

Hibersap: Integration von Java- Anwendungen mit SAP



Newsletter
Januar 2010

Java spricht mit ABAP

In Java entwickelte Geschäftsanwendungen stehen selten allein in der Anwendungslandschaft. In vielen, vor allem größeren, Unternehmen müssen SAP-Systeme eingebunden werden. Über die Jahre hinweg entstanden hier verschiedene Kommunikationstechnologien, die die SAP-Welt mit der Außenwelt verbinden. SAP fokussiert sich selbst in zunehmendem Maße auf Java-Technologien. So basieren die Hauptinnovationen bei SAP seit einigen Jahren fast ausschließlich auf dem Java-Stack.

Betrachtet wird in diesem Newsletter die Kommunikation mit der SAP ECC (ERP Central Component, dem ehemaligen R/3), also dem ABAP-Stack von SAP. Hier läuft noch der Großteil der existierenden Geschäftsanwendungen. Der Java-Stack hingegen setzt auf die Java Enterprise Edition (Java EE [1]) auf und bietet von Hause aus Standard-konforme Integrationstechnologien. Es bieten sich somit dem Java-Entwickler verschiedene Möglichkeiten an und die Wahl ist wie so oft im Leben nicht einfach und hängt vom konkreten Einsatzszenario ab.

SAP setzt seit einigen Jahren verstärkt auf Web Services. So lassen sich mit relativ wenig Aufwand Web Services als Adapter für ABAP-Funktionen generieren, vorausgesetzt, es wird eine einigermaßen aktuelle SAP-Version eingesetzt. Diese Web Services lassen sich dann von außen mit den Standardmitteln aufrufen und natürlich können auch aus ABAP-Code heraus Web Services angesprochen werden.

Mit SAP Process Integration (PI, vormals Exchange Infrastructure, XI) bietet SAP eine Integrations-Middleware an, die eine Vielfalt an Funktionalität bietet, welche auf Web Services sowie auf SAP-proprietären Protokollen aufsetzt, jedoch hohe Anforderungen an Installation, Administration und Ressourcen stellt.

In jedem Fall rechtfertigen nicht alle Szenarien den Einsatz einer relativ schwergewichtigen SOA-Plattform wie PI, insbesondere auch dann nicht, wenn eine solche im Unternehmen noch nicht vorhanden ist und neu eingeführt werden muss.

Sowohl direkt verwendete Web Services als auch der Einsatz von SAP PI haben einen weiteren entscheidenden Nachteil: Sie unterstützen keine Transaktionen und sind somit eher für lose gekoppelte Systeme geeignet.

Erfahrungsgemäß benötigen Geschäftsanwendungen jedoch häufig eine engere Kopplung. Erstens sind dann nicht-funktionale Anforderungen wie Geschwindigkeit und Transaktionalität wichtiger. Zweitens sind bei einer engen Kopplung Geschäftsobjekte, mit denen eine Anwendung arbeitet, über mehrere Systeme verteilt. Daraus ergeben sich weitere Anforderungen:

- Minimale Redundanz: Es sollte vermieden werden, Geschäftsobjekte über Systemgrenzen hinweg zu duplizieren.
- Aktualität: Es sollte möglichst immer auf den aktuellen Daten aus den anderen beteiligten Systemen gearbeitet werden.
- Datenkonsistenz: Die Daten müssen über Systemgrenzen hinweg zu jeder Zeit konsistent sein. Dies wird i.d.R. durch verteilte Transaktionen sichergestellt.

SAP Java Connector

Das in der SAP-Welt wohl am meisten verwendete Kommunikationsprotokoll ist RFC (Remote Function Call). Ursprünglich für die Kommunikation zwischen ABAP-Systemen verwendet, entwickelte SAP auf Basis dieses Protokolls bereits vor etlichen Jahren den SAP Java Connector (JCo).



Newsletter Januar 2010

Hibersap: Integration von Java-Anwendungen mit SAP

Der Java Connector ermöglicht entfernte Funktionsaufrufe zwischen Java-Anwendungen und SAP-Systemen, und zwar in beide Richtungen (Java ruft ABAP und ABAP ruft Java auf). Da er das RFC-Protokoll verwendet, unterstützt er auch Transaktionen, ist also gut geeignet, das oben beschriebene Szenario einer engen Kopplung umzusetzen.

Der Java Connector liegt inzwischen in der Version 3 vor und ist sehr ausgereift und performant. So verwendet SAP den Java Connector für die Kommunikation zwischen ABAP- und Java-Stack und er wird häufig in Integrationsszenarien eingesetzt, die SAP PI verwenden.

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 nötig [2].

JCA-kompatible Resource Adapter

In Java entwickelte Geschäftsanwendungen laufen meist in einem Java EE kompatiblen Applikationsserver. Die Java EE unterstützt die Zusammenarbeit mit Fremdsystemen und definiert hierfür die Spezifikation der Java Connector Architecture (JCA) [3]. In der JCA übernimmt ein Resource Adapter die Kommunikation mit einem bestimmten Fremdsystem. Resource Adapter werden unabhängig von den Fachanwendungen im Applikationsserver deployt, konfiguriert und verwaltet. Die Anwendungslogik muss dazu i.d.R. nur die standardisierte Connector API kennen.

Neben der von den Anwendungen getrennten Verwaltung von Resource Adaptern gibt es einen weiteren großen Vorteil gegenüber der Verwendung proprietärer Schnittstellen wie dem Java Connector: Die Verwaltung von Transaktionen kann vom Transaktionsmanager des Applikationsservers übernommen werden. Mehr dazu weiter unten.

Der SAP Java Resource Adapter (SAP JRA [2]) ist eine JCA-Implementierung, die SAP mit dem eigenen Applikationsserver ausliefert. Leider ist dieser nicht ohne weiteres auf anderen Applikationsservern lauffähig, da er intern den SAP Java Connector in einer speziellen Ausprägung verwendet.

Für andere Plattformen bietet es sich an, den SAP BAPI JCA Adapter zu verwenden, ein Open Source Produkt, das kompatibel zu JCA 1.0 ist und intern ebenfalls den Java Connector benutzt, und zwar in der Standardvariante, sodass er auf sämtlichen Java EE kompatiblen Applikationsservern lauffähig ist [4].

JCo und Resource Adapter in der Praxis

Neben den oben beschriebenen Vorteilen haben beide Technologien auch einige Nachteile. So 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 verunreinigen und um wiederkehrende Aufgaben an einer zentralen Stelle zu kapseln. Für unterschiedliche Architekturen (Stand-alone Anwendungen, Java EE Server-Anwendungen, etc.) muss eine solche Abstraktionsschicht immer wieder neu entwickelt werden.

Beim Resource Adapter fällt dieser Punkt nicht so schwer ins Gewicht, da hier technische Details zum Teil durch die zentrale Konfiguration abgedeckt werden und gegen die schlankere und wohlbekannte Connector API entwickelt wird.

Sowohl bei Einsatz des Java Connectors als auch eines Resource Adapters fällt zudem erheblicher Aufwand beim Mapping der Schnittstellenparameter an. In beiden Fällen geschieht dies programmatisch, sodass jeder Parameter einer SAP-Funktion im Pro-



grammcode gesetzt bzw. ausgelesen werden muss. Da diese Schnittstellenparameter benannt sind und im Programmcode als Strings referenziert werden, ist die Prozedur außerdem recht fehleranfällig.

SAP hat dieses Problem bereits vor einiger Zeit erkannt und deshalb den Enterprise Connector entwickelt. Dieser ist in die SAP-Entwicklungsumgebung (Netweaver Developer Studio) integriert und bietet einen Wizard, mit dem der Anwendungsentwickler sich zu SAP-Funktionen Java-Klassen generieren lassen kann, die dann als eine Art lokaler Proxy für die entfernt aufzurufenden Funktionen dienen. Der Enterprise Connector hat jedoch zwei entscheidende Nachteile: Er ist erstens nur auf der SAP-Plattform verfügbar und erzeugt zweitens pro Schnittstelle sehr viele Java-Klassen und sehr schwer lesbaren Code, der nur aufwändig refaktoriert werden kann, da es keine saubere Trennung zwischen Geschäftsobjekten und technischem Programmcode gibt.

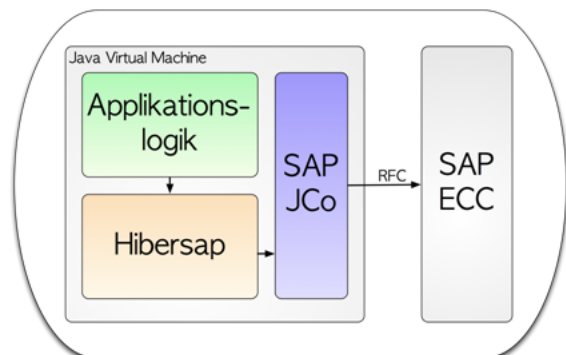
Während der Java Connector in beliebigen Umgebungen eingesetzt werden kann, also auch Integrationstests leicht zu implementieren sind, ist Code für einen Resource Adapter zunächst einmal nur in einer verwalteten Umgebung, also einem Java EE Applikationsserver, lauffähig. Dies erschwert Tests, die echte SAP-Systeme integrieren. Wegen des fehleranfälligen Parameter-Mappings (siehe oben) ergibt sich die Notwendigkeit solcher Tests jedoch fast zwangsläufig.

Hibersap

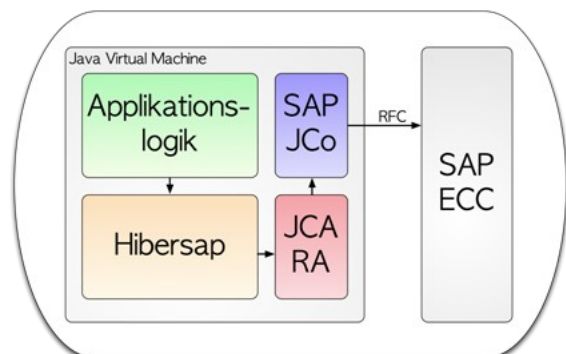
Mit den gesammelten Erfahrungen aus vielen Java-Entwicklungsprojekten, die SAP-Systeme auf die unterschiedlichsten Weisen integrieren, wurde von Entwicklern der akquinet das Framework Hibersap entwickelt [5] [6]. Hibersap ist ein Open Source Projekt und steht unter der offenen GNU Lesser General Public License (LGPL).

Hibersap stellt eine Abstraktionsschicht oberhalb des SAP Java Connectors oder eines JCA-kompatiblen Resource Adapters dar, wobei die Wahl zwischen diesen beiden Technologien konfigurationsabhängig ist und keine Änderung des Programmcodes notwendig macht. So kann derselbe Anwendungscode in der Produktivumgebung beispielsweise einen Resource Adapter verwenden, während Integrationstests mit dem Java Connector arbeiten.

Diese Skizze gibt einen Architekturüberblick bei Verwendung von Hibersap mit dem SAP Java Connector:



Das folgende Bild zeigt die Architektur bei der Verwendung von Hibersap mit einem Resource Adapter, der auf den Java Connector aufsetzt:





Parameter-Mapping

Das Mapping der Schnittstellenparameter bei der SAP-Anbindung mit Hibersap geschieht außerhalb des eigentlichen Programmcodes durch Java Annotationen. Für jede implementierte Schnittstelle, d. h. jede SAP-Funktion, die von der Java-Anwendung aus angesprochen wird, muss eine Java-Klasse implementiert werden. Diese Klasse wird mit der `@Bapi` Annotation markiert, und hier wird auch der Name der entsprechenden SAP-Funktion spezifiziert:

```
@Bapi( "BAPI_SFLIGHT_GETLIST" )
public class FlightListBapi
{ ... }
```

Jeder Funktionsparameter kann nun mit der `@Parameter` Annotation markiert werden, wobei der Name des Parameters spezifiziert wird. Zusätzlich wird mit einer der Annotationen `@Import`, `@Export` oder `@Table` die Art des Parameters angegeben.

```
@Import
@Parameter ( "FROMCITY" )
private final String fromCity ;

@Export
@Parameter (
    value = "RETURN",
    type = Parameter-
Type.STRUCTURE )
private BapiRet2 returnData;

@Table
@Parameter ( "FLIGHTLIST" )
```

Parameter mit einfachem Datentyp können so direkt einem Feld der Java-Klasse zugeordnet werden. Für Parameter mit komplexem Datentyp wird eine eigene Java-Klasse benötigt. Diese Klasse kann auch wiederverwendet werden, wenn in unterschiedlichen Funktionen dieselben Strukturen verwendet werden.

```
@BapiStructure
public class Flight
{ ... }
```

Die Java-Klasse zur Aufnahme der SAP-Daten wird mit der `@BapiStructure` Annotation ausgezeichnet. Die einzelnen Elemente der Struktur werden wie oben mit der `@Parameter` Annotation markiert, wobei der Name des Elements angegeben wird.

```
@Parameter ( "CONNID" )
private String connectionId;
```

Tabellenparameter in ABAP entsprechen Collections oder Arrays in Java. ABAP-Tabellenparameter haben einen Struktur-Datentyp, sodass auch hier eine weitere Java-Klasse für das Mapping der Strukturfelder benötigt wird. Diese wird ebenfalls mit der `@BapiStructure` Annotation markiert. Der Datentyp für den Tabellenparameter entspricht in Java dann einer Collection der Strukturklasse als generischem Typ.

Die so erzeugten Klassen können einfache Datencontainer sein, z. B. in der Form von Java Beans mit Zugriffsmethoden für die einzelnen Felder. Sie können darüber hinaus jedoch auch beliebige weitere Felder, Methoden oder Konstruktoren enthalten. Hibersap liest und setzt abgebildete Felder per Java-Reflection, sodass die Namen der Felder und Methoden sowie ihre Sichtbarkeit frei wählbar sind. Ausschlaggebend sind beim Zugriff durch das Framework die mit den Annotationen spezifizierten Informationen und die Datentypen der Felder.

Konvertierung von Datentypen

Bei der Abbildung von SAP-Strukturen mit Hibersap-Annotationen ist darauf zu achten, dass die Java-Felder den richtigen Datentyp haben. Dieser wird dadurch bestimmt, wie der Java Connector zwischen ABAP- und Java-Datentypen konvertiert. So werden z. B. Ganzzahlen nach `int`, Fließkommazahlen nach `double`, Datums- und Zeitangaben nach `java.util.Date` und Strings nach `java.lang.String` konvertiert.



Um auf die Konvertierung Einfluss zu nehmen, können Hibersap-Konverter implementiert werden. So ist es beispielsweise möglich, in Java Felder vom Typ boolean zu verwenden, obwohl es diesen Datentyp in ABAP nicht gibt (meist werden hier Textfelder für die Repräsentation boolescher Werte verwendet).

```
@Import
@Parameter ( "AFTERNOON" )
@Convert (
    con-
    verter=BooleanConverter.class )
```

Konfiguration

Die Konfiguration von Hibersap kann programmatisch, oder – in der Regel die empfohlene Variante – mit einer XML-Konfigurationsdatei erfolgen.

Hierbei wird vor allem festgelegt, ob der SAP Java Connector oder ein Resource Adapter verwendet wird und welche Mapping-Klassen Hibersap jeweils berücksichtigen soll.

Aufruf von SAP-Funktionen

Die Programmierschnittstelle von Hibersap ähnelt der des Datenbank-Mapping-Frameworks Hibernate [7] und bietet somit eine vielen Java-Entwicklern vertraute API.

Hier der Code, der nötig ist, eine SAP-Funktion aufzurufen:

```
FlightListBapi flightList
    = new FlightListBapi(
        "DE", "Frankfurt",
        "DE", "Berlin" );

session.execute( flightList );

session.close();
```

Zunächst wird eine BAPI-Klasse, wie sie oben (unvollständig) definiert ist, erzeugt und die benötigten Import-Parameter gesetzt (im Beispiel werden Flüge von Frankfurt nach Berlin im SAP gesucht). Mit dem Aufruf der Methode execute() an der Hibersap-Session findet der eigentliche Aufruf ins SAP-System statt. Hierbei wird die BAPI-Klasse mit den Ergebnisdaten, die SAP zurückliefert, von Hibersap „angereichert“. Diese können dann z. B. mit getter-Methoden gelesen werden. Um die Verbindung schließlich wieder freizugeben, wird die Methode close() an der Session aufgerufen.

Transaktionen

Um die Datenintegrität zwischen dem SAP-System und der Java-Anwendung sicherzustellen, sollten Transaktionen verwendet werden. Beim direkten Einsatz des SAP Java Connectors stehen in Hibersap entsprechende Methoden zur Verfügung, um Transaktionen manuell zu starten und zu beenden.

Wird Hibersap mit einem JCA-kompatiblen Resource Adapter eingesetzt, so stellt dieser lokale Transaktionen zur Verfügung. Das bedeutet, dass bei der Verwendung von Enterprise Java Beans (EJB) entweder Bean Managed Transactions (BMT) oder Container Managed Transactions (CMT) zum Einsatz kommen können. Im Gegensatz zu BMT muss sich der Anwendungsentwickler bei CMT nicht explizit um das Transaktionshandling kümmern, da dies der Transaktionsmanager des Applikationsservers übernimmt.

Mit lokalen Transaktionen kann zunächst jedoch nur sichergestellt werden, dass mehrere Aktionen, die im SAP-System ausgeführt werden, von einer Transaktion abgesichert werden, dass also entweder alle Änderungen im SAP festgeschrieben oder im Fehlerfall alle zurückgerollt werden. Speichert die Java-Anwendung zusätzlich Daten in einer lokalen Daten-



Newsletter Januar 2010

Hibersap: Integration von Java-Anwendungen mit SAP

bank oder sendet Nachrichten an eine Message-Queue, so ist eine verteilte Transaktion (XA-Transaktion) nötig, um die Datenintegrität über Systemgrenzen hinweg zu erhalten. Leider beherrscht SAP jedoch kein Two-Phase-Commit, das jedes System implementieren muss, das an einer verteilten Transaktion teilnimmt, und ist somit nicht XA-fähig. Zum Glück bieten die Transaktionsmonitore moderner Applikationsserver, wie z.B. auch der JBoss – Server, aber in der Regel ein Feature namens Last Resource Commit Optimization. Dies erlaubt es, dass maximal eine Ressource, die nicht XA-fähig ist, an einer verteilten Transaktion teilnehmen kann.

Szenarien, in denen verteilte Transaktionen benötigt werden, können somit mit Hibersap und einem Resource Adapter problemlos und mit wenig Aufwand entwickelt werden.

Zusammenfassung

Mit Hibersap steht Anwendungsentwicklern ein flexibles und leichtgewichtiges Open Source-Framework zur Verfügung, das es erlaubt, mit wenig Aufwand, sowohl bei dem initialen Aufsetzen als auch bei der Weiterentwicklung und Wartung, Java-Anwendungen mit SAP zu integrieren.

Insbesondere in Kombination mit dem SAP BAPI JCA Adapter können unternehmenskritische Java EE -Anwendungen entwickelt werden, die hohe Anforderungen an die Datenintegrität zwischen den Java- und SAP-Systemen erfüllen.

Weitere Informationen erhalten Sie unter:
info@akquinet.de

Verweise

- [1] <http://java.sun.com/javaee/>
- [2] <http://service.sap.com/connectors/>
- [3] <http://java.sun.com/j2ee/connector/>
- [4] <http://sourceforge.net/projects/sapbapijcaadapt/>
- [5] <http://sourceforge.net/projects/hibersap/>
- [6] <http://hibersap.sourceforge.net/>
- [7] <http://www.hibernate.org>