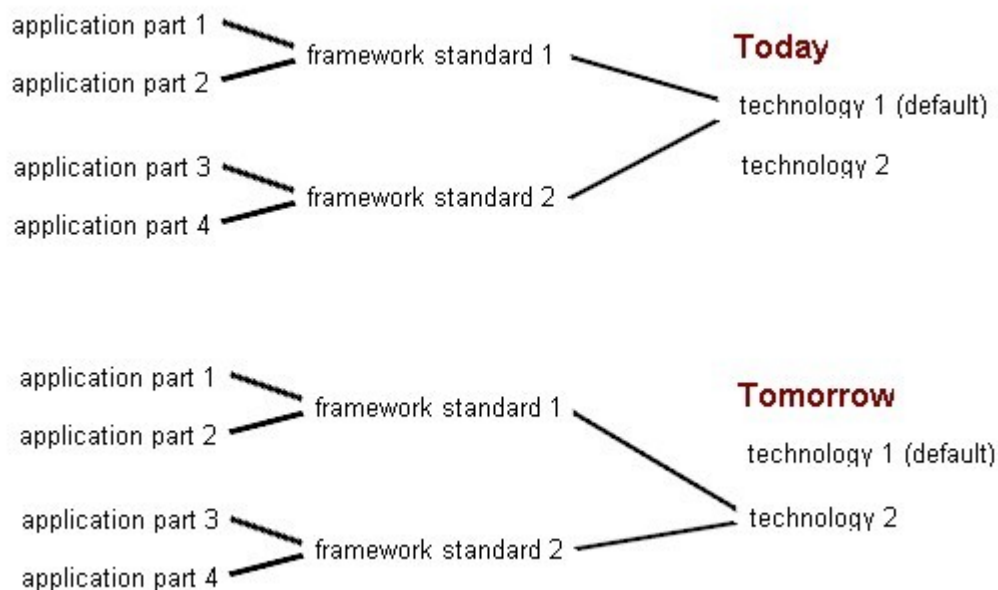


Einleitung/Einführung



Motivation

Die hauptsächliche Idee des Cameleon Open Source Projekts (Cameleon OSP) ist sich von der Technologie zu befreien. Häufig auftauchende Formulierungen sollten in der denkbar einfachsten Art gelöst werden können; der Anwendungsentwickler sollte in diesen Fällen nicht mit der verwendeten Technologie in Berührung kommen. Deswegen möchten wir, wann immer es uns gelegen erscheint, die Technologie austauschen und sie den Betriebserfordernissen anpassen.



Hauptaugenmerk

Zuallererst sei bemerkt, dass bei der Entwicklung von Datenbankanwendungen folgende Dinge diejenigen sind, die am öftesten benötigt werden, dies ist z.B. das Navigieren durch große Datenmengen, das Auswählen eines Eintrags zum Bearbeiten, zum Kopieren, etc. und zum Speichern von neuen oder geänderten Daten. Um Bestandteile zu ändern, wie sie in einer Datenbank vorkommen, ist z.B. eine Klasse namens DataPropertyAdministration vorhanden, welches die Superklasse von einigen Datensatz-Verwaltungsklassen ist.

Die Angebotspalette

Im Augenblick werden etwa 400 Cameleon OSP Komponenten angeboten, die Sie bei der Bildung von datenbanknahen Benutzerschnittstellen mittels Swing, Servlet und JSP Technologie unterstützen. Bitte lesen Sie die entsprechenden Kapitel für jede dieser Lösungen. Momentan werden die SQL-Dialekte folgender Datenbanken unterstützt: Oracle, MySQL, DB2/400 und MS Access.

Wie die Cameleon OSP Komponenten verwendet werden – ein Überblick

Voraussetzungen

Bevor sie beginnen das Cameleon OSP zu verwenden, sollten sie generell mit der Laufzeit-Konfiguration, dem Schreiben, Kompilieren und Laufen lassen von Java Anwendungen vertraut sein. Wenn Sie Server Anwendungen entwickeln wollen, empfehlen wir Ihnen sich zuerst Grundwissen in Unix/Linux und MySQL anzueignen.

Hier erhalten Sie die Quelldateien

Besuchen Sie <http://www.must.de/cameleon.html>

Konfigurationen

Da der Autor Jbuilder3 verwendet, folgen die eingeschlagenen Wege oft diesen Konventionen. Die Jbuilder Projektdateien sind Teil der Distribution. Um Cameleon OSP an andere IDEs anzupassen, kopieren Sie die Sourcedateien (Verzeichnis src) in das entsprechende Sourceverzeichnis Ihrer IDE und importieren Sie die Dateien in die betreffenden Projektdateien.

Laufenlassen der Beispielanwendungen

Client(-Server) Anwendungen

- Erzeugen Sie eine ODBC Datenquelle namens „Marketing“, basierend auf dem MS Access Treiber und generieren Sie eine (leere) Datenbank (mdb-Datei) mittels den ODBC Verwaltungsfunktionen.
- Starten Sie die Main-Klasse.

Server Anwendungen

- Erzeugen des Datenbank-Generierungs-Skripts „createDB.sql“ mittels der Klasse `de.jugs.cookbook.DbScriptCreator`; passen Sie es Ihrer Umgebung an und lassen sie dieses Skript die Datenbank „cookbook“ erzeugen (z.B. mit dem Befehl „mysql < createDB.sql“).
- Stellen Sie die Klassen oder JAR-Dateien in das betreffende Verzeichnis Ihres Applikationsservers und konfigurieren Sie diesen. Vergessen Sie nicht den JDBC-Treiber zu setzen, z.B. den MySQL-Treiber.
- Starten Sie den Applikationsserver.
- `Http://<IhrServer>:<IhrPort>/IhrPfadZurHauptklasse` – Beispiel:
<http://192.168.5.16:8080/cookbook/servlet/de.jugs.cookbook.Main>

Der Denkansatz der hinter dem Cameleon Open Source Projekt steht

Erarbeiten Sie Ihre Anwendung auf einem abstrakten Weg und lassen Sie das Framework die konkrete technische Lösung realisieren.

Beispiel:

Wenn der Anwender darüber informiert werden soll, dass ein Datensatz nicht gelöscht werden kann,

könnten Sie

- ein Status Label mit dieser Information füllen, indem Sie die Methode `setText()` verwenden
- mit dieser Information einen Dialog öffnen, oder ein modales Fenster öffnen und einen OK Button einfügen

Anstelle dessen sollten wir jedoch die Methode `setMessageToKeep()` verwenden und das Framework entscheiden lassen, auf welche Art das zu realisieren wäre. Kann sein, dass in der Zukunft etwas anderes als „hip“ gilt.

Swing / Servlet / JSP Entscheidung

Zuerst: Egal wie sie starten, Ihre Arbeit wird niemals unbrauchbar sein, weil viele Dinge im Framework gleich gehandhabt werden. Egal, Sie müssen zuerst entscheiden welchen Weg Sie einschlagen. Wenn Sie sich entscheiden eine Server Anwendung zu erstellen: Der Vorteil einer reinen Servlet-Variante sind wenige Anweisungen, Betriebssicherheit und die große Flexibilität das Verhalten und das Erscheinungsbild (Layout) der Anwendung nachträglich zu ändern. Der Vorteil von Java Server Pages (JSP) sind die Individualität und Kreativität.

Aufsetzen Ihres eigenen Projekts

Was Sie grundsätzlich wissen sollten

Bezüglich Datenbank

Die Datenbankverbindung wird spezifiziert in der Java-Klasse `de.must.dataobj.ConnectionSpecification`, diese Klasse stellt die Verbindung mittels der Methode `getConnection()` zur Verfügung. Es wird empfohlen die Hauptverbindung zur Datenbank global für alle Anwendungen in der Subklasse von `de.must.middle.GlobalStd()` bereit zu stellen.

Tabellen werden in Objekten beschrieben, welche die Klasse `de.must.dataobj.DataObject` erweitern. Wenn Ihre Datenbank bereits existiert, müssen Sie nicht mehr tun, als die Tabellennamen zu definieren. Falls nicht, wird empfohlen, dass Sie Ihre Tabellen als eine Subklasse von `DataObject` definieren und erzeugen daraus Erstellungsskripte oder die Datenbank selbst, außerhalb dieser Beschreibungen durch Subklassen der Klasse `de.must.dataobj.TableCreatorStd` (siehe Beispiel).

Der primäre Schlüssel wird verwaltet durch `DoIdent`, ein `de.must.dataobj.DataObject`. Sie brauchen dieses Objekt und die damit zusammenhängende Tabelle „Identity“ um Datensätze in die Datenbank aufzunehmen.

Betreffend Daten-Veränderung

Das Konzept um sich durch große Mengen an Daten zu bewegen ist:

Bestimmen eines Datensatzes / mehrerer Datensätze, welche zu ändern wären, durch eine Abfrage mit einem Kriterium oder mehreren, um das Abfrageresultat (Suchergebnis) bedeutend zu reduzieren.

Durchlaufen des Suchergebnisses und wechseln in den Änderungsmodus durch einen Doppelklick auf einen einzelnen Datensatz.

Ändern des Datensatzes und klicken auf OK.

- Für Swing Benutzerschnittstellen siehe: `de.must.wuic.SimpleDataListFrame` und `de.must.wuic.DataPropertyAdministration`.

- Erkunden Sie entsprechenden Beispielanwendungen.

Quellen

Icon Quellen

Wir fanden eine kleine Icon Sammlung bei

<http://developer.java.sun.com/developer/techDocs/hi/repository>. Die Firma Sun erlaubt es diese weiter zu verteilen.

Weiterführende Informationen

Informationen im Internet

Schauen Sie unter <http://www.must.de/cameleon.html> nach der aktuellen Dokumentation.

API Dokumentation

Erkunden Sie die API Dokumentation. Um das Framework und seinen Gebrauch zu verstehen empfehlen wir:

- Erkunden Sie die Funktionen der Beispielanwendungen, indem Sie diese laufen lassen und vergleichen Sie diese mit Ihren Anforderungen.
- Erkunden Sie den betreffenden Quelltext (Sourcecode).
- Verwenden Sie die API Dokumentation für das genauere Verständnis des Frameworks und für die Verwendung anderer Framework-Funktionen, wie sie in den Beispielanwendungen zum Einsatz kommen. In der API Dokumentation sind die Beispiellassen unter der Rubrik „Direct Known Subclasses“ aufgeführt. Wie auch immer, es hilft mehr den Sourcecode der Beispielanwendungen zu erkunden anstelle die API der Beispiele zu lesen.

FAQ

Die erwartungsgemäß ersten Fragen die Ihnen kommen, werden (hoffentlich) in dem Dokument FAQ.html beantwortet. Sie sind jedoch darüber hinaus eingeladen, dem Cameleon Open Source Projekt beizutreten und zusätzliche Fragen im Cameleon Hilfe-Forum zu stellen, um das FAQ zu erweitern.

Definieren von Datenstrukturen

Einführung

Um Zusatzfunktionen für bereits existierende Anwendungen zu erstellen, kann auf einer existierenden Datenbank aufgebaut werden. Wenn ein neues Projekt angefangen wird, ist empfohlen die Datenbankstruktur in der Java Syntax zu definieren und aus diesen Datenbank neutralen Anweisungen heraus eine x-beliebige Datenbank zu generieren.

Die Herangehensweise

Betrachten Sie Beispiele wie z.B. `de.jugs.cookbook.DoCookbook`. Jedes dieser `DataObject` erweiternden Klassen repräsentiert eine Tabelle in der Datenbank.

- `TableName` ist der Name der Tabelle
- `AbstractAttribute[] attributes` ist die Liste der Tabellenspalten, sie beschreibt die Spaltennamen, deren Typ und Länge.
- `Index[] indices` ist die Index-Beschreibung, um den Datenzugriff zu beschleunigen

Die Klasse `mkt.TableCreator` ist ein Beispiel wie eine Tabelle dynamisch erzeugt werden kann, wenn eine leere Datenbank gefunden wird. Sie erweitert die Klasse `de.must.dataobj.TableCreatorStd`.

Die Klasse `de.jugs.cookbook.DbScriptCreator` ist ein Beispiel dafür wie ein Skript erstellt werden kann, um einen MySQL Dialekt zu erzeugen.

Wie auch immer Sie es machen, das Prinzip ist, die Datenbank innerhalb der Anwendung zu definieren und dann beliebig auszubauen.

Attribute Typen (Feldtypen)

- `de.must.databobj.CharAttribute`
- `de.must.databobj.VarcharAttribute`
- `de.must.databobj.NumericAttribute`
- `de.must.databobj.BooleanAttribute`
- `de.must.databobj.DateAttribute`

Erkunden Sie die Klassen `Index` und `IndexItem` um zu sehen, wie Indices definiert werden.

Erstellen von Swing Anwendungen

Voraussetzungen

Bevor Sie beginnen Swing Anwendungen zu entwickeln, sollten Sie zuerst den Abschnitt „Definieren von Datenstrukturen“ gelesen haben. Swing Anwendungen verwenden diese data objects um aus Datenbanktabellen zu lesen, oder in diese hinein zu schreiben.

Einführung

Um den Zugriff auf große Mengen von Daten zu erhalten, müssen wir typischerweise den Datenbestand genauestens durchlaufen, einen Datensatz auswählen um dann diesen zu ändern. Damit Sie sehen, von was wir hier reden: Verschaffen Sie sich einen Überblick, indem Sie sich anschauen, wie die Anwendung „Marketing“ aufgebaut ist. Durchlaufen Sie die Bildschirmausdrucke mit den Verknüpfungen zum Sourcecode und der Dokumentation der Superklassen. Um den Effekt der verwendeten Befehle zu sehen, versuchen Sie die Beispielanwendung „Marketing“ (Paket mkt, Klasse Main) zum Laufen zu bringen.

Verwalten großer Datenmengen (Bildschirmausdruck)

Zum Aufruf eines Datensatzes (abfragen und selektieren eines Datensatzes) verwenden wir die Klasse `de.must.wuic.ColumnDataListFrame` oder `de.must.wuic.SimpleDataListFrame` – beispielhaft verwendet in der Klasse `mkt.FrKontaktSI`). Um einen Datensatz zu ändern, diesen zu kopieren, oder einen neuen Datensatz zu erfassen, verwenden wir die Klasse `de.must.wuic.DataPropertyAdministration` – beispielhaft verwendet in der Klasse `mkt.FrKontaktPr`.

Wichtige Benutzerschnittstellen Komponenten für die Datenerfassung sind

- `de.must.wuic.DataTextField`
- `de.must.wuic.DataDateField`
- `de.must.wuic.DataComboBox`

und weitere dieser Komponenten, welche das Interface `de.must.wuic.DataComponent` implementieren.

Verwalten kleiner Datenmengen (Bildschirmausdruck)

Manche Datenmengen haben nur wenige Reihen und Spalten. Diese können in einem Tabellenblatt bearbeitet werden. In diesem Fall verwenden Sie bitte `de.must.wuic.DataTableAdministration`, wie in der Klasse `mkt.FrGrPj` geschehen.

Menü und Toolbar (Bildschirmausdruck)

Um einem Anwender Teile Ihrer Anwendung nutzbar zu machen, sollten Sie ihm einen entsprechenden Menüpunkt und/oder eine Toolbar Schaltfläche anbieten. Menüs sind Subklassen der Klasse `de.must.wuic.MenuFrame` wie z.B. die Klasse `mkt.MainMenu`. Sie können (Zugriffs-)Berechtigungen kontrollieren, indem sie Themenbereiche und Benutzer miteinander verbinden (gruppieren). Betrachten Sie das Beispiel `mkt.Entitlement()`, welches die Klasse `de.must.middle.EntitlementStd` erweitert. Für eine flexible Zugriffskontrolle implementieren Sie

isEntitled(), Sie können den aktuellen Anwender (siehe `mkt.CurrentUser`) verwenden, um festzustellen, ob dieser berechtigt ist einen Themenbereich auszuführen. Und außerdem werden Toolbar-Definitionen in diesem Beispiel verwendet.

Swing Bildschirmausdrucke

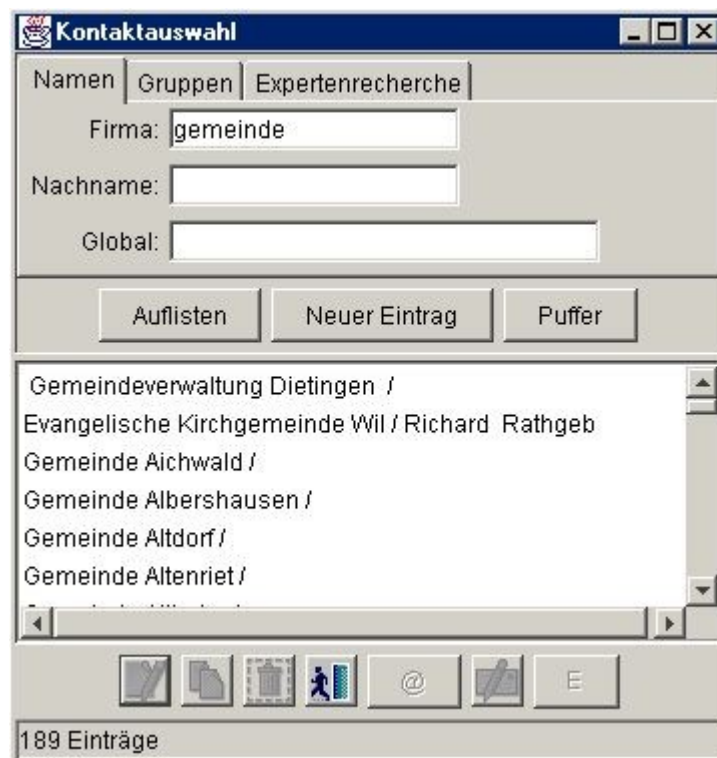
Um zu sehen, wie die am häufigsten benötigten Typen der Benutzerschnittstellen der Swing Datenbankanwendungen erstellt werden, sind diese Beispiele hier mit Bildschirmausdrucken aufgelistet. Außerdem ist der individuelle Sourcecode und die API Dokumentation der verwendeten Framework Klasse hier aufgeführt.

Menu



Erstellt durch die Klasse `mkt.MainMenu`, welche die Klasse `de.must.wuic.MenuFrame` erweitert.

SimpleDataListFrame



Erstellt durch die Klasse `mkt.FrKontaktSl`, welche die Klasse `de.must.wuic.SimpleDataListFrame` erweitert.

DataPropertyAdministration

The 'Neuer Kontakt' dialog box is a standard Java Swing window with a title bar containing a red icon and the text 'Neuer Kontakt'. It features a tabbed interface with five tabs: 'Postadresse' (selected), 'URL / e-mail / Fon / Fax', 'Sonstiges', 'Gruppen', and 'Notizen'. The 'Postadresse' tab contains several text input fields: 'Firma:' (a wide field followed by a smaller one), 'Zusatzzeile 1:', 'Zusatzzeile 2:', 'Name, Vorname:' (two side-by-side fields), 'Geschlecht / Grad:' (radio buttons for '?', 'männlich', and 'weiblich' followed by a text field), 'Strasse:', 'Land/PLZ/Ort:' (three side-by-side fields), and 'PLZ/Postfach:' (two side-by-side fields). At the bottom of the dialog are two buttons: 'Ok' and 'Abbrechen'.

Erstellt durch die Klasse `mkt.FrKontaktPr`, welche die Klasse `de.must.wuic.DataPropertyAdministration` erweitert.

DataTableAdministration

The 'Kontakt-Gruppen' dialog box is a Java Swing window with a title bar containing a red icon and the text 'Kontakt-Gruppen'. It displays a table with two columns: 'Beschreibung' and 'Pos'. The table contains the following data:

Beschreibung	Pos
Ämter	10
meine Kunden	20
meine Partner	50
Stadtverwaltungen	30
	0
	0
	0
	0
	0
	0
	0
	0
	0

The row 'Stadtverwaltungen' is currently selected. To the right of the table is a vertical scrollbar. At the bottom of the dialog are three buttons: 'Entfernen', 'Ok', and 'Abbrechen'.

Erstellt durch die Klasse `mkt.FrGrPj`, welche die Klasse `de.must.wuic.DataTableAdministration` erweitert.

Erstellen von Servlet Anwendungen

Voraussetzungen

Bevor Sie beginnen Servlet Anwendungen zu erstellen, sollten Sie zuerst den Abschnitt „Definieren von Datenstrukturen“ gelesen haben. Swing Anwendungen verwenden diese data objects um aus Datenbanktabellen zu lesen, oder in diese zu schreiben.

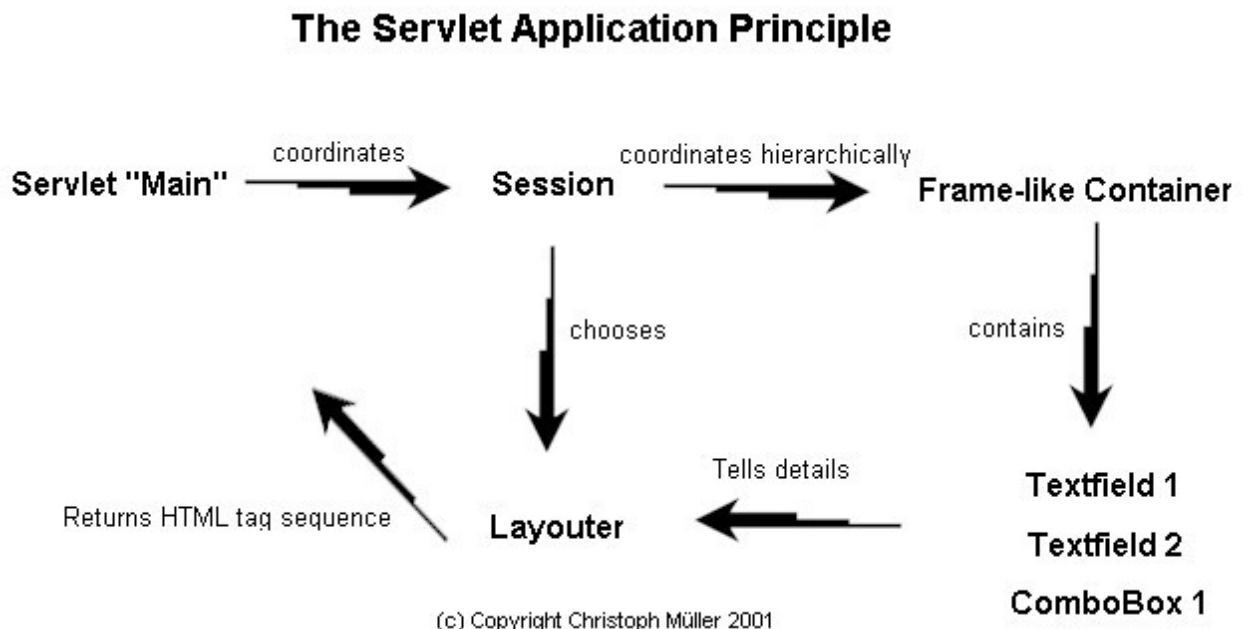
Einführung

Um den Zugriff auf große Mengen von Daten zu erhalten, müssen wir typischerweise den Datenbestand genauestens durchlaufen, einen Datensatz auswählen um dann diesen zu ändern. Damit Sie sehen, von was wir hier reden: Verschaffen Sie sich einen Überblick, indem Sie sich anschauen, wie die Anwendung „Marketing“ aufgebaut ist. Durchlaufen Sie die Bildschirmausdrucke mit den Verknüpfungen zum Sourcecode und der Dokumentation der Superklassen. Testen Sie die Live-Anwendung auf <http://www.must.de/cameleon.html>, Schaltfläche „Test Sample Application“.

Was sind „remarkable“?

Die Anwendung hat ein einziges Servlet, welches alle Anfragen (Requests) entgegennimmt und abhandelt, es erweitert die Klasse `de.must.markup.MainStd`. Es unterteilt alle Requests in so genannte Sessions (Sitzungen), Superklassen von `de.must.markup.SessionStd`. Jede Session hat einen hierarchischen Spiegel auf die offenen Sachen eines Anwenders.

Das Servlet Anwendungs-Prinzip



Legende

Element	Beschreibung	Repräsentant
Servlet Main	Zentrales Servlet für alle Requests. Es unterteilt	<code>de.jugs.cookbook.Main</code>

Element	Beschreibung	Repräsentant
	Sessions voneinander und überträgt den Request an das entsprechende Session Objekt	
Session	Abbild der Session mit seinem Dialog Stack (Stapel). Jeder Dialog wird repräsentiert durch ein so genanntes Invokable.	de.jugs.cookbook.Session
Container	Ähnlich einem umrahmten Behältnis, welches alle Eingabe- und Ausgabe Felder einschließt, wie z.B. Textfelder die mit einer Datenbank in Beziehung stehen.	de.jugs.cookbook.CookbookAdministration
Fields	Java-Objekte, deren Methoden identisch sind zu den Swing-basierten Objekten im Paket de.must.markup.	de.must.markup.DataTextField
Layouter	Offener Erzeuger der (HTML) Markup Tag Sequenz. Er sammelt Informationen von allen aktuell aktiven Komponenten einschließlich deren Status und entwirft diese. Das Layout kann während der Ausführung der Anwendung ausgetauscht werden und kann sich von Session zu Session unterscheiden.	de.must.markup.HostLayout

Das stack Kontroll Prinzip

Die grundlegende Klasse um den stack (Stapel) der Dialogschritte zu kontrollieren, ist die Session Klasse. Dies bedeutet die Verzweigung in tiefere Dialoge und fortsetzen des ersteren Dialogs nach der Rückkehr bzw. Beendigung. Dialogschritte müssen also Informationen untereinander austauschen um sich zu koordinieren. Wie wird das nun gemacht?



...

(c) Copyright Christoph Müller 2001

1. Die Klasse Session ruft den so genannten Stammdialogschritt auf (dialog step 1)
2. Der Dialogschritt 1 wertet die Benutzereingaben aus und entscheidet sich den Dialogschritt 2 auszulösen. Der Dialogschritt 1 erzeugt außerdem eine Info der Eingaben, damit diese später vom Dialogschritt 2 abgefragt werden können.
3. Die Klasse Session ruft nun Dialogschritt 2 auf, so wie das vom Dialogschritt 1 angefordert wurde und informiert diesen darüber, dass dies vom Dialogschritt 1 beantragt wurde.
4. Der Dialogschritt 2 fragt nun seinerseits die detaillierten Eingabeinformationen des Dialogschritts 1 ab.
5. Nachdem die Arbeit getan ist, fordert Dialogschritt 2 die Session auf zu Dialogschritt 1

zurückzukehren und im Stapel einen Schritt (-1) zurückzugehen.

Dieses Prinzip kann in beliebig hohen Stapeln wiederholt werden.

Entwickeln einer Server Anwendung

Voraussetzungen

Erzeugen Sie ein package ihrer Wahl.

Global erzeugen

Erzeugen einer Klasse Global, welche die Klasse de.must.middle.GlobalStd erweitert. Dies bedeutet dass alle enthaltenen Objekte für die gesamte Anwendung öffentlich sind, z.B. die Haupt Datenbankverbindung. --> Beispiel

Main erzeugen

Erzeugen Sie eine Main Klasse, welche die Klasse de.must.markup.MainStd erweitert. Die einzige Sache die Sie tun müssen ist, Informationen über das globale Objekt mittels der Methode `getGlobal()` anzubieten und auf die Klasse Session mittels der Methode `getSessionClass()` hinzuweisen. --> Beispiel

Diese Main Klasse ist das einzige Servlet, welches in der URL der gesamten Anwendung verwendet wird.

Entitlement erzeugen

Seit auf Internetanwendungen von jedermann zugegriffen werden kann, ist es wichtig, dass Berechtigungen einfach gegeneinander abgegrenzt werden können. Die zentrale Methode der Klasse Entitlement dies zu regulieren ist `getLevel(int subjectArea)`. Wie funktioniert das nun? Die Klasse Entitlement signalisiert dem nachfragenden Dialogs bzw. Prozesses was bei dem in „subjectArea“ übergebenen Sachbereich getan werden kann. Sie können nun entscheiden, ob dem Anwender dies erlaubt wird und falls ja, bei welchem Level. Zum Beispiel können Sie mittels eines solchen Levels anzeigen, dass der Anwender zwar Daten sehen kann, diese aber nicht ändern darf. Kontrollieren können Sie das, indem Sie den betreffenden Level zurück geben. Um zu beginnen, geben Sie anfangs einfach `LEVEL_ALL` zurück.

Session Klasse erzeugen

Definieren Sie „post action“ und den Titel der Anwendung mittels dem Konstruktor. In der `build()` Methode gibt es die Möglichkeit jede x-beliebige Sache zu realisieren. Zumindest jedoch sollten Sie dort die folgenden Werte setzen:

- `sessionData.setResourceBundle()`
- `sessionData.entitlement() = new Entitlement(sessionData);`
- `sessionData.menuBar = new MainMenu(sessionData);`
- `setToolBar(new ToolBar(sessionData));`
- `setLayout(new HostLayout());`

Am Schluss sollten Sie noch den Stammdialog mittels der Methode `baseInvoke()` definieren, dies kann ein Invokable oder ein MustMenuBar Objekt sein. Die Methode `build()` der Klasse Session bestimmt das Erscheinungsbild der Server Anwendung. --> Beispiel

MainMenu Klasse erzeugen

Das Erscheinungsbild der Menüs wird im Konstruktor der Klasse MainMenu definiert. Das Stammmenü (Kopf) ist definiert mittels der Methode setRootMenuDescription(). Sie können einen hierarchischen Menübaum erzeugen, indem sie die Methoden addMenu(), addSubMenu() und closeMenu() verwenden. Füllen Sie die Menüs mit Menüelementen mittels der Methode addMenuItem().

addMenu()	Erzeugt ein neues Menü auf der obersten Ebene, vorherige Untermenüs werden gleichzeitig geschlossen.
addSubMenu()	Erzeugt ein neues Menü unterhalb der aktuellen Ebene.
closeMenu()	Schließt das aktuelle Menü und erlaubt eine Menüebene höher weiterzufahren.
addMenuItem()	Fügt ein neues Menüelement dem aktuellen Menü hinzu. Die am häufigsten aufgerufenen Dialoge werden hier eingebunden.

ToolBar erzeugen

Eine ToolBar ist nicht erforderlich, jedoch nützlich. Sie ist identisch mit dem Menü hat jedoch nur eine Ebene. Wenn eine ToolBar-Option verwendet wird, unterbricht dies den laufenden Dialogschritt. Nachdem die ToolBar-Option abgearbeitet wurde, geht die Kontrolle wieder zurück an den Stammdialog. --> Beispiel

Dialoge erzeugen

Einführung

Sie sind nun soweit Ihre Anwendung zu entwickeln, indem Sie Dialoge erzeugen und diese dem Menü und der ToolBar hinzufügen. Dialoge sind so genannte invokeables, welche das Zusammenspiel mit dem Anwender und die stack-Bewegungen (aufwärts und abwärts) kontrollieren.

Verwalten großer Datenmengen

Im Gegensatz zum package wuic, gibt es hier getrennte Dialogschritte für die Abfrage und die Auflistung von Daten. Weil Internetanwendungen immer öfter den Zugriff „nur lesend“ zulassen, werden zusätzliche Komponenten zum Durchblättern verwendet. Betrachten Sie die Klasse de.must.markup.EntitlementStd und deren Subklassen de.jugs.cookbook.Entitlement um zu sehen, wie die Zugriffskontrolle hier angeboten wird.

Datenbankabfrage

Die Abfrageformulierung wird kontrolliert von der Klasse de.must.markup.Enquiry. --> (Bildschirmausdruck)

Auflistung der Daten

Das Ergebnis einer Abfrage ist ein Überblick auf die zutreffenden Einträge (Datensätze) mit der Möglichkeit diese zu durchblättern, Detaillinformationen abzurufen, den Datenänderungsmodus anzufordern, oder unter anderem Daten zu löschen. Diese Funktionalität wird angeboten von der Klasse de.must.markup.DataList. --> (Bildschirmausdruck)

Detailansicht

Falls der Anwender auf ein Element im Datenüberblick klickt, wird ihm die Detailansicht dieses Elements geboten. Diese Funktionalität wird durch die Klasse `de.must.markup.DataPropertyPresentation` realisiert. --> (Bildschirmausdruck)

Änderungsmodus

Falls berechtigt, kann der Anwender Einträge verändern, indem ihm die Subklasse der Klasse `de.must.markup.DataPropertyAdministration` angeboten wird. --> (Bildschirmausdruck)

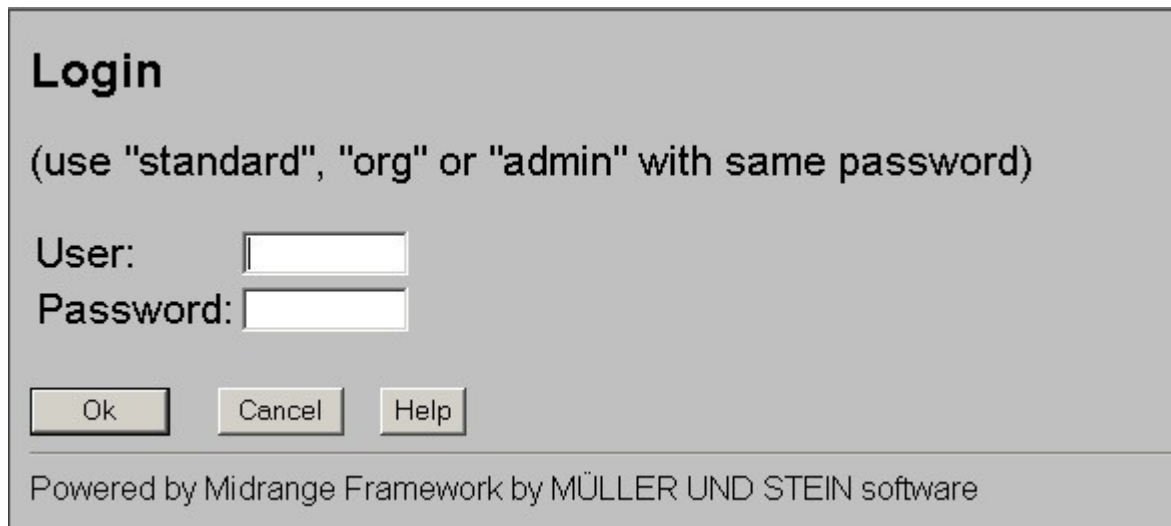
Verwalten kleiner Datenmengen

Manche Datenmengen haben nur wenige Reihen und Spalten (Felder). Diese können direkt in einem Tabellenblatt geändert werden. In diesem Falle verwenden Sie `de.must.markup.DataTableAdministration` wie das in `de.jugs.cookbook.TypeAdministration` gemacht wurde.

Servlet Bildschirmausdrucke

Um zu sehen, wie die am häufigsten benötigten Typen der Benutzerschnittstellen von Servlet Datenbankanwendungen erstellt werden, sind hier die Beispiele als Bildschirmausdrucke aufgelistet. Außerdem ist der individuelle Sourcecode und die API Dokumentation der verwendeten Framework Klasse aufgeführt.

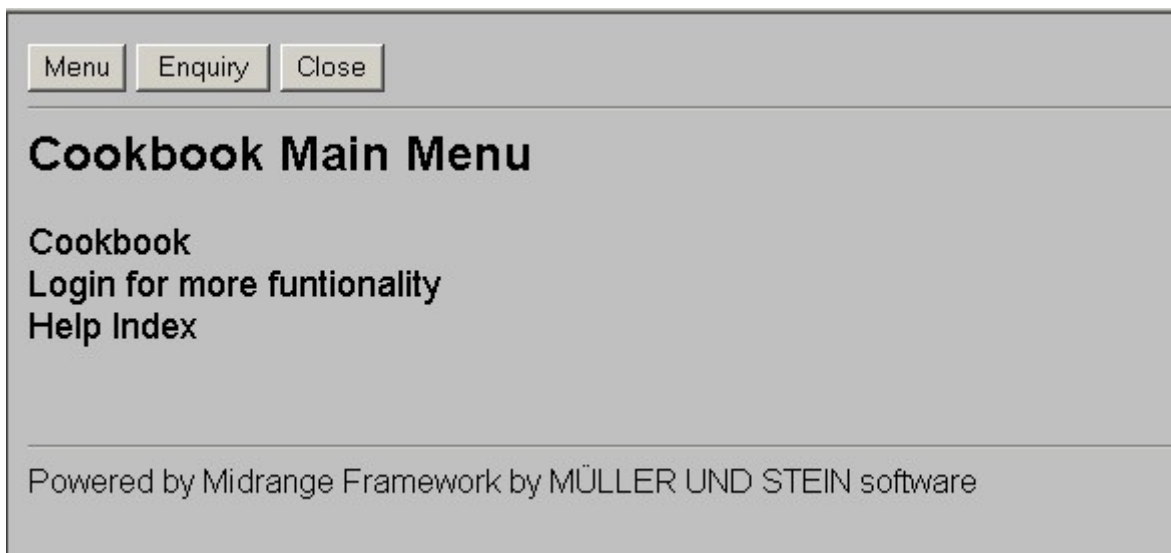
Login



The screenshot shows a Java Swing dialog box titled "Login". Below the title is a hint text: "(use 'standard', 'org' or 'admin' with same password)". There are two text input fields: "User:" and "Password:". Below the fields are three buttons: "Ok", "Cancel", and "Help". At the bottom of the dialog, it says "Powered by Midrange Framework by MÜLLER UND STEIN software".

Erstellt durch die Klasse `de.jugs.cookbook.CookBookLogin`, welche die Klasse `de.must.markup.LoginDialog` erweitert.

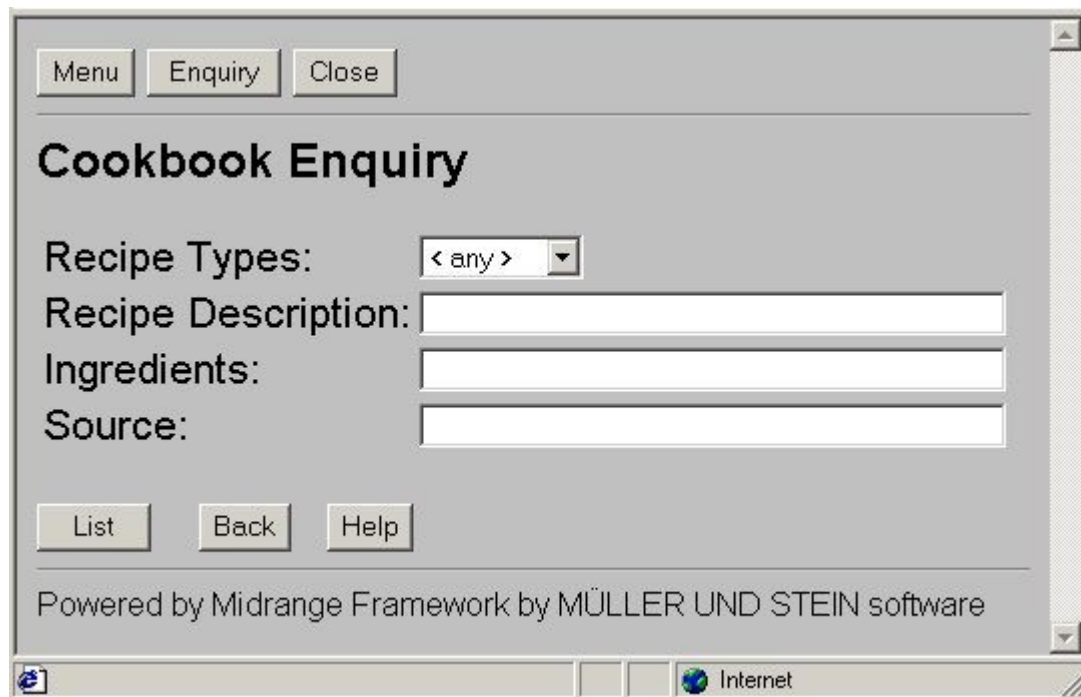
Menu (Menü)



The screenshot shows a Java Swing dialog box titled "Cookbook Main Menu". At the top, there are three buttons: "Menu", "Enquiry", and "Close". Below the buttons, the text "Cookbook" is displayed, followed by "Login for more funtionality" (note the typo) and "Help Index". At the bottom of the dialog, it says "Powered by Midrange Framework by MÜLLER UND STEIN software".

Erstellt durch die Klasse `de.jugs.cookbook.MainMenu`, welche die Klasse `de.must.markup.MustMenu` erweitert.

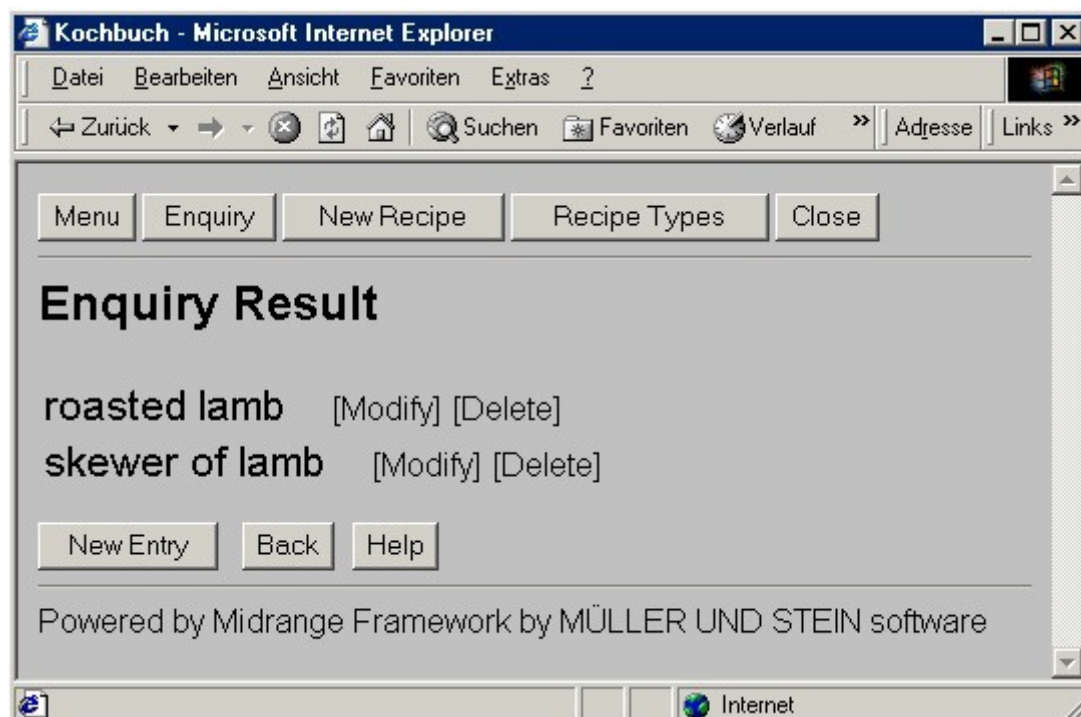
Enquiry (Abfrage)



The screenshot shows a web browser window displaying the 'Cookbook Enquiry' application. The interface includes a top navigation bar with buttons for 'Menu', 'Enquiry', and 'Close'. Below this, the title 'Cookbook Enquiry' is prominently displayed. The main form area contains four input fields: 'Recipe Types' with a dropdown menu currently set to '< any >', 'Recipe Description', 'Ingredients', and 'Source'. At the bottom of the form are three buttons: 'List', 'Back', and 'Help'. A footer message states 'Powered by Midrange Framework by MÜLLER UND STEIN software'. The browser's status bar at the bottom shows the 'Internet' icon.

Erstellt durch die Klasse `de.jugs.cookbook.CookbookEnquiry`, welche die Klasse `de.must.markup.Enquiry` erweitert.

SimpleDataList (einfache Dateiliste)



The screenshot shows a web browser window displaying the 'Enquiry Result' application. The browser's title bar reads 'Kochbuch - Microsoft Internet Explorer'. The application's top navigation bar includes buttons for 'Menu', 'Enquiry', 'New Recipe', 'Recipe Types', and 'Close'. The main content area is titled 'Enquiry Result' and lists two items: 'roasted lamb' and 'skewer of lamb'. Each item has associated '[Modify]' and '[Delete]' links. Below the list are three buttons: 'New Entry', 'Back', and 'Help'. A footer message states 'Powered by Midrange Framework by MÜLLER UND STEIN software'. The browser's status bar at the bottom shows the 'Internet' icon.

Erstellt durch die Klasse `de.jugs.cookbook.CookbookList`, welche die Klasse `de.must.markup.SimpleDataList` erweitert.

DataPropertyPresentation (Datensatz darstellen)

The 'View Entry' dialog box displays the following information:

- Menu** | **Enquiry** | **New Recipe** | **Recipe Types** | **Close**
- View Entry**
- Recipe Title:** roasted lamb
- Ingredients:** Lamb, tomatoes, pepper, potato, onion, garlic, black pepper, salt, oil, rosmmary, safron, mustard.
- Preparation:** spice the lamb with salt, pepper and mustard. then fry it in a pan. Put later all together in the oven. Serve with frensh baguette and red wine.
- Source:** Australia
- Origine (Date):** 12/12/2000
- Ok**

Erstellt durch die Klasse de.jugs.cookbook.CookbookPresentation, welche die Klasse de.must.markup.DataPropertyPresentation erweitert.

DataPropertyAdministration (Datensatz bearbeiten)

The 'Modify Entry' dialog box contains the following fields and controls:

- Modify Entry**
- Recipe Title:**
- Recipe Type:** ☐ **Diabetic recommendation**
- Ingredients:**

Lamb, tomatoes, pepper, potato, onion, garlic, black pepper, salt, oil, rosmmary, safron, mustard.
- Preparation:**

spice the lamb with salt, pepper and mustard then fry it in a pan. Put later all together in the oven. Serve with frensh baguette and red wine.
- Source:**
- Origine (Date):**
- Ok** | **Cancel** | **Help**

Erstellt durch die Klasse de.jugs.cookbook.CookbookAdministration, welche die Klasse de.must.markup.DataPropertyAdministration erweitert.

DataTableAdministration (Datensatz bearbeiten in der Tabellenansicht)

Description	Pos	
Soup	100	[Delete]
Starter	400	[Delete]
Salad	600	[Delete]
Dessert	910	[Delete]
Delicious	950	[Delete]
Others	999	[Delete]
	0	
	0	
	0	
	0	
	0	

Erstellt durch die Klasse de.jugs.cookbook.TypeAdministration, welche die Klasse de.must.markup.DataTableAdministration erweitert.

Erstellen von Java Server Pages Anwendungen

Vorkehrungen

Bevor Sie anfangen das JSP Toolkit zu verwenden, sollten sie den Abschnitt „Definieren von Datenstrukturen“ lesen. Das Toolkit verwendet diese „data objects“ um aus Datenbanken zu lesen oder in sie hinein zu schreiben.

Einführung

Kapselung

Es gibt eine Menge Dinge in den JSPs, welche von Java Objekten eingekapselt werden können. Die meisten der INPUT (Dateneingabe) Elemente sind geeignet, um sie in entsprechende Objekte zu spiegeln. Wir machen das analog den Objekten der Swing Komponente Jcomponent. Wir fassen diese unter dem Namen „markupables“ zusammen, was bedeutet, dass diese sich selbst in einer Markupsprache (z.B. HTML) abbilden können, indem die Methode getCreationTag() implementiert wird. Betrachten Sie hierzu de.must.markup.Markupable.

Bezüglich Persistenz kann man durch Kapselung der Logik gerade hier viel Gewinn erzielen. Um die ermüdende Programmierung von Fehler-abfangenden getters und setters zu umgehen, können Sie mit der Datenbank verbundene Markup Komponenten verwenden. Betrachten Sie die Klasse de.must.markup.Storable. Komponenten, welche dieses Interface implementieren sind im Augenblick der Klassenkonstruktion und dem Auslösen des Primärschlüssels beim Laden der Klasse, allein durch die Nennung der Tabelle und des Spaltennames (Feldname) mit dem entsprechenden Datenbankfeld verbunden.

Um das mehrfache Durchlaufen von Laden- und Speichern-Methoden auf mehrere Komponenten des Frameworks zu übertragen, wird die Gruppierung unterstützt. Betrachten Sie hierzu de.must.markup.GroupOfStorables und de.must.markup.GroupOfMarkupables.

Das Butler Prinzip

JSP sollte Beans verwenden, wir haben das bereits gesagt. In Ordnung, lassen Sie uns diesen einen Namen geben. Wir könnten Sie Server nennen, aber dieser Ausdruck ist bereits in Gebrauch. Was ist mit „butler“? Ein Butler bedient seinen Herrn, und das ist es auch, was diese Bean für seine JSP tun sollte: Gebt ihm die Sorglosigkeit sich nur um das Aussehen (Design) kümmern zu müssen.

Jede JSP hat eine butler Klasse. In der JSP wird sie einfach nur „butler“ gerufen. Somit müssen wir dieses Objekt nie umbenennen, falls wir identischen Code für gleichartige Seiten verwenden wollen. Diese butler Klassen sind verknüpft mittels jspSession.getButler(). Warum nicht mittels useBean? Cameleon OSP bietet Ihnen die Kontrolle über alle butler in ihrer Session um diese in einer frei wählbaren Sequenz zu erzeugen und wieder zu zerstören. Zum Beispiel ist es vorteilhaft zuerst eine butler Klasse zu erzeugen, diesem Anfangswerte zu geben wie zum Beispiel eine Abfragekondition (where condition), und dann die JSP anzurufen den existierenden butler zu verwenden.

Beispiel

JSP

```
<%@ include file="header.inc" %>
<%@ page import = "de.jugs.cookbook.jsp.CbpropButler" %>
```

```

<% CbpropButler butler = (CbpropButler)jspSession.getButler(CbpropButler.class); %>
<%@ include file="afterbf.inc" %>
<html>
<head>
    <title>Rezept-Eingabe</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <meta name="Author" content="Christoph Müller">
    <%@ include file="headtags.inc" %>
</head>
<body>
<h2>
Rezept-Eingabe</h2>
<table CELLPACING=0 CELLPADDING=4 COLS=1 WIDTH="100%" >
<tr>
<td class="subtitle">Das geht alles dazu</td>
</tr>
</table>
<p>
<form method="POST">
<table COLS=2 WIDTH="100%" >
<tr>
<td>Rezept-Bezeichnung:</td><td><%=butler.rezeptBez%></td>
</tr><tr>
<td>Rezept-Typ: &nbsp;  </td><td><%=butler.typ%> <%=butler.diabetiker%></td>
</tr><tr>
<td>Zutaten:</td><td><%=butler.zutaten%></td>
</tr><tr>
<td>Zubereitung:</td><td><%=butler.zubereit%></td>
</tr><tr>
<td>Quelle:</td><td><%=butler.quelle%></td>
</tr><tr>
<td>Vom:</td><td><%=butler.vom%></td>
</tr>
</table>
<p><%=butler.okButton%> <%=butler.cancelButton%>
</form>
</body>
</html>

```

Die verknüpfte (butler) Klasse:

```

/*
 * Public Domain Sample Code
 */
package de.jugs.cookbook.jsp;
import de.jugs.cookbook.DoCookbook;
import de.jugs.cookbook.DoType;
import de.must.markup.*;
import javax.servlet.http.HttpServletRequest;
/**
 * Devoted butler of the cookbook property administration JSP.
 * See super class JspButler for documentation.
 * @author Christoph Mueller
 */
public class CbpropButler extends JspButler {
    private DoCookbook doCookbook;
    private GroupOfStorables storables;
    public DataTextField rezeptBez;
    public DataComboBox typ;
    public DataCheckBox diabetiker;
    public DataTextArea zutaten;
    public DataTextArea zubereit;
    public DataTextField quelle;
    public DataDateField vom;
    public OkButton okButton;
    public CancelButton cancelButton;
    private int counter;
    public CbpropButler(SessionData sessionData) {
        super(sessionData);
        doCookbook = new DoCookbook();
        storables = new GroupOfStorables(doCookbook, sessionData);
        rezeptBez = storables.createTextField("RezeptBez");
        typ = storables.createComboBox(sessionData, "Typ", new DoType(), "TypBez");
        diabetiker = storables.createCheckBox("Diabetiker-Empfehlung", "Diabetiker");
        zutaten = storables.createTextArea("Zutaten");
        zubereit = storables.createTextArea("Zubereit");
        quelle = storables.createTextField("Quelle");
        vom = storables.createDateField("Vom");
    }
}

```

```

        okButton = storables.createOkButton(sessionData);
        cancelButton = storables.createCancelButton(sessionData);
    }
    protected void handle(HttpServletRequest request) {
        storables.fetchValuesFromRequest(request);
        if (okButton.wasPressed()) {
            if (storables.inputCheckOk() && isInputAccepted()) {
                storables.save();
                goBack(); return;
            }
        } else if (cancelButton.wasPressed()) {
            goBack(); return;
        } else {
            String idString = request.getParameter("ID");
            if (idString != null) {
                storables.loadValues(Integer.parseInt(idString));
            } else {
                storables.newInput();
            }
        }
    }
    private boolean isInputAccepted() {
        if (rezeptBez.getText().length() == 0) {
            setMessageToKeep("Rezeptbezeichnung muss gefüllt sein.");
            return false;
        }
        return true;
    }
}

```

Erläuterungen

Das Beispiel zeigt wie man Textfelder, Textbereiche, Combo-Boxen und Check-Boxen erzeugen kann, diese ihre Werte aus einer Datenbanktabelle beziehen lassen kann, diese dann präsentiert und, falls von einem Anwender geändert, das Weggelassen dieser Daten unterstützt.

Um die Datenbankbindung zu unterstützen, konfigurieren Sie bitte die Global Klasse, welche eine Subklasse von `de.must.middle.GlobalStd` ist und die Spezifikation der Datenbankverbindung enthält.

Übersetzung aus dem Englischen ins Deutsche durch Gunter Koch im Juni 2008