Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Thomas Ekert

Java unter Lotus Domino

Know-how für die Anwendungsentwicklung

Mit 121 Abbildungen, 80 Listings und CD-ROM

Springer

Thomas Ekert
Kommunikationsdesign
Hamburg
ekert@tom-quadrat.de
www.tom-quadrat.de

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

ISSN 1439-5428
ISBN-10 3-540-22176-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-22176-0 Springer Berlin Heidelberg New York

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

Satz: Druckfertige Daten des Autors
Herstellung: LE-TEX, Jelonek, Schmidt & Vöckler GbR, Leipzig
Umschlaggestaltung: KünkelLopka Werbeagentur, Heidelberg
Gedruckt auf säurefreiem Papier 33/3142 YL – 5 4 3 2 1 0

Vorwort

Mit einer explosionsartigen Verbreitung zog Java vor einigen Jahren auch in die Domino-Welt ein. Groß geworden mit Pascal, entsprach Java meinen Ansprüchen an die Ästhetik einer Programmiersprache, die gleichzeitig den aktuellen Anforderungen an die Leistungsfähigkeit moderner Anwendungen gerecht wird, und so machte ich Java schnell zum wichtigsten Standbein der Anwendungsentwicklung in vielen Projekten.

Vor allem in der Webentwicklung versprach – und verspricht – die Kombination mit Java große Vorteile für die Domino-Anwendungsentwicklung und so lag es nahe, meine seit Mitte der 90er Jahre in der Entwicklung verschiedenster kleiner und großer Domino-Anwendungen gesammelte Erfahrung nun um den Einsatz von Java zu erweitern.

Während für einen Domino-Programmierer – gewohnt an ein geschlossenes Entwicklungsframework bestehend aus dem Domino Designer, in dem alle Aspekte der Anwendungsentwicklung vom Debugging bis zur GUI Entwicklung abgedeckt werden – der Einsatz externer Komponenten aus Java neu ist und die Einarbeitung in neue Entwicklungsabläufe notwendig macht, steht der Java-Profi der komplexen und hoch integrierten Infrastruktur Dominos mit seinen ausgefeilten Sicherheitsmechanismen und der schier endlosen Zahl an Diensten und Anwendungen gegenüber.

Im Rahmen der Entwicklungsarbeit an einem großen Projekt bei der Erstellung eines geschlossenen Anwenderportals einer Versicherung wurde im Team aus Domino- und Java-Entwicklern schnell klar, dass der Austausch von Know-how an erster Stelle stehen musste, um eine gemeinsame Sprache zu finden, die es den jeweiligen Spezialisten ermöglichte, die Bedürfnisse und Notwendigkeiten der Domino-Entwicklung auf der einen und der Java-Entwicklung auf der anderen Seite zu verstehen und zu berücksichtigen. Hierbei entstand auch die Idee zu diesem Buch, da sich herausstellte, dass es zwar an Beschreibungen des Domino-Java-APIs nicht mangelt, aber Praxiserfahrungen, Best-Practice-Beispiele und vor allem Know-how in Domino-spezifischen Design Pattern und Lösungsansätzen rar sind, was nicht zuletzt in der noch recht jungen Historie der Domino-Java-Entwicklung begründet liegt.

So haben Domino- und Java-Entwickler jeweils einen spezifischen Blick auf die Aufgaben und Lösungsansätze für ihre eigenen Anforderungen und Bedürfnisse. Dieses Buch ist weder ausschließlich ein Handbuch für die Domino-Anwendungsentwicklung noch ein Java-Lehrbuch, sondern beleuchtet die speziellen Anforderungen der Domino-Anwendungsentwicklung in Java. Alle Fragestellungen, Lösungsansätze und Best-Practice-Beispiele sind aus der täglichen Praxis entnommen und weiterentwickelt und Bestandteil vieler tatsächlich im Einsatz befindlicher Anwendungen.

Als begeisterter und bekennender XP-Programmierer sind automatisierte Tests und Qualitätssicherungsverfahren wichtige Bestandteile meiner Arbeit und so enthält dieses Buch – last but not least – ein umfangreiches Kapitel zum Debugging oder besser gesagt zu dessen Vermeidung, ergänzt durch Verfahren, die automatisierte Tests von Domino-Java-Anwendungen ermöglichen.

Die Java-Programmierung für Domino ist aktueller denn je. Mit der derzeitigen Version gehören durch die DB2-Integration Themen wie SQL und J2EE nun zu den wichtigen Fragestellungen im Domino-Umfeld. Große Begeisterung und Erwartungen ruft der Blick auf die nächsten Domino-Generation hervor, deren Client auf Java

und dem Eclipse-Framework basieren wird. Damit wird die Basis für Domino-Plugins und Erweiterungen gelegt, die sich mittels des Domino-Java-API nahtlos integrieren lassen.

Dieses Buch war nur durch die tägliche Arbeit und im Austausch mit vielen Domino- und Java-Spezialisten in meinem Umfeld möglich. Ich möchte daher meinen MitarbeiterInnen, aber auch den KollegInnen bei unseren Kunden danken, die es durch Fragen, Gespräche, Anregung und Kritik möglich gemacht haben, Antworten und Lösungen zu finden und Design Pattern zu entwickeln.

Eine große Stütze war der Springer-Verlag, der schnell von Idee und Thema überzeugt war und mich mit Geduld begleitet hat.

Thorsten Dietsche gilt mein Respekt und Dank. Er hat mit Ausdauer das Manuskript konstruktiv und kritisch hinterfragt und viel zur Geschlossenheit des Inhalts beigetragen. Ihm verdanke ich unter anderem die Anregung für das Kapitel über die Kapselung von Schleifen.

Danken möchte ich meinen beiden Familien und meinen Freunden, die mich während der fast zwei Jahre der Entstehung dieses Buches nicht vergessen haben.

Mein besonderer Dank gilt Nico, die nicht nur auf viel gemeinsame Zeit verzichtet, sondern die mich antrieb, dieses Buch zu beginnen und dessen Abschluss erst ermöglicht hat.

Thomas Ekert
Hamburg im April 2006

Über dieses Buch

Dieses Buch richtet sich vor allem an Java-Programmierer, die neu in der Domino-Welt sind. Dementsprechend bietet der erste Teil eine Einführung in die für die Java-Entwicklung wichtigen Domino-Komponenten, stellt die Inhalte aber bereits für die Java-Sicht optimiert dar. So eignet sich dieses Buch auch für Domino-Anwendungsentwickler, die in die Domino-Java-Programmierung einsteigen, wobei allerdings solide Java-Grundkenntnisse vorausgesetzt werden.

Um die praktische Arbeit zu erleichtern, werden an vielen Stellen administrative Techniken beschrieben, die notwendig sind, um bestimmte Verfahren anwenden oder zugehörige Services und Servertasks aktivieren zu können. Auch wird auf viele Aspekte der Sicherheitsfunktionen eingegangen, auch wenn diese über die reine Anwendungsentwicklung hinausgehen oder administrativer Natur sind.

Teil I

Domino ist zugleich Datenbank und Anwendungsserver und dazu eine Plattform für Kollaborationsanwendungen wie E-Mail, Kalender, Aufgaben oder Ressourcenverwaltung.

Datentypen, Gestaltungselemente und der Domino Designer liefern die Grundlage, auf der das Buch aufbaut. Masken, Ansichten, Agenten und natürlich der XML-ähnlichen, offenen Schemastruktur Dominos gilt das Hauptaugenmerk, stellen diese doch die Grundlagen für sämtliche Anwendungen dar. Im ersten Teil dieses Buches werden die wichtigsten Komponenten erläutert und dienen als Basiswissen für die weiteren Kapitel.

Teil II

Der zweite Teil dieses Buches legt die Grundlagen für jede Domino-Java-Anwendung und erweitert dieses Wissen durch die Erläuterung von Best-Practice-Beispielen, Design Pattern und Spezialfällen. Die Domino-Session, als lokale oder remote Anwendung, bildet den Einstieg, ist sie doch das Herz einer jeden Domino-Java-Anwendung, erzeugt durch die NotesFactory, der zwei Kapitel gewidmet sind.

Die Handhabung der Domino-Objekte erfordert viel Spezialwisssen, so dass sich jeweils ein Kapitel mit den Objekten Session, Document, Database, View und RichText beschäftigt. Hierbei ist der Überblick über die gesamte Objektstruktur ebenso berücksichtigt wie die Aufgabenstellungen der täglichen Arbeit, von der Performanceoptimierung über die Handhabung von Threads bis hin zur Garbage-Collection und dem in Domino sehr wichtigen Recycling, dem ebenfalls ein eigenes Kapitel gewidmet ist.

Werden die wichtigsten Objekte beherrscht, geht es nun an den praktischen Einsatz. Dieser wird vor allem in den Kapiteln 13 bis 15 über die Objekt- und Speichertechniken und die verschiedenen Suchalgorithmen berücksichtigt, so dass ein kompletter Wissensschatz entsteht, der den Aufbau solider, alltagstauglicher Java-Anwen-

dungen für Domino ermöglicht. Dort findet sich eine Anleitung für den Einsatz von Domino mit DB2 ebenso wie ausführliche Modelle zur Erweiterung und Implementierung von Domino-Java-Objekten.

Teil III

Wie eingangs erwähnt, bietet vor allem der Einsatz im Web-Umfeld, aber auch in großen Enterprice-Infrastrukturen eine besondere Motivation für den Einsatz von Java für Domino.

Daher liefert Teil III das Spezialwissen für dieses Einsatzgebiet von der Entwicklung von JSP-Seiten mit den domtags über das Wissen für die Einbindung von Domino in eine J2EE-Infrastruktur.

Abgerundet wird das Buch durch das letzte Kapitel über Debugging, JUnit Tests, Logging und Qualitätssicherung.

Sourcecode

Alle Sourcecode-Listings der Beispiele dieses Buches liegen als vollständige Code Sammlung auf einer CD bei. Eine begleitende Web-Site befindet sich unter http://www.tom-quadrat.de/dominojava. Darüber hinaus wurden der CD ergänzende Beispiele und Klassen beigefügt, die als Listing nicht abgedruckt wurden. Dies ist jeweils im Text erwähnt.

Notationelle Konventionen

Werden im Text Java-Klassen oder Notes-Formeln erwähnt, so sind diese in Courier gesetzt, es sei denn es ist allgemeinsprachig vom gleichnamigen Domino-Objekt die Rede (z.B. „Die Domino-Session“ oder „Die Klasse `Session`“). Den Beispiel-Listings im Buch wurden in der Regel grafische Nummernsymbole hinzugefügt, über die die jeweiligen Codestellen im Text referenziert werden. Methoden sind nicht immer in ihrer vollen Signatur genannt, insbesondere dann, wenn es mehrere Signaturen gibt und eine Methode im Allgemeinen erwähnt wird.

In Tabelle 6-1 befindet sich eine vollständige Übersicht über alle Klassen des Domino-Java-API. Ein vollständiges Klassen- und Methodenverzeichnis ist in den Index im Anhang eingearbeitet.

Über den Autor

Thomas Ekert ist seit 10 Jahren freier IT-Berater und führt seit 7 Jahren als CTO und geschäftsführender Gesellschafter die IT-Agentur BITSDONTBYTE. Als zertifizierter Domino- Java- und DB2-Entwickler ist er Spezialist für Domino- und Web-Application-Server-Entwicklung.

Als Hauptverantwortlicher für große Projekte für öffentliche und private Auftraggeber ist er erfahren in Controlling und Qualitätssicherung von umfangreichen TÜV-geprüften Java-basierten Domino-Projekten, in denen er neben Projektierung, Konzept und Leitung auch Schulungen in Unternehmen konzipiert und durchführt.

Inhalt

Teil 1

Notes und Domino

Teil 1 von *Java unter Lotus Domino* wird Sie mit Domino infizieren. Grundlagen der Programmierung unter Domino und das besondere Konzept der Domino-Datenbanken und deren Datenstrukturen bieten eine besondere Chance im Rapid Development, insbesondere beim Einsatz von Java für Domino-Anwendungen.

Das grundlegende Verständnis der besonderen Domino-Programmiertechniken wird zur Basis für die Verwendung der Domino-Java-Objekte, abgerundet durch einen kurzen historischen Exkurs der Entwicklung der Domino-Plattform.

Neben der Einführung in die Datenstrukturen gehört ein Überblick über die Domino-Entwicklungswerkzeuge ebenso dazu, wie das Verständnis des Domino-Servers als Plattform, der Domino-Designelemente und der speziellen Techniken der Domino-Anwendungsentwicklung.

1 Das Domino-Prinzip

In diesem Kapitel:

Die Vision von Lotus Notes und Domino

Besonderheiten von Domino als Datenbank

Geschichte

Lotus Domino ist als meistverbreiteter Application- und Collaborationserver für viele mittlere und große Unternehmen der Quasi-Standard für Groupware und Anwendungsplattform.

Durch die gleichzeitig explosionsartige Verbreitung von Java als Basis vieler Anwendungen, die Einbindung von Java in Domino und die konsequente Bindung an Standards durch Lotus/IBM findet Java zunehmend Verbreitung als Basis für Domino-Anwendungen und führt gleichzeitig zur weiteren stetig steigenden Verbreitung von Domino.

Einer der Erfolgsfaktoren von Lotus Domino ist die Einbindung von einigen wesentlichen Funktionalitäten schon von der ersten Version im Jahr 1989 an. Dies sind Prinzipien, die heute zur Basis der meisten Anwendungsarchitekturen im Bereich Collaboration geworden sind.

Hierzu gehören:

- Konsequente Umsetzung der Client-Server-Architektur und das Konzept des Remotezugriffs mit der Bereitstellung von Mechanismen für die Remoteeinwahl.
- Verschlüsselung und Signatur von Daten und Authentifizierung durch die Verwendung einer RSA-Public-Key-Infrastruktur, um Dokumente und Daten nicht nur sicher zu transportieren, sondern auch gegen unerlaubte Veränderung zu schützen.
- Namens- und Adressbücher. Bereitstellung von zentralen Namens-Diensten zur einfachen Administration und Verwaltung von Benutzern und Gruppen und deren Berechtigungen.
- Replikation von Daten. Die Replikation ist eins der Alleinstellungsmerkmale von Lotus Domino. Sie ermöglicht nicht nur den einfachen Austausch und sicheren Abgleich von Daten zwischen mehreren Servern, sondern auch zwischen Server und Client. So wurde von der ersten Stunde an die Ressourcen sparende Offlinearbeit am Remoteclient möglich.
- Schutz von Daten auf Datensatz- und Feldebene durch ACL-Zugriffskontrolllisten.
- Organisation von Daten auf Basis von Schemata.
- Portierbarkeit der Datenbanken und Anwendungen auf eine Vielzahl von Server- und Client-Plattformen und eine Vielzahl von Betriebssystemen und Prozessoren. Erreicht wird dies durch eine Architektur in Layern, die die Betriebssysteme von der eigentlichen Domino-Anwendung trennen, so dass der Code für die verschiedenen Plattformen parallel entwickelt werden kann.
  Dies gilt nicht nur für die Entwicklung der Kern-Anwendung Lotus Domino, sondern auch für die Anwendungen, die auf Basis von Lotus Domino entwickelt werden. Die Erstellung von Designelementen, wie Masken und Ansichten, aber auch von Code auf Basis von LotusScript, einem Visual-Basic-Derivat, oder auf Basis von Java erfolgt Plattform-unabhängig und kann auf allen Plattformen, die Lotus Domino unterstützt, eingesetzt werden.
- Remoteadministration.
- Customization der Datenbanken.

- Rapid Development auf Basis von integrierten Entwickler-Tools. Lotus Domino war schon in der ersten Version eine Plattform, die neben einer Vielzahl von Basisanwendungen, wie E-Mail, Dokumentenmanagement, Diskussionsdatenbanken, eine Plattform für die Entwicklung neuer Anwendungen darstellte, die sich der durch die Plattform zur Verfügung gestellten Tools und Infrastruktur bedienen konnten.
- Verwendung internationaler Standards für alle Protokolle und Programmiersprachen. Kaum ein Produkt integriert so konsequent wie Lotus Domino internationale Standards, bei strikter Einhaltung der RFC.
- Konzept des RichText zur Eingabe von gestalteten Texten.
- Kontextuelle Hilfe.
- Verlinkung von Inhalten.

Neben der einerseits strikten Einhaltung von Standards und der gleichzeitig umfangreichen Implementierung vieler dieser Standards – so kann ein Anwendungsentwickler bei der Implementierung direkt auf vorgefertigte Server und Tools, von LDAP über POP und SMTP, von Newslettersystemen bis zum HTTP und nicht zuletzt auf eine robuste RSA-Public-Key-Infrastruktur zurückgreifen, ohne hier selbst neu entwickeln zu müssen – basiert Lotus Domino auf Prinzipien, die die große Verbreitung und gleichzeitig die Alleinstellung und den Erfolg von Lotus Domino begründen.

1.1 Replikation

Immer wieder wird die Replikation als Argument für Lotus Domino ins Feld geführt. Zu Recht. Kein anderer Application Server vereinfacht die Replikation von Daten derart radikal.

Datenbanken werden über die gesamte Infrastruktur einfach als so genannte Repliken, man könnte auch von Instanzen reden, angelegt. Serververbindungen machen die Server miteinander bekannt und regeln den Zeitplan, nach dem der Datenabgleich erfolgen soll. Zugriffskontrolllisten sorgen für die Sicherheit.

Der Vorgang der Replikation als solcher ist völlig transparent für die Anwender. Es werden alle Repliken im Serververbund – sofern die Replikation aktiviert ist – miteinander abgeglichen. Änderungen auf allen Instanzen werden auf alle anderen beteiligten Instanzen übertragen. Der Vorgang lässt sich so fein und granulär kontrollieren, dass selbst Änderungen einzelner Felder zwischen Datensätzen gemischt werden können.

Konflikte, die durch Änderungen gleicher Datensätze auf verschiedenen Servern entstehen können, werden erkannt und in Konflikt-Datensätzen gespeichert.

Bei der Java-Programmierung der Backendverarbeitung ist es wichtig das Konzept der Replikation zu verstehen und zu berücksichtigen. Das Verständnis verteilter Umgebungen ist daher Bestandteil dieses Buches (s. Kap. 2.3).

1.2 Schemabasiertes Objekt- und Speichermodell

Das Herz jeder Domino-Anwendung ist die Domino-Datenbank oder auch die „Note"-Datenbank – vereinfacht gesprochen ein Container für „Datensätze". Das Konzept von Lotus Notes – an dieser Stelle wird zur Verdeutlichung der ursprüngliche Name des Lotus-Domino-Servers verwendet, der seinen Namen erst im Jahr 1996 erhielt – sieht vor, dass alle Bestandteile einer Anwendung in der Notes-Datenbank, dem NSF (Notes Storage Facility), File gespeichert werden.

Gemeint ist, dass wirklich alles, vom Datensatz bis zur Zugriffskontrollliste im NSF-File gespeichert wird. Umgekehrt ist jeder Bestandteil einer Notes-Anwendung eine „Note". Jedes Designelement – so heißen neben WYSIWYG Gestaltungselementen wie Masken, die Programm-Bibliotheken von Lotus Notes – wird in einer „Note" gespeichert. Auch z.B. die ACL oder Datenbank-Eigenschaften werden in speziellen „Notes" gespeichert.

Dieses Konzept, nur eine einzige Datei für eine gesamte Anwendung als zentrales Repository anzulegen, ist unerlässlich für eine leicht zu administrierende Replikation und natürlich ein großer Vorteil für die Administration insgesamt. Selbstverständlich werden insbesondere die eigentlichen Daten in Notes gespeichert. Diese Notes werden auch als Dokumente bezeichnet.

Jedes Dokument und jede „Note" besteht nun wieder aus mehreren so genannten Items, die in einem ersten Ansatz als Felder bezeichnet werden können, vergleichbar mit den Spalten einer SQL-Tabelle, wobei später noch näher auf die Eigenschaft von Items eingegangen wird. Items können einerseits mehrere Werte und andererseits in bester objektorientierter Manier wiederum Items aufnehmen.

Das Speicherkonzept von Lotus Domino unterscheidet sich jedoch wesentlich von SQL-Datenbanken. In diesen Datenbanken wird exakt festgelegt, welche Regeln den Daten zugrunde liegen und in welchen Verhältnissen diese zueinander stehen. Dies erfordert eine strikte Administration und umfangreiche Planung, auch in der späteren Handhabung der Daten.

Das Konzept von Notes geht hier einen anderen Weg und speichert die Daten in losen Schemata, d.h. ein einzelnes Dokument kann beliebige Items oder auch Felder enthalten. Lediglich über Masken, die eine unverbindliche Vorgabe bei der Erstellung von Dokumenten geben, werden Dokumente kategorisiert und mit einem – zunächst – stringenten Schema versehen, das jedoch nicht bindend ist oder erzwungen wird.

Am ehesten lässt sich die Datenstruktur von Domino mit der von XML vergleichen. Auch dort werden die Daten in Namens-Wert-Paaren gespeichert und sind nicht zwingend, wenn auch in der Regel üblich, einem Schema unterworfen. Die Tatsache, dass Lotus Domino nicht relational aufgebaut ist und insbesondere, dass die Schemata nicht erzwungen werden, wird oft als Kritik an Lotus Domino genannt und ist ungewohnt für Programmierer, die aus der SQL-Entwicklung kommen.

Die Einhaltung eines Schemas und der daraus entstehende Verwaltungs- und Programmieraufwand ist jedoch nicht unerheblich und erfordert ein hohes Niveau an Kenntnissen und Erfahrungen. Dies hat die Begründer der Lotus-Notes-Datenstruktur dazu bewogen sich zugunsten eines leicht zu handhabenden und zu erweiternden Datenkonzeptes zu entscheiden.

Die offene Schemastruktur und Objektorientierung der Lotus Notes NSF-Datei ist eine Stärke von Lotus Domino und ermöglicht die Erstellung von robusten Datenbankanwendungen im Rapid Development.

Hieraus folgt nicht nur, dass in Lotus Domino Datenbanken bevorzugt nicht strukturierte Daten, wie z.B. in Diskussionen gesammelte Beiträge, gespeichert werden können, sondern auch, dass entsprechende Anwendungen, wie Workflow- oder Kollaboration-Anwendungen bevorzugt realisiert werden.

Die offene Schemastruktur und Objektorientierung der Lotus Notes NSF-Datei ist eine Stärke von Lotus Domino und ermöglicht die Erstellung von robusten Datenbankanwendungen im Rapid Development.

SQL-Datenbanken dagegen speichern bevorzugt strukturierte Daten, wie z.B. Produkt-Kataloge, die stark auf relationale Verknüpfungen zwischen Einzeldaten angewiesen sind, um Redundanzen zu vermeiden.

Mit der Einführung von Version 7 von Domino hat dieses Bild eine grundlegende Erweiterung erfahren. Mit Lotus Domino R7 wird die Möglichkeit eingeführt, die Domino-Daten in einer DB2-Datenbank zu speichern, also in einer relationalen SQL-Datenbank, wobei diese Art der Speicherung für den Notes-Programmierer oder Anwender zunächst transparent bleibt. Zusätzlich wird es die Möglichkeit geben, Domino Views anzulegen, die SQL-Daten aus DB2-Datenbanken direkt über SQL Queries laden können. Durch die Speicherung von Domino-Daten in DB2 einerseits und die Selektion von SQL-Daten andrerseits werden neue Möglichkeiten eröffnet. So wird es hierdurch zum Beispiel möglich sein, Daten aus mehreren Notes Views oder auch Notes-Datenbanken in einem neuen View anzuzeigen, der diese Daten entsprechend mischt. Dies war bisher in Domino nicht ohne weiteres möglich.

Zum einen folgt hiermit IBM den Anforderungen des Marktes, SQL-Daten mit Hilfe der leicht zu erlernenden RAD Tools des Domino Designers anzeigen zu können und andrerseits die durch Domino-Anwendungen erzeugten Daten in einer relationalen Struktur speichern zu können. Gleichzeitig eröffnet IBM hiermit den Entwicklern die Möglichkeit das beste aus beiden Konzepten zu vereinen. Diese Weiterentwicklung bietet einen weiteren Schritt der Integrationsmöglichkeiten.

Java-Anwendungen auf Basis von J2EE können noch nahtloser in Domino-Anwendungen integriert werden.

1.3 Erweiterbarkeit

Eines der Schlüsselkonzepte von Lotus Domino war und ist die Erweiterbarkeit. Zwar bietet Domino seit Beginn an Basismodule aus dem Bereich der Kollaboration, aber diese waren immer nur die Basis und ein Angebot an die Programmierer, sie zu verwenden und zu erweitern.

Bei der Entwicklung von Domino-Anwendungen kann zwischen zwei Elementen unterschieden werden, einerseits den so genannten Designelementen, mit denen die GUIs entwickelt werden und zum anderen programmatischen Code, der im Wesentlichen zur Programmierung von Backend-Prozessen verwendet wird.

Die wichtigsten Designelemente, mit denen Notes-Datenbanken programmiert werden sind Masken und Ansichten. Masken sind Formular-Seiten, die zur Benutzer-Interaktion, insbesondere zur Eingabe und Anzeige von Daten entwickelt werden.

Ansichten dienen der Selektion von Daten und der Darstellung als Listen oder Tabellen. Sowohl in Masken als auch in Ansichten können so genannte @-Formeln eingesetzt werden, die im Wesentlichen der Validierung von Eingaben oder der berechneten Anzeige von Inhalten dienen. Durch die Designelemente können Anwendungen effizient und schnell erstellt werden.

Java war und ist der Booster für die Erweiterbarkeit von Domino-Anwendungen, insbesondere durch den weiten Markt an Open-Source-Bibliotheken.

Um im Backend Domino-Daten verarbeiten zu können, führte Lotus im Januar 1996 für Version 4.0 die Programmiersprache LotusScript und im Dezember 1996 für Version 4.5 die Möglichkeit, per Java auf Domino-Objekte zuzugreifen, ein. Im Gegensatz zu LotusScript wurden für Java nur die Backend-Objekte zugänglich gemacht, so dass zunächst LotusScript wesentlich stärker verbreitet war.

Inzwischen ist Java die Core-Sprache bei der Neuentwicklung von Domino-Anwendungen. Java bietet Möglichkeiten der Programmierung, die bisher mit LotusScript nicht möglich waren.

Hierzu gehören:

- Multithreading-Anwendungen
- Verarbeitung von URL-Verbindungen, wie z.B. HTTP, FTP, SSH etc.
- Einbinden von vorgefertigten Open-Source-Bibliotheken
- Verarbeitung von Streams (diese Möglichkeit gibt es seit Version 6 auch für LotusScript)
- SQL-Datenbankbindung durch native JDBC-Treiber

Durch den Einsatz von Java ist Domino zu einem vollwertigen Application Server erwachsen, der viele der J2EE-Standards unterstützt. Mit der Version 6 bietet Lotus die Möglichkeit, zusätzlich zu Domino einen WebSphere Application Server einzusetzen, der über Connector Plugins mit Domino verbunden wird. Hierdurch eröffnet sich die Möglichkeit vollwertige J2EE-Anwendungen zu schreiben, die gleichzeitig Zugriff auf die Domino-Daten haben. Hierdurch ergibt sich ein typisches Szenario, wie z.B. für ein Domino-basiertes Web Content Management. Die Redaktion bedient sich des Notes Client, um Inhalte zu pflegen. Gleichzeitig wird durch WebSphere ein Darstellungslayer realisiert, um die Daten in einer leistungsfähigen und robusten Web-Anwendung darzustellen.

Für die Verwendung von Java bei der Programmierung und Erweiterung von Domino-Anwendungen ergeben sich drei wesentliche Einsatzgebiete:

- Backendverarbeitung von Domino-Daten
- Web-Anwendungen mit Zugriff auf Domino-Daten
- Enterprise-Datenbankanbindungen für Domino-Applikationen

Auf alle drei Anwendungsgebiete wird in diesem Buch eingegangen werden.

1.4 Zugriffskontrolle und Verschlüsselung

Domino ist *die* erste wichtige kommerzielle Software, die Verschlüsselung, Signatur und Authentifizierung auf Basis von öffentlichen RSA-Schlüsseln anbot [Lotus, History]. Gleichzeitig ist dies eines der Hauptargumente, das am häufigsten für Lotus Domino ins Feld geführt wird. Zu Recht: Lotus Domino, aber auch seine Komponenten-Server wie LDAP oder HTTP werden zu den sichersten am Markt gezählt.
Durch das einfache Management der Benutzer in den öffentlichen Notes-Adressbüchern, die entweder nativ oder per LDAP angesprochen werden können, bietet sich dem Java-Programmierer ein robustes und ausgereiftes Framework für die Verwaltung von Rechten.
Das Konzept dieses Frameworks ist ausgereift und sehr granulär steuerbar. In einer Abstufung der Rechte, vom Server bis hin zu einzelnen Feldern in Dokumenten, kann der Programmierer genau festlegen, welche Daten eingesehen werden können.
Durch die RSA-Infrastruktur stehen reichhaltige Werkzeuge zur Verschlüsselung von Daten und Datenströmen und Identifizierung von Benutzern und deren Rechten zur Verfügung. Darüber hinaus können Schlüsselpaare definiert werden, mit denen Benutzer, die Zugriff auf diese Schlüssel haben, ihre Daten oder Teile davon schützen können. Dies wird im Wesentlichen durch das Speicherkonzept der losen Schemata ermöglicht, da hierdurch objektorientiert zu schützende Items definiert werden können.
Der Zugriff auf Daten wird in mehreren Schritten abgesichert. Zunächst wird der Zugriff auf den Server überprüft. Der Zugriff erfolgt immer gegen das so genannte Notes-Adressbuch. Allerdings kann das Adressbuch über die so genannte Directory Assistance entweder durch sekundäre Adressbücher oder durch LDAP-Verzeichnisse erweitert werden. So ist es zum Beispiel einfach möglich, ein Active Directory in ein Notes-Adressbuch einzubinden.
Grundsätzlich wird beim Zugriff nach zwei Arten unterschieden. Zum einen der Zugriff über den Notes Client und zum anderen der Zugriff als so genannter Web-Access. Der Zugriff über den Notes Client erfolgt immer mittels einer ID-Datei, in der die privaten RSA-Schlüssel gespeichert sind. Beim Web-Zugriff können verschiedene Authentifizierungsmethoden gewählt werden. Es stehen Login-Verfahren mit X.509-Client-Zertifikaten, mit Name und Passwort oder das anonyme Login zur Verfügung.

Nach der Überprüfung auf den Serverzugriff erfolgt eine Überprüfung der ACL, der so genannten Zugriffskontrollliste (Access Control List). In dieser werden die einzelnen Rechte nach Benutzern, Servern und Gruppen eingetragen, die unterschiedliche Rechte als Leser, Autor, Datenbankdesigner oder Manager (s. auch Kap. 9.5) erhalten können.

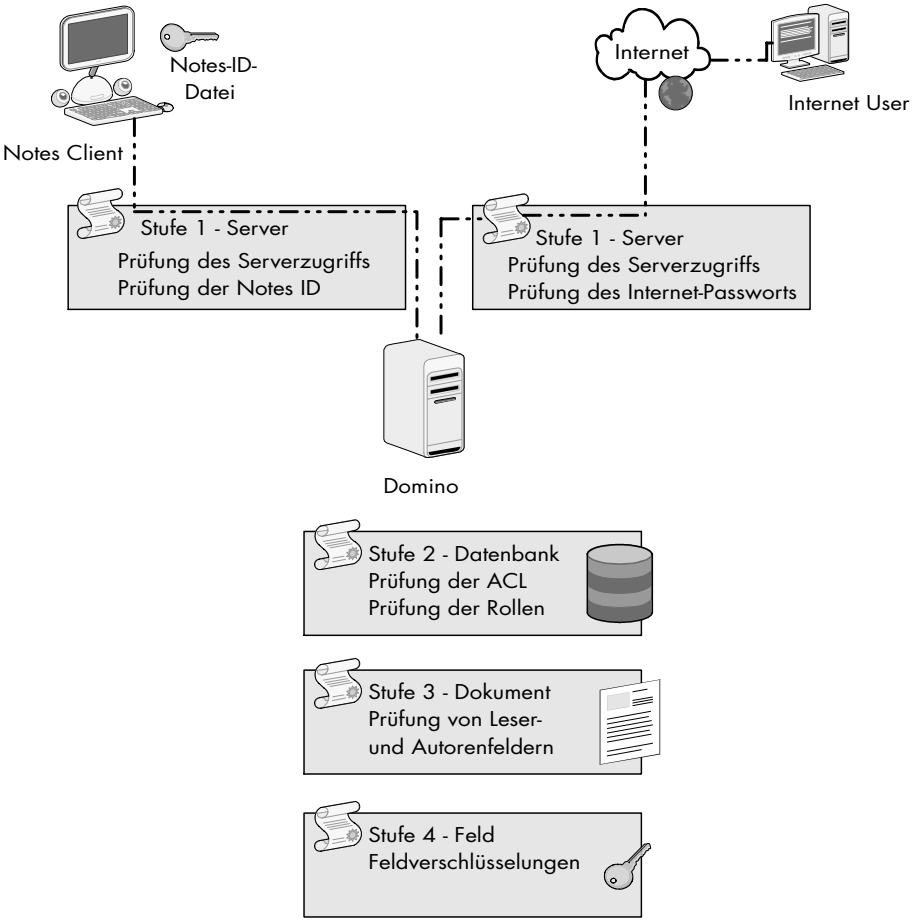

Abb. 1-1 Domino-Sicherheit

Zusätzlich können in der ACL Rollen definiert werden, so dass einzelne Bereiche oder Daten von Anwendungen abhängig von dem Erhalt dieser Rollen gesteuert werden können.

Mit so genannten Leser- und Autorenfeldern können dann auf Dokumentenebene für jedes einzelne Dokument explizit Rechte vergeben werden. Mit Leserfeldern wird festgelegt, wer ein Dokument lesen darf. Mit Autorenfeldern wird festgelegt, wer ein Dokument verändern darf.

Zusätzlich kann dann noch auf Feldebene eine Verschlüsselung erfolgen, so dass derart verschlüsselte Felder nur von Inhabern dieses Schlüssels gelesen werden können. Da diese Schlüssel in der ID-Datei gespeichert werden, kann diese Art des Schutzes nur beim Zugriff über den Notes Client, nicht jedoch bei der Benutzung über Browser erfolgen (Näheres hierzu auch in Kapitel 7.5.1ff.)

Im Einzelnen ist die Sicherheitsstruktur wie folgt angelegt:

1 – Serversicherheit – Überprüfung der Notes ID, bzw. des Internet-Benutzernamens und Passwortes.
2 – Datenbanksicherheit – Überprüfung der ACL.
3 – Dokumentensicherheit – Leser- und Autorenfelder.
4 – Sicherheit auf Feldebene – verschlüsselte Items (s. Abb. 1-1).

Insbesondere das Konzept der Leser- und Autorenfelder und der Feldverschlüsselung wird erst durch das für Lotus Domino einzigartige Konzept der offenen Schemastruktur ermöglicht. Jeder Datensatz wird in einem Dokument zusammengefasst. Dieses Dokument kann Felder (Items) enthalten, die von dem einen Benutzer gelesen werden können, von einem anderen nicht.

Ob das Dokument diese Felder tatsächlich enthält ist nicht relevant, da es durch das Schema nicht erzwungen wird. Dadurch hat ein Benutzer, der einen Schlüssel nicht zur Verfügung hat, dennoch die Möglichkeit ein entsprechend reduziertes Dokument mit den ihm zur Verfügung stehenden Feldern zu erzeugen.

Die Java-Programmierung greift, wie bereits erwähnt, nur auf die Backend-Objekte von Domino zu. Daher ist an dieser Stelle erwähnenswert, dass auch die Java-Zugriffe, egal ob diese über DIIOP[1] (s. Kap. 5.3), über einen Web-Browser, oder über einen Lotus-Domino-Agenten erfolgen, Zugriff auf Schlüssel haben können. Diese müssen in diesen Fällen in der so genannten Server-ID gespeichert sein. Die Server-ID ist der RSA-Schlüssel mit den Private Keys, unter denen die Server-Prozesse laufen.

1.5 Geschichte

Die ursprüngliche Idee von Lotus Notes reicht zurück ins Jahr 1973, als am Computerbased Education Research Laboratory (CERL) an der Universität Illinois PLATO Notes auf der Basis von Host-Systemen entwickelt wurde. Diese Software diente dazu, Bug-Reports an Projekt-Administratoren zu senden, die diese Bug-Reports wiederum mit Antworten versehen konnten.

Dieses frühe Diskussionsinstrument sah bereits vor, Dokumente mit den User IDs der Benutzer zu versehen, um nachträgliches Löschen oder Verändern von Inhalten auszuschließen. Unter anderem ist hier auch schon ein für Notes sehr typisches Szenario sichtbar, nämlich dass Dokumente in Dokument- und Antwortdokument-Hierarchien dargestellt werden.

Drei Jahre später wurde PLATO zu PLATO Group Notes erweitert und konnte in dieser Version bereits zwischen privaten und öffentlichen Inhalten unterscheiden, Zugriffskontrolllisten verwalten und Links zwischen Dokumenten und anderen PLATO-Systemen herstellen. Auf Basis von PLATO Group Notes entwickelte ein Team um

[1] DIIOP = Domino IIOP = Internet Inter-ORB Protocol; ORB = Object Request Broker, als Teil der CORBA-Architektur. CORBA = Common Object Request Broker. CORBA wird von der Object Management Group OMG spezifiziert. [CORBA], [OMG]

Ray Ozzie, Tim Halvorsen und Len Kawell gegen Ende des Jahres 1984 mit der Unterstützung von Lotus die Basis von Lotus Notes, das aufgrund der Entwicklung des PCs nun auf DOS oder OS/2 laufen sollte, unter dem Dach von Iris Associates Inc.

Für die Ur-Version von Notes waren die Funktionen E-Mail, Diskussionen, Telefonbücher und Dokumenten-Datenbanken geplant. Aufgrund der damals langsamen Netzwerke und der noch nicht weit entwickelten PC-Systeme wurde es zu einem der großen Ziele von Notes, diese Hürden zu überwinden.

Auf der Basis von zentralen PC-basierten Gruppen-Servern, die bereits mit anderen solchen Servern Daten replizieren konnten, entstand so die erste wichtige Groupware-Software.

Ein weiterer wesentlicher Punkt war zu dieser Zeit die Entscheidung, kein Out-Of-The-Box-Produkt zu erstellen, sondern eine Plattform zu bieten, die es den Benutzern ermöglicht die Anwendungen an ihre Bedürfnisse anzupassen. Diese Entscheidung spiegelt sich auch in der heutigen Version wider. Vor-Versionen von Lotus Notes wurden bereits 1986 eingesetzt, wobei die Rechte daran 1987 von Lotus gekauft wurden.

Die Version 1.0 von Lotus Notes wurde dann 1989 ausgeliefert und hatte zu dieser Zeit wesentliche Funktionen des heutigen Lotus Notes implementiert. Hierzu gehören Encryption und Signaturen, Dial-Up und Import/Export-Funktionalität, Zugriffskontrolllisten, Online Help, die @-Formelsprache und Funktionen für die Remoteadministration.

Die Versionen 1.1 und 2.0 und 3.0 brachten zum einen neue Fähigkeiten für mittlere und große Unternehmen, so dass auch 10.000 und mehr Benutzer unterstützt wurden, zum anderen weitere Funktionen, wie zum Beispiel die C API, die Unterstützung von RichText und Funktionen für die Volltextsuche, wobei insbesondere mit der Version 3.0 die Cross-Plattform-Fähigkeit weiter ausgebaut wurde.

1994 kaufte Lotus Iris Associates und 1995 kaufte IBM Lotus, insbesondere wegen Lotus Notes.

Die Versionen 4.0 und 4.5 spiegeln die Entwicklung dieser Zeit wider, so dass etliche der Internet-Protokolle Einzug in die Notes-Technologie fanden. Neben Socks und HTTP-Proxy wurden SMTP/MIME, POP3 und HTTP unterstützt, wobei Lotus Notes mit einem eigenen HTTP-Server aufwartete. Der Notes-Server wurde mit Version 4.5 in Domino umbenannt, wobei der Client weiter Lotus Notes Client hieß. Version 4.0 erhielt mit LotusScript, basierend auf Visual Basic, eine erste Programmiersprache, mit der komplette Anwendungsentwicklung betrieben werden konnte. Bereits in Version 4.5 wurde Java unterstützt, so dass Agenten komplett in Java programmiert werden konnten.

Lotus Domino konnte nun die meisten der Notes-Funktionalitäten und Anwendungen auch über den Web-Browser darstellen. Hierzu gehört auch die Fähigkeit des Notes Client, Internetdienste abzurufen und umgekehrt Notes-Dienste wie E-Mail über den Browser abzurufen.

Mit Version 5.0 erhielt Lotus Domino 1999 ein komplettes Face-Lifting. Die etwas aus der Mode gekommene Oberfläche wurde vollständig überarbeitet.

Technisch wurde die Integration von Internet-Funktionalitäten vorangetrieben. Standard-Protokolle wurden integriert, wie LDAP, JavaScript und CORBA, bereits vorhandene Technologien, wie MIME und SMTP wurden überarbeitet und insbeson-

dere streng an den Industriestandards ausgerichtet. Gleichzeitig wurden die Entwickler- und Administrationswerkzeuge (Notes Designer, Notes Administrator) verbessert. Alle wichtigen Internet-Protokolle und -Standards werden mit der Version 5.0 unterstützt.

Mit einem eigenen Servlet Manager geht Domino seine ersten Schritte in Richtung Web Application Server. Zusätzlich bietet diese Version Plugins und Schnittstellen zum Apache Webserver, Tomcat Application Server und dem IBM-eigenen WebSphere Application Server, die so die Brücke zur J2EE-Welt schlagen.

Java hat in alle Teile des Domino-Servers Einzug erhalten, wird aber nur in der Version 1.1.8 unterstützt, so dass hier einer der wichtigen Kritikpunkte an dieser Version ansetzt.

Der Lotus Notes Client in Version 6.0 und 6.5 wurde erneut optisch überarbeitet und ähnelt nun immer stärker Portal-orientierten Collaboration-Anwendungen.

Viele der langjährigen Basis-Funktionalitäten wie Kalender, Aufgaben und E-Mail erhalten in diesen Versionen eine Überarbeitung, um sie an die modernen Anforderungen einer Groupware-Software anzupassen. So wurden Spam-Filter und Messaging-Funktionen eingebaut bzw. verbessert. Gleichzeitig wurde der Web-Zugriff auf die Notes-Funktionalitäten, die ihren Niederschlag im iNotes Web Access Client fanden, erheblich verbessert, insbesondere die Offlineservices sind eine Besonderheit in diesem Bereich und ermöglichen es Anwendern über den Browser zu arbeiten, ohne dauerhaft online sein zu müssen.

Der Server wurde in vielen Bereichen überarbeitet und verbessert. In vielen Bereichen konnten erhebliche Geschwindigkeitssteigerungen erreicht werden. Native Internet-Protokolle sind nun zur Basis vieler Prozesse im Domino-Server geworden, hierzu gehört z.B. nun die nahtlose Integration einer R.509-Zertifikat- und S/MIME-Infrastruktur, die Möglichkeit, LDAP als Protokoll für Namens-Anfragen zu verwenden und die Implementierung vieler XML-Funktionen in die Domino-Core-Klassen von Java und LotusScript.

Java wartet nun mit J2SE 1.3.1 auf. Um die fehlenden J2EE-Funktionalitäten auszugleichen hat sich IBM für ein neues Lizenzierungsmodell entschieden, das es erlaubt mit einer Domino-Lizenz einen WebSphere Application Server neben einem Domino-Server zu betreiben, sofern beide auf der gleichen Maschine betrieben werden und WebSphere nur auf Domino-Objekte zugreift. Diese beiden Server können nahtlos miteinander integriert werden. Single Sign On gehört ebenso zu den Standards wie ausführliche Custom-Tag-Bibliotheken (*Domtags Tag Library*) für den Einsatz von JSP.

Die Entwickler-Umgebung von WebSphere, der WebSphere Application Developer (WSAD), wird mit dem Lotus Domino Toolkit for WebSphere Studio ausgeliefert, einem Plugin, das die Erstellung von JSP auf Basis der *Domtags Tag Library* erleichtert. Dieses Toolkit kann auch mit Eclipse, einem der inzwischen beliebtesten Java IDEs, das als Open-Source-Projekt von IBM gefördert wird, betrieben werden.

Durch diese Strategie wird eine Entwicklung deutlich, die einen erheblichen Einfluss auf den Einsatz von Lotus Domino hat: Es bilden sich verschiedene Layer heraus, die von den verschiedenen Servern bedient werden. Lotus Domino konzentriert sich wieder stärker auf seine ureigenen Fähigkeiten, der Collaboration und Dokumenten-Management, das sich zum Beispiel im Redaktion-Layer eines Content Ma-

nagements ausprägen kann. Backend-Batch-Verarbeitung, aber auch typische Web-Applikationen, also z.B. der Display Layer eines typischen Web-Content-Management-Systems und MVC-Pattern-basierte Anwendungen (Modell-View-Controller, s. auch Kap. 17.6), die eine robuste J2EE-Application-Server-Umgebung benötigen, finden im WebSphere Application Server eine Plattform.

Die Fähigkeiten von Domino, Java und Servlets zu unterstützen, erleichtern diese Entwicklung erheblich, können doch Anwendungen zunächst ausschließlich auf Domino betrieben und später für einen Application Server erweitert werden.

Version 7, die Ende 2005 ausgeliefert wurde, stellt einen weiteren Meilenstein für Domino dar. Diese Version bricht alte Datenstrukturen auf und ermöglicht die Speicherung der Domino-Daten im RDBMS/SQL-Umfeld. Ab Version 7 wird es möglich sein, Domino-Daten direkt in DB2-Datenbanken abzulegen und diese aber im gewohnten Domino-Umfeld, also in Masken und Views darzustellen.

Java-Entwicklern steht ab Version 7 das JDK 1.4.1 zur Verfügung.

Linux findet seit Domino Version 5 Einzug in die Multiplattform-Fähigkeit von Domino und ist mit Version 7 zum Standard herangereift. Die Performance unter Linux soll in dieser Version erheblich verbessert werden.

Ein weiterer Schwerpunkt sind viele neue Monitoring-Funktionen für das Server Management und eine neue Multi-Domain-Fähigkeit, so dass mehrere Notes Domains auf einem Server betrieben werden können.

Die Entwicklung der Internet-Standards findet erneut Einzug in die Domino-Entwickler-Tools. Unter anderem können nun WebServices direkt im Domino Designer (dem Domino IDE) entwickelt werden.

1.6 Lotus Domino und Java

Java fand bereits Ende 1996 mit Version 4.5 Einzug in Domino. Es gehört zu den Grundsatzentscheidungen von IBM, international anerkannte und verbreitete Standards für Protokolle, Frameworks, Konzepte, Plattformen und Sprachen zu unterstützen. Hierzu gehört z.B. die Entscheidung, Linux als Plattform in die Entwicklung aufzunehmen und konsequent zu unterstützen. Java ist ein zentraler Punkt in dieser Strategie. Java ist die Core-Programmiersprache im Application-Server-Bereich.

Durch diese frühzeitige Unterstützung ist das Java-Framework zu einem robusten Entwickler-Werkzeug für Domino erwachsen, obwohl die Gemeinde der Domino-Entwickler erst in letzter Zeit nach und nach von LotusScript auf Java umschwenkt, oder zumindest ihr Wissen in dieser Richtung erweitert. Wie bereits zuvor aufgezeigt hat der rasant wachsende Application-Server-Markt einen großen Einfluss auf die Weiterentwicklung von Lotus Domino, so dass Java eine zentrale Bedeutung zukommt.

Im Objektmodell von Lotus Domino wird unterschieden nach den so genannten UI-Klassen und den Backend-Klassen. Die UI-Klassen bilden alle Objekte ab, die in direkter Interaktion mit dem Benutzer stehen. Öffnet z.B. ein Benutzer ein Dokument, so werden die Daten dieses Dokuments mit einer so genannten Maske dargestellt.