FACHHOCHSCHULE KÄRNTEN

Fachhochschulstudiengang
Telematik / Netzwerktechnik

B A C H E L O R A R B E I T

Entwicklung eines webfähigen Templates

Unter Lotus Notes/Domino 8.5

zur Erlangung des akademischen Grades
Bachelor of Science

Autor: Dietmar Pichler
Matrikelnummer: 0810286040

Erstbetreuer: Ing. Andreas Aichholzer
Zweitbetreuer: FH-Prof. Dipl.-Ing. Dr. Manfred Baltl

Tag der Abgabe: xx.xx.2011

Name: Dietmar Pichler
Matrikelnummer: 0810286040

Geburtsdatum: 17.07.1981

Adresse: Dr. Franz Palla Gasse 17, 9020 Klagenfurt

Eidesstattliche Erklärung

Mit dieser Erklärung versichere ich die erstmalig vorgelegte Bachelorarbeit selbständig erstellt zu haben.
Es wurden nur die von mir angegebenen Hilfsmittel und Quellen verwendet.

Datum: xx.xx.2011
Ort: Klagenfurt

Unterschrift:

ABSTRACT

ENTWICKLUNG EINES WEBFÄHIGEN TEMPLATES

UNTER LOTUS NOTES/DOMINO 8.5

In dieser Arbeit wird die Erstellung einer Datenbank für die Groupware Software Lotus Notes/Domino© beschrieben. Diese Groupware Software unterstützt eine innerbetriebliche Kommunikation zwischen den Mitarbeitern und wird genutzt um administrative sowie auch projektbezogene Abwicklungen durchzuführen. Die Daten werden mit Lotus Notes/Domino verwaltet und in dafür speziell entwickelten Domino Datenbanken abgelegt.
Diese Arbeit dokumentiert das Erstellen einer Vorlage, unter Verwendung der aktuellsten Entwicklungsumgebung. Der Bedarf für diese Entwicklung ist von Nöten, da ein Großteil der Datenbanken unter einer älteren Entwicklungsumgebung erstellt wurden. Diese Datenbank soll als Hilfestellung dienen, um zu zeigen wie Datenbanken auf den aktuellsten Stand der unterstützten Technologien zu bringen sind. Ziel ist es, bestehende Datenbanken unter der Verwendung der neue gebotenen Elemente und Designmöglichkeiten, zu aktualisieren.

Schlüsselwörter: Lotus Notes, Datenbanken, Lotus Domino, Design-Elemente, XPages;

This thesis describes the development of a database for the groupware software Lotus Notes/Domino©. This groupware is used to support the communication in a company as well as for administrative and project related executions. Within Lotus Notes/Domino the data will be maintained and stored in specially developed domino databases. This thesis deals with the development of a template which is using the latest design environment. The template is needed because most of the already existing databases were created with an outdated design environment. Therefore, this template can be used to convey existing databases into the latest supported technologies. Because of the fact that the latest available layout and properties should be supported, this documentation describes which steps to follow to reach this goal.

keywords: Lotus Notes, databases, Lotus Domino, Design Elements, XPages;

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

1.1 Allgemeines

Die Firma Uniquare ist ein Softwarehaus mit dem Hauptfirmensitz in Krumpendorf/Österreich. Dieses Unternehmen hat sich auf die Entwicklung von Banken-Software spezialisiert. Uniquare Software Development GmbH verwendet als Workgroup-Software Lotus Notes/Domino von IBM©. Mit dieser Software werden annähernd die gesamten administrativen Prozesse, sowie auch Projektabwicklungen im Unternehmen verwaltet und unterstützt.
Lotus Notes/Domino wird fälschlicherweise oft als ein E-Mailsystem verstanden, jedoch handelt es sich hierbei um ein System, welches mehr bereitstellt. Dieses System stellt unter anderem Datenbanken, Workgroup-Software, Web-Server etc. zur Verfügung. Es eignet sich um firmeninterne Kommunikationsprozesse zu vereinfachen und diese zu beschleunigen [11].

1.2 Aufgabenstellung

Die sich im Unternehmen in Verwendung befindenden Datenbanken wurden seit der Einführung von Lotus Notes im Betrieb fortlaufend entwickelt. Lotus Notes und später auch Lotus Domino, werden seit der Version 2.0 in diesem Unternehmen verwendet.
Da nicht alle Kunden Lotus Notes/Domino nützen, gibt es den Wunsch, auf projektbezogene Datenbanken über einen Browser zugreifen zu können. Somit kann sich der Kunde selbst, aktiv am Projektablauf beteiligen. Die technischen Möglichkeiten ohne Lotus Notes auf Domino Applikationen zuzugreifen waren bis zum Release der Version 4.x nicht gegeben.
Das Ziel dieser Arbeit ist es, eine Schablone für die Überführung von Datenbanken (Version 4.x zu 8.5) zu erstellen. Aus diesem Grund wird ein Template (eine Vorlage), unter der Verwendung der XPages-Technologie, erstellt. Mit diesem Template kann in weiterer Folge, unter geringem Aufwand, eine Überführung auf die Version Lotus Notes/Domino 8.5.2 durchgeführt werden.

Kapitel 2

Einführung in Lotus Notes/Domino

2.1 Was ist Lotus Notes/Domino?

Lotus Notes/Domino ist eine Anwendung, welche in Form einer Server/Client Architektur (Kapitel 2.3) aufgebaut ist. Bis zur Version 4.5 (erschienen 1996) existierte lediglich *Lotus Notes*. Mit der stärker werdenden Popularität des Internets wurde die Unterscheidung von Notes und Domino komplexer. Ab der Version 4.6 war vom *Domino Designer* die Rede, zu diesem Zeitpunkt, war Domino jedoch nur eine Teilanwendung, welche ein Hypertext Transfer Protocol (HTTP)-Übersetzer war. Diese Teilanwendung sorgte dafür, dass Inhalte von Notes-Datenbanken in Hypertext Markup Language (HTML) übersetzt wurden. Es wurde von nun an nicht nur in Richtung des *Rich Client* (Notes Client), sondern parallel dazu auch in Richtung des *Internets* entwickelt. Das gesamte Server-Bundle und nicht nur eine einzelne Teilanwendung, wurde ab sofort in Domino umbenannt. Weiters wurde entschieden, dass die Seite des Clients weiterhin als Notes bezeichnet wird. Zusammenfassend kann gesagt werden, dass Lotus Notes die Client- und Domino die Serverseite darstellt [4].

2.2 Domino Designer-Kurzüberblick

Die Entwicklungsumgebung um Anwendungen zu entwerfen nennt sich *Domino Designer*. In diesem wird eine Vielzahl von Design-Elementen zur Verfügung gestellt, auf welche in Kapitel 3.1 eingegangen wird [6].
Der Domino Designer stellt eine integrierte Entwicklungsumgebung zum Entwurf von Domino-Anwendungen bereit. In diesem Zusammenhang bedeutet integriert, dass der gesamte Prozess der Anwendungsentwicklung in einem einheitlichen Rahmen und einer einheitlichen graphischen Benutzeroberfläche durchgeführt wird. Beginnend beim Entwurf, über die Implementierung, bis hin zur Dokumentation [4].
Im Domino Designer, werden die Programmiersprachen Java, JavaScript, LotusScript sowie auch die Formelsprache von Lotus Notes unterstützt. Diese unterstützten Programmiersprachen, werden in Kapitel 3.3 angeführt. Damit öffnet sich das Integrated Design Environment (IDE) von Domino Designer allen unterstützten Programmiersprachen. Somit wird ein fehlerfreies und rasches Codieren ermöglicht. Nicht zuletzt deshalb, weil ein syntaxsensitiver[1] Editor mit Highlight-Funktionen zur Verfügung gestellt wird [6].

[1] Gibt unterschiedliche Codeelemente in einer farblichen Unterscheidung wieder.

2.3 Server/Client-Prinzip

In diesem Abschnitt wird das Server/Client Prinzip erklärt. Für detailliertere Informationen wird auf entsprechende Fachliteratur verwiesen.
Anstatt die gesamte Funktionalität einer Anwendung zuzuteilen, wird eine Trennung in *mindestens* zwei logische Schichten vorgenommen. Es ergeben sich der *Server* und der *Client*. Der Server hat die Aufgabe auf Anfragen von einem oder mehreren Clients eine bestimmte Funktion bereitzustellen. Das bedeutet, dass der Client Daten darstellt, welche vom Server bereitgestellt werden. Der Server kann Daten extrahieren ohne sich um die Darstellung dieser zu kümmern. Diese Architektur hat den Vorteil das Ressourcen besser genutzt werden können [4].

2.3.1 Notes Client

Ein Domino-System ist für die Arbeit mit mehreren verschiedenen Typen von Clients ausgerichtet. Es können zum Beispiel Teile der Domino Funktionalität auch von Post Office Protocol Version 3 (POP3) - oder Internet Message Access Protocol Version 4 (IMAP) - fähigen E-Mail-Clients genutzt werden. Das POP3 ist ein Übertragungsprotokoll, über welches ein Client auf E-Mails von einem Mailserver zugreifen kann [17]. Das IMAP4 ist ein Protokoll, das den Online-Zugriff auf ein E-Mail-Postfach ermöglicht [16].
Es gibt nur einen Client welcher in der Lage ist alle Funktionalitäten von Domino zu übernehmen: der *Notes Client*. Der Notes-Client ist ein grafischer Personal Information Manager (PIM). Eine Besonderheit des Notes-Clients ist, dass er seine eigene Anwendungsentwicklung, namens *Domino Designer* (Kapitel 3.1), bereitstellt [4].

2.3.2 Domino-Server

Der Kern des Domino-Servers besteht aus einem Datenbank-Server. Vergleichbar mit einem relationalen Datenbank-Server, ermöglicht dieser einen gleichzeitigen Zugriff für mehrere Benutzer. Es muss jedoch klar festgelegt werden, dass Notes *keine* relationale Datenbank ist. Dieser Datenbank-Server wird als *Task*, welcher auf einem Domino-System läuft, bezeichnet. Neben dieser Aufgabe werden ein Reihe von Modulen bereitgestellt, welche als separate Anwendungen beendet oder gestartet werden können [4]. Von einer vollständigen Auflistung aller Server-Tasks wird in dieser Arbeit abgesehen.

Die Tasks können in folgende Gebiete aufgeteilt werden:

1. Erweiterung der Kernfunktionalität

   (a) Einige der Kernfunktionalitäten von Domino sind in Form von separaten Tasks am Server realisiert. Beispiele für diese Server-Tasks sind die Volltextindizierung, der Replikationsprozess, der Agent-Manager sowie der Mail-Router. Volltextindizierung bedeutet, dass der vollständige Text erfasst und in Einzelteilen registriert bzw. katalogisiert wird.

2. Administrationsfunktionen

   (a) Da es umfangreich wäre die Administration manuell durchzuführen, sind einige Administrator-Tätigkeiten in Form von Server-Tasks beinahe vollständig automatisiert. Es handelt sich hierbei um die Generierung von Statistiken, das Komprimieren sowie das Reparieren und die Verwaltung von Datenbanken.

3. Internet-Tasks

   (a) Diese Tasks besitzen in erster Linie die Aufgabe, Domino-Funktionalität für Internet-Clients zur Verfügung zu stellen wie beispielsweise der HTTP-Task, aber auch der Mail-Zugriff über POP3 und der IMAP4-Task.

Der Server verfügt über keine graphische Benutzeroberfläche, sondern wird mittels Konsolen-Kommandos bedient. Mit diesen Kommandos können einzelne Server-Tasks manuell gestartet oder beendet werden [4]. Es besteht die Möglichkeit, mittels des Domino-Administrators (Kapitel 2.3.3) die Konfiguration über eine graphische Oberfläche vorzunehmen.

2.3.3 Domino-Administrator

Der Domino-Administrator ist der Administrator-Client für Notes und Domino. Er bietet *Tasks* an, welche für Administrationsaufgaben verwendet werden. Ein solcher Task ist zum Beispiel Domino Domain Monitoring (DDM). Mit DDM werden Überwachungsfunktionen von Domino-Servern durchgeführt. Ziel ist es, den Status von mehreren Domino Domänen zentral zu überblicken.
Dieser Administrator bietet Serverinformationen und Administrationsmöglichkeiten der Infrastruktur an einer zentralen Stelle. Für zusätzliche Informationen wird auf [2] verwiesen.

2.4 Domino-Datenbank

Die Informationen für dieses Kapitel wurden sofern nicht anders angegeben, aus [2] und [4] entnommen.

2.4.1 Allgemeine Beschreibung

Eine Domino Datenbank, auch als Notes Datenbank bezeichnet, enthält alle Daten und Informationen einer Anwendung. Die Datenstruktur, die Funktionalität und die Oberfläche werden durch das Design einer Anwendung festgelegt. Eine Datenbankdatei kann als eine Art Container gesehen werden. In diesem Container sind Gestaltungs-/Design-Elemente, wie auch die Daten der Anwendung abgelegt. Domino-Datenbanken sind die Grundbausteine einer Domino-Anwendung. Diese unterscheiden sich von klassischen Datenbanken. Während klassische Datenbanken eine reine Ansammlung von Daten sind, können Domino-Datenbanken auch beliebig gestaltet werden. Eine Domino-Datenbank ist eine Ansammlung von miteinander in sachlicher Beziehung stehenden Dokumenten. Jedes Dokument wiederum besteht aus einem oder mehreren Feldern [4].
Eine Domino-Datenbank ist *keine* Datenbank im herkömmlichen Sinn, sondern eine fertige Applikation, welche eine spezielle Ablaufumgebung benötigt. Diese kann entweder der Notes-Client (Kapitel 2.3.1) oder ein Domino-Server (Kapitel 2.3.2) sein [4].

Relationale Datenbank versus Domino-Datenbank

Gegenüber klassischen relationalen Datenbanken, ist eine Domino Datenbank besonders für eine flexible Verwaltung von unstrukturierten Inhalten geeignet. Eine Domino-Datenbank hat

gegenüber einer relationalen Datenbank einige Vorteile. Insbesondere, wenn es sich um die Verwaltung von unstrukturierten Inhalten handelt, besitzt die Domino-Datenbank Vorteile. Es können Inhalte von beliebigen Typen wie zum Beispiel

- Office Dokumente,
- Applets[2],
- Bilder,
- oder Videos

verwaltet werden. Hierbei kann die Struktur von einzelnen Datensätzen verändert werden [4]. Zu den Vorteilen einer Domino-Datenbank gehört, dass mit Rich-Text-Feldern jede erdenkliche Information, wie zum Beispiel Texte, Bilder, Tabellen, Grafiken sowie Object Linking and Embedding (OLE)-Objekte oder ganze Dateien abgelegt werden können. OLE ist ein Objektsystem und Protokoll, das die Zusammenarbeit unterschiedlicher (OLE-fähiger) Applikationen ermöglicht [2].

Der Nachteil dieser Anwendungen ist, dass es keine Sicherstellung der Datenqualität gibt. Ein weiterer Nachteil ist, dass keine automatische Vermeidung von Redundanzen vorhanden ist. Darum soll eine Domino-Datenbank nicht eingesetzt werden, wenn es sich um die Verwaltung von großen Mengen von strukturierten Daten handelt. Wie in Kapitel 1.1 erwähnt, handelt es sich um eine Groupware-Anwendung. Aus diesem Grund kann ausgeschlossen werden, dass ein solcher Fall eintritt [4].

2.4.2 Datenbankarchitektur

Physisch gesehen wird eine Domino-Datenbank durch eine Datei mit *.nsf*-Endung abgebildet. Die Endung *.nsf* steht für Notes Storage Facility. Wie dem Namen zu entnehmen ist handelt es sich um die Möglichkeit Notizen (Notes) aufzunehmen. Diese Notizen, oder in anderen Worten gesagt, Dokumente oder Datensätze enthalten Felder, welche mit Daten oder Informationen gefüllt sind [2].

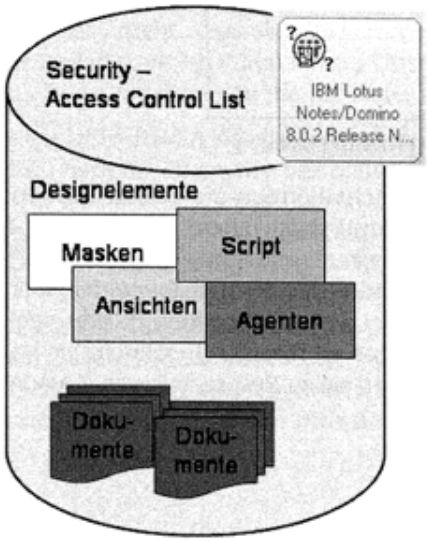

Abbildung 2.1 Elemente einer Domino-Datenbank [2]

[2]Ein Applet ist ein im Web-Browser laufendes Java-Programm.

Die Abbildung 2.1 zeigt den Aufbau einer Domino-Datenbank. Es ist erkennbar, dass Design-Elemente und Dokumente in einem gemeinsamen Container abgelegt sind.

2.4.3 Datenbank-Informationen

Die Datenbank-Informationen umfassen alle Eigenschaften einer Datenbank. Diese Eigenschaften sind für die gesamte Datenbank gültig.

Zu diesen Informationen gehören unter anderem:

- Identifikation einer Datenbank,
- Datenbanktitel,
- Name der Datenbankdatei,
- maximale Größe,
- verfügbarer Speicherplatz.

Die Eigenschaftseinstellungen werden beim Erstellen einer Datenbank durch den Ersteller spezifisch angegeben. Weiters besteht auch die Möglichkeit, dass die Einstellungen durch das System automatisch vorgenommen werden.

Design-Elemente

Wie in Kapitel 2.4.1 erwähnt, besteht eine Domino-Datenbank nicht nur aus Daten sondern bietet auch eine Reihe von Gestaltungs- oder Design-Elementen. Informationen über Design-Elemente werden in Dokumenten verwaltet. Dies bietet die Möglichkeit sie zu kopieren, löschen oder replizieren zu können. Die notwendigen Design-Elemente für das Template, werden in Kapitel 3.2 aufgelistet und beschrieben.

Daten

Die Verwaltung der eigentlichen Daten wird, wie auch die der Design-Elemente, über Dokumente erledigt. Ein Dokument ist eine autonome Einheit, welche die Struktur eigenständig verwaltet. Dies hat den Vorteil, dass die Größe und die Struktur des Dokumentes zu jedem Zeitpunkt beliebig verändert werden kann. Dokumente besitzen die Funktion sich wie ein Behälter für Grafiken, Texte, Objekte oder multimediale Daten zu verhalten. Diese Funktionalität wird als *compound-document* bezeichnet und ist für relationalen Datenbanken, nur über umständliche Wege realisierbar. Ein compound-document ist ein aus mehreren Objekten zusammengesetztes Dokument [18].

In einer dokumentorientierten Umgebung wie Notes bilden compound-documents die Grundlage einer jeden Internet-/Groupware-/Workflow-Anwendung [4].

2.4.4 Zugriffs-Möglichkeiten auf eine Domino-Datenbank

Um Anwendungen, welche sich auf dem Domino-Server befinden, auszuführen, gibt es unterschiedliche Möglichkeiten. In diesem Kapitel werden diese Zugriffsarten behandelt.

Benutzer Zugriff

Im Gegensatz zu Lotus Domino stellt Lotus Notes die Seite des Clients dar. Der Notes Client verwaltet die Benutzeroberfläche und ist für die grafische Aufbereitung der Daten zuständig. Notes Clients interagieren mit Datenbanken, welche sich am Domino-Server befinden, in unterschiedlicher Weise. Grundsätzlich gibt es zwei mögliche Zugriffsarten:

- Zugriff mit Notes Client,
- Zugriff über einen Web-Browser, zum Beispiel mit Internet Explorer oder Mozilla Firefox.

Zugriff über Notes Client

Für die Kommunikation eines Notes Clients mit einem Domino Server werden Protokolle benötigt. Diese Kommunikation wird entweder über Notes-Protokolle (Notes Remote Procedure Call (NRPC)) oder über Internet-Protokolle wie POP3 oder IMAP4 abgewickelt (Kapitel 2.3.1). Notes RPC ist eine Variante von Remote Procedure Call und kann über Protokolle wie TCP/IP (Transmission Control Protocol/Internet Protocol) geroutet werden. Der Anwender greift dabei auf Datenbanken, auf die er die Zugriffsberechtigung besitzt, zu. Diese Datenbanken werden als Datenbanksymbole auf dem Arbeitsplatz im Notes Client abgelegt (Abbildung 2.2) [2].

Abbildung 2.2 Datenbanken im Notes Client

Zugriff über Web-Browser

Um den Zugriff über einen Browser zu ermöglichen, wird vorausgesetzt, dass sich die Anwendung auf einem Domino-Server befindet, welcher als Webserver agiert [2]. Die eindeutige Adresse einer Ressource oder eines Dokuments im Internet bezeichnet man als Uniform Resource Locator (URL). Eine URL setzt sich nach bestimmten syntaktischen Regeln zusammen. Diese Regeln enthalten unter anderem den Namen des Host-Rechners und den Pfad bzw. den Namen des Dokuments [4].

Eine URL kann sich zum Beispiel wie folgt zusammensetzen:

$$http://s124/work/template.nsf/default.xsp \tag{2.1}$$

In diesem Beispiel wird eine XPage, namens *default*, aufgerufen. Diese XPage ist ein Teil der Datenbank *template* und *work* ist das Verzeichnis auf dem Server *s124*, der zu diesem Dokument führt. Hierbei handelt es sich um einen Webserver, welcher HTTP zum Informationsaustausch benutzt. Das hier angeführte Beispiel, beschränkt sich auf den lokalen Zugriff über das Intranet.

Wird der Zugriff über das Internet durchgeführt, so setzt sich die URL wie folgt zusammen:

$$http://server.domain.com/work/template.nsf/default.xsp \tag{2.2}$$

Die sich auf dem Domino Server befindenden Datensätze können somit über einen Web-Browser aufgerufen werden.

Kapitel 3

Design einer Datenbank

3.1 Domino Designer 8.5

In diesem Kapitel werden der Domino-Designer, die *Tools* und die Elemente, die dieser bereitstellt, behandelt. Es werden jedoch *nicht* alle verfügbaren Elemente angeführt, sondern lediglich diese Design-Elemente welche notwendig sind, um das geforderte Template (Kapitel 3.4) zu erstellen. Für weitere Design-Elemente wird auf Fachliteratur verwiesen.

Die Abbildungen 3.1 und 3.2 zeigen die Benutzeroberfläche des Domino-Designers.

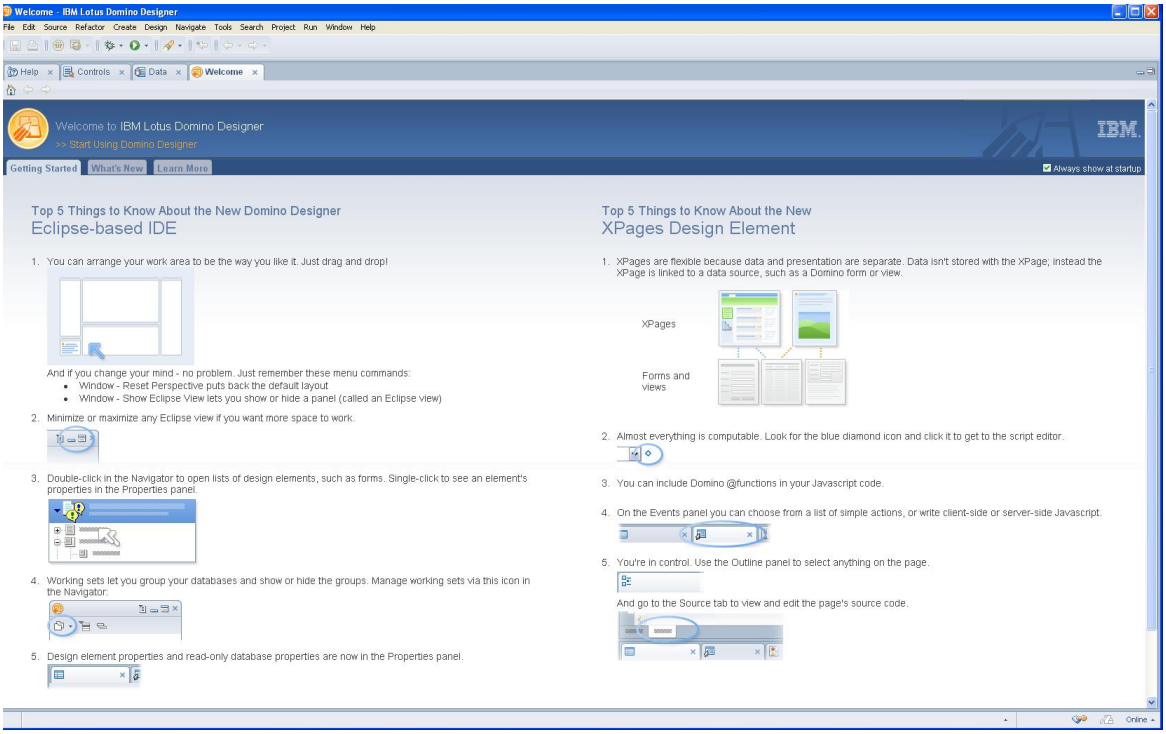

Abbildung 3.1 Startansicht Domino-Designer 8.5

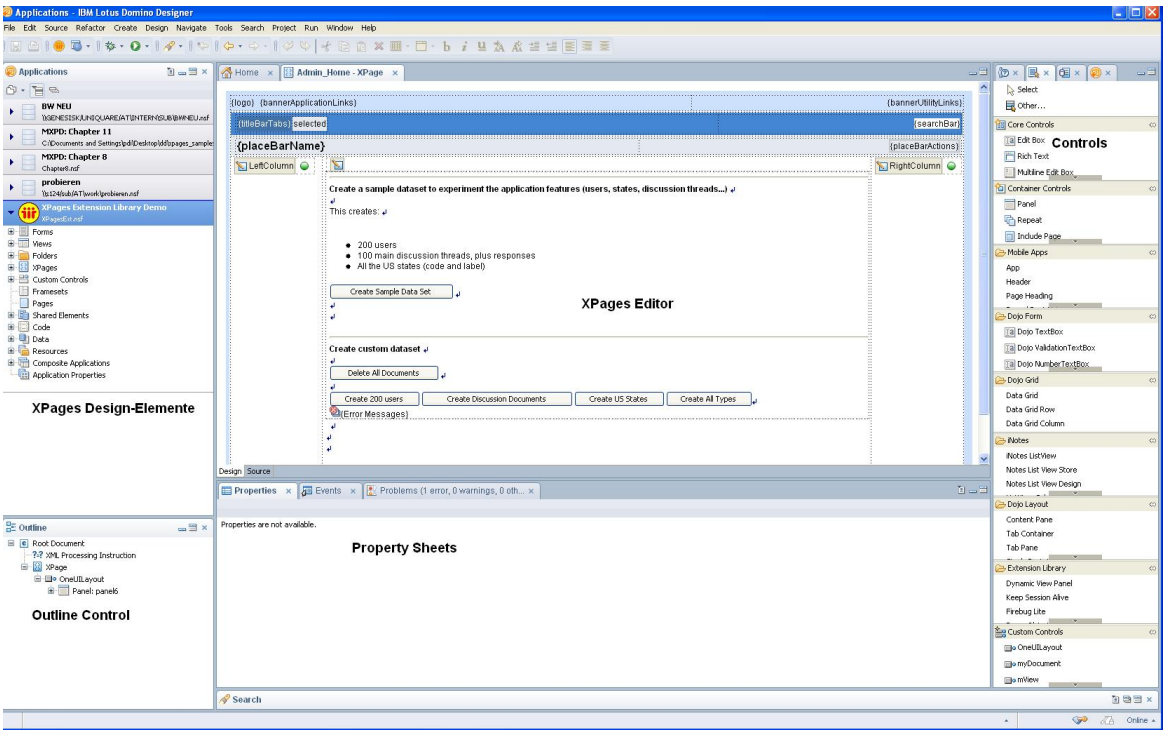

Abbildung 3.2 Verfügbare Tools im Designer 8.5

Abbildung 3.2 zeigt den Designer und wie dieser nach dem Erstellen eines Design-Elements, in diesem Fall einer XPage, verfügbar ist. Erst nach dem Erstellen einer XPage werden die Tools des Designers angezeigt und können auch verwendet werden.

Diese Tools sind:

- **XPages Editor,**
- **Controls Palette,**
- **Outline Control,**
- **Property Sheets,**
- **XPages Design-Elemente.**

Die Informationen für die Beschreibung der Tools, wurden aus [1] entnommen.

XPages Editor

Dieser Editor wurde speziell für XPages entworfen und unterstützt zwei Arten von Operationen:

1. **Default Mode:** In diesem Modus kann *visual editing* vorgenommen werden. Das bedeutet, dass mittels Doppelklick in den Editor, zum Beispiel direkt Text eingegeben werden kann. Design-Elemente können mittels *drag and drop* eingefügt werden.

2. **Source Mode:** Eine XPage kann auch mittels Programmierung erstellt werden, da es sich um eine Extensible Markup Language (XML)-Datei handelt (Kapitel 3.2.4). In diesem Modus können tags auf direkte Weise eingegeben werden.

> *Ein tag ist ein HTML Sprachelement, mit dem man unter Verwendung eines WWW-Browsers HTML Dokumente erstellen kann [15].*

Controls Palette

Hier werden die verfügbaren *controls*-Elemente des Designers abgebildet, nachdem die *Extension Library* (Kapitel 3.4.1) eingebunden wurde. In diesem Tool des Designers sind alle *standard user interface controls* aufgelistet, welche zu einer XPage für die Entwicklung einer Applikation hinzugefügt werden können.

Outline Control

Um die Navigation innerhalb einer XPage zu erleichtern, wird eine Outline-Control zur Verfügung gestellt. Dieses Tool bildet eine hierarchische Darstellung der XPage ab, in welcher alle Elemente einer XPage aufgelistet sind.

Property Sheets

In diesem Teil des Designers, kann für ein ausgewähltes Design-Element die spezifische Konfiguration vorgenommen werden.

Dieser beinhaltet:

- **Properties:** Ermöglicht unter anderem die Namensgebung des Elements oder auch die Darstellung für das ausgewählte Element.
- **Events:** Hier können events, wie zum Beispiel *onClick*-Events, bestimmt werden. Ein onClick-Event bestimmt welcher logische Schritt beim Anklicken eines Elements ausgeführt wird. Unter Verwendung dieser Events, kann die Logik der Programmierung in diesem Tool des Designers vorgenommen werden. Das bedeutet, die Abfolge und Zusammenhänge der einzelnen XPages können an dieser Stelle festgelegt werden.
- **Problems:** Hier werden Fehlermeldungen und Warnungen der aktuell geöffneten XPage angezeigt.

XPages Design-Elemente

Hier sind alle verfügbaren Design-Elemente, welche der Domino Designer bereitstellt, angeführt. Die XPages Design-Elemente sind der Kernpunkt dieser Arbeit, darum werden diese im anschließenden Kapitel ausführlich behandelt.

3.2 Details der XPages Design-Elemente

3.2.1 Informationen zu Design-Elementen

Eine Datenbank ist der Aufbewahrungsort für die Daten und die Gestaltungselemente, die sich in einer Anwendung befinden. Design-Elemente sind die Bausteine mit welchen eine Anwendung erstellt wird.

Die hier angeführten Elemente sind nur ein Teil der verfügbaren Design-Elemente:

- Seiten,
- Masken,

- Felder,
- Ansichten,
- Agenten,
- XPages,
- Custom Controls.

Um eine Applikation in Lotus Notes zu entwickeln, sind Design-Elemente zwingend notwendig. Es wurden für die Version 8.5 neue Design-Elemente, wie zum Beispiel XPages und Custom Controls, implementiert. Diese dienen im speziellen der weborientierten Darstellung. Haupt-Design-Elemente wie *forms* und *views* sind bereits seit älteren Versionen implementiert und werden auch weiterhin unterstützt. Die neu implementierten Elemente zielen auf die Entwicklung für Intra-und Internet. Es wird jedoch dem Entwickler einer Notes-Anwendung, mit dem Schwerpunkt auf den Notes-Client, eine Reihe von Möglichkeiten geboten, um die Entwicklung zu vereinfachen [6]. Wie bereits in Kapitel 1.2 erwähnt, wird in dieser Arbeit ein Template entwickelt, um Notes-Datenbanken mittels Browser-Zugriff zu verwalten. In Folge werden die notwendigen Elemente im Einzelnen angeführt. In Abbildung 3.3 sind die verfügbaren Design-Elemente des Domino Designers 8.5 ersichtlich.

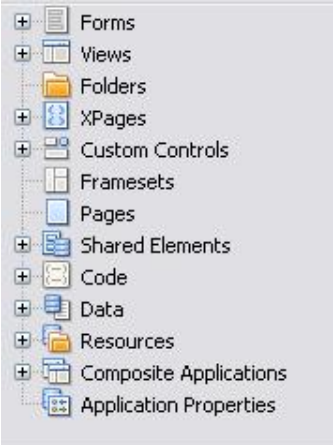

Abbildung 3.3 Design-Elemente

3.2.2 Masken (Forms)

Masken sind elektronische Formulare, welche für die Erstellung von Dokumenten über das graphical user interface (GUI) verwendet werden können. Notes-Dokumente werden aus Masken erstellt, die von einem Anwendungsentwickler gestaltet werden. Eine dieser Masken besteht in der Regel aus mehreren Felddefinitionen. In diesen ist die Datenstruktur eines bestimmten Dokumenttyps festlegt. Des weiteren kann eine Maske Formatierungsmerkmale wie zum Beispiel Text, Grafiken und Tabellen beinhalten. Es können auch fortgeschrittene Features, wie OLE-Objekte (nur im Notes Client) und Java-Applets in Masken enthalten sein. Die Notwendigkeit von Masken liegt darin, da sie der häufigste Weg sind, um Inhalte einer Domino-Datenbank zu erstellen.
Masken dienen als *Schablonen für Dokumente*. Bei der Erstellung einer Maske, wird die interne Struktur des Dokumentes nach der Schablone festgelegt. Realisiert wird diese Strukturierung unter der Verwendung von Feldern. Domino erstellt intern eine Verknüpfung zwischen dem erstellten Dokument und der gewählten Maske. Wird ein Dokument aufgerufen, wird dieses

automatisch im Rahmen der jeweiligen Maske dargestellt (Abbildung 3.4). Weiters kann somit auch die Formatierung eines Dokumentes beeinflusst werden [4].

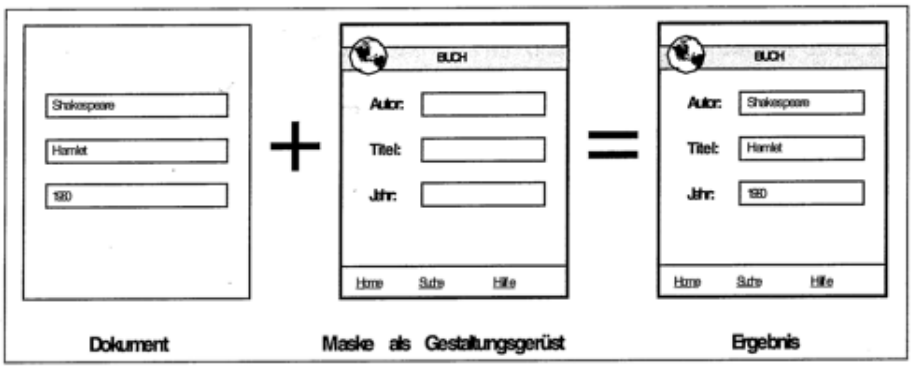

Abbildung 3.4 Masken-Gestaltungsgerüst [4]

Es gibt die Möglichkeit eine Maske von Grund auf neu zu erstellen, da es aber für die bestehenden Datenbanken bereits Masken gibt, wird dies nicht weiter behandelt.

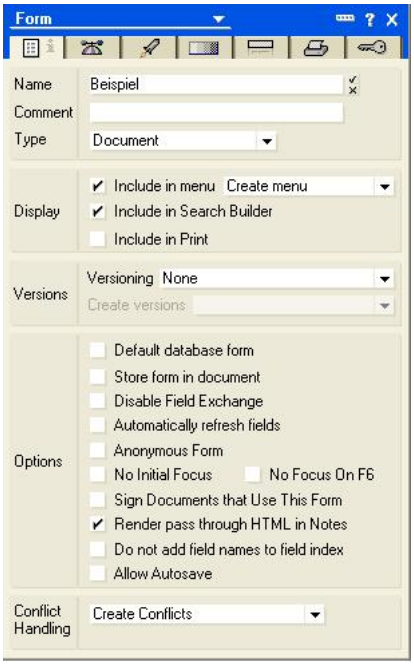

Abbildung 3.5 Eigenschaften einer Maske

Die Abbildung 3.5 zeigt das Eigenschafts-Fenster einer Maske. In diesem Bedienfenster können unter anderem der Namen und das Verhalten einer Maske festgelegt werden. Für eine detailliertere Beschreibung wird auf die Fachliteratur verwiesen.

3.2.3 Ansichten (Views)

Das Ziel eines Entwicklers ist es, dem Anwender genau *die* Information zu liefern, welche er benötigt bzw. erwartet. Für die Darstellung von Dokumenten sind Ansichten notwendig. Eine solche besteht aus Spalten, in welchen die Werte aus Dokumenten oder berechnete Werte angezeigt werden. Die berücksichtigten Dokumente werden über eine Selektionsformel ausgewählt.

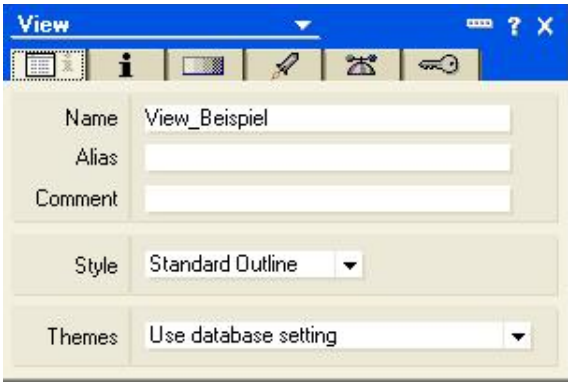

Abbildung 3.6 Eigenschaften einer Ansicht

Im Eigenschaftsfenster einer Ansicht werden die *eigentlichen* Einstellungen vorgenommen. Neben den Basic-Konfigurationen wie der Namensgebung, können hier die Ansichtsoptionen, die Formatierung und eine Reihe weiterer Einstellungen einer Ansicht festgelegt werden.

Vor der Version 8.5 des Designers war eine teilweise, separate Programmierung für die Darstellung im Web oder im Notes-Client notwendig. Um dies zu erreichen, konnten Masken und Ansichten optional, freigegeben oder verborgen werden. Somit ergaben sich unabhängige Programmierungen für die Benutzung durch Client und Browser, jedoch stieg damit auch der Programmieraufwand.

3.2.4 XPages

Wie in diesem Kapitel bereits erläutert, ist jedes Design-Element für eine spezielle Aufgabe zugeschnitten. *Masken* sind notwendig um Notes-Dokumente zu erstellen und zu bearbeiten, während *Ansichten* für die Darstellung der Dokumente benötigt werden. Im Grunde genommen sind XPages, statische Seiten die im Domino-Designer definiert werden. Bis zur Version 8.5 musste für die Darstellung im Web oder Notes-Client in der Programmierung Rücksicht auf die gewünschte Art der Darstellung (Web oder Client) genommen werden. Es wurde meist eine unterschiedliche Version für das Web und den Client erstellt.
XPages ermöglichen es, mit einer einzigen Implementierung eine Darstellung und Benutzung im Notes-Client *und* im Web. Wenn eine XPage über einen Browser aufgerufen wird, beginnt der Java Server Faces (JSF[1])-Lebenszyklus [1].

Eine XPage ist ein speziell entwickeltes leistungsstarkes Notes Design-Element, für die Darstellung von Notes-Datenbanken im Web 2.0 sowie für den Notes-Client. XPages bieten die Besonderheit, dass die Darstellung im Web und im Client, sowie auch die Bedienung, identisch sind [1].

Dateiformat einer XPage

XPages und auch Custom Controls (Kapitel 3.2.5) verwenden XML als Dateiformat. Die Informationen für HTML und XML wurden sofern nicht anders angegeben, aus den Lehrveranstaltungsunterlagen zu Webengineering, von FH-Prof. Univ.-Doz. Dr. Elke Hochmüller entnommen. XML dient der Strukturierung, der Speicherung und dem Austausch von Daten.

[1] Java Server Faces (JSF) ist eine Technologie zur Entwicklung von Webanwendungen [7].

Das Wesen von XML:

- Daten können mit XML beschrieben werden.
- XML ist ein hardware-, software- und applikationsunabhängiges Datenformat.
- XML ermöglicht den Datenaustausch zwischen inkompatiblen Systemen.
- XML ist eine Auszeichnungssprache und vergleichbar mit HTML.

HTML ist die am weitesten verbreitete Strukturierungssprache. Trotz Parallelen zu XML, werden diese klar unterschieden.

Die markantesten Unterschiede sind:

- Dokumente müssen korrekt genestet (well formed) sein.
- Dokumente müssen ein schließendes *tag* besitzen.

Diese beiden Sprachen besitzen eine unterschiedliche Zielsetzung, sie stehen in keiner Konkurrenz zueinander. Mit XML werden Daten beschrieben, HTML ermöglicht die Darstellung von Daten. XML tags werden vom Anwender selbst, in der Document Type Definition (DTD) definiert.

XPages und XML Syntax

In XPages wird XML benutzt, um ein erklärendes Programmierungs-Modell zu definieren. Es wird rein die Darstellung definiert, ohne sich darum kümmern zu müssen, wie diese tatsächlich geschieht. Ziel ist es, dem Entwickler mit vorgefertigten Design-Elementen dahingehend zu unterstützen, dass mit möglichst wenig Programmieraufwand eine Applikation erstellt werden kann. Es ist nicht möglich, komplett ohne Codierung eine Applikation zu erstellen [1].

Jedoch werden vom Domino-Designer folgende Anwendungsmöglichkeiten, ohne Programmieraufwand, bereitgestellt:

- Erstellen des *user interfaces*[2] für eine Applikation.
- Definieren der Daten, welche geändert und dargestellt werden.
- Definieren der Logik, welche bei Anfragen an die Datenbank ausgeführt werden soll.

Hierfür werden tags benötigt, diese sind die Grundbausteine, um eine Anwendung darzustellen. Jeder dieser tags entspricht einem Kontrollelement für den Benutzer, einer Datenquelle, der vordefinierten Logik, oder einer Eigenschaft die von einer dieser Komponenten genutzt wird. Es ergibt sich eine klar definierte Interaktion zwischen der Benutzeroberfläche, den Daten und der logischen Komponenten.
Jede XPage ist ein XML Dokument und muss einen *root-tag* besitzen. Der root-tag einer XPage wird beim Erstellen einer neuen XPage automatisch generiert und ist im Source-View (Abbildung 3.7) ersichtlich.

[2]User interface ist die Darstellung, welche dem Benutzer zur Interaktion mit dem System bereitgestellt wird.

Abbildung 3.7 root-tag

Für weitere Informationen zu diesem Kapitel wird auf [1] verwiesen.

3.2.5 Custom Controls

Das letzte Designelement, welches in dieser Arbeit beschrieben wird, sind Custom Controls. Als Quelle dieses Kapitels diente [1]. Das grundlegende Konzept, das hinter diesem Design-Element steckt, sind Programmbausteine. Custom Controls sind die Bausteine einer XPage.

Divide and Conquer

Mit Custom Controls wird ein Divide and Conquer Prinzip verfolgt. Dieses Prinzip baut darauf, eine Datenbank-Anwendung aus kleineren Programmbausteinen zusammenzusetzen. Ein *Custom Control Block* besteht aus Daten und dem Layout. Diese wiederverwendbaren Bausteine können mittels drag and drop in die bestehenden XPages eingefügt werden. Wie auch das Design-Element XPages werden beim Entwerfen von Custom Controls, XML-tags generiert. Dies bringt den Vorteil mit sich, dass kleinere Bausteine schneller entwickelt und beliebig zusammengesetzt werden können. Der größte Vorteil von diesen Bausteinen ist ihre vielseitige Einsatzmöglichkeit. So können diese mehrmals in die selbe XPage, innerhalb der selben Anwendung beliebig oft oder in andere Anwendungen eingefügt werden. Eine Besonderheit dabei ist, dass der Geltungsbereich von eingebetteten Custom Controls in XPages, bei jedem Einfügen, *eindeutig instantiiert* wird.

Um ein Custom Control zu erstellen, gibt es wie in Kapitel 3.1.1 erwähnt, zwei Möglichkeiten. Wird ein Custom Control im Default Mode erstellt, erfolgt dies nach dem WYSIWYG Prinzip.

Verwendungsmöglichkeiten von Custom Controls

Dieses Design-Element kann auf zwei verschiedene Arten verwendet werden:

- Ein Custom Control kann als *Aggregate Container* benutzt werden. Für diese Variante wurde für dieses Design- Element entwickelt. Wie schon im Abschnitt Divide and Conquer erläutert, können die erstellten Custom Controls beliebig oft in eine Anwendung und auch in andere Datenbank- Anwendungen eingebunden werden. Die Entwicklung eines Custom Control-Elementes ist nur einmal notwendig. Danach kann das erstellte Element beliebig oft wieder verwendet werden.
- Eine andere Verwendungsmöglichkeit ist das Benutzen eines Custom Controls als *Layout Container*. Hierbei können innerhalb eines Custom Controls, Design-Elemente für das Layout verschachtelt werden. Es wird somit ein Design-Template erstellt.

Verbindet der Entwickler beide möglichen Varianten, kann eine Datenbank Anwendung mit wenig Zeitaufwand erstellt werden. Für weitere Informationen zu diesem Design-Element wird auf [1] verwiesen.

3.3 Unterstützte Programmiersprachen

3.3.1 Formelsprache

Signifikant für die Formelsprache in Lotus Notes sind @-Befehle und -Funktionen. Diese Befehle und Funktionen können für alle Bereiche, in denen Formeln zur Programmierung zur Verfügung stehen, über Referenzlisten ausgewählt werden. Der Editor unterstützt für die Formelsprache die farbliche Darstellung zur Unterscheidung der einzelnen Codesegmente. Wird ein Fehler erkannt, wird dieser Code-Teil in Rot dargestellt. Weiters wird in der Statuszeile des Editorfensters eine Beschreibung des gefundenen Fehlers abgebildet [6].
In der Version 8.5 des Designers, ist für @-Funktionen eine Bibliothek hinterlegt. Diese Funktionen der Bibliothek, sind eine Sammlung von JavaScript Methoden. Die Methoden bieten häufig in Notes benutzte Operationen, wie zum Beispiel der Rückgabewert des Autors des aktuellen Dokumentes [1].
Ein Beispiel für die Formelsprache in Notes finden sie in Folge. Eine Selektionsformel gibt an, aus welchen Dokumenten eine Darstellung, beziehungsweise, wie die Darstellung ausgeführt werden soll.
Eine in der Formelsprache implementierte Selektionsformel beginnt *immer* mit dem Schlüsselwort *select*. Mit der Selektionsformel *Select@All* werden alle Dokumente einer Datenbank in dieser Ansicht dargestellt. Da es aber nicht das Ziel ist alle Dokumente anzuzeigen, kann mit spezifischen Formeln, auch nur bestimmte Feldinhalte ausgegeben werden.
Als Bedingung wird eine Maske angegeben, wie in diesem Beispiel ersichtlich:

$$Select(form = \text{“}Beispiel\text{”}) \tag{3.1}$$

In diesem Fall werden nur Dokumente mit einem bestimmten Feldinhalt (aus der Maske Beispiel) ausgegeben. Des weiteren ist es auch möglich, mehrere Dokumente als Bedingung anzuführen.
Um dies zu erreichen, gibt man eine zusätzliche Maske wie folgt an:

$$Select(form = \text{“}Beispiel\text{”} : \text{“}NeueMaske\text{”}) \tag{3.2}$$

3.3.2 LotusScript

Die Programmiersprache mit der höchsten Integration in Lotus Notes ist LotusScript. LotusScript ist voll in die IDE integriert. Es stehen für alle LotusScript -Befehle und -Funktionen Referenzlisten zur Verfügung. Wird ein Ereignis ausgewählt, stellt Lotus Notes automatisch alle Bestandteile der Script-Routine zur Verfügung. Der Sourcecode dieser Sprache wird Events zugeordnet, deshalb ist es nicht möglich den gesamten Code, wie in anderen Editoren, einzusehen. Ziel dieser Sprache ist es, nur kleine überschaubare Einheiten des Source-Codes einzusehen, da die Wartung und das Handling vereinfacht werden soll [6].

3.3.3 JavaScript

JavaScript steht für den Zugriff einer Datenbank über einen Web Browser. Hierfür wurde speziell für JavaScript, eine Adaptierung des W3C Document Object Model erstellt. Wie in Abschnitt 3.3.1 bereits erwähnt, ermöglichen XPages dem Entwickler die Benutzung von Java- Script. Unter Verwendung dieser Programmiersprache, kann eine eigene Logik in eine Anwendung implementiert werden [1].
Um diese Logik einzubinden gibt es die Möglichkeit einer *serverseitigen-* oder *clientseitigen-*Programmierung. Abhängig von der gewählten *Seite*, von welcher das Script aufgerufen wird, gibt es Unterschiede:

- **Object Model:** Dieses Model wird für die Darstellung der XPage benutzt.
- **Global objects:** Eingebettete Objekte, welche referenziert werden können.
- **System libraries:** Bibliothek mit Klassen, welche benutzt werden können.

Serverseitige Programmierung

XPages bieten eigene Object Models um JavaScript zu verwenden. Diese Object Models sind eine Kombination aus JSF Object models, Document Object models und einigen Objekten welche die Entwicklung vereinfachen sollen. Das Document Object Model ist eine Schnittstelle, mit welcher auf eine Bibliothek zugegriffen werden kann. Diese Bibliothek enthält eine Sammlung von Klassen und Methoden, welche dafür benutzt werden können, um ein XML Dokument zu erstellen oder zu verändern [1].

Mit serverseitiger Programmierung können:

- Design-Elemente manipuliert werden,
- Informationen zu Anfragen ausgelesen werden,
- Interaktionen mit dem laufenden Prozess durchgeführt werden,
- Auslesen von Informationen zum aktuellen Status der Anwendung.

Auf die *global objects* und *System Libraries* kann über die Property Sheets (3.1.4), im Events Tab zugegriffen werden.

Clientseitige Programmierung

Die Entwicklung von clientseitigem JavaScript unter XPages, ist dieselbe wie die einer Webanwendung.

Es müssen dabei folgende Faktoren beachtet werden:

- Control ID versus Client ID,
- Einbinden der Server Daten im clientseitigen JavaScript,
- Hinzufügen der Server-und Clientseitigen Logik für dasselbe Ereignis.

Control ID versus Client ID: Die ID, welche in der XPage ausgezeichnet wird, entspricht nicht derselben ID, welche vom dazugehörigem Element in der HTML DOM erzeugt wird. Dies geschieht aufgrund dessen, weil die JSF Funktionseinheit eine unterschiedliche ID erzeugt, welche für die allgemeine Auszeichnung gültig ist.

Einbinden der Server Daten im clientseitigen JavaScript: Dieser Schritt muss vorgenommen werden, um das Ergebnis einer serverseitigen Berechnung, im clientseitigem Script zu verwenden.

Server-und clientseitigen Logik für dasselbe Ereignis: Es kann eine server- und clientseitige Logik, welche demselben Ereignis zugeordnet ist, hinzugefügt werden.

3.3.4 Java

Java ist eine plattformunabhängige objektorientierte Programmiersprache. Diese Sprache und das DOM sind kompatibel. Das bedeutet, dass Java auf die Dokumente von Domino zugreifen kann. Es können zum Beispiel Daten ausgelesen, oder verändert werden.

Es gibt zwei Möglichkeiten Java-Code einzubinden:

- inside Java,
- outside Java.

Wird inside Java für eine Anwendung verwendet, kann dies zum Beispiel für die Erstellung von Agenten benutzt werden. Agenten sind Programme, die eine Reihe automatisierter und zeitkritischer Aufgaben ausführen [2]. Mit Java outside hingegen, ist es möglich Anwendungen zu schreiben die auf eine Domino Datenbank beeinflussend wirken. Weiters wird Java dazu benutzt um mit serverseitigen Anwendungen, den Servlets, zu interagieren [8].

3.4 Template

In diesem Kapitel werden Konfigurationen und Vorkommnisse dokumentiert, welche im Laufe der Template-Entwicklung aufgetreten sind. Um Design-Elemente wie XPages und Custom Controls sowie die erweiterten Funktionalitäten dieser Elemente nutzen zu können, mussten erweitere Konfiguration vorgenommen werden.

3.4.1 Anforderungen

Um weitere Gestaltungs-und Funktions-Elemente zu verwenden, muss zu Beginn einer Entwicklung eine weitere Konfiguration vorgenommen werden. In dieser Konfiguration wird eine erweiterte Bibliothek namens *XPages Extension Library* eingebunden. Diese Bibliothek wird als Open-Source-Project angeboten[3] und kann kostenfrei heruntergeladen werden.
Unter Open-Source versteht sich, dass der Quellcode einer Software dem Anwender zur Verfügung gestellt wird. Es existiert eine Dokumentation, welche den korrekten Weg beschreibt um die Bibliothek einzubinden. Weiters beinhaltet diese Dokumentation eine Auflistung der verfügbaren Control Design-Elemente. Für den exakten Ablauf der Installation wird an dieser Stelle auf diese Dokumentation verwiesen.

Hier werden die durchzuführenden Schritte angeführt:

1. Bibliothek in Domino-Designer einbinden,
2. Bibliothek auf Server installieren,
3. Installieren und ausführen der *Demo-Applikation*.

Wichtiger Hinweis: Es gilt den in der Dokumentation angegeben Pfad **unbedingt** einzuhalten, ansonsten ist die Nutzung der External Library nicht möglich.

[3] http://extlib.openntf.org

3.4.2 Virtueller Server

Um sicherzustellen, dass es nicht durch etwaige Firewalls zu einer Einschränkung der Funktionalität führen kann, wurde ein virtueller Server angelegt. Dies sollte weiters dem Zweck dienen, dass die Tests der Implementierung, sollte es zu Fehlern kommen, nicht zu Lasten eines aktiven Servers gehen. Somit wurde sichergestellt, dass die Entwicklung in einer abgekapselten Umgebung durchgeführt wurde.

3.4.3 Template-Realisierung

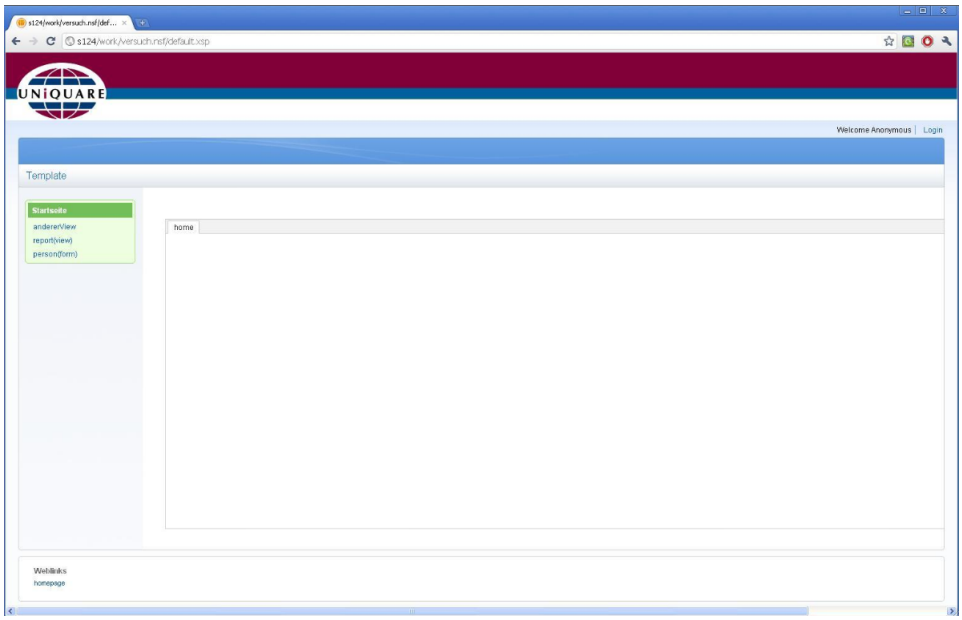

Abbildung 3.8 Template in Browser

Diese Anwendung kann von jedem berechtigten Notes-Client innerhalb der erlaubten Domäne, sowie auch über einen Webbrowser aufgerufen und ausgeführt werden. In Abbildung 3.8 ist die erstellte Anwendung in ihrer Startansicht ersichtlich.

Aufbau des Templates

Für jeden Link auf der dargestellten Seite und auch das Layout, sind XPages in Verbindung mit Custom Controls erstellt worden. Für diese Anwendung wurden vier XPages implementiert. Jede dieser XPages ist für eine unterschiedliche Verwendung als Vorlage zu sehen.

Diese XPages sind:

- **default.xsp** - Darstellung der Startseite.
- **another.xsp** - Zeigt einen mit DOJO erstellten Container mit der Darstellung einer Ansicht [14].
- **report.xsp** - Ist eine XPage mit implementierter Ansicht.
- **person.xsp** - Gibt eine editierbare Darstellung einer Maske wieder.

Um ein Auswahlmenü zu erstellen wurde eine Stilvorlage, welches als Custom Control in der Extension Library verfügbar ist, bereitgestellt und eingefügt. Diese kann der Entwickler nach Vorgaben gestalten. Das heißt das Layout wird durch ein Custom Control bestimmt und die logischen Verknüpfungen werden in einem implementierten File vorgegeben. In dieser Stilvorlage, unter Property File, werden die XPages mit den dargestellten Titeln der Links angegeben (Abbildung 3.9).

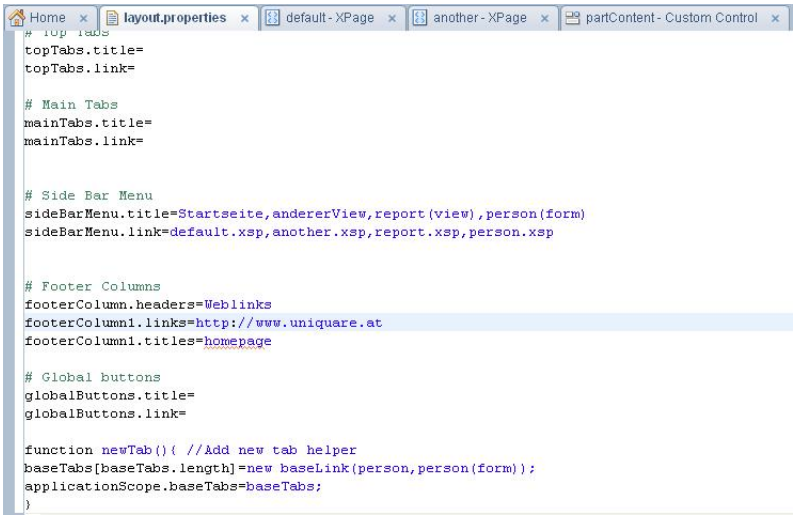

Abbildung 3.9 Property File

Jede der im Auswahlmenü verfügbaren XPages wird als Link dargestellt. Wird ein Link ausgewählt, wird die jeweilige XPage in den Browser oder Notes-Client geladen und kann bearbeitet oder gelesen werden.
Wie in Kapitel 3.1 beschrieben, wird eine XPage in einem Editor zusammengestellt. Eine mögliche Zusammensetzung wird in Abbildung 3.10 anhand der default.xsp des erstellten Templates, abgebildet.

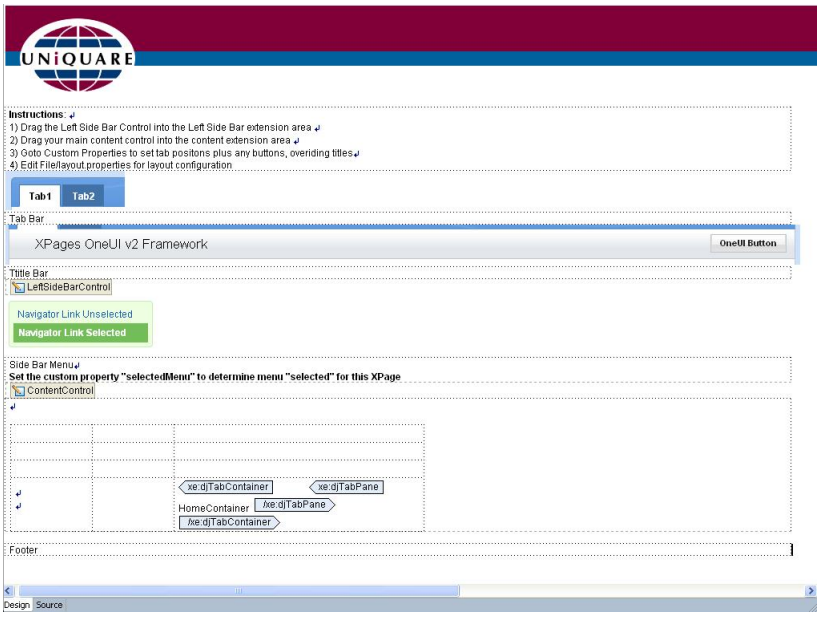

Abbildung 3.10 WYSIWYG-Editor

Unter Verwendung des Dojo Toolkits wurde bereits die Darstellung eines Containers realisiert. Das Dojo Toolkit ist eine freie JavaScript Bibliothek für die Entwicklung von Asynchronous Javascript and XML (AJAX) oder JavaScript basierenden Anwendungen. Wird innerhalb dieses Containers ein Datensatz ausgewählt, öffnet sich ein neuer Tab innerhalb des Containers. Um dies zu erreichen, muss eine serverseitige Programmierung vorgenommen werden. Die Realisierung ist in Abbildung 3.11 ersichtlich.

AJAX beschreibt eine standardisierte Vorgehensweise, um eine Reaktion einer Webseite in Echtzeit gewährleisten zu können, obwohl neue Daten vom Web-Server abgefragt werden. Anstatt einer vollständigen Webseite, wird eine Datenanfrage für einen bestimmten Teil abgerufen, und dieser in die bereits zuvor geladene Abfrage eingebaut [14].

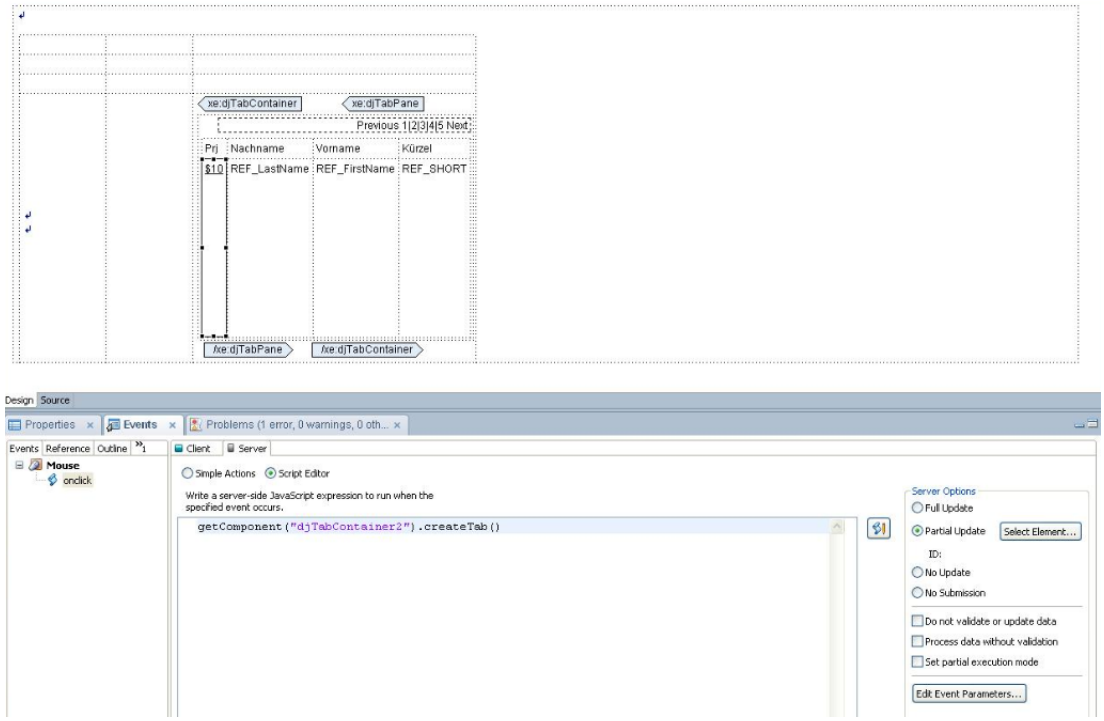

Abbildung 3.11 Dojo Tab Container

3.4.4 Fehlermeldungen

Um sich den Vorgang einer Entwicklung von Grund auf zu erleichtern, empfiehlt es sich bestehende Templates zu verwenden. Wird ein vorgefertigtes Template verwendet, können Fehler wie zum Beispiel *Unexpected runtime error* auftreten.
Es wird ein Beispiel angeführt, welches bei der Entwicklung aufgetreten ist. Zugleich wird auch der Lösungsweg beschrieben. Wird ein Template verwendet, muss beim Verwenden von diesem, auf die Verschlüsselung geachtet werden. Hierfür muss im Notes-Client, unter *properties - Encryption Settings* die lokale Verschlüsselung *deaktiviert* werden. Wird dies nicht beachtet, kann außer dem Ersteller niemand auf die Datenbank zugreifen. Ist die Verschlüsselung konfiguriert worden, muss die Datenbank auf dem Server gespeichert werden.

Abbildung 3.12 Runtime error

In Abbildung 3.12 ist eine Fehlermeldung ersichtlich. Dieser Fehler erscheint wenn, die Signaturen der Datenbank nicht übereinstimmen. Die Signatur des Erstellers und des Benutzers sind nicht dieselbe. Mit dem Ergebnis, dass der Server die Datenbank nicht im Browser darstellen kann.

Um diesen Konflikt zu vermeiden, sollten folgende Schritte eingehalten werden:

- Sollten im Designer bereits Design-Elemente geöffnet sein, müssen diese geschlossen werden.
- Ändern der Signatur im Domino Administrator-Client.

Es besteht auch die Möglichkeit, jedes Designelement einzeln zu *signen*.

Kapitel 4

Mögliche Weiterentwicklung

Erweiterungen wie AJAX, bieten dem Entwickler eine zusätzliche Unterstützung für die Darstellung und das Verhalten von Anwendungen. Mit der Verwendung von AJAX können weitere, gewünschte Darstellungen und Zugriffsmöglichkeiten bewirkt werden.

Erweiterte Einbindungsmöglichkeiten

Die Möglichkeit diese Technologien und Features zu nutzen, ist bereits durch das Einbinden der Extension Library (Kapitel 3.4.1) gegeben. AJAX bietet ein weiteres Tool, mit welchen eine Realisierung von gewünschten Verhalten ermöglicht werden kann - das *Dojo Toolkit*.
Für Informationen zu Dojo Toolkit wird auf [14] verwiesen.

Abkürzungsverzeichnis

AJAX	Asynchronous Javascript and XML
DTD	Document Type Definition
DOM	Document Object Model
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Design Environment
*IMAP*4	Internet Message Access Protocol Version 4
IP	Internet Protocol
JSF	Java Server Face
NRPC	Notes Remote Procedure Call
NSF	Notes Storage Facility
OLE	Object Linking and Embedding
PIM	Personal Information Manager
*POP*3	Post Office Protocol Version 3
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
*W*3*C*	World Wide Web Consortium
XML	Extensible Markup Language

Literaturverzeichnis

[1] Donelly, Martin, Marc Wallace und Tony McGuckin: *Mastering XPages*, Indiana/U.S. , 2011.

[2] Ebel, Nadin: *Lotus Notes Domino 8/8.5-Administration, Band 1*, München, 2010.

[3] Jendryschik, Michael: *Einführung in XHTML, CSS und Webdesign*, 2.Auflage, München, 2009.

[4] Knäpper, Matthias, Perc Primoz und Volker Perplies: *Anwendungsentwicklung unter Lotus Notes/Domino 5*, München, 2000.

[5] Lassmann, Wolfgang: *Wirtschaftsinformatik-Nachschlagewerk für Studium und Praxis*, 1.Auflage, Wiesbaden, 2006.

[6] Mann, Raimund: *Domino Designer R5*, Vaterstetten, 1999.

[7] Marinschek, Martin, Michael Kurz und Gerald Müllan: *Java Server Faces 2.0*, 2.Auflage, Heidelberg, 2010.

[8] McCoy, Cate: *Application Development*, Alameda/CA, U.S., 2001.

[9] Messerschmidt, Christian M., Sven C. Berger und Bernd Skiera: *Web 2.0 im Retail Banking*, 1.Auflage, Wiesbaden, 2010.

[10] Metzler, Susann: *Groupware Systeme*, 1.Auflage, Norderstedt, 2002.

[11] Muhs, Thomas und Boerries Klatt: *Notes/Domino 5: Einführung in die LotusScript-Programmierung*, München, 2000.

[12] Schicker, Edwin: *Datenbanken und SQL*, 3.Auflage, Stuttgart/Leipzig/Wiesbaden, 2000.

[13] Steiner, Rene: *Grundkurs relationale Datenbanken*, 7.Auflage, Wiesbaden, 2009.

[14] Steyer, Ralph: *AJAX Frameworks-RIAs mit Dojo und Co*, München, 2008.

[15] *Was ist ein tag.* Berlin: Abteilung Pädagogik und Informatik der Humboldt-Universität [Format: HTML, Zeit: 04.05.2011, Adresse: http://www.educat.hu-berlin.de/kurse/kurs20/wasisttag.html].

[16] *Was ist IMAP4.* Graz: Edis GmbH [Format: HTML, Zeit: 04.05.2011, Adresse: http://www.edis.at/was-ist-imap4_50_16.htm].

[17] *Was ist POP3.* Wien: CSOFT Allerstorfer und Radl OEG [Format: HTML, Zeit: 04.05.2011, Adresse: http://www.wiki.csoft.at/index.php/Was_ist_POP3].

[18] Wind, Martin und Detlef Kröger: *Handbuch IT in der Verwaltung*, Heidelberg, 2006.