Mission Critical - Java auf dem Mainframe

Praxisbericht: Neue Bankanwendung für den Filialbetrieb und die Massendatenverarbeitung

Rainer Janz, Burkhard Seck

Das Thema „Java auf dem Mainframe" wird bei der Modernisierung von Altsystemen immer wichtiger. Ein erfolgreiches Beispiel aus der Praxis zeigt, wie man Mainframe- und Java-Welt einander annähern kann. Die Aareal First Financial verfolgt mit dem hier vorgestellten Projekt das Ziel, ihre Altsysteme im Bereich Kontoführung/Wohnungswirtschaft abzulösen und eine einheitliche Betriebsplattform für Massendatenbetrieb auf dem Mainframe und Filialbetrieb auf Clients zu schaffen. Dieses Projekt wird durch ein eigens entwickeltes, auf Java EE-Technologie basierendes Framework namens TIBET unterstützt.

Eine moderne einheitliche Betriebsplattform

Die Aareal First Financial Solutions AG ist ein Unternehmen der Aareal Bank Gruppe. Sie ist Innovationsführer im Bereich IT-Lösungen rund um den Zahlungsverkehr und die Kontoführung für immobilienwirtschaftliche Unternehmen. Gleichzeitig entwickelt die Aareal First Financial IT-Lösungen, die in der Mutterbank eingesetzt werden. Neben Standardanwendungen für die Kunden der Bank betreut und entwickelt die Aareal First Financial die zentrale Anwendung für die Kontoführung der wohnungswirtschaftlichen Kunden der Aareal Bank. Dieses zuletzt komplett neu entwickelte Kontoführungssystem umfasst neben den Kernbestandteilen für den Zugriff auf Stammdatenverwaltung, Kontoführungs- und Buchungsmodul umfangreiche in Java implementierte Batchkomponenten für die Massendatenverarbeitung auf dem Mainframe.

Mit Start des Projekts im Jahr 2003 verfolgte die Aareal First Financial das Ziel, die Altsysteme im Bereich Kontoführung/Wohnungswirtschaft abzulösen und eine einheitliche Betriebsplattform zu schaffen. Die Altsysteme, die im Wesentlichen aus einer Oracle-basierten Filialanwendung und eigenentwickelten Lösungen auf einem IBM-Mainframe bestanden, sollten ersetzt werden durch zeitgemäße und zukunftsfähige Neuentwicklungen. Außerdem sollte erstmals in der Aareal Bank ein echtzeitfähiges Banksystem für den Filialbetrieb integriert werden. Dies machte auch eine Ablösung des alten Kontoführungs- und Buchungsmoduls notwendig.

Als Plattform für die Massendatenverarbeitung und für die Stammdatenverwaltung und Kontoführung war von vornherein das moderne IBM-Mainframesystem z gesetzt.

Die Anforderungen an das neue System waren von Anfang an klar definiert:

- *Echtzeitfähigkeit:* Buchungs- und Zahlungsvorgänge sollten in Echtzeit verarbeitet werden, um die Kunden schnell zu bedienen und interne Prozesse zu beschleunigen. Die Konzentration der Aareal Bank auf die Immobilienwirtschaft bedingt dabei eine besondere Rolle der Massendatenverarbeitung. Die Kunden liefern aufgrund ihrer Größe ihre Aufträge in der Regel als Massendaten an.
- *Kapselung des wohnungswirtschaftlichen Know-hows der Bank in einer Java-Komponente:* Vorgabe der Bank war, dass das zentrale fachliche Wissen der Bank im Bereich Wohnungswirtschaft in einer Komponente zusammengefasst wird. Dadurch sollte verhindert werden, dass die Prozesse, die die Kernkompetenz der Aareal Bank ausmachen, in unterschiedlichen Systemen verstreut implementiert werden. So liegt das Wissen an einer Stelle und die Restsysteme sind vom wohnungswirtschaftlichen Teil abgegrenzt. Dadurch sind beispielsweise die Prozesse der eigentlichen Kontoführung von den wohnungswirtschaftlichen Besonderheiten unabhängig. Diese Anforderung machte es unter anderem nötig, Batchprozesse in Java zu implementieren.
- *Kosteneinsparungen:* Die neue Lösung soll durch die Verwendung von Java-CPUs deutlich kostengünstiger arbeiten. Anders als das althergebrachte Lizenzmodell für IBM-Mainframes, bei dem die CPU-Leistung abgerechnet wird, sieht das Lizenzmodell für Java-CPUs in IBM-Großrechnern vor, dass die CPU einmalig gekauft wird und dann mit voller Leistung zur Verfügung steht. Damit sorgt diese Variante für deutliche Kostentransparenz.
- *Reduzierung des Administrationsaufwands für Anwendungen und Support:* Durch die Zusammenlegung der Datenbestände aus mehreren unterschiedlichen Systemen und die vereinfachte Struktur soll der Administrationsaufwand begrenzt werden. Die tägliche Betreuung des Produktivsystems profitiert ebenfalls von der Übersichtlichkeit des neuen Systems.
- *Effektive Anwendungsentwicklung durch die Verwendung von Java:* Java als Programmiersprache ist den Kinderschuhen entwachsen. Kurze Entwicklungszeiten, Portierbarkeit, Sicherheit und ein breites Einsatzspektrum sind nur einige Vorteile der objektorientierten Sprache. Mit dem Java-Ansatz lässt sich außerdem eines der größten Probleme in der Mainframe-Welt beheben: Die Ressourcen für die Entwicklung in Sprachen wie COBOL oder PL1 werden immer knapper, weil

der Nachwuchs an Spezialisten in diesem Bereich fehlt. Für Java hingegen ist auf längere Sicht davon auszugehen, dass genügend neue Kräfte nachrücken.

Java im Batch- und Onlinebetrieb

Mit dem neuen Kontoführungssystem entstand eine Onlinelösung für den Sachbearbeiter, die mit einer Client-/Server-Architektur realisiert wurde. Das neue System nimmt aber auch die Umsetzung der Massendatenverarbeitung außerhalb des Buchungsmoduls mittels Java-Technologien auf dem Mainframe vor.

Die Vorteile dieser Lösung liegen in der optimierten Anwendungsentwicklung und der Möglichkeit zur Vermeidung von Doppelimplementierungen. Große Teile des fachlichen Java-Programmcodes werden parallel auf dem Großrechner im Batchbetrieb und online im Client-/Serverbetrieb verwendet. Eine zukunftsfähige Softwarearchitektur gewährleistet dabei sowohl die Effizienz der Systeme als auch ihre Offenheit gegenüber neuen Technologien.

Seit Ende 2005 ist das neue System im produktiven Einsatz und wird seitdem ständig weiterentwickelt und an neue Anforderungen angepasst. Der Übergang zwischen Alt- und Neusystem erfolgte nicht abrupt. Die Kundendaten wurden schrittweise in das neue System migriert.

Komplexität der Abhängigkeiten reduzieren mit sauberer Struktur

Die Onlineanwendung basiert auf einer klassischen Drei-Schichten-Architektur (s. Abb. 1). Der Client verwendet die Java-Swing-Technologie. Als Server kommt ein Cluster von IBM-Websphere-Applikationsservern zum Einsatz. Ein IBM z10-Mainframe als Backend-System stellt die Stammdaten durch DB/2 unter z/OS hochverfügbar und mit hoher Performance bereit. Parallel dazu läuft ein COBOL-Buchungssystem auf demselben Großrechner.

Die Clientanwendung beschränkt sich auf die Steuerung der Onlineanwendung und die Darstellung der Daten. Die Entwicklung der fachlichen Prozesse findet ausschließlich auf der Serverseite statt. Zu Projektbeginn 2003 befanden sich die heute so erfolgreichen Java-Frameworks, beispielsweise Hibernate und Spring, noch nicht in einem wirklich produktionstauglichen Zustand. Für das neue COBOL-Buchungssystem, das zeitgleich mit der Onlineanwendung eingeführt wurde, waren zudem einige spezielle Anforderungen zu erfüllen.

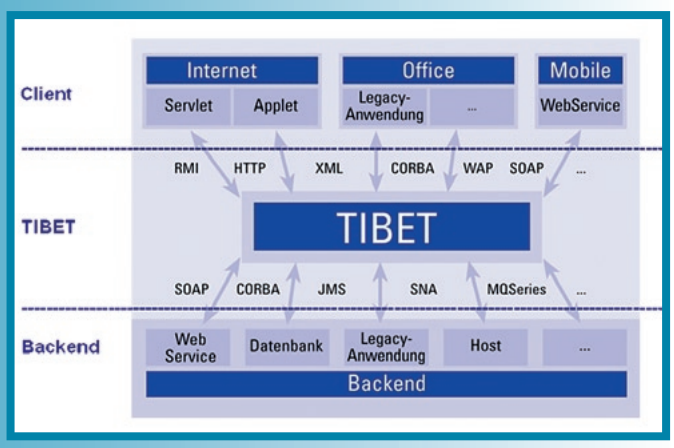

Abb. 1: Drei-Schichten-Architektur

Das führte zur Entwicklung eines auf Java EE-Technologie basierenden Frameworks mit Namen TIBET (Tricept Integrated Business Enterprise Technologie). Das Framework umfasst vorimplementierte Schnittstellen zur Datenbank und zum COBOL-Kontoführungs- und Buchungsmodul. Es deckt die Bereiche Transaktionssteuerung und Berechtigungs- und Kompetenzsteuerung ab und erleichtert durch standardisierte Geschäftsobjekte und -funktionen die fachliche Entwicklung.

Dabei kapselt TIBET die technischen Aspekte von Java EE so weit, dass sich der Entwickler auf die fachlichen Aspekte seiner Aufgabe konzentrieren kann. Die Unabhängigkeit des fachlichen Codes von Java EE geht so weit, dass man auf Java EE als Frameworkbasis verzichten und mit TIBET entwickelte Programme in einer ganz normalen virtuellen Maschine laufen lassen kann. Das ermöglicht es, TIBET auch für die Programmierung von Java-Batchprozessen einzusetzen.

Aufgrund der verwendeten Java-Technologie ist das Framework offen für zahlreiche Kommunikationsprotokolle, unabhängig von der Systemarchitektur und leicht adaptierbar an zukünftige Entwicklungen.

Das TIBET-Framework bildet die Basis der Serveranwendung für den Filialbetrieb und wird produktiv unter Linux auf IBM WebSphere eingesetzt. Java-Batchanwendungen mit TIBET laufen in der USS-Umgebung unter z/OS auf dem IBM-Mainframe als ganz normale Java-Prozesse in der Standard-VM von IBM. Für den Batchbetrieb wurde das Framework noch einmal optimiert und mit einer speziellen Batchkomponente ausgestattet, dabei konnten Schnittstellen und bisherige Implementierungen weiterverwendet werden. Abbildung 2 zeigt einen typischen Batchablauf.

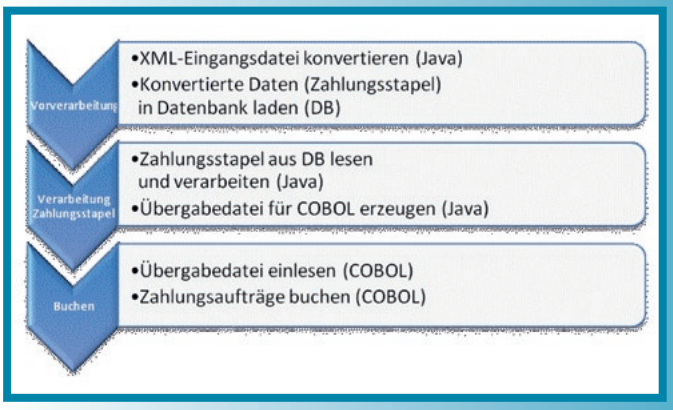

Abb. 2: Massendatenverarbeitung bei Zahlungseingang

Die Batcherweiterung des Frameworks vereinfacht die Implementierung von Batchjobs in Java und damit die Verlagerung von Batchprozessen auf die Java-Prozessoren im Mainframe, denn die Entwickler müssen sich bei der Programmentwicklung nicht mit den speziellen Gegebenheiten am Großrechner beschäftigen. Die Performance der Batchjobs ist dabei so hoch, dass die Aareal First Financial ihren Massenzahlungsverkehr und ihre Massenbestandspflegen komplett auf dem Host durchführen und dabei die engen Zeitvorgaben einhalten kann.

Während der Projektlaufzeit ergab sich weiterhin die Notwendigkeit, den bestehenden COBOL-Jobs am Großrechner eine einfache und effiziente Möglichkeit zu bieten, Java-Funktionen aus der Massendatenverarbeitung heraus als Batchjobs

TIBET im Batchbetrieb (Java Batch Anwendung auf dem Host)	TIBET im Client-/Serverbetrieb (Onlinelösung mit Client-Server-Architektur)
✓ Optimiert für die Massendatenverarbeitung ✓ Performanceoptimierte Java-Mainframe-Kommunikation ✓ Weniger Ressourcenverbrauch als im Onlinebetrieb ✓ Optimierte Anbindung an Core Banking Systeme in beide Richtungen ✓ Verwendung der gleichen fachlichen Schnittstellen	✓ Multikanalfähig, mandantenfähig ✓ Multilingual ✓ Plattformunabhängig (mit und ohne Application Server) ✓ Hohe Flexibilität durch einheitliche Komponentenorientierung ✓ Mechanismen der Performanceoptimierung ✓ Logische und physikalische Transaktionen

Tabelle 1: TIBET im Batch und online

aufrufen zu können. Dazu wurde ein hochperformanter Java-Kommunikationsserver für die Interaktion zwischen COBOL und Java entwickelt. Über ein XML-Interface werden die in Java implementierten Fachfunktionen angesprochen. So können Java-Host-Programme ohne größere Performanceverluste aus COBOL-Anwendungen genutzt werden.

Gute Gründe für Java auf dem Mainframe

Mit dem Einsatz von Java-Anwendungen auf dem Mainframe lassen sich nach kurzer Zeit Einsparungen an Kosten sowie an Zeit erzielen. Der Einsatz von Java als etablierter und zukunftsträchtiger Programmiersprache stellt sicher, dass Neuentwicklungen auf Jahre hinaus mit aktuellsten und modernsten Techniken und Methoden erfolgen können. Dazu kommt noch, dass im Vergleich zu den althergebrachten Lizenzmodellen das IBM-Lizenzmodell für Java-CPUs eine attraktive Alternative ist. Java-CPUs werden nicht nach Verbrauch bezahlt, sondern einmalig gekauft und stellen die volle Leistung unabhängig von der Laufzeit und Auslastung zur Verfügung.

Neben Kosteneinsparungen bringt die mögliche Zentralisierung der Systeme eine Reduzierung des Administrationsaufwands. Ebenso vereinfachen die vorgefertigten Strukturen des im Projekt entstandenen Frameworks TIBET die fachliche Entwicklung. Durch Wiederverwendung von Fachfunktionen im Batch- und Onlinebetrieb werden unnötige Doppelimplementierungen im System vermieden.

Performance über alles

Man darf allerdings auch nicht verschweigen, dass es aufgrund der enormen Unterschiede zwischen Mainframe- und Java-Welt eines längeren Lernprozesses bedarf. Es beginnt schon damit, dass Host- und Java-Entwickler unterschiedliche Sprachen sprechen. Die Faktoren Performance, Sicherheit und Stabilität spielen in der Mainframe-Welt eine viel größere Rolle als anderswo.

Java-Entwickler brauchen deshalb in der Regel zusätzliche Ausbildung, damit sie für den Mainframe geeignete Programme schreiben können. In dieser Ausbildung müssen sie z. B. lernen, dass man in bestimmten Fällen sogar auf anerkannte objektorientierte Praktiken verzichten muss, weil die Performance absolut im Vordergrund steht.

Listing 1 gibt einen Einblick in die Implementierung der Persistierung von Rückmeldungen in der Datenbank. Dieses Implementierungsbeispiel zeigt gleich mehrere Dinge:

- Auf den ersten Blick unterscheidet sich Java-Quellcode für den Mainframe kaum vom Java-Quellcode für andere Plattformen. Wenn man allerdings genauer hinschaut, fällt hier der Verzicht auf eine strikte Trennung zwischen Geschäftsobjekt und Persistenzschicht auf. Das geschah hauptsächlich, um die Erzeugung neuer Java-Objektinstanzen zu minimieren. Der Zugriff auf die Datenbank erfolgt hier nicht indirekt über ein Persistenzframework, sondern direkt über Prepared Statements, um auch große Mengen von Rückmeldungen mit der notwendigen Performance persistieren zu können.

```
public class RueckmeldungFactory
 extends biz.ff.batch.wowi.WoWiObjectFactory {
  protected static final java.lang.String INSERT_KEY = "RMINSERT";
  protected static final java.lang.String INSERT_SQL = "INSERT INTO " +
    biz.ff.batch.wowi.transaction.Connection.getSchema() +
      ".VWRUECKMELD (STAPEL_NR,AUFTRAGS_NR,RM_NR,STAPEL_NR_RM,“ +
      "MELD_STATUS,MELD_GRUND,MELD_ANZAHL," +
      "POSITIV_KZ,FINAL_KZ,OPTIM_SPERR_KONTR) “ +
      "VALUES (?,?,?,?,?,?,?,?,?,?)";
    // [...]
  protected RueckmeldungFactory() {
  }
  // [...]
  public void insertBO(
   biz.ff.batch.wowi.session.SessionContext newSessionContext,
   biz.ff.batch.wowi.WoWiObject newObject)
    throws java.lang.Exception {
    Rueckmeldung object = (Rueckmeldung) newObject;
    if (biz.ff.batch.wowi.base.Debug.DEBUG_TRACE) {
      // [...]
    }
    // Ermittele ein evtl. bereits prepared-tes Statement
    java.sql.PreparedStatement stmt =
      newSessionContext.getConnection()
       .getRegisteredStatement(INSERT_KEY);
    // Falls Statement noch nicht prepared,
    // dann tu das und registriere es
    if (stmt == null) {
      stmt = newSessionContext.getConnection().getJDBCConnection()
               .prepareStatement(INSERT_SQL);
      newSessionContext.getConnection()
               .registerNewStatement(stmt, INSERT_KEY);
    }
    stmt.clearParameters();
    stmt.clearWarnings();
    stmt.setLong(1, object.getStapelNr().longValue());
    stmt.setInt(2, object.getAuftragNr().intValue());
    // [...]
    stmt.setShort(10, object.getOptimSperrKontr().shortValue());
    stmt.executeUpdate();
    if (stmt.getWarnings() != null) {
      throw stmt.getWarnings();
    }
  }
}
```

Listing 1: Ein Blick in die Implementierung der Persistierung von Rückmeldungen in der Datenbank

- Auffällig sind auch die Verwendung voll qualifizierter Klassennamen und der Verzicht auf Import-Statements. Im Verlauf des Projekts hat sich gezeigt, dass es dadurch vor allem Umsteigern aus der Mainframe-Welt leichter fällt, sich in der Java-Welt zurechtzufinden. Sie wissen einfach schneller, mit welchen Klassen und Objekten sie es gerade zu tun haben.

Die Verwendung im Batchbetrieb kann auch Einfluss auf das Objektdesign haben, wie die Vererbungshierarchie in Abbildung 3 verdeutlicht. All diese Kontonummernklassen repräsentieren Datensätze aus einer einzigen Datenbanktabelle mit knapp zwanzig Spalten. Bestimmte Batchvorgänge belegten nun bei Benutzung von Standard-Kontonummernobjekten zu viel Hauptspeicher, benötigten gleichzeitig aber nur eine Teilmenge der Kontonummernattribute. Somit wurden aus rein technischen Gründen spezielle Objekttypen entwickelt, die nur noch genau über diese Teilmenge der Attribute verfügen. Dadurch verringerte sich der Hauptspeicherbedarf enorm.

Fazit

Das Thema „Java auf dem Mainframe“ wird uns die nächsten Jahre sicherlich noch begleiten. Allein vor dem Hintergrund, dass sich immer mehr Entwickler der klassischen Mainframe-Programmiersprachen dem Rentenalter nähern und kein Nachwuchs in Sicht ist, wird der Einsatz von Java auf dem Mainframe immer häufiger diskutiert werden müssen. Da ein Ende der technischen Entwicklung von Java als Programmiersprache noch lange nicht abzusehen ist, darf man gespannt sein, welche neuen Möglichkeiten sich dort noch ergeben und welche Chancen diese Entwicklungen der Mainframe-Welt zukünftig eröffnen.

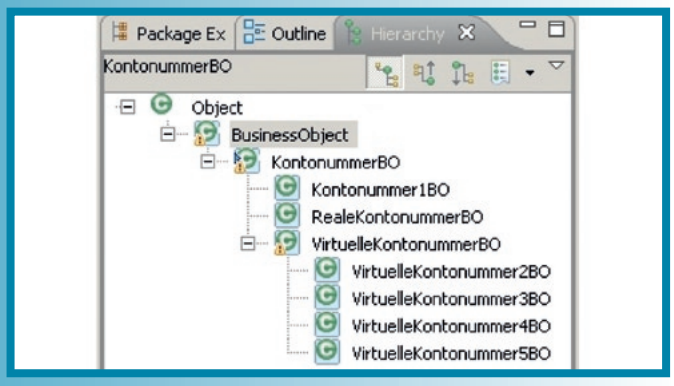

Abb. 3: Vererbungshierarchie der Kontonummernobjekte

Diplom-Informatiker **Rainer Janz** ist seit der Gründung 2002 bei der Aareal First Financial Solutions AG beschäftigt. Seitdem leitet er als Business Unit Manager im Bereich Development die Entwicklung des neuen Kontoführungssystems.

Burkhard Seck ist Senior Consultant und seit 1999 bei der Tricept Informationssysteme AG in Detmold beschäftigt. Er unterstützt die Aareal First Financial seit 2003 bei der Einführung und Entwicklung des Kontoführungssystems. E-Mail: burkhard.seck@tricept.de.