Gut zugeschnitten

Alternativen für die Integration von SAP-Systemen mit Java EE

Carsten Erker, Torsten Fink

Individuelle Geschäftsanwendungen treiben die fachlichen Unternehmensprozesse. Die kaufmännischen Prozesse werden dagegen sehr häufig von SAP-Systemen unterstützt. Daher ergibt sich stetig wiederkehrend der Bedarf, Individualanwendungen mit SAP-Systemen zu koppeln. Wir stellen die wichtigsten Alternativen für Integrationstechnologien im Java-Bereich vor mit einem Schwerpunkt auf Open-Source-Lösungen.

Geschäftsanwendungen stehen selten allein in der Anwendungslandschaft. In vielen, vor allem größeren Unternehmen werden die kaufmännischen Prozesse von SAP-Systemen gestützt. Um einen hohen Grad an Automatisierung und damit Optimierung der Geschäftsprozesse zu erreichen, ist häufig eine Integration anderer Systeme mit SAP-Systemen notwendig. Über die Jahre hinweg sind verschiedene Technologien entstanden, um SAP mit der Außenwelt zu verbinden.

In diesem Artikel konzentrieren wir uns auf Individualanwendungen, die mit der Java Enterprise Edition (Java EE) erstellt wurden. Die Java EE bietet Anwendungsentwicklern eine leistungsfähige Infrastruktur, um zuverlässige skalierbare transaktionale Anwendungen zu realisieren. Als Implementierung setzen wir die JBoss Enterprise Middleware von Red Hat ein, die über ein Subscription-basiertes Modell verfügbar ist [RedHatMW]. Zusätzlich beschränken wir uns hier auf die Kopplung mit der SAP ERP Central Component (kurz ECC, dem ehemaligen SAP R/3), also dem ABAP-Stack von SAP [SAPERP]. Auf dieser läuft noch immer der Großteil der existierenden Geschäftslogik in der SAP-Welt.

Die JBoss Middleware und die Java EE bieten von Hause aus standardisierte Integrationstechnologien. Dem Entwickler stehen somit verschiedene Alternativen zur Verfügung; die richtige Wahl ist, wie so oft, nicht trivial und hängt vom konkreten Einsatzszenario ab. Dieser Artikel gibt einen Überblick über die, aus unserer Sicht, wichtigsten Technologien für die Integration Java-basierter Unternehmensanwendungen mit den Daten und Geschäftsprozessen in SAP ERP-Systemen. Abbildung 1 zeigt die jeweiligen Integrationstechnologien, deren Vor- und Nachteile in diesem Artikel aufgeführt werden, und es wird dabei erläutert, in welchem Kontext welche Technologie sinnvoll eingesetzt werden kann.

SAP NetWeaver Gateway

Die jüngste und modernste der hier vorgestellten Integrationstechnologien ist das SAP NetWeaver Gateway [SAPNetWeaver]. Hierbei handelt es sich um eine Reihe von Erweiterungen für einen existierenden ABAP-Stack. Das NetWeaver Gateway bietet eine REST-basierte Schnittstelle zum SAP-System. Klienten verwenden also HTTP als Kommunikationsprotokoll, als Datenformate werden JSON und XML unterstützt. Das Open Data Protocol (kurz OData, [OData]) wird hierbei als Standard für die Definition der Datenmodelle sowie für die Identifizierung der über die REST-Schnittstelle zur Verfügung gestellten Ressourcen verwendet.

Um Daten aus dem SAP-Backend zur Verfügung zu stellen, müssen im SAP NetWeaver Gateway sogenannte Gateway

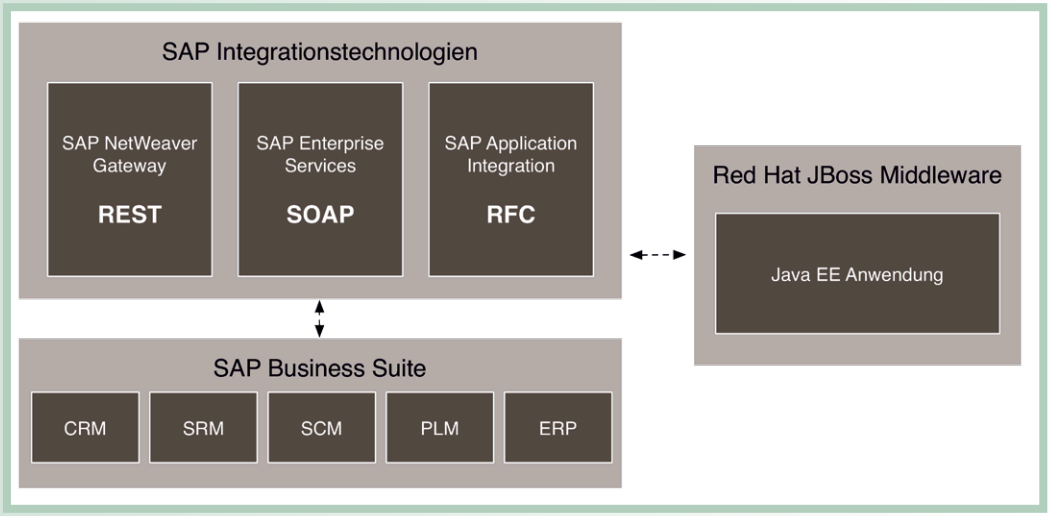

Abb. 1: Überblick über alternative Integrationsmöglichkeiten

Services erzeugt werden, welche die OData-Strukturen definieren und die benötigte Laufzeitlogik enthalten. Bereits vorhandene Dienste wie Remote Function Modules, beziehungsweise Objekte und Dienste, die vom Business Object Repository (BOR) zur Verfügung gestellt werden, können auf einfache Weise als Gateway Services definiert werden. Das NetWeaver Gateway bietet darüber hinaus die Möglichkeit, weitere Daten und Dienste aus dem Backend auf maßgeschneiderte Datenstrukturen abzubilden und nach außen hin verfügbar zu machen.

Auf der Klientenseite kann wegen des REST-basierten Ansatzes und der damit verbundenen offenen Protokolle und Formate eine Vielzahl vorhandener Technologien und Frameworks genutzt werden. So kann direkt aus Java-Code oder mittels Integrationsplattformen wie JBoss Fuse Service Works und JBoss Data Virtualization ohne die Verwendung proprietärer Technologien und ohne spezialisiertes ABAP-Wissen auf das SAP-Backend zugegriffen werden [RedHatDV].

Java-Entwickler können mit den gewohnten Entwicklungswerkzeugen arbeiten, aber auch für andere Programmiersprachen und Umgebungen stehen Entwicklungswerkzeuge zur Verfügung. So gibt es Werkzeuge zum Auffinden von durch das NetWeaver Gateway bereitgestellten Diensten und für die Generierung von Klientenstubs für die Eclipse IDE, Microsoft Visual Studio und Xcode für iOS-Anwendungen.

Der Preis für die einfache Integration auf der Klientenseite liegt jedoch in einem höheren Aufwand für das SAP-Backend. So müssen zusätzliche Betriebskomponenten installiert und gepflegt werden, wobei SAP für Produktionsumgebungen dedizierte Server für das SAP NetWeaver Gateway empfiehlt. Zudem ist die Erzeugung von Gateway Services für komplexere, auf den jeweiligen Anwendungsfall zugeschnittene Szenarien nicht trivial, sodass zunächst entsprechendes Know-how im ABAP-Team aufgebaut werden muss.

Ein weiterer Nachteil ist die fehlende Transaktionalität, die in der Natur des REST-Ansatzes liegt. Zwar ist sichergestellt, dass die Daten innerhalb des betreffenden SAP-Systems konsistent bleiben, es gibt jedoch keine Möglichkeit, den Transaktionskontext beim Aufruf von Diensten von außen zu steuern oder gar Aufrufe in verteilte Transaktionen einzubinden, um die systemübergreifende Datenkonsistenz zu gewährleisten. In sehr einfachen Einsatzszenarien, etwa ein mobiles Frontend zur Anzeige von Kundendaten, stellt dies häufig kein Problem dar, bei komplexeren Geschäftsanwendungen kann jedoch nicht sichergestellt werden, dass die Daten in den beteiligten Systemen jederzeit konsistent bleiben.

Zusammenfassend lässt sich sagen, dass das SAP NetWeaver Gateway einen datenzentrierten Zugang zum SAP ERP ermöglicht, der klientenseitig mit leichtgewichtigen Technologien und ohne SAP-Spezialwissen umgesetzt werden kann. Dem gegenüber stehen der Mehraufwand im Betrieb sowie der nötige Wissenserwerb für das ABAP-Team. Für Anwendungen, die Daten hauptsächlich informativ zur Verfügung stellen und kaum Schreibzugriffe beinhalten, stellt das NetWeaver Gateway einen geeigneten Ansatz dar.

SAP Enterprise Services

Bereits seit einigen Jahren gibt es die SAP Enterprise Services [SAPESWorkplace], die es ermöglichen, zwischen SAP- und anderen Systemen über Webservice-Schnittstellen zu kommunizieren. Zum Einsatz kommen HTTP als Transportprotokoll sowie die bekannten, auf XML basierenden Standards SOAP und WSDL. SAP Enterprise Services sind standardmäßig in jedem SAP ERP-System integriert, sodass keine weiteren Komponenten installiert und betrieben werden müssen.

Zu einer bestehenden ABAP-Funktion kann in SAP auf einfache und schnelle Weise ein Webservice erzeugt werden. Für diesen Webservice-Wrapper wird automatisch eine Schnittstellenbeschreibung in Form einer WSDL erzeugt, die als Grundlage für die Entwicklung von Klientencode auf der Java-Seite verwendet werden kann. Zusätzlich bietet SAP in seinem Enterprise Services Workplace bereits viele fertige Pakete für Kernfunktionalitäten und branchenspezifische Aufgaben an, die heruntergeladen und installiert werden können [SAPESWorkplace].

Da es sich bei SOAP-basierten Webservices um eine Reihe offener, wohldefinierter, weit verbreiteter und in der Java-Welt sehr gut unterstützter Standards handelt, erfordert die Entwicklung von Klienten keine Spezialkenntnisse. So können Java-Entwickler ihre gewohnten Entwicklungswerkzeuge verwenden, um den Schnittstellencode zu generieren und zu testen.

Der Hauptnachteil der SAP Enterprise Services liegt – wie beim SAP NetWeaver Gateway – in der fehlenden Transaktionalität. Das transaktionale Verhalten kann auch beim Aufruf eines Webservice nicht von außen gesteuert werden und die Aufrufe können nicht im Rahmen einer verteilten Transaktion ausgeführt werden, wodurch es aufwendig ist, eine systemübergreifende Datenkonsistenz sicher zu stellen.

SAP Enterprise Services verwenden sowohl auf der SAP- als auch auf der Java-Seite mit den Webservice-Standards wohlbekannte Technologien, die in den meisten Organisationen schnell und mit geringem Aufwand eingesetzt werden können. Insbesondere in Integrationsszenarien mit einer Integrationsplattform wie JBoss Fuse Service Works sind Enterprise Services interessant, da die Integration über Webservices bereits mit Standardkomponenten unterstützt wird. In komplexen Szenarien, in denen die Konsistenz der Daten über Systemgrenzen hinweg zugesichert werden muss, sind Enterprise Services nur eingeschränkt zu empfehlen.

Remote Function Call mit dem SAP Java Connector

Remote Function Call (RFC) ist eine Schnittstellentechnologie von SAP für den Aufruf von ABAP-Funktionen in entfernten Systemen. RFC ist die älteste der hier besprochenen Technologien und wird für die Kommunikation sowohl von SAP-Systemen untereinander als auch von Drittsystemen mit SAP eingesetzt. Da RFC vor allem auch für die Kommunikation zwischen SAP-Systemen bestimmt ist, war es von Anfang an darauf ausgelegt, Funktionsaufrufe im erweiterten Transaktionskontext zu ermöglichen. RFC basiert zudem auf einem leistungsfähigen Binärprotokoll, das auch sehr große Datenmengen sicher und schnell bearbeiten kann; eine Eigenschaft, die häufig HTTP-basierten Ansätzen fehlt.

Mit RFC aufrufbar sind alle ABAP-Funktionsmodule, die „remote-fähig“ sind, was alle Funktionen einschließt, die dem BAPI-Standard (Business Application Programming Interface) entsprechen. Funktionalitäten, für die es noch keinen entsprechenden ABAP-Funktionsbaustein gibt, müssen auf SAP-Seite implementiert werden. Dies ist jedoch Standard-ABAP-Entwicklung, sodass das benötigte Know-how in der Regel vorhanden ist.

Für die Integration in Java-Anwendungen stellt SAP seinen Java Connector (kurz JCo) zur Verfügung. Dieser besteht aus einer Java-Bibliothek mit dem benötigten API sowie einer nativen Bibliothek, die das RFC-Protokoll implementiert und von der Java-Bibliothek verwendet wird. Beide müssen in die Java-An-

wendung integriert werden, auf der SAP-Seite sind keine zusätzlichen Komponenten nötig.

Der SAP Java Connector unterstützt eine bidirektionale Kommunikation. So ist es nicht nur möglich, SAP-Funktionen aus Java heraus aufzurufen (JCo-Client), sondern aus ABAP-Code können auch Funktionen in Java-Anwendungen aufgerufen werden (JCo-Server). Neben der eigentlichen Kommunikation bietet der Java Connector Dienste für Connection Pooling, Transaktionen und Authentifizierung.

Obwohl der Java Connector über ein eigenes kommerzielles Lizenzmodell verfügt, verursacht er neben den Lizenzen für die ohnehin vorhandenen SAP-Systeme keine weiteren Lizenzkosten. Er kann jedoch nicht öffentlich heruntergeladen werden. Hierzu ist ein Zugang zum SAP Service Portal notwendig [SAPMP].

Der SAP Java Connector erfüllt zwar die Anforderungen, um komplexe Anwendungsszenarien umzusetzen, jedoch gibt es zwei Nachteile. Erstens handelt es sich beim JCo und dem darunterliegenden RFC-Protokoll um proprietäre Technologien, die sich nicht ohne Weiteres in die entsprechenden Java EE-Standards (insbesondere für Connection Pooling, Security und Transaktionen) integrieren.

Zweitens bietet der Java Connector aus Sicht des Anwendungsentwicklers eine eher niedrige Abstraktionsebene der darunterliegenden Konzepte, sodass relativ viel technischer Code geschrieben werden muss. Dies führt erfahrungsgemäß dazu, dass eine zusätzliche Schicht zwischen dem Java Connector und der Anwendungslogik benötigt wird, um den Anwendungscode nicht mit technischen Details zu vermischen und um wiederkehrende Aufgaben an einer zentralen Stelle zu kapseln. Für unterschiedliche Architekturen (Stand-alone Anwendungen, Java EE-Server-Anwendungen usw.) muss eine solche Abstraktionsschicht immer wieder neu entwickelt werden.

Zusätzlich fällt ein erheblicher Implementierungsaufwand bei der Abbildung der Schnittstellenparameter an. In beide Richtungen geschieht dies programmatisch, sodass jeder Parameter einer SAP-Funktion im Programmcode gesetzt beziehungsweise ausgelesen werden muss. Da diese Schnittstellenparameter benannt sind und im Programmcode als Strings referenziert werden, ist die Prozedur recht fehleranfällig.

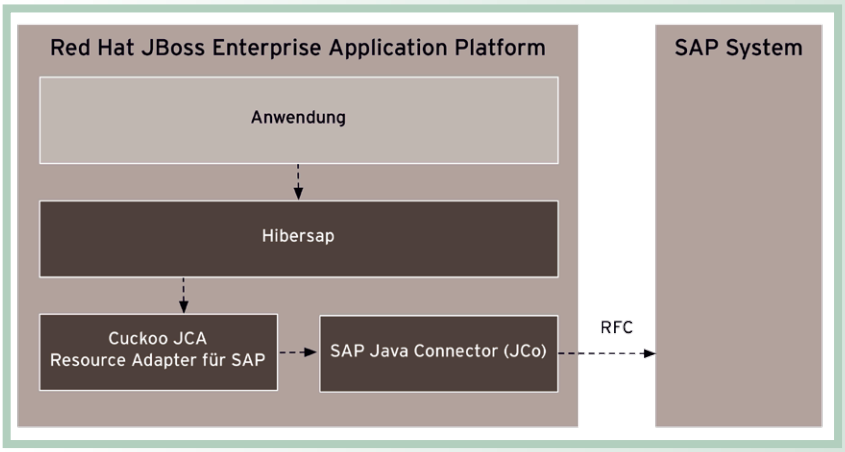

Abb. 2: Zugriff per Hibersap über einen JCA Resource Adapter

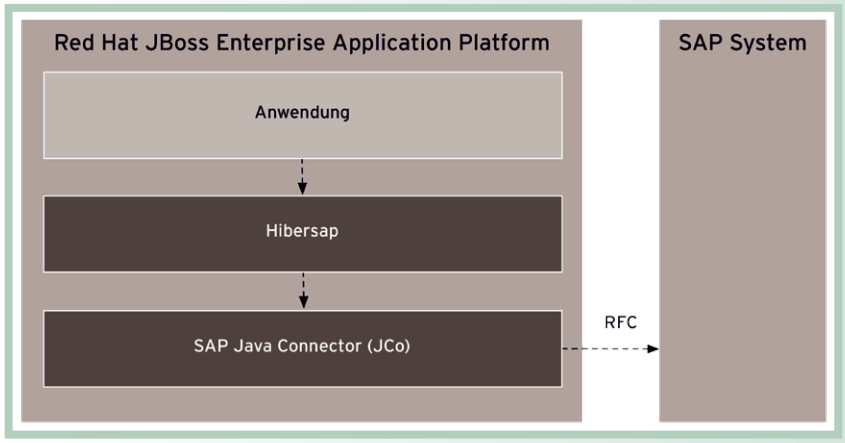

Abb. 3: Zugriff per Hibersap direkt über JCo

Der Einsatz in Java EE

Die Java EE bietet einen eigenen Standard, der für die Integration von Java Enterprise-Anwendungen mit Fremdsystemen zuständig ist: die Java EE Connector Architecture (kurz JCA, [JSR322]). Ein sogenannter Resource Adapter ist dafür zuständig, den Zugriff auf ein Fremdsystem über Java-Schnittstellen abzubilden. JCA definiert Standardschnittstellen für den Zugriff auf das Drittsystem seitens der Anwendung und für den Java EE-Applikationsserver zur Verwaltung von Systemressourcen.

Neben dem Zugriff auf Systeme wie SAP dient die JCA zum Beispiel auch der Ansteuerung von Datenbanken und Nachrichtendiensten. Sie definiert Dienste wie Verbindungsmanagement, Security, Transaktionsverwaltung und Konfiguration und unterstützt Aufrufe aus Java in das Fremdsystem (Outbound-Kommunikation) und umgekehrt Aufrufe aus Fremdsystemen in Java EE-Anwendungen (Inbound-Kommunikation).

SAP bietet zwar einen JCA Resource Adapter an, der mit dem SAP-eigenen Applikationsserver ausgeliefert wird. Leider ist dieser nicht ohne Weiteres auf anderen Applikationsservern lauffähig, da er intern den SAP Java Connector in einer speziellen Ausprägung verwendet, die auf andere Plattformen nicht einfach übertragbar ist.

Für Anwendungen, die auf anderen Applikationsservern wie der JBoss Enterprise Applicatiom Platform (JBoss EAP) betrieben werden, steht als Open-Source-Variante der von uns mit entwickelte Cuckoo Resource Adapter zur Verfügung [CuckooRA]. Dieser verwendet intern ebenfalls den SAP Java Connector, jedoch standardkonform, sodass dieser auf sämtlichen mit Java EE kompatiblen Applikationsservern lauffähig ist.

Hibersap

Der Einsatz eines JCA-kompatiblen Resource Adapters adressiert jedoch nur den ersten der beiden genannten Nachteile des SAP Java Connectors: Die vom JCo bereitgestellten technischen Dienste werden auf die Standardschnittstelle der entsprechenden Java EE-Dienste abgebildet. Wie beim SAP Java Connector fällt auch bei der Verwendung eines Resource Adapters erheblicher Aufwand beim Abbilden der Schnittstellenparameter auf Java-Klassen an. Hier kommt die, ebenfalls von uns mit entwickelte, Open-Source-Bibliothek Hibersap zum Einsatz [Hibersap].

```
@Bapi("BAPI_SFLIGHT_GETLIST")
public class FlightListBapi {
  @Table
  @Parameter("FLIGHTLIST")
  private List<Flight> flightList;
}
```

Listing 1: Beispiel für die Konfiguration eines BAPI-Zugriffs in Hibersap

Hibersap stellt eine Abstraktionsschicht oberhalb des SAP Java Connectors oder eines JCA-kompatiblen Resource Adapters dar. Die Wahl zwischen diesen beiden Technologien ist konfigurationsabhängig und kann ohne Anpassung des Programmcodes geändert werden. So kann derselbe Anwendungscode in der Produktivumgebung beispielsweise einen transaktionalen Resource Adapter verwenden, während lokale Integrationstests mit dem Java Connector ohne umhüllenden Applikationsserver arbeiten. Abbildung 2 zeigt den Zugriff über den Resource Adapter, während Abbildung 3 den direkten Zugriff über JCo darstellt.

Das Mapping der Schnittstellenparameter bei der SAP-Anbindung mit Hibersap geschieht außerhalb des eigentlichen Programmcodes durch Java-Annotationen. Listing 1 stellt exemplarisch die Konfiguration einer Klasse für einen SAP-Zugriff dar. Für jede implementierte Schnittstelle, das heißt jede SAP-Funktion, die von der Java-Anwendung aus angesprochen wird, sowie für jeden komplexen Schnittstellenparameter innerhalb dieser Funktionen wird eine Java-Klasse mit den entsprechenden Hibersap-Annotationen implementiert.

Die Verwendung ist ähnlich der Verwendung des JPA API. Das API von Hibersap zum Aufrufen von Funktionen und für das Transaktionshandling ist recht schlank und ebenfalls von JPA-konformen O/R-Mappern wie Hibernate inspiriert. Beispielanwendungen für Hibersap und Cuckoo finden sich unter [Exampl].

Beim Einsatz von Enterprise JavaBeans (EJB) unterstützt Hibersap den Anwendungsentwickler zusätzlich durch die Bereitstellung der benötigten Ressourcen. Es ist insbesondere kein zusätzlicher Code für die Integration mit Java EE-Transaktionen erforderlich. Somit wird der Anwendungscode für den Aufruf von SAP-Funktionen deutlich verschlankt.

Darüber hinaus sind in der Hibersap-Community weitere Werkzeuge entstanden. So kann die Erstellung von Klientencode durch ein JBoss-Forge-Plug-in [SchwEr12] teilweise automatisiert werden. Der Anwendungsentwickler kann hierbei das SAP-Backend nach Funktionen durchsuchen und sich für einen gewählten Funktionsbaustein die entsprechenden Java-Klassen mit den Schnittstellenmappings durch das Plug-in erzeugen lassen.

Anwendungen, die Apache Camel und darauf basierende Produkte wie JBoss Fuse für Integrationsaufgaben verwenden, können auf eine Camel-Hibersap-Komponente zurückgreifen, die es ermöglicht, mittels Camel-Routen auf SAP zuzugreifen und dabei die Vorteile des Hibersap-Programmiermodells zu nutzen.

Fazit

Für einfache Anwendungsszenarien stehen mit dem SAP NetWeaver Gateway und den SAP Enterprise Services leichtgewichtige und offene Schnittstellentechnologien zur Verfügung, die auf dem Ansatz der RESTful Webservices beziehungsweise dem SOAP-Standard basieren. Bestehen allerdings höhere Anforderungen, wie Transaktionalität, hohe Geschwindigkeit oder die Verarbeitung großer Datenmengen, ist der Einsatz der Java Connectors (JCo) von SAP zu erwägen.

Mit Hibersap steht Anwendungsentwicklern ein flexibles und leichtgewichtiges Open-Source-Framework zur komfortablen Nutzung von JCo zur Verfügung. Insbesondere in Kombination mit dem Cuckoo Resource Adapter können unternehmenskritische Java EE-Anwendungen entwickelt werden, die hohe Anforderungen an die Datenintegrität zwischen den Java- und SAP-Systemen erfüllen.

Hibersap und Cuckoo wurden mit den gesammelten Erfahrungen aus vielen Java-Entwicklungsprojekten, die SAP-Systeme auf die unterschiedlichsten Weisen integrieren, entwickelt. Beide sind Open-Source-Projekte und stehen unter der offenen GNU Lesser General Public License (LGPL).

Literatur und Links

[CuckooRA] Cuckoo Resource Adapter for SAP,
`http://sourceforge.net/projects/cuckoo-ra`

[Exampl] Beispielanwendung für Hibersap und Cuckoo,
`https://github.com/cerker/cuckoo-hibersap-example`

[Hibersap] Hibersap,
`http://hibersap.sourceforge.net/current/index.html`

[JavaEE] Java EE,
`http://www.oracle.com/technetwork/java/javaee`

[JSR322] Java Specification Request 322: Java™ EE Connector Architecture 1.6,
`https://jcp.org/en/jsr/detail?id=322`

[OData] Open Data Protocol,
`http://www.odata.org`

[RedHatDV] Red Hat JBoss Data Virtualization,
`http://www.redhat.com/products/jbossenterprisemiddleware/data-virtualization/`

[RedHatMW] Red Hat Middleware,
`www.redhat.com/de/technologies/jboss-middleware`

[SAPERP] Enterprise Resource Planning von SAP,
`http://www.sap.com/germany/pc/bp/erp.html`

[SAPESWorkplace] SAP Enterprise Services Workplace,
`http://esworkplace.sap.com`

[SAPMP] SAP Service Portal,
`http://service.sap.com/connectors/`

[SAPNetWeaver] SAP NetWeaver Gateway,
`http://scn.sap.com/community/netweaver-gateway`

[SchwEr12] M. Schwab, C. Erker, JBoss Forge Plugin für Hibersap, 2012,
`http://blog.akquinet.de/2012/07/12/use-jboss-forge-to-generate-hibersap-classes-calling-sap-functions/`

Carsten Erker ist Senior Berater und Architekt bei der akquinet mit einem Schwerpunkt auf Java EE-Anwendungen. Neben seiner Projektarbeit engagiert er sich in verschiedenen Open-Source-Projekten.
E-Mail: carsten.erker@akquinet.de

Dr. Torsten Fink ist Geschäftsführer der Berliner Einheit der akquinet. Neben der Leitung von Java EE-Projekten führt er Architektur- und Technologieberatungen durch.
E-Mail: torsten.fink@akquinet.de