



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 andererseits 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 andererseits 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.

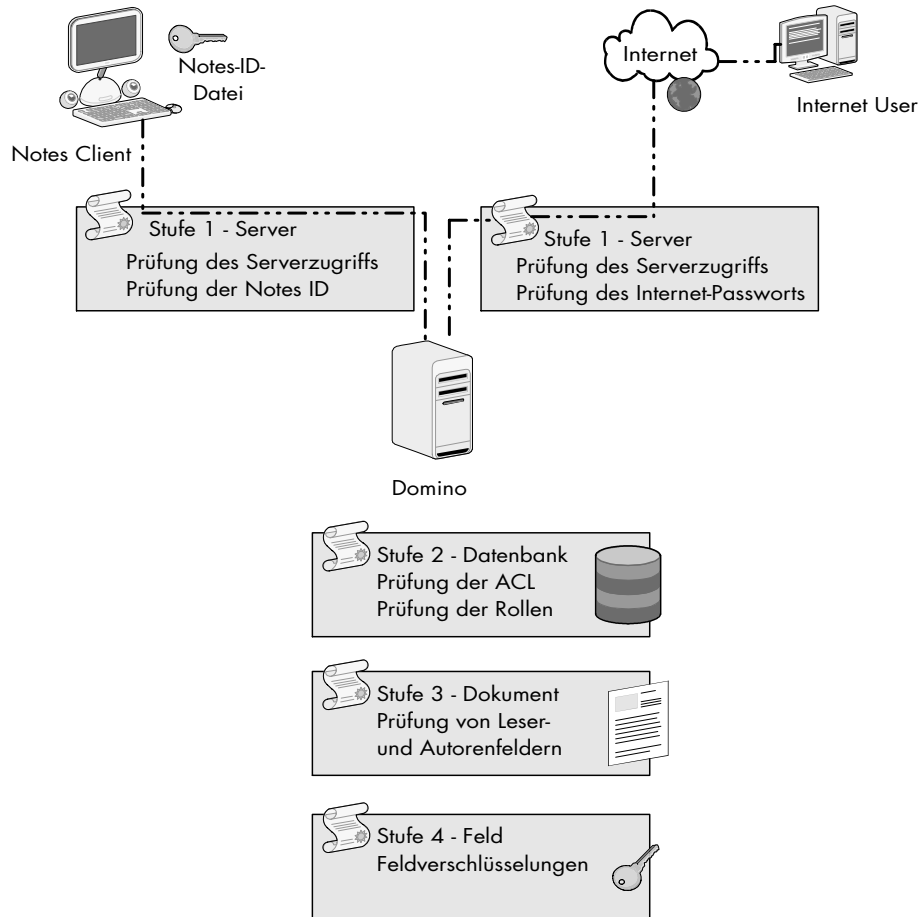
Das Konzept dieses Frameworks ist ausgereift und sehr granulär steuerbar. In einer Abstufung der Rechte, vom Server bis hin zu einzelnen Feldern in Dokumenten, kann der Programmierer genau festlegen, welche Daten eingesehen werden können.

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 DIIOP¹ (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.

Die UI-Klassen bilden die meisten Funktionen und Objekte ab, die im Lotus Notes Client zur Verfügung stehen. Das derart geöffnete Dokument wird in den UI-Klassen abgebildet. Mit den Methoden dieser Klassen können dann z.B. Cursor bewegt oder Fenster geschlossen werden.

Gleichzeitig hat ein Dokument eine Repräsentation in einer Backendklasse, in Java ist dies die Klasse `Document`. Eine solche Instanz eines Dokuments kann gleichzeitig mit der UI-Instanz existieren. Die Instanz des Backend-Objekts hat jedoch keinen Zugriff auf Objekte oder Funktionen, mit denen der Benutzer interagiert, sondern lediglich auf den Datensatz als solchen. Bei der Klasse `Document` sind dies alle Items (Felder) eines Dokuments und Methoden zur Manipulation, die unabhängig vom Benutzer stattfinden können, wie z.B. das Verschlüsseln (`document.encrypt()`) oder Speichern (`document.save()`) von Daten, oder das Versenden von Daten als E-Mail (`document.send()`).

Java hat, im Gegensatz zu LotusScript, ausschließlich Zugriff auf die Backend Repräsentation eines Notes-Objektes.

Dies prädestiniert Java für folgende Anwendungsbereiche:

- Backend- und Batch-Verarbeitung von Domino-Daten
- Web-Anwendungen mit Zugriff auf Domino-Daten, auch über Remotezugriffe und Application Server wie WebSphere
- Enterprise-Datenbankanbindungen für Domino-Applikationen

Die fehlende Möglichkeit, Frontend-Objekte durch Java zu steuern, bedeutet nicht, dass Java nicht für Frontend-Anwendungen (Lotus Notes Client-Anwendungen) eingesetzt oder aus diesen angesteuert werden könnte. Hierzu mehr in Kapitel 4.

Ohnehin haben sich vorwiegend die Backend-Klassen in den letzten Versionen weiterentwickelt. In ihnen liegt die eigentliche Macht des Domino API. Oft wird LotusScript mit Java verglichen und versucht herauszukristallisieren, wo die Vor- und Nachteile der beiden Programmiersprachen im Zusammenhang mit Lotus Domino liegen.

Java ist die modernere der beiden Programmiersprachen und blickt auf ein weites Umfeld an Open-Source-Projekten und -Bibliotheken, die es dem Programmierer ermöglichen, auf vorgefertigte (Teil-)Ergebnisse zurückzugreifen und so die Entwicklungszyklen erheblich zu beschleunigen. Auch wenn LotusScript im Ansatz auch Objektorientierung und Klassen-Strukturen bietet, fällt bei der Bevorzugung einer objektorientierten Sprache eindeutig die Wahl auf Java.

Interessant ist, versucht man beide Sprachen zu vergleichen, dass beide auf die gleichen C++ Objekte im Hintergrund zugreifen, denn die Domino-Java-Objekte sind im Wesentlichen nicht als originärer Java-Code implementiert, sondern basieren auf dem Java Native Interface JNI. Folglich ist sichergestellt, dass äquivalente Methoden absolut äquivalent arbeiten.

Durch die besseren Möglichkeiten in Java, das Speichermanagement zu beeinflussen, lässt sich, wie später gezeigt werden wird, mit Java eine nicht unerhebliche Performance-Steigerung erreichen.

Durch die bereits in der Core-Sprache verankerten IOStream-Klassen kann Java erheblich besser mit Daten- oder Character-Strömen umgehen, wobei erwähnt werden soll, dass IBM mit der Version 6.0 auch Stream-Klassen für LotusScript ausliefert.

Sollen Multithreading-Anwendungen programmiert oder URL-Verbindungen realisiert werden, muss auf Java zurückgegriffen werden, da diese Möglichkeiten nicht für LotusScript zur Verfügung stehen. Neben den nativen XML-Methoden, die das Domino API für Java (und LotusScript) zur Verfügung stellt, bietet es sich an, bei Projekten, die XML verarbeiten, auch die XML-Klassen des Apache Projektes (Xerxes, Xalan, FOP und andere) anzusehen [Apache, XML].

Durch die recht alte JVM 1.1.8, die mit Domino R5 ausgeliefert wird, empfiehlt es sich Java-Anwendungen für Domino R6 zu entwickeln, wo dies möglich ist, da diese Version mit JDK 1.3.1 ausgeliefert wird. Sollen JSP- oder Servlet-2.0-Funktionalitäten zum Einsatz kommen, können die Plugins für Domino oder WebSphere (oder auch Tomcat und sogar JBoss – hierzu mehr in Kapitel 16.1ff.) verwendet werden. Mit diesen modernen J2EE-Containern lassen sich Domino-Daten optimal verarbeiten, z.B. auch über eine Remoteverarbeitung über IIOP.

Müssen Domino-Daten regelmäßig periodisch oder eventgesteuert verarbeitet werden, bietet es sich an, diese als Domino-Agenten (so heißen die Batch-Prozesse unter Domino) umzusetzen.

Durch diese vielfältigen Schnittstellen ist Java zur der wichtigsten Programmiersprache herangewachsen, mit der Domino-Prozesse und -Daten verarbeitet werden.

1.7 Lotus Domino als Datenbank

Der Lotus Domino Server ist in seinem Selbstverständnis weit mehr als eine Datenbank. Lotus Domino ist vielmehr eine Entwickler-Plattform für Groupware-Anwendungen in verteilten Umgebungen.

Daher stellt Domino zusätzlich zu den Datenbanken viele Dienste zur Verfügung, die für die Programmierung von Unternehmensanwendungen herangezogen werden können. Einer der wichtigsten Dienste in der Domino-Infrastruktur ist das Namens- und Adressbuch. Hier werden alle Benutzer- und Berechtigungsinformationen, aber auch alle Konfigurationen für den Server gespeichert. Ganz im Sinne der Erweiterbarkeit steht seit Version 5 dieses Adressbuch als LDAP-Dienst zur Verfügung. Weiterhin stehen folgende Basisdienste zur Verfügung:

- POP3, IMAP, SMTP, jeweils auch über einen SSL-verschlüsselten Kanal
- Domino Router
- LDAP, auch über SSL
- News
- IIOP, auch über SSL
- HTTP und HTTPS
- Indexer (erstellt Volltext-Indizes über einzelne Datenbanken)
- Domänen übergreifender Volltext-Index über Datenbanken und File-Systeme

- Agent Manager (zur periodischen und eventgesteuerten Abwicklung von Prozessen – auch Java-Anwendungen können hierdurch gesteuert werden).
- diverse administrative Domino-spezifische Dienste, wie z.B. Replikation, Schedule Manager, Statistik und Monitoring

Alle diese Dienste basieren auf den RFC-Standards und können über die üblichen Java-Klassen angesprochen werden. Domino basiert auf einem Namens- und Adressbuch, dem Domino Directory oder auch „names.nsf“, das auch als LDAP-Verzeichnis betrieben werden kann und so nicht nur in der Java-Programmierung vielfältig genutzt werden kann, sondern auch Drittanwendungen zur Verfügung steht.

Die Domino-Datenbanken selbst basieren (vgl. aber auch die Verwendung von Domino-Datenbanken in DB2, s. Kap. 13.4), wie bereits erwähnt, auf einer vereinfacht gesprochen losen Sammlung von Datensätzen oder auch „Notes“.

Organisiert werden die Daten durch den Aufbau von Indizes über so genannte Ansichten (Views). Ansichten werden in der Regel im IDE, dem Domino Designer, angelegt. Die in den Ansichten selektierten Dokumente werden über Select-Formeln ausgewählt, die den SELECT-Formeln des SQL recht ähnlich sind. Ansichten können auch über das Backend aus Java heraus generiert werden, sollten aber jeweils nur einmal angelegt werden.

Wichtig für die spätere Selektion von Dokumenten ist, dass mindestens die erste Spalte einer Ansicht sortiert ist. Diese Eigenschaft kann entweder über den Domino Designer oder programmatisch erstellt werden. Über sortierte Spalten legt Domino automatisch Indizes an.

Daten werden entweder über das UI Frontend von Lotus Domino in Masken (Form) erfasst und in Dokumenten gespeichert. Dokumente besitzen neben einigen Standard-Feldern, für Berechtigungen, Revisionen und Zeitstempel, eine beliebige (nur durch einige Speicherlimits begrenzte) Anzahl von Items, in denen die Daten gespeichert werden.

Der große Vorteil der Dokumente ist, dass weder die Größe einzelner Felder, noch die Anzahl der Felder in einem Dokument durch ein Schema erzwungen wird. Daher sind Dokumente nur in einer starken Vereinfachung mit einer Zeile einer SQL-Tabelle zu vergleichen.

Bei der Backend-Programmierung in Java werden die Dokumente als Instanz der Klasse `Document` abgebildet. Felder werden über spezielle *getter*- und *setter*-Methoden hinzugefügt, verändert oder gelöscht.

Beispiel für eine Select-Formel einer Ansicht:

`SELECT`

`((@LowerCase(Form) = "logfile") |`

`(@LowerCase(Form) = "fo_logfile"))`

Über sortierte Spalten legt Domino automatisch Indizes an.

Es gibt verschiedene Möglichkeiten, Dokumente in Domino-Datenbanken zu selektieren. Hierzu mehr in Kapitel 7.1.4ff. Der wichtigste und effizienteste Weg erfolgt über Ansichten. Diese Ansichten können mit verschiedenen Methoden durchsucht werden. `getAllDocumentsByKey ()` durchsucht z.B. die erste(n) Spalte(n) einer Ansicht anhand eines Schlüsselwortes.

Eine Java-basierte Domino-Datenbank, oder besser gesagt Domino-Anwendung besteht in der Regel aus einigen Masken und Ansichten (und vielen anderen Gestaltungselementen, s. Kap. 3 und 6), die das graphische Interface für den Benutzer darstellen und Java-Programmen, die im Hintergrund Funktionen ausführen, bzw. Funktionen, die über Aktionsbuttons durch den Anwender ausgelöst werden.

Alternativ kann Domino natürlich auch aus stand-alone Java-Programmen oder auch mit Servlets und JSP angesprochen werden. Nicht zuletzt können alle Domino-Anwendungen oft nahtlos, ohne zusätzliche Programmierung, über einen Browser angezeigt werden. Hier kommen dann, neben den nativen Rendering-Möglichkeiten von Domino, die es erlauben, Notes-Masken „on the fly“ im Browser anzuzeigen, die J2EE-Techniken zum Einsatz. Insbesondere das Lotus Domino Toolkit for WebSphere Studio ist ein sehr hilfreiches Plugin für den Einsatz der *Domtags Tag Library*, mit der Domino-Objekte durch die Verwendung von JSP dargestellt werden können.

1.8 Die Client-Server-Umgebung Lotus und Domino

Lotus Domino war schon immer mit Remotefähigkeiten ausgestattet, eines der wichtigen Features von Domino war für viele Jahre die Remoteeinwahl in einen Domino Server über Telefon.

Anwender konnten Domino-Applikationen lokal auf ihren Rechnern betreiben und bedienen. Sollten Daten zwischen dem Client-System und dem Server aktualisiert werden, wählte der Benutzer sich ein und über die Replikation erfolgte ein Datenabgleich. Der gesamte Datenabgleichsvorgang ist für den Benutzer transparent.

Inzwischen sind die Leitungskapazitäten im LAN und WAN erheblich größer, so dass sich die Infrastruktur von Domino ebenfalls verändert hat. Immer mehr Remoteanwender wählen über WAN direkt die Notes-Anwendungen an oder verbinden sich mit ihrem Home-Server über so genannte Durchgangsserver, die zur Verbindung von Netzwerken aus Sicherheitsgründen zwischengeschaltet werden.

Die Replikation ist weiterhin ein wichtiges Instrument, um in verteilten Umgebungen die Anwendungsdaten verschiedener Server aktuell zu halten und um verschiedene Netzwerke und Sicherheitszonen miteinander abgleichen zu können.

So genannte Replikationsregeln ermöglichen es, in vorgelagerten Servern, die dem Internet „ausgesetzt“ sind, Daten entgegenzunehmen und „nach innen“ zu replizieren, so dass auf Servern, in weniger sicheren Umgebungen, Daten immer nur kurz vorliegen und dann per Replikation in das sichere Intranet transportiert werden können.

Die Netzwerkfähigkeit von Domino in seinen verschiedenen Ausprägungen hat einen großen Einfluss darauf, wie in der Java-Programmierung auf Domino zugegriffen wird.

- Zugriff lokal auf Domino-Datenbanken des Clients (Kapitel 4.3)
- Zugriff lokal auf einen Domino-Server (Kapitel 4.3)
- Zugriff per DIIOP (Domino IIOp) auf einen entfernten Server (Kapitel 5.3.2)
- Zugriff auf eine entfernte Domino-Datenbank auf einem als vertrauenswürdig eingestuften Server über die aktuelle Notes Session (seit Domino Version 6) (Kapitel 5.3)

Um eine Verbindung zu Domino-Datenbanken, egal ob lokal oder remote, herzustellen, bedarf es immer einer so genannten Notes Session. Eine Notes Session wird bei der Java-Programmierung innerhalb einer Domino-Datenbank, also bei der Erstellung eines Java-Agenten, über `AgentBase.getSession()` zur Verfügung gestellt.

Für alle anderen Arten der Verbindungsherstellung muss über die Klassen `NotesThread` oder `NotesFactory` eine Session aufgebaut werden. Diese verschiedenen Zugriffe können auf unterschiedlichen Wegen ausgelöst werden:

- Stand-alone Java-Programm (Kapitel 4.4)
- Domino-Agent (Kapitel 4.10)
- Servlet im Domino Servlet Manager (läuft im HTTP-Task des Domino-Servers) (Kapitel 5.2.1)
- Servlet oder JSP in einem Servlet Manager außerhalb von Domino (z.B. WebSphere Application Server) (Kapitel 16 und 17)

Diese unterschiedlichen Methoden erfordern verschiedene Authentifizierungen.

Ein Servlet, das z.B. im Domino Servlet Manager aufgerufen wird, erhält bei der Initialisierung einer Notes Session per Default die Rechte des Domino-Servers, genauer gesagt der so genannten Server-ID, unter der der Domino-Server läuft. Standardmäßig werden dann die Aktionen dieser Notes Session als lokale Zugriffe behandelt, d.h. aus Sicht des Servlets ist der Server lokal. Alternativ wird die Notes Session des Servlets mit den Rechten des über den Browser angemeldeten Benutzers aufgebaut. Hierbei übernimmt Domino die Authentifizierung gegen sein Adressbuch oder ein entsprechend nachgeordnetes LDAP-Verzeichnis. Wird ein Application Server auf der gleichen Hardware wie der Domino-Server betrieben, dann sind ebenfalls lokale Zugriffe möglich.

Alle anderen Fälle erfolgen dann remote. Die Authentifizierung kann im so genannten SSO (Single Sign On) erfolgen. Alternativ kann ein spezieller Domino-Benutzer definiert werden, unter dem dann authentifiziert wird. Hieraus ergeben sich grundsätzlich zwei Konzepte für die Authentifizierung:

- A Authentifizierung über einen dedizierten Benutzer oder über die Server-ID. Alle Domino-Daten sind für diesen einen Benutzer/Server zugänglich. Die Unterscheidung der Rechte für die einzelnen Endanwender erfolgt innerhalb der Anwendung.
 - Vorteil: Kein administrativer Aufwand für die Benutzerverwaltung.
 - Nachteil: Die Möglichkeiten der granulären Rechtevergabe in Domino werden nicht ausgeschöpft.

- B Authentifizierung über registrierte Domino-Benutzer. Alle Lese- oder Schreibzugriffe auf Daten werden mit den Rechten des angemeldeten und im Domino-Adressbuch registrierten Benutzers abgeglichen.
- Vorteil: Jeder authentifizierte Benutzer kann nur die Daten sehen / ändern, die im Domino-Rechte-System für ihn vorgesehen sind. Rollen, Leser- und Autorenfelder.
 - Nachteil: Mehraufwand für Administration der Benutzerverwaltung.

Die Architektur einer Java-Anwendung wird letztendlich festlegen, wie die Verbindung und die Authentifizierung mit dem Domino-Server hergestellt werden muss.

1.9 Zusammenfassung

Domino ist ein robuster Application Server mit einer langjährigen Geschichte. Durch die Vielzahl integrierter Services auf der Basis von RFC-Standards bietet sich dem Java-Entwickler eine Plattform, die auf viele vorgefertigte Tools und Dienste zurückgreifen kann.

Insbesondere für Groupware-Anwendungen in verteilten Umgebungen zeigt Domino seine Stärken. Durch das Konzept der offenen Schemata für die Speicherung von Daten, Objekten, Gestaltungselementen und Zugriffskontrolllisten ist Domino eine Plattform, die durch den Anwender selbst einfach erweitert werden kann. Das wichtigste Grundprinzip der Organisation von Daten in einer Domino-Datenbank ist:

- Speicherung der Daten in Dokumenten
- Aufteilen der Daten in Items
- Erstellung von Ansichten zur Indizierung von Dokumenten
- Selektion von Dokumenten über die Suche in Ansichten

Interessant ist für den Java-Programmierer das Konzept der Dokumente und Items, durch das Daten und Felder in Datensätzen beliebig erweitert werden können. Ansichten sind ein wesentliches Selektionswerkzeug für die Auswahl von Dokumenten.

Java ist die wichtige Programmiersprache für die Anwendungsentwicklung in Lotus Domino und kann auf vielfältige Weise eingesetzt werden, insbesondere Servlets können direkt in den Domino-Server integriert werden oder über externe Application Server wie WebSphere auf Domino-Daten zurückgreifen.

Mit der Version 7 wird Domino mit vielen neuen Möglichkeiten aufwarten. Insbesondere die Integration von DB2-Daten stellt eine interessante Erweiterung dar und lässt die J2EE-Application-Server-Welt und die Domino-Welt näher zusammenwachsen.

Da die Stärken von Domino im Groupware- und Collaboration-Bereich liegen, eröffnet sich durch die Verwendung von Java die Möglichkeit, solche Anwendungen im Web mit einem robusten J2EE Layer auszustatten ohne auf die Stärken von Domino verzichten zu müssen.

Java unter Lotus Domino

Know-how für die Anwendungsentwicklung

Ekert, Th.

2006, XVIII, 803 S. Mit CD-ROM., Hardcover

ISBN: 978-3-540-22176-0