Java® Anbindung an EIB mit Eiblet® Technologie

Robert Ott *Professional JNet Solutions AG*

Erfolge durch Standardisierung

Kaum ein Begriff ging so schnell um die Welt wie es der Begriff Java in der Informatik-Welt tat. Die immer noch sehr junge Java Plattform konnte sich schon heute von properitären Plattformen wie Microsoft Windows abheben und beweisen, dass offene Standards und Interoperabilität zu den wichtigsten Erfolgsfaktoren für moderne IT-Lösungen gehören.
Das EIB System ist von den Grundgedanken her der Java Plattform sehr ähnlich. EIB wird offen durch die EIBA standardisiert und EIB Komponenten werden durch einen stehenden Zertifizierungsprozess auf Einhaltung der Standards überprüft. Obwohl EIB und Java zwei völlig verschiedene Technologien sind, lässt sich zumindest eine Ähnlichkeit behaupten. Die EIBA ist für das EIB System **der** Erfolgsfaktor für eine offene Standardisierung. Bei Java ist dies der stehende '100% Pure Java' Zertifizierungsprozess der durch Zertifizierungsstellen sichergestellt wird, welche von Sun Microsystems Inc. (dem Erfinder von Java) unabhängig sind. Dies ermöglicht heute, dass die Interoperabilität zwischen Java Systemen von verschiedenen Herstellern gewährleistet werden kann.

Entscheidung Gerät oder Plattform?

Bis vor kurzem mussten sich Anwender sowie Softwareentwickler immer die Frage stellen, welches Gerät man anwenden will oder für welche Geräteart man Software entwickeln möchte. Anwender und Entwickler verfechten noch heute Betriebssysteme wie MacOs, Sun Solaris, HP-Unix, NewtonOS, VMS, IBM AIX, Linux, MS-Windows 95/98/NT und Silicon Graphics Irix.
Erste Anstösse für eine betriebsystemunabhängige Programmierung glaubte man mit der Programmiersprache C und C++ erreicht zu haben. Die Realität zeigte jedoch, dass Programmcode, der für ein bestimmtes Betriebssystem geschrieben wurde, nicht einfach auf andere Systeme übertragen werden konnte. Der Grund für diese Nicht-Portierbarkeit liegt weniger in der Sprache selbst, mehr doch in den verschiedensten Arten von Programmbibliotheken, die verwendet werden. Zudem muss C und C++ immer für eine bestimmte Microprozessorart kompiliert werden und ist nach einer solchen Kompilation nur auf der entsprechenden Microprozessorart ausführbar.

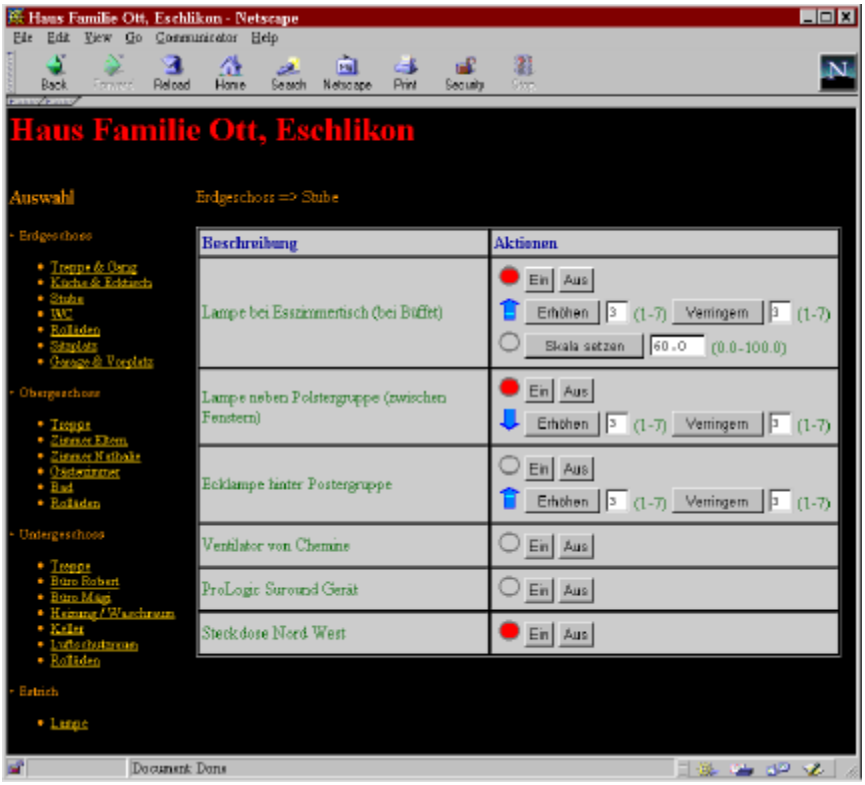

Bild 1. *EIB Ansteuerung via Web Browser*

Auch die Programmiersprache Smalltalk scheiterte aufgrund von ähnlichen Gründen. Zwar kann kompilierter Smalltalk Code auf verschiedenen Microprozessorarten ausgeführt werden, Programme unterscheiden sich jedoch durch verwendete Programmbibliotheken die nicht auf allen Systemen erhältlich sind.
Mit der Java Plattform wurde nun eine Abstraktionsschicht zur unterliegenden Microprozessorart und zum jeweiligen Betriebsystem geschaffen. Java Programme können auf verschiedensten Geräten ausgeführt werden, ohne das diese Programme neu kompiliert werden müssen.
Die Entscheidung Gerät oder Plattform fällt dadurch sehr leicht. Man muss sich heute nicht mehr primär für eine Geräteart wie MS-Windows PC, 3Com's Palm Pilot Serie oder Psion festlegen, man kann sich heute für die

Robert Ott (33), Master of Science (Honours), ist Geschäftsführer der Firma Professional JNet Solutions AG, 9542 Münchwilen, Schweiz

Java 2 Plattform entscheiden und damit fast sämtliche Betriebssysteme und Microprozessorarten abdecken.
Bild 1 zeigt eine vollständig in Java realisierte Anbindung an ein EIB System, welches durch einen Internet-Browser bedient werden kann. Das Benutzerinterface in Bild 1 enthält wenig Grafiken, was in einer enormen Anwendungsgeschwindigkeit via Internet resultiert. Natürlich kann das Benutzerinterface beliebig mit Grafiken angereichert werden, was vor allem im IntraNet Bereich für Firmengebäude und grössere Eigenheime sinnvoll sein kann. IntraNet Netzwerke bieten meist einen sehr hohen Datendurchsatz und erlauben dadurch die Übertragung von grösseren Grafiken.
Bild 2 zeigt die gleiche Java Anbindung an EIB, die diesmal jedoch auf einem PalmV Gerät von 3Com ausgeführt wird.

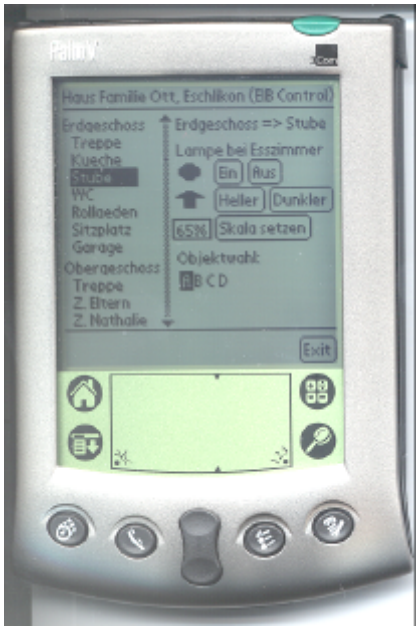

Bild 2. *EIB Ansteuerung via PalmV*

Java EIB Anbindung

Bild 3 zeigt die primäre Architektur der Java Anbindung an EIB. Ein sogenanntes EIBAgent Objekt ist verantwortlich, Instanzen von EIB-GroupObject's zu verwalten und diese an Java Applikationen weiterzugeben.
Das EIBAgent Objekt ist auch verantwortlich, dass die EIBGroup-Object's immer den Zustand der entsprechenden EIB Gruppenadressen repräsentieren. Dies wird erreicht indem das EIBAgent Objekt mit einem EIBAgentCommunicator Objekt

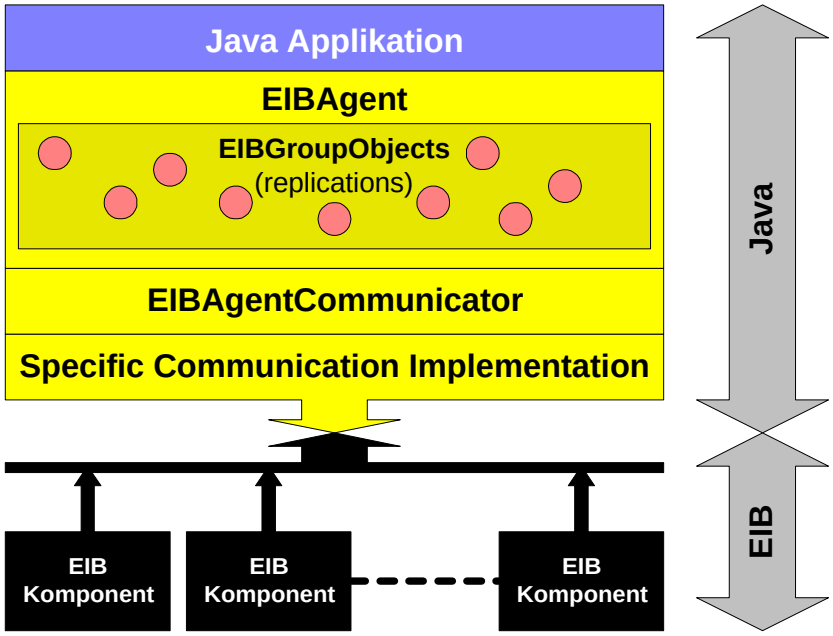

Bild 3. *EIB Anbindung von Java Applikationen*

kommuniziert. Das EIBAgentCommunicator Objekt ist verantwortlich, Meldungen (Messages) zwischen dem Java Programm und einem EIB System auszutauschen. Das heisst, wenn sich der Zustand einer EIB Gruppenadresse ändert, wird das Telegramm vom EIBAgentCommunicator Objekt aufgefangen und an das EIBAgent Objekt weitergeleitet, welches dann die Änderung am entsprechenden EIBGroupObject vornimmt. Dies wird auch in entgegengesetzter Richtung gewährleistet, indem Änderungen an EIBGroupObject's, die durch eine Java Applikation erfolgen können, sofort vom EIBAgent Objekt wahrgenommen und an das EIBAgentCommunicator Objekt weitergeleitet werden. Diese Änderungen werden dann vom EIBAgentCommunicator Objekt als Telegramme an das entsprechende EIB System weitergegeben.
Die EIBAgentCommunicator Klasse ist eine abstrakte Java Basisklasse, die durch Spezialisierung (Subclassing) implementiert werden kann. Diese Architektur erlaubt zum Beispiel, dass eine Implementation eines EIBAgent-Communicator's den EIB ComClient Gateway von Schlaps & Partner verwenden und eine andere Implementation eines EIBAgentCommunicator's beispielsweise das FM-Modul von Levy Fils benutzen kann. Neue Schnittstellen zum EIB System können also mit sehr wenig Aufwand in Java implementiert werden und die Kommunikation zum EIB System gewährleisten.
Durch die klare Schnittstelle des EIB-Agent Objektes können leicht mehrere EIB Systeme durch eine Applikation angesprochen werden. EIB Applikationen zur Generierung von Web-Seiten können zum Beispiel als Java Servlets, die Zugriff auf einen oder mehrere EIBAgents haben, realisiert werden.

Eiblet™ Software Komponente

In der Java Welt werden übertragbare Programmteile meist mit Kurzformen wie Applet, Servlet oder Spotlet benannt. Für übertragbare Java Software-Komponenten wurde im EIB Bereich der Begriff Eiblet eingeführt. Ein Eiblet ist ein Stück Java Code der in ein laufendes Java System geladen und danach ausgeführt werden kann. Ein Eiblet hat Zugriff auf einen EIB-Agent und kann damit auf EIB Gruppenaddressen (EIBGroupObject's) zugreifen.
Durch die Portabilität von Java Code können Eiblets auf Systemen wie MS-Windows, Linux, Sun Solaris, Macintosh oder gar PalmV entwickelt, getestet oder konfiguriert werden und danach auf andere Systeme, sogenannte Eiblet Engines, übertragen und ausgeführt werden.
Beispielsweise könnte ein Eigenheimbesitzer auf einer gemütlichen Zugsreise ein Logik-Eiblet für die Weihnachtsbeleuchtung seines Gar-

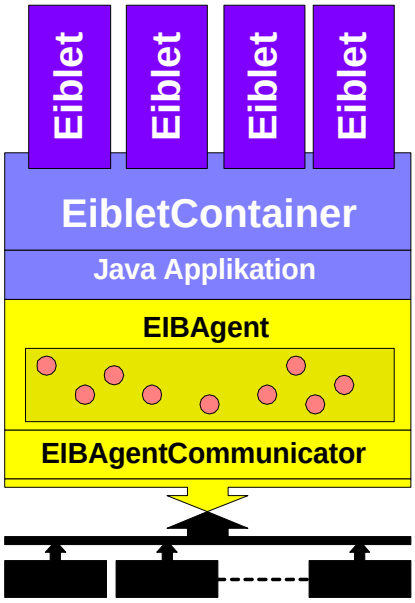

***Bild 4**. Eiblet Engine mit EIBAgent*

tens auf einem PalmV Gerät konfigurieren. Zu Hause angekommen könnte das konfigurierte Eiblet auf einen Home-Server, z.B. einem mit dem EIB verbundenen Linux System, übertragen und ausgeführt werden.
Bild 4 zeigt den Aufbau einer Eiblet Engine die zur Ausführung von Eiblet Software Komponenten benutzt werden kann. Klar erkennbar ist die Wiederverwendung der Grundarchitektur der EIB Anbindung aus Bild 3. Die Eiblet Engine enthält ein EibletContainer Objekt welches für die Verwaltung von Eiblets verantwortlich ist.
Wie auf Bild 4 ersichtlich, ist der EibletContainer eine reine Java Applikation die auf ein EIBAgent Objekt zugreifen kann.

Eiblet™ Code Beispiel

Eiblets sind wie erwähnt als Logik-Komponenten zur Steuerung eines EIB Systems definiert. Einerseits können Eiblets mit Hilfe von Java Beans erstellt werden. Java Beans sind vorprogrammierte Java Bausteine die visuell zusammengefügt und konfiguriert werden können.
Andrerseits können komplexere EIB Logik-Komponenten auch mit Hilfe der Java Programmiersprache erstellt werden.
Bild 5 zeigt ein kleines Code Beispiel, wie ein Java Eiblet aussehen könnte. Das hier gezeigte Eiblet wartet auf Änderungen an der EIB Gruppenadresse 4/3/0 und gibt diese mit einer Verzögerung von 2 Sekunden

```
 1 import com.eiblet.eib.*;
 2
 3 public class MyEiblet extends Eiblet
 4                       implements EIBGroupObjectDataListener {
 5
 6   private EIBGroupObject myInputObject = null;
 7   private EIBGroupObject myOutputObject = null;
 8
 9   public void init() {
10     try {
11       EIBAgent agent = getEIBAgent();
12
13       myInputObject = agent.resolveEIBGroupObject(4, 3, 0);
14       myOutputObject = agent.resolveEIBGroupObject(4, 4, 0);
15
16       myInputObject.addEIBGroupObjectDataListener(this);
17
18       myInputObject.setEISTypeRequest(EIBGroupObject.EIS_TYPE_SWITCH);
19       myOutputObject.setEISTypeRequest(EIBGroupObject.EIS_TYPE_SWITCH);
20
21     } catch (Exception exception) {
22       System.err.println("Exception: " + exception.getMessage());
23     }
24   }
25
26   public void handleEIBGroupObjectDataChanged(EIBGroupObject object) {
27     try {
28       if (object == myInputObject) { // is that my input object?
29
30         // read the input
31         boolean value = myInputObject.getEIBDataSwitch().getValue();
32
33         // wait 2 seconds
34         Thread.currentThread().sleep(2000);
35
36         // set the output (inverted)
37         value = !value;
38         myOutputObject.getEIBDataSwitch().setValueRequest(value);
39       }
40
41     } catch (Exception exception) {
42       System.err.println("Exception: " + exception.getMessage());
43     }
44   }
45
46 }
```

***Bild 5**. Eiblet Code Beispiel*

invertiert an die EIB Gruppenadresse 4/4/0 weiter.
In Zeile 1 wird das vorgegebene Java Package 'com.eiblet.eib' importiert. In Zeile 3 wird nun eine Eiblet Klasse, in diesem Falle mit dem Namen MyEiblet, definiert welche von der vordefinierten Klasse Eiblet abgeleitet ist. Auf Zeile 4 wird mittels eines Listeners das Interesse angemeldet, bei Zustandsänderungen von EIB Gruppen Objekten informiert zu werden.
In Linie 6 und 7 werden zwei Instanzvariablen deklariert, welche die zwei verwendeten EIB Gruppenadressen 4/3/0 und 4/4/0 repräsentieren.
Die Methode init() wird beim laden des Eiblets ausgeführt und beinhaltet Initialisierungen wie die Erstellung von Referenzen auf die EIB Gruppen Objekte. Des weiteren wird in Zeile 16 das Interesse an Änderungen am EIB Gruppen Objekt 4/3/0 angemeldet.
Die Handlermethode beginnend auf Zeile 26 übernimmt die eigentliche Logik. Zuerst wird geprüft, ob die angekommene Änderung wirklich das Input Objekt betrifft. Danach wird dessen Wert eingelesen, 2 Sekunden gewartet und danach das Output Objekt mit dem invertierten Wert gesetzt.

Demonstration der Eiblet™ Technologie

Die Eiblet Technologie wurde erstmals an der JISA'99 Konferenz (Java for Industrial Automation) im Silicon Valley (USA) durch Professional JNet Solutions AG und EIBA vorgestellt. Weitere Demonstrationen werden an der diesjährigen EIB Scientific Partnership Konferenz in München und an der ETFA'99 in Barcelona stattfinden.