PHP to Notes Connector – Zwei Welten verbinden

Mit über 120 Mio Lizenzen weltweit ist Lotus Notes eines der weitverbreitetsten Groupwaresysteme und eine der führenden Plattformen für Messaging und Collaboration. Vorwiegend im professionellen Umfeld eingesetzt ist Notes darüberhinaus als Datenbanksystem ein wertvoller Speicher für Unternehmens- und Kundeninformationen. Dieser Artikel zeigt Wege Notes als Datenquelle für PHP nutzbar zu machen.

Lotus Notes – ein kurze Historie

Von den Einen innig geliebt, von den Anderen ebenso innig gehasst, reduziert sich Lotus Notes in der Wahrnehmung vieler Außenstehender meistens auf ein überfrachtetes E-Mail-System mit Workflowfunktionalität. Damit tut man Notes sicherlich unrecht, denn über die E-Mail Funktionalität hinaus ist Notes ein dokumentenorientiertes, verteiltes Datenbanksystem (siehe Kasten 1). Seit den ersten Anfängen 1984 hat sich Notes, bis 1995 unter der Führung von Lotus und ihrer Entwicklerschmiede Iris, zu einer sehr leistungsfähigen Entwicklungsumgebung webfähiger Groupwareanwendungen entwickelt. Seit 2000 ist Lotus neben DB2, Rational, Tivoli und Websphere als eine von fünf Brands unter dem Dach der IBM integriert und bildet eine wichtige Komponente im IBM Software-Portfolio. Die konsequente Weiterentwicklung und die Integration neuer Technologien wie z.B. Webservices hat Notes/Domino zu einer festen und zukunftssicheren Größe im Bereich kollaborativer Unternehmenssoftware gemacht.

Notes „anzapfen“

Um Notes-Daten für PHP nutzbar zu machen bieten sich mehrere Möglichkeiten an. Zunächst stellt sich die Frage, ob man einen „stateful“ oder „stateless“ Ansatz wählen soll – beide Ansätze haben sowohl Vor- als auch Nachteile. Eine Anbindung über DIIOP, der Notes-Implementierung von CORBA, bietet Vorteile für transaktionsorientierte Anwendungen, da Notes-Objekte „stateful“ im Speicher gehalten werden können. Hat man z.B. eine Anwendung mit sehr vielen lesenden und schreibenden Zugriffen auf kleine Datenmengen ist ein solcher Ansatz zu bevorzugen. Die Realisierung einer CORBA Anbindung in PHP ist allerdings relativ aufwendig und bietet aufgrund der Sessionverwaltung einige Fallstricke, die auf dem angesprochenen Notes Server bei einer hohen Userzahl schnell zu erheblichen Performance-Problemen führen können.
Um einen schnellen und einfachen Zugriff auf Notes-Daten zu realisieren, bietet sich HTTP als Übertragungsprotokoll, wahlweise in der abgesicherten Variante HTTPS, und XML als Datenformat an. Eine solche Implementierung folgt einem service-orientiertem Ansatz und erlaubt eine „stateless“ Kommunikation zwischen PHP und Notes. PHP holt die Daten also am HTTP-Server von Notes ab. Die damit erzeugte Serverlast ist, auch bei vielen parallelen Zugriffen, deutlich einfacher zu handhaben. Alternativ, jedoch etwas aufwendiger, ist auch eine Implementierung als Webservice über SOAP als Protokoll möglich. Welchen Ansatz man letztendlich wählt ist von der konkreten Anwendung abhängig. Zur Umsetzung der dargestellten Lösungen haben wir uns für die Implementierung über HTTP und XML entschieden.

Installation und Zusammenspiel mit Apache

Für die nachfolgenden Betrachtungen gehen wir von einem installierten Domino-Applikationsserver ab Version 6 mit laufendem HTTP Task aus. PHP muß den Domino Server über HTTP bzw. HTTPS vom Apache Server aus erreichen können.
Werden Apache und Notes-Server auf getrennten Maschinen betrieben stellt sich das Setup vergleichsweise einfach dar. Sollen sowohl Notes als auch Apache zu Entwicklungs- oder Testzwecken auf der gleichen Maschine betrieben werden, gestaltet sich die Installation etwas trickreicher. Einer von Beiden, zweckmäßigerweise Notes, muß dann auf einen anderen Port konfiguriert werden.
Daneben können für beide Setup-Szenarien die Apache Module *mod.proxy* und/oder *mod.rewrite* zur Weiterleitung von Anfragen an den Domino Server verwendet werden. Einige beispielhafte Regeln für *mod.rewrite* listet Kasten 2. Eine vollständige Beschreibung der Konfiguration würde den Rahmen dieses Artikels sprengen – detaillierte Installationsanleitungen finden sich im Web unter [1] oder [2].

Zugriff auf Notes Daten

Notes speichert Daten in Form von Dokumenten. Jedes dieser Dokumente kann eine beliebige Anzahl Felder enthalten, in denen die verschiedenen Inhalte gespeichert werden. Mit einem Notes Client kann man über zwei Wege auf die Dokumente zugreifen. In Ansichten (Views) werden Dokumente mit bestimmten Eigenschaften in tabellarischer Form aufgelistet. Die Auswahl der Dokumente und die angezeigten Eigenschaften werden dabei vom Entwickler der Anwendung bestimmt. In kategorisierten Views werden Dokumente mit gleichen Eigenschaften ähnlich wie bei einem Newsthread in einer Baumstruktur dargestellt (siehe Abbildung 1a/1b). Die Inhalte einzelner Dokumente können durch Gestaltungsmasken (Forms) angezeigt und bearbeitet werden. Beide Möglichkeiten stehen auch bei einem HTTP Zugriff zur Verfügung. Besitzt der PHP-Entwickler das notwendige Notes-KnowHow kann er über entsprechende Forms und Views die Datenausgabe so gestalten, wie sie auf der PHP-Seite benötigt wird. Ist das KnowHow nicht vorhanden oder kann die Notes Anwendung aus anderen Gründen nicht in dem erforderlichen Maß angepasst werden, müssen alternative Ansätze in Betracht gezogen werden. Dazu stellt Notes folgendes „Standard-Bordmittel“ bereit.

ReadViewEntries
Der HTTP-Zugriff auf einen Notes-View erfolgt normalerweise über den URL-Syntax *http://dominoserver/notesdatenbank.nsf/ viewname?OpenView*. Mit dem URL-Kommando *http://dominoserver/notesdatenbank.nsf/viewname?ReadViewEntries* erfolgt die Ausgabe des View Inhalte als XML Code. Dieser Ansatz bringt uns schon ein Stück weiter, sofern ein entsprechender Notes-View für den Webzugriff freigegeben ist und im benötigten Format mit den benötigten Feldinhalten vorliegt. Tut er das nicht, was in der Realität häufiger der Fall sein dürfte, ist wiederum Notes-KnowHow oder die Mitwirkung eines Notes-Entwicklers gefragt, da die Notes-Anwendung entsprechend modifiziert werden muß. Außerdem können über Views nur Feldinformationen abgegriffen werden, die ein *Summary-Flag* besitzen, also z.B. keine RichText-Inhalte. Weiterhin ist zu beachten, daß die XML-Ausgabe in einem Notes eigenen XML-Syntax erfolgt, nämlich DXL (Domino XML nach eigenem DTD-Standard [3],[4]), was in der Praxis jedoch keine Probleme bereitet. Probleme bereiten da schon eher die begrenzten Viewlängen, die im Namens- und Adressbuch (names.nsf) über das Notes-Serverdokument bzw. über Website-Dokumente (im Abschnitt „Domino Web Engine“) konfiguriert werden können. Hier ist sowohl die Default-Viewlänge, die durch URL-Parameter überschrieben werden kann (siehe Kasten 3), als auch die maximale Viewlänge, die nicht überschrieben werden kann und eine absolute Begrenzung darstellt, festgelegt. Man bekommt also beim Standard-Zugriff auf einen View nicht notwendigerweise alle Dokumente – überschreitet die Anzahl der Dokumente die festgelegte Maximal-Viewlänge müssen zusätzliche Requests gestellt werden um weitere Dokumente zu erhalten. Die Information über die Anzahl der Dokumente in einem View erhält man über das Attribut „toplevelentries“ im Tag „viewentries“ und kann somit gegebenenfalls „nachfassen“. Kategorisierte Views sind aufgrund ihres hirachischen Aufbaus weniger für die Datenübertragung geeignet. Man erkennt die kategorisierten Spalten im XML Code durch das Attribut *category="true* im Tag „entrydata“ (siehe Abbildung 2a/2b). Der besondere Vorteil von Views sollte bei allen bisher genannten Problemen aber nicht unerwähnt bleiben: alle Daten in Views werden in einem Index gespeichert und stehen damit sehr viel schneller zur Verfügung als die entsprechenden Daten in den Dokumenten.
Ein weiteres nützliche URL-Kommando ist der Befehl *ReadDesign*, der zusätzliche Designinformationen über einen View liefert. Leider gibt es keinen vergleichbar einfachen Weg ein Dokument als XML auszugeben – das ist nur per Java-Funktion *NotesDocument.generateXML()* oder Lotusscript möglich, die Notes-seitig implementiert werden müssen.

Die PHP-Klassen
Mit unserem PHP Script müssen wir nun den passenden HTTP Request zum Domino Server senden und den zurückgelieferten XML Code in geeigneter Form weiterverarbeiten. PHP stellt mit dem ereignisbasierten SAX Parser Expat einfache und dennoch effiziente Funktionen für diese Aufgabe bereit.
Holt man den gesamten View mit allen Dokumenten und schreibt diesen in ein Array, kann das Array, wenn der View entsprechend viele Dokumente enthält, sehr viel Speicher verbrauchen. Außerdem gilt es dann noch das bereits oben beschriebene Problem der begrenzten Viewlängen zu lösen.
Performanter und speichertechnisch unkritischer ist ein Ansatz, der üblicherweise bei Datenbanktreibern verfolgt wird, nämlich eine sequentielle Verarbeitung der einzelnen View-Einträge. Dies reduziert die Speichernutzung erheblich und löst gleichzeitig elegant das Viewlängen-Problem, da man auf diesem Weg die Daten aus allen Dokumenten erhält. Die PHP Anwendung kann dann einfach die View Einträge durchloopen (Listing 1/Abbildung 3). Zu Demozwecken finden Interessierte unter [5] eine einfache PHP-Klasse, die für kleine Datenmengen und Testzwecke durchaus ausreichend ist. Unter [6] steht eine Notes-Demodatenbank zur Verfügung, die anonymen Zugriff erlaubt und für erste Testzwecke eingesetzt werden kann, ohne gleich selbst einen eigenen Notes-Server aufsetzen zu müssen.

Zugriffsrechte in Notes
Generelle Zugriffsrechte auf Notes-Datenbanken werden über Zugriffskontrollisten (ACLs) und innerhalb einer Applikation über Rollen gesteuert. Ein unauthentifizierter Zugriff ist dabei über den User *Default* bzw. den User *Anonymous* möglich. Alternativ bietet sich der Zugriff über einen generischen *PHPUser* an. Um Daten aus Notes-Datenbanken auszulesen ist zumindest Reader-Access erforderlich.
Authentifizierter Zugriff z.B. auf Mailfiles ist grundsätzlich möglich, erfordert aber die zusätzliche Verwaltung der Notes-User in PHP, oder einem zentralen LDAP-Directory. Username und Password müssen dann bei jedem Request übertragen werden, da der Zugriff ja „stateless“ erfolgt. Komfortabler, aber auch aufwendiger sind Single-Sign-On Lösungen, bei denen ein LPTA-Token in einem Cookie gespeichert wird, deren detaillierte Beschreibung indes genügend Stoff für einen eigenen Artikel böte und daher an dieser Stelle nicht erfolgen kann. Alternativ bietet sich hier eine sessionbehaftetete Implementierung über CORBA an (siehe oben).

Grenzen
Notes unterscheidet verschiedene Feld- bzw. Datentypen, wie z.B. Text, Number, Time/Date, Listen (multi value) und RichText. In Notes RichText-Feldern können formatierter Text, Datei-Anhänge oder eingebettete Objekte (Bilder, OLE-Objekte) gespeichert werden. Leider können diese Inhalte nicht einfach in HTML umgewandelt werden, womit sich die Verarbeitung in PHP sehr komplex gestaltet. Einfacher ist jedoch die Verarbeitung von Text-Abstract (Text ohne Formatierungen) und Attachments. Dabei sollten Attachments wenn möglich durch entsprechende *mod.proxy*- oder *mod.rewrite*-Regeln direkt vom Notes-Server geholt werden, um ein aufwendiges Ausparsen in PHP zu vermeiden. Für formatierten Text und eingebettete Objekte sind zusätzliche Tools für Notes-RichText Bearbeitung erforderlich.

Der Connector als generische Lösung
Möchte man die Nachteile der bisher gezeigten Ansätze vermeiden, bietet sich der Einsatz einer generischen Lösung an, die es erlaubt auf Notes-Daten zuzugreifen ohne die Notes-Anwendung selbst modifizieren zu müssen. Zu diesem Zweck steht ein spezieller PHP-to-Notes-Connector zur Verfügung, der die benötigte Funktionalität in einer .nsf-Datenbank bereitstellt. Der Notes-Connector agiert quasi wie ein Proxy-Server und erlaubt den Zugriff auf Dokumente in anderen Notes-Datenbanken, auch wenn die Zieldatenbanken nicht für den Webzugriff geöffnet wurden. Der Zugriff erfolgt dabei direkt auf die Dokumente, spezielle Views werden nicht benötigt. Der in Lotusscript realisierte Connector bietet darüberhinaus nützliche Funktionen zur Dokumentenselektion oder die Begrenzung auf bestimmte Felder, um die Menge der zu übertragenden Daten zu begrenzen (siehe Kasten 4).

Schreibender Zugriff
Neben dem Auslesen von Notes-Dokumenten bzw. bestimmten Feldinhalten erlaubt der Connector auch schreibenden Zugriff. Auf diesem Weg können Notes-Dokumente über PHP bearbeitet und Daten nach Notes zurückgeschrieben werden. Dies erfordert natürlich Editor-Access auf die entsprechenden Notes-Datenbanken.

Integration in Contentmanagment-Systeme
Besonders interessant ist die vorgestellte Lösung zur Integration von Notes-Daten in PHP basierte Content-management-Systeme. Eine Referenzimplementierung für das Open-Source Enterprise CMS eZ publish liegt bereits vor und kann über ez.no bezogen werden. Hier ist die Integration über eine Extension realisiert, die einfach in eZ publish eingebunden werden kann. Der Notes-Zugriff und die Formatierung der Notes-Daten kann sehr komfortabel über Templates gesteuert werden.

Fazit
Notes-Daten lassen sich als XML Code über ReadViewEntries auslesen und in PHP integrieren, allerdings sind dazu in den meisten Fällen umfangreiche Änderungen der Notes-Applikation notwendig, um die Daten in der benötigten Form zu erhalten. Darüberhinaus stellt die erforderliche Öffnung der Anwendung für Webzugriffe in der Praxis nicht selten ein Sicherheitsrisiko dar und zieht einen hohen Aufwand nach sich, um die Notes-Applikation „webfest“ zu machen.
Mit dem vom IBM-Business Partner Visual Solutions entwickelten PHP-to-Notes Connector können Notes-Daten einfach und ohne Notes-KnowHow abgegriffen und in PHP integriert werden, ohne das eine Anpassung der Notes-Datenbanken oder eine Öffnung für Webzugriffe erforderlich ist. Der administrative Aufwand ist Notes-seitig auf ein Minimum beschränkt.
Der Connector ist für die Notes Versionen 6 und 7 lieferbar. Alternativ ist die oben beschriebene Funktionalität als DSAPI-Filter in Form einer .dll als Servertask verfügbar. Für größere Integrationsprojekte stellt Visual Solutions auch entsprechende Supportangebote bereit.

Weitere Informationen unter *http://www.visol.de/php*.

Die Autoren
Stefan Weber, Dipl. Inf.
Andreas Croll, Dipl. Ing.

[1] http://www.lotus.com/ldd/nd6forum.nsf/0/f4833adee01587cb852570480078f52d
[2] http://notestips.com/80256B3A007F2692/1/NAMO5RX3PX
[3] http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf
[4] http://www.lotus.com/ldd/notesua.nsf/0/5130904a82cbab128525703b006440ea
[5] http://www.visol.de/pages/phpconnector.htm
[6] http://www.visol.de/php/ezdemo.nsf/menuflat?ReadViewEntries

Kasten 1

Lotus Notes – Die wichtigsten Merkmale im Überblick

1. Nicht-relationales, dokumentenorientiertes Datenbanksystem mit enger E-Mail-Integration, integrierter Benutzerverwaltung und zahlreichen Services (HTTP, SMTP, POP3, IMAP, LDAP, Webservices) zum verteilten Arbeiten an räumlich getrennten Datenbeständen.
2. Rapid Application Development Platform: Anwendungen können mit geringem Aufwand in mehreren Sprachen (LotusScript, Makrosprache @-Formulas, Java) entwickelt und verteilt werden.
3. Replikation: Notes-Datenbanken können sowohl zwischen Servern als auch zwischen Client und Server repliziert (=bidirektional synchronisiert) werden. Daten und Designelemente werden so über mehrere Instanzen einer verteilten Datenbank automatisch abgeglichen.
4. Offline-Funktionalität: Lokale Repliken von Notes-Anwendungen sind in der Regel auch ohne Server-verbindung in vollem Umfang z.B. auf Notebooks nutzbar.
5. Public-Key-Infrastruktur (PKI): Userzertifikate und die granulare Rechteverwaltung bis auf Feldebene erlauben eine detaillierte Zugriffssteuerung auf sensible Unternehmensdaten.
6. Sicherheit: 128-Bit Verschlüsselung für Datenbanken und Replikation.

Kasten 2

Konfiguration von *mod.proxy* und *mod.rewrite*

Apache und Notes-Server auf der gleichen Maschine, Notes-Server auf Port 81 konfiguriert. Beispeilhafte Einträge für httpd.conf:

LoadModule proxy_module modules/mod_proxy.so
ProxyPass /icons/ http:///localhost:81/icons/
Diese Regel ermöglicht die Anzeige der Notes-Standardicons die in Views verwendet werden.

LoadModule rewrite_module modules/mod_rewrite.so
<IfModule mod_rewrite.c>
RewriteEngine on
RewriteRule ^(.).nsf(.*) http:///localhost:81$1.nsf$2 [P]*
</IfModule>
Diese Regel leitet alle URLs mit .nsf im Pfad zum Notes-Server durch. Damit funktionieren Notes-URLs in PHP. Je nach Anwendung können weitere Regeln erforderlich sein, z.B. zum Zugriff auf Servlets. Eine komplette Aufzählung kann an dieser Stelle nicht erfolgen.

Oft arbeiten auch Notes-Anwendungen mit Server-Regeln (Redirects, Substitutions) um.nsf aus dem URL-Pfad zu eliminieren, z.B.:
/products/gb/.htm -> /produkte.nsf/gb/*!OpenDocument*

In diesem Fall funktioniert die obige Rewrite-Regel natürlich nicht. Eine Lösung für solche Fälle ist es, mit virtuellen Hosts zu arbeiten und der Notes-Applikation einen eigenen Domänennamen zu geben. Dann kann man alle Referenzen auf diesen Domänennamen zum Notes-Server umleiten. Nachteil: URLs müssen immer absolut referenziert werden.

Kasten 3

Domino URL-Commands (Auswahl)

Domino URL-Kommandos werden mit einem Fragezeichen von der URL getrennt, je nach Serverkonfiguration ist auch ein Ausrufezeichen möglich. Parameternamen sind durch ein & getrennt, die Parameter selbst werden mit = angegeben. Genereller Syntax ist: *http:///Host/Database/DominoObject?Action&Arguments*

?OpenDatabase	Öffnet Datenbank in festgelegter Default-Ansicht
?OpenView	Öffnet einen View
?OpenDocument	Öffnet ein Dokument
?ReadViewEntries	Gibt einen View als XML aus
?ReadDesign	Gibt Designinformationen als XML aus
Database/0/Notes-ID	wenn die Notes-ID eines Dokumentes bekannt ist, kann der Zugriff auch ohne View über den Platzhalter /0/ erfolgen

Viewparameter

&Start=n	Erster Eintrag der in einem View gelesen wird
&Count=n	Anzahl der Einträge die in einem View gelesen werden (Default und Maximum über Serverdokument konfigurierbar)
&CollapseView	Liest in einem kategorisierten View nur die Einträge der ersten kategorisierten Spalte (der View ist also eingeklappt)
&ExpandView	Liest in einem kategorisierten View alle Einträge (der View ist also ausgeklappt)
&Collapse=n	Liest einen eingeklappten View beginnend mit der n-ten Zeile
&Expand=n	Liest einen ausgeklappten View beginnend mit der n-ten Zeile
&RestrictToCategory=String	Begrenzt die Viewansicht auf die angegebene Kategorie
&StartKey=String	Liest den View beginnend mit dem ersten, dem String entsprechenden Eintrag
&preformat	Liefert in Verbindung mit ?ReadViewEntries die Daten in der Form, wie sie im View dargestellt werden, z.B. Zahlen mit zwei Nachkommastellen

Kasten 4

Beispiele für Dokumenten- und Feldselektion

Zur Dokumentenselektion können beliebige @Formulas ausgeführt werden, z.B.

@All	Alle Dokumente
Form= "String "	Nur Dokumente die auf diesem Form basieren
Feld= "String "	Nur Dokumente die der Feldauswahl entsprechen
@Modified=@Today	Nur Dokumente die am heutigen Tag geändert wurden
@Left(Feldname;1)= "A "	Dokumente bei denen der Inhalt des definierten Feldes mit "A " beginnt

Parameter zur Feldselektion

docmode=native	Originaldokument mit allen Feldern, (RichText und Attachments in DXL, der XML-Code kann dabei sehr umfangreich werden!)
docmode=custom	Kein RichText, keine Attachments
docmode=abstract	RichText als PlainText
fieldlist=Feldname[n]	Beschränkung auf bestimmte Felder

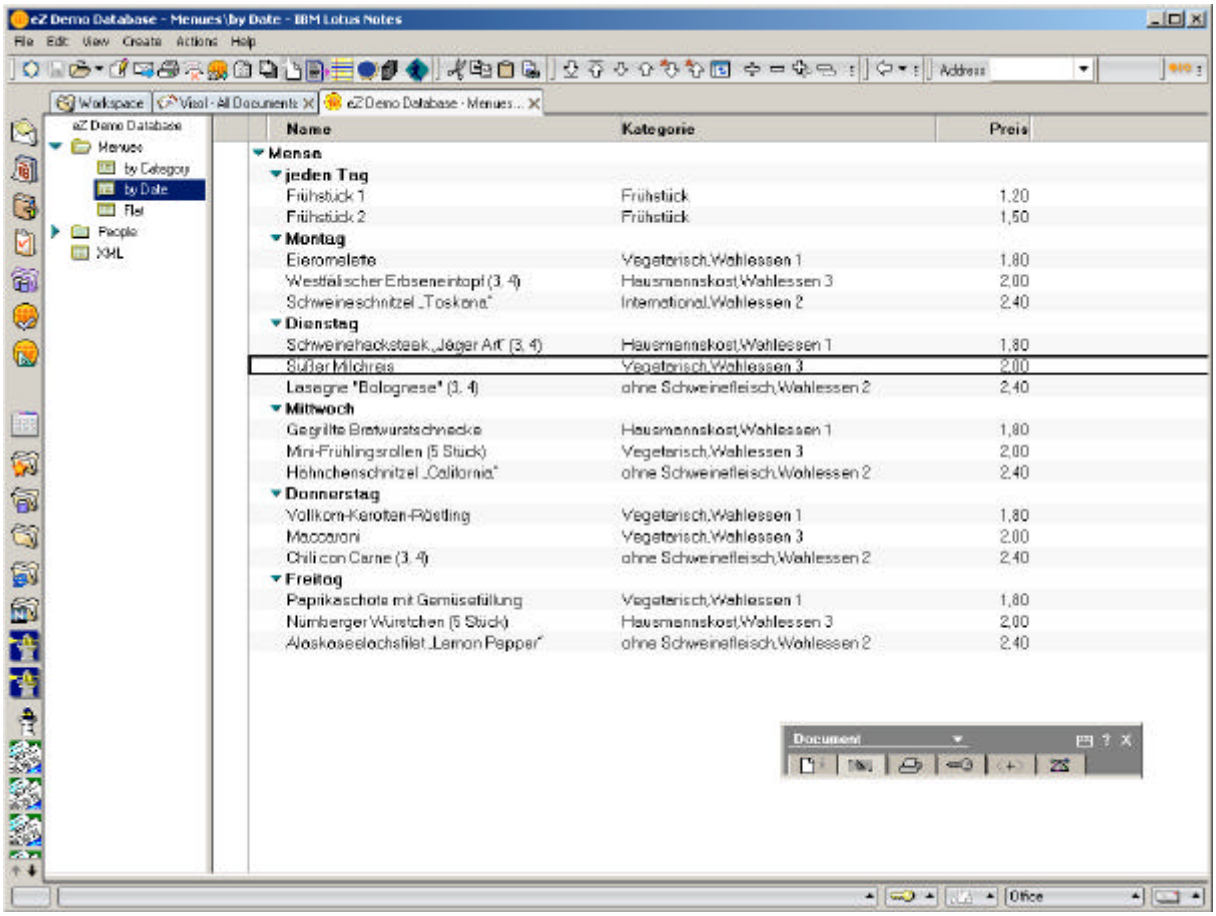


Abbildung 1a: Ein kategorisierter View im Notes-Client. Diese Art und Weise der View-Darstellung erlaubt durch Ein- und Ausklappen der Kategorien einen schnellen und individuellen Daten-Zugriff und ist deshalb die übliche Darstellung in Notes.

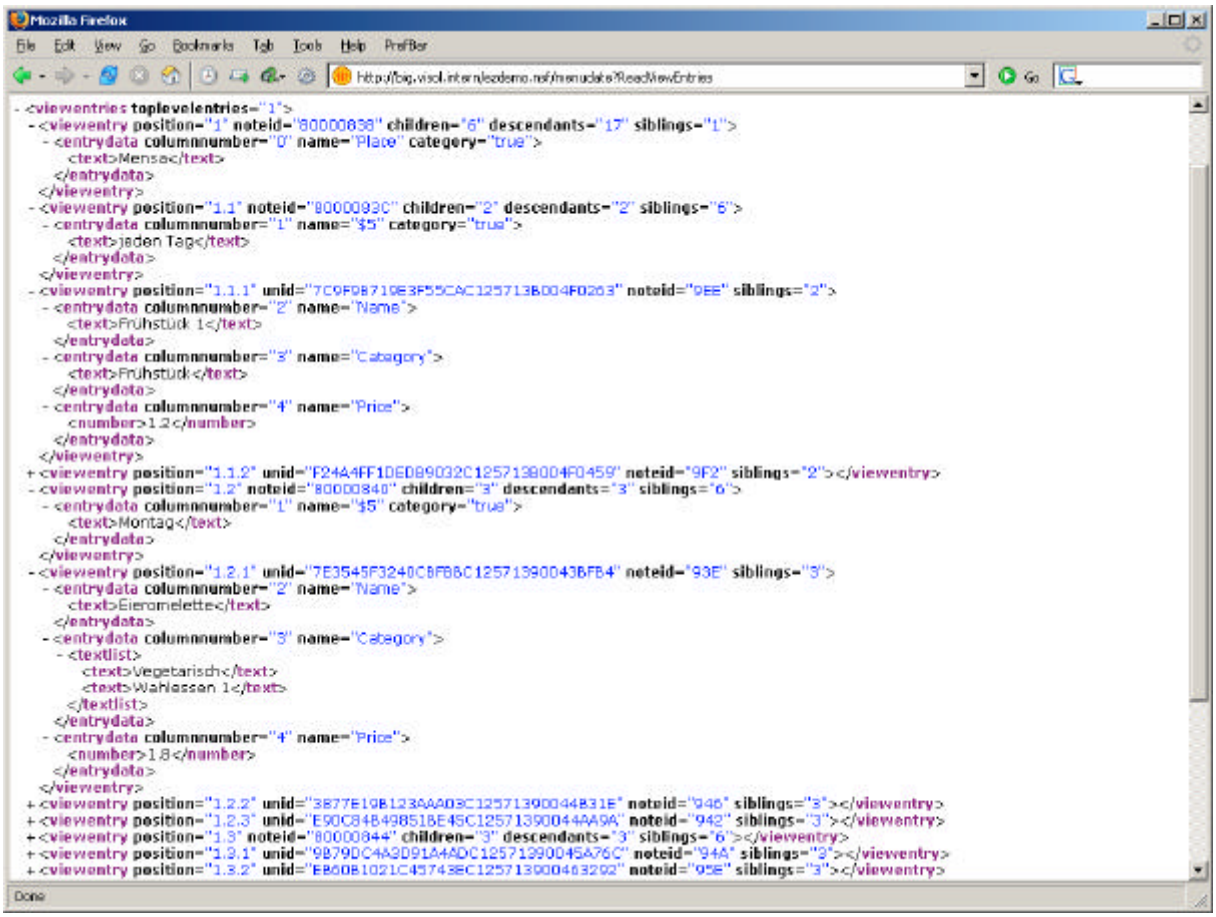


Abbildung 1b: Derselbe View als XML-Code: durch die Kategorisierung wird die XML-Ausgabe sehr komplex und ist daher zur Weiterverarbeitung in PHP wenig geeignet.

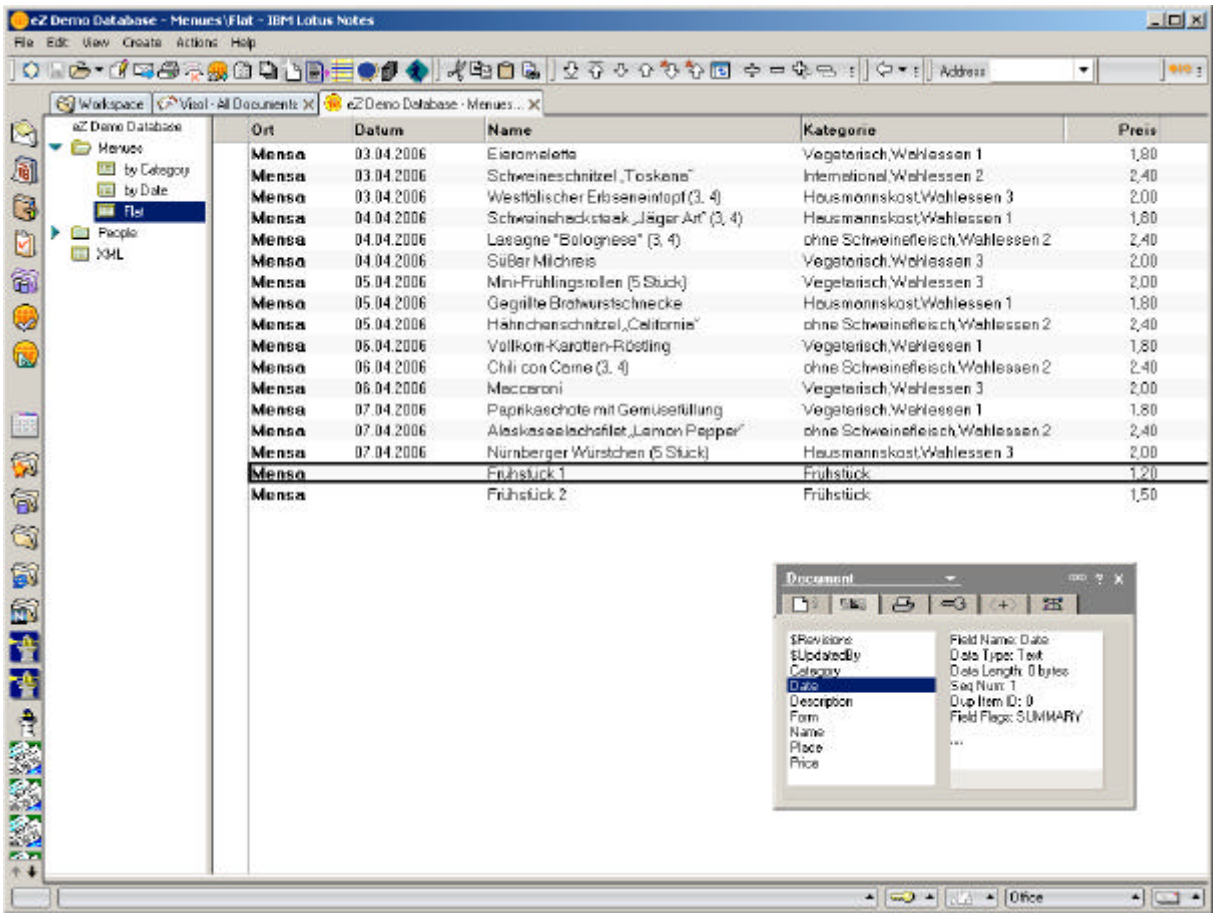


Abbildung 2a: Ein nicht-kategorisierter Notes-View. Spezielles Augenmerk verdient der markierte Eintrag: wie in der Properties-Box rechts unten zu sehen werden leere Date/Time-Felder als Text interpretiert, dies gilt auch für leere Number-Felder.

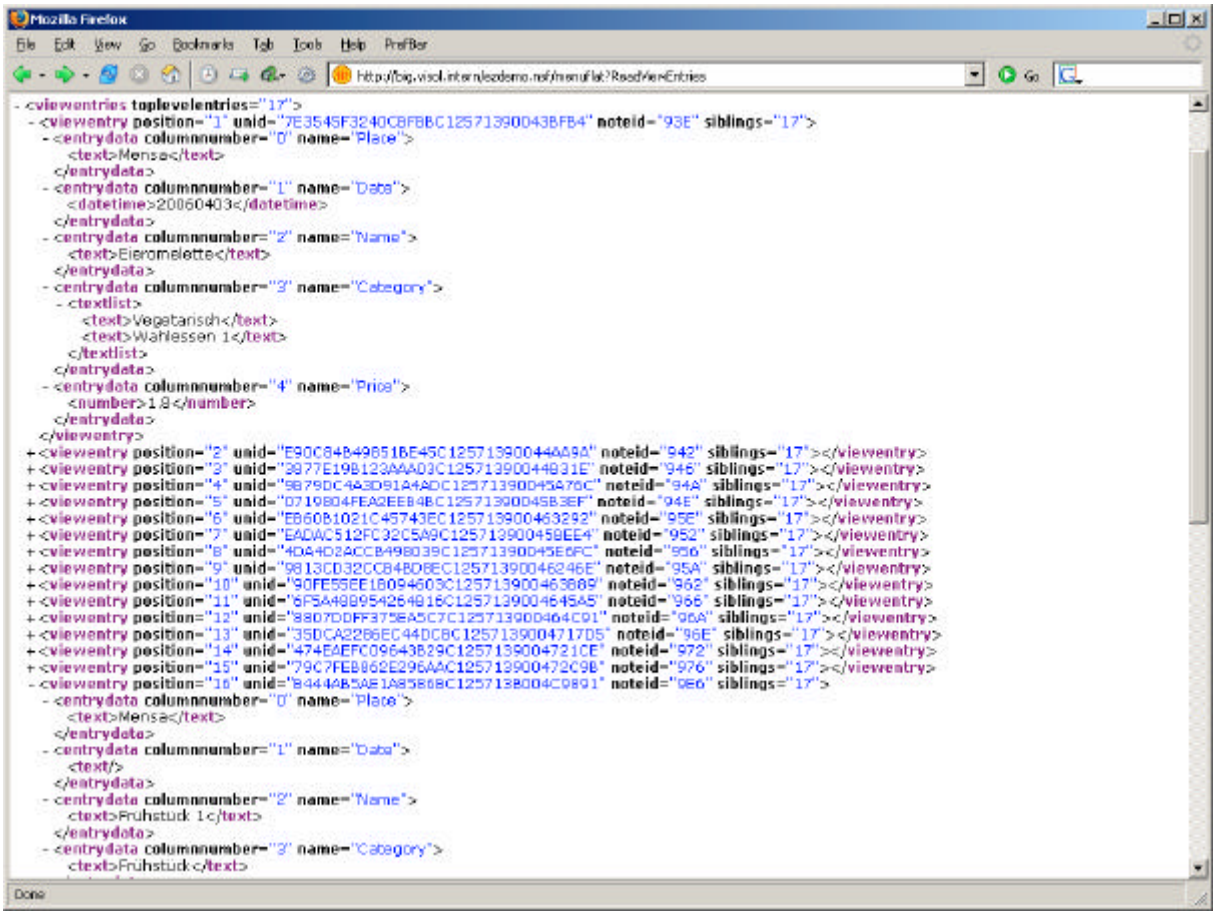
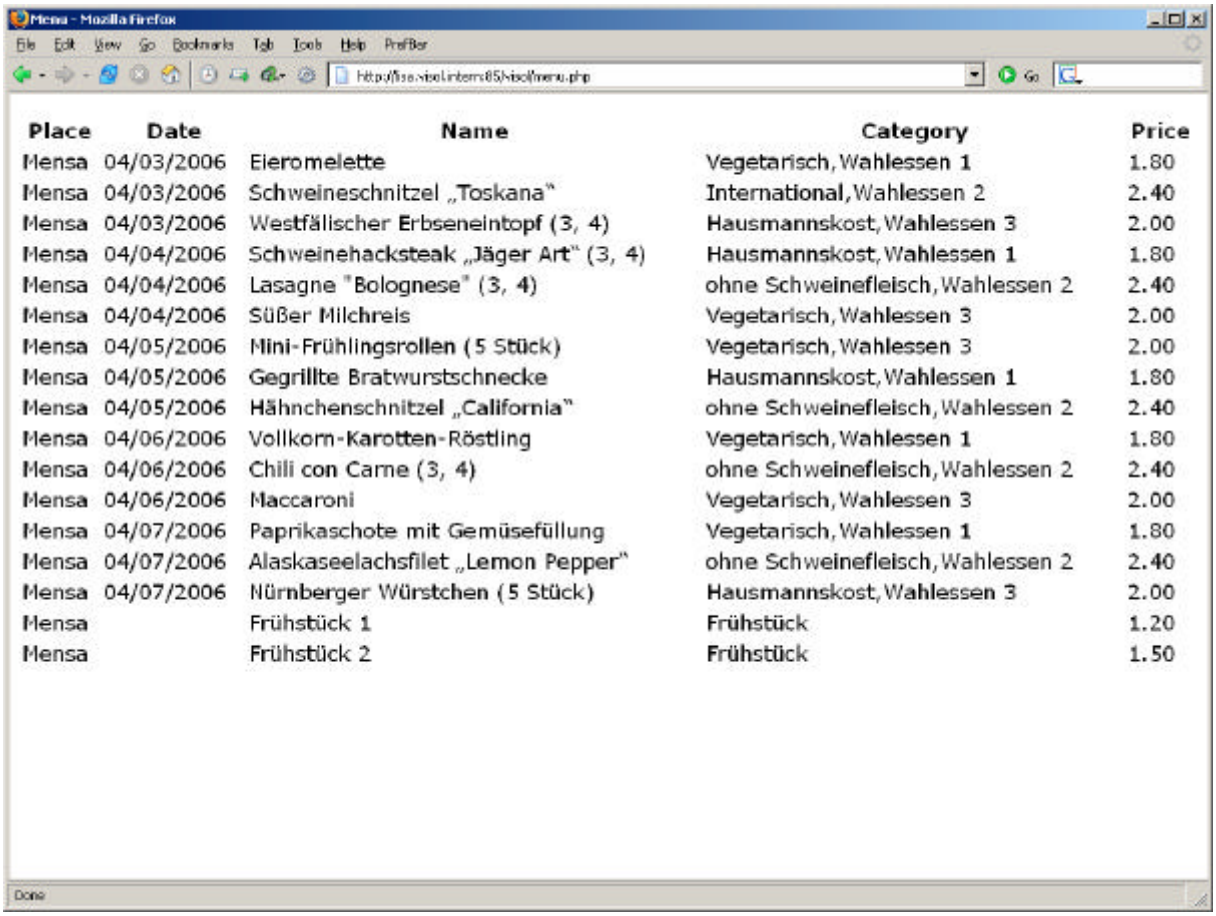


Abbildung 2b: Die XML-Ausgabe des nicht-kategorisierten Views ist wesentlich übersichtlicher. Auch hier sieht man, daß das leere Date-Feld des letzten Eintrages als Text interpretiert wird. Bei Listen-Feldern mit nur einem Eintrag entfällt der Tag <textlist> (siehe „Category" am unteren Bildschirmrand) – beides sind „Stolpersteine" beim Ausparsen in PHP.



Place	Date	Name	Category	Price
Mensa	04/03/2006	Eieromelette	Vegetarisch, Wahlessen 1	1.80
Mensa	04/03/2006	Schweineschnitzel „Toskana"	International, Wahlessen 2	2.40
Mensa	04/03/2006	Westfälischer Erbseneintopf (3, 4)	Hausmannskost, Wahlessen 3	2.00
Mensa	04/04/2006	Schweinehacksteak „Jäger Art" (3, 4)	Hausmannskost, Wahlessen 1	1.80
Mensa	04/04/2006	Lasagne "Bolognese" (3, 4)	ohne Schweinefleisch, Wahlessen 2	2.40
Mensa	04/04/2006	Süßer Milchreis	Vegetarisch, Wahlessen 3	2.00
Mensa	04/05/2006	Mini-Frühlingsrollen (5 Stück)	Vegetarisch, Wahlessen 3	2.00
Mensa	04/05/2006	Gegrillte Bratwurstschnecke	Hausmannskost, Wahlessen 1	1.80
Mensa	04/05/2006	Hähnchenschnitzel „California"	ohne Schweinefleisch, Wahlessen 2	2.40
Mensa	04/06/2006	Vollkorn-Karotten-Röstling	Vegetarisch, Wahlessen 1	1.80
Mensa	04/06/2006	Chili con Carne (3, 4)	ohne Schweinefleisch, Wahlessen 2	2.40
Mensa	04/06/2006	Maccaroni	Vegetarisch, Wahlessen 3	2.00
Mensa	04/07/2006	Paprikaschote mit Gemüsefüllung	Vegetarisch, Wahlessen 1	1.80
Mensa	04/07/2006	Alaskaseelachsfilet „Lemon Pepper"	ohne Schweinefleisch, Wahlessen 2	2.40
Mensa	04/07/2006	Nürnberger Würstchen (5 Stück)	Hausmannskost, Wahlessen 3	2.00
Mensa		Frühstück 1	Frühstück	1.20
Mensa		Frühstück 2	Frühstück	1.50

Abbildung 3: Die Ausgabe des Notes-Views in PHP. Gestaltung und Formatierung können über die PHP-Datei gesteuert werden.

PHP-Code der Notes-View Ausgabe:

```
<html>
	<head>
		<title>Menu</title>
		<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
	</head>
	<body bgcolor="ffffff">
		<table width="100%">

	<tr><th>Place</th><th>Date</th><th>Name</th><th>Category</th><th>Price</th></tr>

<?php
include_once "notesviewconnector.php";

$domino = new NotesViewConnector( 'big.visol.intern', 'ezdemo.nsf', 'menuflat', true);

while ( $viewentry = $domino->getViewEntry() )
{
	list ($place, $date, $name, $category, $price) = $viewentry['column'];
	echo("<tr><td>$place</td><td>$date</td><td>$name</td><td>$category</td><td>$price</td>
</tr>");
}
?>
		</table>
	</body>
</html>
```