



Nexirus GmbH
Sonnhaldenweg 66
CH-4450 SISSACH
Switzerland
Tel: +41 61 973 01 23
marcel.baumann@nexirus.ch

Architekturübersicht



Application Framework für Java™

Dieses Dokument beschreibt eine kurze Übersicht über die Architektur von ***jnex***

Author: Marcel Baumann
Version: 5.0
Date: 6/18/2009 6:28:00 PM
File: C:\marcel\projects\src\jnex\docs\ArchitecturalOverview_de.doc

1	<i>Einführung</i>	3
1.1	Wer soll dieses Dokument lesen?	3
2	<i>Wozu dient jnex?</i>	3
2.1	<i>jnex</i> und Swing™	3
2.2	<i>jnex</i> und HTML	4
2.3	<i>jnex</i> und Software Design	4
2.4	<i>jnex</i> ist nicht epidemisch	4
3	<i>Das Model View Controller Pattern</i>	5
3.1	Ziel	5
3.2	<i>jnex</i> und MVC	6
3.3	Verschachtelte Datentypen	7
3.3.1	Ereignisse (Events)	8
4	<i>Applikationen</i>	9
4.1	Pulldown Menu und Tool Bar	9
4.2	Internationalisierung	9
4.3	Dialog Fenster	10
5	<i>Persistenz</i>	11

1 Einführung

Dieses Dokument beschreibt eine Übersicht über die Klassen von jnex und deren Zusammenspiel. Das jnex Framework kann als Basis für Java Applikationsentwicklung eingesetzt werden. In erster Linie wird durch den konsequenten Einsatz von jnex die Gliederung des Programmcodes erleichtert und gefördert.

Diese Tatsache führt dazu, dass der Datenteil und dessen Logik (Business Modell) von dessen Darstellung getrennt programmiert werden kann.

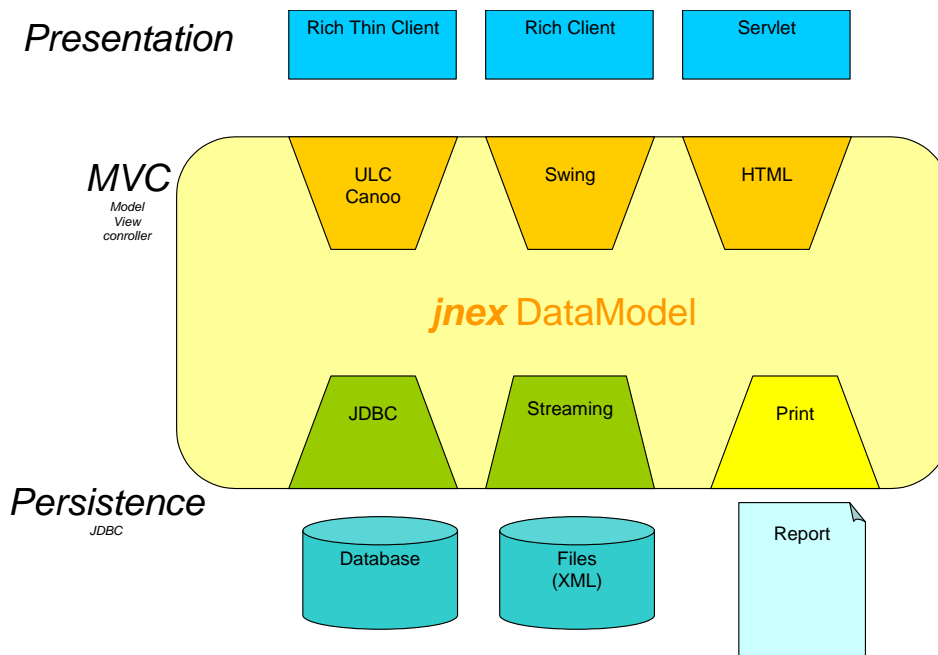
Die gleichen Datenmodelle können sogar für die Generierung verschiedenster graphischer Benutzerschnittstellen (GUI) Technologien herangezogen werden (Rich Clients und HTML Clients, Canoo ULC).

Im Gegensatz zu den meisten Model View Controller (MVC) basierenden Frameworks können die Datenmodelle von jnex beliebig hierarchisch verschachtelt werden.

1.1 Wer soll dieses Dokument lesen?

Der typische Leser dieses Dokuments will einen raschen Überblick über die Funktionalität von jnex gewinnen. Es ist gleichermassen für Entwickler als auch für Management Personal geeignet.

2 Wozu dient jnex?



2.1 jnex und Swing™

Swing™ ist Teil des von Sun Microsystems herausgegebenen Java™ Developer Kit. Es beinhaltet die Klassen, die für die Darstellung von Fenstern, Eingabefeldern, Knöpfen usw. benötigt werden.

Der Umgang mit Swing ist nicht einfach. Ausserdem hat man mit dieser Bibliothek im Rahmen eines grösseren Projekts immer wieder das Bedürfnis sehr ähnliche Elemente zu programmieren. Die jnex Bibliothek erleichtert den Umgang mit den Swing Komponenten, und verhindert die Redundanz der GUI Elemente Programmierung.

Jedes Modell kann in einen Editor verwandelt werden, der für die Bearbeitung dieses Modells geeignet ist. Dabei sorgt das Modell dafür, dass die Gültigkeitsregeln für die Datenerfassung eingehalten werden. Es muss also nicht bei jedem Eingabefeld der Wertebereich und die Integrität des eingegebenen Werts geprüft werden.

Zu dem umfangreichen Satz von vordefinierten Modellen und deren Editoren kann der jnex Programmierer neue eigene Datenmodelle und deren Editoren definieren. So wird es sehr leicht möglich, zum Beispiel ein Modell für die Kreditkartennummer zu entwerfen, das bei jeder Eingabe die Prüfziffer testet und die Erfassung von ungültigen Werten verhindert.

2.2 *jnex* und HTML

Die Datenmodelle von jnex sind in der Lage sich selbst in HTML zu verwandeln. Wenn der Programmierer HTML Templates (Muster) zur Verfügung stellt, so werden die Daten der Datenmodelle an speziell markierten Stellen dieser Templates in das HTML Dokument eingefügt. Das jnex Framework ist in der Lage ein Formular für die Datenerfassung von verschachtelten Datenmodellen zu generieren, oder einfach die Datenfelder (input) eines verschachtelten Datenmodells in ein HTML Formular Template zu integrieren.

Zu diesem Zweck werden im Template die eingefügten Muster (Funktionen) durch den Inhalt der Modelle ersetzt.

Die mitgelieferte Servlet Basisklasse ermöglicht den Aufbau einer Web Applikation. Die gesamten Dialoge der Applikation werden in einem Statusdiagramm (State Transition Diagram) formuliert. Jeder Status entspricht mehr oder weniger einer HTML Seite. Die dem Status zugeordneten Datenmodelle und das entsprechende HTML Template generieren dann die fertige HTML Seite. Durch spezielle in den Templates eingefügte Befehle werden die HTML Hyperlinks erzeugt, welche die Status Maschine mit Ereignissen steuern.

2.3 *jnex* und Software Design

jnex ändert den Fokus bei der Software Entwicklung. Der Programmierer verbringt mehr Zeit mit dem Design der Datenmodelle und deren Business Funktionalität und kann sich dann in einer zweiten Phase auf dessen Darstellung kümmern. Es ist auch möglich die Funktionalität einer Applikation zu testen, bevor das GUI perfekt gestaltet ist.

2.4 *jnex* ist nicht epidemisch

Epidemische Frameworks haben die Eigenschaft, dass Sie nicht mit anderen Programmteilen, die nicht auf der Basis des Frameworks entwickelt wurden zusammenarbeiten können. Ausserdem generieren viele Frameworks Java Code. Dies ist insofern nicht gut, da der entstehende Code nicht immer genau das macht, was eigentlich erwünscht wäre, dass aber ein generierter Code nicht auf individuelle Bedürfnisse angepasst werden sollte.

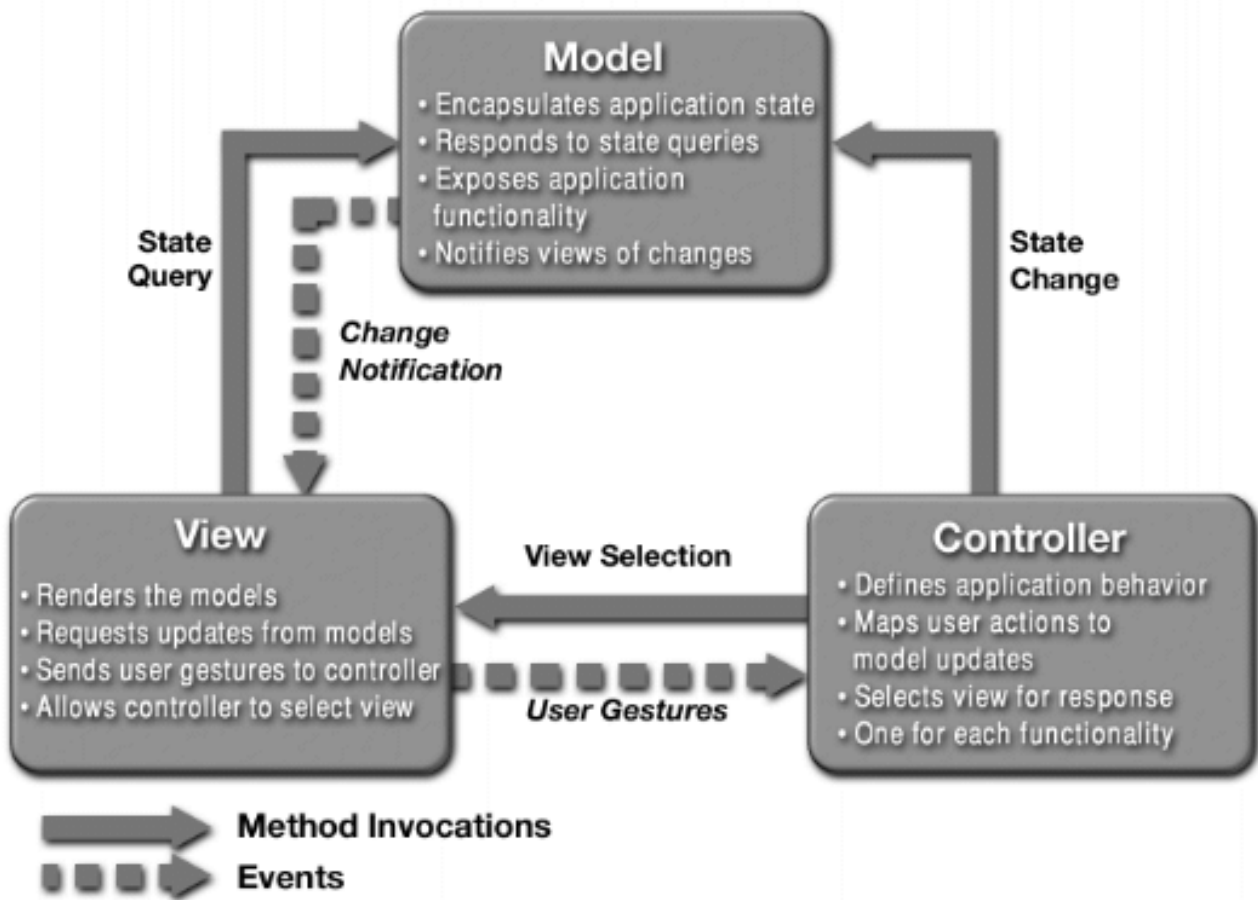
jnex kann jederzeit durch eigene neue Datentypen und graphische Elemente erweitert werden. Dies ist nicht kompliziert, da die bestehenden Modelle und Viewer im Source Code vorliegen und für die meisten Erweiterungen bereits eine geeignete Basisklasse gefunden werden kann.

3 Das Model View Controller Pattern

3.1 Ziel

Grundsätzlich besteht das Ziel, die reinen Daten programmiertechnisch von deren graphischer Darstellung zu trennen.

Zu diesem Zweck übernimmt ein Controller das Zusammenspiel der graphischen Komponenten mit den zugeordneten Datenobjekten. Die Bindung des aktuellen Zustands (Wert) des Modells und des dazugehörigen GUI Elements wird über Ereignisse (Events) gesteuert. Dadurch wird die direkte Kopplung der beiden Welten verhindert. Das Modell „weiss“ sozusagen gar nicht, dass es zur Zeit graphisch dargestellt oder gar bearbeitet wird.

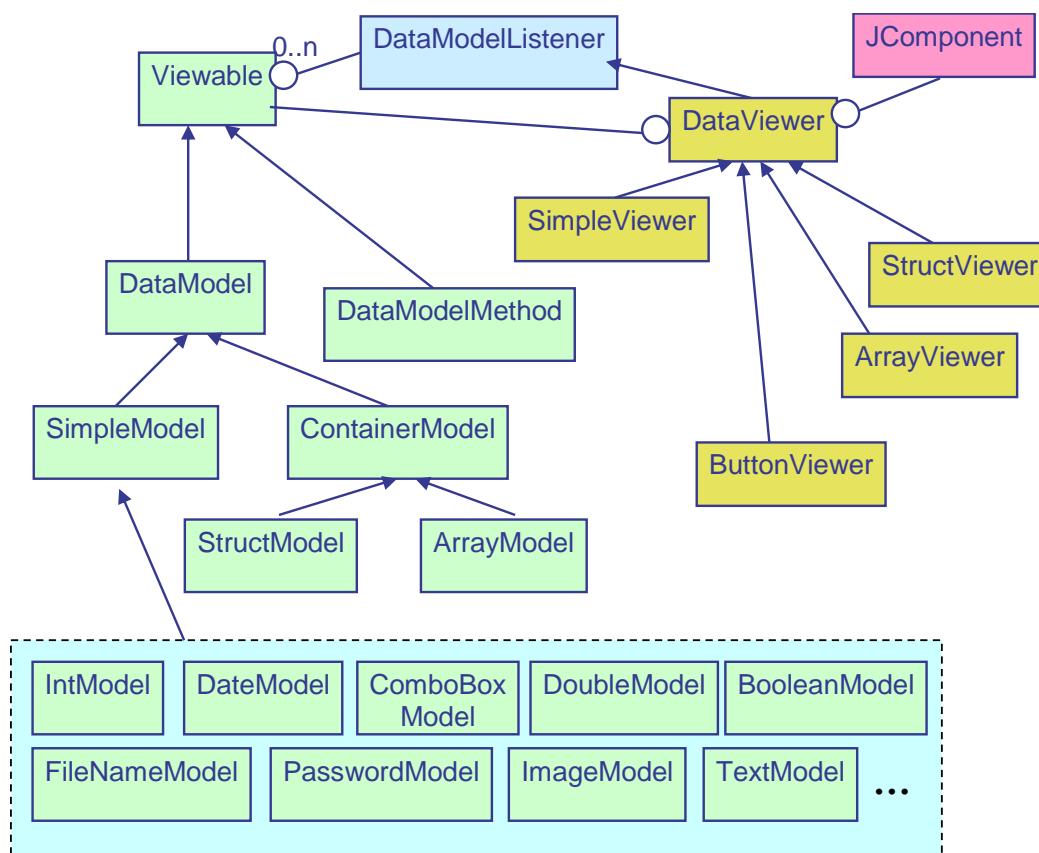


3.2 *jnex* und MVC

jnex unterscheidet einfache (SimpleModel) Datenmodelle und zusammengesetzte Datenmodelle (ContainerModel). Zu den einfachen Datenmodellen gehören Typen wie String, Integer, Boolean, Auswahlfelder (Comboboxen), Datum, Floating Point Zahlen, Währung usw.

Dann gibt es noch die zusammengesetzten Datenmodelle (ContainerModel). *jnex* unterscheidet Arrays und Strukturen. Arrays enthalten eine beliebige Anzahl Datenmodelle, die per Index adressiert werden und normalerweise vom gleichen Basistyp abstammen. Strukturen bestehen aus einer genau definierten Anzahl Elementen (Attribute), die mit einem Namen angesprochen werden, und deren Typen vordefiniert sind.

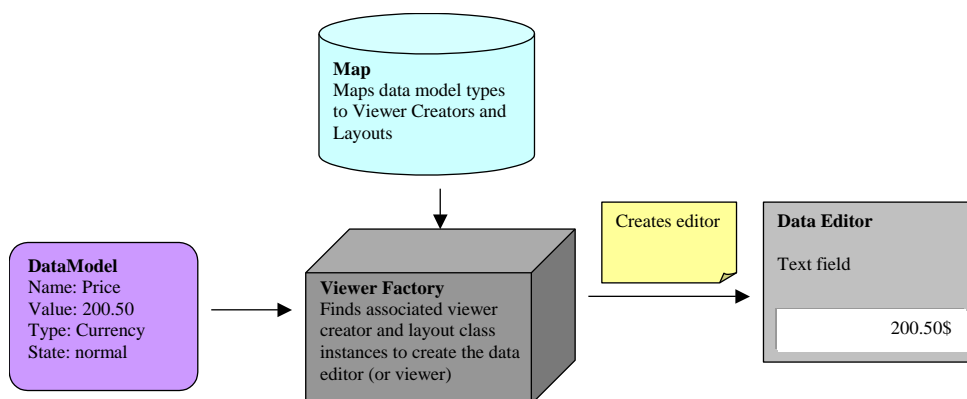
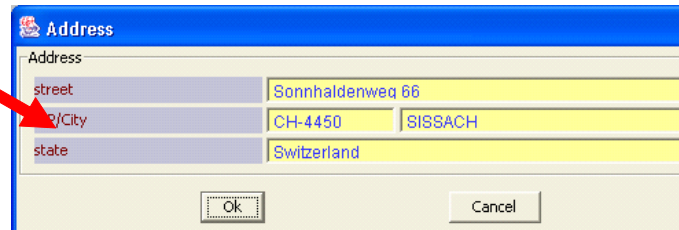
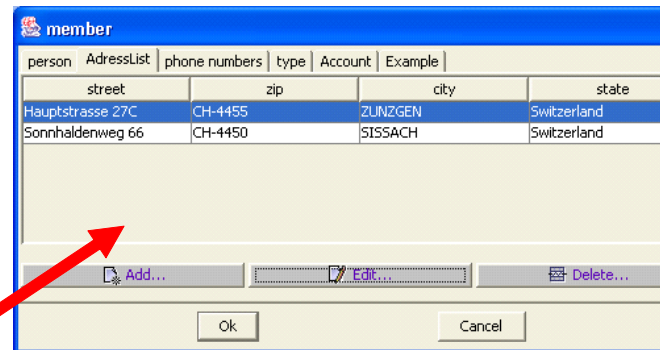
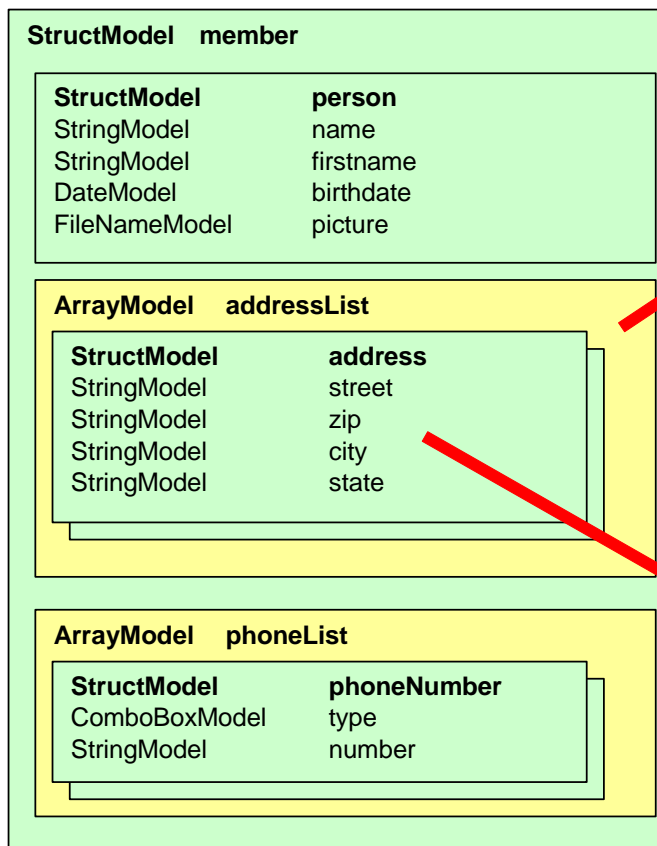
Zu jedem Datenmodell können noch typenspezifische Methoden definiert werden. Diese Methoden werden in der graphischen Darstellung normalerweise als Knöpfe (Push Button) oder als Menü Einträge dargestellt.



Type (Data Model)	Default Viewer	Default Editor
String	Label	Text field
Integer	Label	Text field
Floating point number	Label	Text field
File Name	Label	File selection dialog
Bitmap file name	Image viewer	File selection dialog
One out of a list	Label	Combo box
Boolean	Label	Check box
Array	Table	Table with New, Edit, Delete
Method		Push button
Structure	Viewer panel	Editor panel
Password	Label (displaying *)	Password editor (displays *)

3.3 Verschachtelte Datentypen

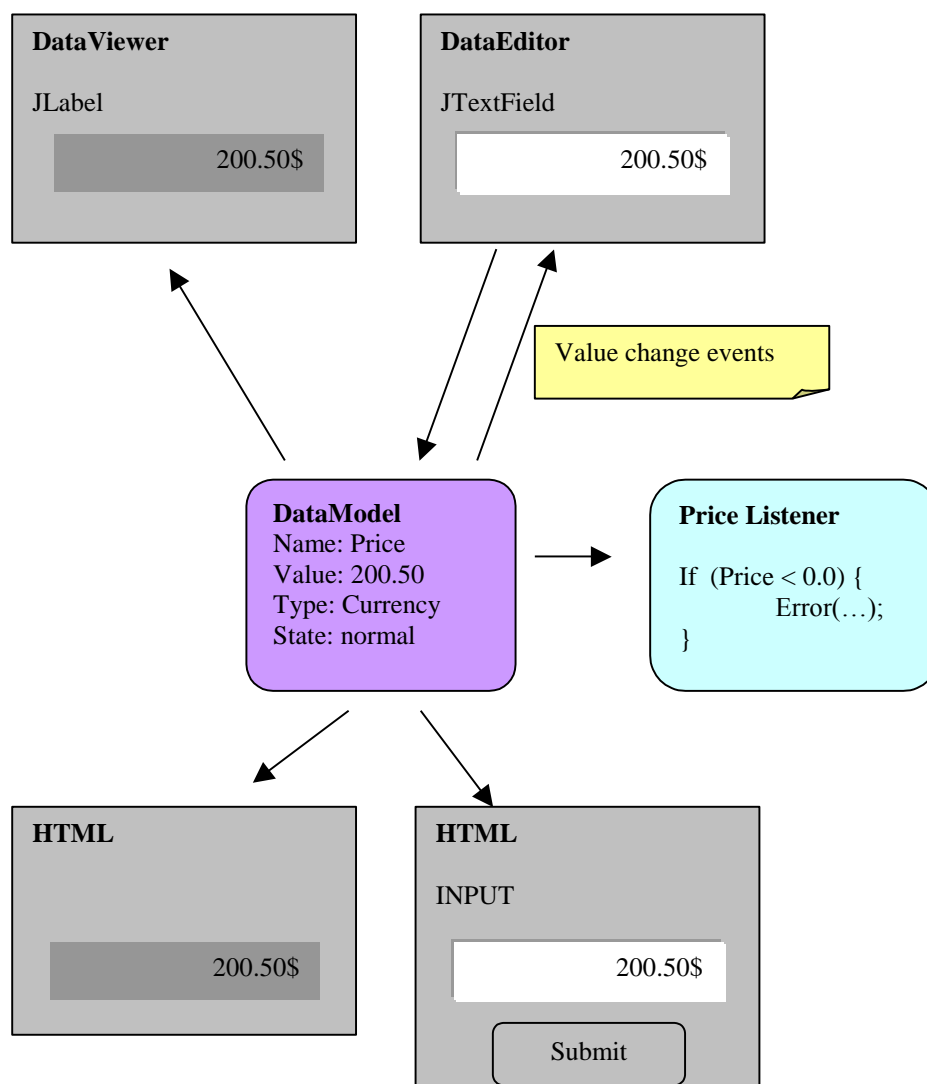
Verschachtelte Datentypen können mit der ViewerFactory in Dialoge umgewandelt werden. Falls für das entsprechende Modell ein Layout in der ViewerMap registriert ist, so wird dieses verwendet, um das Fenster zu gestalten. Das Layout bestimmt die Positionen, Grössen und die Properties (Farben Fonts, Alignment, etc.) der untergeordneten Komponenten. Jede untergeordnete Komponente wird ihrerseits auch über die ViewerFactory generiert, und übernimmt dadurch einen definierten rechteckigen Bereich im übergeordneten Layout. Dies bedeutet nicht, dass ein bestimmtes Modell immer auf die gleiche Art dargestellt wird. Das übergeordnete Layout kann auch das in der ViewerMap registrierte Standardlayout für ein bestimmtes Modell ignorieren und ein in seinem Zusammenhang passendes Layout zuordnen.



3.3.1 Ereignisse (Events)

Ereignisse steuern die wechselseitige Synchronisation zwischen den Datenmodellen und ihren zugeordneten Komponenten. Dabei können einem Modell eine beliebige Anzahl Komponenten für die Darstellung zugeordnet sein. Falls das Modell seinen Wert ändert, werden alle Interessenten mit einem Event auf die Veränderung hingewiesen. Falls eine darstellende Komponente den Wert des Modells verändert so wird automatisch vom Modell die Wertänderung an alle registrierten Komponenten weitergeleitet.

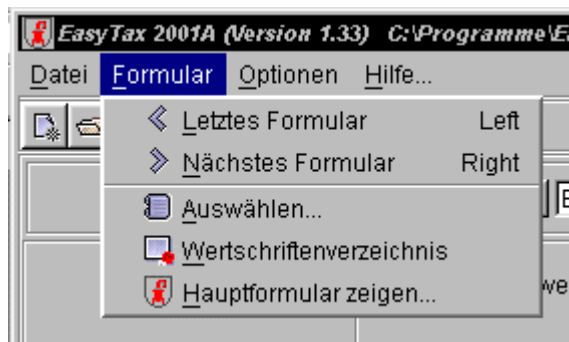
Da das Modell jede Änderung weitergibt, können sich natürlich auch andere Elemente beim Modell registrieren, die an der Wertänderung interessiert sind. Auf diese Weise können Modelle untereinander verknüpft werden, sodass ein Modell zum Beispiel immer die Summe einer Liste von anderen Modellen darstellt, und bei jeder Änderung automatisch seinen neu berechneten Wert annimmt.



4 Applikationen

4.1 Pulldown Menu und Tool Bar

Der Applikationsteil von **jnex** ist die Basis für eine schnelle Entwicklung einer Applikation. Dazu gehört das Hauptfenster mit seinem Pulldown Menu und mit dem Tool Bar (Symbolleiste), sowie das Verwalten von Dialogen (Error, Warning, Question). Der Aufbau des Menüs und der Symbolleiste geschieht auf der Basis einer sehr einfachen Text Datei. Die Internationalisierung und die Abkürzungen und Symbole werden (wie in **jnex** üblich) über die Property Files geladen.



4.2 Internationalisierung

Alle **jnex** Elemente suchen automatisch in den Property Files nach Übersetzungen und Icons für Ihre Darstellung. Da jedes Model mit einem benutzerdefinierten Namen generiert werden kann, erhält auch das zugehörige GUI Element diesen Namen und sucht in den Properties nach Übersetzung, Tool Tip, Icon, Farbe, Font, Accelerator usw. Dadurch ist eine **jnex** Applikation von Anfang an mehrsprachenfähig.

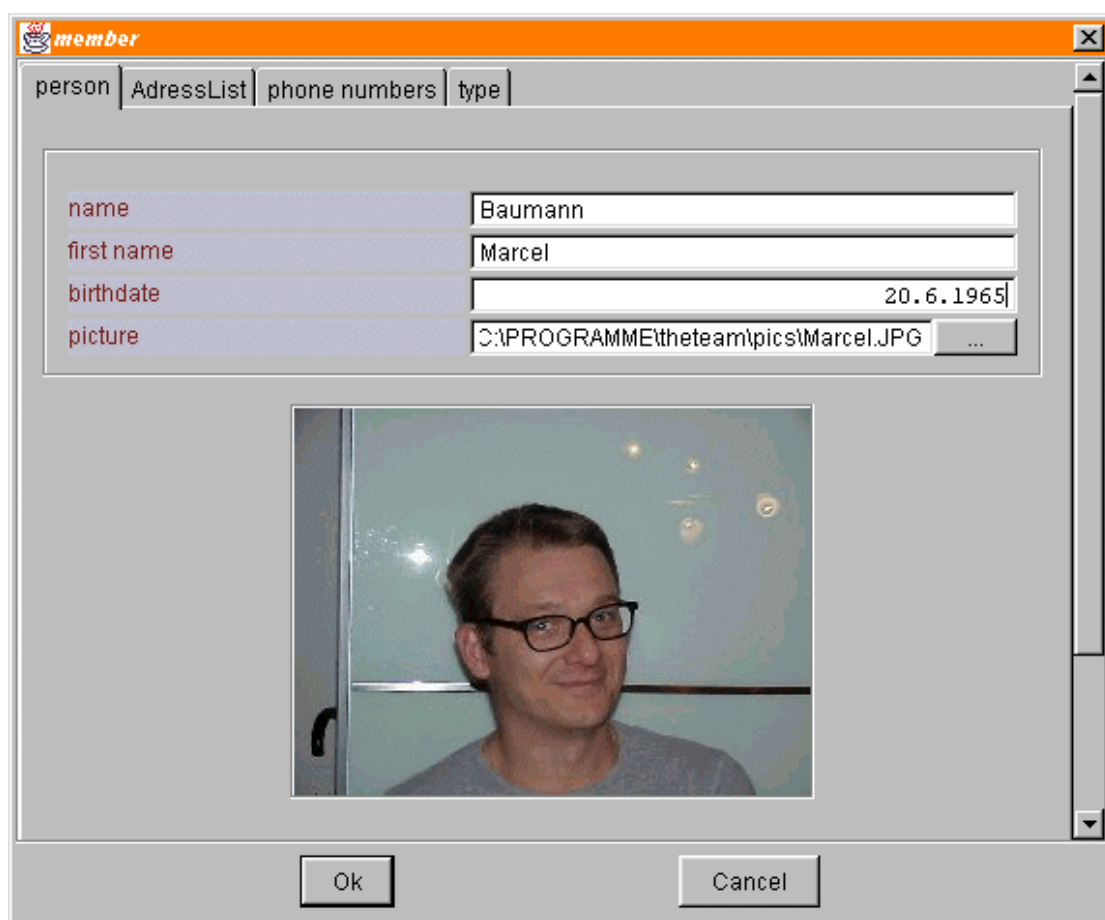
4.3 Dialog Fenster

Mit **jnex** ist sehr einfach, ein Dialogfenster für die Bearbeitung eines Datenmodells zu erzeugen. Eigentlich braucht es dazu in der Regel genau einen Funktionsaufruf (mir der Modellinstanz als Attribut).

Bei diesem Aufruf wird automatisch eine Kopie des Modells erstellt. Diese Kopie wird dann während der Bearbeitung im Dialogfenster verändert. Wenn das Fenster mit OK geschlossen wird, wird der Inhalt der Kopie auf das Originalmodell übertragen. Dabei wird für jedes Feld geprüft, ob sich dessen Wert geändert hat und falls ja, wird ein Value Changed Ereignis vom Originalmodell gefeuert.

Falls das Dialogfenster mit Abbrechen verlassen wird, wird einfach die zuvor erstellte Kopie des Datenmodells gelöscht.

Die Darstellung des Datenmodells im Dialogfenster wird wie in **jnex** üblich über das Mapping von Datenmodelltyp zum entsprechenden Editor ausgesucht.



5 Persistenz

Alle Datenmodelle können auf einfachste Weise in eine Datenbank abgelegt und gelesen werden. **Jnex** verfügt über Klassen, die grundsätzlich in der Lage sind, JDBC Befehle für create, update, delete und search zu generieren und aufzurufen. Wenn spezielle (von den defaults abweichende) Mappings für gewisse Felder benötigt werden, ist dies natürlich auch sehr einfach konfigurierbar. Es ist sogar sehr einfach ein Mapping zu definieren, welches ein (oder mehrere) Attribut(e) eines Datenmodells auf ein (oder mehrere) Felde(r) in der Datenbanktabelle abbildet.

jnex Datenmodelle können auch in Textdateien abgelegt werden. Grundsätzlich ist ein solches Datenmodell immer in der Lage seinen Inhalt in eine Zeichenkette zu verwandeln und umgekehrt.