klassen.

2.4 Hardware-Schnittstellen

Zur Kommunikation mit der Hardware verwendet die Alarmanlage je nach Anschlussbox entweder eine ButterflyEthernetBox oder eine ButterflySerialBox. Die Geräte mit USB und Bluetooth Schnittstelle werden über virtuelle serielle Ports angesprochen.

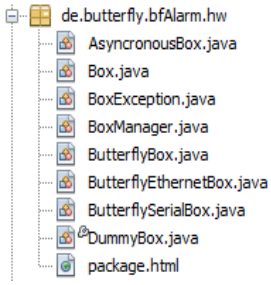
Um möglichst kostengünstig und schnell weitere Hardware unterstützen zu können, wurden diese Treiber modular aufgebaut. Die Architektur und Hierarchie dieser Klassen entspricht dem folgenden Schichten-Modell:



Die abstrakte Box Klasse definiert die Schnittstelle zur Alarmanlagen-Software (Input,Output und SwitchKlassen). Die Klassen ButterflyEthernetBox und ButterflySerialBox definieren die konkrete Umsetzung der Schnittstelle zur Hardware.

Auf Basis dieses Schichten-Modells können beliebige andersartige Anschlussboxen hinzugefügt werden. Sie müssen letztendlich lediglich die wenigen simplen Funktionen der Box Klasse implementieren, von denen die meisten auch noch optional sind.

Die relevanten Klassen befinden sich im Paket de.butterfly.bfAlarm.hw:



Der BoxManager erzeugt Box-Klassen zum Zugriff auf Anschlussboxen. Der Manager stellt sicher, dass für jede Anschlussbox genau ein Objekt erzeugt wird, auch wenn es von mehreren Bereichen verwendet wird.

Die Klasse Box ist ein abstrakte Grundlage für konkrete Implementierungen. Sie legt fest, dass jede Box die folgenden Funktionen haben muss:

- Float getAnalog(Portname) zum Abfragen analoger Eingänge.
- Boolean getDigital(Portname) zum Abfragen digitaler Eingänge.
- setDigital(Portname, Wert) zum Setzen digitaler Ausgänge.
- GetStatus() und hasError() zur Abfrage des Zustandes (z.B. ok oder defekt).

Die drei erstgenannten Funktionen sind alle Optional. Es kann also rein digitale oder rein analoge Anschlussboxen geben.Oder auch welche, die nur Ausgänge haben aber keine Eingänge.

Alle Box-Klassen melden funktionale Störungen durch BoxExceptions.

Die DummyBox dient zu Simulationszwecken. Sie kommunziert nicht mit externer Hardware,sondern speichert den Zustand der Eingänge und Ausgängelediglich im RAM.

Die Klasse AsynchronousBox ist eine abstrakte Grundlage für Anschlussboxen mit Asynchroner Kommunikation. Die Asynchrone Box sorgt für einen automatischen Verbindungsaufbau bei Bedarf und sie empfängt die Antworten auf Befehle in einem eigenen Thread, damit der Programmablauf durch eventuelle Kommunikationsstörungen nicht blockiert wird.

Da bisher keine synchrone Anschlussboxen entwickelt wurden, gibt es noch keine entsprechende SynchronousBox Klasse.

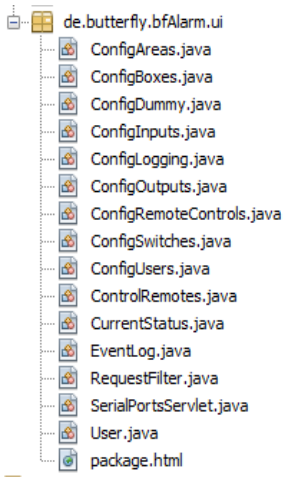
Die ButterflyBox ist eine abstrakte Grundlage für Anschlussboxen mit der Butterfly I/O Firmware, welche asynchron mit dem Computer kommunizieren. Um die Performance zu verbessern fragt diese Klasse alle Eingänge der Hardware mit einem einzigen Befehl ab und hält das Ergebnis einige Millisekunden im Speicher, so dass mehrere Aufeinanderfolgende getDigital() Abfragen nur eine Nachricht auf der physikalischen Verbindung zur externen Hardware erfordern. In dieser Klasse sind die Befehle definiert, die an Butterfly Anschlussboxen gesendet werden, sowie die Auswertung der Antworten.

Die ButterflySerialBox wird für Anschlussboxen mit serieller Schnittstelle verwendet. Dazu gehören auch die Geräte mit USB und Bluetooth Schnittstelle, welche letztendlich über virtuelle serielle Ports angesprochen werden. Die Klasse öffnet und schließt den seriellen Port.

Die ButterflyEthernetBox wird für Anschlussboxen mit Ethernet Schnittstelle verwendet. Sie baut den TCP Socket auf und ab.

2.5 JSF View Beans

In dem Paket de.butterfly.bfAlarm.ui befinden sich die JSF Beans zum Erzeugen der dynamischen Webseiten. Jeder Webseite-Case hat ihre eigene Klasse.



Die meisten Webseiten dienen der Konfiguration der Alarmanlage, entsprechend viele Config... Klassen gibt es auch.

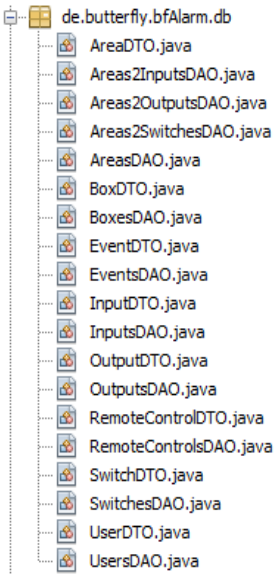
Das SerialPortsServlet wird von einer Hilfeseite verwendet, um die verfügbaren seriellen Ports aufzulisten. Weil dies unter Windows mehrere Sekunden dauern kann, wird diese Anzeige per AJAX nachgeladen.

Der RequestFilter fängt vor der Verarbeitung durch die Servlets alle eingehenden Requests ab. Er überprüft, ob der Benutzer eingeloggt ist und ob er ausreichende Rechte hat, die gewünschte Funktion aufzurufen. Falls dem nicht so ist, wird der Benutzer aufgefordert, sich einzuloggen bzw. mit einem anderen Namen einzuloggen.

Das User Bean liefert Name und Zugriffsrechte des aktuellen Benutzers, und es enthält die Aktions-Methoden zum Ein- und Aus-loggen. Damit der Benutzername auch außerhalb der JSF Beans verfügbar ist, wird der Name in die HTTP Session geschrieben.

2.6 Datenzugriff

Zugriffe auf die Datenbank sind in Daten-Transfer-Objekte und Daten-Zugriffs-Objekte gekapselt. Auf diese Weise reduzieren sich Aufwände für Weiterentwicklung und Anpassung an andere Datenbanken.



Die DTA Klassen repräsentieren jeweils einen Datensatz aus einer Tabelle.

Die DAO Klassen enthalten die SQL Befehle, mit denen Daten geladen und gespeichert werden.

3 Bibliotheken

Die Applikation verwendet folgende Bibliotheken:

3.1 RxTx Comm

Diese Bibliothek erweitert die Java Laufzeitumgebung um die Unterstützung von seriellen und parallelen Schnittstellen. Dazu gehört für jedes Betriebssystem eine native C Library, welche die Brücke zwischen Java und dem Betriebssystem darstellt.

Für die Alarmanlage ist nur der Teil für serielle Schnittstellen relevant.

Die Applikation nutzt die Java Library in Version 2.1. Die C Libraries dürfen ruhig auch in Version 2.2 vorliegen.

3.2 MySQL Connector

Diese Library gehört zur MySQL Datenbank. Die Java Applikation kommuniziert über diese Library mit dem Datenbank Server. Die Library stellt eine Schnittstelle gemäß JDBC Standard zur Verfügung.

3.3 Butterfly Utilities

Die Butterfly Utilities sind eine Sammlung von nützlichen projekt-unabhängigen Klassen. Für die Alarmanlage sind im Wesentlichen folgende Bestandteile relevant:

- Das Paket de.butterfly.config ermöglicht einfachen Zugriff auf Konfigurationsparameter, die wahlweise in einer Datei oder in einerDatenbank-Tabelle gespeichert werden. Die Alarmanlage verwendet die Datenbank-Variante. Jeder Parameter hat einen Namen, einen Wert, einen Standardwert und einen Hilfetext.
- Das Paket de.butterfly.database stellt einen Verbindungs-Pool zur Datenbank zur Verfügung. Der Pool verbessert die Performance der Applikation erheblich,weil der Aufbau von Verbindungen sehr Zeitaufwendig ist. Der Pool hält einmal geöffnete Verbindungen für spätere Wieder-Verwendung offen. Ein Timer schließt alle Verbindungen automatisch, die für längere Zeit nicht verwendet wurden.
  Der Tomcat Webserver bietet die gleiche Funktionalität an, jedoch hat sich in anderen wesentlich größeren Web-Anwendungen mehrmals gezeigt, dass der Datenbank Pool von Tomcat nach langer Laufzeit sporadisch ausfällt und dann keine hilfreichen Fehlermeldungen ausgibt.
  Ein zweiter Grund für die Verwendung eines eigenen Datenbank Pools ist, dass die Applikation auch auf anderen Webservern laufen kann, die keinen Datenbank Verbindungs-Pool haben,oder einen,der anders funktioniert.
- Das Paket de.butterfly.logging stellt Funktionen zum Schreiben von Logfiles zur Verfügung. Sie basieren auf den java.util.logging Klassen, ermöglichen jedoch eine flexiblere Konfiguration der Meldungstexte (Zeilenformat), und sie bieten die Möglichkeit, individuelle Logfiles pro Web-Applikation zu schreiben,wenn mehrere Applikationen in einem Webserver installiert sind.
- Das Paket de.butterfly.template stellt eine kleine Template Engine zur Verfügung, die Platzhalter für Variablen und Schleifen-Konstrukte verarbeitet und mit Werten füllt. Sie wird vom SerialPortsServlet verwendet.

Quelltexte und Dokumentation der Butterfly Utilities kann man im Internet frei downloaden.

4 Datenbank Modell